A Computational Model of Derivational Morphology

Dissertation
zur Erlangung des akademischen Grades
des Dr. rer. nat.
an der Universität Hamburg

eingereicht von
Bruce Mayo
Konstanz

den 4. Oktober 1999

Genehmigt vom Fachbereich Informatik der Universtät Hamburg
auf Antrag von Prof. Dr. Christopher Habel (Betreuer),
Prof. Dr. Manfred Kudlek (Gutachter),
Prof. Dr. Christoph Schwarze (externer Gutachter)

Tag der Disputation: 26. Mai, 2000

Prof. Dr. Leonie Dreschler-Fischer
Dekanin des Fachbereichs Informatik

Zusammenfassung

1. Überblick

Die vorliegende Arbeit präsentiert ein computerlinguistisches Modell der Verarbeitung komplexer Wörter im Satzkontext. Sie ist an der Universität Konstanz in dem von Ch. Schwarze geleiteten Forschungsprojekt „Morphologie der Derivation“ entstanden, gefördert von der Deutschen Forschungsgemeinschaft. Ziel der Arbeit ist es, ein rechner-implementierbares Modell der Speicherung und der Verarbeitung von Wörtern sowohl im Kompetenz-System der Sprache als auch im Performanz-System der Erkennung und Produktion zu erstellen. Das Modell ist im theoretischen Rahmen der Lexikalisch-Funktionalen Grammatik (LFG) [Bre82b] angesiedelt und benutzt teilweise deren formalen Apparat [KB95]. Die Implementierung erfolgte in Prolog II+ [Pro96] und wird im folgenden „Konstanzer LFG-Umgebung“ genannt, kurz KLU. Das zugrundeliegende kognitive Spezifikations-Modell wird in dieser Arbeit als KLU–Modell bezeichnet.

Das KLU–Modell bietet formale Lösungen für drei wichtige Probleme der Wortverarbeitung, die in der LFG und in ähnlichen Theorien keine adäquate Berücksichtigung gefunden haben. Erstens: es zeigt, dass eine stark lexikalisch orientierte Theorie wie die LFG so erweitert werden kann, dass spontane Wortschöpfungen in die Analyse des Satzes integriert werden, und zwar auf eine Weise, welche die großen Produktivitätsunterschiede verschiedener Wortbildungsmuster berücksichtigt, ohne das Prinzip der Lexikalischen Integrität [BM95] zu verletzen. Zweitens: es zeigt, dass die sogenannten Klammerungs-Paradoxien, die bei manchen derivierten Wörtern beobachtet werden, aus einem zweistufigen Segmentierungsverfahren entstehen, in welchem die Segmentierung des Wortes getrennt von der morphologischen Analyse erfolgt. Drittens: es zeigt, dass die scheinbar widersprüchlichen psycholinguistischen Daten zur Erkennung von Pseudo-Präfixen sich aus derselben Trennung von Segmentierung und morphologischer Analyse erklären lassen.

Gleichzeitig stellen das KLU–Modell und seine Implementierung einen Versuch dar, die Bildung kognitiver Modelle an allgemein anerkannten Prinzipien des Software-Engineerings auszurichten, und exemplarisch unter Rückgriff auf diese Prinzipien die Modellierungsprobleme der Wortderivation zu lösen. Die dem Modell zugrundeliegende, breit gefaßte Anforderungsanalyse stellt Bedingungen, die unter Beibehaltung der in der Computerlinguistik üblichen Trennung zwischen Syntax und Wortanalyse nicht erfüllbar sind. Während nämlich die meisten monomorphemischen Wörter ohne weiteres unmittelbar in die Struktur eines Satzes eingefügt werden können, sprechen viele linguistische und experimentelle Daten dafür, dass flektierte Wörter zuerst segmentiert und dann nach morphologischen Regeln analysiert werden,

bevor sie der Satzgrammatik übergeben werden. Bei nicht-lexikalisierten derivierten Wörtern findet ein ähnlicher Vorgang statt, der aber wesentlich komplexer sein kann. Dieser Schritt führt in manchen Fällen zunächst zu morphologischen Merkmalsstrukturen, die manchmal die Komplexität eines Satzes erreichen können, aber im Sinne einer Unifikationsgrammtik [Shi88] mit der Struktur ihres Matrixsatzes nicht unmittelbar unifizierbar sind.

Während die Analyse eines flektierten Wortes im Satzkontext ohne Aufwand wiederholt werden kann, erfordert die Wiederholung einer Wortderivation manchmal einen viel höheren Berechnungsaufwand. Um diesen Aufwand zu vermeiden, ist es wahrscheinlich, dass die lexikalische Einfügung (engl. „lexical insertion“) einen Zwischenspeicher oder „cache“ benutzt, um die Ergebnisse einer einmal durchgeführten Derivation festzuhalten und für die Satzanalyse bereitzustellen. Obwohl formal redundant, bietet die Annahme eines solchen Zwischenspeichers eine einfache Erklärung für viele rätselhaften Daten, die über Derivationsmorphologie vorliegen. Um Zugriffe auf den Zwischenspeicher zu verwalten, muss die Schnittstelle zwischen Wortbildung und Satzanalyse mit einer prozedurale Kontrollstruktur ausgestattet sein. Die Zwänge, die aus dieser Kontrollstruktur entstehen, stellen die weitverbreitete Ansicht in Frage, dass prozedurale Elemente aus formalen Grammatikmodellen verbannt bleiben sollen.

2. Formale Ansätze und Software-Engineering

Sicherlich kann die Entwicklung von Formalismen, die durch Unifikation von Termen, Graphen und Gleichungen sprachliche Strukturen ohne prozedurale Mittel erfassen können, als bisher wichtigster Beitrag der Computerlinguistik zur Modellierung natürlicher Sprachen gelten. Mit diesen Techniken wurde es möglich, die Strukturen und Bedeutungen von sprachlichen Ausdrücken allein aus der Kombinatorik ihrer Elemente in implementierbaren Formeln zu beschreiben, ohne auf die Reihenfolge oder auf die Präzedenzrelationen der zugrundeliegenden kombinatorischen Operationen achten zu müssen. Der große Vorteil solcher Ansätze liegt darin, dass die rein deklarativ gedachten Aussagen einer strukturalistisch gewonnenen Sprachbeschreibung sich beinahe eins-zu-eins in Aussagen des Beschreibungsformalismus umsetzen lassen, und dass man jede dieser Aussagen für sich auf die sprachlichen Daten zurückführen und verifizieren kann. Die Korrektheit einer unifikationsbasierten Grammatik hängt also nicht von komplexen Interaktionen zwischen einzelnen Aussagen der formalen Beschreibung ab. Mit einigen wenigen Ausnahmen haben deklarative, unifikationsbasierte Formalismen sich daher in der computerlinguistischen Forschung nicht nur als Werkzeuge, sondern auch als Sicht- und Denkweise durchgesetzt.

Jedoch haben Versuche, rein deklarativ basierte Formalismen für die Beschreibung der Wortderivation einzusetzen, zu keinen vollständig befriedigenden Ergebnissen geführt (z. B., [KJ94], [KuMvBF93], [Kun95]). Die Probleme, die sich dabei ergeben haben, könnten natürlich in der ungeahnten Komplexität der Derivation liegen. Die Kernaussage des vorliegenden Modellierungsversuchs ist hingegen, dass das Verharren der computerlinguistischen Forschung auf rein deklarativ formulierten Modellierungssprachen die Sicht zu einer Lösung hin versperrt hat, die zumindest in Umrissen schon in den Anfängen der LFG in Bresnan & Kaplan [BK82] enthalten war. Bresnan & Kaplan haben gegen die transformationelle Analyse von Passivformen mit

der Begründung argumentiert, dass die Komplexität des erforderlichen Regelwerks für eine Passiv-Transformation in Echtzeit (d. h., während der Satzanalyse) einen unvertretbaren Berechnungsaufwand nach sich ziehen würde. Sie haben die Möglichkeit in Betracht gezogen, dass eine Transformation diesen Aufwand reduzieren könnte, wenn sie das Ergebnis der Transformation sofort ins Lexikon schreiben und als Lexikoneintrag für spätere Zugriffe bereithalten würde. Somit wurden die Tranformationen weitgehend ins Lexikon verlagert, wo sie als syntax-unabhägige, lexikalische Regeln weiterlebten. Der Gedanke, dass die Theorie weiterhin erstmalige Analysen komplexer Wörter (etwa von noch nie gesehenen Passivbildungen) beschreiben können sollte, wurde in der LFG-Literatur kaum mehr verfolgt.

Das KLU–Modell nimmt diesen Gedanken wieder auf. Es versucht, die unifikationsbasierte, lexikalische Sichtweise der LFG und ähnlicher Theorien so auszubauen, dass sie auch mit Daten zur produktiven Wortbildung so weit wie möglich kompatibel bleibt. Gleichzeitig war das Projekt von Prinzipien und Erfahrungen des Software-Engineerings geleitet, die sich für die Entwicklung großer Anwendungsprogramme bewährt haben. Dazu gehörte die Überlegung, dass jeder Beschreibungsformalismus, der für eine Problemlösung eingesetzt wird, sich sowohl einer logischen als auch einer Kontrollkomponente bedient [Kow79]; beide Komponenten sind stets in der Lösung implizit, auch wenn sie nicht explizit notiert sind. Wenn die Kontrollkomponente völlig außer Acht gelassen wird, können wesentliche Aspekte einer vorgeschlagenen Lösung leicht aus dem Blick verloren gehen. Aus einer Vielzahl von Untersuchungen zum Software-Engineering ist bekannt, dass eine zu frühe Fixierung auf eine bestimmte formale Darstellung für die Entwicklung eines Anwendungsprogramms hinderlich ist [Boe86], und dass die unterschiedliche Realisierung von Anforderungen durch die logische Spezifikation und durch die Kontrollkomponenten eines Programms von Beginn an sorgfältig überlegt sein muss. Anhand der logischen Spezifikation kann man am leichtesten die formale Korrektheit einer Implementierung nachweisen, doch bürgt Korrektheit noch lange nicht für die Validität des Programms, also für die Eigenschaft, seine tatsächlichen Anforderungen zu erfüllen [Blu94]. Die Erfüllung vieler Anforderungen kann von verdeckten prozeduralen Aspekten abhängen, die in der logischen Spezifikation nicht explizit zum Vorschein kommen.

Die für das KLU–Modell vorgeschlagene Lösung geht aus diesem Grunde nicht ausschließlich aus der logischen Problemanalyse hervor, die man aus strukturalistisch orientierten linguistischen Untersuchungen gewinnen kann. Die logisch korrekte Formulierung einer Problemstellung läßt oft mehrere Implementierungswege offen, und die Wahl der Speicherungs- und Kontrollstrukturen kann unter Umständen ungeahnte Auswirkungen auf das Verhalten des Programms haben. Um alle Implementierungsmöglichkeiten gegeneinander abzuwägen, ist es daher oft von Vorteil, möglichst viele ‚Interessenvertreter' („stakeholders" in der englischsprachigen Literatur) heranzuziehen und ihre Anforderungen an die Lösung zu berücksichtigen. Für die Entwicklung des KLU–Modells wurden deshalb Ergebnisse von drei ‚Interessenvertretern' bzw. aus drei Forschungsdisziplinen herangezogen, die ein Modell der Derivationsprozesse betreffen könnten, und zwar die lexikalische Statistik, die Wahrnehmungspsychologie, und die linguistische Syntax und Semantik. Erst nach der Auswertung dieser Anforderung wurden Beschreibungsformalismen gesucht, welche die Daten verständlich darstellen können. Schließlich mussten die einzelnen Komponenten integriert werden. Das Ergebnis dieser Arbeit zeigt, dass wesentliche

Daten zur Derivationsmorphologie eine prozedurale Spezifikation der lexikalischen Einfügung erforderlich machen.

3. Formale Probleme der Derivationsmorphologie

Ähnlich wie für Sätze gilt in vielen Fällen auch für komplexe Wörter, dass ihre syntaktischen und semantischen Eigenschaften sich durch Unifikation von Konstituentenmerkmalen gut rekonstruieren lassen. Damit ist aber noch nicht gesagt, ob und wie die zwei Systeme – Satzgrammatik und Wortbildung – interagieren. Vor allem ist nicht gesagt, ob sie als Gesamtsystem aktiv an der Generierung und Analyse von Sätzen beteiligt sind. Computerlinguistische Ansätze sind bislang auf eine weitgehende Trennung der Systeme ausgerichtet. In den letzten Jahren ist aber mehrfach gezeigt worden, dass Sprecher – so wie sie immer neue, noch nie gesagte Sätze produzieren – immer wieder neue Wörter in ihre Rede einführen [Baa92]; davon sind viele Wörter durchaus verständlich, aber so unüblich, dass sie von Lexikographen gar nicht registriert werden [BR6b]. Die Produktion von neuen Wörtern unterliegt aber anderen statistischen Gesetzen als die Produktion von Sätzen: Ein bestimmter Satz wird selten zweimal geäußert, aber Wörter werden gerade daran als solche erkannt, dass sie im Diskurs immer wieder, und zwar mit einer reproduzierbaren Häufigkeit, erscheinen [Orl82]. Die spontan gebildeten Wörter sind manchmal form- aber nicht bedeutungsgleich mit bekannten Wörtern, und sie weichen von den festen, lexikalisch gespeicherten Bedeutungen in unterschiedlichem Maße ab.

Um das Problem näher zu charakterisieren, sei ein Beispiel aus [Sti96, 143] angeführt, stellvertretend für weitere, die in den Kapiteln 4 und 5 diskutiert werden.

(65, S. 131) Max wird seinen Mitgliedsbeitrag für den Alpenverein abwandern.

Das Verb *abwandern* wird oft mit der Bedeutung ‚entlang laufen' verwendet. Aber in der oben exemplifizierten Verwendung kann es nur heißen ‚Geld durch Wandern verbrauchen', eine Bedeutung, die im Wörterbuch nicht stehen wird. Durch eine Wortgrammatik läßt sich die unübliche Bedeutung von (65) ableiten, wie B. Stiebels [Sti96, 143] gezeigt hat. Dabei bleiben aber einige wesentliche Aspekte der Beziehung zwischen dem komplexen Wort und dem einbettenden Satz rätselhaft:

- Ein ambiges, komplexes Wort wie *abwandern* kann in bestimmten Kontexten zu einer ähnlichen Ambiguität des Satzes führen (vgl. ‚Max wird sein Erbe (Grundstück oder Geldsumme) abwandern'); dies zeigt, dass die lexikalisierte und die spontan gebildete Lesart gleichzeitig verfügbar sind.

- Ein frei deriviertes Wort kann Argumente verlangen, die weder im lexikalisierten Homonym noch in der Basis vorkommen, wie das Akkusativobjekt von *abwandern* in der Bedeutung ‚Geld durch Wandern verbrauchen'.

- Viele der frei nach einem bekannten Wortbildungsmuster gebildeten Wörter werden von Sprechern trotzdem auch im Satzkontext als ungrammatisch oder nicht interpretierbar bezeichnet.

- Die Bedeutung eines häufig benutzten, komplexen Wortes weicht mit der Zeit

oft zunehmend von der Bedeutung ab, die aus seinen Konstituenten rekonstruiert werden kann.

Diese Probleme lassen vermuten, dass die Einfügung eines derivierten Wortes in einen Satz auf unterschiedliche Weise geschehen kann, und dass eine Darstellung der lexikalischen Einfügung durch Unifikation der Wortkonstituenten mit Konstituenten des Satzes zu einfach wäre, um diese Beobachtungen zu erklären.

4. Die Problemanalyse

Entsprechend dem im Software-Engineering üblichen Vorgehen der Anforderungs-Analyse („Requirements Analysis"), beginnt die Problemanalyse für das KLU-Modell mit einem Streifzug durch drei benachbarte Forschungsdisziplinen, die wichtige Ergebnisse zur Wortbildung erbracht haben. Aus diesen Ergebnissen sollen Anforderungen an das Modell abgeleitet werden, nach dem Vorbild der Erstellung eines Lastenhefts für ein Entwicklungsvorhaben. Durch eine verbesserte Kenntnis der zugrundeliegenden Einfügungslogik, aber auch der damit verbundenen Kontrollelemente, können, so ist zu erwarten, einige Widersprüche aufgehoben werden, die viele rein deklarative, linguistisch gewonnene Hypothesen über Derivation durchziehen.

Anforderungen der lexikalischen Statistik

Statistische Untersuchungen großer Textcorpora haben Daten über die Häufigkeiten von Wörtern in Texten und über das Wachstum des Wortschatzes als eine Funktion der Corpusgröße (die Wachstumskurve) identifiziert. Diese Daten zeigen, dass die Wörter im Corpus eine ungleichmäßige Häufigkeitsverteilung aufweisen, in der kurze Wörter, wie Konjunktionen, Präpositionen und Verbalauxiliare, häufig vorkommen, während einzelne längere Wörter, wie Verben und Nomina, relativ selten anzutreffen sind. Diese Verteilung wird durch die Zipf-Mandelbrot Gleichung beschrieben [Orl82]. Andere Analysen haben versucht, die Größe des Vokabulars in einem Corpus als Funktion der Corpusgröße zu charakterisieren. Solche Verfahren haben gezeigt, dass die Größe des Wortschatzes eines unendlichen Corpus potentiell unbegrenzt ist, da selbst bei Textmengen von 80 Millionen Wörtern die Wachstumskurve des Wortschatzes keine feststellbare Obergrenze erreicht [BR6b]. Die klassischen Studien zur lexikalischen Statistik setzten ein unendlich großes, aber statisches Lexikon voraus. Neuere Untersuchungen [Baa92] haben aber klar gemacht, dass der Wortschatz eines Corpus in vielen Wortkategorien durchaus begrenzt ist, und dass seine Wachstumskurve sich aus einer Untermenge der beschreibbaren Wortformen ergibt. Unter der Annahme eines statischen, aber sehr großen Lexikons läßt sich dieses Phänomen durch Überlegungen der Kodierungstheorie wie folgt erklären: Die häufig vorkommenden, kurzen Wörter (Funktionswörter) bilden eine geschlossene Klasse ähnlich zu den Selektionspräfixen eines Huffman-Codes, während die viel größeren, offenen Wortklassen (z. B. Verben, Substantive) die weniger häufigen, aber informationsreicheren Symbole einer Sprache darstellen. Weil die Wörter der offenen Klassen oft in unterschiedlichen Verwendungen benutzt werden (Tempus, Numerus), tragen die Selektionspräfixe der Sprache zu einer optimalen Informationsdichte bei. Der Vergleich mit dem Huffman-Code legt die Vermutung nahe, dass die Wörter der zwei Klassen unterschiedliche Darstellungs-

formen im Lexikon haben. Da die Selektionspräfixe relativ wenig Information tragen, dürften sie mit einfachen Datenstrukturen beschreibar sein, während die Inhaltswörter mit ihrem hohen Informationswert größere Datenstrukturen verlangen.

Statistische Untersuchungen großer Textcorpora zeigen, dass das Vokabular bestimmter Wortbildungsmuster praktisch unbegrenzt ist. Wörter dieser Klassen müssen also durch einen der Satzgrammatik ähnlich produktiven Apparat entstehen, aber die Ergiebigkeit der verschiedenen Muster variiert in hohem Maße. Weil sie stets von einem bestimmten Corpus abhängt, ist die statistisch ermittelte Produktivität aber lediglich eine Erscheinung der linguistischen Performanz, und sie kann nicht unmittelbar auf das zugrundeliegende generative System zurückgeführt werden. Um einen Produktivitätsparameter in das generative System der Sprache, d.h. in die Kompetenzbeschreibung einzuführen, bedarf es eines corpusunabhängigen, aber empirisch bestimmbaren Maßes der Produktivität. Die Ableitung einer solchen corpusunabhängigen Definition der Wortbildungsproduktivität aus einer Charakterisierung der Wachstumskurve von V. M. Kalinin [Orl82, 156] wird skizziert. Damit kann die typische Streuung der Produktivität bei Wörtern als inhärenter Teil der Wortdarstellung im Kompetenz-System der Sprache gesehen werden, und nicht als eine Nebenwirkung der Performanz.

Anforderungen der Psycholinguistik

Als zweites Gebiet stellt die psycholinguistische Forschung Anforderungen an ein Modell der Wortbildung. Mit welcher Geschwindigkeit ein Wort erkannt wird (die Erkennungslatenz), ist eine logarithmische Funktion seiner corpusstatistischen Häufigkeit. Die Erkennungslatenz wird typischerweise mit einem sogenannten lexikalischen Entscheidungsverfahren als die Zeit bestimmt, die der Proband im Experiment für die Entscheidung benötigt, ob eine gesehene oder gehörte Zeichen- oder Lautfolge ein Wort seiner Sprache ist. Experimente zeigen, dass die Erkennungslatenz für ein flektiertes Wort meistens von der Frequenz (relative Häufigkeit in großen Corpora) des Stamms abhängt, aber nicht von der Frequenz der Oberflächen- (flektierten) Form [Taf79b]. Allerdings zeigen einige Daten zum Englischen auch die umgekehrte Relation: Bei sehr häufig vorkommenden, flektierten Wörtern korreliert die Erkennungszeit mit der Häufigkeit der Oberflächenform. Diese *Dualität der lexikalischen Repräsentation* ist besonders deutlich für flektierte Wörter des Italienischen gezeigt [CLR88] worden. Interessanterweise tritt der erste, stammbasierte Effekt auch bei Wortpräfixen auf, die keine Morpheme sind. Diese bedeutungslosen Pseudoaffixe (wie *re-* im englischen *reproach*) werden während der lexikalischen Entscheidung demnach getrennt erkannt. In anderen Experimenten, die z.B. auf assoziativen Effekten beruhen („cross-modal priming“), scheint es hingegen, als ob das Pseudoaffix nicht getrennt von seinem Stamm erkannt wird. Da die Assoziationseffekte aus einer tieferen Ebene der Wortverarbeitung stammen müssen, legt dieser Unterschied die Annahme nahe, dass die Pseudopräfixe als Zugriffsschlüssel auf lexikalisch vorhandene Einträge benutzt werden, dass sie aber keine Rolle in der morphologischen Analyse eines zerlegbaren Wortes spielen.

Für Wörter, die ein Derivationsaffix enthalten, sind die Zerlegungseffekte weniger stark. Zumindest eine Studie zeigt jedoch, dass die Erkennungslatenz in bezug auf den Stamm mit der Stärke der corpus-statistischen Produktivität des Affixes korreliert; bei wenig produktiven Affixen ist die Häufigkeit der Oberflächenform des Wortes ausschlaggebend. Aus diesen und ähnlichen Ergebnissen läßt sich schließen, dass viele komplexe Wörter während der Erkennung zerlegt werden, dass aber häufig verwende-

te Wörter über ihre Oberflächenform erkannt werden. Die Zerlegung eines derivierten Stamms scheint weniger deutlich zu sein als die einer flektierten Form. Sehr häufig vorkommende Wörter müssen also eher in einer unzerlegten Form im Lexikon vorliegen. Da im Verlauf des Sprachwandels komplexe Wörter oft ihre semantisch Transparenz und demnach ihre Zerlegbarkeit verlieren, muss ein relativ fließender Übergang von der zerlegten zu der unzerlegten Darstellungsform möglich sein. So scheinen auch diese Überlegungen, die Hypothese der Dualität der lexikalischen Repräsentation zu stützen.

Diese Ergebnisse können aber auch als Evidenz für eine weitere Teilung des Lexikons interpretiert werden. Als vom Stamm getrennt erkannte Einheiten müssen gebundene Affixe und Pseudo-Affixe auch eine eigene lexikalische Darstellungsform haben. Die eingeschränkten Erkennungseffekte der Pseudo-Affixe legen den Gedanken nahe, dass die Pseudo-Affixe in einem anderen lexikalischen Bereich als die eigentlichen Wörter und Morpheme gespeichert sind. Aufgrund ihres unterschiedlichen Verhaltens sollten auch die Flexions- und Derivationsaffixe möglicherweise getrennten Regionen des Lexikons zugewiesen werden.

Anforderungen der linguistischen Syntax und Semantik

Die letzte Instanz in Fragen der Bildung und Benutzung komplexer Wörter ist wohl die linguistische Morphologie. Als dritter ‚Interessenvertreter' in der Anforderungsanalyse wurden deshalb linguistische Studien über den strukturellen Aufbau von Wörtern und Sätzen herangezogen. Zahlreiche linguistische Daten bestätigen den Hinweis aus der lexikalischen Statistik, dass das angestrebte Modell Syntax und Lexikon als getrennte Module behandeln muss. Die in unifikationsbasierten Grammatiktheorien aufgestellte Hypothese der Lexikalischen Integrität [BM95] geht aus Beobachtungen hervor, die zeigen, dass Wörter nicht nach den gleichen Prinzipien strukturiert sind wie Sätze, und dass die Satzstruktur keine unmittelbaren Verweise auf die innere Struktur eines Wortes enthält. Ferner wird gezeigt, dass es ein Irrtum ist, die Bedeutung aller komplexen Wörter kompositionell in die Bedeutung eines Satzes (wie bei *abwandern*, oben) einbinden zu wollen. Für viele durchaus verständliche Wortschöpfungen ist die Bedeutung nämlich nicht rein kompositionell rekonstruierbar, und andere Merkmale wie Argumentstruktur sind ohne Rekurs auf tieferliegendes konzeptuelles Wissen nicht erklärbar. Anhand französischer, deutscher und italienischer Beispiele wird gezeigt, dass es im Prinzip möglich ist, formale Ableitungsschritte für die Bedeutung vieler Derivate anzugeben, indem man tieferliegendes konzeptuelles Wissen in die Derivation mit einbezieht. Gleichzeitig lassen die angedeuteten Analysen vermuten, dass die Verarbeitungskomplexität der Wortderivation sehr hoch sein kann. Um diese Verarbeitungskosten zu begrenzen, scheint die Annahme unumgänglich, dass das kognitive Sprachsystem eine Strategie entwickelt haben muss, diese Neuberechnungen des Derivationsschrittes zu vermeiden, indem es komplexe Derivate meistens direkt aus dem Lexikon eingefügt.

Der Schritt zu einer Spezifikation des Modells

Die eng begrenzte statistische Produktivität der Wortbildung, die widersprüchlichen experimentellen Daten zur Erkennung von Affixen, und die hohen Verarbeitungsko-

sten, die bei der Derivation entstehen, mussten im Spezifikations-Modell von KLU berücksichtigt werden. Dafür wurden einige formale Erweiterungen der LFG eingeführt. Die Möglichkeit, Derivations-Semantik in einem auf dem Lambda-Kalkül basierenden Formalismus zu beschreiben, wird in den formalen Apparat aufgenommen. Die Generierung und Analyse von komplexen Wörtern mit einer gegebenen statistischen Verteilung kann mit stochastischen Produktionsregeln in einer reduzierten Version des LFG-Formalismus erfaßt werden. Um komplexe Wörter in ihre Bestandteile als Eingabe zur Wortanalyse zu zerlegen, musste auch eine phonologische bzw. orthographische Komponente spezifiziert werden; es wird gezeigt, dass die in der Literatur bereits bekannten stochastischen Umwandler diesen Zweck erfüllen [Lev74].

Die statistischen, experimentellen, und linguistischen Daten legen eine Aufteilung des Lexikons in drei Regionen nahe. Diese sind:

- Wörter, auch komplexe lexikalisierte Wörter, die semantische Prädikate enthalten, wie Hauptverben, Nomina, relationale Präpositionen.
- Monomorphemische Wörter bzw. Stämme, die deiktische Prädikate (wie Pronomina) oder kasusmarkierende und ähnliche Merkmale enthalten; ebenso Derivationsaffixe, die Derivationsprädikate enthalten.
- Gebundene Flexionsaffixe, die monomorphemisch sind und keine Prädikate tragen.

Ein nützlicher Effekt dieser Aufteilung ist, dass das Wachstum eines Corpus fast ausschließlich auf die erste Region des Lexikons beschränkt ist. Eine weitere Motivierung der postulierten Aufteilung bietet die Annahme, dass die phonologisch motivierten Schichten des „level-ordering“-Modells von Kiparsky [Kip82a] mit den hier postulierten Regionen des Lexikons grob übereinstimmen. Die unterste Schicht des level-ordering-Schemas enthält nur Stämme und Wurzeln, welche ähnlich wie die Einträge der ersten Lexikon-Region, sehr spezifische semantische Informationen enthalten. Die nächste level-ordering-Schicht enthält typischerweise Derivationsmorpheme, die zur zweiten Region des Lexikons gehören. Die dritte Schicht der Phonologie wirkt in der Regel nur auf Flexionsmorpheme, die nur in der dritten Region des Lexikons vorkommen. Ein derart geschichtetes Lexikon erlaubt ein Segmentierungsverfahren, das die Anwendung von phonologischen Regeln auf bestimmte Segmente des Wortes beschränken kann. Gleichzeitig schränkt es auch die Suche nach lexikalischen Segmenten ein, besonders in der äußersten Schicht des Wortes (Flektion), in der (wegen Segmentierungs- und Erkennungsambiguitäten) vermutlich am häufigsten im Lexikon gesucht werden muss. Eine naive Suche nach Segmentierungsgrenzen im Wort kann durch die regionale Aufteilung des Lexikons gesteuert werden, ohne Kenntnis der morphologischen Struktur des Wortes, die erst nach der Feststellung der Segmentgrenzen sichtbar wird. Diese Strategie vermeidet die Komplexität eines Modells, in dem Segmentierung, Phonologie und morphologische Analyse zusammenfallen, z.B. [Tro91].

5. Ergebnisse der Modellbildung

Die Modellbildung hat zu einem Computer-Programm geführt, das wesentliche Aspekte des formalen Modells implementiert, und das anhand einiger Beispielsätze zeigt,

dass die postulierten Verarbeitungsschritte zu den erwarteten Zwischen- und Endergebnissen führen können.

Die Implementierung des KLU–Programms in Prolog II+ beinhaltet einen Compiler für einen leicht geänderten LFG-Formalismus. Im Gegensatz zu den im KLU–Modell vorgeschlagenen stochastischen Regeln erkennt der KLU–Compiler lediglich ungewichtete Regeln. Das Laufzeit-System, das die compilierte Grammatik interpretiert, wird weitgehend durch den Prolog-Beweiser realisiert, da der Compiler die LFG-Regeln und die Lexikon-Einträge in Definite-Clause-Grammars [PW80] übersetzt. Die orthographischen Umwandler und die Einträge einer konzeptuellen Wissensbasis müssen vom Benutzer in Prolog geschrieben werden. Einige einfache Beispiele der Satzbearbeitung mit Eingabe- und Ausgabe-Protokollen werden präsentiert. Die Analyse von zwei italienischen Wortschöpfungen im Satzkontext wird als Beispiellösung vorgeführt.

Die Anforderungen aus der Problemanalyse werden in dem zugrundeliegenden kognitiven KLU-Modell wie folgt berücksichtigt:

Implikationen der Corpusstatistik

Um das Wachstum des Wortschatzes in einem Corpus zu simulieren, müssen Informationen im Lexikon vorhanden sein, welche die statistischen Häufigkeiten der Wörter kodieren. Unter der Annahme eines unendlich großen Lexikons würde zu diesem Zweck ein System von terminalen Produktionen ausreichen, in dem eine produktive Wortklasse durch eine unbegrenzte Anzahl von Regeln mit verschwindend kleinen Wahrscheinlichkeiten dargestellt wäre. Da diese Annahme kognitiv unplausibel ist, ist es notwendig, rekursive Wortbildungsregeln einzuführen, die das unbegrenzte Vokabular der produktiven Klassen aus Morphemelementen generiert. Andererseits enthalten auch diese Wortklassen einzelne Stämme und Wörter, die wie Wörter der nichtproduktiven Klassen häufig erscheinen und nicht zum Wachstum der Klasse beitragen. Die lexikalische Einfügung von Wörtern einer bestimmten Wortbildungsklasse muss daher sowohl Einträge aus dem Lexikon übernehmen, als auch neue Wörter durch eine Morphemkombinatorik erzeugen können. Für die Einfügung komplexer Wörter in die Struktur eines Satzes werden diese Alternativen im KLU–Modell durch eine lexikalische Einfügungsprozedur realisiert, die beide Möglichkeiten vorsieht.

Analogisch gebildete irreguläre Wortformen, wie z.B. **rept* als eine verständliche, aber nicht anerkannte Vergangenheitsform von Englisch *reap* (ähnlich zu *keep – kept*) können mit stochastisch gewichteten Umwandlern modelliert werden, die im KLU–Programm darüberhinaus auch die Lemmatisierungsregeln für den Abgleich der Wortsegmente mit dem Lexikon implementieren (allerdings ohne Gewichtung).

Diese statistischen Aspekte des Modells sind von großem theoretischen Interesse; ohne Zugang zu passenden Corpora und einer umfangreichen Wortgrammatik kam für das KLU–Projekt jedoch ein Implementierungsversuch nicht in Frage. Die Regelsysteme, die erforderlich wären, um die Streuung der Produktivitätswerte in Corpora zu simulieren, unterstützen in jedem Fall die nicht-statistische, duale Repräsentation von komplexen Wörtern im KLU–Modell.

Implikationen der Psycholinguistik

Die psycholinguistischen Daten legen die Annahme nahe, dass viele komplexe Wörter einerseits als lexikalische Einheiten, und andererseits auch durch Analyse in ihre Bestandteile erkannt werden. Im ersten Fall liegt im mentalen Lexikon eine Art Abbild des Wortes vor, das mit einer statischen Datenstruktur repräsentiert wird. Im zweiten Fall hat das zerlegte komplexe Wort keine eigene Darstellung im mentalen Lexikon. Es existiert intensional, als mögliches Ergebnis von Wortbildungsregeln, die mit einem vereinfachten LFG-Formalismus dargestellt werden.

Eine weitere Dimension der lexikalischen Darstellung stellen die bedeutungslosen Pseudoaffixe (wie *re-* im englischen *reproach*) dar, die scheinbar getrennt erkannt werden, aber keine entsprechende Darstellung in der Wortsemantik haben. Daten zur Verarbeitung der Pseudomorphemen unterstützen eine Darstellung von manchen atomaren Wörtern als Listen von lexikalischen Segmenten, die durch Listenunifikation erkannt werden. Für diese Wörter führt die Erkennung eine Segmentierung, aber keine morphologische Wortanalyse durch. Folglich wird *reproach* in *re-* und *proach* segmentiert, aber die Erkennung erfolgt einfach dadurch, dass ein Eintrag für *re-* einen Zeiger auf den Eintrag *proach* enthält, welches wieder mit den syntaktischen und semantischen Merkmalen eines morphologisch unzerlegbaren Eintrags *reproach* verbunden ist. Die Segmente sind also keine Morpheme, sondern lediglich inhaltslose Zugriffsschlüssel. Die Segmente müssen in einem Bereich des Lexikons liegen, auf den die Wortsegmentierung schnell und ohne Aufwand zugreifen kann, weil Phoneme bzw. Zeichen, die ein Wort bilden, oft unsicher erkannt werden, und weil mögliche homonyme Teilwörter die Segmentierung rechnerisch äußerst aufwendig machen können. Das Modell postuliert, dass diese Segmente zu einer besonderen Region des Lexikons gehören, und dass ihre Repräsentationen nicht unmittelbar mit den Datenstrukturen verbunden sind, welche die syntaktischen und semantischen Eigenschaften von Wörtern beschreiben.

Nicht-lexikalisierte komplexe Wörter hingegen haben keine eigenen Einträge im Lexikon. Ihre Segmente können erst nach der Segmenterkennung und nach einer Umsetzung der Segmente in Morpheme zu einem Wort zusammengefügt werden. Aus der Hypothese der Lexikalischen Integrität folgt (anders als für ein segmentierbares, aber monomorphemisches Lexem wie *reproach*), dass ein Wort wie die nicht lexikalisierte englische Wortschöpfung *re-caress* nicht ohne weiteres in die Syntax eingefügt werden kann. Es wird zwar ebenfalls zunächst segmentiert, aber die Verkettung von *re-* mit *caress* liegt im Lexikon nicht vor, was bedeutet, dass für *re-caress* keine syntaktischen und semantischen Attribute auffindbar sind. An dieser Stelle muss eine morphotaktische Analyse eingeleitet werden. Diese ist vermutlich zeitlich viel aufwendiger als die reine Erkennung einer Segmentliste, was von einigen experimentellen Daten bestätigt wird, sowie auch von den Analysen der linguistischen Semantik, die oben erwähnt wurden.

Diese verschiedenen Möglichkeiten der Erkennung werden im KLU–Modell durch eine lexikalische Einfügungsprozedur realisiert, die unten eingehend beschrieben ist.

Implikationen der Syntax und Semantik

Die Hypothese der Lexikalischen Integrität unterstützt eine weitgehende Trennung des Sprachsystems in ein satzsyntaktisches Modul und ein lexikalisches Modul; sie erklärt

aber nicht, wie Wortschöpfungen, die im Lexikon nicht gespeichert sind, in die Satzstruktur gelangen. Aufschlußreich sind dagegen Fälle, wie das oben angeführte Beispiel von Stiebels, in denen es scheint, dass die Syntax die Konstituenten des Wortes unmittelbar in die Satzstruktur einbauen kann. Diese Daten schließen die Möglichkeit aber nicht aus, dass die Wortanalyse getrennt stattfindet, und dass allein ihre Ergebnisse der Syntax überreicht werden. Aufgrund des hohen Verarbeitungsaufwands der Derivation ist anzunehmen, dass die Wortbildung ein getrenntes Regelsystem bildet, und dass dessen Ergebnisse nicht jedesmal neu berechnet werden, sobald ein bereits analysiertes Wort wieder erscheint. Statt dessen postuliert das KLU–Modell einen lexikalischen Puffer, der wie ein „cache“ die Ergebnisse einer Derivationsberechnung zwischenspeichert und für eine bestimmt Zeit bereithält, um die Verarbeitungskosten einer erneuten Derivation zu vermeiden.

Die lexikalische Einfügungsprozedur sieht folgende Alternativen vor, die sich nicht gegenseitig ausschließen. Sie kann also u.U. mehrere Interpretationen eines komplexes Wortes gleichzeitig anbieten:

- Ein atomares Wort wird unmittelbar aus dem Lexikon mit dem präterminalen Symbol der Syntax unifiziert.

- Ein segmentierbares Wort mit Pseudoaffixen oder mit einer lexikalisierten morphologischen Segmentierung wird unmittelbar aus dem Lexikon mit dem präterminalen Symbol der Syntax unifiziert.

- Ein segmentierbares Wort mit einem Flexionsaffix, das mit dem Stamm unifiziert, wird durch eine unifikationsbasierte Wortgrammatik erkannt, und alle Merkmale werden mit dem präterminalen Symbol der Syntax unifiziert. Eine Kopie wird aber auch im lexikalischen Puffer abgelegt, um eine mögliche Lexikalisierung der flektierten Form zu erlauben.

- Ein Wortstamm mit einem Derivationsaffix, das mit seiner Basis unifiziert, wird durch eine unifikationsbasierte Wortgrammatik erkannt. In den meisten Fällen wird ein Prädikat als Ergebnis der Unifikation, aber keine wohlgeformte lexikalische Form (d.h., keine korrekte Argumentstruktur bzw. Mapping) resultieren. Diese Ungrammatikalität der Wortstruktur löst einen Einfügungsfehler aus, der von einer übergeordneten Fehlerbehandlung abgefangen wird. Um die fehlende lexikalische Form zu erzeugen und mit einer Argumentstruktur zu versehen, muss eine konzeptuelle Auswertung mit anschließendem Argument-Mapping stattfinden; außerdem müssen Eigenschaften wie lexikalische Kategorie, Flexionsklasse, Genus und besondere Subkategorisierungsbedingungen berechnet werden. Gelingt es, eine wohlgeformte und semantisch interpretierbare Beschreibung des Wortes zu bilden, wird diese in den lexikalischen Puffer eingetragen. Die Kontrolle wird dann an die lexikalische Einfügung zurückgegeben. Diese kann jetzt den Eingabestring als wohlgeformtes grammatisches Wort aus dem lexikalischen Puffer holen und mit den Merkmalen des präterminalen Symbols der Syntax unifizieren.

- Wiederholte Zugriffe auf ein im Puffer abgelegtes Wort verstärken seine Repräsentation, aber sie machen die semantische Repräsentation des Wortes un-

abhängig von der Analyse, die ursprünglich aus den morphologischen Bestandteilen gewonnen wurde. Durch die Loslösung der worteigenen semantischen Struktur von der morphologischen Struktur kann das Wort eine nicht mehr transparente Bedeutung bekommen.

Programm-Protokolle für die Derivation von italienisch *funghini* ‚kleine Pilze' und *disiscrivere* ‚ex-matrikulieren' in einem Satzkontext zeigen, wie die Wortsegmentierung, Wortanalyse, Zwischenspeicherung und anschließende Satzanalyse in der KLU–Implementierung durchgeführt werden.

Ergebnisse

Die Struktur komplexer Wörter kann mit denselben Mitteln der Unifikationsgrammatik gut beschrieben werden, die sich für die Satzanalyse bewährt haben. Auch die Semantik von Derivaten kann wahrscheinlich weitgehend mit solchen formalen Mitteln erfaßt werden, obwohl die Rolle der zugrundeliegenden konzeptuellen Operationen noch viele Fragen offen läßt. Viele morphologischen Daten sind aber nur unter der zusätzlichen, prozedural formulierten Bedingung erklärbar, dass komplexe, lexikalisch nicht gespeicherte Wörter außerhalb der Syntax analysiert werden, und dass das Ergebnis einer Wortanalyse nicht unmittelbar in die Satzstruktur integriert, sondern in einen lexikalischen Puffer abgelegt wird. Ein prozedural definierter lexikalischer Einfügungsmechanismus macht erst klar, wie die Ergebnisse der Wortanalyse in die Syntax des Satzes eingefügt werden können, ohne die Lexikalische Integrität zu verletzen. Im Falle der Derivation dient der Puffer dazu, den hohen Verarbeitungsaufwand bei wiederholtem Zugriff auf das Wort zu vermeiden.

Es ist plausibel, dass die Inkompatibilität zwischen Wortstruktur und Satzstruktur auf die gleiche Weise wie viele andere mögliche Fehler bei der Erkennung eines Wortes behandelt wird. Ein Fehlerbehandlungsmechanismus kann die Vermittlung zwischen Syntax und Derivation formal gut beschreiben, und ein solcher Mechanismus ist darüberhinaus nötig, um andere Einfügungsfehler, wie Abkürzungen, Rechtschreibefehler, usw., zu behandeln. Der unterschiedlich starke Grad der Lexikalisierung innerer und äußerer Segmente eines Wortes muss durch einen Regelapparat erfaßt werden, der die unterschiedlichen Prioritäten von Phonologie und Lexikon in den inneren und äußeren Wortsegmenten berücksichtigt. Wenn man diese relativ einfachen prozeduralen Ergänzungen zu einer deklarativen Beschreibung von Wortbildung und Syntax in Kauf nimmt, kann ein beinahe vollständiges Modell der Derivationsmorphologie im Formalismus einer Unifikationsgrammatik wie LFG erstellt werden.

Probabilistische Ergänzungen des Modells, welche den Grad der Lexikalisierung und die Produktivität von Wortbildungsmustern beschreiben sollen, konnten im KLU–Programm nicht implementiert werden. Die Implementierung selbst wäre vermutlich nicht schwierig, doch standen die Corpusdaten, die nötig sind, um sinnvolle Werte in die Regeln des Systems einzusetzen, bisher nicht zur Verfügung.

Preface

> although the artificial intelligence approach was necessary to haul us out of our false preconceptions ... it ... became limited and hidebound because of its failure to recognize what a true computational theory is and how it should be deployed (D. Marr, *Vision* [Mar82, 345]).

I remember being told as a school child that the worst kind of mistake one could make in an essay, next to a spelling error, was to use a "made-up" word that was "not in the dictionary". Today we have computer programs that can find many of our spelling errors, but probably everyone who has used such a program at any length has discovered how difficult it is for such a program to say what is really in the dictionary. Nevertheless, an indispensible assumption of modern linguistics is that languages can be described as sets of relations among symbols in a lexicon, and computational models of languages have by and large modeled the lexicon as a known and finite set of atomic symbols. Many of the well-known models have enjoyed the good fortune of being constructed for English, a language whose linguistic culture tends to see "the dictionary" as a kind of social check on the excesses of personal linguistic indulgence, and whose morphology in any case severely limits the amount of word-forming creativity that a text-processing computer program is likely to encounter. But natural languages on the whole probably exhibit more creativity in the lexicon than English and make it much harder to say what is and is not "in the dictionary". As language processing technology is applied to an increasingly wide variety of the world's languages, it will be increasingly confronted with the inadequacy of a model that makes a strict distinction between the lexicon and the creative, combinatorial system that generates sentences.

This study is addressed to the question of how lexical creativity can be described by a formal computational system. It is certainly not the first study to address this question — the problems are indeed well known — but it may be the first attempt to apply a strategy of gathering a wide range of detailed "requirements data" in order to apply principles of software engineering to the construction of the model. It has been guided, on one hand, by the experience of software practitioners that large computer programs suffer most often not from poorly devised algorithms and data structures, but from inadequate analysis of the task the program is meant to fulfill; and on the other hand from the conviction that structural linguistic analysis is not the sole, royal road to knowledge about how linguistic systems function. It is my hope that it will

convince computer scientists and linguists that, as D. Marr has argued for theories of vision, computational linguistic models can do more than mimic superficial behavior — that they can make a significant contribution to the understanding of underlying representations and processes.

In a limited way, the model of lexical creativity that will be presented here has been implemented as a computational system called KLU (for *Konstanz LFG-Umgebung*), in conjunction with research projects on lexical semantics and derivation sponsored by the Deutsche Forschungsgemeinschaft and led by Christoph Schwarze. However, I shall not devote much space to describing this computer program here. Just as no one can understand a complex program by "simply reading the code", even lengthy technical descriptions of its algorithms and data structures help little if one does not first understand what the program is meant to accomplish and why various implementation strategies were chosen.

KLU began with a natural language dialog system, based on a model of the syntax-semantics interface conceived by Christoph Schwarze and Peter Pause, that was meant to answer questions about the semantic implications of sentences containg verbs of motion (described in [May95]). In connection with studies of derivation in Italian [GT91], and influenced by a study of compounding in German [Kan85], this system was reorganized and extended to interpret sentences containing newly derived, hence non-lexicalized verbs and other categories. A outline specification of the system appeared in [May93], but a really functional program had to await the development of fully developed specifications of word segmentation, word parsing, semantic evaluation, argument mapping, and lexical insertion. The implementation of these modules turned out be more complex than anticipated, but it was precisely these implementational problems that revealed a number of important possible constraints on the structure of the mental lexicon. The proposed structure of the lexicon also had to be refined in other ways, in order to come to grips with apparent contradictions between what has been called the Lexical Integrity Hypothesis [BM95, 91-95]), which requires that words be presented as atomic wholes to the syntax component of language, and the opposing necessity, evident both in psycholinguistic data and in computational constraints, of representing individual morphological segments to on-line processing, which is often seen as the domain of syntax.

Although some of its required modules are still only available as mock-ups, KLU now presents a picture of the interface from word formation to syntax that appears to resolve these contradictions, and the implementation work has now made it gratifyingly evident that the interface from syntax to the lexicon must be quite complex and computationally expensive by comparison with syntactic processing, as our original deliberations had predicted. Most importantly, attempting to come to grips with the full complexity of derivation in the implemented model has revealed that there may be good reasons, grounded in a kind of computational optimization, why the language faculty has evolved separate but interacting generative systems for sentences and for words. While there are many other sources of evidence for the theoretical model that has emerged, the working computational model demonstrates nicely, I believe, the kind of contribution a computational implementation can make to the study of language.

To be sure, an implementation by no means proves the rightness of a theory. Model-building of any sort involves abstraction and simplification, and just like any paper-and-pencil model, an implemented computational model can simplify or simply

ignore significant data. But it is in general less easy to sweep large theoretical problems under the carpet in an implementation. Any account of derivational morphology brings with it a head-on confrontation with a wide gamut of linguistic issues, ranging from allomorphy to knowledge representation, which forces one to take a decidedly integrationist perspective on theory construction. Futhermore, a research implementation of a language processing model can only hope to test very small linguistic corpora, which imposes a problematic abstraction from the real task of describing what Chomsky identified as language's "infinite use of finite means" [Cho65, 8]. A model built to process a small number of test sentences is hardly equal to this task. It is, however, possible to further constrain the model, or to limit the number of possible models, by integrating further data from the modeling domain, such as statistical evidence concerning word formation in large corpora, psychological data on the processing of complex words, or the history of certain morphological changes.

Each of these sub-domains is, of course, a field of study unto itself, with a complex research methodology and an enormous literature that only specialists can evaluate adequately. It comes as no surprise that cognitive science has made relatively few visible attempts to integrate coherently the many diverse sources of information about language processing now available. In practical computer science, however, the problem of integrating a wide diversity of complex and unrelated domain constraints for an application is often inescapable, however formidable the task may be. This has led to the growth of a proto-discipline known as "requirements engineering", meant to address the problem of formulating what a complex program must do, at a stage prior to any assumptions about the how the solution could be implemented, but in terms that can lead directly to a formal specification. Requirements analysis in application programming thus corresponds in some ways to what T. Kuhn characterized in the history of science as an early, "pre-normal" stage of theory construction, prior to the development of a standard, widely accepted formal model [Kuh62]. Getting from the set of requirements to a formal specification of a solution is thus often, in miniature, like the distillation of an explanatory paradigm from a mass of superficially unrelated data. That cognitive linguistics still lacks a single, binding explanatory model is surely what makes it such an exciting field of study at present, but widely accepted models will inevitably emerge. A secondary, if not too ambitious, goal of this study is to give an example of how this might come about. I hope to show how an explicit consideration of the requirements analysis stage of application development can help us better understand what a model of language processing needs to do, and how a model constructed according to principles advocated in software engineering can return a meaningful contribution to linguistic theory.

As a final prefatory remark let me emphasize that "we" as I occasionally use it indeed refers to a number of collaborators, all of whose work has shaped the model that will be described here. Integrating linguistic theory and computer science is not for the faint of heart, and I have benefitted over several years from the breadth and depth of my colleagues' understanding of the real issues. As mentioned above, the original conception for KLU built on a model of the syntax-semantics interface proposed by Christoph Schwarze and Peter Pause. Linguistic insights of Marie-Theres Schepping and Christoph Schwarze have continually steered the development of KLU, and more recently the construction of larger grammars for French and Italian by Christoph Schwarze and Veronika Knüppel has influenced its current form. My understanding of

important issues in lexical semantics was deepened over the years in numerous discussions with P.-M. Hottenroth. Katrin Mutz, Heike Necker and Vieri Samek-Lodovici have provided many valuable observations. Numerous other users of the KLU program have suffered through its growing pains, pointed out its weaknesses, and have suggested improvements. Aditi Lahiri introduced me to the complex and often puzzling data from psycholinguistics research. Christopher Habel I wish to thank for a number of fruitful suggestions and for the opportunity to present and discuss early sketches of this work in seminars at the Institut für Informatik at the University of Hamburg. Special thanks are due to Regina Eckardt, Veronika Knüppel, Aditi Lahiri, Peter Pause and Marie-Theres Schepping for helpful comments on portions of the manuscript, and to Judith Meinschaefer for many inspiring discussions and comments. Remaining errors and misunderstanding are, of course, my own.

Contents

Chapter 0

Introduction

0.1 Computation, Logic and Control in Word Formation

Probably the single most important development in computational linguistics has been the invention of formalisms for unifying terms, graphs and sets of equations, and of techniques for describing natural languages with them. Using such techniques, it is possible to formulate grammars for natural languages that derive the structure and semantics of a sentence by combining attributes of the sentence's constituents, without reference to the ordering of operations involved in the calculation. The enormous advantage of such formulations is that each of the facts or relations in the description of a language is a statement that can be independently verified against a purely structural description of the language; the correctness of one structural fact does not depend immediately on other facts or relations in the description in the way that statements in a procedural description do. Some of the well-known declarative, unification-based formalisms include PATR-II, Lexical Functional Grammar (LFG), GPSG, and HPSG. The general approach is surveyed in [Shi88]; more specific arguments in its favor are presented in [BK82] and in [PS87].

While declarative formalisms have made it possible to obtain abstract but easily understood descriptions of a wide range of linguistic phenomena, they do not easily lead to solutions for all problems. One problem that has stubbornly resisted neat, purely unification-based description is the interface from word-formation to sentence structure. Using the same techniques used for analyzing sentences, linguists can identify structures within words, and as for sentences they can construct unification-based word grammars that produce structural and semantic descriptions of complex words. If word analysis and sentence analysis never needed to interact, purely unification-based analyses of word and sentence structure might be adequate. But to a greater extent than has been generally appreciated, speakers of natural languages introduce spontaneously generated, new words as if they were parts of sentences, suggesting that the word-generating and sentence-generating apparatus might be the same. On the other hand, rules for producing new words seem to be used much less frequently than those for producing sentences, and sentence grammar often uses complex words with meanings and argument structures that cannot be derived in any transparent way by a word grammar, indicating that it would be a mistake to have the sentence grammar analyze

all words into their constituent parts.

Consider briefly a case that will be discussed in more detail in chapter 4. In the sentence (1), the German verb *abwandern* is used in the unusual sense of 'use up by hiking', although it usually means 'emigrate or depart'.

(1) Max wird seinen Mitgleidsbeitrag für den Alpenverein abwandern.
'Max will hike off his dues to the Alpine Association' [Sti96, 143]

Since a non-existent word meaning cannot be fetched from the lexicon, it must be possible to interpret the word's constituents as parts of the sentence. In a unification grammar, this could be accomplished easily by treating the word's constituents as atomic symbols of the grammar, unifying them with the remaining constituents of the sentence. Sections 4.4.3 and 5.6 will later demonstrate how a declaratively formulated grammar can do this. But for a variety of reasons that will be explored in subsequent chapters, the purely unification-based approach gives a false picture of the actual relationship between word formation and sentence grammar. Some problems representative of the difficulties involved are

- the simultaneous availability of both an atomic, lexicalized word (e. g., *abwandern* in the sense of 'depart') as well as an interpretable word-syntactic structure (*[ab[wandern]]*);
- changes in the argument structure of a word, e. g., the subcategorization for a direct object by a new verb (*abwandern* in the sense of 'hike off', which neither the lexicalized *abwandern* nor the base *wandern* 'hike' allows);
- the considerable difference in statistical productivity between rules of word formation and of sentence generation, with many word-formation rules hardly ever being used productively.
- the tendency of new words to acquire non-transparent meanings over time if used frequently.

There is at present no computational model that gives a convincing account of these phenomena. The reason may well lie less in the complexity of the data, however, than in an inappropriate conception of how computational linguistic models should be constructed. It is a curious fact that the preoccupation of computational linguistics with compact and powerful declarative formalisms stands in stark contrast to the methods that have proved successful in conventional software engineering. Conventional software projects often make use of a wide range of programming paradigms, ranging from the relatively declarative languages of relational data bases to the highly imperative, low-level languages used for machine control, and complex application programs very often mix a number of programming languages to obtain best fits to the various domains the application must deal with. Furthermore, studies of successful software engineering have repeatedly shown the danger of abstracting too quickly from a project's requirements to a single, neat formal specification. Careful requirements analysis, taking stock of all the demands that can be made on a program, has become recognized as the most critical phase in the development of an application program.

Yet it would be foolish to think that constructing computational models of cognitive systems could be any simpler than more conventional application programming. In many cases, of course, progress has been possible in computational linguistics only by ignoring many evident requirements posed by linguistic systems — by concentration on partial solutions to problems in syntax, phonology or morphology without regard to their interfaces to other domains. An account of how new, derived words are incorporated into sentences cannot allow itself this luxury, however, for derivation is subject to constraints arising from a wide range of linguistic phenomena, from phonology to sentence semantics. A central assertion of this study will be that the abstraction and simplicity offered by declarative, unification-based formalisms tend to deflect attention from issues of storage, control and efficiency that would be raised by a thorough requirements analysis of the problem.

0.1.1 Declarative Formalisms for Computational Linguistics

To appreciate why the declarative approach of unification grammar might be inadequate for a comprehensive description of word formation, it is useful to recall some of the basic issues that arise in giving a declarative formulation of any computational algorithm. In an influential article, R. Kowalski argued that "Algorithms = Logic + Control" [Kow79], i.e., that any description of a computational procedure can be broken into a logic component and a control component. The logic component can be seen as a set of statements about static aspects of the procedure, including data structures but also including abstract procedural relations like the static calling tree and quantification of variables over data structures. The control component consists of statements describing how computations are carried out, for example, the order in which data structures are searched, strategies for deciding among alternative execution paths, methods of fetching and storing data, and methods of synchronizing parallel paths of execution. In many cases the same information can be represented as both a statement of logic, e.g., that a variable has universal quantification, and as a control statement, e.g., that the variable is assigned a series of values inside a loop.

Early efforts at machine-based natural language processing were formulated in highly control-oriented or procedural terms, based in many cases on the ways in which transformational generative grammars were stated (cf. [BF95]). Some computational systems developed in the 1970s, such as the LUNAR system [WKNW72] or the programs of Roger Shank and his associates [Sha81], mixed declarative and procedural descriptions, but the resulting systems were found to be unmanageably complex and hard to understand, often containing theoretically unmotivated control statements. This situation undoubtedly contributed to the welcome reception given in the 1980s to unification-based formalisms, and it was one of the primary reasons that the designers of Lexical Functional Grammar (LFG) turned to a completely non-procedural formalism [Kap95b] in which the control component was not accessible to the grammar writer. While not thought of as programming languages, formalisms like LFG shared the insight of computer scientists that the programming of complex tasks required a shift to less procedurally-oriented notations, using various forms of structural or equational unification as their main operation, just as did the programming language Prolog and its descendants [Kow74]. These declarative formalisms helped to keep linguistic descriptions empirically motivated and understandable as well as separate from the

often arbitrarily chosen implementation, and they allowed procedural optimizations to be removed to the compiler.

A distinguishing characteristic of declarative formalisms like LFG is that they are typically formulated in terms of equational relations rather than in terms of changes of state. A simple algebraic equation like $A = B$ can be seen as a logical statement that can be evaluated in various ways, depending on how its variables are bound. If A has the value 2 but B is unbound, an implication of the equation is that $B = 2$; to *obtain* this implication, it is necessary to assign the value 2 to B, i.e., to change B's state. This is the essence of the distinction: while the logic component of the algebraic formalism provides a description of relations among objects in the problem domain, it provides no notation for elementary processes like change of state that are required to solve problems in the domain. This is the task of the control component, which in this case must provide an assignment operation in order to draw implications from the equation.

The PRODUCTION RULE formalism used, for example, in LFG for describing the constituent structures of words and sentences, has a similar characteristic. The rule

(2) $\quad S \rightarrow NP \quad VP$

has the logical reading that if a string of words is found to contain a noun phrase *NP* followed by a verb phrase *VP*, then it is a sentence *S*; alternatively, it can be read as meaning that a sentence *S* can be generated by generating an *NP* followed by a *VP*. A further implication of the rule is that if a string *S* is known to be a sentence and is known to end with a verb phrase *VP*, then the string preceding *VP* is necessarily a noun phrase. For obtaining each of these inferences there exist one or more well-known control strategies, such bottom-up shift-reduce parsing; the formalism of Definite Clause Grammars, sketched in section 7.2.1, allows all inferences to be drawn in the framework of a theorem prover for Horn clauses (Prolog). While these methods have their own logical representations, they too are built on a control level that contains state-changing operations needed to interpret the logic component of the formalism, assigning truth values to logical statements or strings to symbols of the grammar.

Another useful declarative formalism for describing natural languages defines FEATURE STRUCTURES as sets of ATTRIBUTES paired with VALUES. Attributes must be atomic symbols, but values can be either atomic symbols or feature structures. A unification operator defined over feature structures lends itself well to describing a great many relations found in natural languages, like agreement among constituents (cf. [Shi88]).

An important characteristic of these formal descriptive devices is that the operations they define are both commutative and associative. This means that the control component has a great deal of freedom in choosing the order in which it carries out state-changing operations. The user of such a formalism is thus easily led to think that order of execution plays no significant role in the domain being described, and that where and when intermediate assignment operations take place can have no effect on the outcome. However, even in relatively simple computational problems, the order in which commutative or associative operations are carried out can have an enormous effect on efficiency. The problem of specifying an algorithm adequately to describe how word grammar is integrated into sentence grammar, I shall eventually show, is at

bottom not a problem of logic but a problem of control, and I shall present reasons for thinking that the specific form of the algorithm in the human linguistic apparatus has resulted as a kind of computational optimization. Here I simply want to prepare the road ahead by bringing this fundamental, underlying issue into focus.

0.1.2 Pitfalls of Declarative Specifications

A characteristic of most large and useful computer programs is that they do make large adjustments to minimize computation time and to best utilize the resources of the machine on which they must run, even when they are written in languages that are meant to hide the machine's architecture. I want to illustrate here how constraints imposed by the storage architecture of a computer — and possibly of the human language faculty — can affect the specification of a computation that relies heavily on access to large storage areas like the lexicon of a language.

In complex computational problems it is often the case that an algorithm repeatedly calculates intermediate results that could be simply saved and recalled, or that it could be reformulated to save time by reusing such results. Logarithms, which permitted calculations in astronomy and navigation that would have been otherwise impossible, were in effect a device for storing the results of complex but very general intermediate computations, in order to allow quick multiplication, division and exponentiation by means of simpler "on-line" operations like addition and subtraction. Of course, the faster calculations entail the nuisance of having a large book of logarithm tables continually at hand. The use of logarithms is an example of how of storage space can be traded off against processing time.

Stored intermediate results like logarithms are of little use, however, if they cannot be fetched from storage quickly, and how they are fetched can make a difference. For storing large numbers of similar items, like those found in a table or in a dictionary, imperative computer languages (Fortran, Pascal) typically offer an array data type as a fundamental data structure. This data structure provides a convenient way of notating mathematical objects such as vectors and matrices. A two-dimensional, $M \times N$ matrix can be conveniently described as an array of size M of vectors of size N as in (3), and a compiler will reserve $M \times N$ cells of storage for it in the program's data space, typically as a one-dimensional array of length $M \times N$, as in (4).

(3)
$$\begin{pmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,N} \\ X_{2,1} & X_{2,2} & \dots & X_{2,N} \\ X_{3,1} & X_{3,2} & \dots & X_{3,N} \\ \dots & \dots & \dots & \dots \\ X_{M,1} & X_{M,2} & \dots & X_{M,N} \end{pmatrix}$$

(4)
$$\begin{pmatrix} X_1 & X_2 & X_3 & X_4 & \dots & X_{M\times N} \end{pmatrix}$$

In principle, there is no limit to the size of such an array, apart from the amount of storage available to the program. In a computer that has a single layer of storage (no cache, no virtual memory), the access time to each cell will be more or less the same, but the amount of directly accessible storage (e. g., random-access memory or RAM) is severely limited by electrical and cost considerations. Almost without exception, modern computing systems implement some form of layered or HIERARCHICAL STORAGE

rather than the uniform, random storage implied in the array data type. As a result, a very large array often resides in a virtual memory area, meaning that the data of the array are not always present in the RAM layer. Instead, the system can, as needed, redefine the storage locations of cells in the array. Cells that appear not to be in current use can be displaced to positions on a secondary backing device, usually a magnetic disk, which has a capacity much larger than the RAM but also responds to requests for data much more slowly.

This re-mapping to the disk makes the RAM cells available for other data that are presumed to be needed frequently for the current calculation. It in no way affects the mathematical properties of the array, but it can drastically affect the time for computations on it. A cell of the array that has been displaced from RAM to the disk may have a latency, or access time, of some tens of milliseconds, while one that is available in the RAM can be read or written to in a microsecond or less. Because of the long access time, transfers to and from the disk generally do not take place cell-wise; to fetch one cell of the array it is advantageous to fetch an entire section of the array, called a "page". This is, of course, immediately due to the way the apparatus is built, but it has deeper causes relating to a granularity relationship between total size and unit or "chunk" size in a storage structure, to which I shall return shortly. [1]

Imagine now that this matrix is to be inverted. For a matrix X of size $M \times N$, this means that the contents of each cell $X_{m,n}$ must be exchanged with cell $X_{M-m,N-n}$. In a Pascal-like pseudo-code,

```
for n ← 0 to N do {For each X_{...,n}}
   for m ← 0 to M do {For each X_{m,...}}
      Temp ← X_{m,n}
      X_{m,n} ← X_{M-m,N-n}
      X_{M-m,N-n} ← Temp
   end for
end for
```

If M and N are very large and if much of the array must be held on the backing storage, this computation can be greatly slowed. When element $X_{1,1}$ of the array is accessed, the operating system may find that this cell does not currently reside in RAM. Because of the granularity relationship, the cell cannot be fetched individually from the backing storage. Instead, a PAGE FAULT occurs, meaning that the operating system must fetch an entire block of data from the backing storage and place it somewhere in RAM, so that the sought-for cell of the array can be accessed. Only then can cell $X_{1,1}$ be fetched individually from this block to the variable *Temp*. If the RAM is already largely filled, there will be no space in which to place this additional section of the array, and the operating system must overwrite a page currently in RAM. The access to cell $X_{M-m,N-n}$ may encounter the same problem, causing another page of RAM to be overwritten. Now, as far as the logic of the operation is concerned, it makes no difference in what order the cells are addressed. The procedural or control semantics of Pascal arbitrarily specifies that the inner loop runs to completion before the loop

[1] The complex subject of hierarchical storage management in computers is reviewed by P. Denning in [Den80].

variable is incremented. Therefore, the next array access, to $X_{2,1}$, is to a cell N cells distant from the first. If N is greater than the page size, this access will also require a page replacement, because the cell will not reside in the page that was previously fetched for the access to cell $X_{1,1}$. Once the inner loop of the inversion algorithm has run once to completion (i.e., to $m = M$), the outer loop counter m will be incremented, and an access to cell $X_{1,1}$ will be started. By this time the page that contained $X_{1,1}$ — and also contains $X_{1,2}$ — will have been overwritten and will need to be fetched again. This phenomenon, commonly called "page thrashing", arises because nearly every access to the array results in an access to the slow backing storage. It results from disregarding the control semantics of the higher-level, formally correct representation of the problem.

The standard solution to such problems in storage management is to "optimize for locality" the operations to be carried out. This means that the algorithm must be reformulated in such a way that successive operations are carried out on data that reside close to one another in the underlying data structure. In this well-known example, the solution takes the form of simply exchanging the inner and outer loops of the algorithm, so that successive assignment statements refer for as long as possible to contiguous storage. This amounts to a reformulation in which the first and the last 'chunk' of the array are fetched and elements are exchanged within the two chunks. Then elements in the second and next to last chunk are exchanged, and so on.

This access problem arises from the granularity relationship between successive levels in the storage hierarchy. While there is perhaps no general, physical law that necessitates such a relationship, without exception hierarchical storage architectures in computers have followed a principle of increasing storage granularity from primary to higher-order levels. In principle, each of these layers could be divided up into units of the same size, but this is not done because of the difficulty in addressing. Imagine a library which removed the bindings of books and stored only individual pages. Each page would require its own call number, and to obtain a book, a user would need to request hundreds of call numbers, which would have to be searched for and retrieved one by one. Yet access to an entire book would be prohibitively slow if all of the pages had to be requested individually. Chunking the pages into books in no way changes their meaning, but it has enormous advantages when one wants to read more than a few lines.

Users of, and compilers for, large computing systems must often reformulate the initial, abstract definition of a problem in such a way that often-used, intermediate results can reside in contiguous chunks of storage. Operations are so organized that once a program selects a chunk, it carries out computations on it as long as possible. The optimization must, in effect, reorganize the computation in such a way as to obtain optimal granularity relationships among entities involved in the computation.

It is not unlikely that problems similar to these arise in syntactic and semantic processing. The cognitive architecture may well also provide means of gaining processing speed by storing commonly used intermediate results, and by optimizing its calculations for locality. Unlike most mathematical calculations, parsing can be an extremely indeterminate process. Not only are the grammatical rules often ambiguous, but the inputs to the parser — heard or seen words — are rarely recognized with any degree of certainty. In speech processing this problem is in fact overwhelming, and it has been the major stumbling block for automatic recognition of continuous speech,

but even written text is recognized far less well than our conscious awareness leads us to believe. This means each word in the input stream can trigger several accesses to the lexicon, and it is easy to show that the number of possible candidates for syntactic analysis grows exponentially in the length of the sentence. Consequently, most accesses to the lexicon, needed to check whether various recognition candidates can match hypothesized grammatical structures, are wasted effort and do not contribute to the analysis.

It makes sense that the lexicon of a language would optimize its representational structures for locality, trying to chunk lexical data in such a way that accessing one part of a lexical item automatically fetches other data that are likely to be needed immediately, but not everything that is available in the lexicon. During phonological word segmentation, for example, it would be useless bring in conceptual data attached to the many phonological forms that need to be checked against possible, alternative phonetic interpretations of the sound stream, but it would also be inefficient to fetch the phoneme representations in the access form one-by-one rather than in a single block. I shall later (chapter 3) examine evidence that the lexicon possesses a representational stratum I call Indexical Structure, which supplies phonological representations of word segments along with a few pieces of information like inflectional class and lexical category to word recognition, but without any of the information needed for syntactic or semantic analysis.

In addition to chunking, the lexicon may also provide computational optimizations by storing many data required for efficient recognition in a kind of pre-computed form, like the entries in a table of logarithms. The lexical forms posited in LFG — essentially lexical predicates with argument structure, which will be discussed in detail in chapter 5 — may in fact be structures that provide just the right amount of information required for building syntactic hypotheses. As parts of the logical structure of the lexicon, lexical forms may often be redundant to the extent that they can be computed via mapping relations from deeper levels of lexical structure. But these computations may be too expensive to carry out in real time during syntactic analysis. Since they are small structures, a great many lexical forms can be fetched from memory, tried out against the syntactic structure, and discarded, without incurring large processing costs. For reasons of control efficiency, the lexicon may provide representations that the logic of the lexicon does not require.

When a new, derived word is presented to syntax, its meaning, argument structure, and other attributes must be available from some computation over its recognizable morphological constituents — otherwise it would not be used. However, since it is new, a lexical form cannot be available for it in the lexicon. The lack of a lexical form makes syntactic analysis difficult, because conceptual structure and mapping rules must be consulted to infer the word's category and subcategorization requirements. Thus, it is to be expected that the linguistic system will make every effort in the future to avoid these costs by buffering the results of the derivation along with a lexical form, and by making them available the next time the word is used.

0.2 Unresolved Problems with Derivation in LFG

Curiously, the basic insight has been around for some time that an account of derivation in a unification framework requires a control component capable of assigning the results of derivation to a lexical buffer and of steering lexical insertion during online sentence analysis. Nevertheless, this insight appears to have been lost in debates about the nature of the overall architecture of the formal system, debates that have largely ended in a complete separation of the lexicon with its derivational component from the syntactic component. Lexical rules have been relegated to the status of redundancy relations that, in implementations, function simply as macro processors to expand compactly formulated lexical entries into sets of related items.

Early in the history of formal generative linguistics, a number of proposals were made to set word formation on an equal footing with the generative process postulated for sentence formation, but in an influential paper on nominal derivation in English, N. Chomsky [Cho70] was able to persuade most generative linguists that the creation of words is not something that takes place in the way that sentences are created, since word formation — at least in English — is often too unsystematic to be accounted for in syntax. In formal linguistics, and especially in computational linguistics, this view has prevailed.

In addition, computationally oriented linguists realized that even the most transparent and seemingly unproblematic derivational relations among lexical items, such as the relation between the active and the passive form of a verb, are computationally so expensive that they could not be considered a part of sentence grammar. For this reason, lexicalist grammars have excluded even these relatively unproblematic morphological relations from syntax, in order to avoid the enormous computational complexity that transformational accounts of these relations introduced. In distinguishing their "lexical-functional" grammar from the transformational model in *The Mental Representation of Grammatical Relations* [Bre82b], Bresnan and Kaplan emphasized the barrier between syntactic and lexical processes, and in many formulations they seem to have isolated the lexicon altogether from the generative dimension of syntax.

Nevertheless, a close reading of the "Introduction" to this seminal collection reveals that they had considered the possibility that not only diachronic but also synchronic, spontaneous derivations might take place "in the lexicon", allowing speakers to make up new words undeliberately and to incorporate these newly formed words simultaneously, without explicit definition, into grammatical sentences. In one passage, they described the result of deriving a previously unknown passive verb form as being buffered and made immediately available to syntactic analysis [Bre82b, xxxi ff]. In this way, passivization would be allowed for any appropriate verb, whether or not it already had a known passive form, without resorting to a transformation; and once the passive form had been placed in the lexical buffer, it could be fetched repeatedly without incurring the computational cost of repeated derivation. Their terminology, however, did not identify the lexical buffer as an entity distinct from the lexicon itself, which seems to have led to endless misunderstandings and to a wide-spread impression that LFG and similar unification-oriented models of grammar are unable to give a formal account of spontaneous word formation. Across research traditions, lexicalism has been identified with a conception of the lexicon as a static set of well-formed words that do not participate in the creative, spontaneous introduction of new forms

typical of syntax.

Some current computational architectures for natural language processing do provide mechanisms for expressing derivational relationships among words as a form of feature inheritance in the lexicon e.g, [KJ94], [KuMvBF93], [Kun95], but this approach alone cannot provide a full account of the myriad ways in which word formation appears to interact with and determine both sentence structure and meaning. It is, in fact, morphology's complex interaction with syntax that has made derivational morphology one of the most intensely studied areas of non-computational, theoretical linguistics in the last decade. From many far-ranging morphological studies has emerged a picture of word and sentence processing in which the boundary between the two appears much more fluid than it did twenty years ago. Attention has been drawn, for example, to the ways in which the deep-seated semantic argument structures of words directly constrain syntactic structures, as well as to ways in which freely combining elements of the sentence seem to be able to merge into lexically fixed entities. While much of the data documenting these processes has been drawn from relatively little-studied languages, many of the important phenomena can be found in modern European languages like Italian, French, and German, as well. It remains to be seen whether these data can be incorporated into a convincing computational model; the present study represents an attempt on a very small fragment of the available data.

0.3 Overview of Further Chapters

The following chapters ultimately will attempt to save as much as possible for the unification-based approach to linguistic modeling, but they will show that a realistic model of how complex words are integrated into sentence structure cannot avoid making statements about temporal sequences and priorities. As good software engineering practice dictates, however, the first step in constructing a computer program must be to forget all known computational strategies and look, as naively as possible, at the data; then known computational strategies can gradually be introduced as methods for organizing and understanding the data. Finally, if all goes well, an implementation can be specified that will model those characteristics of the data that seem most important, using formal devices that are appropriate, in a fashion that is both comprehensible and a good mirror of the modeled reality.

Chapter 1 discusses in further detail why a computational model of word formation needs to be concerned with a wide range of domain data. It summarizes a number of findings that have been propagated in the software engineering literature about the craft of program construction, and it draws some consequences from these for the craft of constructing cognitive models. One finding is that the implementation of an application program nearly always returns an unexpectedly large contribution to the requirements document that is meant to precede it. An application program is, of course, different from a cognitive model in important ways, but evidence for this kind of feedback in application programming suggests that specifying and implementing a cognitive model may be more than a last step in verifying a model — it may in fact be an important step in constructing the model itself, if tests of the model's validity have been correctly identified. Validity is something that is harder to grasp than formal

requirements, however. It requires an open-ended appreciation of the true nature of the task a program or a model is meant to fulfill and careful attention to the requirements of all "stakeholders" or relevant disciplines that may have something to say about the model.

Chapter 2 turns to the first of three important stakeholders or sources of data about the set of words used in a language. The Zipf-Mandelbrot distribution of word frequencies in large text corpora shows that the vocabulary of a large corpus is unbounded and is a function of the size of the corpus. This strongly suggests that the rules of word formation, as postulated by linguists, are not merely well-formedness conditions describing possible words in a language. To some extent the unbounded Zipf-Mandelbrot distribution can be explained by constraints on word structure in the light of coding theory. However, the most important statistical result is that word frequency distributions in large corpora are not uniform for all morphological classes in the lexicon. Instead, unbounded vocabulary growth takes place in narrowly defined subsets of the lexicon. These subsets can be identified as complex words that can be generated by concatenating affixes and bases according to certain patterns, at rates that can be characterized by stochastic combinatory processes. Like the rules of sentence syntax, word formation rules specify generative processes used in real time in language production and understanding. However, these rules are not applied as freely as the rules of syntax; the statistical data indicate that further factors must tightly constrain their application. These findings indicate that stochastic production systems could provide an important specification formalism for a model of the lexicon.

Chapter 3 presents psycholinguistic data showing, on the other hand, that many apparently complex words may in fact not be analyzed or produced at all, but may instead be stored as simplex forms, with the result that their psychologically 'implemented' representations appear to differ from their formal descriptions in the stochastic production system. Data about the relative speeds of critical operations in word and sentence processing often fail, however, to indicate clearly whether certain words are stored as atomic entities or are recognized by identifying their constituent parts. This ambiguity, which was long taken to reflect a weakness of the experimental methods, can now be seen as evidence that many words have "dual representations", existing simultaneously as atomic symbols and as structures of symbols. The data make it clear that many inflected words as well as derived stems can have dual representations. This hypothesis appears to require a radical revision to any notion of the lexicon as a simple array of atomic items. It suggests that lexical access must be modeled by both substitution and by a parallel but prioritized analytical procedure.

Chapter 4 summarizes some data from the third and most important stakeholder, namely structuralist linguistics. These data give further evidence concerning dual representation of lexical items, and they tend to support the Lexical Integrity Hypothesis, which states that on the structural level the rules which describe word structure are not an integral part of the syntactic system. Other, semantic data show, however, that many complex words may not require representation as atomic entities, since their syntactic and semantic properties can be analyzed from their constituents. On-line analysis and

generation of complex words would, of course, save much space in the lexicon, but in many cases it appears that a complex word is used in ways that are not compatible with the results obtainable from an on-line analysis. Furthermore, in many cases the derivation of words makes reference to complex conceptual knowledge. If the access to such knowledge is computationally slow or expensive, it may be expected to inhibit regular use of on-line analysis for complex words, and there is in fact some very tentative evidence from Italian that beyond a certain threshold of complexity, computable words are not readily accepted or remembered by users of a language. For complex words below this threshold, the same cost effect appears to favor storage as atomic lexical items rather than repeated re-analysis, as is also shown by psychological evidence in chapter 3.

Chapter 5 examines some formal tools that can be used to give a symbolic account of word formation. Findings like those reported in the previous chapters have led to various speculative and informal models of word processing, one of which describes how a kind of "lexical connectionism", based on a model of spreading activation of phonological representations, can be applied to some important problems in morphology and historical linguistics. While computational neural networks suggest themselves as a possible means of formalizing connectionist models, to date the results obtained with linguistic neural simulations (e.g., Rumelhart & McClelland's program for learning past tense inflections of English verbs, [RM86]) have been disappointing, perhaps because the training procedures currently available are inadequate. But it can be shown that, at least in principle, a model partially equivalent to the symbolically oriented, spreading activation account can be defined in terms of statistically weighted production rules and transducers. Since well-studied techniques exist for implementing linguistic specifications in terms of these formalisms, such a re-formulation seems to offer a promising alternative to the spreading activation account, and it suggests ways to formulate a perspicuous, symbolic model of word-formation.

This chapter introduces three symbolic formalisms for specifying word structure. At the phonological (actually orthographic) level, it is shown that lemmatization of both regular and irregular forms can be specified with context-sensitive rules implemented as stochastically weighted tranducers mediating between level-ordered word segments and the lexical access forms. At the level of word structure, stochastically weighted production rules can explain the statistical growth described by the Zipf-Mandelbrot equation. Finally, the semantics of complex words can be described to some extent in a formalism based on the lambda calculus. Together, these declarative formalisms provide much of the apparatus necessary for specifying a cognitively plausible model of word formation. They do not however, allow one to explain the data relating to dual representation of lexical items, nor are they fully adequate in many cases for deriving the meanings of many complex words.

Chapter 6 discusses the problem of word segmentation. A symbolic account of word formation presupposes that its input data are available as unambiguous symbolic atoms, but the actual input to the language processing system does not present this kind of structure in any obvious way. Identifying the boundaries between units of word analysis is in fact computationally difficult, and its complexity imposes far-reaching

constraints on the amount of run-time analysis the cognitive system can perform. To minimize the number of hyphotheses that word segmentation must consider, it is suggested that word recognition takes advantage of a kind of hierarhical structure in the lexicon similar to one predicted by coding theory and which, in chapter 2, is proposed as an explanation for the statistical structure of large corpora.

The KLU model assumes that word analysis proceeds in two stages. In the first stage, candidate word segments are identified on the basis of an indexical structure, which is simply a compilation of all minimal morpheme-like units found in the lexicon. This list makes no distinction between morphemes and pseudo-morphemes, and it therefore leads segmentation to draw segment boundaries both where genuine morphemic boundaries occur, as in *re-analysis* and where only a superficial similarity to morphemic structure is present, as in *reproach*. However, the search for possible segments is constrained by a level-ordered structure similar to one that has been proposed in lexical phonology, so that segment hypotheses from the periphery of the word need be compared only to that small subset of the lexical index that contains inflectional affixes. Segmentation then works its way into the center of the word, checking for possible matches to derivational affixes, stems, and finally roots, but without much regard for morphological compatibility. This approach thus avoids producing all possible naive segmentations, but it requires a second stage of word analysis to identify genuine morpheme boundaries and morphological structure. It is, however, able to explain the psychological evidence for run-time segmentation of pseudo-prefixes, as well as linguistic bracketing paradoxes. Further, it is argued that it offers significant computational advantages for the kinds of ambiguous input data that word recognition encounters in auditory or visual processing, by comparison with approaches that integrate segmentation and morphological analysis.

Chapter 7 describes the main features of the KLU implementation. It discusses briefly the important algorithms and data structures used to specify the KLU model, as well as a factorization of lexicon and syntax into further modules. The KLU parser is based on a well-known, conventional strategy for implementing Lexical Functional Grammars, but it contains two innovations. First is the separation of the syntactic system into two synchronized, parallel processes, one for sentence-level, syntactic analysis, and a second for word-level, morphotactic analysis. A second innovation is a complex procedure for lexical insertion. Instead of describing lexical insertion as simple term replacement, the KLU model defines a prioritized set of steps that can be invoked to satisfy a request for a terminal symbol of the syntax. An indispensable part of this procedure is an error generation mechanism and an error handler that are responsible for making a non-existing word available to syntax by generating its lexical representation ‘on the fly’ in a lexical buffer when it cannot be found in the static lexicon. Repeated analysis of the same word can lead to lexicalization.

The KLU system compiles language descriptions written in an modified Lexical Functional Grammar formalism to elaborated Definite Clause Grammars. Allomorphy is treated simultaneously with word segmentation, in line with the level-ordering hypothesis of Kiparsky and others. Segmentation is steered by tables of possible word segments compiled from the lexicon, and word segments are matched to lexical segments via orthographic (pseudo-phonological) rules. These rules must be translated by

hand to state tables for non-deterministic transducers. Semantic formulas and conceptual facts are notated as logical relations in Horn clauses. The implementation techniques also include graph unification and a limited form of thread synchronization, as well the error handling mechanisms and a form of default symbol name inheritance, used for modularization, offered in Prolog II+. All of these are based on standard textbook implementation techniques and are not discussed here in any detail. Some alternative strategies for implementing unification-based parsers that were considered for the implementation are discussed briefly. The KLU system alone, apart from any language description, comprises more than 12,000 lines of Prolog code, which would make a detailed program description tedious.

Simple program traces illustrate how the KLU implementation deals with complex words, as well as a representative case of a sentence containing a spontaneously derived word.

Chapter 8 summarizes the important results obtained with the KLU model as well as some issues that need to be explored by further research. It also draws some implications of the model for the implementation of large-scale natural language processing systems.

Chapter 1

Requirements Engineering and Computational Modeling

The present study represents an attempt to construct a computational model, called KLU, of how words are represented and processed in human cognition. Like other kinds of computational model, the KLU model simulates the mapping of inputs to outputs, and it tries to account for observable properties of processes involved in the mapping. Because current research has achieved no unanimous agreement about how cognitive faculties like the one that underlies language should best be modeled, or indeed about the nature of the objects to be modeled, the present study also addresses, on a deeper level, the question of how the tools and methods of computer science can help to resolve questions posed in linguistics and cognitive science. It is thus also a study of the nature of computational models and how they can best be developed and specified.

When we look from a distance at the task of such a study, it may strike us that asking how words are structured and processed is a pursuit of commonplace and in some way 'unscientific' knowledge, for unlike the objects of most other kinds of scientific investigation, the facts about word formation in language — or at least in one language — appear to be already known by almost everyone. Although the facts about a given language may have interesting linguistic and historical backgrounds that are not common knowledge, the empirical facts as such are readily available. As speakers of a language, we know which words are in our language, their meanings and patterns of use, and we know what combinations of words are or are not sentences, as well as what they do and do not mean. Furthermore, there are evidently no deeper 'causes' of our linguistic knowledge that science could find in other, yet undiscovered, non-linguistic facts. Of course, we could not use words if our physiology gave us no means to perceive or produce them, and we may be unable to answer questions about our language when we are drunk or have suffered head injuries, but this does not mean that our linguistic knowledge is part of our physical make-up, directly deducible from the composition of our blood or from the structure of our brains, any more than the behavior of a computer program can be derived from the circuit topology of the machine on which it runs. A native speaker's implicit knowledge of a language (what the generative tradition in linguistics, e. g., [Cho65], has called "competence") can be made explicit in the form of rules and lexical descriptions, but these are never bet-

ter than contingent, empirical generalizations over observed linguistic behavior; they do not represent linguistic knowledge per se, which can ultimately be validated only by introspection or by eliciting introspective judgments from other qualified speakers. Introspective givenness is, moreover, a quality that language shares with other kinds of implicit cognitive knowledge, such as knowledge about smells, sounds, visible and tactile forms. It is knowledge of given mental entities that can be described in terms of judgments about symbolic relations. To put a polemical point on the matter: these judgments concern purely mental objects and relations; they can be described and classified, but their causes cannot be discovered in terms of, or reduced to, other physical facts.

There is, to be sure, a long tradition of debate about the ontological status of purely mental entities, such as grammatical categories or mathematical objects, but it is fair to say that the philosophical tradition has uniformly recognized their status as different from the status of physical phenomena available to the senses. Psychologists and linguists influenced by behaviorism have tended to see mental states and operations as hypotheses lacking empirical reality and as therefore undesirable in the vocabulary of scientific explanation, and they have thus been skeptical of linguistic explanations in terms of operations on 'invisible' mental symbols. In attempting in this study to give a symbolic account of implicit linguistic knowledge, I shall be siding with the realist position in this debate, which has been willing to grant mental entities the status of "really existing", and thus of being proper objects of scientific study. While non-subjective, empirical sources of information about language like vocabulary statistics, speech errors, reaction times to linguistic stimuli, etc., may reveal much about the processing of language, they cannot provide the proper data for an empirical study of a particular language. Linguistic knowledge itself is epistemologically prior to the data such studies provide,[1] and such peripheral data cannot be considered the objects of a study of language per se. Simply put, were we able to show by observation of neuron activity that the agreement of subjects with their verbs is enforced by a set of neuron connections in the Broca region of someone's brain, it would be an interesting fact, but a physiological disturbance to those same neurons in someone's head, leading to speech errors, would never convince us that the linguistic facts about agreement were any different.

Thus, as some theoretical linguists have argued, linguistic science and linguistic models have a character distinct from the physicalist models of physiological psychology or biology, one that is concerned with the organization but not with the processing of the elements of language. A. Zwicky, for example, maintains that a linguist's contribution to understanding word-formation ought to be solely

> to find a logic connecting semantics and phonology at the word level. This logic should be linguistically appropriate, that is, its instances should be interpretable as claims about what particular people know about (certain aspects of) their languages. ...

[1] There are no neo-Cartesian, dualistic assumptions lurking here; the underlying philosophical position is that of formalism, as expounded by J. A. Fodor [Fod81] and followed in much recent work in symbolically-oriented cognitive psychology, e.g., G. Miller & P. Johnson-Laid [MJL76]. In essence, the formalist position regards mental phenomena as being similar to software, "real" in the sense that they exist in, but are not determinate properties of, material mechanisms.

> But I make no claims about how the processing of language proceeds in real time, using real memories. My concern is with abstract inventories of expression types, properties of expressions, and generalizations about those properties; it makes no sense to say that such generalizations "take time." ... Linguistic theories are well advised to attend to psycholinguistics and the theory of computation ... but they cannot expect that questions about the way languages are organized will be reducible to problems already solved in one of these other domains. [Zwi92, 327-8]

On this line of thought the scientific modeling of an area of cognition like language seems to require a methodology different from that commonly employed for modeling physical objects, one that seeks to uncover and formulate abstract structures governing the organization of data and logical relations, rather than the internal workings of the organism in which cognition takes place. Because its primary task is to analyze and simulate implicit, acquired human knowledge and behavior, such a methodology might well more closely resemble that of software development than the strategies of sciences that are primarily concerned with finding causal foundations of phenomena.

Regrettably, cognitive science, in the form practiced by linguists and by computer scientists, has rarely cast more than a glance at the methods of large-scale software engineering. Twenty and thirty years ago, developing computer programs to simulate human cognitive functions was a formidable challenge, the main question being whether cognitive functions could be simulated at all; whether the simulations were in a larger sense accurate mirrors of mental reality was outside the purview of the undertaking. The projects now being undertaken in the framework of artificial intelligence are, however, slowly merging with the development of generalized, 'intelligent' software applications. As cognitive simulations are scaled up to meet the needs of useful applications, they, too, will have to be validated against the large-scale demands of systems meant to model and carry out human functions. The demands for generality, extensibility, validity and correctness placed on application programs are thus an unavoidable issue for artificial intelligence as well, since cognitive simulations will need to serve both as the carriers of theory and as prototypes for useful applications.

One would hope that well-constructed computational models of cognitive faculties could have more than practical value, however. Unfortunately, their role in the construction of cognitive theory is often seen as simply that of extending the complex details of a formal theory to a lower, implementational level. Since at present the physiological mechanisms that implement cognitive faculties like language are not understood in detail, a full computational reconstruction of their functioning is not possible, and any implementation of a linguistic theory can thus only be based on a more or less arbitrarily chosen underlying machine. According to the Church-Turing hypothesis, if an "effective procedure" can be computed on one machine, it can be computed on any other general-purpose machine; from a computational perspective, there is no single apparatus that is exclusively appropriate for implementing a computational procedure [LP81, chapt. 5]. Presumably, the physiological system which implements the human language faculty will also belong to the set of Turing-equivalent mechanisms, or some subset thereof, and we have no choice but to hope that there will exist a translation from an arbitrarily chosen implementation to the real physiological mechanisms, if and when these are uncovered (and if we are capable of understanding them). If this hope is justified, then — so it has been argued — for cognitive linguistics

there is no pressing need to understand the characteristics of the specific, physiological machine that implements the human language faculty, since like any computational implementation, the physiological mechanisms are also arbitrary. It is, in fact, is quite astonishing that the physiological apparatus most obviously associated with language, namely our ears and vocal tract, appear to be unnecessary for the acquisition of language as such, for deaf children will spontaneously acquire sign language [Pin95, 36], and written language can be acquired by persons having no ability to hear or speak. The specific computational implementation, whether in living bodies or in computers, is thus apparently a consequence, rather than a premise of the abstract linguistic system or of any cognitive theory that describes that system, which means that, as linguists like Zwicky have argued, the linguistic system cannot be studied by studying the machine in which it is implemented.

This rather seductive line of argumentation appears to be supported not only by linguistic arguments; the picture of the software development process often promulgated in texts and in industrial guidelines on software engineering suggests a similar conclusion regarding the status of implementations. On this view implementation is the filling in of details left unspecified in a more abstract formal analysis of what a system or application requires, and it has become usual to distinguish three successive phases in the development of a computer program, usually identified as

- requirements analysis
- formal specification
- implementation

From the perspective of such guidelines, software engineering at first appears rather cut-and-dry, like cabinetry. Someone who needs a function 'computerized' looks, figuratively speaking, at the corners and niches of an organization, finds a place where the solution would fit in well, measures the available space, and puts it all together in a list of REQUIREMENTS to be presented to a designer. Drawing on experience in building previous cabinets, the designer then draws up a SPECIFICATION. This leads to plans and drawings that can be given to assistants to cut and assemble into an IMPLEMENTATION of the design, and if the process is carried out competently, the cabinet will fit the space and fulfill the function originally desired. Needless to say, in many details a different but, so to speak, Turing-equivalent cabinet would fulfill the requirements just as well; the requirements are not linked to a specific implementation, but they do entail a narrowly circumscribed set of solutions. This may be fine for the practical exploitation of a cognitive theory, but if this view of software development is correct, then it would appear unlikely that an implementation study could return anything of substance to a cognitive theory. The role of computational simulation appears to be confined to checking the implemented theory for consistency and completeness; and this is indeed the way it is seen by many linguists and cognitive scientists, e.g., [And92].

1.1 Interaction of Requirements with Implementation

Numerous studies of the software development process in large-scale industrial settings, e.g., [Boe86], [PTA94], have shown that in real-world software development, the progress from requirements to specification to implementation is rarely linear and rarely as uni-directional as non-computational linguists and cognitive scientists are inclined to see it. More often it is the case that requirements are elaborated and even altered, and become more precise only as their full implications become evident in the evolving implementation. Only with the passage of time does a computer program together with its surrounding documentation become an authoritative representation of the function it was meant to perform and of the shared knowledge it implements. Thus, in fact, the development of a program typically feeds information back into the formulation of its requirements, making implicit knowledge explicit and sometimes even bringing new knowledge into being that was not available nor even anticipated during the initial formulation of requirements, as S. Shieber [Shi85] has observed in linguistic theory building (cf. also [May96a]).

This has been taken for granted in some kinds of artificial intelligence work, where the paradigm of "exploratory programming" has been developed and even raised to the status of a method in its own right, as a way of discovering possible cognitive mechanisms, unprejudiced by prior theoretical analysis of what those mechanisms might be. The paradigm of unconstrained, exploratory programming, with simultaneous development of specification and implementation, and without much explicit requirements analysis, has in fact become one of the main contributions of artificial intelligence to computer science, and it has made us aware that a great many things can be computed that at one time seemed unspecifiable and explainable only as the inscrutable operations of a mysterious mental substance. But it is increasingly recognized that unbridled exploratory programming more often than not merely leads to dead ends. To be sure, some kinds of 'exploration' are in fact unavoidable even in the development of a wide range of more conventional software applications, owing to the very complex and inadequately expressible structure of human knowledge in general. But, as D. Marr has argued, cognitive theory consists of much more than sets of behavior–mimicking algorithms [Mar82, 75]. Given a simplified abstraction of some cognitive domain, defined by a sharply restricted set of test cases, a program can produce required results or correctly correlate sets of input and output data while still containing serious errors, incorrect assumptions or incorrect representations of the domain. Indeed, it has been a commonly repeated experience in cognitively oriented artificial intelligence that algorithms which apparently solve a restricted version of a problem often break down as soon as some apparently trivial, simplifying assumption is removed. A well-known case in point is an algorithm that was devised by D. Waltz for identifying blocks represented in a two-dimensional line drawing. Waltz solved the puzzle with an algorithm that identified the three-dimensional objects on the basis of an ingenious logic of line intersections. Cognitively speaking, however, he solved a puzzle that in the human visual system is apparently never posed, for as as soon as the objects in the scene were allowed to have curved surfaces, the algorithm failed, and it has since proved impossible to generalize it to recognize such objects (cf. [Win84], [HMT89], [Mar82, 226]).

Quite everyday computer applications do, however, often take over relatively so-

phisticated tasks from human beings, such as locating hotel rooms, finding airline connections, or managing inventory, in ways that can be seamlessly extended and integrated into other skilled tasks. There is no reason to think that the modeling performed by such applications is achieved by methods essentially different from those used in the construction of cognitive theories or that ought to be used in artificial intelligence. In planning the design of a conventional application program, the initial goal is typically to simulate some human task well enough that the computer can take over from people who have performed the task previously. In the case of a contracted project, a customer will typically draw up an initial set of goals or requirements for a more or less familiar function in a business; in other cases marketing specialists will try to identify a function that a program might usefully take over. However, because the function has been carried out by human beings, it can often be described only in general terms. Like linguistic and other forms of cognitive knowledge, the specific knowledge required to perform such a task has no explicit formulation, and because many persons may be involved, it will often not be evident where to find the most reliable informants. The more useful the function, the more difficult it often is to identify exactly how it is performed. In the most complex cases, like expert systems and real-time control of complex processes, the program may need to formalize abstract knowledge from arcane and possibly unrelated specialties.

Thus, in developing computer applications the task of "eliciting requirements" is itself often merely exploratory at first, and it is similar in many ways to linguistic field work. The things the program must do are rarely fully known when the coding begins, and often the basic, organizing structures upon which it must be built cannot be stated. The list of requirements will have to be elaborated and modified repeatedly in the course of development, using the application's customer as a kind of informant. An essential part of the development work consists in discovering, with increasing precision, what criteria must be met in order to meet the goal of emulating the required human function. As requirements are translated into formal rules and procedures, criteria for testing whether the program realizes each requirement must also be defined. Meeting these formal criteria is, however, rarely the ultimate goal of the development effort; rather, the goal is reached when a customer or the market is satisfied that the software can replace a human function with sufficient reliability and flexibility to be safe, useful and profitable. For this reason, highly developed applications nearly always contain large amounts of very specific but unformalized structure and knowledge that reflect the structure of the world that they operate in, and it is often the case that this knowledge is made explicitly available for the first time in the form of the computer program and its documentation. Even the programming of 'unintelligent' application software thus very often has the open-ended character of simulating human intelligence at a number of levels, and studies of the ways in which it is typically developed may tell us much about how programming can lead to genuine understanding of a cognitive domain rather than mere mimicry of selected behavioral patterns.

1.2 Exorcising the 'Homunculi'

Viewing the development of a cognitive simulation as a kind of application development, we find many points of similarity, but there are also two principal areas of di-

vergence. Most obviously, while a cognitive simulation must in some sense also "take over" a human function, as a research tool, it has no customer and no market to satisfy, and the criteria for its success or failure must be formulated somewhat differently. A second difference, which may be more apparent than real, is that typical application programs carry out functions that are already thoroughly understood, at least tacitly. While a great deal of work may be necessary to find out, for example, how an organization manages its book-keeping or how a pilot lands an airplane, there are people to ask and documents to consult, and above all, there is already at least a partial definition of what the development task is and what an effective solution for "computerizing" the book-keeping or landing procedure would be. There may not be an explicit theory available to describe the task, but the persons who carry it out can usually explain what they do and why. For a cognitive domain these things may simply not be given. The givens are cognitive judgments as well as behavioral or other empirical data, but these do not in themselves constitute a set of requirements for a theory nor for a simulation. They are merely inputs to the activity of theory construction.

I shall argue here that the building of theories, in particular of cognitive linguistic theories, is a process of first observing and describing phenomena, guessing at possible, vague explanations, and then, in J. A. Fodor's words [Fod81, chapt. 2], step by step driving out the "homunculi" in the explanatory description — those little men invoked at various points in cognitive theories to impose complex constraints or carry out unexplained operations, in the form of as yet unexplained principles, factors, agents, etc. The progress from description of a phenomenon to its explanation is, on this view, nothing other than the step-by-step clarification of unexplained terms in the initial explanatory description. One fairly convincing way of "driving out the homunculi" in the construction of a theory is to follow a set of steps similar to those involved in creating a conventional computer application. The goal of a computational simulation is then not necessarily to replace an actual human work function within an organization, but to turn an imprecisely formulated description of a cognitive function into one that is maximally explicit. To do this, the computational model can replace obscure terms, i. e., individual "homunculi" or human-like functions in the theory, with specifications and finally with implementations of these functions or at least with approximations of them.

This process has similarities to one described by N. Wirth in his well-known paper "Program Development by Stepwise Refinement" [Wir71]. Wirth takes as his programming example a small but non-trivial task, that of finding a way to line up eight queens on a chess board so that no queen can capture another. Looking at the problem from a slightly different angle than Wirth did, let us assume we have found a proficient chess player who can quickly find a solution to this problem, but who cannot tell us in much detail how the solution is found. As chess idiots, we would like to learn something about the chess expert's thought processes — in other words, we want to create a model of the cognitive processes by which the chess expert finds his solutions. Of course, the chess player can offer us a sort of theory: he or she will tell us the rules for how queens capture other pieces, about ways the queens could be arranged on the chess board, and about arrangements that prima facie don't need to be considered. But then the chess expert's explanation probably becomes murky; somehow all of the arrangements are eliminated in which one queen threatens another, so that a mutually non-threatening pattern emerges.

How can the chess expert do it? Stated in purely declarative or logical terms, the problem is to find, in all the combinatorily possible arrangements of eight queens A, the one arrangement a where no queen is threatened. What is required is:

$$a \in A \mid \neg\text{queen-is-threatend}(a).$$

But it is clear that the solution does not come instantly, and the chess expert suggests that there is some mental searching going on. Taking account of this procedural aspect, Wirth proposes that the solution might be found by (I have simplified his notation slightly)

repeat
  generate the next arrangement *a* of eight queens
until no-queen-is-threatened(*a*) OR no more possible arrangements

This procedural decomposition partitions the problem into two parts. The outer structure of the solution, the **repeat** loop, is an algorithmic control structure that can be translated directly to some machine formalism. Viewed as a component of a nascent theory, the control structure is thoroughly transparent in the sense that it requires no further decomposition or explanation. Of course, we could ask at another level how the chess player is able to repeat anything until a condition is fulfilled, but this is a question we could also pose for climbing stairs, counting sheep, etc. The control structure is not something that is peculiar to thinking about chess boards, and it is not likely to be an important part of a theory of chess expertise. In any case, the existence of a well-understood machine implementation of the **repeat** construct can be regarded as sufficient proof that there are no appeals to mysterious or obscure principles in this part of the theory.

The embedded operations pose some problems, however. For the second condition, we probably can test the condition *no-queen-is-threatened(*a*)* by examining all pairs of queens against the rules for capture, although this remains to be worked out in detail. For this we need to specify how to generate all of the possible arrangements of eight queens and how to keep track of them. It is not hard to show that this, too, can be done in principle, but there are 232 possible arrangements of eight queens. Does the chess expert really set out to test all of these?

In saying that there are some arrangements that do not need to be considered, our chess expert actually suggests a different strategy, a kind of 'preselection', that generates as candidates for testing only those arrangements that meet some unknown criterion, namely *possibly-non-threatening(a)*. Limiting the search in this way to a much smaller number of arrangements, a theory of how the eight queens problem is solved by the chess expert might be

repeat
  generate the next arrangement *a* of eight queens
  require: possibly-non-threatening(*a*)
until no-queen-is-threatened(*a*) OR no more possible arrangements

This simply says that the search is somehow constrained to a much smaller set of cases and shifts the burden of explanation to a more plausible but at the same time

much more mysterious procedure for generating preselected board arrangements. But it does give us a chance, not previously available, of identifying a strategy that does not require stupidly testing an enormous number of cases.

However, it also makes the theory appear to rely on a "homunculus" in the form of a little chess expert within the description of how the big chess expert finds a solution. Without some assurance that *possibly-non-threatening(a)* can be carried out without appealing to a complex but otherwise unexplained operation, the whole theory is not worth much, for as it stands, it might well be circular: it says that the chess expert, whose behavior we are trying to explain, finds possible non-threatening arrangements simply by successively testing the arrangements suggested by an embedded expert. Showing that this particular homunculus can finally be driven out of the simulation has become a standard programming exercise in recursive search (e.g., [Bra86, 108ff], [Par90, 15-18]); whether the real chess expert does it the same way as the recursive algorithms in programming textbooks does not matter at first. What matters is that in an otherwise perspicuous theory of eight-queens-solving, this opaque term *can* be expanded to a form that leaves no circularity and no further mysteries.

Many standard software development methods are in essence elaborations of the basic strategy described by Wirth, although Wirth's partition of the problem into transparent and opaque terms changes in the contemporary methods according to whether the programming paradigm emphasizes control structures, modules, objects, or logical relationships. When filled out with a limited set of precise and known-correct ways of expanding the opaque terms, the method of stepwise refinement can help ensure correctness and completeness of an implementation [Lin94]. What the strategy of stepwise refinement does not entirely account for, however, is how the initial partition is obtained.

In this example I have implied that some sort of search is carried out for "transparent" explanatory structures that can be imposed on the problem, like the **repeat** ... **until** control structure, allowing the developer to factor out a remainder of opaque terms that can be further analyzed. But what if the overall structure is hard to nail down, as it often is in the domain of cognitive theory? An alternative strategy to Wirth's "top-down" approach (as it was frequently called in the 1970s) is the "bottom-up" method conceived by E. Dijkstra for operating system development. Rather than worrying about the overall structure of a difficult problem at first, it sometimes makes sense to concentrate on small, better understood sub-problems that can be solved independently of the whole solution. If one adheres to the rule that bottom level terms are not allowed to make references to higher level terms in the proposed solution, one obtains a collection of functional elements that can be randomly combined as one needs, just like the primitive operations of a computing machine. Taken together, the low-level operations present a "virtual machine" in terms of which the next higher level of the problem can be solved [Dij68]. Further, the structure of the low-level operations will impose constraints on the structure of the next higher level in the form of interface requirements. If an operation requires certain functional arguments and returns certain results, the higher levels of the theory will be constrained to combine it with just those other operations that can furnish its required arguments and can use its results.

Thus, complementary to functional decomposition (high-level structure transparent, lower levels opaque), there is also a justification for partitioning a problem in such a way that the top-level terms are allowed to remain unexplained or obscure while

the lower-level terms are being clarified: instead of working in the trenches, Fodor's homunculi can occupy executive positions in the theory until they are exorcised. Instead of decomposition, theory can be built by incremental functional composition. It is not impossible that a single, large-scale structure of the theory may emerge from a purely combinatorial search for the unique structure in which all of the lower-level terms can be combined (e.g., where for each function only a uniquely defined set of lower-level functions could supply the necessary input). Most often, however, the search through all possible combinations of primitive operations will be prohibitively tedious, and some amount of structure will also have to be imposed at the top, allowing the low-level terms merely to constrain the possible high-level structures. Where can these high-level structures come from, when they are not evident in the data to be explained?

Some authors have suggested that, in programming, our cognitive abilities restrict us to using a sharply limited number of top-level patterns or "programming discourse principles" [Sol86]. These principles provide us with a restricted set of templates from which intelligible program structures can be built. In developing cognitive or linguistic theories, the task is often so enormous that finding any intelligible initial factorization at all is a reasonable achievement, but software development imposes an additional, disciplining constraint on theory construction. If the project is to be carried through to a working program, the top-level patterns chosen and the resulting factorization must be made in such a way that all opaque terms can, at some later point, be resolved into transparent, implementable structures. An initial factorization conceived as a sketch for a computer implementation must choose opaque terms that bear some resemblance to items in the programmer's repertoire of solvable problems. Programmers typically have a store of known patterns at their disposal, libraries of data structures and algorithms for dealing with stereotypical problems like copying, searching, sorting, logical inference, etc. (cf. [KP74]). For constructing a cognitive or linguistic theory, it may seem quite reasonable to introduce apparently non-mysterious operations like "find a similar form", "find the shortest path in the network", or "find the logical extension", and indeed, much progress in linguistic theory has depended on the research community's willingness to accept such terms as carrying explanatory content. But to the extent that there are often no known easy implementations of such operations, they may only hinder the expansion of the description into a completely implementable structure, and they leave the theory open to the charge of obscurity. Nevertheless, virtually every categorical assertion about "What Computers Can't Do" [Dre86] has proved to be little more than a bet against time, and being too restrictive with regard to what is seen as implementable can also block the development of a theory. The fine art of theory construction lies perhaps in the ability to steer a clear coarse between the dangers of these two shoals.

1.3 Strategies for Requirements Analysis

As in the development of cognitive theories, in application projects the initial stages of requirements analysis can be surprisingly unstructured and unsystematic. Typically it is necessary to formulate goals and take account of various constraints on the development work, such as expected reliability and the physical limitations of the tar-

get hardware. These requirements are sometimes divided roughly into behavioral and non-behavioral constraints (e. g., [Dav93, chapt. 2]. While the non-behavioral requirements are typically atomic constraints ("The program cannot occupy more than 2 MB of RAM") and are often easy to identify, the behavioral requirements can be arbitrarily complex. Many investigators have found that, for typical commercial applications, users generally are unable to state at first how a desired application should behave in detail [PTA94]; hence, the process of eliciting initial behavioral requirements often has an exploratory, investigative character. In this phase the developer must identify all of the "stakeholders" in a project, that is, persons and organizational entities who may have knowledge or perspectives that may need to be taken account of in a complete list of requirements. Of course, users may discuss their needs from many perspectives and with varying terminology; the distinguishing characteristic of an informal requirements analysis, however, is that the program developer must find matches between users' descriptions of what they want and the developer's repertoire of known-implementable abstractions. In simpler cases, the developer can ask questions leading to descriptions of sequential actions or scenarios; more complex cases, arising in the construction of expert systems, have required eliciting rules and complexly weighted heuristics. In all cases, the exploration must lead to a set of behavioral descriptions that can be reformulated unambiguously in formalisms for which implementable specifications can be written. Frequently used formalisms are based on finite-state-automata in various forms, Petri nets, decision tables, logical formalisms and formalized objects and actors. For natural language simulations, formal specifications in terms of automata and transducers (or higher-order production systems, such as context-free and context-sensitive grammars) as well as logical systems suggest themselves and are in common use, and they have been used extensively as specification formalisms for KLU and many similar systems; KLU has also borrowed from the object model (e.g., [Jac92]) to capture certain requirements for modularity and data visibility.

1.4 Correctness and Validity

Once a set of detailed requirements has been established, software developers need to find ways of ensuring that the implementation will in fact fulfill them. Drawing on the experience of constructing a large clinical data base, B. Blum [Blu94] has divided the development process into two concentric areas. The outer area describes the progress from informal requirements to their implementation, a process that is successful only if the implementation can be VALIDATED against the needs expressed or implicit in the informal requirements and in the situation in which the program is used. The inner area comprises the development process from formal requirements to VERIFYING the implementation for correctness against the formal requirements. While the outer area subsumes the inner, in the sense that it is impossible to progress from informal requirements to a valid system without at least going through the motions of drawing up formal requirements and checking their correctness, neither process requires the success of the other. "Verification and validation are independent processes, and one may have an incorrect product that corresponds to the organization's needs or an invalid product that is correct with respect to its formal model" [Blu94, 84]. Thus, a CORRECT program faithfully implements its formal requirements, but only a VALID

program meets the requirements of its users.

Programming exercises (what Blum classifies as problem-oriented programming), like "For a radius *r* larger than zero, compute the surface area of a sphere of radius *r*" can typically be solved by referring only to the inner area, where the requirement is simply a mathematically precise formulation of a problem that must be transformed to an equivalent (in the sense of correct and complete), executable machine formalism. Even in the case of a trivial exercise, however, there is unavoidably an outer area of requirements, as well, where a host of other needs could be considered: How precise and how fast must the calculation be? What about rounding errors? Does the program help the user if the input data are unreadable? How these questions are resolved can determine whether the program is not only correct but also valid in the context of some larger need. But in contrast to the development of the inner area solution, for the outer area there are no well-defined ways of ensuring that the requirements are met. While requirements analysis and implementation for strongly formal domains like mathematical modeling or machine control can take place largely within the inner area, other domains like data bases and information systems (Blum calls these "product oriented programming") demand much more attention to the outer area. The inner area process of creating and verifying formal requirements does not disappear, but it is surrounded by a much larger process of identifying informal and possibly ambiguous requirements that cannot be stated in formally verifiable terms. Within the inner area it is possible to ensure the program's correctness using one or another form of stepwise refinement — expanding the formal specification by means of known solutions to each of the initially opaque terms in the specification — and it is possible to verify it by comparing its behavior against that predicted by the formal requirements, as well as by deductive inference. Whether formal inferences about its behavior can amount to a "proof" of correctness is doubtful, and in fact the entire analogy between program verification and mathematical proof fails at important points, as J. Barwise [Bar89] has argued. But validation, in the outer area of informal requirements, is in any case much more difficult than verification. Often the users themselves, rather than the informal requirements, must be consulted, simply because all of their needs will not have been entirely known when the informal requirements were drawn up, and these needs may in fact remain unknowable until situations arise that bring them to the fore. In a sense, the informal requirements are limitless and unknowable because they include a potentially limitless number of situations in which the program might be used, and they require an accordingly limitless number of appropriate responses.

1.5 Transformations from the Specification

The previous discussion has described the path from a cognitive theory to its specification and implementation in a computational model, and it has identified criteria for judging the model's adequacy. It has suggested that implementation is a process of working out and filling in details of the theory, but it may also have created an unwanted impression that the large-scale form of the computational model will thus necessarily mirror the large-scale form of the theory.

In the literature on linguistics and artificial intelligence, the hypothesis of a more or less explicit correspondence between symbolic, rule-based generalizations about cog-

nitive processes and the underlying cognitive mechanisms has been called the "symbolic approach" to cognitive modeling (paradigmatically in [Win84]). Some behaviorist (or connectionist) psychologists have objected to such a straightforward mapping from the constructs of a linguistic theory to the cognitive system, seeing in the symbolic approach a kind of projection of the theorists' constructs into a domain that may operate on quite different principles. D. E. Rumelhart and J. L. McClelland have argued that "the mechanisms that process language and make judgments of grammaticality are constructed in such a way that their performance is characterizable by rules, but [...] the rules themselves are not written in explicit form anywhere in the mechanism" [RM86, 217]. They plead for a kind of model in which linguistic behavior is produced by complex, statistically weighted connections between the linguistic inputs and outputs, and in which no explicit correspondence is postulated between their underlying psychological model and the rules and symbolic operations employed in formal linguistic theories. The constructs used in a linguistic theory are indeed likely to be chosen more for their clarity, simplicity and communicative convenience than for their accurate representation of underlying psychological mechanisms. The "connectionist approach" of Rumelhart & McClelland arguably frees a linguistic model based on neural networks from a kind of PROJECTIVE FALLACY, inherent in the symbolic approach, of presuming that the underlying model ought to function by using the same symbols and operations employed in the linguistic theory. By divorcing descriptive linguistic models from their cognitive implementations, the connectionist approach avoids the projective fallacy, and it allows theoretical generalizations to be seen as perspicuous behavioral characterizations of a mechanism that probably functions much differently from the constructs of the theory. For these reasons many cognitive scientists and even some linguists have found in it an appealing alternative to computational modeling based on the symbolic approach.

Computational modeling based on the symbolic approach is less guilty of the projective fallacy than its critics would lead one to believe, however. An important part of software engineering in fact has to do with recognizing and bridging the large gap that exists between the symbolic structures and operations that are clear, simple and convenient for human beings to use, and those symbols and operations that can be implemented easily with electronic (or other kinds of) devices. The executable code that runs on a computer often has little recognizable formal similarity to the computer program from which it was compiled. After an optimizing compilation, it typically will have undergone many simplifications, elaborations and restructurings before being presented to the machine for execution. A valid and correct implementation of a specification thus usually exists in the machine as a TRANSFORMATION of the specification, possibly in a form in which even the structural outline of the specification is no longer visible.[2] It is thus misleading to characterize a symbolic theory as one that inevitably projects the structure and operations of the theory into the underlying cognitive realm, and it is important to understand that the path from the formalization of a theory to its implementation does not necessarily preserve the superficial structures

[2] A computer program that has gone through a highly optimizing compiler often has a form so different from its source that it cannot be de-compiled back into an understandable program. Thus, it would be incorrect to accept the objection of Rumelhart & McCelland that in computational models built on the symbolic approach the rules of a symbolic theory are necessarily "written in explicit form" in their computational implementation.

of the theory, nor does it require that the symbols and rules be explicitly present in the implementation. There are several ways in which the constructs of a formal theory or specification can become invisible in a computational implementation; all can be characterized as transformations that adapt a specification to the capabilities and limitations of the underlying mechanism while preserving functional equivalence, and thus formal correctness, between the two levels. Under some circumstances transformations may not preserve validity, a difficulty to which I shall return shortly.

1.5.1 Transformations via a Virtual Machine

One type of transformation in which operations of a theory or specification become invisible at lower levels results from their implementation via a "virtual machine" that is built on top of the actual implementing mechanism. It is often the case that the operations defined as primitives in a given formal specification (e.g., the operation of graph unification in a unification grammar) will have no corresponding representation on the machine or in the programming language chosen for the implementation. Even simple operations such as set union or string concatenation are not always available in a given programming language, which means that a further stage of functional decomposition is often necessary, not of the specification, but of the formalism in which the specification is stated. This typically leads to a library of formal primitives that is not directly subordinated to any particular component of the specification, but rather becomes a part of the virtual machine (the physical machine plus its libraries of primitive operations) which is the assumed basis for the implementation. The virtual machine provides those operations postulated in the formal specification that do not exist in the machine on which it is implemented.

Rather than constructing a virtual machine by hand for the abstract operations postulated in a theory, an implementation can turn to a high-level programming language that provides many of the required primitive operations 'ready-made' and in efficient implementations. For natural language simulations, for example, the implementation language Prolog offers many primitive operations like term and string unification, as well as a form of logical inference, that map easily to the formalisms used by linguists. In Prolog, the operation of unification appears intuitively simple, and it expresses easily the generalizations postulated in some unification theories of grammar. To the programmer it then appears as if the implementation were being evaluated by a powerful machine actually capable of carrying out the unification and logical operations used in the formalization of the theory, i.e., as if unification operations belonged to the underlying mechanism. Unification is, however, an inherently complex operation. It has no counterpart on a von Neumann-style computer, and it must be realized by a procedure typically expressed in many pages of simpler operations [Boi88, 15-20]. Thus the powerful but abstract operation of unification, used in the formalization of the grammar, is not in fact "projected" into the underlying machine; it is instead transformed into operations that have a different kind of complexity suited to the underlying mechanism.

1.5.2 Optimizing Transformations

Many operations specified in an abstract formalism cannot be translated automatically to efficient code for a virtual machine of whatever kind, and consequently programmers must often adapt them to the capabilities of the underlying virtual and real machines, which can ultimately give an implementation a form much different from its formal specification. H. Partsch has called the process of reorganizing a formal specification in such ways as to give it an efficient machine representation TRANSFORMATIONAL IMPLEMENTATION [Par90]. Typical of all transformations is that rather than merely filling out the specification, as in stepwise refinement, they alter its structure in some way while maintaining its operational semantics. One simple transformation consists in substituting a more or less equivalent data structure or algorithm for the one specified, e.g., implementing a structure defined as a set by using a linked list instead, or choosing a more efficient control structure than the one specified (e.g., implementing a **while** loop as an **until** loop). Transformations that can lead to large differences between the form of a specification and the form of the implementation include INTERLEAVING, where entities from various, separate parts of the formal specification are realized in a single piece of program code, and DELOCALIZATION, where the opposite occurs, i.e., a single entity of the specification is spread among two or more data structures, procedures or modules of the implementation [BM98]. Highly interactive, mouse-and-menu driven programs, for example, often have an implementation structure dominated by the user interface, even though their formal specifications are dominated by a structure defining the problem they are meant to solve. In addition, all of the optimization techniques known to compiler writers, such as storing intermediate results, eliminating recursion and redundancy, partial evaluation (constant folding), and the like (cf. [ASU86]) can, on a larger scale, be used by application programmers in order to better adapt the specification to the virtual machine on which it is implemented.

A commonly used kind of transformation involves computing intermediate results beforehand (“constant folding”) or at least storing (“caching”) them if they are computed repeatedly at run time. For example, if a retail price is specified as the result of multiplying wholesale price times the sum of markup plus value-added tax, the programmer might add the markup and the value-added tax to obtain a single multiplier, and use this single, simplified term in the implementation to compute retail from wholesale price. This transformation discards information that was present in the specification, namely, that the simplified value is derived from and is thus dependent on two other values, with the result that in the implementation the multiplier appears to be a primitive term. A naive reconstruction of the specification from the program code would then show retail price as being a simple multiple of wholesale price. It would be formally correct, but it would be misleading, since it would not document how the mysterious multiplier came into being.

The transformation steps used by programmers are thus not always unique, one-to-one mappings. While preserving behavioral equivalence between the formal specification and the implementation, they are not always reversible, and thus they can alter or destroy information in such a way that it becomes impossible to reconstruct the specification from the implementation. Most obviously, it is the explicit outer-level requirements that can be lost in transformations. For example, in a natural language

application, the informal requirements for a sentence-generating system would typically specify that the length and embedding of a sentence is not bounded, and accordingly the inner-level requirements for sentence structure — the sentence grammar — would be specified in terms of a system of recursive rewrite rules that allow sentences of infinite length with an unbounded number of embedded clauses. But if a particular grammar contains no embedded recursions (recursions occurring neither at the beginning nor at the end of the expansion of the rule head), an alert programmer would probably recognize the grammar as specifying a regular language that can be implemented with a finite-state automaton, which gives a fully equivalent but more compact and more efficient realization of the specified grammar. The recursive form of the original specification, however, allows for other, more general grammars than can be realized with the automaton, and this idea, not used but least recognized and provided for in the production rule formalism of the specification, gets lost and could not be recovered from the finite-state implementation, which appears to allow only grammars for regular languages. The implementation thus fulfills the inner-level requirements but it no longer realizes the outer-level requirement of allowing, in principle, embedded clauses in a language. If we think of a linguistic system as a set of outer and inner-level requirements concerning ways utterances can be formed and used in a language, this discrepancy suggests that data about the psychological 'implementation' of the linguistic system, such as are obtained in experimental studies of word recognition and association, probably also stem from transformed versions of the linguistic system rather than the system itself. The psychological data therefore do not necessarily correctly represent the structure of the linguistic system from which they have been derived.

Transformations can therefore be possible sources of confusion in trying to understand the relationship between the specification of a system and its implementation, whether in a computer or as a part of human cognition. A transformation that will be of particular interest for the KLU model is called "memo-ization" [Par90, 280-282] or "caching", which is frequently used to avoid re-computing intermediate results that remain the same but are needed repeatedly in the course of a computation. A formal specification is often most clear and compact when it implicitly requires such recomputations, but an implementation that avoids them via caching can be enormously more efficient. Indeed, a central theme of the present study will be that the formal, declarative description of word formation inferred by structuralist linguists solely from linguistic data contains just such a specification of the word formation system, one that deliberately ignores when and where results are computed. The evolution of the cognitive "implementation" of word formation appears to have discovered the efficiency of caching, but it has also found advantages in permitting a kind of "cache incoherency", a kind of error in the implementation of the cache transformation that arises when the intermediate results are allowed to differ from the values from which they are derived. This may well be the source of puzzling discrepancies between the structures of complex words and the behavior of the linguistic system using them, as we shall see later (sections 3.7 and 4.5).

1.6 Handling Errors and Exceptions

The ability to deal with inputs and situations not described by a program's specification is often the single and most important difference between a valid implementation and one that is merely correct. Large applications often devote a large portion of their code to catching user errors, checking consistency of data, monitoring access privileges and the like. Yet even the software engineering literature often underplays the important role these functions serve in creating genuinely useful and usable implementations.

In cognition, much of our ability to respond to unexpected situations probably must also be attributed to a kind of general, supervisory error-handling intelligence that oversees the operation of individual cognitive systems. Climbing stairs, for example, is a function that we carry out without much conscious attention, but when we stumble, suddenly other systems are called to aid in order to find what went wrong and search for a solution. Once the problem has been found, we restore our normal posture, place a foot on the next step, and reactivate the largely autonomous stair-climbing program.

Modern operating systems and programming languages provide primitives for detecting errors and exceptions to well-formedness constraints on operations ranging from the simplest machine instructions to complex operations like building up network connections. At each level of the system, an operation that violates its constraints triggers a mechanism that aborts the computation, takes note of where the error occurred and what was going on, and then transfers control to an error handling program that tries to figure out what to do next. The error handler is often assigned to a specific, local operation or piece of code, called its SCOPE, and it embodies a kind of local intelligence about things that can go wrong in its area of responsibility. If it cannot deal with an error, it generates a new error, possibly adding additional information, and calls a higher and more general error handler that is responsible for more general sorts of problems, as shown in Fig. 1.1. Higher level error handlers have increasingly broad scope and are responsible for greater numbers of program modules. They must accordingly embody progressively higher levels of general error-solving 'intelligence'. Typically, the last error handler belongs to the operating system, which often appeals to the user of the computer to figure out what must be done.

A similar strategy for responding to events external to a highly interactive program is often employed in the object-oriented paradigm. An object receiving notification of an event tries to respond to it, but if it cannot, it passes the notification up to an object of the class from which it was derived. Ultimately, if no intermediate object in the "chain of command" can respond, the notification lands in a single, most general "super-class" from which the lower classes are derived.

A computational cognitive model cannot, of course, hope to simulate both its chosen domain and the kind of general, overriding intelligence that intervenes to correct things gone greatly wrong; indeed, ambitions in this direction would look more like science fiction than cognitive science. What is worth some attention, however, is how a given cognitive domain identifies its local, specific errors and what it may undertake locally to restart a calculation after an error has occurred. Local error handling is in fact likely to play a very important role in the processing of morphologically complex words, as later chapters will show.

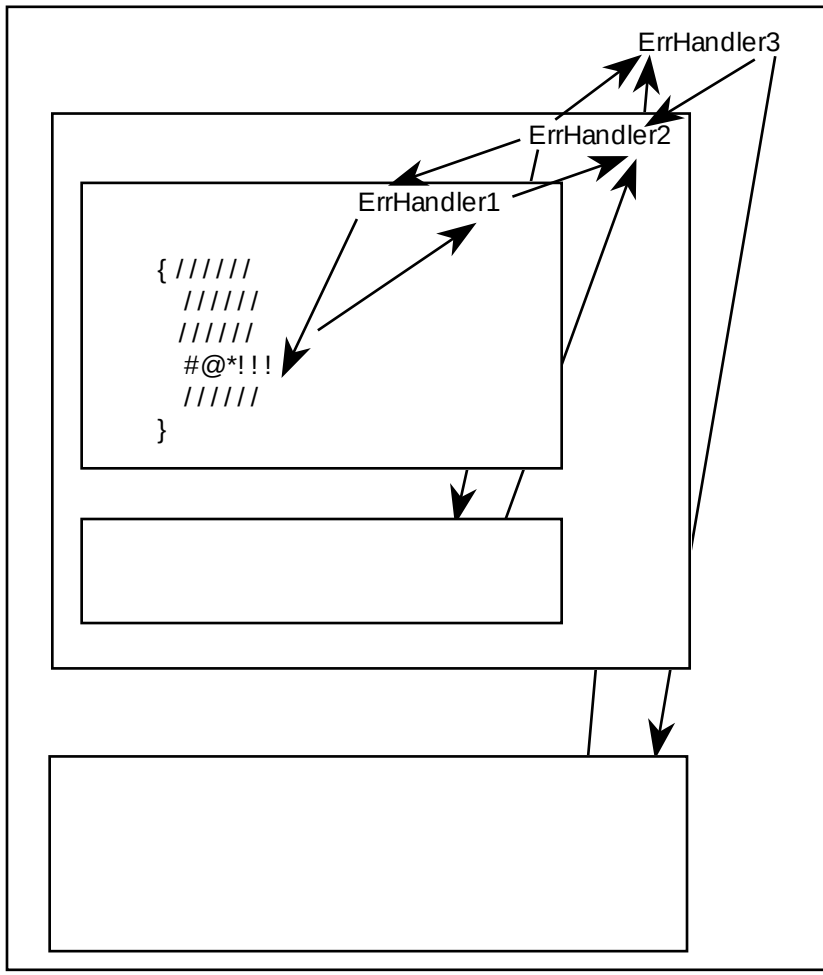


Figure 1.1: Hierarchy of Error Handlers in a Computational System. Each nested box represents a program module lying in the scope of its own error handler.

1.7 Requirements in Linguistic Modeling

A property of natural language that has much occupied theoretical linguists since the appearance of Chomsky's *Syntactic Structures* [Cho57] is its limitless creativity, which imposes a requirement that a finite description of a language must account for a potentially infinite number of sentences. Because of the myriad ways in which even the simplest sentences of a natural language can interact with other sentences and with situations in which they are used, it is often hard to factor particular linguistic phenomena neatly into independent theoretical structures. Constructing a theory by any kind of "top-down", stepwise refinement requires, however, that certain parts of the description be, so to speak, nailed down as known and as requiring no further explanation, as for the **repeat** ... **until** structure invoked for the eight queens 'theory' in section 1.2. It is in fact, however, difficult to find clear partitions of linguistic phenomena into perspicuous control structures, functions or objects, on one hand, and obscure entities for further investigation on the other hand, and much of the haggling among theoretical linguists takes place over precisely the question of the correct top-level structure of the

linguistic system.

Approaching the requirements for a natural language simulation "bottom-up" in Dijkstra's sense is often easier, for in many cases formal requirements for a low-level subsystem, such as noun phrase inflection (agreement among determiner, noun, adjectives) can be formulated very precisely and without recourse to further, unexplained elements in the theory. Conventional dictionaries can be seen as an extremely useful, bottom-up factorization of knowledge about a particular language. They contain a great deal of low-level detail about components of the linguistic system, with little commitment to presuppositions about the high-level structure of the language processor. Textbook grammars, on the other hand, state requirements for other, possibly higher-level sections of the language system. Dictionaries and grammars can thus be seen, from the perspective of software engineering, as informal requirements that can be used to draw up formal specifications of a language system, in the form, say, of lexical entries and syntax rules. Once these formal specifications are implemented, the system can be verified against them.

Nevertheless, grammars and dictionaries leave a great many terms only vaguely specified, and they often avoid making categorical, systematic statements about the exact rules governing the grammaticality or meanings of expressions that can be encountered in the language they describe. Where uncertainties remain, the linguist's only recourse is to the intuitions of the community of speakers of the language. Since these intuitions ultimately encompass the unbounded, "creative" dimension of language use, they cannot be given even an informal specification. Because they have the form of implicit knowledge, they must be relegated to the outer area of requirements, as conditions that can be validated only by the judgments of native speakers in the course of testing the simulation.

It is these outer-level, informal requirements that pose the most formidable challenges for any kind of linguistic simulation. While we may be able to find out from a descriptive grammar what are the basic, formal conditions for well-formedness of words (and sentences), and while we can easily translate many of these to some appropriate, verifiable formalism, a VALID language simulation will also need to include in its requirements the ability to simulate the many intuitive judgments that competent speakers of a language can make concerning previously unencountered words and sentences. We find, for example, that speakers can often produce finely differentiated judgments concerning well-formedness, giving responses like, "that is not a word in my language", "that might be a word but I can't tell you what it means", or "that is not a word I have heard, but I can certainly tell you what it might mean". Such judgments must be reproduced by a simulation, even when it is not possible to specify exactly how they are made. They thus impose outer-level requirements that can be tested — and thus validated. Outer-level requirements may not lead directly to formal specifications, but they impose strong constraints on the top-down architecture, for any architecture that does not compute these intermediate results or that computes judgments different from those of competent speakers cannot be regarded as empirically valid, even though it may produce final outputs that are correct in terms of the inner-level requirements.

Other linguistic observations, as we shall see, impose constraints of a still more global character, requiring, for example, that the lexicon be extensible in certain areas but not in others, that meanings be derived in certain ways, or that some linguistic productions should occur more frequently than others. Statistical and experimental

observations concerning how words are produced and recognized can also serve as a secondary source of requirements, insofar as they impose conditions which any plausible model cannot violate. Although such outer-level constraints cannot be translated directly into formalizations in the form of automata, production systems, logical relations, etc., this does not mean that they are not implementable or are outside the realm of possible simulations — just as requirements for application programs concerning clarity, response time and ease-of-use resist formal specification but not concrete implementation. Such non-behavioral requirements do often suggest how tradeoffs should be made in the choice of large-scale data or processing structures, and they can affect the choice of algorithms. The difficulty of describing the steps by which specific program structures are derived from specific non-behavioral requirements has to do with the difficulty of describing theory construction in general. It should be seen as evidence that this aspect of computational simulation genuinely belongs to the construction of a scientific theory of the object simulated and is not merely implementational hack-work.

1.8 Requirements for a Model of Word Formation

Drawing some consequences from the above deliberations, it seems reasonable to try to give some careful thought to how the wide range of linguistic, statistical and experimental data now available concerning the cognitive processing of words can be integrated into a set of requirements for a computational simulation. The larger and publicly documented language processing systems (e.g., Eime88, Alsh92, Kapl96) have by and large concentrated on implementing "inner-area", or "problem-solving" (in the words of Blum [Blu94]) requirements that have arisen in the frameworks of formal linguistic theories like Head Driven Phrase Structure Grammar (HSPG) [PS87] or Lexical Functional Grammar (LFG) [Bre82b], and they have been specified in related formalisms. Although these systems have succeeded in showing that various formally specified linguistic descriptions can be made to 'work' in a computational implementation, they have ignored many of the outer-area, informal requirements deriving from statistical and experimental data about language production and recognition, as well as many well-differentiated linguistic judgments about word formation, particularly semantic judgments.

A second, related development is the large body of detailed findings from psycholinguistics about the relative speeds of critical operations in word and sentence processing. Although these data contribute little to a description of a linguistic system per se, they can be used to exclude many data structures and algorithms for implementations of the linguistic system that would be formally correct, in the sense discussed above, but invalid with respect to outer-level timing requirements. For many cognitive scientists such outer-level requirements are not merely implementational issues; they are seen as crucial conditions for a psychologically adequate model of linguistic processing, and they provide the essential data upon which recent psychologically-oriented theories of word-processing have been built, as we shall see in chapter 3. As G. Rickheit and H. Strohner observe:

> Erst die Hinzunahme des Zeitbedarfs ermöglicht Schlüsse darüber, inwieweit das Modell die beim menschenlichen Sprachverstehen tatsächlich ablaufenden

> Prozesse simuliert oder aber mit Hilfe ganz anderer Prozesse möglicherweise zu denselben Resultaten gelangt. Stimmt der Zeitbedarf der menschlichen mit dem der maschinellen Sprachverarbeitung überein, so kann vermutet werden, daß auch die Operationen, die dem Zeitbedarf zugrunde liegen, übereinstimmen [RS93, 123] [3]

Temporal behavior, however, is just one of many outer-level requirements that can be imposed on a computational model. The model can also be validated against other constraints, such as statistical behavior and the various intermediate intuitions regarding well-formedness and meaning that were mentioned above. Certainly a significant advantage of an implementable, computational model is that it tolerates a degree of complexity that cannot be mastered in a pencil and paper sketch of a model [JC87, 10]. If the computational model can be shown to be both formally correct and valid with respect to its outer-level requirements, it may open new and insightful perspectives on its domain, despite its complexity. I hope the analysis and results presented in the following chapters will show that computational implementation does, in fact, offer a way of developing rather than simply testing theories of cognition.

[3] Not until timing requirements are taken into account can one draw conclusions about the extent to which the model simulates processes involved in human language comprehension or perhaps achieves the same results by the aid of completely different processes. If the human processing time corresponds to the time required by the model, the suspicion is justified that the operations requiring the time correspond.

Chapter 2

Corpus Statistics and Word Formation

Oftentimes important outer-level requirements for a computer application can be derived from the statistical behavior of a program's inputs and its expected outputs. Knowing with what relative frequencies various kinds of input and output data can be expected, a designer can structure an application so that it deals optimally with its most frequently occurring tasks and does not waste resources on those that it encounters less often. Such adjustments are often necessary even when the application's formal specification involves no probabilistic operations. The inner-level, behavioral requirements of most large data-base systems, for example, can be specified purely in terms of logical inferences from facts stored in the data base, without statistical weighting of the data or inference rules. However, given that not every form of possible query is equally probable, the data will be typically be stored in special formats (e.g., using special keys and index tables) that allow fast responses to the most frequent queries. For infrequent queries, the additional cost of such optimizations may not be justified, with the result that infrequently occurring queries bring about considerable on-line searching and computation and are therefore answered only after some delay. As a further consequence, some infrequently asked-for kinds of information may be made inaccessible to most users of the data base, even when the information is logically derivable from the stored data. Thus the query optimizations may ultimately affect the apparent logical structure of the system.

Assuming that the designer of a data-base system will try to find the most economical structure for a particular distribution of query types, information about the frequencies of various kinds of queries can often help outsiders make confident guesses about the internal organization of a data base. If, as is frequently assumed, the lexicon of a language is a mental data base of static, record-like structures defined by a language, then it, too, may well be subject to processing constraints to which the cognitive apparatus may be specially adapted. Statistics about the kinds and distribution of accesses to which the mental lexicon must respond should thus give us some help in trying to 'reverse-engineer' its internal structure. In particular, statistical relationships that have come to be known as the laws of Zipf and Mandelbrot appear to define statistical distributions which the lexicon, as a kind of mental data base, probably reflects in its internal structure.

These statistical relations at one time attracted considerable interest because they seemed to promise methods for constructing mathematical models of language production and understanding by purely computational means. This promise has never been fulfilled, and much of the early work on linguistic statistics has now fallen into obscurity. In fact, the "laws" that were discovered in these studies have turned out to be at best rough generalizations about language production, and they are no longer of great interest in and of themselves. What makes them interesting for a study of word formation, however, is that they reveal important constraints on the internal structure of the lexicon. They also offer a key to the somewhat vague and puzzling notion of morphological productivity. In fact, we shall see that some of the results from linguistic statistics provide a well-founded basis for characterizing productivity in the framework of a formal computational grammar, although this was not the goal of most of the early studies.

2.1 Zipf's Law

Although others had noticed it before him, it is G. K. Zipf's 1935 discovery of an inverse relationship between the frequency of a word and its frequency-based rank-ordering which has given it the name Zipf's law [Zip65] and has inspired the investigation of similar relationships in a wide range of sociological data beyond linguistics. As recounted in [Rap82], Zipf found relationships similar to the rank-ordering of words in a wide variety of non-linguistic data, and he suspected that he had stumbled on a relation that was not specific to language or communication but was a structuring principle of social interaction in general. For example, rank-ordered lists of populations of cities or of frequencies with which books are borrowed from a library will show distributions similar to those Zipf found for words in texts. In a list of randomly chosen cities rank-ordered by increasing city size, there will be an inverse relation between the size of a city and its rank. The city ranked first will be of largest size, and for progressively smaller cities, the rank number times the size will tend to be constant. If a library ranks its books by popularity and lends 30 books of the highest popularity per day, at rank 10 in the popularity list, Zipf's law predicts that only three per day will be lent.

2.1.1 The Zipf Equation

Applied to texts, Zipf's law states that, if each different lexical item (lemma) found in a text is ranked according to the number of times it appears, a roughly inverse relationship between rank and frequency will be found. To clarify both this relationship and some of the terminology used to describe it, consider some of the statistics that might be gathered from a very short corpus, such as

(5) The cited examples illustrate the derivational function. In the second example, the object illustrates that the functional object can have the second semantic and . . . [73 further words]

Statistical data about this corpus are presented in Table 2.1 and summarized as a histogram in Fig. 2.1. The text has a length or EXTENT N of 97 words. It thus contains

97 word TOKENS or instances of words, like *the*, *has*, *objects*, *illustrate*, etc., many of which recur, sometimes in variant forms. Recurring tokens are counted as instances of the same type. A TYPE is a distinct lexical item in a dictionary or a similar listing of words distinguished by meaning, form or other attributes. Among the words in this sample, 17 distinct types have appeared in the text (5) and are listed with statistics drawn from the eintire 97-word corpus in the table. These 17 types are actually present as 27 distinct word forms, but some of the words have been LEMMATIZED or counted as variants of the same type. Thus the variant tokens *is*, *are*, *was*, and *were*, all contribute to the statistic for the single type *is*, just as *object* and *objects* are counted as a instances of a single type.

Table 2.1: Tokens and Types in a Small Corpus, N = 124

RANK (i)	TYPE (LEMMA)	TOKEN COUNT N_i	VARIANTS
1	the	18	-
2	is	15	are, was, were
3	that	11	-
4	have	9	has
5	and	8	-
6	for	7	-
7	like	5	-
8	object	5	objects
9	in	4	-
10	cite	4	cites, cited
11	can	3	could
12	example	2	examples
13	illustrate	2	illustrates
14	second	1	-
15	derivational	1	-
16	functional	1	-
17	semantic	1	-

The FREQUENCY N_i of each type i is given by the number of times it or one of its variant forms appears in the corpus. In the table, types have been assigned to RANKS i in order of decreasing frequency (higher rank means lower frequency). Note that in the higher ranks some of the frequencies are the same, e.g., *like* and *object* each occur five times, but they occupy different ranks (7 and 8). Types having the same frequency are assigned to successive ranks randomly, and in a typical corpus the highest ranks (large i) will all have the same frequency, namely $N_i = 1$. If the type frequencies found in a particular corpus are generalized to an arbitrarily large sample, one can speak of the probability P_i of finding a type of rank i; this is simply its frequency divided by the extent, or size, of the idealized, indefinitely large corpus.

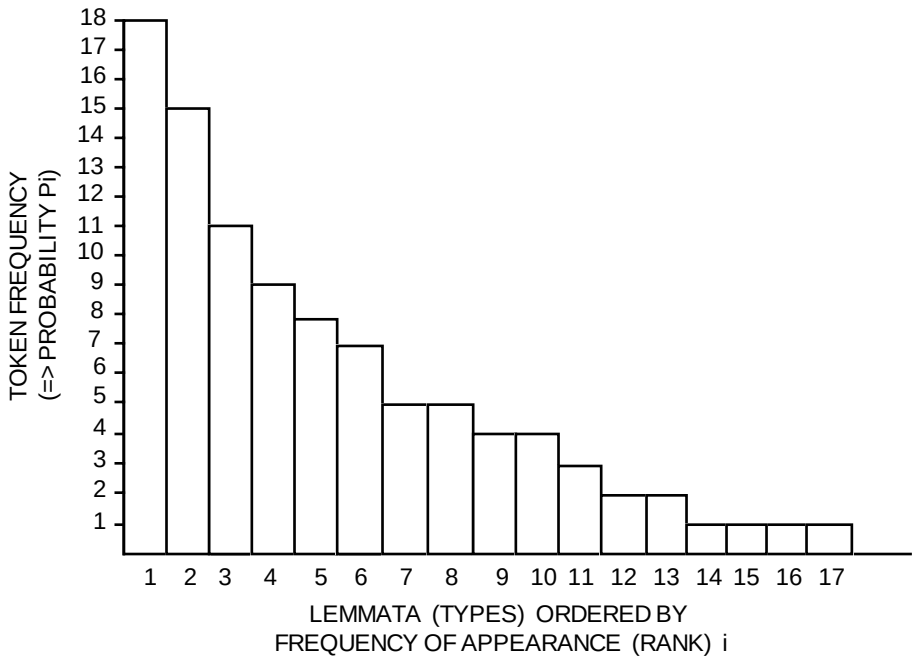


Figure 2.1: Histogram of a Ranked Frequency Distribution (Data from Table 2.1).

Once the size of a corpus reaches a few thousand words, the relationship between rank-ordered types and the corresponding token frequencies typically begins to take on a form like that of Fig. 2.1. Zipf found that the shape of this RANK-ORDERED DISTRIBUTION could be generalized to an inverse relationship between rank and token frequency: low rank means high frequency, or more generally, the product of rank and associated token frequency is a constant C. Viewing an indefinitely large sample, the probability P_i of finding a type of rank i will be inversely proportional to its rank. To get a close fit to his observed distributions, Zipf raised the value of the rank i to a power γ having a value close to one, giving the relation commonly cited as Zipf's law, (6).

(6) $$P_i i^{\gamma} = C$$

2.1.2 Zipf's Law in Practice

As with any kind of statistical investigation, the applicability of Zipf's law to texts depends crucially on the way in which the data are defined and gathered. For one, the sample size of a text has been found to influence the distribution, with very large and very small samples often deviating sharply from the expected pattern. This discrepancy inspired important revisions of Zipf's law, to which I shall return shortly. Another source of discrepancies lies in the lemmatization rules chosen for identifying the entities to be counted and from ambiguities inherent in the notion of word. Words can be both abstract signs in the linguistic system (elements of Saussure's *langue*), as well as strings or enunciations that may have been modified for the purposes of discourse (*parole*). For many purposes it is the lemmas, or elements of the more abstract set of signs constituting *langue*, that appear relevant for characterizing the vocabulary of a

text, rather than their individual morphological variants. For example, to follow how a text shifts its focus from one subject area to another, we might want to gather statistics about the objects it mentions. But in English we cannot gather statistics about a text's use of nominal concepts (a subset of the *langue*) by blindly counting occurrences of nominal character strings, for this would accumulate a separate, unrelated statistic for each singular noun, its plural, and its possessive. Moreover, since the frequencies of morphological variants of an item are likely to differ widely, their statistics will cross-contaminate the statistical distribution, i. e., a high-frequency lexical item (e.g., the French present tense verb *aime*) occurring in an infrequent morphological variant (e.g., past historic, second person plural *aimâtes*) will contaminate the low-frequency end of the distribution, suggesting incorrectly that a rare concept has been named in the text. Further problems arise in correctly counting homographs, strings that are identical in form but not in meaning, and with collocations, or groups of words whose meaning is unrelated to the meanings of the words they comprise.

To count the use of such forms correctly, a textual analysis must pay attention to context and to knowledge that cannot be easily programmed into an automatic counting procedure. To illustrate the effect differing lemmatization rules can have, C. Muller reports data from three studies of word frequencies in Racine's *Phedre* [Mul77, 12]. These reveal discrepancies of as much as nine percent according to the ways in which words were lemmatized, as shown in Table 2.2. The first sampling reflects a lemmati-

Table 2.2: Tokens and Types in Racine's *Phedre*, from [Mul77]

SAMPLING	N (WORD COUNT)	V(OCABULARY)
1	13,075	1,609
2	14,217	1,653
3	14,415	1,642

zation carried out by hand, while samplings 2 and 3 were carried out automatically. It is apparent that the manual evaluation has led to both a smaller number of word instances N (TOKENS) and a smaller vocabulary or number of LEXICAL TYPES V, probably because the human lemmatizer could better identify collocations and idiomatic expressions as single tokens and could better reduce morphological variants to their underlying types. Although an automatic sampling program can carry out a limited amount of morphological analysis, Muller mentions a number of subtleties in French that an automatic evaluation would almost certainly miss, e.g., conversions of adjectives and verbs to nouns (*les blancs* 'the white people', from the homographic plural adjective 'white'; *le parler* 'utterance, dialect' from the infinitive verb 'to talk'). Muller also observes that the so-called grammatical or function words, e.g., articles, pronouns and auxiliaries, can skew the frequency distributions, and some scholars are inclined to ignore them because they are felt not to contribute to the content of a text. Nevertheless these words often occupy the highest frequency ranks and sometimes they need to be counted in order to obtain an overall distribution of types in accord with Zipf's law. Function words often have more than one grammatical category; e.g., French *que* can be found both as a subordinating conjunction and as an object pronoun. Curiously, if

they are counted, Muller finds that unlike homographic content words, function words seem to behave as if each had but a single category. Their non-lemmatized frequencies are more consistent across large texts than the frequencies of the corresponding distinct, homographic lemmata [Mul77, 27]; thus, for function words the context-less rules of the automatic procedures give more consistent results than the linguistically better informed rules available to a human analyzer, at least for French. Later I shall propose that the kind of meaning that can be attributed to function words is much different from that of 'content words', and that this difference may offer an explanation for Muller's statistical anomaly.

2.2 The Zipf-Mandelbrot Law

The range of variation among the three samplings in Table 2.2 is not excessive (about 9 percent), but it reveals well enough that corpus statistics are too variable to be seen as the immediate expression of natural laws. Furthermore, it shows that any statistical study of a text needs to be explicit about the lemmatization conventions it adopts, particularly if its results are to be reproduced and compared by others; yet unfortunately the research literature has not generally provided this information.

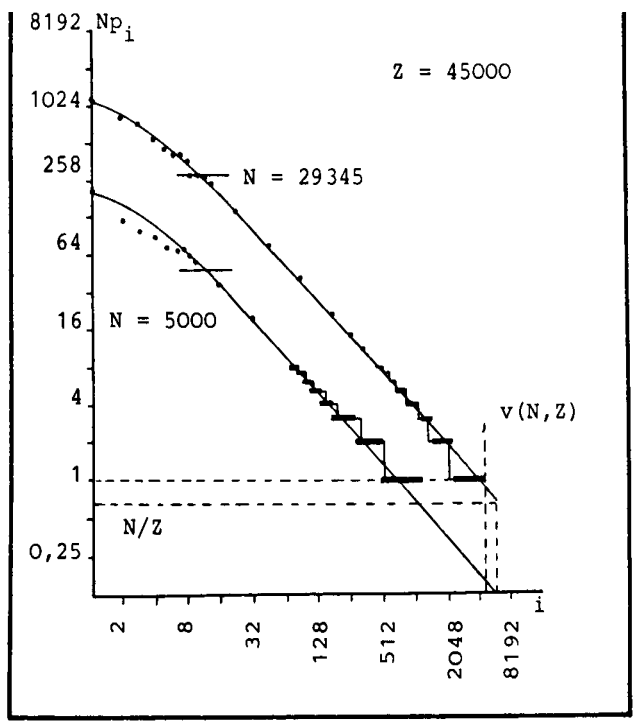


Figure 2.2: Ranked Frequency Distribution in Pushkin's "The Captain's Daughter" (upper curve) and in a 5000 token sample thereof (lower curve). i = type rank; N_i = token count in each rank; Z = Zipf-extent. From [Orl82, 166], ©Brockmeyer 1982.

Muller, like other investigators, found that far from being governed by exact mathematical relations, the frequency distributions in text corpora tend to be irregular, de-

pending on many factors like genre, size, and similarity of the chosen texts, and that they often depart appreciably from the distribution predicted by Zipf's law. Although more complex equations that give better fits to large samplings of text have been obtained by Mandelbrot, Herdan, Orlov, Carroll and others (for a survey, cf. [Baa3a]), what will be of most interest for our purposes are some general tendencies in vocabulary growth that ultimately none of the proposed equations accounts for convincingly. Some of these tendencies are illustrated in Fig. 2.2, taken from [Orl82, 166].

The data in the figure is plotted on log-log scales, so that the hyperbolic curve described by (6) would give a straight line for an ideal ranked frequency distribution. The smoothed curves represent best fits of the data (separate points) to the empirically more satisfactory Zipf-Mandelbrot equation (7), in which an additional constant B is used to shift the curve as needed along the horizontal axis; C is again an arbitrary, empirically determined constant.

$$P_i = \frac{C}{(B+i)^{\gamma}} \tag{7}$$

At the highest token frequencies (ranks $i = 1$ to 10), the empirically sampled tokens (shown as dots) occur less frequently than (7) would predict, resulting in a dip at the left end of the curve. Below the short horizontal line the equation holds until very low token frequencies ($N_i < 8$) are reached. At this end of the range a characteristic step function becomes visible, where ever greater numbers of types have identical token frequencies. This is most evident for the singly-occurring tokens, or HAPAX LEGOMENA (Greek for 'said once') at the bottom right end. Within each of these step-ranges the assignment of type to rank is entirely arbitrary, so that in the ranks from about $i = 500$ to 1100 all of the types are hapaxes, i.e., $N_i = 1$.

2.3 Kalinin's Equation

The data chosen in Fig. 2.2 probably have a fortuitously good fit to (7). Orlov [Orl82] does not explain the lemmatization rules used, and he shows other samplings with rather poorer fits, but this graph illustrates a number of general patterns that other investigators also observed. For large corpora, in particular, (7) often fails to describe the rankings at the left-hand end of the curve, where the largest numbers of tokens appear; and at the other end of the ranking, the shape of the curve also seems to depend on the size of the sample. Small samples typically yield a higher number of types at the lowest ranks (highest frequencies), while large samples show the opposite effect, as can be seen from the ways the stepped portions of the data depart from the smoothed curve. The total vocabulary V of the sample, given by largest rank number attained, i_{max}, can be seen to increase with sample size.

Orlov reports that a rough estimate of the vocabulary $V(N)$ found in a text of extent N is frequently predicted roughly by a relation due to P. Giraud. With R an arbitrary constant that is presumably a property of an author's style and subject matter, the vocabulary V of a sample of text by the author of size N is given by

$$R = \frac{V(N)}{\sqrt{N}} \tag{8}$$

This equation can give a good fit to successively larger samples of a single text. In "The Captain's Daughter" Orlov found $R = 22.2$ at $N = 5000$, 24.3 at 10000, and 27.9 at 29,345 [Orl82, 155]. For Tolstoy's *War and Peace*, R remained at 21.5 even for $N = 472,000$. This means that after nearly half a million words from a single author, the vocabulary of the corpus is still growing, with a new word added once in every fourteen thousand words.

The value of R varies unpredictably from one corpus to another, however, and for a wide variety of texts and authors, Giraud's equation is not very accurate. A better fitting equation due to V. M. Kalinin describes the potential vocabulary $V(N)$ that a text will reach when N tokens have been sampled. It requires knowing the vocabulary $V(N_0)$ for some arbitrary base extent N_0, as well as the type frequency spectrum at N_0. (This is the distribution of type frequencies vs. rank, of the form shown in Fig. 2.1.) Since the type frequency spectrum expresses the relative usage of uncommon words by an author or in a corpus, it takes account of the rate at which low frequency items are being introduced into the corpus via the types with high rank and low token frequency, e.g., for which $N_i = 1$ or 2. Kalinin's equation can thus account for variations in the rate of vocabulary growth as a function of the style of the author or authors of the corpus, expressed in a function K of N_0, N, and the rank frequencies V_i summed over all ranks found in the N_0 sample.

$$V(N) = V(N_o) + \sum_{i \geq 1} K(V_i, N_o, N) \quad (9)$$ [1]

Of course, as purely empirical relations, equations such as (7) and (9) say little about the underlying causes of the relation between the vocabulary size and the extent of a corpus, and thus they offer little help in formulating guesses about structures in the lexicon that could be causing such distributions. However, a great deal of effort has been expended to derive these and similar equations from general statistical models, all of which have assumed a kind of lottery in which words are randomly drawn from the lexicon by a process operating under constraints such as the desire to transfer a maximal amount of information with a minimal number of tokens. Muller [Mul77] speculates that the on-going growth in vocabulary even in texts by a single author has to do with continual shifts in subject matter that require calling out new, specialized terms from remote areas of the lexicon. Orlov also found that in a given corpus the Zipf-Mandelbrot distribution (7) holds only at a particular vocabulary size $V(Z)$, where Z is the "Zipf-extent" or number of tokens that need to be sampled to reach the "ideal" vocabulary at which the log-log plot of frequency vs. rank becomes a straight line. Orlov somewhat fancifully claims that the rank frequency distribution of a text is a function of an author's mastery of his intended theme, and that the Zipf-extent Z represents an aesthetic quantity, the optimal size for a story or novel, beyond which the author's capabilities would be exhausted, leading to long-winded prose and redundant narrative. In Fig. 2.2 it appears that Pushkin is using too many words of low frequency

[1] The formula given by Orlov and attributed by him to Kalinin is

$$E[V(N)] = E[V(N_o)] - \sum_{i \geq 1} E[V_i(N_o)](1 - N/N_o)^i$$

where E signifies mathematical expectation. I take it on faith that an expected $V(N)$ can be computed, although it is not clear to me how the expression is evaluated when $N > N_o$ because the term $(1 - N/N_o)^i$ gives a power series of terms with alternating signs, hence a sum approaching zero. Cf. [Orl82, 156]. In any case, if the empirical growth curve (cf. Fig. 2.4) is known up to N_o, it can be extrapolated by parametric curve fitting to any value $N > N_o$.

for a story of only 5000 tokens, but at 29,345 words he has come close to Orlov's ideal, which would be reached at extent 45,000.

2.4 The Log-Normal Law

A more useful hypothesis about the causes of vocabulary growth, described by H. Baayen in [Baa3a], is the log-normal law of J. B. Carroll, which gives another empirically good description of the rank-frequency distribution, similar to the Zipf distribution, based on a model of the lexicon and a stochastic lexical insertion procedure. Lexical insertion can be seen as a stochastic procedure based on a lexicon organized as an *n*-ary decision tree and a randomly generated selection vector *v* of variable length *m* as the stochastic element. If the search vector is $< 7,2,4 >$, for example, $m = 3$ and the lexical insertion procedure searches the tree to a depth of three, choosing branches 7, 2, and finally 4. The edges of the tree j are annotated with probabilities X_j. Since *m* describes the length of the path via which the lexicon must be searched to retrieve a lexical item *w*, the probability that a given word will be used reflects the number of differentiating decisions needed to identify the item uniquely.

$$P(w_m) = \prod_{j=1}^{m} X_j \tag{10}$$

The probability that a given item will be used is thus the product of the probabilities of the edges that must be traversed to reach the item.[2] For decision vectors with a random distribution of lengths *m* (e.g., $< 7,2 >$, $< 9,2,8,6,7 >$, $< 4 >$, etc., with $m = 2,5,1$, etc.) this produces a small vocabulary of frequent items coming from the apex or start node of the tree, and an indefinitely large number of items of very low probability from the periphery of the tree.

If each decision or element of the search vector represents the selection of an additional semantic feature, the decision tree acquires a hierarchical organization, as has often been proposed in semantic theories, in which relatively general, undifferentiated words (few semantic features) are reached quickly, while highly specific words, with many distinguishing features, are accessed infrequently. Since the insertion probability for an entry is highest near the apex and falls rapidly as concepts become more highly specific (i.e., require more differentiating decisions to be reached in the decision tree), the log-normal model predicts that a small set of semantically undifferentiated words will be used most frequently, while, individually, a large spectrum of highly differentiated concepts will be used infrequently.[3] The probabilistic lexical insertion model gives a rank-frequency distribution similar to that of Zipf's law, but it also can be construed as making an interesting further prediction: the amount of specific semantic information carried by a word is inversely related to its frequency.

[2] I assume that all search vectors are generated equally often, i.e., with the same probability.

[3] An observation of G. Raisbeck, based on the parlor game "Twenty Questions," suggests that values of *m* specifying between 1 and 23 decisions should suffice to search an English lexicon organized as a binary tree. The object of this game is to identify some object or concept by asking a series of 20 yes-or-no questions, followed by three guesses at a specific object. In Raisbeck's experience [Rai66, 29] this number of questions often suffices to guess any unknown concept. If most of the questions can be phrased as binary decisions ("is it alive or inanimate?"), the decision tree will encompass around 2^{20} or 1,048,576 nodes. If all nodes of the tree are attached to named concepts without redundancy, this corresponds well to the number of distinct word senses in a large dictionary.

2.5 Word Cost and Word Frequency

An attempt by B. Mandelbrot to derive Zipf's law from a model of efficient communication is developed in [Rap82]. Mandelbrot showed that the Zipf distribution could result from a linguistic strategy designed to communicate a maximum amount of information with a minimum of "effort", by distributing the frequencies of use among words in an optimal way. If words were coded as signals of fixed length, like the letters in a teletype code, the production of each word would entail the same "effort", and the optimal encoding would be one in which each word has the same probability. However, given the possibility of creating words from a simpler, finite repertoire of elementary symbols, such as letters or phonemes, the effort involved in writing or uttering a word can vary in proportion to its length, and the average cost C_{av} of using the words in the language will be the sum of the length of each word (giving its cost C_n) multiplied by its frequency of use, or probability P_n. For a language containing N words,

$$C_{av} = \sum_{n=1}^{N} C_n P_n \tag{11}$$

the average cost is at a minimum when the frequently used words (large P_n) are short and thus have have low individual cost C_n.

Mandelbrot reasoned that language can be expected to have evolved in such a way as to minimize the overall effort required to communicate a typical message by finding an optimum distribution of word lengths and frequencies. The cost of a given message is minimized when the sum of the costs of the individual words in the message is minimized; thus the message sender must try to use words that are both low in cost and high in communicative content. Drawing on the mathematical information theory of C. E. Shannon [Sha48], Mandelbrot could show that a lexicon which lets speakers communicate as much information as possible in a message of limited size will attain a unique tradeoff of words' lengths against their frequencies of use.

Roughly speaking, information theory defines information in terms of the transmission of messages as chains of symbols over a communication channel. The amount of information a message carries is defined as the inverse of the receiver's ability to guess at random what the message might be without actually receiving it. Thus, for a message of one symbol in a code containing only two symbols used equally often, say 0 and 1, a receiver could guess the message correctly half of the time without actually receiving it. For messages of two symbols, this would be true for only one-fourth of all messages. It follows that longer messages are more "informative" in the sense that their content is less easy to guess beforehand than that of short messages, and unexpected signs similarly carry more information than expected signs. Actually, Shannon expressed the amount of information H carried by a single sign or message as the logarithm of the inverse probability of the sign,

$$H = log\frac{1}{P_n} = -logP_n \tag{12}$$

since this corresponds to the intuition that, all things being equal, a message of two signs (or a report of 200 rather than 100 pages, two telephone lines rather than one) ought to carry two rather than four times as much information. The amount of information in a message is thus directly proportional to its length, and the amount of

information in a given message is a sum over the information carried by each of its signs. For the average information H_{av} carried by a sign in a code of N symbols, Shannon derived an expression based on the sum of the information carried by each individual sign $-logP_n$, weighted by the sign's frequency of use, or probability P_n:

$$H_{av} = -\sum_{n=1}^{N} P_n logP_n \tag{13}$$

This shows that a code with many symbols packs more information into each symbol than a code with few symbols. Furthermore, it follows that the average amount of information each sign carries is maximized when all signs carry the same amount of information, which requires that all signs be equally probable. [4] To maximize information, all signs in a code or language should be used with equal frequency. However, for a given amount of information, (11) requires us to minimize the average cost per word; i. e., the longer a word, the less frequently it can be used.

Given a language with a vocabulary of V words, Mandelbrot showed that for a lexicon that uses all possible letter combinations as word types, and where the types are labeled from shortest to longest with the index r (each type having a different index), minimum cost of information transfer is obtained with maximum information if for each type of length rank r

$$P_r = \frac{A}{(m+r)^B} \tag{14}$$

which is strikingly similar to (7). The sole substantive difference is that r represents not the statistical frequency ranking (i in (7)) but the cost ranking or position in the length-ordered list of vocabulary items. The adjustment exponent B corresponds to γ in (7), and the constants A and m are represented in (7) as C and B. This produces a curve for size vs. frequency rather than number-of-types vs. frequency. But if the shortest possible words are used most frequently — more exactly, if $rf(r)$ is a constant, where $f(r)$ is the frequency with with each type r is used in a corpus — then (14) suggests that the rank-frequency law (7) can be derived from general principles of communicative economy, rather than from the operation of specific cognitive or linguistic structures. For reasons to be discussed shortly, length probably does not correspond accurately to frequency rank in most languages, meaning that Mandelbrot's equation, like that of Zipf, must be seen as a suggestive but ultimately inadequate characterization of the rank-frequency distribution.

Other, in some cases more elaborate models were proposed to derive equations describing the rank-frequency distributions found in corpora from general principles, but it would be futile to pursue their history further because no sound data have been found which would allow a clear, empirically justified choice among them, as Baayen [Baa3a] has found. Nevertheless, we can end this brief survey of statistical descriptions of the lexicon with two important observations. One is that, whatever the cause,

[4] This can be easily verified by checking two possible ways of redistributing the probabilities P_n without changing their sum. If one of the signs in the message is given a very low probability, its contribution $P_n logP_n$ to the sum vanishes by virtue of the low value of P_n (not offset by the larger $logP_n$), while the other terms change little in value. If one of the signs is given a very high probability, its contribution again disappears by virtue of a low value for $logP_n$ (not offset by P_n near 1), while the other signs' contributions diminish roughly in proportion to their reduced P_n. Cf. [Rai66, 12-15]

queries to the lexical data bank certainly vary widely in frequency, with most queries being directed to a small subset of the whole lexicon, suggesting that any reconstruction of lexical structure ought to allow quick and reliable access to a small number of most frequent lexical entries. The other is the prediction that, to minimize the overall communication cost, the high-frequency words are likely to have a low information content, perhaps implying simple lexical structures, whereas the lower-frequency words will probably have the most elaborate lexical representations. [5]

2.6 Variable and Sub-Optimal Word Lengths

The Zipf-Mandelbrot law implies a continuous decrease in probability with a corresponding increase in information across the vocabulary spectrum of a corpus. Its mathematical derivation assumes, however, that all combinations of phonetic signs yield possible words, something we simply do not find in natural languages. In fact, for the lowest-cost words at phoneme length 1, we seem to find far fewer words than the number of available phonemes would allow. French, for example, uses a few vowels to represent words like the clitic *y*, the conjunctions *et* and *ou*, and the preposition *à*; English has *I* (in careful speech actually a diphthong), which in German is the noun for 'egg'; but in general, some competing principle seems to set a minimum length of two or three phonemes per word in European languages. For longer, purely random strings of phonemes or characters, we find that the majority are not words (a fact that spelling check programs take advantage of). It turns out that these constraints on the phonotactic structure of words can also impose very significant constraints on the rank-frequency structure of the lexicon.

2.6.1 Block Codes and Phonotactic Constraints

Information theory characterizes codes that constrain the possible encodings to subsets of the available symbol combinations as BLOCK CODES. One common form of block code is the class of run-length limited codes used for storing data on magnetic disks, where the physics of the medium require that no more than a certain number of 1s or 0s can follow in succession. A simple example of such a block code is shown in Table 2.3, which can represent any sequence of 1s and 0s in such a way that more than two 0s never occur together. Thus, 0001, which would be illegal, is represented in the block code as 010011, which is a legal sequence.

Table 2.3: A Simple Block Code

ELEMENTARY SIGNS		REPRESENTATION
00	$\longleftrightarrow$	010
01	$\longleftrightarrow$	011
10	$\longleftrightarrow$	110
11	$\longleftrightarrow$	101

[5] Other empirical and theoretical evidence for this hypothesis is cited in [Baa3a, 359].

Similarly, natural languages do not allow their primitive signs to appear in all possible combinations. There are phonotactic rules governing how phonemes can be combined; these rules require, for example, that no more than a certain number of consonants can occur together in a cluster, that no more than two or three vowels can occur together, and that words must be built up of syllables of alternating consonants and vowels. Unlike the code of Table 2.3, however, the code words in natural languages are variable in length, and running speech provides little explicit segmentation of the information stream, which means that it can be difficult to tell where one code symbol ends and the next one begins. An important result in information theory is that codes containing blocks of variable length can be optimized relatively easily if they are recoded as PREFIX CODES, in which certain special prefix sequences never occur as independent signs [Bla91, 38 ff]. Prefix codes can also help the receiver of a message to identify the boundaries of the individual signs.

2.6.2 Huffman Codes

A special coding problem posed by a natural language is the wide range of frequencies (or insertion probabilities) with which the words making up the code are used. A class of prefix codes, called HUFFMAN CODES, commonly used by text compression programs, often allows very good compaction of text data transmitted via a block code, such as the ASCII encoding of the Roman alphabet. The Huffman code elaborates block code encoding by taking account of the relative frequencies of the signs in defining the set of prefixes. In a Huffman code the most frequent signs are allowed to appear without prefixes, allowing them to be relatively short. Less frequent signs are given short prefixes, and the least frequent signs receive the longest prefixes. Adding prefixes in this way, it is possible to construct a mapping of abstract signs to code words in such a way that the frequent and therefore least informative signs consume little space in the stream, while the infrequent but most informative signs occupy correspondingly more space. If a language were to consist of the set of abstract signs (or 'meanings') A, B, C, D, E, F, G, with relative frequencies ranging from $3/8$ to $1/32$, it could be represented in terms of the elementary signs (or 'phonemes') 0 and 1 in a fixed-length block code with the code words 000 to 110 (in a binary numbering), but this encoding would be inefficient, as is easily seen by considering that every time the high-frequency, low-information item A appears, it consumes three elementary signs (binary digits) for its representation. Using the Huffman code shown in Table 2.4, strings of abstract signs with the given frequency distributions would be represented in nearly as few elementary signs as possible, and the cost of transmitting a message, as defined by the Mandelbrot equation 11, would be minimized.

In this encoding, 01, 00, 000, and 0000 can be regarded as prefixes selecting various word classes, such that each class contains at most two words, represented by 1 and 0. Only the sign A is not a member of a prefix class. As can be easily verified, the prefixes have been chosen in such a way that any string can be unambiguously segmented left-to-right by a determinate finite-state transducer that maps the code words to the meaning symbols, owing to the fact that 0 always marks the beginning of a prefix. The lengths of the code words match roughly the logarithms of the reciprocals of the frequencies, which according to (12) give the amount of information carried by each code word, and in this encoding the ratio of length, or transmission cost, to

Table 2.4: Huffman Code for a 7-Word Lexicon (from [Bla91, 69])

MEANING	FREQUENCY	OCTAL CODE	HUFFMAN CODE
A	$3/8$	000	1
B	$3/16$	001	$01+1$
C	$3/16$	010	$01+0$
D	$1/8$	011	$00+1$
E	$1/16$	100	$000+1$
F	$1/32$	101	$0000+1$
G	$1/32$	110	$0000+0$

information is roughly constant across the set of symbols. An equivalent suffix code can be declared by arranging for the code words to be analyzed from right to left, but it cannot help segmentation if the analyzer is working from left to right. This may help to explain the special role that prefixes and meaningless "pseudo-prefixes" seem to have in word recognition, as we shall see in the next chapter.

2.6.3 Coding Optimizations in Natural Language

Although it is true that natural languages are far from being as systematic as formal codes can be, it takes no great leap of the imagination to suspect that such coding schemes, adapted to constraints on the minimum length and phonological form of a separately communicable sign, may have been discovered by natural languages. Natural languages seem to provide evidence of coding optimization in two areas: in their use of pseudo-prefixes within words, and in their use of short, low-information words and affixes as a kind of "meaning selector" in phraseological constructions and complex words.

As was mentioned above, an important result in information theory is that codes containing blocks of variable length can be optimized relatively easily if they are organized as prefix codes, in which certain special prefix sequences never occur as independent signs, and only occur at the beginnings (or perhaps at the ends) of larger symbols [Bla91, 39]. Such prefixes can also aid segmentation of a variable-length code. If used in a natural language, such meaningless prefixes would preserve well-formedness of phonetic sequences that, in principle, could be coded more compactly, but which would then violate phonotactic constraints. Such prefixes might also help a listener to identify word boundaries. They would not be meaning-carrying constituents of the word but would rather function as easily recognized "dummy segments" to keep the word long and clearly distinguishable. If, in addition, the pseudo-prefixes appear mainly in words of low frequency, they may be helping to optimize the linguistic code, in accordance with (11), by making shorter and thus less expensive phonemic sequences available to encode the more frequently used words in the language. In the next chapter we shall encounter evidence that some languages, like English, do in fact make use of meaningless "pseudo-prefixes" (like *re-* in *rejoice*) that may be motivated by coding constraints like those for which Huffman codes are employed. Like the run-length prefixes pre-

sented in Table 2.4, these do not map independently to any values or meanings of their own; i.e, *re-* in *rejoice* has no independent meaning — *rejoice* does not mean 'joice again' — just as the prefix 00 by itself does not code any of the lexical meanings in in Table 2.4. However, even though they carry no meaning, pseudo-prefixes do appear to preserve the prosodic and phonotactic well-formedness of a word, which might be lost if the word were allowed to collapse to the minimal number of segments needed to distinguish if from similar phonetic strings. They may help to prevent infrequently used words from evolving into shorter phonemic sequences that must be reserved to represent frequently used words.

Low-information, "function words" also suggest an optimization of language to minimize the cost of transmission. A noun or a verb at a given position in a sentence usually represents a selection from thousands of possible alternatives, and thus it carries a large amount of information. The majority of these high-information words are long and polysyllabic. At the same time, we often find short, mono- or bisyllabic, and thus "inexpensive" words, like pronouns, prepositions and the verbs used as auxiliaries, that select from a small range of rather unspecific and abstract meanings, like a possessive association, a causal relation, or a temporal marking for perfect or future. Owing to the small set of alternative symbols that the language would permit in the same distributional position, these words are fairly predictable and thus they carry relatively little information. To illustrate, in the sentence *Think of my refrigerator* there are probably many tens of thousands of possible alternatives to the polysyllabic content word *refrigerator*. The short possessive pronoun *my*, however, specifies one of a small number of choices, namely *my*, *your*, *their*, etc. Short words like *my* may in some way be functioning like the prefixes of the Huffman code, allowing infrequent and thus heavily information-bearing abstract signs, or meanings, to be encoded as longer variants of more common signs. (The words and morphemes under discussion here do carry *some* information, however, in contrast to the pseudo-prefixes mentioned above; the nature of this distinction will have to be worked out later.)

Likewise, within words we also find a small number of very compact signs, such as tense and case morphemes, that perform a similar, meaning selecting function (as genuine, rather than pseudo-affixes). Although unspecific, these meanings can be varied in highly systematic ways (e.g., with respect to number with nouns, temporal reference with verbs), and among the variants some are typically used much more frequently than others. Rather than burdening the code with separate, long, and thus "costly" representations of all the semantic variants, many languages appear to supply short, thus inexpensive markers that greatly multiply the number of possible meanings of a given word but still allow the frequently used meanings to be represented compactly. Consider, for example, that forming plural nouns in English by addition of the morph /s/ allows the more frequent singular forms to have a more compact representation. French verbal inflections show this sort of economy even more dramatically, representing the present tense, singular, in most cases with no audible suffix (*je, il aime*), the corresponding plurals with a single additional syllable (*nous aimons*), while the less frequently used forms add additional syllables (fut. *nous aimerons*; pres. cond. *nous aimerions*).[6] The use of short, low-information symbols thus appears to optimize

[6]The *meanings* of the present tense, singular forms are probably not simpler than those of the less frequent forms, but because of their high frequency of use, information theory makes it clear that much

the information vs. cost ratio of the linguistic system in a way somewhat similar to that employed in Huffman codes, and in the overall linguistic system these symbols may have a status different from that of the longer, more heavily information-bearing stems to which they are appended. The low information content of these morphs may, indeed, have something to do with the lemmatization discrepancy Muller found (section 2.1) between function and content words in French texts: functioning as "meaning selectors" rather than carriers of meaning, it may be incorrect to distinguish homophonous function words by meaning. As "meaning selectors" they may in fact have no proper meanings of their own in the way that content words do. Further, systematic evidence for a hierarchy of meaning types distinguished by lexical 'region' in the lexicon will be presented in chapter 5.

These diverse speculations on possible causes of the frequency distributions of words in texts begin to suggest that a study of the restrictions individual languages place on the ways they allow words to be constructed and varied in form could lead to a more exact understanding of the rank-frequency distribution. Unfortunately, the research tradition in lexical statistics tended to see morphology primarily as an annoyance, since morphology was responsible for the lemmatization problem that hindered access to the information-bearing level of the text. Morphology was little discussed, even though some sort of morphology was inevitably necessary for unambiguous lemmatization, and it was overlooked as a factor that could be isolated and invoked to explain the distribution of word frequencies in finer detail.

2.7 The Statistics of Word Formation Classes

As we have seen, statistical investigations of large corpora show time and again that the vocabulary of a corpus tends to grow indefinitely and without bound. While the program of research into rank frequency distributions made the vocabulary growth of large corpora a central research issue, a specific underlying causal mechanism was never clearly identified; the growth curve was seen rather as the result of a probability distribution in a lexicon that included vast, virtually limitless numbers of rarely used words. What apparently escaped notice by the entire tradition (surveyed in [Baa3a]) is that much of the observed vocabulary growth takes place within a small number of lexical categories and morphological patterns. This implies that the word selection process is not a random sampling from a near infinite population of word types, but that for some reason the words of lowest probability can only be drawn from certain sub-sections of the lexicon. In fact, it has turned out to be most fruitful to define these sub-sections in terms of word category and morphology. Shifting the focus of interest in lexical statistics from lemmatized semantic types to morphological types — from *langue* to *parole* — has brought about a major revision in the approach to lexical statistics. This revision is due above all to both statistical and psycholinguistic studies carried out over the last decade by H. Baayen and numerous collaborators.

The essential feature of rank-frequency distributions that earlier studies overlooked is evident in Fig. 2.3, taken from [Baa3a, 357]. The right-hand curve shows a log-log

of the meaning of the present tense, singular forms does not have to be carried explicitly, i.e., as differentiating features of their verbal signs. Instead, these features are reconstructed at the receiving end, as in the decoding of a block code.

plot of lemmatized stems in a Dutch corpus of one million words; it closely approximates the straight line predicted by (6). If words are truly being selected at random from the lexicon, any subset of the corpus should give a similar curve; any other result contradicts the basic statistical sampling theorems, which assume that the whole of a population can be characterized by studying random subsets. The curve on the left, however, representing just the monomorphemic content words, departs strongly from the Zipf relation. The vocabulary for these words is more evenly distributed across the lowest ranks, while at the high ranks the curve falls off sharply, implying that contributions to the highest (most infrequent) ranks in the full corpus must come from other areas of the lexicon. Such a striking departure from the distribution of the corpus as a whole pleads for an explanation. Yet, taking stock of previous efforts, Baayen observes that "None of the [proposed] rationales for word frequency distributions ... is of any help" in explaining this peculiarity. It is, instead, morphology that holds the key to an explanation. "The analysis of the frequential characteristics of morphological categories reveals that each category has its own (conditional) growth rate and theoretical vocabulary size, depending on the productivity and extent of use of the category." [Baa3a, 357-8].

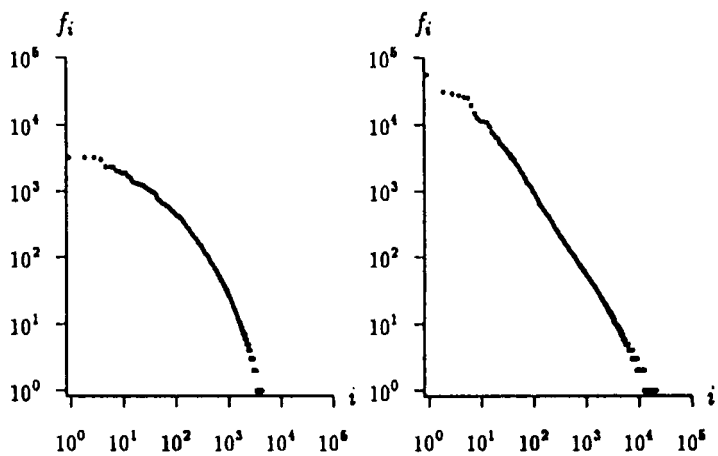


Figure 2.3: Ranked Frequency Distributions for Dutch stems. Left: monomorphic content words only. Right: All words. $N = 1,000,000$. From [Baa3a, 357], ©Walter de Gruyter 1992.

Why should the vocabulary of monomorphemic content words be so sharply limited? For another class of monomorphemic words that comprises the lexical categories for conjunctions, pronouns and prepositions (the "function words" of a language), the vocabulary is well known to be "closed", in the sense that these classes are limited to a small number of words that can be easily listed. This means that once a certain number of function-word tokens from a corpus have been sampled, all members of these classes will have been seen, and further tokens will not contribute to any higher ranks in the ranked frequency distribution. To explain the distribution found in the full sample from which Fig. 2.3 is derived, we must assume that the monomorphemic content words are also a relatively closed class, while a vast, perhaps limitless number of other, non-closed types are, so to speak, buried so deeply in the lexicon that they

rarely surface in texts, and that these classes alone account for the high ranks with very low frequencies of occurrence. The lower probabilities of these polymorphemic words is partially accounted for in Mandelbrot's relation (14), since polymorphemic types are bound to be longer, thus higher in length rank r, more expensive to use, and of lower insertion probability P_r, than the monomorphemic types. This explanation is not entirely satisfying, however. Like Zipf's law, Mandelbrot's relation defines no upper limit to the number of ranks n and thus to the number of types to be found in a corpus, and in fact Giraud's inverse square relation (8) implies that the vocabulary of a corpus is limited only by the size of the corpus. That this has been confirmed repeatedly by investigations of very large corpora seems to require a generalization from the finite vocabulary of a finite corpus to an infinite sampling space encompassing all possible texts, an idealization that statisticians like to obtain for mathematical reasons, but one that leads to the rather unsettling assumption that speakers must have a virtually limitless number of polymorphemic words available in their mental lexicons.

2.8 From Frequency Distribution to Growth Rate

Much like the tradition of lexical statistics, computational linguistics has by and large adhered to a picture of language that postulates a lexicon containing a potentially very large but static inventory of linguistic signs along with a combinatorial component called syntax. Like many basic decisions in computational modeling, this partitioning is to some extent arbitrary, but it has made it possible to solve the problem of describing language's "infinite use of finite means", to which N. Chomsky drew attention in [Cho57] and [Cho65], that is, the ability of speakers to produce an unlimited number of meaningfully distinct utterances on the basis of a limited number of words and grammatical constructions. In most computational architectures the "creativity" of language is realized by formal systems in which the "means", i.e., the lexicon and the set of syntactic rules, are finite, but the combinatorial possibilities for producing sentences are unbounded, owing to recursive structures in the syntax. Over time, languages display creativity in the lexicon as well, introducing new words and discarding others, and accordingly computational models acknowledge that the static lexicon is a simplifying assumption. But it is usually assumed that historical changes are slow enough to allow a valid synchronic snapshot of the state of the language at a given time.

There are several computational justifications for adopting this convention. For one thing, from a theoretical perspective it is difficult to know much about the computational properties of a grammar in which the set of combining elements can change during the course of deriving a sentence — for example, whether or not the system generates only grammatical sentences. If the static lexicon is abandoned, derivations of sentences come to depend on the order in which operations are performed, since one of the operations in the grammar must be that of creating a new symbol (e.g., a word) required by other operations. This, in turn, requires a highly procedural and therefore highly specific definition of the grammar, allowing far more possibilities for the description to be empirically incorrect. From the practical point of view, the rule system and lexicon of most useful natural language applications must in any case represent the current language of the system's users; allowing for the use of non-current

or merely "possible" words could be seen as distracting.

The description of the current language must, however, be derived from — or at least justified by — a corpus of texts, all of which can hardly be produced at the same instant. Compilers of dictionaries, of course, often extend their corpora to include a broad, historical view of a language, which runs counter to the ideal of a synchronic description of the linguistic system and would unnecessarily complicate a computational lexicon. But if a corpus is gathered within a narrowly limited time interval, the effects of historical change will be vanishingly small, and the size of the corpus can be quite large. If we scan a very large, quasi-synchronic corpus from end to end, we can create the illusion of a word-producing process like that of spontaneous speech. If we pass the corpus through a program that gathers a list of differing strings — i.e., that constructs a concordance to the corpus — the assumption of a finite, static synchronic lexicon predicts that we should find some point in the stream where all words will have been seen at least once, and the concordance will cease to grow. But, as we have seen, even large corpora fail to reach an upper bound for vocabulary. Not only does the lexicon change historically, but even within a single corpus, obtained over a short enough time span to be considered a synchronic sampling of what speakers of a language can produce, we apparently find evidence of a generative component in the lexicon itself, continually introducing new words that have not been previously registered, just as new, never-heard and never-seen sentences are continually introduced within the synchronic time-frame. Seeing the lexicon as the set of all words encountered in an arbitrarily large corpus hides these generative processes in the infinite population assumed in classical sampling theory.

On the other hand, the generative component might become visible within the synchronic time frame if we were to think of the corpus not as a static set but as the output of a process that both inserts words from the static lexicon and also produces new word tokens according to some language-specific stochastic procedure. To maintain the distinction between synchronic and diachronic study of the language system, we can examine the output of this process as a function of time, but on a compressed time scale that has no relation to the much more extended scale of linguistic change. The description of the process thus will belong to the synchronic description of the language, not to a description of its history, and it can nevertheless reveal any word-creation processes that might be at work in the synchronic system of the language. It may, of course, also offer insights into the nature of historical change, as well, but this possibility will not concern us until much later.

Returning to the Dutch data represented in Fig. 2.3, try to consider how monomorphemic content words sampled from the corpus would accumulate when viewed in this way. Since the monomorphemic types tend to have high insertion probabilities, the text generation process will quickly produce a large number of them. However, since only types from relatively closed classes are being counted, most of the available types will be produced in a short time, and further tokens will contribute ever fewer types to the total vocabulary under investigation. In other words, at the start of the sampling, it will appear that many new types are being produced, but the rate of production then falls off rapidly. If we tabulate the number of types V accumulated vs. time, represented by the number of tokens sampled N, we can expect a curve that rises sharply and then flattens out, as in 2.4 (a). For the corpus as a whole, the curve should be less flat, as in (b); this is owing to the additional contributions of polymorphemic types that have growth

curves like that of (c), where it appears that the generation process is continuously able to find and insert new, previously uncounted items into the output stream.

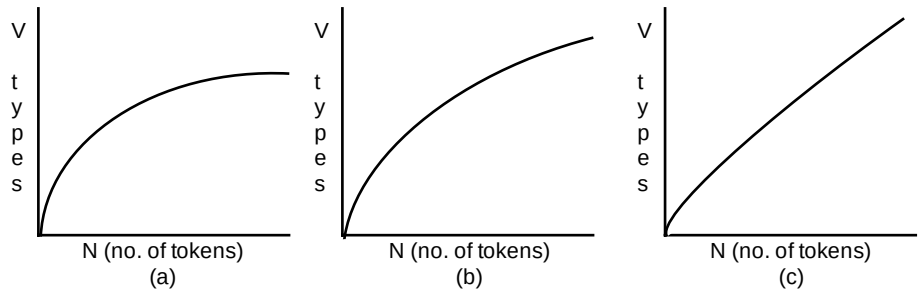


Figure 2.4: Expected Growth Curves. For a single morphological formation in a static lexicon with (a) small V and high $P(i)$; medium V, medium $P(i)$, (c) very large V, very low $P(i)$.

How is the continued growth in vocabulary to be explained? Kalinin's equation for vocabulary growth (9) predicts that, in the static analysis of the corpus, it is the tokens of the highest ranks (lowest token frequencies) that contribute most to growth, since the exponent j tends to weight the contributions from the higher ranks, in particular the hapaxes, most heavily (refer to the footnote to Equation 9). Statistics over the rate at which hapaxes are introduced in a corpus thus give a good diagnostic of the rate of vocabulary growth to be expected, at least within the limits of the data against which this equation has been tested; this has been confirmed empirically for English corpora by Baayen & Sproat [BS6a]. Nevertheless, it strains credulity to maintain that all such words, used once and never again, are drawn from a vast but nevertheless static lexicon. It would almost invite us to maintain that sentences, too, are drawn from a vast mental storage space in which all required utterances are held ready for use. To be sure, it might be objected that the appearance of limitless growth is merely an artifact of the relatively small samples that most statistical studies have been based on. The mental lexicon may in fact be much more vast than either dictionaries or small corpora would lead us to believe, and repetitions of the lowest frequency tokens might well appear in larger samples. The largest of Orlov's corpora contains 689,214 tokens, and most of his other corpora contain far fewer than 100,000. The Dutch corpus from which Fig. 2.3 was drawn contains about 1,000,000 tokens. But in the course of many years, mature readers will have been exposed to a far larger sampling of their language than this, and one might suppose that at some point in the experience of mature speakers a vocabulary limit is reached.

To test this hypothesis, Baayen & Renoulf [BR6b] investigated the growth curves of a small number of polymorphemic types in a corpus of 80 million words, drawn from 40 months of the daily London *Times*. This study produced two important findings: one is that a significant number of types in the corpus do not appear in a comprehensive ("unabridged") dictionary; the second is that in certain identifiable morphological classes, the growth rate remains surprisingly high even at $N = 80,000,000$. While surprising, the implications of these findings are not entirely obvious.

Baayen & Renoulf think it "highly unlikely that words occurring with a frequency of 1 in 80 million ... are available in the mental lexicons of individual language users"

[BR6b, 77]. This undoubtedly overstates their case, for it is to be expected that in any large corpus many words not documented in dictionaries will be drawn from technical specialties or foreign languages and will in fact be found in the mental lexicons of a small portion of readers. In fact, although the corpora widely used for studies of English (the Brown Corpus [KF67] and the LOB Corpus, described in [HJ80]) were sampled from a wide variety of text genres, it has been shown that a similar sampling for German, the 1973 Limas corpus (described in [HBL83]), grossly underrepresents many specialty vocabularies. Drawing on a corpus for medicine constructed from 11 German journals and four Internet resources, R. Hausser found many items that were not found at all in the Limas corpus, words like *radiologisch* 'radiological' (183 tokens), *Insulin* (135 tokens), and *prospektiv* 'prospective(ly)' (127 tokens). In Hausser's specialty corpora for sports and religion, proper names contributed most of the items not found in the Limas corpus. An informal study of technical manuals conducted by the present author[7] found that for a value of N equaling several hundred thousand tokens, new tokens were being introduced at the surprisingly high rate rate of one or two per hundred. These were by and large designators for machine parts or names of software functions in a specific product, and as such they would be of little interest to lexicographers, but they definitely represent fixed lexical items known to a small subgroup of speakers. The high rate of vocabulary growth in manuals results, of course, from their intended purpose, which is to name and describe unique objects with which the intended readers will not be familiar. Likewise, it is to be expected that in a corpus of newspaper texts like Baayen's *Times* corpus, a significant number of new types will result from the journalist's task of supplying readers with new information, which will often include names and descriptions of terms from technical specialties so limited that they escape the notice of lexicographers, or from foreign cultures and languages not covered by a mono-lingual dictionary. Mechanically gathered word-frequency statistics also tend to inflate the apparent frequency of hapaxes by including misspellings and typographical errors, as Hausser found in the concordance to the 100-million-word British National Corpus [Hau98, 14].

Baayen does not break down the sources of new monomorphemic items in his study of the Dutch corpus mentioned above [Baa3a]. Among the 36 percent of monomorphemic hapaxes this study found, a good portion may well have come from specialty areas (although the technical terms are more likely to have recognizable, polymorphemic structure). For the *Times* corpus similar results might be expected, although Baayen & Renoulf present no statistical data to confirm this. (They do note that many of the hapaxes in *in-* appear to be technical terms seldom used in a general-readership newspaper, e.g., *incongruent* and *indistinctive*.) Because such words are familiar to small subgroups in the linguistic community and may in fact already have a long history, it would be legitimate to see them as belonging to the established lexicon of the language. Within various linguistic subgroups, their use may be frequent, but for the speech community as a whole the average frequency is so low that even an unabridged dictionary would not want to include them. Vocabulary growth from these sources is thus consistent with the vast static lexicon postulated in the early work on lexical statistics, and it cannot be seen as the result of genuine synchronic lexical inno-

[7] These data were collected in the course of developing a spelling correction program at GenRad,Inc., Concord, Massachusetts, in 1982 and 1983.

vation; the hapaxes are "taken from" specialty vocabularies rather than invented, and a reader wanting to know exactly what they mean will turn to the relevant textbooks or experts.

Table 2.5: Examples of Nonce Words in the London *Times*, from [BR6b]

-ly	*un-*	*-ness*	*-ity*
abroadly	unabove	cowness	anality
blondely	unbuild	endness	assurity
conductorly	uncaress	footballness	avuncularity
oftenly	unconclusion	itness	deviosity
onely	uncheesy	joyness	loyality
tumescently	unflat	outsideness	spectacularity
whyly	unfishy	wonderness	terrority

This still leaves a large proportion of polymorphemic neologisms that are less likely to be existing importations from technical specialties or foreign languages. In the *Times* corpus, Baayen & Renoulf found large numbers of hapaxes formed with the prefix *un-* and with the suffixes *-ly*, *-ness*, and *-ity*. Examples, listed in Table 2.5, are moderately to completely transparent, and none has a meaning that could only be associated with a particular technical jargon. However, many of the examples in Table 2.5 are likely to be judged out of context as semantically anomalous or ill-formed, even though they have been used in a newspaper that conforms to widely accepted standards. Baayen & Renoulf observe that the undocumented hapaxes in the *Times* corpus often appear not to be deliberate coinages but seem rather to be created to fill a transitory need not met by an apparently equivalent, lexicalized item. Many would appear to be redundant with existing lexical items, e.g., *unbuild = take apart*, *oftenly = often*, *loyality = loyalty*. For these reasons they are likely to be ignored by lexicographers and not taken seriously by morphologists. Of the hapaxes suffixed with *-ly*, 560 are not listed in the *Webster's Third New International Dictionary of the English Language* (1981), which Baayen & Renoulf found to be the most comprehensive available. For *un-*, *-ness*, and *-ity*, 450, 348 and 143 types were not listed. In some cases the communicative need for which such "nonce words" are created may be to name something that normally does not or cannot exist and that would otherwise be seen as semantically ill-formed, such as the 'un-doing' of an irreversible action in a sentence like *Beware, you cannot* uncaress *someone you do not like*. Such formations are thus instances of "the spontaneous, unintentional and ephemeral use of productive word formation, not . . . conscious and deliberate lexical creativity in which a novel expression is carefully constructed to express a new concept intended for repeated use within an — often specialized — domain" [BR6b, 78]. Like sentences, nonce words are evidently used creatively to describe actions, objects, or situations that do not belong to the inventory of lexicalized concepts. Because of their anomalous semantics, they are often forgotten as easily as they are created.

To a smaller but non-negligible extent, it would then appear that the lexicon must also contain a generative sub-system, capable of producing word-like symbols for con-

cepts that are formulated and possibly discarded again. Indeed, Baayen reports that in the large Cobuild corpus of written English, 64 percent, or nearly two-thirds of the hapax legomena belong to types that can be classified as morphologically complex [Baa3a, 358]. A striking difference to sentence-level generativity, however, is that lexical generativity is highly restricted. In a large text it will be hard to find repeated instances of the same sentence, but most words will in fact be drawn from the static inventory of the lexicon, suggesting that the generative apparatus for building words is used comparatively infrequently. Furthermore, whereas speakers tend to favor a varied diet of sentence formation rules, often using extraposition, coordination, subordination, etc. merely for the purpose of avoiding monotony, spontaneous word formation tends to be activated only when a compelling semantic need exists, such as a lexical gap or a rhetorical effect that cannot be obtained easily by other means. The patterns available for "spontaneous, unintentional and ephemeral" creation of words are not identical with the patterns of word formation in general, however, and they may not coincide with the patterns that the morphological literature has identified as productive morphology.

2.9 Measures of Spontaneous Productivity

To characterize more precisely the spontaneous word-formation patterns that contribute to vocabulary growth in a large synchronic corpus, Baayen has tried to break the somewhat vague linguistic notion of morphological productivity into a set of complementary, statistically defined, empirical measures of word formation activity. These measures clarify a number of vagaries that have plagued the descriptive literature on morphology.

We have seen that even very large corpora may fail to exhaust all of the possibilities of a given word formation pattern, such as the derivation of adjectives in English prefixed with *un-*. Dictionaries attest many forms built on this pattern, and in the *Times* corpus new, unlexicalized forms are continually being introduced. This pattern is clearly able to produce a large number of types and can be described in a general way as productive. Other patterns for which dictionaries attest few types may also not appear to be particularly productive in texts, being represented in few hapaxes, and these patterns can be described as unproductive. But in yet other cases, such as derivations in English of verbs with *un-*, dictionaries show a large number of types, and indeed speakers' intuitions are generally that nearly any goal-directed verb can be given a reversed meaning by prefixing *un-*, even in such peculiar cases as *uncaress*. It is therefore certainly not wrong to describe the derivation of goal-directed verbs with *un-* as productive. Yet while Baayen's *Times* corpus contains 1672 verbs prefixed with *un-*, in 80 million tokens only 10 of these appear as hapaxes, meaning that the spontaneous derivation of new verbs on the pattern of *undo* is in fact rare. For whatever reason, it appears that most of the many verb forms which can be created with *un-* are already being used repeatedly, and that there are few possibilities left for creating more. If the *Times* sample were greatly extended, we could thus expect to find very few new 'reversative' verbs formed with *un-*, but new adjectives formed with *un-* would continue to appear. Thus, in one special sense, derivation of reversative verbs in *un-* is hardly productive at all.

To characterize these differences, Baayen [Baa92] defines four complementary measures of productivity. For a given word-formation pattern *w* in a given corpus of size *N* we can determine:

- the number of attested (lexical) types V_w ("extent of use")
- the rate at which new types appear $\mathcal{P}_w$ ("degree of productivity") at corpus extent *N*
- the number of potential forms S_w ("population of types"), the 'virtual' vocabulary belonging to an abstract, synchronic description of the language, independent of *N*
- the ratio of potential to attested forms, $\mathcal{I} = S_w/V_w$ ("index of pragmatic potentiality") at extent *N*.

Summing over all the patterns, we get a total attested vocabulary *V*, a rate of vocabulary growth $\mathcal{P}$ and a potential or virtual vocabulary *S* for the language as a whole. Assuming an unambiguous definition of the word-formation pattern *w* under investigation, the number of attested instances N_w of the pattern *w* in a corpus will be obtained simply by examining tokens and counting those that match the pattern. The vocabulary V_w (obtained by eliminating token repetitions of types belonging to *w*) will grow as some function of the corpus size *N*.

As we have already seen (Fig. 2.4), the growth curve for a given word formation pattern can assume various forms. The slope of the curve $\mathcal{P}_w$ gives the growth rate as a function of *N*, the number of tokens that have been sampled, and $V_w(N)$, the number of types accumulated as a function of *N*.

(15) $$\mathcal{P}_w = \Delta V_w(N)/\Delta N$$

The growth curve for *w* can be obtained by integrating the growth rate $\mathcal{P}_w$ over the length *N* of the corpus.

It is evident that *V* and $\mathcal{P}$ measure two distinct and incommensurable forms of morphological productivity: *V* is likely to reflect the number of types *in common use*, whereas $\mathcal{P}$ reflects the ease with which new forms, *not in common use*, can be introduced in the language. To compare the relative productivities of various word-formation patterns at a given overall corpus size *N*, we can compare their relative vocabularies $V_w(N)$, or we can compare their $\mathcal{P}_w(N)$ values. Estimates of productivity on this basis suffer from their dependency on the sample size, however. It is evident from Fig. 2.4 that at low values of *N* the formation pattern in (a) would appear most productive, but that at larger sample sizes it is overtaken by (c). Although at large sample sizes $\mathcal{P}_w$ gives a better measure of the ease with which new words can be produced than V_w, it is defined only for a given, arbitrary sample size; it, too, is a measure of productivity tied to a specific corpus of a particular size, rather than an abstract measure that could be built into a stochastic model of language production. It is thus not strictly a parameter of the abstract linguistic system (*langue*) but rather a measure of forms that have been produced in a particular (hopefully representative) communicative context.

Thus, estimates of productivity based on finite corpora do not really quantify the linguist's intuitive notion of productivity. This was evident in the plausible intuition

that the prefix *un-* is similarly productive for verbs and adjectives. For the verb-formation pattern, Baayen & Renoulf's data on formation of reversative verbs in *un-* show that in moderately sized corpora this pattern appears to have considerable $\mathcal{P}$-type productivity, but that as $N \longrightarrow \infty$, V_w asymptotically reaches a constant value. For adjectives formed with *un-*, on the other hand, V_w may be virtually unbounded. Insertion of *un-*verbs, in constrast to the adjectives, is thus probably not a spontaneous, generative process. These verbs may well come from a large, static vocabulary of items, many of which have very low insertion probabilities. Only the adjectives formed with *un-* appear to require a set of recursive productions, e. g., $Adj \longrightarrow$ *un-* Adj; $Adj \longrightarrow Noun$ *-ish*; $Adj \longrightarrow Verb$ *-ing*; etc.[8] The difference in the growth curves for verbs and adjectives formed with *un-*, however, only becomes evident in extremely large corpora.

A more suitable measure of the abstractly given, generative productivity of a pattern *w* would need to be based on the number of potential types S_w that would, in theory, be found after sampling an infinite corpus. Fortunately, Kalinin's equation (9) permits an objective, statistically based estimate of the degree of productive potential of each word formation pattern, at least in principle. Just as it can predict total vocabulary $V(N)$ for a corpus of arbitrary size N, (9) can also be applied to a corpus consisting of just those tokens belonging to a given word-formation pattern *w*. Assume we have available a suitably large corpus of N_0 words conforming to the pattern *w*, with the rank-frequency distribution $N_{0\,w,j}$. In this sampling we find $V_w(N_0)$ types built on the pattern *w*. Then (9) predicts a value for the vocabulary $V_w(N)$ to be expected in an arbitrarily large sample N. For $N \rightarrow \infty$, this vocabulary can be regarded as equivalent to the number of potential words S_w that can be constructed on the pattern *w*, i. e., $V_w(N) \approx S_w$. The same possibility is suggested by the curves of Fig. 2.4, which could be fitted to appropriate polynomial expressions that, in the case of curve (a), would approach a finite limit, while (c) might increase without limit.[9]

It is evident that for an aribitrarily long sampling, a large portion of V_w results from the rate at which hapaxes on the pattern *w* have been introduced. If unrestricted, spontaneous productivity is at work, $V_w(N)$ may well be unbounded for arbitrarily large extents N. Hence, since Baayen's index of pragmatic potentiality $\mathcal{I}_w = S_w/V_w$ actually is more or less the same as $V_w(N)/V_w(N_0)$ when $N \rightarrow \infty$, and since the divisor $V_w(N_0)$ is finite, $\mathcal{I}_w$ may in some cases also be unbounded. $\mathcal{I}_w$ therefore does not appear to be useful as a corpus–independent measure of productivity. However, if the potential vocabulary of a given word-formation pattern $V_w(N)$ is unbounded, the total potential vocabulary $V(N)$, as a superset of $V_w(N)$, must also be unbounded, and for any arbitrarily large corpus size N the quotient $V_w(N)/V(N)$ will be a finite number.

Since $V(N)$ and $V_w(N)$ can be regarded as expressing the potential vocabularies

[8]The limited vocabulary of verbs in *un-* would, of course, also be produced by a rule that restricts the class of possible bases to a finite set, blocking the recursive derivation of further base verbs. Which is psychologically the correct representation is an empirical question, but the result would be the same.

[9]I have deliberately chosen to consider only corpora of arbitrarily large but not infinite size. A mathematical development of the limit expressions for $N \rightarrow \infty$ would be interesting but pointless for the subject at hand. The number of word types on a recursively generative pattern *w* can be large but it is not mathematically infinite, since there are constraints on the length of acceptable words and on the universe of communicatively interpretable word meanings; e. g., *uncheesingfulnessist* belongs to the set of recursively derivable adjectives but, because it is virtually impossible to interpret, not to the set of usable words.

of the morphological pattern w and of the language as a whole ($N \to \infty$), this ratio can be expressed more simply as S_w/S, the ratio of potential words built on pattern w to the total potential vocabulary. I shall call this ratio the RELATIVE PRAGMATIC POTENTIALITY.

As Baayen & Sproat [BS6a] found, and as is predicted by (9), most of the types V_w responsible for growth in a very large corpus will be hapaxes. This means that the relative rate at which new types are inserted, $\Delta V_w/\Delta$, is roughly equal to the rate at which freely derived tokens on the pattern w are being inserted. For very large extents N, $\Delta V_w/\Delta$ gives the relative probability that a stochastic source will issue a non-lexicalized word on the pattern w. Since $\Delta V_w/\Delta \approx S_w/S$, the relative pragmatic potentiality of w, S_w/S can be seen as a measure of the 'activity' of the word formation pattern w, independently of the sample and of the number of already lexicalized types. A pattern like prefixation of verbs with *un-* that has produced a large number of lexical items will have a large attested vocabulary V_w, and in smaller corpora it will show appreciable growth rates $\mathcal{P}_w$ simply because of the large number of tokens that must be sampled before all types are seen. But the low value of $\Delta V_w(N)/\Delta N$ obtained in a very large corpus reveals that verb prefixation in *un-* is seldom being used 'on-line' to create new forms. For such patterns, and indeed for many morphological types, $\Delta V_w(N)/\Delta N$ will be close to zero, revealing that most of the types found in the finite corpus are not being produced spontaneously but are instead being fetched from the static lexicon. Other patterns, such as nominalizations in *-ness*, reveal a contrasting kind of productivity: a large attested vocabulary V_w is accompanied by a large number of hapaxes even in the 80 million token sample of the *Times* corpus, indicating that the value of S_w/S will be appreciable. It is thus correct to say that verb prefixation in *un-* has a high extent of use, but a low degree of relative pragmatic potentiality; while suffixation in *-ness* has both a high extent of use and a high degree of relative pragmatic potentiality.

It is important to observe that the sum of the S_w/S values over all morphological types will necessarily be one, since the arbitrarily large vocabulary S is the sum of the vocabularies of all its morphological types S_w.[10] The sum of the S_w/S over w thus gives a well-defined probabilistic distribution that allows relative comparisons among, and a uniquely defined ranking of, the relative pragmatic potentialities of all the available word-formation patterns.

Unlike the growth rate $\mathcal{P}$, based on finite values for V and N, S_w/S allows us to define a stochastic, generative word production mechanism for each morphological pattern, independent of any particular sampling, that inserts lexically unattested forms at a *fixed rate* into the stream of words issuing from the lexical interface. Thus, at least within the accuracy allowed by equation (9), the values S_w/S can legitimately be regarded as part of the abstract, synchronic description of the competence system of a language rather than as values dependent on the contingencies of linguistic performance. Most importantly, each S_w/S is an empirically defined lexical parameter

[10] The stochastic insertion probability of each word formation pattern $P_w = S_w/S$, and the total vocabulary S is the sum of the vocabularies of the w word–formation classes; i.e., $S = \sum_1^w S_w$. Hence,

$$\sum_1^w P_w = \sum_1^w \frac{S_w}{\sum_1^w S_w} = \frac{1}{S}\sum_1^w S_w = 1$$

that describes more than the set of attested lexical items; indeed, many of the words whose frequencies it describes exist only in a virtual sense, since they are not part of the sample from which S_w/S is calculated. S_w/S must therefore be associated with the morphological patterns *w*, rather than with individual types, i. e., lexical entries. While the lexicon can be described as a set of attested words and morphemes, to describe insertion of words from the lexicon, we evidently need more than this lexicon alone.

2.10 Requirements Deriving from Lexical Statistics

The statistical findings examined here have far-reaching implications for the set of possible models of the language processing system. First of all, they show that queries to the lexicon are not evenly distributed among the lexical entries. This suggests that the lexicon might be optimized in some way to allow quick access to small numbers of frequently used words. Carrol's explanation of the Zipf distribution (10) suggests that the highly frequent words may be in some way undifferentiated, perhaps in terms of semantically distinguishing features. Information theory also predicts that the frequently accessed words are likely to bear relatively few features contributing to the meaning of a message. Results from coding theory lead to a similar suspicion: given that words in natural languages do not compactly fill the space of possible phonetic or orthographic symbols, sets of "selector items" of low information content may be present in natural languages in order to maximize their overall information density. Later on I shall argue that these and other pieces of evidence justify an outer-level requirement that partitions the lexicon into two or three functionally separate modules ranked in order of their abilities to represent more or less specific semantic information.

Second, we have seen that some morphological patterns are not merely generalizations over subsets of the lexicon. In a stochastic word-insertion process, the frequencies with which most words appear can be described by probabilities associated directly with each word. In the case of words which appear with very low frequency, no precise insertion probability can be derived for a particular item, but its general morphological pattern can be associated with an empirically well-defined probability of occurrence expressed by the relative pragmatic potentiality S_w/S of the word formation pattern. This suggests that, in addition to the attested lexical items with their associated probabilities, the lexicon — or more exactly, the module responsible for lexical insertion — must also contain a separate tabulation of word formation patterns, independent of the list of lexical items, responsible for the non-attested items entering a sampling of the language. A further inner-level requirement will be for a formalism that describes the rules of word formation and can also generate the frequency-ranked distribution of words in corpora of arbitrary size. The well-defined statistical behavior of these rules suggests that they may operate separately from the far less predictable rules of sentence generation.

At this point it is too early to consider how these requirements might be translated into formal data structures and algorithms, but it is evident that a static lexicon organized as an unstructured set is too simple, and that in some cases lexical insertion must involve more than merely substituting items from the lexicon into appropriate positions in a syntactic structure. The statistical data on rank-frequency distributions thus give

us some ideas about how lexical storage as a whole is organized, but they tell us little about how individual lexical items are represented. For further clues we shall next turn to timing and word association data that have been gathered in psycholinguistic experiments.

Chapter 3

Word Access and Representation

3.1 Overview

We have seen that the statistical properties of word-frequency distributions in large corpora are hard to reconcile with a description of the mental lexicon as an unstructured, static list of words. Theoretical attempts to explain the statistical distributions suggest that word types may need to be differentiated according to size and information content. In addition, the classes of productivity that can be found in large text corpora are reflected in the distinctions linguists draw among morphologically simple and various kinds of morphologically complex words. Yet the statistical studies do not tell us where or how these properties of words might be represented in the mental lexicon, nor do they indicate how individual lexical items are internally structured. Seen as requirements for a cognitive model, they provide some important constraints on possible ways of representing the lexicon, but they do not yet give us a clear picture of the data structures that will be needed to represent lexical items, nor of the operations we may need to define over these structures.

It would be useful to know more about the requirements the statistical differences among simple and morphologically complex word classes might impose on an eventual representation of the lexicon in terms of its formal data structures. Is extent of use, for example, simply a property of the communicative utility of an item? If so, its cause would lie in some source outside the lexicon, responsible for selecting words, and the lexical representation of an item would not include frequency as a feature of the item. Will information content, defined by coding theory as a function of token frequency and predictability, correlate with the size of the data structure required to represent a lexical item? If so, it may be necessary to define classes of data structures according to the amount of information that must be represented. This is by no means a necessary consequence, however, for words referring to complex ideas may well be represented in the lexicon by simple pointers to structures lying elsewhere in the cognitive system. A third question is whether morphological complexity is an atomic feature that can be attached to an item, or does it, too, require a corresponding complexity in the data structure used to represent the item? If this is the case, lexical entries might need to vary widely in representational complexity as well as size, and the complexity might be related to a frequency feature.

For linguists there has never been any doubt that languages do distinguish various

classes of words according to mode of use and structural complexity, but structural analyses of languages have not been able to answer these more particular questions about words' mental representations decisively. In the case of frequency, for example, words do not present specific structural or distributional features — e.g., specific affixes or syntactic properties — that would show how often they are likely to be used. Yet the fact that samples of text from a language show (within broad limits) a reproducible spectrum of word frequencies implies that word frequency information is being represented somewhere in the generating system. Likewise, linguists' structural analyses of words' morphological structures do not necessarily predict the words' mental representations, for it is not always the case that the morphological constituents of a word determine the word's meaning or grammatical properties (as in an opaque and syntactically ambiguous word like *enjoining*), suggesting that in some cases the structure evident to linguistic analysis may have no correlate in the word's mental representation.

In attempting to gain a better view of the internal organization of the language processing system and of the lexicon, in particular, a great deal has been learned by paying close attention to how people respond to various kinds of queries to the lexical system, applying techniques developed in experimental psychology for studying the perceptual and memory systems. The raw data from these studies are not in themselves very revealing, but like the word-frequency data examined in the last chapter, the psycholinguistic data also invite explanations in terms of a structured data base of code symbols. Many of the results obtained in the last 25 years strongly suggest that words' frequencies are in fact represented in the mental lexicon, that those words which linguistic analysis show to have complex structure, such as compounded, inflected or derived forms, are in fact often — but not always — stored in some correspondingly structured representation, and that the lexicon does in fact distinguish its representations by the amount and kind of information they carry. Moreover, some findings give evidence of specialized index structures like those employed in data base systems.

The following sections survey a sampling of the experimental literature on word recognition in order to gather further outer-level requirements for a model of word processing. Although many of the experimental results allow more than one interpretation, there seems to be much evidence that morphologically complex words are not necessarily stored and recognized as complex structures, but that they often are. This evidence is most convincing in the case of inflected words, where it appears that the inflectional morpheme is often recognized and processed separately from the stem. Some prefixes that look like derivational morphemes, the so-called pseudo-prefixes, appear to be recognized separately from their bases only during early processing. On the other hand, there is evidence that genuine derivational morphemes can remain attached to their bases during the early stages of recognition, but at the level of word meaning they appear to be represented separately. The final sections of this chapter survey three recent attempts to construct processing models that might account for these data.

3.2 Experimental Assumptions and Paradigms

Experimental studies of perception by and large share the assumption that there is a layer of autonomic processing prior to conscious awareness and deliberate analysis of sensory stimuli. In visual perception, for example, familiar optical illusions demonstrate that we cannot consciously control how our visual system analyzes many patterns. We cannot see the box of Fig 3.1 both from the top and from the bottom at once, even when we know that both interpretations are valid, just as we cannot correct our misperceptions of M. C. Escher's enticing illusions by conscious effort. In perceptual psychology this automatism has been called the "immediacy of interpretation" [JC87, 40].

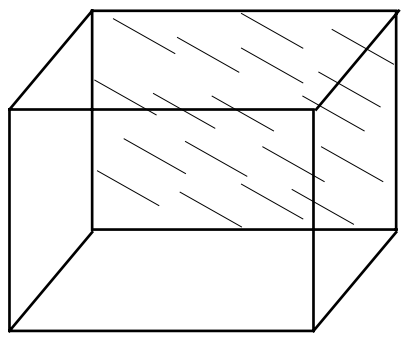


Figure 3.1: Ambiguous Box. Blink and look again!

Nevertheless, it is not always easy to identify the level at which an immediate interpretation arises. Studies of color perception have shown, for example, that in some cases the color context of an object will influence the color we see, so that a white box on a green background appears to be tinted red. This illusion appears to result directly from the object–background contrast, and must arise at an early stage of processing. But in other cases, it is our higher-level knowledge of the inherent color of an object that assigns the color we perceive, letting us see an apple as red in a representation where the image of the apple emits no red light at all; the illusion can only arise after the apple has been isolated and classified by a later level of visual processing (cf. [Mar82]).

Experimental studies of word perception are confronted with similar dilemmas. The mere identification of an effect does not always tell a great deal about where in the perceptual system it arises, and in some cases effects can arise from clues that lie entirely outside the range of causes one would like to investigate. Moreover, it is exceedingly difficult to control all of the experimental factors that lead to a given result, and many factors, such as the way in which instructions are given to subjects in an experiment, can induce effects lying entirely outside the cognitive system that the experiment is meant to examine [RS93, 96-100]. It is these uncertainties that still pose the main challenge to the experimental investigation of word recognition.

Until recently, much of the psychological work on word processing has been reluctant to draw on the more elaborate constructs of linguistic theory as a basis for structuring or evaluating experimental findings, and many linguists (e.g., Zwicky, cf. p. 16) have encouraged this attitude by emphasizing the abstract and non-psychological

character of linguistic analyses. Nevertheless, as was pointed out in chapter 1, any attempt at constructing a model must begin with an initial and inevitably somewhat arbitrary factorization of the observational data. A maximally simple factorization of the lexicon could insist that lexical entities are represented as sets of attributes including orthographic and phonological form, syntactic features and semantics. On this view, a lexical entry would represent a word as a single structured entity resembling a single data-base record, and recognition would be a single operation consisting in no more than a match between an input pattern and a key in the data-base record. At the other extreme, it can be argued that an adequate representation of the lexicon requires a complex, network-like structure, in which lexemes can at best be identified as sets of relationships among relatively independent items representing phonological, orthographic, syntactic and semantic entities of various kinds, and in which word recognition involves complex interactions among these. In the psycholinguistics literature, various simple views of the lexicon have been constructed around the notion that most words are stored and recognized as atomic wholes, and that any internal structures in the lexical entry are not involved in recognition. Contrasting with this FULL LISTING HYPOTHESIS (FLH), the MORPHOLOGICAL DECOMPOSITION HYPOTHESIS claims that words are stored and recognized as strings of smaller parts; in extreme versions, it states that many words actually have no representation at all but are processed by assembling meanings from their smaller, constituent morphemic segments.

On one maximally complex view, the matching of a word's perceived form to its lexical representation could involve many processing stages and modules. Orthographic recognition could be preceded by a stage that identifies each of the characters and then matches these one by one to the lexical representation. Alternatively, matches might well be based on the outline of the word's shape as a whole, disregarding individual characters altogether, or the pattern matcher might seek patterns of curves and lines that do not necessarily coincide with the individual characters. It might also be the case that only as many characters are compared as are needed to distinguish the percept from all similar strings in the lexicon, and the comparison might proceed from the left edge of the word, from the right, or from the extremities toward the center of the word. Yet another possibility is that the word could be recognized in pieces, by identifying groups of characters or particular character-to-character transitions and subsequently matching word segments to the lexicon. Different segments of the word might in fact be recognized in different ways, perhaps taking advantage of internal morphological structure when it is available, or using "selector prefixes" like those used in Huffman codes (section 2.6.2). Similar alternatives can be entertained for auditory recognition, although the speech signal makes left-to-right processing at the lowest levels inescapable.

Just what constitutes recognition is another issue that may lead to a complex rather than a simple specification of the lexical data structures. At the interfaces to higher levels of processing like syntactic and semantic analysis, it is evident that various forms of information become available, such as the syntactic and semantic attributes involved in sentence analysis. These do not necessarily become available simultaneously with the phonetic or orthographic representation, and there is no reason a priori why they would need to be stored in one unitary structure. Moreover, it is evident that decisions about what is or is not a word can be ambiguous. Speakers can form independent and even contradictory judgments as to category, meaning, and well-formedness of

a word. Recalling the nonce formations found by Baayen & Renoulf in the London *Times* (Table 2.5), it is likely that speakers would reject *oftenly* as a known word, yet they could identify it as an adverb and assign it a meaning.

If the various components of the word-recognizing system are linked in a serial fashion, they may reveal themselves by making intermediate results available at various times and under varying contexts during recognition experiments. However, it is also quite likely that many subprocesses run in parallel, making their results available at overlapping times and thus possibly masking one another. This means that the failure to find direct experimental evidence of a postulated process cannot exclude the possibility that it exists. From the standpoint of a computational requirements analysis, the experimental data from word recognition experiments are thus unlikely to reveal in any directly obvious way which data structures and processing algorithms ought to be incorporated into the recognition component of a simulation of word processing; this can only be accomplished at the point where all inner and outer-level requirements are integrated by creating a computational specification. However, within the large space of possible models, experimental data about word recognition can restrict the range of plausible data structures for the representation of the lexicon and of the processing mechanisms that operate on them.

Much of the psychological evidence now seems to suggest that word representation involves at least three sets of data structures. At a level closest to the sensory input there appear to be pattern classifiers that serve only to establish links from the orthographic or phonetic representations of words or word segments to a more abstract level of representation containing combinatorial attributes like those marking words for lexical category (as nouns, verbs, prepositions, etc.) and those marking variable grammatical features, (singular, past, passive, etc.) At a more abstract level there is some evidence of data structures storing semantic information that can either be atomic or internally structured in a way that mirrors the morphological structure of the word. Finally, these same results may imply that semantic information is specific to the lexicon and is stored separately from the general conceptual knowledge that serves other modes of cognition, like visual recognition, planning, or reasoning.

These observations make it worthwhile to consider consider what some conceivable, maximally complex factorizations of the data might look like, rather than insisting (as has much of the psychological literature) that the experimental evidence be interpreted with reference to the simplest possible assumptions. Because the available experimental techniques rarely allow decisive answers to particular empirical issues, it might turn out that, to account for the experimental data, we must find a global best fit to some complex factorization rather than choose one of a number of simple alternative models.

3.2.1 Experimental Paradigms

The experimental results currently available on word recognition and interpretation are often difficult to understand and seem in many cases to have contradictory implications in terms of specific modeling requirements. To gain some perspective on them, I shall review some of the experimental methods that have been used to investigate the role of morphology in word recognition (mainly visual). To understand what the experiments might be revealing, it is useful to have in mind a tentative reference model of how

word recognition might function, although we shall later see that this model needs much elaboration.

A simple, early hypothesis about word recognition advanced by J. Morton was that the lexicon included for every word a pattern recognizer that could be triggered by the presence of the word in the visual or auditory perceptual field. Morton called the word recognizers "logogens", and he postulated separate sets of logogens for the visual and auditory systems [Mor79], each set indexing a single central lexicon, where each lexical item's meaning and grammatical attributes were stored, as shown for the visual and auditory recognition systems in Fig. 3.2. In this model, visual recognition of a word takes place by presenting the image of a word to all of the logogens at once. Those logogens matching most closely the word's shape "fire", sending a signal to the corresponding lexical entry and to higher levels of cognition, where the fact of recognition becomes consciously available. In addition to storing the word image, the logogen has an activation threshold corresponding to the word's statistical frequency. Before the logogen can activate the corresponding central lexical item, the visual stimulus must in some way exceed this threshold, e.g., by being applied to it for a certain amount of time.

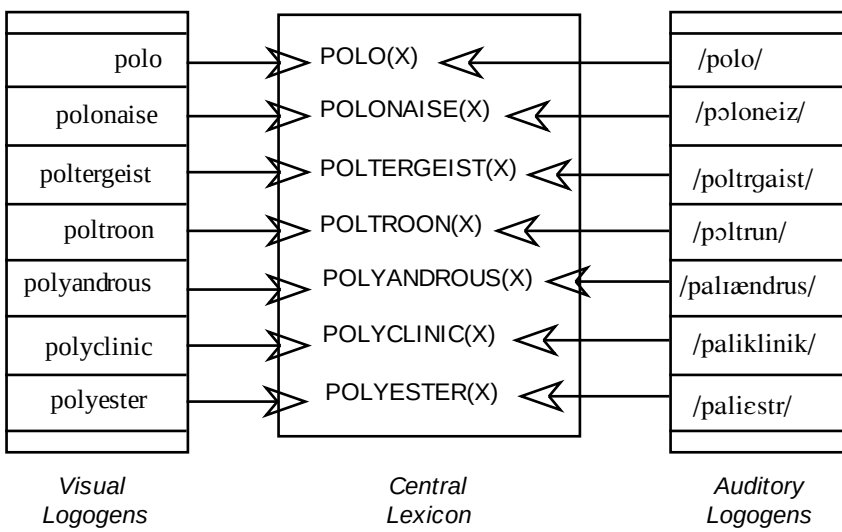


Figure 3.2: Logogens and Central Lexicon.

An important property of the logogen model is that, because of the varying activation thresholds of the recognizers, the images of infrequent words will take longer to trigger the central lexicon than frequent words, and their recognition can be more easily hindered. An effect of this sort has been confirmed repeatedly in experiments, and it demonstrates nicely that corpus-statistical frequency, as described in the previous chapter, is indeed physically represented in individual items of the lexicon.

The recognition times described by the logogen model are defined for single words shown in isolation, but the frequency effect is not confined to this controlled situation. In uncontrolled reading, the amount of time during which the reader fixes a word in the visual field is a function of the size of the word and of its frequency, and it is little influenced by the surrounding words; thus, *thermoluminescence* fixes the gaze

much longer than *glow*, regardless of the sentence context[JC87, 46]. On average, the fixation time per word is about 30 ms per letter of length, with up to an additional 170 ms as an inverse function of the logarithm of the word's corpus-statistical frequency [JC87, 72].

Lexical Decision and Naming In the experimental paradigm known as the LEXICAL DECISION TASK, words are presented one after another, in isolation, as visual or auditory stimuli. Sometimes they are also embedded in a short context of other words to provide a less artificial recognition environment, e.g., "The next word is ...". At some point after a genuine word has been presented, subjects become conscious that they have recognized the word and they are able to report that it is known to them, usually by pressing a button. To ensure that identification has actually taken place (rather than a response to spurious cues, such as a tell-tale pattern of letters or intonation), similar-looking or sounding non-words are interspersed among the words under investigation, and these must be classified as non-words by pressing a second button. In some cases, subjects will discover common features of the genuine words and start to identify them more quickly than they could without the additional clues [Har95, 80], but by varying the proportion of non-words to words, control studies have generally been able to factor out such "strategy" effects.

A number of factors besides corpus-statistical frequency are also known to influence lexical decision time. As a rule, the more similar a non-word is to a known word, the longer it takes to reject it [San94]. Semantic abstractness vs. visual concreteness [Dre89, 84], ease of pronunciation, and syllable count [Har95, 76] are also known to influence recognition time. Morphological complexity, too, can play a large role, which has made lexical decision an important tool for studying the psychological representation of word structure. While artificial, the lexical decision task's psychological validity is strongly supported by the finding, mentioned above, that fixation times in the more natural context of uncontrolled reading are described by the same inverse logarithmic function of corpus-statistical frequency.

Alternatively, in the NAMING TASK subjects can be asked to say the word rather than simply to press a button, which provides a more sensitive test in situations where the experiment needs to ascertain that the word has been identified correctly. Reaction time is measured to the start of the spoken word. However, the time it takes to access and activate the pronunciation of the word introduces a further variable that also needs to be controlled, namely the time required to access the pronunciation and initiate speaking (cf. [Lev89]) Nevertheless, the naming task is regarding by many experimenters as giving a more reliable indication of the true amount of time required to recognize a word (the RECOGNITION LATENCY) than the decision task [Har95, 81].

Repetition and Cross-Modal Priming When a light is flashed, the image we see does not disappear at once; the neural activation rather decays as a function of time, producing an afterimage. A somewhat related effect is presumed to affect word recognition tasks. If a word is presented (visually or spoken) once (to PRIME the sensory system) and is then repeated shortly thereafter (as a TARGET), the recognition time for the second presentation is shorter than for the first, presumably because it is facilitated by additional input from the prime's 'afterimage'. This is not surprising, but the effect

is also obtained when the prime is only *similar* to the target in one of several ways. This makes the priming paradigm valuable for investigating recognition. By altering the prime in various ways, one can find out which features of the target are important for its recognition, and which are irrelevant. Among the factors known to contribute to priming of word targets are semantic relatedness, as well as superficial orthographic and phonetic similarity [Dre89, 142]. Moreover, certain variants of the prime can have an inhibitory rather than a facilitory effect: they make recognition of the target slower than it would be with no prime. Such primes are of course more difficult to interpret, since it is inherently unclear whether the processes needed to recognize the target have been inhibited or whether other, competing targets have been facilitated.

Repeating a stimulus that is merely varied with respect to the features under investigation is called REPETITION PRIMING. With repetition priming it is difficult to determine what stage of recognition is facilitated by the prime, since both prime and target are processed in the same way. By contrast, it has been argued that the method of CROSS-MODAL PRIMING can exclude priming effects at early stages of recognition by presenting the prime stimulus through a different sensory system [MWTWO94]. For example, a lexical decision can be primed by showing a picture of the object that the target word names. If a lexical decision is facilitated or hindered, it stands to reason that the effect probably arises in that part of the lexical representation that the picture shares with the orthographic or phonological form of the word, namely the abstract, semantic or conceptual representation. If so, one would expect that the cross-modal facilitation effect would be less pronounced than repetition priming, and this is in fact what is most often observed. The cross-modal task thus offers unambiguous access to a deeper level of representation than do any of the intra-modal tasks.

Gating A recognition task that has been mainly used for auditory recognition is the GATING TASK. Here only a fragment of the word to be recognized is presented at first. Then increasingly long fragments are presented, and the subject is asked to report what he or she thinks the word might be. Alternatively, the task is to report when the word can be recognized with certainty. The resulting IDENTIFICATION POINT in the sound stream is generally found to correspond to the point at which the word becomes uniquely distinct from all others in the lexicon (the UNIQUENESS POINT), when compared from left to right against the entire lexicon. The progressively smaller set of possible recognition alternatives to the left of the uniqueness point has been called the word's COHORT in the well-known model of W. Marslen-Wilson [MW92a]. Gating experiments thus seem to show that the auditory word recognizer works from left to right and does not necessarily need the full word pattern.

In most gating studies the fragments have been constructed by starting at the beginning of the word, but it is also possible to obtain recognition from fragments that do not include the very beginning [RS93, 198]. The word recognizer evidently can reconstruct the full stimulus once the identification point is reached. However, it appears that information toward the end of the word is much easier to reconstruct than information at the beginning.

Phoneme Monitoring In auditory recognition, if subjects are told to press a button as soon as they hear a certain phonemically specified sound in a word or a series of

words, it has been found that the time required to respond is not uniform, and that it can be influenced by context. In particular, if the word has already been recognized by the time the sound is heard, the response is faster [SZR91]. This PHONEME MONITORING TASK thus appears to give indirect clues about how individual phonetic segments are being processed, and it also can be used to identify the recognition point of a word without the artificial series of increasingly long repetitions required in the gating task.

Aphasias On the assumption that the modular structure of word processing is reflected in a modular distribution of tasks to different areas of the brain, language impairments arising from strokes, tumors and brain injuries can provide a much different source of information about how words are processed. Certain brain injuries, for example, have been associated with a loss of the ability to connect utterances with corresponding conceptual data; most striking are cases where pictures of objects cannot be connected with their names, although the names themselves can be used in phrases and sentences. The opposite sort of pathology is also found, in which naming ability is intact, but the ability to produce and analyze larger groups of words is impaired. These disturbances have been taken as evidence, albeit controversial, that syntactic and conceptual processing of language are localized to specific areas of the brain [FF90, chapt. 5.6]. Injuries affecting the ability to read complex words and non-words have also been reported; two relevant examples will be mentioned later.

Analysis of Speech Errors Studies of speech production have also been used to gain insight into how linguistic processing is organized. While there may be significant differences between production and recognition, some of the results from studies of production are undoubtedly relevant for a model of recognition as well.

A traditional dogma of structuralist linguistics has been that the sounds making up a word are arbitrary and unrelated to the word's meaning. A type of speech error that seems to belie this "arbitraire du signe" (Saussure) is known as malapropism from the ebulient Mrs. Malaprop in R. B. Sheridan's *The Rivals* (1775), a character who substitutes words that are similar in sound and often comically related in sense to those she evidently means to use. Modern studies of speech errors assume that utterances are formed first as abstract representations, and that they are assembled at some stage prior to speaking into a chain of commands to the speech apparatus. The kinds of entities that get misplaced in speech errors must exist as representational units at some point in the language producing system. This has been taken as evidence that the semantic connection between words that sound similar is psychologically real [Fro71]. A variant of the naming task can be used to induce speech errors experimentally, by asking subjects to name items or reformulate expressions quickly. When speakers commit errors, they do not produce arbitrarily different phonemic strings. Instead, they produce a word that sounds similar to the intended word, except that a phonemic segment is suppressed, inserted or exchanged. Looking for segments that speakers exchange or delete when they commit errors, it should be possible to glimpse the actual linguistic building blocks that are being assembled in order to direct the spoken utterance. Likewise, the complete absence of errors in stretches of an utterance suggests that no assembly is taking place, and that the corresponding portions of the responses are being drawn directly from memory or from lexical storage. Thus speech errors are

thought to show where words are being assembled from smaller units and where they are being processed as unitary wholes.

3.3 The Representation of Word Frequency

As mentioned in section 3.2.1, the correlation between words' corpus-statistical token frequencies and the speed with which they can be recognized is one of the most well documented findings in the experimental literature [Ste95]. In the logogen model, this effect is explained by attaching to each logogen a recognition threshold that can vary within a finite range. An equivalent explanation places the items in some sort of ordered structure in such a way that the frequent words are in some sense at the top of the lexical heap and most quickly found. Words with vanishingly low frequencies and non-words presumably then require searching the entire lexicon before they can be rejected, and so their recognition time ought to set an upper limit for recognition time in general. Logically, it is simply not possible to search more than the whole lexicon.

That this is not the case was reported by M. Taft and K. Forster in [TF75] and followed up in [TF76], and their experiments have been the point of departure for a large number of investigations by other workers. Taft & Forster found that non-words containing orthographic substrings corresponding to real words and parts of words, such as "de" and "juvenate" in **dejuvenate*, take even longer to reject than simple non-words of the same size having no recognizable subsegments. They took this as evidence that lexical search does not stop with the search for a match to the entire string; rather, it appears that substrings are also accessed, and if found, the recognizer takes additional time trying to integrate them into the whole string before the word can be definitively rejected.

A second finding in these studies was that the lexical decision time to reject a (visible) string that could be an initial syllabic segment, such as "plat" in *platform*, was longer than for strings that cannot be identified with a syllabic segment of some larger word, like "brot" in *brother*. This seemed to indicate that non-lexical strings like "plat" must nevertheless have some sort of mental representation that was being accessed and therefore had to be evaluated before the whole string could be rejected. Similar units at the end of a word, such as the string "cule" in *molecule*, did not show this effect, suggesting that the initial segment was being used as a kind of access key to the full representation, but that such keys were not present at the ends of words.

Whether or not such a coding function is the correct explanation of the effect, an inescapable implication is that the presence of internal structure in the orthographic word has a measurable effect on recognition time. It suggests that the units of prosodic or morphological analysis used in descriptive linguistics might have more than descriptive utility; that they are in some sense psychologically real and are part of the physiology of word recognition. Although Taft & Forster described the sub-word units they investigated in [TF75] and [TF76] as morphemes and "abstract syllables", they did not identify significant differences between prefixes (or pseudo-prefixes) in derivatives and initial constituent words in compounds. Their test data show that they were really investigating something different from morphemic units, namely substrings that recur in the lexicon, without regard to meaning, and without much regard for linguistic structure. Later, in fact, they coined the term Basic Orthographic Syllable Structure

(BOSS) to describe the postulated access units, which are perhaps best thought of as recurring patterns used as keys in a pattern matching procedure [Taf79a]. Other investigators, e.g., [San94], [Dre89], have objected that the BOSS is an ad hoc construct. There appear to be no reliable criteria, apart from the experimental results themselves, that define what can and cannot be a BOSS. Thus, according to Taft [Taf79b] *lik* is the BOSS for *like* and *likable*, but *scoff*, which is the morphological root of *scoffer*, does not function as a BOSS, because it does not delay the classification of non-words.

One effect found in [TF76] will be of some relevance later (chapter 6) for specifying a segmentation procedure. Taft & Forster investigated whether the segmentation of the initial BOSS was aided by the presence of letter transitions that would not occur within a word or morpheme, such as the transition from "t" to "m" in **footmilge*. If non-words having easily recognized orthographic segment boundaries can be rejected more quickly that those having unclear internal boundaries, e.g., **trucerin*, it would appear that the orthographic transition probabilities (such as could be described by low-order Markov sources) are being used to mark segments before the lexicon is checked. But in fact Taft & Forster found no such advantage, and they inferred that segmentation of the BOSS from the rest of the string is entirely under the control of the lexicon: starting from the left edge, the word is submitted letter by letter to the lexicon until a BOSS is found, without a prior, heuristic attempt at segmentation.

Thus it appears that BOSS prefixes must be represented explicitly in the mental lexicon, even if they do not correspond directly to units of traditioal linguistic analysis. Like the Huffman code prefixes discussed in chapter 2 (section 2.6.2), they might aid the recognition of word boundaries, and they might function as meaningless segments used to optimize the coding of the lexicon. If used in this way, the resemblance of pseudo-prefixes like the *re-* in *rejoice* to genuine morphemes (like the *re-* in *rebuild*) would be historical accident, probably resulting from diachronic processes that have so shifted the meaning of the word that the pseudo-prefix's relation to the corresponding morpheme is now obliterated. Segments not corresponding to any known morphemes could in principle serve just as well as pseudo-prefixes, but they would have to be introduced deliberately and artificially. They would be recognized as independent coding segments, but not as morphemes, since they would have no meaning and no function other than that of extending the lengths of individual words of low frequency so that shorter phonological strings remain available for words of high frequency. This appears to be at least a plausible hypothesis, and it offers theoretical advantages for word recognition that will be elaborated in chapter 6, but I have seen no pertinent findings in the statistical or experimental literature. Other investigators, however, have pursued the idea that the recognition units Taft & Forster discovered might be more closely related to the morphemic units of traditional linguistic analysis.

3.4 The Representation of Morphological Structure

The analysis of words into constituent units has been one of the central topics in descriptive linguistics, and the principle observations have remained more or less constant. These have included a division of word-forming patterns into COMPOUNDING, in which freely occurring words are concatenated to form a new entity; DERIVATION, in which a freely occurring word is given a new meaning and sometimes a new lexical

category by adding additional material, typically at the beginning or end, that cannot be used freely as an independent word; and INFLECTION, in which grammatical attributes of a word are changed without an accompanying change of basic meaning or category, by adding to or altering a segment of the word. As a rule, word structure is described as being built up from a ROOT carrying the most specific semantic content. Derivational affixes, if present, are added at the periphery of the root, giving a STEM; and inflectional affixes are added outside the derivational affixes (cf. [Kip82b]). Some important exceptions to this overall pattern will be mentioned in section 6.3.2 in connection with an analysis of word segmentation.

Roots and affixes together make up the set of MORPHEMES, defined as the "smallest meaningful units in the structure of the language" in the American structuralist tradition [Gle61, 53]. Speakers of a language often have more or less reliable intuitions about the sub-units of meaning within a word and can use these intuitions to identify morphemes. Under the influence of psychological behaviorism, the American structuralists tried to give the morpheme a more objectively verifiable status by defining it in relation to an empirically given sampling from the language, called a corpus, rather than directly in terms of a speaker's intuitions. In many cases, morphemes can be isolated by looking for segments of sounds or characters [1] (or of gestures in signing languages) that can be substituted one for another within the corpus, in such a way that the substitution brings about a corresponding, systematic change in meaning.

If, for example, we search a corpus for words ending in "ful" and "less" we may find *meaningful*, *regardful*, *useful*, as well as *meaningless*, *regardless*, *useless*. Interchanging the word segments "ful" and "less" leads in all cases to a similar change in meaning (from FULL-OF to LACKING). A further search for patterns using yet smaller substrings of "ful" and "less", like "f", "ul" "le", does not uncover any smaller, substitutable units of meaning. This lets us identify the strings "ful" and "less" as minimally meaningful segments, and it allows us to postulate the morphemes *-ful* and *-less*, which can be combined with other morphemes (like *help*) to produce complex words.

Morphemes are not identical with word substrings or surface segments, however. Although the surface segment "less" can also be found in a word like *unless*, "un" cannot be combined with LACKING to derive the meaning of *unless*, and the complementary substitution produces a non-word **unful*. Both observations indicate that in this case the substring "less" does not correspond to a morpheme. A single substring can also represent different morphemes, as does the "s" in *girls* (plural) and in *finds* (third-person, singular, present). Thus, in the structuralist terminology, morphemes are different from lexical substrings, and morpheme-like substrings that are not genuine morphemes are called *pseudo-morphemes* or in some cases *cranberry morphemes*, from the meaningless substring "cran" that superficially appears to contribute to *cran-*

[1] The modern linguistic definitions are usually formulated in terms of phonetic rather than orthographic structure. Especially in English and French, the orthographic systems carry a great deal of historical ballast that is opaque to most speakers and must be learned by rote, with conscious effort, and at a later age than that in which the main stages of language acquisition are active. Thus, some portions of the orthographic rule system (like the spelling heuristic for English, "i before e except after c") may not be learnable in the way languages are learned, and if so, they are not rules of the linguistic system but are probably acquired by general cognitive intelligence in the way that, e.g., number systems are learned. Nevertheless, since the KLU simulation is concerned with text rather than speech processing, I shall by and large restrict the discussions of morphology here to orthographic analyses.

berry what *blue* contributes to *blueberry*.[2]

Often a number of variant substrings appear to carry exactly the same meaning, as with the plural endings "s" and "es" in English. In such cases, the variant forms are termed ALLOMORPHS of the plural morpheme. What constitutes a legitimate variant of a morpheme is a far-reaching issue, however. If *-d* and *-ed* are allomorphs of the past tense morpheme in English, it would appear reasonable to consider the past vowel *a*, *saw* (past of *see*) as a further past-tense allomorph, but for reasons that will become clear later on, I shall want to exclude this possibility. For the present discussion, only segments that are in the same distribution (occur in the same segmental context) will be described as allomorphs.

Apart from providing some terminology for the following discussions of experimental results, the terms and structuralist techniques mentioned here are of particular interest for a computational simulation of word formation and analysis. Because they are based, at least ideally, on the somewhat mechanical analysis of a corpus, they allow a systematic and traceable development of the formal, inner-level requirements for a morphological component. At the same time, they automatically define a test of correctness (in the sense defined in chapter 1) for an implementation: the implemented GRAMMAR — by which I shall mean the combination of lexicon and rules — is correct just in case it can be shown to generate (or correctly analyze) the same set of words as are contained in the corpus from which it was derived.

Nevertheless, as a set of generalizations about the units of meaning in a corpus, a structuralist morphological analysis is possibly nothing more than a descriptive heuristic. Its descriptive adequacy does not ensure its psychological reality. There is in such an analysis no necessary connection to the cognitive system responsible for the observed data, and this leaves open the possibility that a valid model would need to implement a second, psychologically grounded set of requirements. Given the enormous difficulty of specifying cognitive models of any sort, it would be a great stroke of luck to find out that the morphological structures which linguistic methods can so easily uncover are the same structures actually used in the cognitive system. Fortunately, there is a great deal of evidence that this is the case.

3.5 Evidence for Separate Treatment of Inflection

In the simplest conceivable model of lexical access, all words are recognized by matching them with templates, as in the logogen model of Fig. 3.2; this model was later called the Full Listing Hypothesis (FLH). While some studies have made this model seem plausible for English, many languages of the world have astronomically large numbers of possible word forms that would require individual logogens. J. Hankamer constructed a morphotactic model for Turkish morphology that produces 2 million verb forms using the available roots and affixes without any recursion; if one level of word-structure embedding is allowed, 27 million acceptable Turkish verbs are produced. For nouns the numbers are even more forbidding, 9 and 216 million [Han92].

[2]M. Aronoff has argued in [Aro76] that some meaningless segments like *mit* in *permit* and *submit* should also be seen as morphemes, since they participate in a common set of morphological relationships, like the nominalizations to *permission* and *submission*. Since there is now no productive derivation on *mit*, it is hard to tell in what sense speakers actually have these units available as separate mental items.

However, two large dictionaries for Turkish, [Rad11] and [Red90] each contain only around a hundred thousand entries, which on a strict reading of the FLH would make them nearly useless. Generally, however, proponents of full listing do not claim that words have no internal structure nor that all possible words are in the mental lexicon; instead, they frequently allow for an additional 'backup' procedure to parse and analyze words *after* they are recognized, or to analyze complex words that are unlikely to be listed.

Thus the real psychological issue is not so much whether complex words can be analyzed at all — there can be no doubt that they are analyzed, at least on occasion — but whether complex words that occur frequently enough to be learned and stored whole in memory are nevertheless broken down regularly during lexical access. To gain some perspective on this issue, I shall examine some representative studies that shed light on this question for inflectional affixes and then for derivational affixes, but it should be emphasized from the start that we cannot expect more than a few hints from these studies as to how the lexicon and lexical recognition must be constructed. The investigative methods of psycholinguistics are subtle, the literature is vast, and there are still large areas of disagreement about many fundamental questions. As for compounding, it is sometimes difficult to be sure which examples are genuine cases of word formation. Since the constituents of a compound are free morphemes, they can often also be combined syntactically, as perhaps in *noun phrase construction* — which presents just such an example, one that looks like a noun phrase in English but in German might appear as *Nominalphrasenaufbau*. Compounding will therefore not concern us here, although some of the psycholinguistic studies have gathered data on compounds, as well.

3.5.1 Lexical Decision Effects from Inflection

Since it has been well established that, for monomorphemic words, lexical decision time is a function of frequency, on the FLH, the decision latencies for inflectional variants of a word should directly reflect the frequencies of the various inflected forms, not the frequency of the stem. Thus, if the form *sized* occurs as often as *raked*, the times required for recognition of both should be equal, regardless of the frequencies of the base words *size* and *rake*. In fact, *size* in all forms (also as a noun) occurs much more frequently than *rake*. With test pairs like this one, having equal surface but widely different base frequencies, Taft [Taf79b] found a significant difference in recognition time corresponding to the frequencies of the bases. The difference in recognition time can evidently only be explained if the high-frequency base word *size* was being accessed rather than the whole-word representation of the unfamiliar suffixed form. This BASE or STEM FREQUENCY EFFECT and results like it that have been reported in the literature seem to indicate that the FLH is wrong, and that words are in some way 'disassembled' prior to recognition.

However, in an experiment that inverted the procedure just described, Taft obtained the opposite result. He selected pairs of inflected words having equal base frequencies but widely differing surface frequencies, like *things* (high surface frequency) and *worlds* (low surface frequency). Here the FLH predicts a corresponding difference in recognition time, and that is what was found. It appears that the high-frequency inflected forms were being accessed directly rather than via the stem. In an attempt

to resolve the discrepancy, Taft also looked at inflected pairs where, as with the pair *things – worlds*, the surface frequencies were widely different, but the cumulative frequency of the words formed on the base, e.g., *thing + things*, was the same. He also examined monomorphemic pairs. Such a pair is *rib – tin*. *Rib* is relatively infrequent, but the sum of the frequencies of all related words, like *ribs*, *ribbed*, and *ribbing*, is the same as the sum for all words related to *tin*. On the FLH, the decision times for *rib* and *tin* should be the same, but Taft found that the recognition time for words like *tin* was noticeably shorter than for words like *rib*. For the inflected pairs, however, the difference in lexical decision time was more significant. From this result Taft inferred that two frequency effects were coming into play. One appears to result from the base frequency, as established by the cumulative frequency of all words that might reference the base during recognition; only this effect appears in the recognition latencies of monomorphemic pairs. With the inflected pairs, an additional, stronger SURFACE FREQUENCY EFFECT arises from the frequency with which the base is found in combination with its particular affix.

For Italian nouns, H. Baayen et al. [BBS97] also found that lexical decision times for singular and plural forms cannot be predicted from the surface (whole-word) frequencies, but that they are also not solely a function of the stem frequency. Unlike English, Italian shows inflection on both the singular and the plural forms of most nouns, so that it is possible to compare the processing of the inflection along the dimension singular vs. plural. In an auditory lexical decision task Baayen et al. measured the decision latencies for nouns like *denti* 'teeth' that are DOMINANT PLURALS, meaning that they are used much more frequently with plural than with singular inflection. These were contrasted with DOMINANT SINGULARS, like *naso* 'nose'. For low-frequency stems, no significant difference was found between singular and plural forms, which is incompatible with the FLH, since only one of the two forms (the dominant) had a high frequency. For these forms, it is the stem frequency that appears to determine recognition time, independent of the frequency of the whole word. However, for high-frequency stems of words that are rarely used in the singular, such as *capelli* 'hair(s)', *gambe* 'legs', *scarpe* 'shoes', the decision times for the rare singular forms were significantly longer than for the plurals, and closer to the decision latencies for the infrequent forms. This is difficult to explain on the analysis hypothesis, which predicts that both forms should have short decision latencies based on the frequency of the stem. Instead, it appears that the singulars of the high-frequency, plural dominant nouns (*capello*) are being analyzed, similarly to the low-frequency nouns, while the common plural forms (*capelli*) are being accessed directly, since only these forms show the shorter latency that would be expected from their high frequency. These results are summarized in terms of representative examples in Table 3.1.

It should be added that the variancies in the recognition latencies are rather high. Especially for the low frequency items, singular-plural differences of 50 milliseconds occur frequently, suggesting that other mechanisms may be at work here besides the ones proposed. But like Taft's result, these data indicate that both full listings and morphological stems are being used as the units of lexical access. Why the full listing effect appears only in the plurals of high-frequency, plural dominant words is not obvious. Baayen et al. offer a somewhat speculative explanation in terms of an additional semantic load imposed by the analysis of plural forms that can be spared when the full-form listing is accessed.

Table 3.1: Representative Recognition Latencies (RL) for Italian Nouns. Dominant form in boldface.

SING.	RL	PLURAL	RL	STEM FREQUENCY
piazza	534	piazze	522	High
dente	527	**denti**	503	High
lampada	554	lampade	553	Low
baffo	597	**baffi**	595	Low

3.5.2 Priming Effects on Inflection

It is well-documented that the decision latency for a target word is shortened if, beforehand, its recognition is "primed" by showing a related word. A shared stem will prime an inflected form (e.g., *car* primes *cars*), but shared orthographic segments do not have a comparable effect: *car* does not facilitate the recognition of *card*. The priming effect can also appear when the similarity is purely semantic, e.g., *nurse* primes the recognition of *doctor*. Thus it is not clear from such effects whether morphological decomposition is involved in priming: the fact that *car* primes *cars* may be related to morphology, but it could also arise purely by semantic association, in the same way that *nurse* primes *doctor*.

To isolate a priming effect due to morphological similarity alone, one might try to prime the inflection separately from the stem. It can be argued that inflectional affixes are primarily involved in syntax; in any case they generally do not contribute to sentence meaning, and it is unlikely that they participate in the kinds of semantic associations found between concepts like *nurse* and *doctor*. A separate priming effect for the inflection would imply that both constituents of the word, the stem and the inflection, were being processed separately. To distinguish the priming effects of the affixes, however, one needs a more elaborate inflectional system than is available for English nouns. German verbs, whose inflections distinguish person, number and tense in the majority of cases, offer enough different forms that it should be possible to identify priming effects via affixes, should they occur. To test this hypothesis, E. Drews [Dre89] established, first, that a visual repetition priming effect like that for nouns is also found for verbs in German. Then she tested pairs of inflected verbs in which the similarity of the prime to the target was varied along three dimensions: semantic similarity, surface form, and morphological tense. A semantic priming effect was evident: as expected, *lernte* 'learned' primed *übte* 'practiced'. Purely orthographic similarity in the inflectional position had no facilitory effect, e.g.,*warte* 'wait' did not prime *rollte* 'rolled'; rather, it delayed recognition relative to primes that had no similarity to the target (*öffnen* 'to open' vs. *pflegte* 'took care of'). Because the experiment contained mainly past tense verb forms, Drews speculated that this delay might arise from a strategy subjects developed of segmenting all words at "te" before trying to identify the true morphological constituents. That is, by a habit induced during the experiment, *warte* would first be segmented as "war" + "te" before the correct analysis into *wart* + *e* was found. A small priming effect was evident for pairs in which morphological

tense was the same (*blühte* 'blossomed' primed *warnte* 'warned'); this effect could not, however, be isolated from the spurious segmentation effect.

However, in another visual lexical decision experiment Drews found strong evidence of inflectional priming for present tense German verb forms, with data that did not invite spurious segmentations. Here she added to each prime an additional subject pronoun, in the expectation that the combination of pronoun plus verb would activate recognition of the morphosyntactic information needed to check the agreement of inflectional features between subject and verb. Targets were single verbs (without subjects) that were similar to the primes

- in agreement and semantics (facilitation 51 ms)
  sie übte $\rightarrow$ *lernte*
- in semantics only (facilitation 31 ms)
  wir üben $\rightarrow$ *lernte*
- in agreement only (facilitation 19 ms)
  sie übte $\rightarrow$ *schaute*
- in neither semantics nor agreement (facilitation 0 ms)
  sie übte $\rightarrow$ *schwimmen*

Relative to the targets matching neither in semantics nor inflection, all of the targets were facilitated to some extent. Most striking is that the facilitation for the combined factors inflection and semantics (51 ms) is very nearly the sum of the facilitations measured separately for semantics (31 ms) and agreement (19 ms). This is compatible with Drews' hypothesis that recognition of inflected words in a syntactic context involves separate morphosyntactic and semantic stages of processing, and suggests that these two stages do not overlap much in time. Interpreted literally, her result appears to require two separate processing modules with an interface feeding results of the morphosyntactic analysis to a module devoted to semantic processing.

3.5.3 Error Effects with Inflection

It is has been established as a general rule that high-frequency, stored symbolic items of all types can be recalled more quickly and more accurately than low-frequency items [SM88, 103]. This means that if a word like *ended* is in common use, by comparison with a low-frequency word like *mended* it will be recognized and produced more rapidly, and errors will be less frequent. It is worth noting that in Taft's 1979 experiment (section 3.5.1) the error rates for recognition of low-frequency plurals like *worlds* were larger than for high frequency plurals like *things*, suggesting that analysis of the less familiar plural form involves steps that can introduce additional recognition errors.

Errors in spontaneous speech have also been seen as a possible source of clues about how the linguistic system is organized. While there is some evidence that the system responsible for language production is not entirely isomorphic to the one involved in passive understanding, the structures of the symbolic units with which both modalities work must ultimately be the same; communication would simply not take

place otherwise. Moreover, the correlation that holds between word frequency and recognition speed holds for word production as well.

In a series of studies J. Stemberger and B. MacWhinney [SM88], [Ste95] gathered counts of production errors that occur in rapid, spontaneous speech. In an English corpus gathered by Stemberger, errors were counted in which speakers produced the "base" form of a verb, meaning a present tense, uninflected form, where the immediate context required a past or perfect form. Dividing the rank-frequency spectrum of English [KF67] roughly in half, they found that the error rate for the low-frequency verbs was about 10 per thousand productions, but for the high-frequency verbs, it was less than three per thousand. From the general correlation of frequency with storage, they concluded that the high-frequency past and perfect forms were probably present in memory and thus less subject to errors that presumably were introduced when the low frequency verbs were inflected. This result did not, however, by itself imply that the low-frequency verbs were not in fact stored, since stored items of low frequency are also recallable with reduced reliability.

To clarify how the low-frequency items are stored, Stemberger & MacWhinney made use of a further perceptual phenomenon called the gang effect. Applied to word recognition, the gang effect says that the recognition of individual phonemes or characters is aided by their appearing together with other common neighbors. The larger the shared context, the stronger the effect. If all of the phonemes making up a given lexical form are stored as contiguous phonetic strings in the lexicon, they can profit from the gang effect during recognition and production. As a simple example, English pronunciation realizes the character sequence "ave" as /æːv/ in "have" and as /eiv/ in "cave", "rave", etc. Because the /eiv/ pattern occurs much more frequently at the ends of words, a non-lexical string like "mave", whose possible pronunciation is open, is most likely to be pronounced /meiv/ rather than /mæːv/. By analogy, if a great many past tense forms ending in unvoiced stops like /k/ have the past tense feature realized as /t/, as in *kicked*, and are stored as full-form items in the lexicon, the gang effect could help produce the past tense of an unknown stem like **bick* by reinforcing the most common surface representation in an environment ending in an unvoiced velar stop. Stemberger & MacWhinney see this process as not involving an operation of explicit concatenation but as a kind of connectionist transformation, turning an abstract set of features into a surface representation through a network of associative links. If effects requiring the presence of such gangs of items in the lexicon could be demonstrated for a set of lexical items, one could argue that these forms must in fact be stored.

To investigate whether the gang effect influences the production of inflected verbs, Stemberger & MacWhinney presented subjects with English past continuous forms like "was spanking"; the subjects then had to say quickly the corresponding simple past form of the verb, e.g., /spæŋkt/. From the visual stimulus a subject could derive no more than the stem of the required form. Thus on seeing "was spanking", a subject would need to isolate /spæŋk/ and then try to realize this as the past tense. Following the gang filter hypothesis, on its way to a surface pronunciation, /spæŋk/ + PAST might pass through gangs of similar phonetic forms, some of which might also contain the past tense marking, as with *drank*, *shrank*, etc. These irregular, phonetically similar past tense verbs would thus erroneously let *spank* pass through without receiving any additional marking. If these similar forms were not present, the error would be less likely. For stems resembling many irregular past tense forms, this is in fact what was

found: instead of producing past tenses like *spanked*, subjects sometimes said *spank*, and the rate for this type of error correlated well with the amount of phonetic overlap between the present tense stem and the gang of similar past tense forms.

To test whether the same effect could be found for regular verbs, Stemberger & MacWhinney made use of English verb stems like *gaze* ending in the sounds [z] or [s], which can be confused with the third person singular present inflections, which include [z] and [s]. They required subjects to produce the third person singular present for a visually presented string like "is gazing". On the hypothesis that regular inflected forms for the third person singular present, like *plays*, *stays*, *pays* are represented as full-form entries in the lexicon, production of a third person singular present verb *gazes* from "is gazing" should lead to errors similar to those found for irregular past tense forms. For stems having no such gang of phonetically similar third person singular present forms, the predicted error rate is lower. However, the experimental study found no significant difference in the error rates, implying that the assumed interfering forms like *plays* are not stored.

Together, these results seem to show that relatively few regular inflected forms are stored in the lexicon, while most irregular forms probably are stored. The logic of the inference, to be sure, rests on an number of rather speculative assumptions, and it is not hard to imagine alternative explanations based on other modes of storage. For example, it is conceivable that variant forms of irregular verbs are not stored but are inflected by first identifying a shared lexical stem and then changing the stem's vowel. Because the stem found in the stimulus *was spanking* has one of the vowels used in irregular inflection (/æ/) and is presented in a past tense form (PAST + CONTINUOUS), hurried processing might skip the step that returns the stem of the present participle (*-ing*) form to its default form, which is /spæŋk/ + PRESENT. Then the incorrect combination of *spank* + PAST would immediately match /dræŋk/ + PAST, from the pattern *drink*, *drank*, and by virtue of the gang effect, *spank* could slip through as the required past tense form.

However, a result similar to that of Stemberger & McWhinney was obtained in a more direct way by H. Clahsen by eliciting German past tense verbs and participles from nonsense stems [Cla97]. German has regular and irregular forms for both; in fact in the first 110 frequency ranks the strong verbs dominate the type frequency spectrum. Thus, German lends itself excellently for error comparisons like those of Stemberger & MacWhinney. Moreover, it is well known that German speakers tend to use the regular inflections for derivations of verbs from nouns, even when a simple conversion of the noun to a verb exists and is irregular. For example, related to the noun *Pfeife* 'pipe' one finds a verb *pfeifen* 'to pipe', which is irregular on the pattern *pfiff* (past), *hat gepfiffen* (perfect participle). But if a transitive verb 'to decorate or outfit with pipes' is formed with the derivational prefix *be-*, speakers reject the irregular model of *pfeifen*, preferring instead to use the regular affixes to form *bepfeifen* (present), *bepfeifte* (past), *hat bepfeift* (perfect) [Cla97].

In the experiment described by Clahsen, adult native speakers of German were asked to fill in the blanks in short narratives of the form

> Eines Tages kam mein Freund Peter zu mir und fragte mich, ob ich seinen Zatt *teiden* kann. Es war kein Problem für mich, und ich *tied* seinen Zatt. Es war nicht das erste Mal, daß ich einen Zatt _____. Peter sagte: Danke daß, du meinen Zatt

_____ hast. [3] [Cla97, 79].

In examples like this with an irregular (strong) verb, Clahsen's subjects filled in only 58 percent of the past tense forms correctly, otherwise preferring to supply a regular past form (with the suffix *-te*), and of those who supplied correctly the strong past form, 71 percent nevertheless filled in a regularly formed participle.

For adults, this effect might be explained from the much higher proportion of regular verbs in their vocabularies (although even in adult German the irregular verbs dominate in token frequency). To eliminate this possibility, Clahsen tried to find a similar effect with children between the ages of 1.5 and 2.1 years. Clahsen could demonstrate that the verb vocabularies of these children had not yet reached 110 types, the rank at which the regular verbs start to dominate. Yet even in this age bracket, he found that children tend mainly to overgeneralize in the direction of regular past and perfect forms, despite their knowing more irregular than regular verbs. Thus, it appears that they are learning combinatorial rules for the regular formations but are learning the irregular verbs by rote.

3.5.4 The Nominative Case Effect

A study by G. Lukatela et al. [LKFT83] found evidence that Serbo-Croatian nouns are processed by reference to their nominative forms. In Serbo-Croatian, masculine nominative singular nouns carry no inflectional suffix, while feminine nouns do; other cases (genitive, instrumental) are also marked with suffixes. Recognition latency for the nominative singular was always faster than for overtly inflected forms, even when other forms in the paradigm had higher frequencies than the nominative singular, and it occurred for feminine nouns as well, indicating that the effect was not caused by the absence of an affix. A somewhat similar effect was found by H. Günther [Gün88]. For German words that can be recognized as either an inflected verb or as an inflectionless nominative singular noun (like *Haut*), he found that the noun is identified even when its frequency is much lower than that of the verb. Günther interpreted his results as showing that during single word recognition the nominative singular is a default case-and-number assignment attached to the citation form of the noun; any other marking of the perceived word requires additional processing to discard the default assignment and then add the syntactically required case and number marking. In agreement with this hypothesis, he found that in syntactic contexts the default assignment to nominative singular is not made. Evidently the non-default morphological marking can be integrated at once during recognition in a syntactic context, and it is often preset by other elements in the syntactic context. Interestingly, Günter's results agree with and may explain Drews' finding that an inflectional affix requires a syntactic context to prime the recognition of a second inflected verb (section 3.5.2).

3.5.5 Conclusions Concerning Inflection

Summing up these results, it appears that despite the many riddles experimental investigations of inflectional processing have posed, few results are entirely compatible

[3] One day my friend Peter came to me and asked if I might *tied* his Zatt. It was not a problem for me, and I *ted* his Zatt. It was not the first time that I _____ a Zatt. Peter said, I'm glad that you have _____ my Zatt.

with the Full Listing Hypothesis. Irregular and very frequent forms are most likely to be accessible without morphological decomposition, while less frequent forms are evidently analyzed into constituent morphemes at some level. This does not, however, guarantee that high frequency, full-form entries are stored as atomic units, for it would still be possible that their lexical entries somehow internally mark the boundary found by linguistic analysis between stem and affix.

3.6 Evidence for Separate Treatment of Derivation

Many of the early experimental studies of word processing paid little attention to the linguistic properties of their test data, sometimes mixing compounded, inflected and derived forms in the same investigation. As will become increasingly clear, however, the linguistic distinctions among the categories of word formation are probably grounded in distinct modes of processing and cognitive representation. By comparison with inflectional affixes, derivational affixes are considerably more complex in meaning and behavior. Rather than specifying very general systematic attributes like number and tense, they can add specific semantic attributes or even change the meanings and syntactic categories of their bases. Furthermore, within the word they are distributionally different from inflectional affixes, generally appearing at positions closer to the base. It makes sense, therefore, to examine the experimental data gathered with derivational affixes separately from the inflectional data.

3.6.1 Lexical Decision Effects from Derivation

The early, influential lexical decision experiments of Taft & Forster [TF76] and Taft [Taf79b] also investigated target words with derivational prefixes and pseudo-prefixes, such as *reconsider* and *reproach*. A general result of these experiments was that, as with inflectional suffixes, the lexical decision time for prefixed derivatives correlated with the total frequency of all words containing the stem or pseudo-stem rather than with the frequency of the prefixed word alone. Thus, verbs of quite low surface frequency like *reproach* were nevertheless recognized rapidly, evidently because of the high frequency of *approach*, but when the combined stem frequency was low, as it is for the total frequency of *persuade* and *dissuade*, the recognition time for both words was correspondingly longer. As with the results for inflected words, this indicated that the (pseudo-) prefix was being ignored, and that the stem must be consuming most of the processing time. While this is again an important result, what counts as a prefix in Taft & Forster's model is somewhat arbitrary and largely a matter of orthographic similarity; it is not based on a synchronic notion of linguistic derivation. For sets of words like *reconsider*, *reconceive*, and *recount*, it is evident that *re-* is being combined with and is modifying the meaning of its base in an understandable way. But in cases like *reproach* it was the existence of *approach* with the shared non-word **proach* that was held responsible for the prefix effect, although there is no further connection between the two verbs. Despite the existence of words like *partake*, words like *party* did not exhibit the prefix effect for the simple and perhaps circular reason that *par* does not function as a prefix in their model.

Taft & Forster speculated that stem-sharing could contribute to a kind of lexical

economy, making it possible to store and access sets of related words together through the shared substring. It is nevertheless paradoxical that the stem effect should also appear for pairs like *reproach – approach*, where it is difficult to find any shared elements of meaning. In fact, on a strict reading of Taft & Forster's early model [TF76], if a word **inty* were to exist, the non-sensical stem **ty* shared with *party* would promote *party* to a prefixed word, with the access prefix *par*.[4] Indeed, a later study by S. Andrews [And86] cast doubt on the meaning of Taft & Forster's results, suggesting that they were in part an artifact of the experiment design. In her experiments, Andrews' subjects had to distinguish suffixed and compounded words from non-words. She found that the decision times for suffixed derivatives were unrelated to the frequencies of the stems, except when derivatives and compounds were mixed in the experiment, from which she inferred that the subjects adopt a strategy of decomposing the derivatives after they have done the same for a number of compounds. Without this influence, access to the suffixed derivatives apparently takes place without decomposition. She also found that in compounds the second syllable contributed to recognition just as strongly as the first, calling into question Taft & Forster's definition of the access prefix or BOSS (cf. section 3.3). Whether Taft & Forster's prefix effects could be explained by a similar strategy is not explicitly addressed in her study, however. On the whole, the data from these lexical decision experiments appear unable to decide the question of whether derivational prefixes are processed separately from their bases.

3.6.2 Uniqueness Points with Derivation

In auditory recognition, where many investigators have adopted the cohort model of Marslen-Wilson [MW92a], prefix-stripping effects have been much harder to demonstrate. The cohort model is most strongly supported by gating tasks (section 3.2.1), which have been replicated successfully in many variants. The original cohort model implies that morphologically complex spoken words must be stored in accord with the Full Listing Hypothesis, at least for prefixes, for if words were being accessed via their stems, identification of many prefixed words would only be possible at the point where the stem was uniquely identifiable. This is evident in a pair like *lead – mislead*. The uniqueness point for *lead* is not reached until the final /d/ because of *leave*, *leaf*, etc. But *mislead* is usually identified at the final vowel, which would not be possible if the prefix were not being used directly, as part of the access string.

More recent work has required an elaboration of this simple version of the cohort theory. H. Schriefers et al. found both gating and phoneme monitoring effects (section 3.2.1) that appear to require separate processing of derivational prefixes in auditory recognition [SZR91], though not the kind prefix-stripping with initial access via the stem that Taft & Forster proposed in [TF76]. It has been found in gating tasks that once the uniqueness point has been reached, phoneme monitoring times are faster than before the uniqueness point. Phoneme monitoring thus suggests itself as a way of identifying the uniqueness point without presenting several increasingly long repetitions of the same phonemic string, which Schriefers et al. suspect of inducing left-to-right processing in cases where the underlying mechanism may be proceeding differently.

Schriefers et al. carried out gating and phoneme monitoring experiments on triplets

[4] A similar critique is presented by [San94, 73-76].

of Dutch verbs chosen so that an unprefixed verb and a prefixed derivation had the same uniqueness point (UP). The third verb was a derivation from the same stem but with a UP ahead of the stem's UP, e.g.,

- *staaN* 'to stand'; UP = N
- *opstaaN* 'to get up'; UP = N
- *toestAan* 'to allow'; UP = A

On the cohort theory, the verb with the early UP should be identified earlier. However, what they found was that in a conventional gating task the prefixes caused *both* derivations, regardless of UP, to be identified ahead of the stem's UP. When the UP was sought by phoneme monitoring rather than by gating, the results were the same.

This experiment was repeated with quadruples having two stems with differing UPs, one earlier than the other, and two derived words using the same prefix. The two derived words were chosen so that their UPs were at about the same distance from the word onset, and so that they were matched in frequency. On a continuous processing, left-to-right account, the prefixed verbs should be identified at the UPs, which in both cases were at the same point. Again, the identification points were sought by gating and by phoneme monitoring, and again the prefixed words had earlier identification points. However, for both prefixed verbs the identification points were not at the UPs, but were moved forward by a nearly constant amount relative to the UPs in the stems. Again, it appeared that the prefix was aiding recognition of the stem, but that the stem's UP still mattered. For pseudo-prefixed, monomorphemic words, however, this effect did not appear; the UPs were recognized at the points predicted by the cohort model.

These results, while hard to account for in detail, indicate that morphological segments rather than BOSS-like units (which include pseudo-prefixes) are involved in auditory access. The prefixes are apparently being used in some way that facilitates recognition rather than being ignored prior to stem access. Using an auditory lexical decision experiment, Taft himself came to a similar view [Taf88]. As in gating tasks, auditory lexical decisions can be made for prefixed derivatives like *mislead* at a point before the stem alone (*lead*) can be identified. The cohort model explains this effect on the basis of the earlier UP. To test whether this effect was really a function of the UP, Taft found pairs of spoken words in which the UP of the base and of the prefixed derivation were at the same point. For example, both *tension* and *intention* have their UPs at the phoneme /ʃ/ because of competitors like *tent* and *intensive*. However, he found that a lexical decision could be made sooner for the prefixed word than for the base. Thus, the prefix appears to prime the following access to the base rather than just being ignored until the stem can be identified. How it "primes" access to the base without having any overt similarity is, of course, a bit mysterious. Schriefers et al. propose adding within the full-form representation used for strictly left-to-right access a representation of the discrete constituent morphemes. Equivalently, one might imagine that the phonemic string is stored as one whole, but that the end (or the uniqueness point) of each morpheme segment carries a pointer to the corresponding morphological item; the end of the word would, additionally, carry a pointer to the semantics and syntax of

the whole-word entry.[5] Once the UP of the prefix was reached, its activation would spread directly to the stem, or it might start to activate the stem though an associative network.

A remaining source of uncertainty regarding derivational prefixes lies in the effects found with pseudo-prefixes and semantically opaque derivations. The prefix-stripping model of Taft & Forster includes pseudo-prefixes that have at most a lost historical meaning, as in *reproach* (section 3.6.1), but these can have no independent morphological function for most speakers. Sandra's survey of the experimental literature [San94] found a consensus that derivational prefix-stripping can occur, but it could find no clear experimental consensus about whether or not pseudo-prefixes belong to the set of units that partake in the effect. Thus it remains unclear at this point whether or not derivational prefixes in the linguistic sense are actually separated from their bases during word access, or whether they simply coincide phonologically with pseudo-prefixes which, like Huffman code prefixes, are being used for string matching at a much lower level of recognition that has no necessary connection to the word's morphological structure. That is, it is possible that for the purposes of recognition, all derivational prefixes are nothing more than pseudo-prefixes. This does not imply that the same is true at deeper levels of recognition, however.

A further source of uncertainty in the data has to do with the languages investigated. It may be the case that how derived words are processed is not a cognitive universal but varies widely from language to language. C. Burani et al. found evidence that the degree to which a derivational affix can be confused with an orthographically identical but non-meaningful pseudo-affix affects how the true affix will be processed. In English there are evidently too many pseudo-affixes for reliable segmentation of many true affixes, whereas Italian, which has yielded many of the findings favoring morphological analysis of derivatives, has relatively few pseudo-affixed words. Burani et al. found evidence that segmentation of affixed words is not mandatory, but it is most likely for affixes having high "salience", represented by the proportion of real to pseudo-affixation relative to a sum of the token and type frequencies of surface forms containing the affix [BDTL97]. In English, for example, the derivational prefix string *de-* has a low salience because a high proportion of the words starting with the string "de" (like *deal*) are cases of pseudo-prefixation. Since Italian has far less pseudo-affixation than English, Italian yields more pronounced morphological effects in experiments. This contrast suggests that the two languages may in fact tend to represent derivatives differently. Similar discrepancies between English and Dutch are described in [San94, 83].

3.6.3 Cross-Modal Tests of Derivation

One further, important study, not covered in Sandra's survey, must be mentioned here because of its implications for the architecture of deeper levels of the lexicon. W. Marslen-Wilson et al. [MWTWO94] used a complex series of cross-modal priming experiments to isolate a deeper level of morphological representation than intra-modal lexical decision and repetition priming tasks can reliably test. Using spoken words (which I shall show in italics) as primes and visually presented words as targets (which

[5] An elaboration of this structure will be proposed for the orthographic representation of lexically stored, morphologically segmentable words for the KLU model in section 5.8.10. Cf. also section 7.5.3

I shall show in capital letters), they investigated priming effects between the stems of morphologically transparent pairs.

Specifically, they found that an auditory prime like *tinsel* did not facilitate visual recognition of a word that shared an orthographically equivalent segment, like TIN. The same result was obtained for pseudo-prefixed words. When test subjects heard a pseudo-prefixed word like *approach*, it did not prime visual recognition of a pseudo-prefixed word like REPROACH, nor did pairs containing a pseudo-prefixed word and its stem (*release* – LEASE) show any priming effect. This stands in open contradiction to the intra-modal studies of Taft and Forster [TF75] and Taft [Taf79b], whose results showed clear priming effects with such pairs. It strongly suggests that the effects Taft and Forster reported take place at an early stage of recognition, and that the semantic or conceptual level of representation is not involved. It thus argues against the simple version of the FLH, in which lexical entries are single units, like the records of a data base, in which an access to the phonological form, as a kind of key within the record, necessarily activates the whole record. The cross-modal results speak rather for a model like that shown in Fig. 3.2 (p. 70), in which the phonological and orthographic representations are held in data structures separate from the semantic or conceptual level.

On the other hand, morphologically transparent derivatives do cause cross-modal priming in pairs like *unwind* – REWIND, *govern* – GOVERNOR, *misjudge* – JUDGE. In these pairs it appears that a priming effect arises either from the shared morpheme (*wind*, *govern*, *judge*), or from conceptual associations. As many experiments have verified, conceptually related pairs like *idea* – *notion* also exhibit priming [Har95, 70]. Marslen-Wilson et al. observe, however, that the priming effect from pairs that are only related at the conceptual level dissipates more rapidly than with pairs that also share a derivational base. Further, they found that suffixed pairs like *governor* – GOVERNMENT show no priming effect, in contrast to prefixed pairs like *unwind* – REWIND. If the priming effect were due solely to a conceptual association between prime and target, the position of the derivational affix would not affect priming. Exactly why suffixed derived pairs should fail to exhibit priming is not self-evident, but any explanation of the difference between suffixed and prefixed pairs requires direct reference to the words' morphological structures, not merely to their conceptual relatedness.

Table 3.2: Cross-Modal Priming in Derived Words and Stems. From [MWTWO94].

Auditory prime	Visual target	Facilitation effect	Dissipation speed
tinsel	TIN	none	
unwind	REWIND	60ms	slow
punishment	PUNISH	40ms	slow
govern	GOVERNOR	30ms	slow
governor	GOVERNMENT	none	
distrust	TRUSTFUL	30ms	
idea	NOTION	27ms	fast

The results of these experiments, summarized in Table 3.2, can be interpreted as showing that transparently derived words (with morphologically free bases) retain their morphological structure at some central, modality-independent level of the lexicon. What is perhaps most significant in these results is that they lend empirical support to a model that has been advocated by some, but not all semanticists, in which lexical semantics is built up from primitives roughly corresponding to the morphemic units found in words. These "two-level" models (cf. e. g., [Bie83], [Lan87]), distinguish a conceptual level of meaning, at which words like *idea* and *notion* are related, from a semantic level, at which only schematic, underspecified representations of word meaning are stored. At this level, the difference in surface morphological structure between pairs like *govern* and *governor* is reflected immediately in a parallel difference in the formula which represents the word's lexical semantics. By comparison with the ambiguous intra-modal recognition data, the results of Marslen-Wilson et al. suggest that derivational morphemes have empirical, psychological reality as units of meaning. Furthermore, the morpheme effects revealed in cross-modal priming must be carefully distinguished from the phonological or orthographic morpheme segments involved in low-level recognition. It is evidently possible that a morpheme-like segment, such as a pseudo-prefixed *re-* can be separately identified during recognition, even when it corresponds to no constituent of the word's semantics, as in *reproach*. The ambiguous results from intramodal priming suggest, on the other hand, that genuine prefixes may fail to be identified separately during recognition, even when the cross-modal priming data show that they are represented as distinct units at the lexical-semantic level. In morphology, therefore, the mapping from surface constituency to semantic constituency cannot be one-to-one. Evidently separate data structures will be required for both levels, coupled loosely enough to allow many-to-one mappings in both directions.

3.7 Multiple Access Paths

Much of the early psycholinguistic work on lexical representation was directed to the question of whether words are represented in memory as full-form entries or as decomposed sets of morpheme-like units. As is now evident, the answers obtained have been ambiguous, and consequently a number of more recent studies have tried to formulate the issue differently, asking if both modes of storage might co-exist in some way.

3.7.1 Paralexias and Word Substitutions

In a study of patients with reading impairments, A. Caramazza et al. [CMSL85] noticed selective deficiencies in reading content words, function words (like subordinating conjunctions), and various kinds of non-words containing recognizable morphological segments. In some of the more severe cases, patients could read words and non-words alike only by "sounding out" the orthographic pattern. Similar cases were also reported by Fromkin [Fro87]; her patients in fact had equal difficulty reading words printed normally and backwards. Such disruptions suggest that there is an area of the brain corresponding to the set of orthographic logogens in Fig. 3.2; when the orthographic representations are disabled, patients evidently must first translate the orthographic representation to a phonetic representation in order to access the lexicon

via the phonetic logogen system.

Other patients show a more specific difficulty in reading morphologically complex words and function words. With function words, like pronouns and subordinating conjunctions, they often substitute a different word for the one shown, e.g., saying "which" when "that" is shown. These function word substitutions are frequently associated with a further difficulty, known as morphological paralexia, characterized by the incorrect rendering of inflectional endings, e.g. saying "lover" or "loved" in response to a displayed "love". In these cases, it appears that the ability to recognize full words and word stems is intact, but the connections from the orthographic representations of function words and inflectional morphemes to the central lexicon are selectively impaired.

An Italian patient described in [CMSL85], LB, showed a particularly marked deficit of this type in his selective ability to read non-words. Being unable to read non-words having no recognizable substring, LB was evidently unable to carry out character-to-phoneme translations. However, his impairment was largely restricted to non-words that lacked a recognizable lexical stem. He could pronounce displayed non-words on the pattern of **chiedova*, containing the lexical stem 'to ask for' but a non-existent affix, correctly 80 percent of the time. The opposite kind of non-word, like **chiadiva*, containing the common imperfect, third person singular inflection *-iva* but no lexical stem, was more difficult and was pronounced correctly in only 45 percent of the trials, suggesting that the lexical stem plays a key role in obtaining access to phonological representations through the central lexicon.

Caramazza et al. interpreted these deficiencies as evidence for three different means of accessing the central lexicon, all of which are available simultaneously except when impaired by specific brain injuries. They hypothesized that, as represented in Fig. 3.3, the full-form representations of many words are available for accessing the central lexicon; it is via this route that patients like LB can read familiar words. However, LB's ability to read many non-words as well when they contained recognizable morphological constituents indicates that the constituent morphemes must also be present as entries in the orthographic table, since he was not able to use pure character-to-phoneme mapping to read orthographically well-formed non-words having no lexical sub-units. Why the presence of a lexical stem was of greater help in reading a non-word than a lexical affix is not clear, however; Caramazza et al. conclude that the "lexical status" of stems and affixes must be different in some way. The selective disruptions with function words and inflections presented by patients with the function word impairment and morphological paralexia suggest, in any case, that affixes and function words have representations localized to an area that can be affected independently from the area where content words are represented.

3.7.2 Multiple Effects of Partial Non-Words

In a later study, Caramazza and other collaborators [CLR88] showed experimentally that during word recognition inflectional affixes appear to be processed separately from the stem. They presented both words and non-words in a lexical decision task, constructing the non-words in such a way that they sometimes contained known morphemic segments. Thus the Italian verb *cantare* 'to sing' has the past tense form *cantevi*, but the partial non-word **cantovi*, having the same stem segment, is not a legal

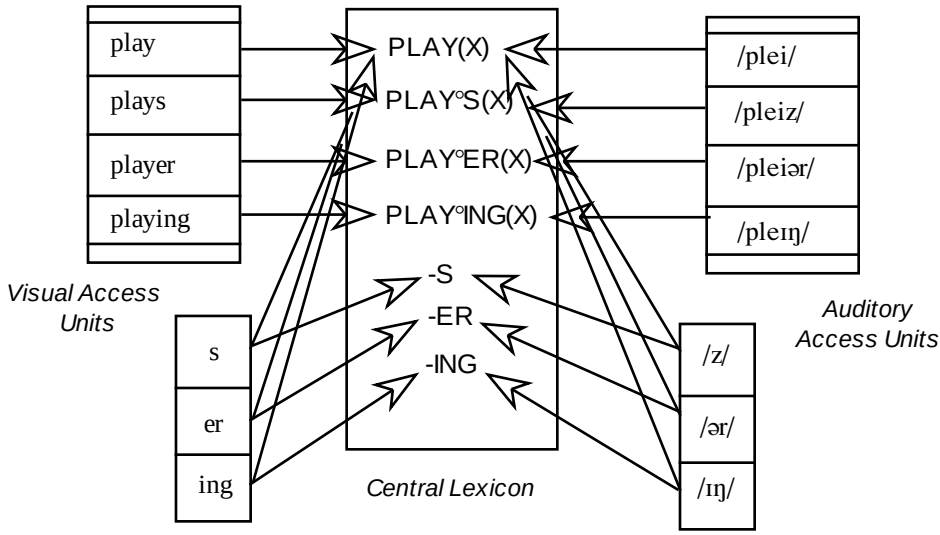


Figure 3.3: Access Units and Central Lexicon in the Augmented Addressing Model.

inflected form, and **canzovi* is a complete non-word having no recognizable morphological segments. On the Full Listing Hypothesis (no morphological segmentation), it should take as long to reject **cantovi* as to reject **canzovi*, since neither form will be present in the lexicon. Alternatively, one might suppose that when the whole word is not found, processing continues, trying to find and combine known word segments. In this case, it would take longer to reject **cantovi* because after finding the known segment *cant-*, the processor would need additional time to check whether the remaining non-affix segment **-ovi* could be combined with it. A similar effect would be expected from **canzevi*, in which a known inflectional segment, *-evi*, would be found, requiring a further test before the word could be rejected, to see that the remaining segment is not a possible stem. The experimental results showed that the non-word decision times were shortest for stimuli with no recognizable morphological segments, like **canzovi*, longer for non-words with recognizable stems, like **cantovi*, and yet longer for non-words with recognizable inflectional segments like **canzevi*. The longest decision times were for well-formed inflected words like *cantevi*. In other words, the lexical decision time is evidently proportional to the number of identifiable morphological segments, suggesting that segments can be found quickly, but that time is required to access their lexical entries and to check their compatibility.

The search for lexical segments alone requires little time, and it appears not to entail full lexical search, since the rejection of non-segmentable, complete non-words is faster than either acceptance or rejection of words having identifiable segments. On the other hand, we have seen that an existing, monomorphemic word of low frequency has a long decision latency. Thus, non-segmentable non-words can be *rejected* quickly, but the time to decide that a word exists can be long. This suggests that monomorphemic words, too, are represented at some low level as segments. If the segment is not present, the word can be rejected quickly. Its presence does not suffice to identify the word, however. Some further, time-consuming access procedure must run to completion before the word can be identified with confidence. Again, these results confirm a structure in which morphological segments as units of recognition are stored

separately from the morphological units of meaning.

A similar effect in a visual lexical decision task using derivational rather than inflectional suffixes on English words and non-words was obtained by L. Manelis and D. L. Tharp [MT77]. Compared to derived non-words like **losker*, they found that subjects required a longer time to reject non-words based on an existing word, like **desker*. Although Manelis & Tharp concluded that morphological structure played no role in recognition, their non-word data were re-evaluated by Taft [Taf79b] as showing that the suffix triggers decomposition, and that a recognized base triggers a lexical search that consumes additional processing time.

Another aspect of complex word formation for which Italian provides possible diagnostics is the organization of nouns and verbs into inflectional paradigms, by which the grammar restricts the range of possible stem-affix combinations. Moreover, in Italian, some verbs have suppletive forms in certain positions in the conjugation paradigm. In these positions, regular stem and affix combinations are prohibited and are replaced by forms that must be specially learned; e.g., *correre* 'to run' is largely regular but uses as its participle *corso* instead of the fully regular form **corruto*. If lexical access is preceded by segmentation into morphemic units, we should expect subjects to need more time to reject **corruto* than to reject a more easily identifiable non-word like **canzevi*, because other stem-affix combinations on the pattern *corr-+-uto*, e.g., *venduto* 'sold', are generally allowed. This is in fact what Caramazza et al. found. This suggests that the segments *corr-* and *-uto* are identified and found compatible in the same way as for *cantevi*, but that the search for the suppletive form that finally prohibits the combination consumes additional time. The suppletive form must be stored as a separate entry that is accessed at the final stage of the lexical search.

An additional test of this account can be made by comparing the decision time for partial non-words like **corruto* to partial non-words like **copruto*. Here again segmentation into two morphemic segments is possible, namely *copr-*, the stem of *coprire* 'to cover' and *-uto*. In this case, however, the combination of the two morphemes is prohibited by virtue of the membership in different inflectional paradigms: *copr-* belongs to the third conjugation (*-ire*), while *-uto* is found only in the second conjugation (*-ere*). Thus, if lexical decision involves first identifying the morphemic segments and then checking their paradigmatic compatibility, it should be possible to reject **copruto* more quickly than **corruto*. Again, lexical decision times confirmed the prediction. In addition, the error rate (misclassification as well-formed) for non-words that could only be rejected on the basis of a competing suppletive form (**corruto*) was 20 percent higher. Combining a suppletive stem (e.g., *cors-*) with the fully regular participle affix (here *-uto*), however, leads to a decision time and error rate like that for **copruto*, even though stem and affix belong to the same conjugation. Caramazza et al. [CLR88] reason that since the suppletive stem *cors-* is not a "full" member of the paradigm, the combination *cors- + -uto* can be rejected immediately, without a further check for the presence of a suppletive equivalent.

3.7.3 Multiple Cross-Priming Effects

Addressing objections that non-word stimuli do not necessarily indicate how well-formed complex words are processed, Badecker and Caramazza [BC89] performed a

follow-up series of experiments to substantiate their claims made in [CLR88]. These used only well-formed Italian adjectives, verbs, and nouns, and were based on a somewhat unconventional cross-priming procedure in which two words were shown simultaneously, requiring a decision as to whether both were well-formed or whether at least one was a non-word. In one of these experiments, four groups of word pairs were presented. In the first group, pairs were chosen such that they shared a common stem belonging to two unrelated lexical items, e.g., *port-* is the stem of both *porta* 'door' and of *portare* 'to carry'. This group was meant to test the influence of the homographic stem's orthographic or morphological cross-priming effect. The second group consisted of pairs that shared visually similar, but otherwise unrelated stems, e.g., *contare* 'to count' and *corta* 'short'. This group provided a measure of the influence of the orthographic surface representation alone on the cross-priming effect, since no priming at the morphemic level was possible. In the third group the pair shared a single, unambiguous stem, as in *posto, posti* 'place, places'. The fourth group contained unrelated pairs like *causa* 'cause' and *ponte* 'bridge'. Here no cross-priming effect of any form was to be expected, so that these pairs could serve as a baseline for the other pairs.

- Group 1: *porta* 'door' — *portare* 'to carry' (957 ms)
- Group 2: *contare* 'to count' — *corta* 'short' (915 ms)
- Group 3: *posto* 'place' — *posti* 'places' (883 ms)
- Group 4: *causa* 'cause' — *ponte* 'bridge' (919 ms)

The decision latencies for the unrelated pairs (Group 4) were 919 milliseconds on average. The latencies for the visually similar but otherwise unrelated pairs (Group 2) were virtually the same, indicating clearly that purely orthographic similarity could be excluded as a source of any cross-priming effects. The single-stem pairs (Group 3) showed facilitation of 36 ms relative to the baseline (Group 4), while the ambiguous, homographic stem pairs (Group 1) showed an inhibition effect of 38 ms relative to the baseline.

The facilitation observed in Group 3 can be seen as a priming effect that could arise either at the semantic level (by virtue of the shared meaning) or at a lower level, perhaps near (but not at, given the results for Group 2) the orthographic access form. However, the inhibitory effect found in Group 1 can only be explained by reference to the presence of two unrelated meanings for each stem in the morphemic lexicon, which is the factor distinguishing the Group 1 pairs from those of Group 3. The presence of two unrelated meanings at the whole-word level is shown by Group 4 to have no cross-priming effect. The fact that priming inhibition can turn into facilitation as a function of the ambiguity of the stem indicates that the stems' meaning representation or representations must be involved in the recognition of well-formed words used as cross-primes. Going beyond the lexical decision tasks used in [CLR88], the cross-priming experiment thus appears to show not only that decomposition procedures are *available* for word recognition, but also that they are *regularly involved* in recognition. It is perhaps worth noting, however, that no cross-priming effects are demonstrated for the inflectional affixes; whether this procedure could identify a priming effect like that found by Drews (section 3.5.2) remains unanswered. Whether similar effects could

be shown for derivatives sharing homographic stems is also an open question. Unlike the lexical decision task with partial non-words, this experiment does not clearly demonstrate a parsing effect; it merely shows that the stem and affix have separate representations at some level where priming can take place.

3.8 Evidence of Level-Ordered Storage

The experiments described so far have been concerned with the question of whether words are represented as wholes or in terms of their morphological constituents; they do not give much information about the internal structure of the representations. Caramazza's model, sketched in Fig. 3.3, assumes that the central lexicon retains some sort of morphological segmentation, which is required in order to explain the stem priming effect for words that are accessed without being decomposed and parsed beforehand; such a representation would also be necessary to account for the inflectional priming found by Drews.

There is some tentative experimental evidence, however, suggesting a further differentiation of the lexical representations, either at the access level or in the central lexicon. In a priming experiment using English derivational suffixes, D. Bradley found a clear stem-frequency effect for suffixes that are statistically productive, like *-ness*, *-ment*, and *-er*, but the effect was less evident for relatively unproductive suffixes like *-ion* [Bra80]. Stem frequencies cannot, of course, be found in tabulations of surface word frequencies like that of Kučera & Francis [KF67]. To obtain an estimate of the stem frequency, Bradley summed the frequencies of all forms in which a stem appears with a productive affix. Thus, to obtain the "cluster stem frequency" for *sharp*, she used the frequencies of *sharpness*, *sharper*, and *sharply*. She then constructed pairs of derived words like *sharpness – briskness* having nearly equal surface frequencies but significantly different cluster frequencies for the bases. For words with productive derivational suffixes like *-ness*, the lexical decision times were strongly related to the cluster frequency of the base, but for less productive affixes like *-tion* (as in *destruction*), the effect was much less pronounced.

Bradley's classification of affixes was actually based on the distinction of stress-neutral and stress-influencing affixation in lexical phonology, later summarized in [Kip82a]; if taken as psychological evidence for this phonological classification, the results of her experiments lend weight to the thesis, developed in lexical phonology and morphology, that word structure is hierarchical rather than merely segmental. Why the level-ordering classification of morphemes probably coincides with $\mathcal{P}$-type productivity will be taken up in chapter 6.

An entirely different experimental paradigm, using production rather than recognition data and based on the elicitation of spontaneously formed, new derivatives, gives additional weight to the idea that there are levels of binding strength in word structure corresponding to linguistically motivated levels of morphological structure as well as to $\mathcal{P}$-type productivity. The results do not, however, directly bear on the issue of storage versus on-line analysis, and although they have played a large role in shaping the KLU model, they have not received much attention in current psycholinguistic models. For this reason I shall postpone discussing them until I take up the problem of word segmentation algorithms in chapter 6.

3.9 Psychological Models of Word Representation

As has been pointed out previously, it is difficult to conduct empirical investigations of the mechanisms involved in word processing without some conception of what these underlying mechanisms might be, yet many of the psycholinguistic studies have avoided constructing detailed models, especially models having the degree of complexity found in current linguistic theories of morphology and grammar. For many the still existing mass of conflicting results and unresolved questions has undoubtedly made a grand synthesis appear premature, but at least three noteworthy proposals for comprehensive models have been presented recently. At least in their published forms, these do not yet have enough detail to count as specifications for a computational model, but they present many ideas similar to those that have informed the KLU model.

3.9.1 The Augmented Addressing Model

Caramazza and his collaborators have tried give an explanatory framework for their results in terms of a model called Augmented Addressed Morphology (AAM), sketched in Fig. 3.3. Like the Logogen Model of Morton (Fig. 3.2) (p. 70), the AAM model postulates separate sets of access representations for written and spoken language that can integrate the input stimuli and 'fire' when their frequency-dependent thresholds are crossed. The separate access lists, or files, for auditory and visual input are necessary to explain the fact that certain brain injuries can impair one kind of access to the lexical information while leaving the other mode intact. The access files contain no lexical information like lexical and morphological category, argument structure or meaning; these are all contained in a central lexicon indexed by the access units. Unlike the logogen model, the AAM model assumes that the access representations include both whole-word representations (which explain the surface frequency effect on lexical decisions on highly frequent, complex words, described in section 3.5.1) as well as representations of individual morphemes. When a whole-word representation is not available (as it may not be for a less frequently used inflectional form), morphological segments can be identified and combined via a parsing mechanism that tests the compatibility of the segments; this explains the longer decision times required for non-words with identifiable morphemic segments relative to words containing no morphemic segments. The combined morphemes then access entries in the central lexicon that represent the entire word in a morphologically segmented form. This form presumably accounts for stem priming effects like those reported by Badecker and Caramazza [BC89], since meanings are not stored with the orthographic access forms, and if the stem meanings were not activated in the central lexicon, there would be no explanation for the cross-priming inhibition observed in pairs with homographic stems.

In the version presented by Caramazza et al. in [CLR88], the AAM model describes in detail the processing of inflected Italian verbs. It specifies that the access units contain precisely those character strings that correspond to a morphemic unit, and as a consequence allomorphic variation and suppletion must be explicitly represented. Based on the effects that were found with partial non-words (like **copruto*), it also specifies a system of "combinatorial links" between the access units that allow the

word recognizer to reject a pair of morphemes before the whole word representation in the central lexical is accessed. These links, as we shall later see, appear to be very nearly the kinds of relations expressed as feature unification equations in unification-based formal grammars.

The model also makes allowance in a less explicit way for something like the "Elsewhere Principle," introduced by P. Kiparsky into lexical phonology (cf. [Kip82b]), by providing for both full-form and decomposed representations of words. The Elsewhere Principle says, in effect, that any lexically represented word or morphological unit has precedence over and can block further morphological analysis. Although the AAM model does not make specific predictions about precedence relations between the full-form and decomposed representations, it assumes that frequently used items will be accessed directly via the full-form access unit, with morphological analysis only being activated when a full-form representation is not available. This has the effect that any full-form access unit can block further morphological processing at the input level. The effect is nevertheless not identical to that proposed in lexical phonology because in the AAM model the representations of complex words in the central lexicon remain present as decomposed morphemes, as represented by the °-markers in Fig. 3.3 (p. 91). If visible to higher levels of word and sentence processing, these morphological boundary markers would violate the Bracketing Erasure Principle of lexical phonology, which makes stem-internal structure invisible to higher levels of processing. What happens with the morphemic units in the central lexical entry is not specified by Caramazza et al., except that the units activate their individual semantic representations, as is evidenced by the stem priming effect. This still leaves open the possibility that the meaning finally integrated into the sentence's meaning arises not by combining the morphemic meanings, but from an access to a separately stored representation of the word as a whole, as the Elsewhere Principle seems to require. However, it may also be the case that syntactic and semantic features from the separate morphemic representations are combined anew on each access to the complex item.

Kiparsky's Elsewhere Principle also has the effect of giving suppletive forms precedence over regular formations. In the account presented by Caramazza et al. for the rejection of non-words like **corruto* (section 3.7.2), the suppletive stem is present as an access unit and processing is so arranged that it has logical precedence over the regular form, but the regular form is rejected latest. Exactly how this might take place is not specified precisely, but it evidently depends on the existence of the suppletive stem *cors-*, in combination with links to its two possible suffixes, *-o* and *-i*. Nevertheless, one has to wonder whether all allomorphs are explicitly represented with separate access units, especially since, as we shall later see, some forms of simple allomorphic variation can be implemented in a computationally inexpensive way.

The AAM model does not contain a separate account of the processing of derived words, which as we shall see in the next chapter in fact pose considerably larger challenges than does inflection.

3.9.2 The Morphological Race Model

Trying to resolve a number of puzzles arising in the experimental literature, U. Frauenfelder and R. Schreuder presented in [FS92] a version of a processing model first sug-

gested by H. Baayen in [Baa92]. This Morphological Race Model is much more detailed in its specific proposals than the AAM model, and it is ultimately meant as a specification for a computational implementation able to model morphological activation in terms of spreading activation effects.

The model's point of departure is the problem posed by two possible modes of access in the AAM model. Whereas the AAM model implies that a full-form access unit for a word will block further morphological analysis, Baayen's race model assumed that a word can, in principle, always be recognized in both ways. The outcome of the access is thus not determinate; rather both modes of access are initiated and compete to deliver a usable result to postlexical processing in minimum time. The race can therefore explain why many experiments seem to show that some words, particularly those of high frequency, are stored whole, while other words are apparently parsed. The model is also of special interest for an account of how derived words are processed.

Baayen reasoned that the recognition race could be decided by measures of productivity: unproductive affixations should be recognized via the direct access route, while productive derivations were more likely to be parsed during on-line recognition. In the Morphological Race Model (MRM), recognition time for the full-listing form is proportional to (the inverse logarithm of) the word's frequency, as has been established repeatedly in lexical decision tasks. In case the word recognizer decomposes the word into morphemic units, recognition time is seen as depending on both transparency (phonological and semantic) properties of the derivative and on recognition latencies of the constituent morphemes, which are a function of the frequencies at which repeated accesses to the morpheme units have been successful, so as to receive reinforcement for later processing. More specifically, these latencies depend on the frequencies with which the morpheme units have been *successfully* accessed. Hearing *repertoire*, for example, does not reduce the latency of *re-* because the morphologically decomposed access via *re-* and **pertoire* does not run to completion. Because the semantics of *repertoire* are non-compositional, the only access that can be successful is one to a central lexical entry that already contains *re-* and **pertoire* in combination, although perhaps as separate morphemic units, as in the AAM model. However, if the parsed access succeeds before the direct access, as might happen with an infrequent and transparent derivation like *waterable*, credit goes to the constituent morphemes and not to the (possibly non-existent) full-form representation.

Thus the MRM scheme predicts that easily parsable derivations will always be parsed, since their constituent morphemes will be reinforced and made easier to recognize with each encounter; whereas the evidence from historical linguistics is that frequently used derivatives tend to become increasingly opaque and thus unparsable. There ought to be a shift from parsing to direct access in the course of lexicalization, but the MRM account does not allow this. Thus Frauenfelder & Schreuder must assume that the full form gradually "obtains its own representation" even though recognition via morphological decomposition is faster. "The idea here is simply that a full representation is created after the first parse or at least after some limited number of exposures" [FS92, 177], and somehow the resting activation of this new lexical form starts to receive reinforcement from subsequent exposure to the word rather than the constituent morphemes. Once this happens, the constituent morphemes can disappear altogether from the lexicon, unless other words continue to activate them. However, words that remain transparent will continue to be easily parsable, and may continue to

win the parsing race against the access to the directly accessed full form. That is, the parsing route can continue to win the race for words that are both transparent and of low frequency. With inflectional affixes this is likely to be the case more often than not, because they will be appear again and again with stems of low frequency, with the combined representation for each different complex form gaining little reinforcement, but with the affix by itself being strongly reinforced. To a lesser extent, the same effect is likely to arise with widely used derivational affixes.

Baayen's parameter of productivity $\mathcal{P}$ is thus thought to be reflected in the recognition latency of the affix that participates in a given inflectional or word-formation process: "The activation level of an affix thus represents the relative ease with which a form with this affix is parsed and is positively correlated with the productivity of the affix" [FS92, 178]. Close examination of the productivity data discussed in the last chapter will show, however, that this solution is too simple; recall for example that in Baayen's study of the London *Times*, the prefix *un-* turned out to be highly productive with adjectival bases, but to have low productivity with verbal bases. The productivity of the pattern is thus not simply a property of the affix. A further difficulty is that the MRM model fails to provide a detailed account of the parsing process, which must somehow arrive at morphological segment boundaries within the speech stream or orthographic representation before accessing the morphological constituents and checking their compatibility. Nevertheless, the MRM account fills in a great many details that have been passed over in the psycholinguistic mainstream.

3.9.3 The 'Meta-Model' of Schreuder & Baayen

Expanding on the Morphological Race Model, R. Schreuder and H. Baayen proposed a "meta-model" to give an account of additional data, including effects known from studies of morphological acquisition [SB95]. Like its predecessor, this model takes account of many of the same effects described by Caramazza and his collaborators (cf. section 3.7). In addition, it is meant to explain the stronger effects obtained in many studies for inflectional relative to derivational morphology, the stronger effects of semantically transparent complex words relative to opaque derivations reported by Marslen-Wilson et al. [MWTWO94] and the obvious ability of listeners to process unfamiliar regular complex words. Finally, it attempts to give an account of pseudo-affixation effects. Because it addresses many of the same issues as the KLU model that will be described in following chapters, the meta-model of Schreuder & Baayen will be mentioned only briefly here.

As sketched in Fig. 3.4, the meta-model reserves a separate structure for phonological and segmental analysis of the input stream, prior to the level at which access units are addressed. One assumption of the model is that in the continuous speech stream a purely phonological process must be invoked to identify possible word and syllable onsets before lexical units can be accessed; otherwise every phonemic segment would need to initiate a potential lexical access. For visual input the problem is not nearly as overwhelming, but even here word segmentation can be far more complex a task than is generally appreciated (as we shall see in chapter 6). Unlike the access units in the AAM model, the assumed access units in the meta-model are not required to match the input strings directly. Instead, in addition to identifying possible word and segment boundaries, this module also resolves local and easily computed postlexical

phonological variations, like those due to vowel harmony in Turkish, so that the access units are not required to contain all possible phonological variants. Exactly how much processing can reasonably take place here has evidently not been worked out; Schreuder & Baayen merely note that "the computational complexity of this mapping operation should not be underestimated" [SB95, 135].

The intermediate access representation leads to candidates for lexical access which are then presented to the lexicon, or more precisely to the set of access representations, which, like the access units of Fig. 3.3, are merely indices to concept nodes. In the case of homonyms or homographs, these indexes can realize one-to-many mappings; synonyms result in many-to-one mappings. Each concept node is attached to a representation of the item's syntactic and semantic attributes, which are accessed simultaneously with the concept (not via the concept, as Fig. 3.4 seems to indicate) and passed on to post-lexical processing. When novel complex words are encountered, having no combined access unit, the word constituents will be identified by the segmentation module, and two or more concept nodes with their attached syntactic and semantic attributes can be activated simultaneously. In such cases, a prohibition on passing any uncombined representations to post-lexical processing comes into play. The lexical procedures identified as licensing and composition must successfully create a new, unified lexical representation before a result can be delivered to syntax.

Schreuder & Baayen also sketch an account of recursive derivation in their model. Assuming that the Dutch word *onwerkbaar* 'unworkable' is heard for the first time, they propose that it will be analyzed into a string of morphemes with lexical category markings as in

(16) $[_{A/A}$*on* $[_{A}$ $[_{V}$ *werk*] $[_{A/V}$ *baar*]]]

Like stems and affixes in the AAM model, derivational bases and affixes are assumed to carry markings for the kinds of units with which they can combine; in addition, an affix can change the category of its base. In this case of *onwerkbaar*, the first morpheme encountered by the procedure, *on-*, subcategorizes for an adjective to give a new adjective (A/A), but its immediate successor to the right, *werk*, is marked with the category verb (V). This prevents *on* and *werk* from being combined at this point, so *on-* is 'stacked' into a temporary working storage. The next pair *werk* + *-baar* is combinable because *-baar* is marked to combine with a verb. This combination takes place, yielding an adjective (A/V), that is still embedded in an unfinished derivational computation. At this point *on-* can be reactivated. Its subcategorization requirement is now fulfilled by *werkbaar*, and the result can be passed on to postlexical processing. The process is, of course, strongly suggestive of a stack automaton, and Schreuder & Baayen consider that it might well be described by a unification-based system of rewrite rules with embedding of semantic arguments.

An important feature of the model is its explicit representation of learning and reinforcement effects that ultimately play a role in lexicalization and in diachronic linguistic development. Like the MRM model, the meta-model allows for reverse reinforcement, or feedback, from later later stages of processing to earlier ones. Whereas reverse reinforcement in the MRM model takes place only after successful word parsing back to the parsed morpheme units, in the meta-model, activation feedback from nodes in the system of conceptual representations can reinforce lexical access representations, and the concept nodes can also receive reinforcement from successful

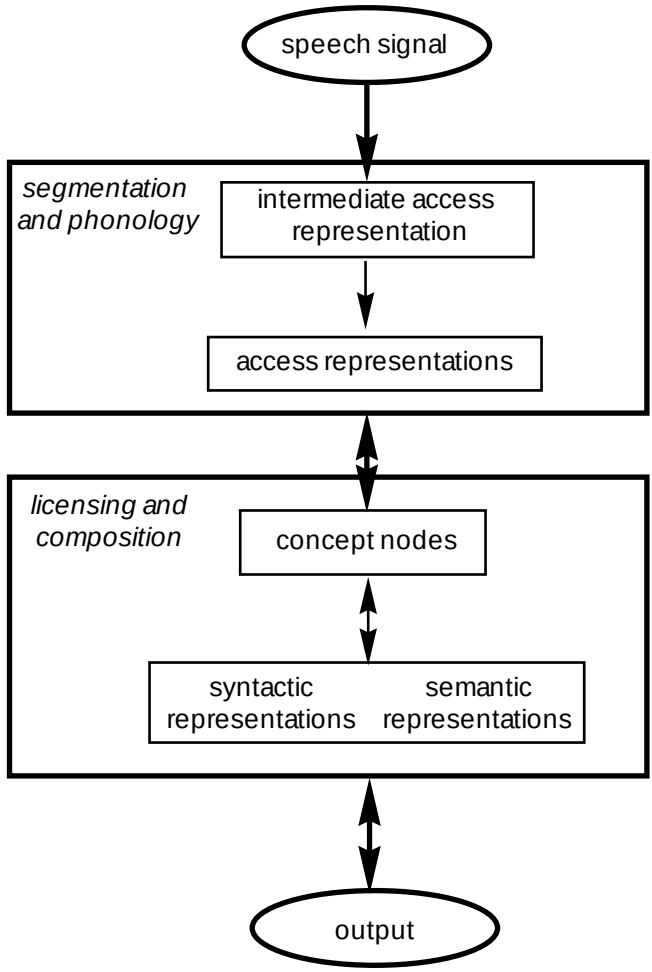


Figure 3.4: Processing Stages in the Morphological Race Model.

activation of syntactic and semantic representations. These relations are indicated by double-headed arrows in Fig. 3.4.

3.9.4 Lexical 'Caching' in Computational Systems

In symbolic computation, an often employed device for 'learning' repeatedly used results of computations is to place their results in an intermediate storage area or "cache" that is searched before starting each full-scale search. In a model of lexical search for non-lexicalized complex words, a cache might be used to store the results of word analysis. On encountering the word again, the lexical access procedure would first consult the cache to see if the word had already been analyzed, to save carrying out the analysis again. Lexical access could, however, also make the results of word analysis available as a secondary search result, in case the cached analysis did not fit the sentence context. The learning effects that the Meta-Model of [SB95] is meant to describe might be simulated by such a model if a result lying in the cache were also given a precedence weight based on the number of times it had been accessed — the more often it was accessed, the higher its precedence would be relative to the result that could be

obtained from a fresh analysis. As chapter 5, section 5.8.10 will describe, this is in fact the solution that has been implemented in the KLU model. Curiously, the possibility of lexical caching was mentioned in passing by Dalrymple et al. in [DKK+87], and it was proposed but apparently not implemented for the morphological module of the Alvey Natural Language Tools [RRBP92] as a means of obtaining quicker access to frequently encountered complex words. Ritchie et al. believed that the cache, however, "would be of little linguistic interest" [RRBP92, 177], and they did not explore its use further. Yet it is clearly evident that the psychological data impose some requirement of this sort, and the next chapter will present linguistic requirements that point in the same direction.

3.10 Requirements from the Experimental Data

The psycholinguistic data that have been presented in this chapter suggest a number of significant constraints on an overall model of word processing. We have seen good evidence that a cognitively plausible model of word processing must be able to break words down into morphological constituents rather than the simple orthographic or phonological substrings investigated by Taft and Forster [TF76], [Taf88]. Further, it is evident that the constituent morphemes participate in the process that recognizes words in a number of different ways. While stems in general have a decisive influence on recognition times, affixes apparently do not, and priming effects are much more visible with stems than with affixes [Dre89]. This may reflect the likelihood that the lexical representations of stems are much more complex and time-consuming to access than those of affixes and thus can benefit more strongly from priming. The stem frequency effect appears strongest in combination with inflectional affixes and least strong with unproductive derivational affixes [Bra80], suggesting that, according to category, affixes may combine more or less strongly with their stems or be more or less accessible to the parsing mechanism. Drews' priming results [Dre89] with inflected verbs indicate, moreover, that affixes and stems may be processed at different, non-overlapping stages of the recognition process.

Nevertheless, some data also show that complex words are not necessarily analyzed during recognition and can be stored as combined units, especially when they have high frequencies or have irregular inflection. This has been demonstrated by Taft's [Taf79b] result for the decision latencies of high frequency, inflected nouns in English, and the result of Baayen et al. [BBS97] for high frequency, dominant plural nouns in Italian. The error counts of Stemberger & MacWhinney [SM88] and Clahsen [Cla97] also show that irregular and high frequency verbs are much less likely to be produced with errors, suggesting that they are fetched as wholes from the lexicon rather than being combined with their inflections during production.

Many languages mark a frequently used form, such as nominative singular for nouns or third person singular present for verbs by displaying no overt inflection. In languages where the corresponding forms are inflected, these forms nevertheless appear to function as the lexical points of reference or defaults for the other inflected forms. This Nominative Case Effect [LKFT83], [Gün88], suggests that these forms, or at least their inflectional attributes, may have special representations, allowing faster access than for the non-default forms.

The data gathered by Caramazza et al. [CLR88] and Badecker and Caramazza [BC89] on the processing of Italian inflected forms give probably the most detailed picture of morphological processing to be found in the literature. These studies indicate that morphological parsing proceeds in steps very like those that would be entailed by a unification-based word parser, with two additional steps. One checks for the presence of a full-form entry in the lexicon to avoid the effort of parsing, if it is not necessary, and the second checks after parsing for the presence of a suppletive form that would invalidate the parse.

Finally, the published efforts to synthesize the important experimental data into cognitive models have reached a consensus that word recognition must activate multiple modes of access, both to whole-word representations and to the constituent morphemes. The most comprehensive model [SB95] is nevertheless not able to account for how newly coined words achieve lexical status and become increasingly opaque.

In summary, nearly all of the experimental data described in this chapter have significant implications for the early stages of word processing, where an input pattern is matched to recognition templates of some sort in memory. Yet word processing involves much more than this, as we shall see in the next chapter. A major issue for the construction of a comprehensive processing model will be to find how models of low-level processing can be integrated into linguistically motivated models of derivation and of sentence processing. Careful consideration of the interface from derivation to sentence syntax and semantics will reveal, in the following chapters, that word processing involves far more than word recognition and parsing alone.

Chapter 4

Linguistic Data on Word Formation

The previous two chapters have uncovered a number of constraints that will apply to potential models of word processing, but they have presented few details about how specific words come into being or about how they can be integrated into the meaning of a text in a given language. In engineering terms, these constraints supply outer-level requirements: they do not tell how to construct the model, nor do they even tell much about the system to be modeled, but they do tell us things that must be true of a model of word processing once it is constructed. This chapter will draw on a third source of outer-level requirements, namely, data about the formal structures of languages that indicate how a large-scale factorization of the linguistic system might be achieved. It will also develop some examples of inner-level requirements from formal linguistic analyses of word and sentence structures.

Linguistic analyses provide the specific, detailed information we need to write inner-level specifications of how the meanings of complex words enter into the communicative use of language via sentences. They are also the final court of appeal where the outer-level claims of other stakeholders must be weighed against one another and against data about specific human languages. As was argued in chapter 1, however, formal linguistic analysis cannot not tell the whole story about how a given language functions. As Descartes observed in the *Meditations* [Des62, 57], a parrot can imitate words and sentences of a human language, but it does not understand or produce language as such. Neither would a mechanical or computational parrot pass as a convincing or valid simulation of language processing. Inner-level requirements and a purely logical, i. e., declarative specification of a language-processing model are likely by themselves to lead to a limited apparatus describing utterances devoid of sense, that is to say, to a computational parrot of one sort or another, which by itself is unlikely to persuade anyone that it 'knows' what human beings know about how to use a language. For this reason it is worthwhile at this point to take a second look at the outer-level requirements which a natural language simulation ought to meet and their relation to the inner-level requirements that can be obtained from detailed linguistic analyses.

4.1 Correctness and Validity of Linguistic Analyses

The statistical and psycholinguistic studies examined so far have been used to identify outer-level requirements of a model of word processing, but it may not be obvious why they cannot also be counted as contributing to the inner-level requirements of the model. They are, in fact, often presented as formally explicit models, such as the Zipf-Mandelbrot statistical model [Orl82] or the Augmented Addressing Model of Caramazza and his collaborators [BC89]. Some, such as the Morphological Race model of Schreuder and Baayen [SB95], are in fact being developed into computer implementations. What these statistical and psychological models describe however, are at best frameworks in which one *could* develop linguistic simulations. The output of the Zipf-Mandelbrot model, for example, will be strings of words having the type and token distributions of natural language texts, but devoid of sense. The outputs of the psychological models that have been examined are reaction times, simple judgments about word-like symbols, and, implicitly, about relationships among words, sometimes in response to other words, but rarely in situations that anyone would characterize as authentic use of language. Both models thus account in various ways for data associated with the production and analysis of words and texts, but they deal only peripherally with the communicative function of language, which consists essentially in the production and reception of well-formed and meaningful utterances. To be convincing as simulations of word processing, they would need to be augmented in ways that simulate the communicative use of words.

However, unlike statistical distributions and reaction times, what constitutes the communicative use of words is hard to circumscribe. If an adequate simulation of word processing means simulating the use of words to communicate information, intentions, desires and the like, then the scope of the required simulation is virtually boundless, equal perhaps to a simulation of general human intelligence. It is hard to imagine how to judge the adequacy of such a simulation. Precisely this problem, that of deciding whether a computer can simulate general human intelligence, was posed by A. Turing in his essay of 1950, "Computing Machinery and Intelligence" [Tur63]. Turing's answer was that it is not possible to define a "test" for a simulation of intelligence in the form of a finite set of questions and answers (like a Binet intelligence test), but the validity of a simulation can be decidably judged in a blind comparison of the simulation's relevant behavior with that of a human being. Turing's idea was to require both the computer simulation and a human control to carry on "intelligent" conversations with an interlocutor, who was required to identify which was the simulation. The simulation could be seen as successful if the interlocutor could identify it correctly no more than half the time, i.e., no more often than by chance. In terms of the software engineering terminology introduced in chapter 1, Turing's test decides the VALIDITY of a simulation of "intelligence", without entailing any specific tests for CORRECTNESS, i.e., a specification of how intelligence is to be imitated.

Hopefully, linguistic knowledge is not as difficult to define as general intelligence, but the task of specifying what constitutes competence in a particular language raises similar issues. As was argued in chapter 1, specific linguistic knowledge is accessible to investigation only via a priori mental judgments made by members of a language community, and like "intelligence", the structure of a language cannot be derived from underlying statistical, psychological or other causes, nor can it be identified with the

information contained in encyclopedias, dictionaries and grammar texts. The only conceivable test of the validity of a linguistic simulation is a comparison like Turing's test between the simulation's performance and the linguistic performance of human speakers. Since we have no direct access to the structures constituting the competence system of an ideal speaker, linguistic generalizations can only be seen as guesses about the rules and structures being used in a language community. To test their validity, the best we can do is to submit a grammar of a language and each of its subsystems to a kind of "linguistic Turing test" [May95], which compares the simulation's performance with the performance of human speakers and listeners. Whereas in Turing's test, the question to be decided was whether the computer simulation was distinguishable from natural human intelligence, here the questions can fortunately be much more restricted in scope; they are, in fact, exactly those that linguists have long used to gather data about languages and to construct grammars: what constitute well-formed sentences or words in a language; when are words or sentences equivalent or different in meaning. If adequately formalized, linguistic descriptions can be translated into production systems to derive possible utterances, or they can be turned into rule-based analyzers that produce judgments of meaning and acceptability. A linguistic description of a language can be seen in software engineering terms as a SPECIFICATION for a simulation of certain formal properties of linguistic utterances, which can lead to an IMPLEMENTATION of the grammar as a computer program. This implementation can be tested for CORRECTNESS by checking whether or not it produces the utterances or analyses required by the linguist's description of the language. Formal correctness does not demonstrate the simulation's VALIDITY, however (cf. section 1.4). The validity of a simulation of word and sentence processing can ultimately be demonstrated only by a blind, open-ended comparison of the simulation against the linguistic performance of a human speaker, by checking from every conceivable perspective whether the simulation uses and responds to various kinds of words, including neologisms, in the way that native speakers do.

The previous chapters have gathered some, if not all, important criteria for the validity of the proposed simulation; in this chapter we look at the kinds of data that will be necessary for obtaining correctness as well.

4.2 Linguistic Analyses as Modeling Requirements

A central though implicit assumption of the studies described so far has been that word structure and word access can be factored out as a well-circumscribed module or component of the overall linguistic system. In this chapter we will confront for the first time the possibility that word structure, which has been the center of attention in the previous chapters, is not necessarily an independent modeling domain at all. Unlike the work in lexical statistics and psycholinguistics, linguistics research of the last three or four decades has focussed more on sentence than on word structure, and many researchers have tended to see word structure as a sub-system of syntax. On this view, the construction of words cannot be studied separately from the construction of sentences.

While persuasive arguments can still be made for this position, the weight of the evidence now seems to favor a factorization of grammar into two fairly independent

modules, one responsible for word formation and one for syntax. The notion that syntax is not involved in the construction of words has been called the LEXICAL INTEGRITY Hypothesis. Linguistic studies surveyed in the next sections show that Lexical Integrity must be seen as a crucially important requirement for the overall architecture of a computational simulation of word formation, but that it poses a number of special problems of its own.

We shall see that, as a modeling requirement, Lexical Integrity must be refined in significant respects. Elaborating the dual representation requirement already presented in the Augmented Addressing Model of Caramazza et al. (section 3.9.1), linguistic data will show that the semantics of some complex words can be calculated compositionally from the semantics of their constituents, but that other words have easily interpretable meanings which cannot be composed directly from their morphological constituents. In addition, completely opaque meanings of some words can paradoxically co-exist with their transparently available meanings. This will require a form of representation in which word semantics are computed in ways sometimes differing from those used for sentences, and in which transparent, semi-transparent, and opaque meanings can be presented simultaneously.

We must also re-examine the question of whether the lexicon can be defined extensionally, as it often is in psychological and computational models, or whether is must be defined intensionally, as the (unbounded) set of well-formed, possible words. Like the statistical evidence presented in chapter 2, linguistic evidence appears to speak for an unbounded lexicon that cannot be represented in finite storage. Stated as a modeling requirement, however, this evidence would lead to a lexicon that either demands potentially unbounded amounts of storage, or that otherwise attributes the spontaneous introduction of new words to the lexicon. It appears to conflict, therefore, with the necessary boundedness of any physically implemented storage device, and it is not amenable to any of the model-building formalisms that define spontaneously creative processes as the exclusive domain of syntax.

4.3 Requirements raised by the Boundary between Syntax and Lexicon

A recurring issue in the psycholinguistics literature discussed in chapter 3 is the question of whether words necessarily are the symbols processed by the linguistic system, or whether smaller symbolic entities, like morphemes, might not be the actual processing units. A parallel problem runs through the literature in theoretical linguistics of the last 30 years or so. It is the question of whether words, as structural units intermediate between morphemes and sentences, require any independent representations. This problem is to some extent an artifact of the generative program introduced by Chomsky in [Cho57], for structuralist linguistics was content to see a hierarchy of constitutive units in language, extending from the phoneme, as the minimal distinguishing feature of utterances, through the sentence as a maximal syntactic unit. By contrast, a generative description of a language based on production rules requires a language to be factored into just two main components. One comprises a set of atomic symbols that are formally equated with a language's lexicon; the other is a set of rewrite rules for expanding a single start symbol into a string of atomic symbols that make up a sen-

tence. The production rules can be thought of as nodes in a graph, and the history of the expansion builds up a tree describing the constituent structure of a phrase or of the sentence. The set of rules describes the syntax of the language. Although this bi-partite factorization into lexicon and rule system is somewhat arbitrary and tends to disregard many important observations about languages, it is nearly impossible to draw on the important recent contributions in theoretical linguistics without accepting it implicitly as a framework for describing linguistic data. For this reason, to a greater extent than in the previous chapters, the expression of many linguistic requirements will unavoidably have the character of preliminary formal specifications. Yet it is important to bear in mind that how various researchers classify items as "rules" and "lexical items" is not entirely empirical, and such classifications cannot be counted as givens of the linguistic requirements in the way that judgments of grammaticality or meaning can. [1]

4.3.1 Morphemes and Words in Production Systems

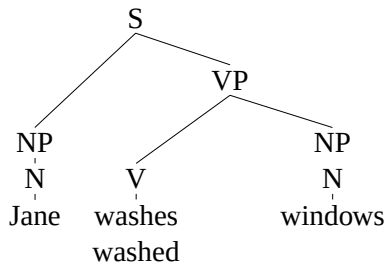
However one chooses to account for sentence structure, it is evident that differences in sentence structure are intimately reflected in corresponding differences in sentence meanings. Likewise many similarities in meaning between sentences can be attributed to similarities in their structures. One way of investigating whether sentence structure extends into word structure is to see whether such correspondences can be followed into the internal structures of words that make up a sentence. Consider some of the minimal differences in meaning and structure evident in (17).

(17) a. Jane washes windows.
b. Jane washed windows.
c. Windows were washed by Jane.

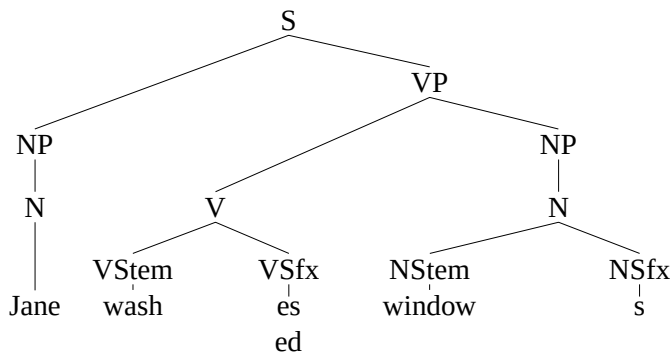
It is evident that (17 a) and (17 b) have identical syntactic structures, but the truth-conditional semantics deriving from these structures are somewhat different, (a) referring to a contemporaneous event and (b) to an event in the past. In (b) and (c) the semantics are the same, but the syntactic structures of the sentences are different. To account for the difference between (a) and (b) in a production rule grammar, one can assume, in accord with Lexical Integrity, that *washes* and *washed* are drawn directly from the lexicon, where they are distinguished by tense, and are otherwise associatively connected by some kind of synonymy. In this case, a phrase structure analysis like (18) will lead to sentence meanings differing according to the semantic difference between *washes* and *washed*; however, the semantic similarities between the sentences can only be explained by reference to a similarity between *washes* and *washed* that must somehow be available via the lexicon. Alternatively, a grammar can account for the semantic similarity by letting the system of phrase structure rules proceed to a morphemic level where the stem morpheme *wash* and its inflections *-es* and *-ed* are the terminal symbols of the grammar, as in (19).

[1] Production rule systems as a notation for specifying the model of derivational morphology are described in more detail in chapter 5.

(18)



(19)



In this case the same stem is inserted in both sentence structures. The analysis violates Lexical Integrity, but it transparently explains the sentences' semantic similarity, with the difference in sentence tense resulting solely from the differing tense morphemes.

In case (c) the syntactic structure has changed relative to (b), while the meaning remains the same. Here what the sentence grammar must explain is the semantic similarity of two structurally different sentences. It is possible to explain part of this equivalence by invoking transformations that move the object *windows* in (b) to the subject position of (c) and place (b)'s subject after a meaningless preposition *by*, on the stipulation that such transformations do not change meaning (as in [Cho65, chapt. 3]). Nevertheless, the semantic equivalence of the passive participle *washed* to the (distributionally different) active form *washed* has to be accounted for. If Lexical Integrity is observed, both forms of the verb must be inserted whole from the lexicon, with the result that the syntactic analysis is unable to explain the semantic equivalence of the sentences; only the lexicon can establish the similarity. Otherwise, throwing Lexical Integrity over board, the syntactic transformation can be allowed to attach a participle marker *-ed* to *wash*, so that both the active and the passive analyses derive the sentence meaning from an identical verb stem.

Although the morphemic decomposition appears in both cases to give a fuller and more compact explanation of the semantic equivalences among (17 a, b, and c) by virtue of the shared stem *wash*, in practice morphemic decomposition is often not so straightforward. To be of any use, the operations of syntax must be formulated in such a way that they operate freely and without reference to complex semantic constraints. But in fact, as J. Bresnan has observed [Bre82c], the passive-active equivalence does not generalize across all transitive verbs, e.g., *the hat fits you well* vs. **you are fitted well by the hat*, or

(20) a. The (many-headed) Hydra craned a neck.
b. *A neck was craned by the Hydra.

A similar issue can be raised where a past participle of a verb is used as an adjective. The simplest way to explain the semantic similarity of the adjectival meaning of e.g., *washed* to the meaning of the verb *washes* would be to let syntax attach the participle ending *-ed*. The fact that the semantic modification undergone by *windows* in *Jane washes windows* is the same as in *the washed windows*, namely BECOME(WINDOWS, CLEAN), would fall out of the syntactic analyses automatically. However, when adjectives are formed from the active verb stems in (21 a) and (21 b), the same syntactic operation leads to the unacceptable sentence (21 c) [Bre82c, 31].

(21) a. Piano music pleases Mary .
b. Mary likes this piano music.
c. *Pleased Mary listens to this liked piano music.

As these examples indicate, it is harder to identify the boundary between syntax and lexicon than one might think at first. It is deceptively easy to formulate succinct, inner-level requirements describing separately the well-formedness of both sentences and words using production rule systems, but it is not clear what might be required to integrate the two rule systems into a comprehensive model. The factorization required by Lexical Integrity has in fact been one of the most recalcitrant issues in the recent linguistics literature, and it has served as a proving ground for some of the best argumentative artillery available in current theoretical linguistics. In the next section I shall try to give a picture of some the important lines of argumentation, finally drawing my own conclusion in favor of an elaborated version of the Lexical Integrity Hypothesis.

4.3.2 For and Against Lexical Integrity

Lexical Integrity has been proposed in a number of forms; in Bresnan's stringent formulation,

(22) only fully inflected and morphologically complete words are lexically inserted into phrase structures. ...rules that alter word structure must be pre-syntactic rules of morphology [Bre82a, 307].

The opponents of Lexical Integrity have accordingly attempted to find tell-tale evidence of syntactic principles working within word boundaries to build up word structure, and the debate has centered on how to interpret the evidence they have offered.

One of the classical tests for identifying the constituents of a sentence is to check whether a suspected constituent, like *when the butter starts to melt* in (23 a), can be reduced to an atomic category, e.g., a pronoun or adverb, like *then* in (23 b). Another is to check whether anaphoric or cataphoric reference to the constituent is possible, in the way that *then* in (23 c) refers to the adverbial clause *when the butter starts to melt*.

(23) a. Shake the mixture vigorously when the butter starts to melt.
b. Shake the mixture vigorously *then*.
c. Do I $then_i$ shake the mixture, [when the butter starts to melt]$_i$?

On the assumption that word constituents are also constituents of sentence structure, it should be possible to replace word constituents with pronouns and to refer to them

anaphorically. But as was observed early in the generative literature (by P. Postal, 1969, cited in [Spr88, 291]), words within a word-like structure seem to be opaque to anaphoric reference, as is *truck* in (24 b),

(24) a. Drivers of $trucks_i$ fill $them_i$ up with diesel.
b. *$Truck_i drivers$ fill $them_i$ up with diesel.

and they cannot be replaced by pronouns, as in (25).

(25) *Bill was a $McCarthy_i$ite and Fred was also a him_iite.

The ungrammaticality of constructions like these has been one of the most important sources of evidence that the analysis of the sentence into constituents stops at word boundaries, and that constituents within the word are formed in a separate component of the grammar responsible for building lexical items. At the point where words are inserted into sentence structure, examples like (24) and (25) seem to show that word-internal constituent structure is no longer visible to anaphoric reference nor to other syntactic operations.

The constituency case to be made for Lexical Integrity is not entirely straightforward, however, and it is still vigorously contested by a number of scholars. Pronoun substitution tests for constituency actually provide poorer evidence than required because they demonstrate constituency only at higher levels of phrase structure; e.g., *it* can substitute for *the red hen* but not for *red hen* (to give **the it*) or *hen* alone (to give **the red it*). Since pronoun substitution does not work within the noun phrase, it could not be expected to work at the level of noun constituents. Lexical Integrity may be needed to explain why *him* is not allowed to replace *McCarthy* in *McCarthyite*, but as R. Sproat has argued [Spr88], it is not necessary for compounds that do not contain proper names. Within a word like *henpecked*, *hen* is not a full noun phrase, which means that *it* cannot replace *hen* by itself anywhere in the grammar, including word structure. This syntactic constraint alone suffices to explain why **it-pecked* is ill-formed, without invoking Lexical Integrity. Moreover, given that many complex words, like *luddite* and *ampere-meter*, contain proper names that do not in any way refer to persons, it is not even clear that *McCarthyite* must contain a syntactic or proper NP that could be a candidate for replacement by a pronoun.

Explicit evidence against Lexical Integrity seems to be provided by the many phrasal compounds that evidently contain sentence-level constituents, as in *I don't like his I'll-think-it-over-attitude*. To demonstrate the possibility of syntactic structures occurring inside of word structures, R. Lieber [Lie92, 11] cites a wide range of examples from English, Afrikaans, Dutch, and German including

(26) a. a connect the dots puzzle
b. God is dood theologie 'God is dead theology' (Afrikaans)
c. lach of ik schiet humor 'laugh or I shoot humor' (Dutch)
d. die Wer war das Frage 'the who was that question' (German).

In these examples, structures that normally would be formed by the syntactic rule system appear inside structures that are evidently the result of word formation, which would indicate that both systems — if there are indeed two — must be able to work

together. Moreover, like simple compounds, phrasal compounds can often take possessive markings (*a friend of mine's book*) and case marking (Lieber cites Warlpiri examples as evidence [Lie92, 15]), and they can participate in further morphological processes like nominalization and verb derivation [Lie92, 16-17]. Since morphological possessive and case marking are word-forming operations, these phenomena suggest that the rules of phrase structure do not stop being available inside word boundaries.

Lieber also claims that principles of sentence-level syntax explain why (27 a) is well-formed but (27 b) is not. She argues that the theory of case assignment in syntax requires the complement of a verb to be a maximal phrase, as described by X-bar theory. Hence, the direct object position following the verb *quench* is reserved for a constituent that can only be a noun phrase, like *my thirst* or *some thirst*. To build the compound using an unspecified (non-maximal) object like *thirst*, the only available position for the combining noun is the unrestricted position ahead of the verb [Lie92, 60], as in (27 a).

(27) a. thirst quencher
b. *quencher-thirst

She claims that the same principle blocks phrasal constituents from occupying the preverb position of synthetic compounds like (28 b)

(28) a. flea-bitten
b. *very tiny flea-bitten
c. Old Testament-enthused theologians

Lieber takes these and similar phenomena as evidence that at least part of the word-formation apparatus must overlap with the apparatus for producing sentences; otherwise, the rules and processes of word-formation would need to duplicate a large portion of the rules and processes provided for syntax.

There can be no doubt that word formation is similar to sentence construction in many respects, including the projection of features and of argument structure, and Lieber goes on to give an extended account of the many parallels that can be found. Nevertheless, for many of her examples, an appeal to word formation processes borrowed from syntax is not necessary. The fact that English places the object argument ahead of a verb in a synthetic compound like (27 a) probably has more to do with the morphology of English compounds than with syntax, because English compounds tend to put the element carrying the most specific semantic information at the end, and modifying constituents in preceding positions. From the perspective of compounding, *quencher* in (27 a) is a noun modified by *thirst*, just as *book cover* is not a kind of *book* but rather a specific kind of *cover*. Comparable French compounds like *porte-parole* 'carries-speech [of someone else]' = 'press representative' in any case show no evidence of obeying the syntax of French case assignment. They place a case-unmarked object (*parole*) after the verb (*porte*), even though the syntax of French verbs is fairly similar to that of English, with the object noun phrase usually following the verb and usually requiring a case-marking determiner. Furthermore, while not every maximal phrase can be placed in the leading position of an English nominal compound, the possibilities are surprisingly broad, as (28 c) suggests.

Drawing on arguments from lexical phonology to which I shall return in chapter 6, H. Clahsen et al. argue that phrasal compounds actually demonstrate a much different sort of phenomenon, one that they call "quoting" [CMBW96]. On their account, only word-like constituents that have grown so familiar as to be 'quotable' can participate in word formation. What makes **very tiny flea bitten* unacceptable is thus exactly the presence of a genuine syntactic constituent within the domain of word formation. Yet there is no doubt that often-heard syntactic constituents, even sentences, can become listed lexical items (or *listemes* in the terminology of DiSciullo and Williams [DW87]), and in this form even entire, well-known sentences can function as single, word-like constituents of compounds, as in *Hoover's let-them-eat-cake-speech*. It remains true, of course, that within phrasal compounds phrases are subject to the same well-formedness constraints as in sentences, and that in this purely descriptive sense operations of sentence syntax are visible inside the boundaries of word formation. This is not equivalent to saying that the syntactic rules by which such phrases are formed are part of the apparatus of word construction; to the contrary, the phrases are probably formed separately from the words, and they can only be incorporated in words when they already exist as single, fixed entities. Moreover, lexicalized phrases used in compounds are sometimes drawn from foreign languages, as in *Kennedy's Ich-bin-ein-Berliner-speech*, or a *c'est-la-vie-attitude*, as Bresnan & Mchombo point out [BM95]. To account for the formation of such phrasal compounds in syntax would require expanding the syntax of English to include additional syntactic fragments of all the languages that can contribute to phrasal compounds.

The issue is therefore not that of whether constituents found at the phrasal level can also appear in words, but rather whether they get into words in the same way. There can be no doubt that words do exhibit constituent structures that are to some degree like the structures of sentences; that certain positions within words are reserved for modifying elements; and others for head-like elements containing syntactic features like gender, tense, etc. Recursive application of word-formation rules is also easy to document, as in the Dutch example

(29) $[_{A/A}$*on* $[_{A}\ [_{V}$ *werk*$]\ [_{A/V}$ *baar*$]]]$

meaning 'unworkable' analyzed by Schreuder & Baayen [SB95] (cf. p. 100). The important point of difference is that, in general, though similar to those of syntax, the operations involved in word formation are simpler in several ways. Moreover, these operations do not interact with those of sentence syntax, and there is no sharing of constituency between word and sentence structure.

Bresnan & Mchombo [BM95] define five specific conditions that distinguish word from sentence constituents, which I shall characterize as five ways in which the rules of word formation are impoverished with respect to syntax. A sixth, the Grammatical Function Impoverishment, may be implied by their observations; it is proposed explicitly in a different framework by Anderson [And92].

(30) a. Word constituents cannot be replaced by syntactic operations like relativization, topicalization or clefting (Extraction Impoverishment).

b. Word constituents cannot be conjoined by syntactic conjunctions (Conjunction Impoverishment).

c. Word constituents cannot be deleted by ellipsis (Ellipsis Impoverishment).

d. Word constituents are not used anaphorically or deictically (Anaphora and Deixis Impoverishment).

e. The rules of word formation project but do not make use of grammatical functions (Grammatical Function Impoverishment).

f. Phrasal word constituents are not generated directly by syntax (Phrasal Recursion Impoverishment).

By and large Bresnan & Mchombo's restrictions on word formation can be characterized as a more general principle that I shall call the IMPOVERISHMENT OF MORPHOTACTICS. This says that the apparatus available for constructing words is a much-reduced version of that available for sentence construction (although ontogenetically it may well be the other way around: that syntax arises from an elaboration of word formation). Specifically, the formal apparatus available for building words (which I shall henceforth call morphotactics) prohibits the indexing devices used in syntax for linking sentence constituents (like pronouns and anaphoras); it does not provide non-referential structuring or function morphemes, like coordinating and subordinating conjunctions; and it lacks the apparatus that supports grammatical functions, like the positional and case markings used in sentences to identify subjects, objects, etc. Morphotactics may also lack full recursivity. S. Anderson [And92, 40-42] identifies a number of substantively similar restrictions on rules of word formation that set them off from the rules used for forming sentences. However, Lexical Integrity in the form (22) proposed by Bresnan [Bre82a] is a stronger condition than the Impoverishment of Morphotactics. It says not only that many operations allowed in syntax are not available in word-formation, but also that word formation takes place separately from sentence construction, so that sentence construction deals only with well-formed, complete words.

In more detail and with some of the important evidence, here are the ways in which word formation can be seen as a reduced or impoverished version of the system that describes sentence formation.

Extraction Impoverishment. It is evident that the apparatus of sentence syntax often allows constituents to be rearranged for the purpose of changing emphasis or focus. These operations include topicalization of a constituent by movement to a more 'visible' position, as in (31 a), or by placing the constituent in an initial clause with a focus-attracting *it*, as in (31 b). A similar sort of operation replaces a constituent with a referential link, like the relative pronoun *who* in (31 c).

(31) a. I can see Bill at football. $\longrightarrow$ Bill I can see at football.
football-player $\longrightarrow$ *player-football

b. Bill plays football $\longrightarrow$ It's Bill who plays football.
football-player $\longrightarrow$ *it-player of football

c. This is Bill. Bill plays football. $\longrightarrow$ This is $Bill_i$, who_i plays football.
This is $Bill_i$, a *football-who_i.

Similar operations within words lead to items that are grossly ungrammatical, and in fact focus and topicalization within word structure are impossible for the simple reason that the order of word constituents is rigidly fixed. Anderson describes a somewhat similar principle, which he calls "locality" in lexical (or word formation) rules; this states that the operation of such rules is restricted to the immediate subcategorization frame of a lexical entry [And92, 40], making them incapable of moving constituents to other, remote positions.

Conjunction Impoverishment. Coordination is regarded by many as one of the surest tests of constituency at all levels in syntax, and a variety of coordinate structures within words would be a strong argument in favor of seeing word structure as being governed by the same structural rules as sentence structure. For example, the possibility of conjoining a complex verbal phrase to a simplex noun, as in *I dislike fleas and having to wash dogs with them* suggests that *having to wash dogs with them* is of the same constituent category as *fleas,* and that syntax is able to operate on both in the same way. Similar coordinations to or among constituents of a word would suggest that those structures over which coordination applies in syntax might be available as part of the word formation system. But in fact, of the conceivable word-level coordinations, most do not produce well formed constructions, as Bresnan & Mchombo illustrate with examples like **John was unwilling and -able to give up* or **John's joyful- and cheeriness kept us going* [BM95, 188].

Coordination of word constituents is not impossible, however, and one does not have to look far to find examples. Bresnan & Mchombo list, among others, *infra e ultrasuoni* 'infra and ultra-sounds' (Italian), *Freund- oder Feindschaft* 'friendship or hostility' (German). They see these, however, as syntactically coordinated noun phrases that have been subjected to a kind of phonologically licensed ellipsis. Following studies of Nespor [Nes85] and others, they propose that such examples may be limited to a class in which the prefix belongs to a phonological word separate from the stem. Where this is not true, as with unstressed derivational prefixes in German (e.g., *bewundern und *-dauern* 'admire and lament'), they observe that coordination is not possible. Bresnan & Mchombo's explanation is a bit forced, however, because phonological structure functions independently; it does not directly reflect either syntax or word-structure. (In *She's here* the phonological word *she's* spans two syntactic constituents, the subject and the verb.) It is thus hard to see why the phonological data would provide evidence that the ellipsis takes place in syntax rather than in word-formation.

Closer examination reveals that in these and many similar cases, when phonologically permitted, the conjoined morphemes must also occur together in some lexically familiar way, e. g., as lexical opposites *infra* vs. *ultra*; *Freund* vs. *Feind* 'friend' vs. 'enemy'; *ein* vs. *aus* in constructions like German *ein- und ausgehen* 'go in and out', or as lexical synonyms, as in *zu- und dichtmachen* 'make closed and sealed'. When a close lexical relation does not obtain, and two unrelated concepts must be combined into a single syntactic constituent, morphological coordination is not acceptable in German. Noun phrase coordination in the subject of *Freundschaft und Wissenschaft wurden gelobt* 'Friendship and science were praised' cannot be reduced to a morphological coordination **Freund- und Wissenschaft wurden gelobt*. In many cases coor-

dination in compounds does not lead to a plural noun, although a similar coordinated noun phrase would take a plural verb, e.g. *Die Industrie- und Handelskammer ist umgezogen* 'The Chamber of Industry and Commerce has moved'. This coordinated pair evidently enters word formation as a single item rather than as a coordinated pair, indicating that no ellipsis (from *Industriekammer und Handelskammer*) is involved. In apparent violation of this word-formation rule, the comical name of a Munich cabaret, *Lach- und Schießgesellschaft* 'Laughing and shooting society' seems to derive its humorous effect from implying that laughing and shooting are closely paired in some lexical sense relation; if syntax were responsible for the coordination, the humorous effect would not arise.

Ellipsis Impoverishment. Closely related to the Conjunction Impoverishment is the prohibition on ellipsis. In syntax it is often possible to eliminate certain repeated constituents, as in *John liked the play and Mary, the movie.* In what sense ellipsis is really a syntactic operation at all is debatable, but in any case implied word constituents cannot be reconstructed from context, as would be required in *John liked the play and Mary* dis-*it*, for *disliked* [BM95, 189]. I suspect that in Bresnan & Mchombo's coordination examples, (*infra e ultrasuoni*), the apparently deleted constituent (*suoni* from *infrasuoni*) was never present. It is rather the lexically coordinate pair (*infra e ultra*) that combines with a single base.

Anaphora and Deixis Impoverishment. The main facts about anaphoric and deictic reference within words have been shown in (24) and were summarized above. Unlike Lieber, Bresnan & Mchombo see the prohibition of reference from within a word to external sentence constituents (or to unnamed contextual objects) as evidence that word structure is opaque to syntactically controlled anaphoric reference. They argue that it is not pronouns as such that are prohibited in complex words, as English allows pronouns to specify gender or intensification of gender in compounds like *she-bear* and *he-man*. Bresnan & Mchombo report that in Bantu languages, pronouns can be incorporated into verbs or nouns when they stand in a determinate grammatical relation to the host, such as object or possessor; one such pattern gives a manner nominalization of the verb with an incorporated pronominal object [BM95, 190-192]. However, when no argument position for the pronoun is furnished by a lexical head, the word is ungrammatical. This suffices to explain the ungrammaticality of **him-ite* (25) because *him* would need to be the head of the word rather than an argument. The syntactic operation in which a head constituent is replaced by a pronoun is thus not available in word formation.

Grammatical Function Impoverishment. Although Bresnan & Mchombo do not note it explicitly, a possible implication of the Extraction Impoverishment is that word formation is not able to mark argument positions by any other means than fixed relative position, i. e., the argument to an affix must usually appear directly in front of or behind it. If the position of an argument is always unambiguous, it does not need any other special marking, which would make markings for grammatical functions like subject, object, etc. superfluous. Anderson cites evidence that word formation rules can make direct reference to thematic roles

associated with particular arguments, something that in syntax appears to be always mediated by functional markings [And92, 41]. Moreover, one important means of marking grammatical functions in syntax, namely case marking, seems to have no correlate within word structure. These two observations — that thematic roles can be referenced directly and that grammatical functions are not marked — suggest that word-internal structure, unlike syntactic structure, does not make use of a parallel level of functional marking or structure. (This does not prevent individual word constituents from projecting grammatical functions to the syntactic level, however.)

Phrasal Recursion Impoverishment. There is evidence that the rules describing word structure have a formal power more highly restricted than that of syntax. For one thing, end–recursive application of the rules in word-formation can be blocked by phonological or lexical restrictions. For example, for Italian Schwarze observes that a derivative in *-mente* cannnot be the input to a further derivation, nor can a polysyllabic word ending in a stressed, non-inflectional vowel, e.g., *falò* 'bonfire' [Sch95a, 603].

The generative power of the word-forming rules also appears to be less complex than those that describe sentences, not normally allowing embedded recursions. Recursive embedding of phrases in syntax at the sentence level, as in (32), was pointed out as one of the defining formal characteristics of natural languages by Chomsky in *Syntactic Structures* [Cho57].

(32) [The cat [that ate the rat [that bit the mouse]] has died].

Although it also is possible to construct words with embedded recursive structures, such as (33),

(33) [anti-[anti-missile]-missile]

there is not much evidence that these are spontaneously and naturally produced; in English, *anti-anti-missile* is more likely to be used to express this recursively defined concept.[2] In general it appears that word structure allows only restricted end-recursions, [3] and no recursive embedding, which implies that the rules of morphotactics can be described formally with regular languages[4] — i. e., with an impoverished form of the apparatus required for syntax.

Although these observations might speak for an additional Impoverishment of Morphotactics with respect to generative power, they cannot be construed as directly supporting Lexical Integrity in the form required by Bresnan (22). The statistical studies discussed in chapter 2 give ample evidence that word formation, like syntax, gives rise to spontaneous, previously unheard or unseen utterances, though probably much less actively than syntax. The simplest way to describe the spontaneous creation of new words would be to provide recursions from the generative rule system for sentences into the word formation rules. The word formation rules could then be invoked in

[2]Even with its richer morphological apparatus, Italian seems to observe a very similar restriction. Vieri Samek-Lodovici, personal communication.

[3]Schwarze [Sch95a, 602] suspects that in Italian the possibilities for recursion are limited to unrepeated affixes, as interpretable end-recursions like Italian *scal-in-in-o* 'little step' are judged unacceptable; additionally, the order of recursive diminutive suffixation is restricted [Sch95a, 514].

[4]Anderson [And92, 42] makes a similar observation.

real time, during sentence generation, to create new combinations of morphemes that would surface as words but would not be lexical items in the sense of having stored representations distinct from those of their constituent morphemes. Although it would violate Lexical Integrity, a rule system of this sort would still allow the rules of word formation to be distinct from and simpler than those of syntax. They would be prior to the rules of syntax with respect to "relative order" of application in Anderson's terms [And92, 41] and, as a distinct group, they could be subject to all of the constraints deriving from the Impoverishment of Morphotactics. An ordering restriction, placing all rules of word formation before those of sentence generation, is compatible with the previously listed constraints on word formation, but it invites seeing the *operation* of the rules of syntax and of word formation as part of the same overall generative system; i. e., it still extends the syntactic rule system into the realm of word formation.

4.3.3 Phrasal Constituents in Words Reexamined

The real issue for Lexical Integrity is thus whether word formation can be seen as a recursive phrasal extension, albeit with many restrictions, of the sentence grammar, or whether it must be described as a completely separate system, as Lexical Integrity requires. It was pointed out above that many complex words, e.g., (26), appear to contain syntactic phrases as constituents. In most cases it can be argued that these phrases are lexicalized and thus available as unstructured entities without an attached syntactic structure for word formation; indeed, Bresnan & Mchombo take as evidence for Lexical Integrity that syntactically permissible operations, like substituting a syntactically more complex but categorially similar constituent, as in (34 b) and (34 c), are not allowed inside word boundaries.

(34) a. $[_A$ happy]-ness
b. *$[_A P$ quite happy]-ness
c. *$[_A P$ more happy [than sad]]-ness [BM95, 192]

It is, however, not difficult to find phrasal compounds and derivatives whose phrasal constituents have a ring of familiarity, but allow a certain amount of syntactic variation and therefore cannot be accounted for as fixed listemes. In appropriate contexts all of the examples in (35) will be more or less acceptable.

(35) a. an ate too much headache
a'. an ate too much prosciutto headache
b. an I told you so attitude
b'. an I told you so yesterday attitude

Even if the unprimed examples in (35) are composed from listemes, their phrasal elaborations in the primed examples are unlikely to be listed; moreover, the semantic contributions of the elaborated phrases to the compounds are exactly the compositional meanings that syntax would yield. To explain this continuum of acceptability, it would appear more parsimonious to collapse word formation and syntax into a single system than to postulate that all phrases which could possibly appear as word constituents are constantly available in some unimaginably large store of common and not-so-common phrases.

Bresnan & Mchombo try to allow for elaborated phrasal constituents by asking that we "recognize that lexicalization of phrases can be innovative and context-dependent" [BM95, 194] — as if this did not raise more problems than it solves. For how can one describe a lexicon whose membership is in continual flux and can change according to the context that is active when the lexicon is queried, let alone construct a formal model of such a lexicon? "Innovative lexicalization" appears to require that when a legitimate word formation process needs a phrasal constituent, then under favorable circumstances that constituent can be introduced into the lexicon. The contradiction is evident: on one hand, Lexical Integrity according to (22) restricts word formation to "pre-syntactic rules", but on the other hand, a new phrasal constituent required within a word can only be formed by directly invoking rules of syntax. If granted the freedom to incorporate "innovative" phrasal constituents into words, the lexicon ultimately comes to incorporate phrasal syntax into morphology, and the model becomes formally similar to Lieber's syntax-oriented account of word formation [Lie92]. Bresnan & Mchombo admit that this issue is what makes phrasal recursion the most controversial of their five restrictions, but at the same time it is likely to be the issue that decides the relevance of all the others.

What the debate over Lexical Integrity reveals is not so much that the evidence for and against the hypothesis is unclear, as that to maintain the hypothesis convincingly, the lexicon must be construed as something more than an extensionally defined, unchanging, static set of insertable symbols. An intensionally defined lexicon, however, is simply not compatible with the set of terminal symbols required by production-rule formalisms, in which the combinatory, generative operations belong, *by definition*, to the syntactic component rather than the lexicon. If Lexical Integrity is construed as requiring both a strict separation of lexicon and syntax and an "innovative" capability within the lexicon, it leads to a requirement for a symbolic formalism differing in some essential way from any of the currently popular symbolic systems used for describing natural languages.[5] Needless to say, this will be a central issue when we turn to formal systems for specification in the next chapter, but hints of what form the solution must take have already appeared.

A solution of sorts to the problem of phrasal compounds and derivatives has already been adumbrated in the psycholinguistic Augmented Addressing Morphology model of Caramazza et al. (section 3.9.1), although not in the form that will be proposed for the KLU model in chapter 5. Recall that in the AAM model, frequently seen complex words at some point acquire a single, unitary lexical representation; this accounts for the whole-word frequency effect in lexical decision tasks. What is not described in the AAM model is how in the course of language acquisition the transition from a rarely seen to a frequently seen representation occurs, i.e., how the constituent morphemes used to recognize an infrequently seen complex word are gathered together and placed in a single access unit (cf. Fig. 3.3, chapter 3). Nevertheless, such a function must be available; access units cannot appear ex nihilo. Moreover, given the way that memory traces tend to decay over time, the creation of a new access unit in memory will not be triggered simply by counting tokens in the course of learning a

[5] As noted in the Introduction (p. 9), the theory of Lexical Functional Grammar seems to allow for the "innovative" operation of lexical rules, but the LFG formalism, which has been most fully worked out in [KM96], provides no apparatus for implementing spontaneous lexical innovations of the form that would be required here.

language. (Hearing a word five times in a day probably has a learning effect different from hearing the same word five times over the course of a year.) More plausible is a function that creates a whole-word access unit each time a word is parsed, but purges it soon thereafter if no further references to it occur; such allocate–deallocate functions are well-known in the literature on computational storage management and are in fact a basic component of all storage management systems in computers. Needless to say, if such a function is available for creating access units for complex words, it might be invoked to create access units for marked phrases as well, and once created during a syntactic analysis, these temporary listemes could provide the input to a syntax-free word formation system. Attempting to analyze *an I told you so yesterday attitude*, for example, word formation would be unable to deal with the phrasal constituent, but it could search for a possible lexical entity corresponding to *I told you so yesterday*. Had this phrase been somehow prominent in the immediate context, a higher level of discourse processing might hold on to it in the form of a fixed phrase with the semantics that arose in the course of sentence analysis. Temporary buffering or "listemization" of the phrase would then allow word formation to pull the phrase as a single unit from memory. Such an account provides a coherent picture of the processing of phrasal compounds, with Lexical Integrity and without phrasal recursivity, [6] that could nevertheless incorporate phrasal constituents within words. To be sure, by itself, such a solution to the problem of phrasal compounds seems implausibly elaborate, but in the context of a lexical structure that must *already*, and for other reasons, provide the same apparatus for word access, it is not implausible. We shall see in chapter 5 that such an "innovative" lexicon can be given a coherent formal specification, one that occupies a central place in the KLU model, but whose detailed specification is not obvious nor simple. Further evidence that the lexicon provides for innovative, temporary lexical entries will be forthcoming in the next sections.

4.4 Requirements Raised by the Semantics of Complex Words

The most persuasive argument for Lexical Integrity is not the evidence for differences between the rules systems of syntax and morphotactics. Rather, it is the finding that when one tries to define how the meanings of words are constructed, it is nearly impossible to use the same apparatus that works so well for constructing the meanings of sentences. When one gets down to writing detailed word grammars, it becomes evident that mechanisms unlike those of sentence grammar are at work, mechanisms that led M. Aronoff [Aro84, 1984] to complain that attempts to explain the semantics of English verb derivations by classifying them into derivational types were futile:

[6] In the KLU model described in chapter 7, this step is *not* a direct recursion from word formation back into syntax, although it may permit some syntactic analysis. It will be formalized as an error handler (cf. section 1.6) that, like the quoting mechanism of Clahsen et al. [CMBW96], requires an appeal to a level of control *above* the linguistic system. In fact, all sorts of objects, like *c'est-la-vie*, can occupy the initial position in a compound with *-attitude* and initiate a special form of lexical search. Not only phrasal structures but even non-linguistic objects like mathematical formulas (*a* χ^2*-test*) or icons, like the ♯**@!-expletives so popular in comic books, can appear as constituents of complex words.

> . . . on closer inspection, these types evaporate into the pragmatic air and the semantics of the entire diverse set can be reduced to a simple statement: the derived meaning is that of a verb which has something to do with the base noun. [Aro84, 46]

In his "Remarks on Nominalization" Chomsky [Cho70] showed that it is impractical to regard the set of morphemes in a language as the terminal symbols of sentence structure. Although the well-formedness conditions for morphologically complex words can be expressed in a system of transformational rules in the same way that sentence structure is represented, the semantics of many derived nouns is not sufficiently consistent with their bases to let the grammar derive the correct sentence meaning. To account for each idiosyncratic derivation, Chomsky showed that a morpheme-based grammar could postulate a separate variant of the base with appropriate selection restrictions, but he argued that this would inflate the grammar enormously, and it would be descriptively no more economical than providing a separate lexical entry for each semantically irregular derivative.

On the other hand, one finds many systematically derived forms, such as English gerunds, that are semantically transparent and that could be accounted for economically by allowing syntax to construct their specific meanings from the constituent morphemes. The problem is thus not the one posed in section 4.3.1 of choosing between lexical (18) and syntactic (19) representations for complex words, but rather one of how to allow for a continuum in which both are possible. The AAM proposal (section 3.9.1) suggests a partial solution for the relatively simple case of inflected words, by allowing both full-form representations of frequently used words, and word parsing for others. Since inflection merely specifies a small number of features not specified on the stem, the interface from the parsed word to syntax runs into no linguistic difficulties. At the level of derivation this interface is far less straightforward, as some recent investigations of the semantics of derivatives, presented in the next section, have shown. Not only semantic attributes, but argument structure, subcategorization, inflectional class, and other attributes may be added or changed with respect to the base. First, however, we must examine what might be necessary to give a formal account of derivation that includes the construction of the derived word's semantics; later I shall turn to the question of how semantic transparency and opacity in derivation relate to the issue of on-line analysis vs. lexical storage.

4.4.1 Some Problems in Derivational Semantics

As has already been shown, whether inflection is represented in the lexicon, as in the Full Listing Hypothesis, or is analyzed in syntactic processing, has consequences for lexical storage. For the analysis of sentence structure or semantics, however, inflectional affixes serve mainly to specify simple, unstructured features like gender, number or tense, and such features are easily integrated into the syntactic and semantic structure of the sentence. It is difficult, however, to give an adequate account of derivation simply by adding features from the derivational affix to a base. The difficulties can arise in nearly all forms of derivation, but they are most visible in cases where the derivation changes the category of the base or alters its semantic structure.

Change of category is not in itself problematic, as syntactic category can be thought of as a kind of feature like gender or tense, but with the category change often comes a

change in the subcategorization requirements of the base. Schematically, the derivation of de-nominal adjectives like *novelistic* from *novel* could be described by operations on sets of attributes and values, as in (36), where the category and the predicate are changed. (Anticipating the notation of Lexical Functional Grammar, which will be introduced in more detail in chapter 5, an indirect reference is marked here with ↑; attributes are, mnemonically, PREDicate and CATegory.)

(36)
$$\begin{bmatrix} \text{PRED} & \text{'novel'} \\ \text{CAT} & \text{NOUN} \end{bmatrix} \longrightarrow \begin{bmatrix} \text{PRED} & \text{'like(}\uparrow\text{BASE)'} \\ \text{BASE} & \text{'novel'} \\ \text{CAT} & \text{ADJECTIVE} \end{bmatrix}$$

The mere changing of attributes does not suffice, however, to describe even this simple derivation. In its sentential context a noun can take various optional complements that modify or define implicit roles (or QUALIA attributes, in the terminology of J. Pustejovsky [Pus95]) belonging to its meaning, e. g., *by Jane Austen* in (37 a). If the conversion of *novel* to an adjective, *novelistic* in (37 b), could be described by monotonically adding the property QUALITY-OF-BEING-LIKE, the base's other properties, such as literary style, would be retained, as is evident in (37 c). The property of having an author ought therefore also to be present in *novelistic*, and we ought still to be able to specify this property with an appropriate phrasal constituent. In fact, however, such an optional argument position is no longer available, as is evident from (37 d). *novelistic* no longer accepts a specification of the conceptually implied author, and semantically *novelistic* can no longer be parameterized for author. It is possible that the derivation of *novelistic* has in fact substituted an altogether different concept in the position of the base, a generic sort of novel lacking many of the specific attributes of particular novels.

(37) a. the novel by Jane Austen
b. the novelistic travelogue
c. the charmingly novelistic travelogue
d. ??the Jane Austenly novelistic travelogue.

A derivation can also eliminate syntactic access to a property that is still clearly part of the underlying semantic structure. Schwarze [Sch95a, 532] presents a kind of minimal pair of derivations in Italian that illustrate vividly how a particular derivation can uncouple a semantic argument from the list of syntactic arguments. The verb *investigare* 'investigate' presumes some matter as the object of investigation, and this matter is expressed syntactically as a direct object of the verb. This verb can be nominalized as *investigatore* or as *investigante*, both meaning 'investigator'. In both cases the verb semantics underlying the noun still presumes an object under investigation, but only *investigatore* permits a syntactic adjunct to name this matter, as in *l'investigatore di quel delitto*, 'the investigator of this crime'). The other nominalization, *investigante* 'one who investigates', would be conceptually meaningless without a presumed matter under investigation, but this matter cannot be specified by a prepositional phrase with *di*.

Another common and thoroughly representative difficulty with derivation is shown by the comparative degree of an adjective. Because of its deceptive transparency, this

derivation is sometimes classified as a form of inflection. While in the positive degree the adjective typically can be represented with a one-place predicate referring to a single argument in syntax,

(38) *the small house* $\Rightarrow$ SMALL(HOUSE))

the comparative is clearly a binary predicate

(39) *the house smaller than a mansion* $\Rightarrow$ SMALLER(HOUSE, MANSION)).

The comparison term is not, strictly speaking, subcategorized (cf. *That's better*), but if present, some means must be provided in the syntactic analysis to signal that the *than*-phrase maps to the second argument of the semantic predicate of smaller; *smaller* cannot simply be substituted into the syntax in the same way that *small* is inserted. However one chooses to represent the syntactic structure, it will be exceedingly difficult to insert both *small* and *smaller* at the same preterminal position of a sentence derivation in such a way that the differences in argument structure are accommodated. Similar observations hold for many category-changing derivations: where a noun presents no syntactic subcategorization requirements, a verb derived from it will require at least a subject (in English) and possibly other arguments. In the reverse direction, a deverbal noun loses the subcategorization of the base verb (*The enemy destroyed the city* becomes *The destruction [of the city, by the enemy] was extensive* (cf. [Gri90]).)

Another difficulty in accounting for derivations arises not in the arguments licensed by the semantic structure but in the compositional relation between the semantics of the constituents and the semantics of the derivative. To be sure, the meanings of many derived words are opaque with respect to their derivations; these words do not need to be accounted for at all in synchronic word formation, since they are lexicalized words that have a low $\mathcal{P}$ or "degree of productivity" in Baayen's terminology (section 2.9) and will almost always be available for look-up in dictionaries. But even within patterns that are in active use, it is sometimes difficult to derive the word's semantics compositionally from its structure. D. Corbin [Cor90] observes that the French suffix *-iste* combines fairly freely with nouns to give nouns meaning SPECIALIST-FOR(N), as in

(40) *bouquin* '(old) book' $\longrightarrow$ *bouquiniste* 'specialist for used books'

At the same time, one encounters other words of recent origin that are regarded as semantically transparent and must have been derived according to the same pattern, yet cannot be derived by combining the meanings of their constituents. An example is *publiciste* 'journalist, public relations expert', whose semantics is not SPECIALIST-FOR(PUBLIC) but rather SPECIALIST-FOR(NEWS/ADVERTISING). Given that *publicité* can mean something like 'news' (or 'journalistic productions'), the semantics of *publiciste* could be derived systematically from a more elaborate word structure

(41) $[[[[\text{public}]_A\text{-ité}]_N]\text{-iste}]_A]_N$

but the constituent *-ité* is simply not present in *publiciste*. A rule introducing it (or its semantics) into derivations with *-iste* would be arbitrary. Applied to other derivatives, it would lead to incorrect interpretations of words like *bouquiniste*. The derivation of the semantics of *publiciste* thus does not seem to proceed compositionally. Corbin suggests that something else, external to the visible word structure, forces insertion of *-ité* (or at least its semantics) to obtain a new noun, *publicité*, from the adjective reading of *public*; this may result from a pragmatic discovery that the noun *public* as a base would lead to a meaning that rarely serves a useful purpose. Where and when such a pragmatic evaluation takes place is unclear, however. Deferring the construction of the semantics of *publiciste* to a pragmatic evaluation of the sentence would explain how *publiciste* obtains its non-compositional meaning, but it would not explain why it is *always* used in the sense of 'journalist, public relations expert' and never as 'specialist for the masses', nor why no context is possible in which it could assume this meaning, since the simpler, compositional semantics ought also to be available. In contrast to sentence semantics, it appears that derivational semantics can be transparent without being compositional.

A number of examples from Italian, illustrating the complexity that must be accounted for in derivational semantics, have been collected and classified in B. Mayo et al. [MSSZ95], drawing in part on Schwarze [Sch95a] and on work reported in Gatti & Togni [GT91]. Here it was shown in a larger sampling of Italian derivations that both category-preserving and category-changing derivations can involve a wide range of semantic modification and restructuring operations on the semantics of the base. As with *bouquiniste*, one finds examples whose semantics seem to follow compositionally from their morphological bracketing, as in the diminutive formation

(42) pianta 'plant' $\longrightarrow$ $[_N[_N$piant]ina] 'little plant'

whose semantics can be given compositionally as LITTLE(PLANT(X)). In the derivation of a negative opposite

(43) comodo 'comfortable' $\longrightarrow$ $[_A$s$[_A$comodo]] 'uncomfortable'

the semantics can be given compositionally as NOT(COMFORTABLE(X)), while the reversative derivation

(44) favorire 'favor, encourage' $\longrightarrow$ $[_V$s$[_V$favorire]] 'discourage'

might be described as OPPOSITE-OF(FAVOR(X,Y)).

Closer examination of other, similar patterns shows, however, that more is involved than a blind mapping of morphological structure to a predicate structure. Many morphologically well-formed derivations in Italian are rejected by speakers for one reason or another. For example, the reversative derivation of an adjective

(45) marcio 'rotten' $\longrightarrow$ *$[_A$s$[_A$marcio 'not-rotten']]

although transparently interpretable as 'not rotten', is rejected by speakers. For some reason word formation appears not to tolerate the evaluation of the implied, formally interpretable semantics OPPOSITE-OF(NEGATIVE-OF(RIPE)). Further, the semantics of many Italian derivatives cannot be constructed in any strictly compositional set of operations . Consider

(46) legare 'tie (to)' $\longrightarrow$ [$_V$s[$_V$legare]] 'untie (from)'

where the agent A, theme T and localizing relation L of *legare*

(47) CAUSE(A, CHANGE(NOT(TIED(T, L)), TIED(T, L)))

become

(48) CAUSE(A, CHANGE(TIED(T, L)), NOT(TIED(T, L))).

in *slegare*. Not only does this derivation require invasive surgery to rearrange the order of the internal predications of the base verb, it also requires a change in the verb's requirement for a class of localizing prepositional phrase in the argument *L*. That is, where *legare* requires that *L* contain a localizing preposition like *à*, *slegare* requires *de* in the same position.

(49) Paolo lega il cane *all'* albero 'Paul ties the dog *to* the tree'

(50) Paolo slega il cane *dall'* albero 'Paul unties the dog *from* the tree'

One might object that reversative verbs displaying such complex semantic alteration have arisen outside of the synchronic grammar and therefore do not require a systematic, compositional description, since in Italian both *legare* and *slegare* are lexicalized verbs. However, many recent examples based on the same pattern are to be found in a dictionary of neologisms [Qua87]. A made-up example to which I shall return later is

(51) iscrivere 'register, matriculate' $\longrightarrow$ disiscrivere 'ex-matriculate'.

This verb is transparently derived and its sense is transparently available to speakers of Italian, but it is not listed in the comprehensive dictionary [PF92] nor in [Qua87]. [7]

Another interesting neologism discussed in Mayo et al. [MSSZ95] is

(52) *sbobinare* 'motion away from spool'

which has been used in the rather opaque meaning of 'transcribe a dictation'. On the basis of a productive word-formation rule

(53) verb $\longrightarrow$ *s-* noun

speakers can also derive for *sbobinare* the sense 'de-spool' from its base *bobina* 'spool' by varying a pattern evident in verbs like *spolpare* 'de-pulp, take the pulp from (a fruit)', which derives transparently from *polpa* 'pulp of a fruit' [Sch95a, 561]. The derivation of the sense 'transcribe' for *sbobinare* is only available to speakers who can remember dictation machines that had spools of wire or magnetic tape and the stenographers whose job it was to 'pull the text from the spool'; for others the meaning must be fetched directly, without morphological analysis, from the lexicon. The simultaneous availability of transparent and opaque meanings of a complex word undoubtedly has something to do with the possibility that lexical storage can contain both a whole-word entry for the word and separate representations of its constituent morphemes, but it remains to be seen whether this insight can be given a systematic formal description.

[7] I thank Vieri Samek-Lodovici (personal communication) for this example.

4.4.2 A Notation for Lexical Semantics

To be able to describe the semantic operations involved in derivations, I shall introduce here a rough and ready informal functional notation that can represent meanings of atomic entities, e.g., nouns, and relations, like those signified by verbs, adjectives and prepositions. This notation, adapted from B. Stiebels [Sti96] and following the tradition of Montague grammar [Dow79], allows compact and perspicuous representations of lexical semantics, but it is too simple to allow detailed descriptions of the interface to sentence syntax, and it will be augmented in the next chapter. In passing it should be mentioned that the semantic fragments to be presented here in this notation give some additional fragments of inner-level requirements for a simulation of derivation, although many other necessary components will remain to be described.

An atomic meaning, usually but not necessarily of a word, I shall notate in the form shown in (54) for the noun *novel*.

(54) NOVEL:BOOK

This means that where a word meaning NOVEL appears in a sentence, e.g., *novel* or French *roman*, the sentence meaning is to be constructed using a restriction of the class of BOOKs conceptually identified as novels.[8]

Meanings can also include variables to specify defining PARAMETERS that can vary in certain ways. To indicate that a novel must have some — but not always the same — author, the semantic description inherited from BOOK can be elaborated to include the parameter *a*, as in (55). Semantic parameters are meant only to represent defining features of the lexical items; other attributes that arise in context, like the temperature or location of a book, are not part of its lexical semantics. If the parameter can be specified by a syntactic constituent belonging to the immediate syntactic environment of the word, its variable appears in a list of lambda-bound ARGUMENTS that precede the semantic predicate. A lambda-bound argument can be assigned a value by the syntactic analysis; a free parameter variable not expressed in the lamda-bound list, however, corresponds to no syntactic constituent, and it can be bound, if at all, only by processes that interpret the semantics of the sentence in which the word is used. The lambda-bound list thus represents the interface from semantics to syntax.

(55) λa NOVEL(a:AUTHOR, ...):BOOK

Lambda-bound arguments can be STRICTLY SUBCATEGORIZED or they can be OPTIONAL. A strictly subcategorized argument is designated by an upper case letter, and it must be present in the syntactic context for grammaticality. An optional argument is designated by a lower-case letter. The optional argument is not required for grammaticality, but if an expression belonging to the type of the variable is present in the immediate syntactic context, it binds the corresponding semantic feature. A strictly subcategorized parameter (upper case variable) *must* also appear in the lambda-bound argument list; but an optional parameter (lower case) may or may not also appear as a lambda-bound argument.

[8]The operator : can be thought of as class inheritance, as in programming languages like C++ and Smalltalk. Object instantiation, however, takes place in the KLU model at the interface from syntax to semantics, at the point where discourse objects are constructed.

Like concepts, variables can also be thought of as inheriting the type attributes of a class. In example (55), assigning the parameter *a* the type AUTHOR has two consequences. One is that only a certain class of intentional beings, like persons, can bind this parameter in the semantic description. The other is that the argument *a* may be marked syntactically, by case or preposition markers or by syntactic position, so that when several arguments are present as constituents in the sentence, the syntactic processor can assign each to the correct parameters in the lexical semantics. This pairing of semantic arguments with specific constituents, called linking or mapping, is highly language-specific. It is not expressed explicitly here, and it will be given its notational form in the next chapter. Semanticists have paid most attention to the ways in which the arguments of verbs are linked to syntactic constituents, but it is evident that nouns, too, have parameters that can be attached to specifically marked syntactic constituents. In the case of NOVEL, the parameter AUTHOR is specified in syntax by a noun of the required type, marked with the preposition *by* as in (56 a). The parameter *a*, specifying the author, can be bound syntactically in other ways, e. g.,, in a clause like *Jane Austen wrote the novel*, and it can disappear altogether in the course of deriving a related word. Other possible attributes of a novel, like its location, can be expressed in the semantics of a sentence in which *novel* is used, but these attributes do not restrict or 'parameterize' the kind of novel. This leads to the semantic specifications of NOVEL and NOVELISTIC shown in (56).

(56) a. novel by Jane Austen
NOVEL(JANE AUSTEN:AUTHOR, ...):BOOK
b. novel by the window
NOVEL(a:AUTHOR, ...):BOOK
c. novelistic
λa NOVELISTIC(LIKE-NOVEL(a:TEXT))

In (56 a), *Jane Austen* can bind the variable for AUTHOR, since *Jane Austen* matches the type of AUTHOR, and AUTHOR is one of the defining attributes of novel. Because the place a novel occupies is not one of its defining attributes, there is no internal attribute of NOVEL that the prepositional phrase *by the window* in (56 b) could bind, and the variable *a* is not assigned a value. (The phrasal semantics for *novel by the window* will, of course, include the relation BY(NOVEL, WINDOW):LOCATION). Finally, the derived adjective *novelistic* is shown in (56 c) to have no internal attribute AUTHOR; somehow this attribute and its related argument must disappear in the course of the derivation from *novel*. There is, of course, much more that can be said about the implicit argument structures of lexical concepts, especially nouns, as Pustejovsky [Pus95] has admirably shown, but for the present purposes descriptions of this sort will suffice.

Unlike nouns, verbs usually require that at least some of their semantic parameters be explicitly specified in the syntactic context, and these upper-case parameters appear also as arguments in the lambda-bound list. In (57), the semantics of *to hear* is described as a perception relation of type EVENT involving two participants which must both be named in the syntactic environment. HEAR could be described as a semantic primitive, but to illustrate a further notational possibility, it is shown as a three-place predicate PERCEIVE that is differentiated by the sense organ used. Thus, its parameters are a participant of type EXPERIENCER, another participant of type THEME, and a

third of type MEANS. Only the first two are expressed syntactically, with lamba-bound variables. The third parameter is internally bound because it is only possible to hear with one's ears.

(57) Hear:
λT λA PERCEIVE(A:EXPERIENCER, T:THEME, EARS:MEANS):EVENT

By contrast, Italian *sentire* is not restricted to perceiving with the ears, and it would be represented by a three-argument function as in (58)

(58) Sentire:
λm λT λA PERCEIVE(A:EXPERIENCER, T:THEME, m:MEANS):EVENT

The sense used to perceive something can be specified by an adjunct, as in *sento il gatto morbido al tatto* 'I feel the soft cat by touch', in which case the prepositional phrase binds the lower-case, non-obligatory argument *m* to specify what sort of perception is meant. Otherwise, the means must be inferred from the context, as in *sento il treno* 'I hear the train' and in *sento l'odore* 'I smell the odor', but it is not obligatory. [9] The EXPERIENCER argument is obligatory, but in Italian the verb inflection alone is often sufficient to specify it.

In the lambda-calculus, lambda-bound variables of an expression bind to, or 'incorporate' terms following the expression, and the left-most variables bind first. Semantic descriptions are typically constructed by placing the least prominent parameters (those most deeply nested in the semantic formula) first (left-most) in the argument list, so that they will bind first to constituents in the immediate syntactic structure. Because it binds last, the right-most argument can be considered the 'external' argument of the semantic structure; it binds to the syntactically farthest removed constituent. In the case of an active verb this is normally the subject; it is typically preceded in the argument list by the object and other adjuncts. These assignments are not entirely consistent or predictable, however, so that it will be necessary in the expanded notation of the next chapter to add mapping relations that explicitly assign the semantic variables to specific sentence constituents via grammatical functions. The mapping relations take advantage of the notion of grammatical function to ensure that the correct syntactic constituent binds each semantic variable.

Many semantic notations provide a further variable for a discourse or situation referent, as has been worked out in detail in the Discourse Representation Theory of H. Kamp and U. Ryle [KR93], but in the notation I am defining here, the lambda-bound variables at the beginning of the semantic formula are used *only* to express connections between the internal conceptual structure and syntactic arguments. Instantiation variables are assumed to be introduced separately, at the interface from syntax to semantics.

Semantic formulas can be combined via the set-theoretic operations of intersection and union, as in (59 b) and (60 b), meaning that a single predicate defined by the joined predicates is applied to the common argument *A*, as in (59 c) and (60 c).

(59) a. large and blue
b. λA LARGE(A) & λA BLUE(A)
c. λA [LARGE(A) & BLUE(A)]

[9] I thank Chiara Frigeni for these examples (personal communciaton).

(60) a. blue or red
b. λA BLUE(A) ∨ λA RED(A)
c. λA [BLUE(A) ∨ RED(A)]

Further, a formula can take another formula as a functional argument, as in (61 b), which simplifies to (61 c). A functional application is evaluated by consuming as many arguments as appear to the right of the formula, starting with the left-most variable. Lambda-bound variables in the substituted term are removed and inserted at the left-hand end of the argument list of the resulting formula, as happens with the variable *T* in (61).

(61) a. cause to die
b. λX λA CAUSE(A,X) (λT DIE(T))
c. λT λA CAUSE(A, DIE(T))

Functional application can also be used to express modification of a concept named by, e.g., a noun

(62) a. long novel
b. λA LONG(A) (NOVEL:BOOK)
c. LONG(NOVEL:BOOK)

A final notation, again adapted from Stiebels [Sti96], will be used to express the derivation of one semantic formula from another, e.g., by applying a semantic operation associated with the derivation to the semantics of the base. I use angled brackets to denote "the semantics of", and square brackets to indicate the derivation function. Thus, $[\ <\text{N}>]_V$ signifies the derivation of the semantic formula for a V(erb) (*not* the verb itself) from the semantics of a N(oun).

With this notational apparatus in hand, I turn to some attempts to give formal specifications of the operations needed to account for the semantics of derivatives. An interesting property of Stiebels' notation is that it makes no inherent distinction between the semantics of sentence elements and word elements. At the purely formal level it does not require Lexical Integrity, and using it thus presents another opportunity to test whether word meanings cannot be constructed directly in the same system used to construct sentence meaning.

4.4.3 Stiebels' Analysis of German Preverb Incorporation

In the linguistics literature a number of accounts of the semantics of word derivation have been presented that use some form of functional composition to describe how the semantics of a base and the derivational affix combine, e.g., R. Jackendoff [Jac90], J. Kunze [KuMvBF93], and Stiebels [Sti96]. These studies have made it evident that the mechanisms involved in derivational semantics can be nearly as complex as those that must be postulated for sentence semantics, and in many cases the boundaries between the two are not clear, as with so-called secondary predication in English and with particle verbs in German. In a study of particle verbs in German, Stiebels [Sti96] argues, for example, that verbs like *aufsetzen* 'put on' simply incorporate a preposition and its arguments into the semantic formula of the verb. With an explicit argument, as

in (63 a), a preposition and its object form a complement to the verb, but if the object is omitted, as in (63 b), the object of the preposition becomes an implicit internal parameter of the verb semantics, and a syntactic entity becomes available that can behave distributionally like a single verb, as shown by (63 c and c), where *auf* 'on' and *setzen* 'to place' together occupy the position of the infinitive in a future construction, and *auf* cannot be separated from its base.

(63) a. Max setzt den Hut auf den Kopf. 'Max puts the hat on (his) head.'
b. Max setzt den Hut auf. 'Max puts the hat on.'
c. Max wird den Hut aufsetzen. 'Max will put the hat on.'
d. *Max wird den Hut auf langsam setzen.
'Max will put the hat on slowly.'

For verbs like *aufsetzen* Stiebels proposes that the semantic structure of *setzen* provides an optional argument *q* for a locative modifier, as shown in (64 a).[10] In sentences like (63 a), the argument *q* absorbs the locative modifier *auf den Kopf* as an optional syntactic argument. When the preposition is present without its required object, as in (63 b), the object-less preposition is incorporated by a morphological operation into the position provided for the modifier, giving the semantic structure (64 c). Unlike the preposition, the verb no longer requires a localizing object argument, but if such an argument is present, it can be incorporated into the semantic structure via the variable *l* of (64 c) [Sti6b, 15]. It is worth noting that the type of the incorporated preposition must match that of the unspecified modifier *q* in *setzen*. This prevents the formation of an ungrammatical particle verb from a non-locative preposition like *wegen* 'on account of', i.e., **wegensetzen* 'place on account of'.

(64) a. *setzen*: λq λU λX [TO-PLACE(X, U) & q:LOC]
b. *auf*: λX λL ON(X,L):LOC
c. *aufsetzen*: λl λU λX [TO-PLACE(X, U) & ON(X,l):LOC]

The semantic formula (64 c) follows straightforwardly from the assumed morphological structure in which *setzen* is the head and *auf* a modifier; it is only necessary to apply the head semantics (64 a) to the semantics of the preposition (64 b) to obtain (64 c). The required localizing parameter *L* in *auf* is demoted to an optional argument *l* by its incorporation as an optional modifier.

In many verbs that Stiebels analyzes, the set of possible modifiers to the verb provide no evidence for an implicit variable like *q*. To obtain the additional arguments that appear in the course of the derivation, she must postulate an intermediate, 'invisible' semantic function NEWARG, which must be applied to the base before the particle can be incorporated [Sti96, 108].[11] Thus, in German it is possible to derive a verb *abwandern* 'use up by hiking' from the verb *wandern* 'hike', as in (65).

(65) Max wird seinen Mitgliedsbeitrag für den Alpenverein abwandern.
'Max will hike off his dues to the Alpine Association.'

[10] Stiebels' notation does not distinguish between syntactically optional and required arguments, and she includes a "situation" argument (s) which I am omitting. I have added type designations to the variables where they seem important.

[11] Stiebels actually names this function ARG.

However, *wandern* does not include the consumption of something as part of its defining semantic structure, i.e., it is not possible to say *das ganze Geld wandern* 'hike all of the money'. Therefore, *ab* in the meaning of 'decrease' cannot fill an existing position in the semantic structure of *wandern*. To carry out the derivation, Stiebels applies the function NEWARG to *wandern*. This function introduces an additional position Q into the list of lambda-bound or "projected" arguments and conjoins it with the semantic structure of the verb:

(66) NEWARG(<Verb>):
<Verb-Semantics> $\longrightarrow$ λQ ... [<Verb-Semantics> & Q]

The derivation of *abwandern* then proceeds as follows: Since the type of the semantics of *ab* does not match the available modifier position in the base *wandern* (which I have given the type PATH), the function NEWARG must be invoked to introduce a new position for *ab*. The resulting formula can then be applied to the semantics of *ab* to obtain the semantics of *abwandern*, as shown in (67a-d). Again, I use <x> to mean "semantics of x".

(67) a. *ab*: DECREMENT(X): CHANGE-QUANTITY
b. *wandern*: λp λX [HIKE(X) & p:PATH]
c. Since CHANGE-QUANTITY $\neq$ PATH, apply
NEWARG(<*wandern*>) $\longrightarrow$ λQ λp λX [[HIKE(X) & p:PATH] & Q]
d. Apply <*wandern*+*Q*>(<*ab*>) $\longrightarrow$
λU λp λX [HIKE(X) & p:PATH &DECREMENT(U)]

This allows the semantics of (65) to be constructed as in (68).

(68) [HIKE(MAX) & p:PATH &DECREMENT(DUES)] [Sti96, 143]

Somewhat more complex are verbs derived from adjectives and nouns. Because in these patterns the verb usually has an argument structure that cannot be extracted from its base, Stiebels postulates verbal templates in which the noun can occupy the lowest position (farthest to the left in the argument list). Recall that the conversion of a noun's semantics to those of a verb is written $[< N >]_V$. These templates include patterns for stative (69 a), inchoative (69 b) and causative (69 c), (69 d) verbs, which like the NEWARG function for deverbal derivations, introduce needed argument positions, as well as additional semantic predicates. The bases can be nouns and adjectives.

(69) a. [<*glow*>]$_V$ $\longrightarrow$ λX POSSESS(GLOW) (stative, e. g., *the coals glow*)
b. [<*dry*>]$_V$ $\longrightarrow$ λX BECOME(DRY) (inchoative, e. g., *to dry the socks*)
c. [<*bundle*>]$_V$ $\longrightarrow$
λY λX CAUSE(X, BECOME(BUNDLE(Y)))
(causative, e. g., *to bundle the papers*)
d. [<*drape*>]$_V$ $\longrightarrow$
λY λX CAUSE(X, BECOME(LOC(R_{prox}(Y, DRAPE))))
(causative, e. g., *to drape the balcony*))

The derivation of a particle verb from a noun must therefore proceed in two stages. In the first, the noun or adjective is converted to a verb; in the second, the particle is incorporated by applying the intermediate verb semantics to the particle, as in (67). In (70) the derivation of *einrahmen* 'to (place in a) frame' from the noun *Rahmen* 'frame' proceeds as follows: The denominal conversion rule (70 c) creates a verb with a causative semantic structure (70 d) meaning 'place in the region R_{prox} of a frame', on the pattern of (69 d). The semantics of this intermediate verb *rahmen*, which Stiebels gives as 'become proximate to a frame' is still unspecific, owing to the vagueness of R_{prox}. To restrict the spatial relation to IN, the derivation has incorporated the localizing preposition *ein*. However, *rahmen* has no modifier position available, since the conversion step has already introduced a vague proximal spatial relation. The function NEWARG must therefore be applied to create an additional, conjoined argument position, and the semantics of *ein* is incorporated via this argument.

(70) a. *in/ein*: λV λU IN(V, U:LOC-OBJECT): LOCATION
b. *Rahmen*: FRAME:LOC-OBJECT
c. Select a conversion template to derive a localizing verb, as in (69 d)
$[< N >]_V$:
λZ λY λX CAUSE(X, BECOME(LOC(Y, R_{prox}(Z:LOC-OBJECT))))
d. Apply $[< N >]_V$(<*Rahmen*>) ⟶
λY λX CAUSE(X, BECOME(LOC(Y, R_{prox}(FRAME))))
e. Since no modifier position is now available to absorb *ein*, apply
NEWARG(<*rahmen*>) ⟶
λQ λY λX [CAUSE(X, BECOME(LOC(Y, R_{prox}(FRAME)))) & Q]
f. Apply <*rahmen*+*Q*>(<*ein*>) ⟶
λY λX [CAUSE(X, BECOME(LOC(Y, R_{prox}(FRAME))))
& IN(Y, IN(Y, FRAME))][12]

Alternative analyses of *einrahmen* are possible (Stiebels presents two), but a question that necessarily presents itself is how a given derivation chooses among the available alternatives. For example, at step (70 d) Stiebels derives an intermediate verb meaning 'make proximate to a frame', but in principle any of the patterns described by (69) could apply as well, giving intermediate verbs meaning 'possess a frame', 'turn into a frame' or 'cause to become a frame'. Moreover, it is not always the case, as Stiebels claims, that the lowest argument in the semantic formula (70 c) binds the base noun or adjective. For example, one of her own examples, *einölen*, more commonly means 'place oil in something' rather than 'place something in oil'; in other words, *öl* is substituted into the theme argument, and the lowest argument position, belonging to *ein*, is used to bind a syntactic argument that tells where the oil goes. Evidently there is some other factor at work in the derivation of words like *einölen* and *einrahmen*, something that forces *to oil* to mean putting oil into something and that makes 'being *in* a frame', rather than '*being* a frame', the result of doing something with a frame. This factor Stiebels attributes to encyclopaedic knowledge, but just how and where in the derivation encyclopaedic knowledge comes into play is not specified.

[12] Stiebels actually substitutes BECOME(LOC(Y, IN(FRAME))) for *Q* in this step, but I do not see the source of the additional BECOME.

Stiebels sees the base of the particle verb as the head of the derived form, since the derivative arises by functionally applying the base semantics to the semantics of the incorporated preposition or prefix. She further justifies this construction by noting that the derived verb retains other morphological attributes of the base, like inflectional class (in German this is, however, often not true for nouns, e.g., the nominalizing suffix *-ung* determines the gender and the plural inflection of the derivative). Thus her word formation rules behave very much like the rules of verb phrase syntax. In both cases, it is the verb which gathers its arguments from the immediate syntactic context. For particle verbs that incorporate a morphologically free preposition, like *aufsetzen*, the derivation does not require a special description of *auf* as a derivational particle. The unaltered *auf* is incorporated, and its required location parameter becomes an optional argument of the verb. This, too, allows a description of word formation that remains close to verb phrase syntax, and it avoids extending the lexicon with additional entries for prepositions in a form used only for word formation.

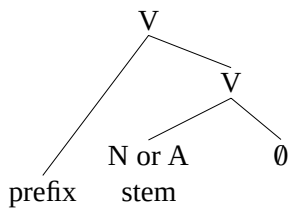
Nevertheless, Stiebels insists on a lexical interpretation of her analyses, mainly because a great many particle verbs have readings that are either semantically displaced from the compositional semantics, or are completely opaque [Sti96, chapt. 10]. For example, compositionally the verb *anlegen* means 'to place on or at', but it also has the readings 'aim at' (*das Gewehr auf den Gefangenen anlegen* 'aim the rifle at the prisoner') and 'look for trouble (with someone)'. On the other hand, verbs like *abwandern* in the reading described in (67 d) are probably nonce formations; in fact, this reading of *abwandern* is not listed in the *Duden Universalwörterbuch* [Dro83], and the lexicalized meanings are closest to 'depart'. One has to ask, therefore, in what sense *abwandern* in the reading of 'hike away' can be said to be "in the lexicon", if it is unlikely to be known beforehand to the speech community.

Assigning the formation of all word forms to the lexicon, Stiebels lands in the same dilemma that was described in chapter 2 in connection with corpus statistics. If all complex words are assigned to the lexicon, then we are forced to accept a bizarre conception of the lexicon as a limitless store of words, a great many of which have never been seen or heard, because as was shown in chapter 2, the set of all complex words can only be found in a limitless corpus, and the vocabulary of a limitless corpus is also limitless. In a model of language processing that postulates an infinite but static inventory of words, a specific account of word formation is descriptively interesting, but it is not necessary in the way that syntax is necessary to describe how never-before encountered utterances can be generated and understood, because it is assumed that all words, however rare, are always available. This dilemma is related to a problem raised by the Phrasal Recursion Impoverishment (p. 118). If phrasal recursion is avoided by allowing all phrasal constituents to be "in the lexicon", it is necessary to assume a limitless static store of phrases made available for insertion at word formation. But a limitless lexicon is simply *not* a lexicon in any sense that can be given a meaningful cognitive interpretation, for however great the storage capacity of the brain or of a community of brains, it is not infinite, not even for speakers of morphologically creative languages like Turkish. The alternative, allowing productive, rule-based production of words within the lexicon module, dodges the question of why and how word-forming rules should be separated from those being used to construct sentences, and in fact Stiebels' notation does not draw this distinction.

4.4.4 Lieber & Baayen's Analyses of Dutch Prefixed Derivations

A somewhat similar attempt to describe the semantics of derived verbs by functional composition has been proposed by Lieber and Baayen in [LB93]. Unlike Stiebels, they assume a basic morphological structure that describes the prefix, rather than the base, as a kind of head, as in (71).

(71)



For category-changing, multi-step derivations like (70), this approach has some decided advantages. Seeing the prefix as the head of the morphological structure allows it to select the correct derivation by checking its subcategorization restrictions on the category and semantics of the base. By contrast, in Stiebels' analyses the burden of sorting out the correct derivational tree falls to the step in which the prefix or particle is incorporated. Since the particle has no special lexical entry as a derivational morpheme, it cannot steer the course of the derivation. When incorporation of the particle is preceded by a possibly ambiguous conversion, e.g., from a noun to a verb, the derivation can have a number of possible intermediate heads, including those listed in (69). If the incorporation of the particle, too, has possibly ambiguous results (as with *einölen*), the ambiguities multiply, and additional apparatus will be necessary to sort out the correct results.

Lieber & Baayen do not go into much detail about how morphological structure maps to the resulting semantic formula, but as chapter 5 will show, the mapping poses no intrinsic formal problems. For the Dutch prefix *ver-* (which is semantically similar to *ein-* in (70)) they present the semantic formula (72).[13]

(72) a. $[<ver-_i>]_V \longrightarrow \lambda U\, \lambda X\, \textsc{Go}(X, \textsc{From}(O, \textsc{To}(U)))$
b. $[<ver-_c>]_V \longrightarrow$
$\lambda U\, \lambda X\, \textsc{Cause}(X, \textsc{Go}(X, \textsc{From}(O, \textsc{To}(U))))$

Many verbs derived from adjectives with *ver-* have both inchoative and causative readings, as in the case of *verarmen* 'become poor' or 'make poor'. Taking account of this, the inchoative semantics is generated by (72 a), and (72 b) adds an additional predicate to obtain the CAUSE relation. Denominal verbs can be derived by substitution into these same formulas, although most tend to be causatives, like *verpakken* 'to package' (73 a). However, inchoatives are also found, like *verharen* 'to shed hair' (73 b) or *verijzen* 'to ice up' (73 c).

(73) a. $[<verpakken>]_V \longrightarrow$
$\lambda U\, \lambda X\, \textsc{Cause}(X, \textsc{Go}(U, \textsc{From}(O, \textsc{To}(\textsc{Pack:container}))))$
b. $[<verharen>]_V \longrightarrow \lambda X\, \textsc{Go}(\textsc{Hair}, \textsc{From}(X, \textsc{To}(u)))$
c. $[<verijzen>]_V \longrightarrow \lambda X\, \textsc{Go}(X, \textsc{From}(O, \textsc{To}(\textsc{Ice})))$

[13] Lieber & Baayen use the semantic primitives and notation of Jackendoff [Jac90]; however, in the interest of consistency, I have expressed their semantic formulas in the notation introduced in section 4.4.2, adding lambda abstractions where appropriate.

Lieber & Baayen do not specify which variables are expressed syntactically, and unlike Stiebels they do not try predict where the prefix's argument is substituted in the formula. In fact, it is difficult to make these predictions, as was witnessed above by the verb *einölen* and here by the examples in (73). While with *verharen* the highest argument (subject) represents the origin of the motion (*de hond verhart* 'the dog sheds hair'), with *verijzen* the same argument is the goal of the change.

Most interesting are pairs like *kopen* 'buy' – *verkopen* 'sell', which seem to indicate that *ver-* can also be used to reverse a complex action. In fact, like German, Dutch also has other pairs like *huren* 'rent, hire' – *verhuren* 'let' with the same contrast, suggesting that a systematic derivational pattern is involved, although it is not actively productive. Lieber & Baayen try to give a compositional account of this pattern by describing *kopen* as a kind of inchoative exchange, using a description due to Jackendoff (that Jackendoff [Jac90, 61] ultimately rejects). Drawing on the pattern found for the derivation of causatives from inchoatives (such as *verjagen* 'to chase away' from *jagen* 'to hunt for'), they propose to derive *verkopen* by making *kopen* the argument of a CAUSE predicate, as in (74).

(74) a. $[<\textit{kopen}>]_V \longrightarrow$
$\lambda m\, \lambda z\, \lambda Y\, \lambda X$ [GO(Y, FROM(z, TO(X)))
& EXCH(GO(m, FROM(X, TO(Y))))]

b. $[<\textit{verkopen}>]_V \longrightarrow$
$\lambda m\, \lambda z\, \lambda Y\, \lambda X$ CAUSE(X,[GO(Y, FROM(X, TO(Z)))
& EXCH(GO(m, FROM(Y, TO(X))))])

Here *X* is the buyer, *Y* is the thing bought, *m* is the money (optional syntactically but not semantically), and EXCH indicates that the second GO is reciprocal with the first. At first glance this solution seems ingenious, but an inchoative analysis of *to buy* is hard to sustain. Inchoatives are generally logically compatible with assertions like *I don't know what caused it*, e.g. *the tree fell; I don't know what caused it (to fall).* With causatives this is not the case, e.g., ??*the lumberjack felled the tree, but I don't know what caused it (to fall).* Similarly absurd is ??*Jack bought a poodle but I don't know what has caused him now to have a poodle*. Basing the derivation on an inchoative *kopen* is thus implausible. A second serious problem is that the internal roles must be remapped to the syntactic arguments, something that they do not explain at all.

The reversative derivation is thus a challenge to any attempt to account for derivational semantics in a manner parallel to that of sentence semantics. For German and Dutch (as Baayen & Lieber admit) this pattern is unproductive and could be dismissed as being similar at the word level to idioms in syntax, which also cannot be accounted for compositionally. If this were the case for all difficult derivations, one might be able to rescue a syntax-like account of word meaning. But, as we shall see in the next section, Italian provides a similar reversative pattern that can be used without restriction to produce nonce derivatives. In other words, word meanings that are difficult to construct from the systematic composition of prefix and base semantics are not always the result of loss of transparency or other lexicalization processes. In Italian such words can be produced spontaneously, and an account of word formation must be able to describe the semantic operations involved in such derivations.

4.5 Italian Derivatives in *s-*, *dis-* and *ata-*

Even though words, like sentences, can be produced spontaneously and with meanings that are transparently evident from their constituents, there is evidence that the system responsible for their construction functions differently from that of syntax. We have already seen evidence that word formation is much less flexible than syntax in many respects (Impoverishment of Morphotactics). Here I shall draw again on data reported in [MSSZ95] to show that, relative to sentence grammar, word formation is much more inclined to rely on remembered patterns than on reconstruction of the compositional sense, that derivations can involve intermediate, 'virtual' word formations, and the operations used for constructing word meaning are much more closely intertwined with conceptual evaluation.

4.5.1 Italian *s-* and *dis-* as Negation

The Italian prefixes *s-* and *dis-* have been widely studied in the morphological literature. Following Schwarze [Sch95a, 553] I assume that *s-* and *dis-* are lexically the same, but that *dis-* is phonologically restricted, and I shall therefore write *s-* to represent both prefixes. A survey of the literature by M-Th. Schepping, reported in Mayo et al. [MSSZ95], resulted in a rather elaborate taxonomy of possible semantic contributions of *s-*, shown in Table 4.1. Needless to say, the amount of polysemy shown in this Table — more than 11 readings — seems suspect, given that even frequently used words seldom have so many distinct readings. I shall not examine all of these patterns here, but rather try to find to what extent meanings from the presumably more productive patterns can be constructed in ways similar to those observed in syntax, or to what extent it will be necessary to postulate a separate, word-specific grammar.

Table 4.1: Semantics of the Prefixes *s-* and *dis-*, from [MSSZ95]

	CATEGORY MAPPING	SEMANTICS
	various	various
1	Adj → Adj	OPPOSITE-OF
2	Verb → Verb	INTENSE
3	Noun → Noun	NEGATIVE-COUNTERPART
4	Verb → Verb	NOT
5	Verb → Verb	NEGATIVE-VALUATION
6	Noun → Noun	NEGATIVE-VALUATION
7	Verb → Verb	CONTRARY-ACTION
8	Noun → Adj	LACKING
9	Adj → Verb	DIMINISH
10	Noun → Verb	REMOVAL-WITH

4.5.2 Qualification of the Base in *s-*

Mayo et al. [MSSZ95] classified the derivations in Table 4.1 in order of increasing suspected complexity, on the basis of semantic operations that were assumed necessary to derive the meanings of the derivatives from their bases. At the low end of the scale of complexity they placed derivations whose meanings can be reconstructed as equivalent noun or verb phrases. Examples, most of which are of recent origin, are

(75) a. informazione 'information' $\longrightarrow$
disinformazione 'lack of information' (Type 3)
b. adattamento 'adaptation' $\longrightarrow$ disadattamento 'misadaptation' (3)
c. contentare 'satisfy' $\longrightarrow$ scontentare 'dissatisfy' (4)
d. leggere 'read' $\longrightarrow$ disleggere 'read incorrectly' (5)
e. ragionare 'think' $\longrightarrow$ sragionare 'think illogically' (5)
f. manager 'manager' $\longrightarrow$ smanager 'terrible, unsuccesful manager' (6)

For each of the derivations in (75) except (75 c), a very nearly equivalent meaning could be produced by constructing a phrase containing modifiers meaning 'bad' or 'poorly', and the semantics of such phrases would be easily expressed as BAD(INFORMATION), POOR(ADAPTATION), etc. Although in the table the derivations are classified further by grammatical categories and specific predicates, the specific effect of the modifier in each case depends to a large extent on conceptual properties of the base. As E. Lang has shown [Lan87], the semantic effect of an adjective depends essentially on the meaning of the word modified. A *wire* can be *thin* but not *narrow*, for example, but a *street* must be narrow; if it is thin, it can be so only in respect to the thickness of the paving material, a dimension perpendicular to the dimension narrow–wide. Facts like these have motivated Pustejovsky [Pus95] to include in a lexical entry detailed information about the dimensions of possible change that a concept can undergo. In the notation used here, each of the dimensional parameters of modification can be expressed as a distinct modifier variable of the semantic formula that steers the conceptual modification, as with the variable *q* in (64), used to specify a direction for the motion component of *setzen* 'to set'. An evaluative adjective like *bad* or *poorly* therefore does not simply attach an attribute to the semantics of the base but rather modifies the underlying concept along an appropriate, concept-specific dimension. On the assumption that the good-bad dimension of *informazione* ranges over a scale specifying quantity of data, the semantics of BAD(INFORMATION) in (75 a) will therefore evaluate to 'lack of information'. The other derivations in (75) can undoubtedly be accounted for in the same way. For example, *ragionare*, in contrast to *pensare* 'think', *riflettere* 'reflect' and *imaginare* 'imagine', emphasizes the logical aspect of thinking, so that POORLY(RAGIONARE) has the specific meaning of 'think illogically', rather than 'vacuously' or 'uncreatively'.

If the prefix *s-* is given a lexical meaning 'poor(ly)' then, following Stiebels [Sti96], for each of the examples in (75) the derivation should give a result semantically similar to the phrasal semantics formed with an adjective or adverb meaning 'bad' or 'poorly'. This approach would simplify the polysemy of Table 4.1 and possibly permit conflating the semantics of NEGATIVE-COUNTERPART and NEGATIVE-VALUATION; OPPOSITE would probably still need to be specified separately because it requires access to relationships between separate concepts rather than to dimensions of variation

within a concept.[14] The incorporation approach of Stiebels [Sti96] would also lend weight to the view that semantics in word formation arises in ways very similar to those of syntax.

However, other, more complex derivational patterns found for *s-* tend in the other direction. It is difficult to give purely compositional accounts similar to those of adjective or adverbial phrases for the derivational semantics of CONTRARY-ACTION and of REMOVAL-WITH (Types 7 and 10 in Table 4.1).

4.5.3 Italian Reversative and 'Removal' Verbs in *s-*

For verb-to-verb derivations of Type 7, CONTRARY-ACTION, Mayo et al. list several examples, all more recent than 1963 and therefore likely to be systematic representatives of an active word formation pattern.

(76) a. allineare 'to align' $\longrightarrow$ disallineare 'to break an allignment'
b. bardare 'to saddle e.g., a horse' $\longrightarrow$
sbardare 'to remove saddle or harness from e.g., a horse'
c. congelare 'to freeze' $\longrightarrow$ scongelare 'to thaw'
d. convocare 'convene' $\longrightarrow$ sconvocare 'dissolve (an assembly)'
e. inibire 'to inhibit' $\longrightarrow$ disinibire 'to remove an inhibition'
f. inquinare 'to pollute' $\longrightarrow$ disinquinare 'to reverse pollution'

By comparison with Types 1 through 6, it is much more difficult to give syntactic paraphrases of these derivatives without invoking other derivations, e.g., 'reverse of saddling a horse'. Since for all of the examples in (76) the semantics of the base can be expressed in terms of a transition to a stative relation, such as 'in alignment' (*allineare*) or ON(SADDLE, HORSE) (*bardare*), the semantics of the bases can be given as BECOME(X, <REL>(X,U)). This invites a description of the corresponding derivatives as BECOME(X, NOT(<REL>(X,U))), an operation that can be described by substitution operations on semantic formulas, along the lines proposed by Stiebles [Sti96]. This formula nevertheless neglects aspects of the semantics that would be needed to map the semantic arguments to syntactic constituents. With verbs denoting a transition to a stative relationship reached by crossing a regional boundary, as in (77 a), an argument of type GOAL is realized with a localizing preposition *su* 'on', but in (77 b) the corresponding argument is of type SOURCE and must be realized with a preposition like *da* 'from', specifying a movement away from the localizing object (cf. (49), (50)).

[14] For the semantic negation of adjectives (Type 1) Mayo et al. reject the contention that the semantics obtained with the corresponding syntactic paraphrase is equivalent. For *una poltrona non comoda*, 'a not-comfortable armchair' they maintain that only the lack of comfortable qualities is implied, whereas for *una poltrona scomoda*, 'an uncomfortable armchair' "someone with knowledge of armchairs will say ... not merely that it lacks softness ... but that some of a number of specific, identifiable attributes are implied: lumpy cushions, a shape that induces aches, perhaps upholstery that has too much softness in particular places" [MSSZ95, 897-8]. This view requires a non-compositional representation of a complex word's semantics, but the cross-modal priming experiments of [MWTWO94], reviewed in section 3.6.3, appear to suggest the opposite, namely that the semantic representation is stored as a complex formula like NOT(COMFORTABLE(X)).

(77) a. L'operaio carica la merce sull' autocarro.
'The worker loads the goods onto the truck.'
b. L'operaio scarica la merce dall'autocarro.
'The worker unloads the goods from the truck.'

On the other hand, *sconvocare* 'dissolve (an assembly)' also involves a BECOME transition to a two-place stative relation, and it could probably be represented as

(78) BECOME(ASSEMBLY, NOT(TOGETHER(*member*$_i$,*member*$_j$)))
for all $i \neq j$

but like its base *convocare*, this verb's implicit arguments *member*$_i$ and *member*$_j$ do not have any expression as optional syntactic arguments, a fact that cannot be derived directly from the semantic formula, but which can be inherited from the base. In general it appears that the argument structure for deverbal reversative verbs in *s-* can be determined from the argument structure and mapping relations of the base, in combination with the semantic formula of the derivative and a limited amount of conceptual information about the internal stative relation.

For deadjectival and denominal verbs formed on the pattern of REMOVAL-WITH, more is required. A number of examples of mainly recent origin, again from [MSSZ95]:

(79) a. fitto 'dense, close' $\longrightarrow$ sfittire 'to thin out'
b. ambigo 'ambiguous' $\longrightarrow$ disambiguare 'to disambiguate'
c. barda 'saddle' $\longrightarrow$
sbardare 'to remove saddle or harness from e.g., a horse'
d. brina 'hoarfrost' $\longrightarrow$ sbrinare 'to defrost, de-ice'
e. dente 'tooth' $\longrightarrow$ sdentare 'to dull (e.g., a saw)'
f. forma 'form' $\longrightarrow$ sformare 'deform'

These derivations are based on a pattern in which the base is typically a quality or an object that is in some way removed, and the direct object designates the object or environment from which it is removed. Thus, *sfittire il bosco* 'to thin out the woods' arises from 'removing the denseness', and *sbardare il cavallo* is to 'remove the saddle from the horse'. The action of 'removal' can be interpreted in terms of related effects, as in *sdentare*, where the teeth of the saw are removed in the sense that they are made ineffective.

The base of the derivation is not necessarily the affected object; it can also be the localizing object or container, as witnessed by (80).

(80) a. barca 'ship' $\longrightarrow$ sbarcare 'to unload from a ship'
b. carta 'paper' $\longrightarrow$ scartare 'unwrap'
c. forma 'form, mold' $\longrightarrow$ sformare 'remove from a mold'

The contrast between the examples (79) and (80) reveals that the REMOVAL-WITH derivation is not controlled entirely by the morphological structure. Which thematic roles are assigned to the base and to the direct object is not uniquely determined; the assignment results rather from the conceptual properties of both, and from knowledge

about actions in which they are likely to be involved. In fact, in some cases the semantics can be ambiguous, e.g., *sbarcare* can mean not only 'unload the ship' but, in an appropriate context, also 'remove the ship(s) from something': *il vento forte sbarca il mare* 'the strong wind clears the sea of ships'. Yet when the base is not a typical container or support of some other removable object, it is not accepted in the localizing role. Thus, **sbardare il cavaliere* formally would permit the reading 'take the rider from the saddle', but Italian speakers resist this interpretation. In other words, their conceptual knowledge about objects like saddles and their uses seems to play a direct role in determining how the semantics of the derivation is constructed. The degree to which the meanings of derivatives depend on specific thematic and conceptual properties of their bases lends support to the idea that derivation, to a much greater extent than syntax, is strongly intertwined with a system of conceptual models. Schwarze has argued that a derivation is not an operation over semantic formulas alone, but that the function producing the derived predicate must be defined over conceptual or "action schemes" that can include narrowly defined roles and actions, and that these are often specific to the derivation [Sch95a, Chapt. 3].

A further complication is that these deadjectival and denominal derivations may well proceed in two steps. On the incorporation model proposed by Stiebels for German *einrahmen* (70), the derivation of *sbarcare* would lead first to a 'virtual' intermediate verb *barcare[15], then to a reversal on the pattern of the deverbal reversative reading of *s-* (76). On the semantic composition model of Lieber & Baayen, it would be necessary to assume that *s-* provides both a semantics for the reversal of telic actions and a more complex template like the one they use to describe the denominal derivation of Dutch *verpakken* from *pak* 'package' (73). In both cases the formal ambiguity of the role assignment multiplies the number of required templates and therefore the polysemy of the affix, but in fact most of these derivatives are regarded as having only a single meaning, determined by the base. It would appear that in many cases, like **sbarda il cavaliere*, the rules of word formation are blocked by conceptual factors.

4.5.4 Italian Nominalizations in *-ata*

Although Mayo et al. presented analyses of denominal and deadjectival verbs along the lines of Lieber & Baayen, there are several good reasons to prefer the derivation in two steps, roughly as described by Stiebels, but with the derivational affix taken as the head of the construction rather than the base. These arguments have emerged from a study of Italian derivations in *-at(a)* carried out by V. Samek-Lodovici, reported in [SL6a] and [SL97], and they include considerations of thematic structure, distribution and lexical statistics.[16]

Italian event nouns derived on a pattern similar to the past participle of a verb reveal a dependency on conceptual structure even more strongly than the REMOVAL-WITH verbs discussed above; they thus furnish the clearest evidence that word formation can involve mechanisms different from the compositional construction of sentence-level semantics. Following examples in Schwarze [Sch95a], nouns derived in *-at(a)* can have the semantics

[15] *Barcare* is blocked by the presence of *imbarcare* 'to embark, load onto a ship'.

[16] Similar arguments were suggested to me by M-Th. Schepping, personal communication. See also [SL98].

(81) a. TYPICAL-EVENT-WITH(B:INSTRUMENT:MOVABLE OBJECT):STRIKE
Example: gomito 'elbow' $\longrightarrow$ gomitata 'a blow with the elbow'
b. TYPICAL-EVENT-WITH(B:AGENT:CHARACTER):PERJORATIVE DISPLAY
buffone 'buffoon' $\longrightarrow$ buffonata 'farce'
c. TYPICAL-EVENT-WITH(B:INSTRUMENT:SENSE):PERCEPTION
occhio 'eye' $\longrightarrow$ occhiata 'a look'
d. TYPICAL-AMOUNT-WITH(B:INSTRUMENT:CONTAINER):MEASURE
cucchiaio 'spoon' $\longrightarrow$ cucchiaiata 'spoonful'

but this list is far from complete. For example, a *spaghettata* is a kind of meal and *telefonata* means typically 'a telephone conversation', but there is not a general semantic category of communicative events formed in *-ata*. Superficially it appears that the compositional semantics attached to the derivation in *-ata* is no more specific than 'an object related to <base>', although the actual derivatives usually have very specific meanings.

To account for the specific semantics of a denominal noun in *-at(a)*, it appears that recourse must be made both to conceptual schemata in which the base is embedded and to a set of stereotypical actions preferred for the derivational pattern. On the basis of counts of both lexicalized nouns and non-lexicalized but acceptable nonce formations in *-at(a)*, Samek-Lodovici found that about 28 percent of all types were formed on the pattern (81 a); 26 percent belonged to pattern (81 b); and 6 percent were on pattern (81 d) [SL6b, 18]. How the conceptual factors interact with the word formation patterns would need to be clarified in a more detailed study; it is striking, however, that the assigned meanings cannot be deduced in a rule-like manner from a semantic formula. While, for example, *coltellata* 'stab with a knife' can be easily understood in terms of an operator that finds the most salient, quick telic action associated with its base *coltello* 'knife', the same cannot be said for *librata* 'a hit with a book'. For books the most typical, quick telic action is more likely to be 'look up something', implying that the semantic formula ought to yield a *librata* meaning 'an event of looking up something', yet *librata* in fact follows the example of *coltellata*.

All of the denominal *-at(a)* derivations give evidence of an underlying action in which the base noun occupies some thematic role in a relatively swift telic or repeated action that is conceptually like an event of thrusting or throwing. Samek-Lodovici finds distributional evidence for this pattern even in the 'measure' nominalizations on the pattern (81 d), because these nouns cannot always be substituted for true measures like *litro* 'litre' or *metro* 'meter'. A simple formulaic description of this derivational pattern as in (81 d) is thus much too simple. It appears that measure nouns formed with *-at(a)* must stand in an immediate relation to the action of taking and moving the named quantity, as in constructions with *dare* 'give' or *gettare* 'throw', where the verb implies a source and a goal of the measuring-out action, as in (82 b). If such an action is not provided in the context, a measure noun derived with *-at(a)* cannot be employed, as shown in (82 c). Moreover, if a required underlying action scheme is not conceptually plausible, the measure sense predicted by (81 d) is not available. A *bottigliata*, from *bottiglia* 'bottle', is felt to be uninterpretable as 'quantity defined by a bottle' because a bottle's contents cannot be 'thrown' in a single, coherent action [SL6b, 19].

(82) a. Questa vasca contiene 23 litri d'acqua.
'This tub contains 23 litres of water'.
b. Getta una secchiata d'acqua sul fuoco!
'Throw a bucket of water on the fire!'
c. * Questa vasca contiene 23 secchiate d'acqua.
'This tub contains 23 buckets of water'. [SL6b, 18]

The necessity of finding an action scheme into which the base noun can be incorporated easily and in some conceptually typical and salient way might seem to be a hindrance to the ready formation and understanding of nouns on the patterns in (81). Deverbal nominalizations in *-at(a)* do not face this problem, since the verb already identifies an action scheme in its semantics. Interestingly, in Samek-Lodovici's counts of lexicalized formations in *-at(a)* only 11 percent of the interpretable derivations from nouns were lexicalized, while 20 percent of the possible deverbal forms had been registered by lexicographers. Thus, it would appear that the conceptual effort involved in creating or interpreting a denominal *-at(a)* noun hinders its memorization and repeated use, although we could probably expect a high degree of $\mathcal{P}$-type productivity on this pattern. The data on derivations in *-at(a)* thus provide evidence for a mode of word formation in which action schemes involving complex unification of conceptual schemata, rather than computations over semantic formulas, play a central role.

A crucial question that has been hinted at previously is whether the intermediate forms that arise in two-stage derivations should be considered words. Samek-Lodovici calls the actions presupposed in denominal *-at(a)* nouns "ghost verbs", noting that as verbs they are sometimes rejected by speakers as non-existing or even as uninterpretable. The same problem can arise in the reversative derivation of a verb. For example, *sbarcare* 'unload a ship' cannot be based on a single-step reversal of **barcare* because 'to load a ship' is *imbarcare* in Italian. The verb required for a two-stage, denominal derivation is **barcare*, which could mean 'do something with a ship', but this verb is not felt to be well-formed. While the derivation of *capata* 'a hit given with the head (as in football or soccer)' from *capo* 'head' is attested, the intermediate verb required for a two-stage derivation **capare* does not exist and is felt to be uninterpretable. This is not, however, a genuine argument for the single-stage kind of derivation proposed by Lieber & Baayen in (73). There is nothing in the two-stage derivation that requires an intermediate *verb*; what is required is the *semantics* of a verb, attached to an appropriate, conceptually salient action scheme. The distinction between a semantic structure that includes a thematic argument structure and a syntactic entity is unfortunately covered up in many current linguistic theories, although as Jackendoff has persuasively argued, it is obviously possible to have words with no syntax or semantics, like *fiddle-de-dee* or *Ho-hum* [Jac97, 94], words with syntax but no (conceptually-based) semantics like *herself*, and it also appears possible to have concepts with no known lexicalized name or syntactic features, like 'carry out a swift action with one's head'.

The data from the derivational patterns described here thus impose some important, detailed requirements on the data structures that must be chosen to represent lexical items. Semantic data must be stored separately from syntactic representations, and lexical representations with limited life spans must be provided. These requirements are further supported by Samek-Lodovici's counts of lexicalized and interpretable but

non-lexicalized derivatives in *-at(a)*. It has been often thought that lexicalization takes place for meanings that do not conform to the transparent semantics available from a derivational pattern, since those meanings that can be computed directly from morphological structure would only be redundant if stored in the lexicon. This is indeed what the non-lexicalized nonce formations, like *cowness* and *unabove*, found by Baayen & Renouf in the London *Times* (Table 2.5), suggest. But Samek-Lodovici's counts imply the opposite: at least in these conceptually complex derivations, it is the words whose meanings appear most difficult to compute, that fail to be lexicalized. Specifically, the denominal event nouns whose meanings fall outside the standard schemes (81 a) through (81 d) show the lowest rate of lexicalization relative to the number of items that can be constructed, and they probably do appear from time to time as nonce formations without being remembered. If this finding could be confirmed in studies of other, similarly complex derivations, it would lend support to the idea that specific semantic entries tend to be created only for words that rise above a certain threshold for ease of production and understanding.

4.5.5 Gender and Inflectional Class

It is easy to overlook the fact that a derivative must be outfitted not only with argument structure and semantics, but with all of the features that any word of its category requires, including gender, conjugation or declension. A full specification of a derivational rule must provide all of these details as well as the more commonly discussed syntactic and conceptual attributes. It has been mentioned that morphological attributes are sometimes inherited from the base, as is often the case with deverbal verbs, but derivational affixes can also determine the morphological attributes of the result. German nominalizations, like those in *-ung*, *-chen*, *-lein*, or French *-(t)ion*, *-ité*, determine the gender and inflectional class of the words they form. On the other hand, the Italian diminutive *-ino/a* inherits the gender of its base. The Italian derivations in *s-* discussed above generally yield verbs of the *-are* (first) conjugation [Sch95a, 559], but the recent neologism *sfittire*, (79a, 1960) has chosen the *-ire* conjugation, perhaps because of a preference for vowel harmony between the base and the most frequently used inflectional endings. Many of the rules involved are probably language specific and unreliable, but they certainly cannot be reduced to simple notions of headedness, or featural primacy of a particular morphological constituent. They may appeal to competing principles of word formation, but in any case, they provide additional evidence of a system separate from that of syntax. Like the more difficult semantic data, they also point to operations that have no direct counterparts in syntax, and they require rules and interfaces that will be used only for building words.

4.5.6 Simultaneous Transparency and Opacity

Many relatively spontaneous derivatives can be used in senses differing from the semantics predicted by the derivational pattern, as has already been noted for words like Italian *sbobinare* (52). On the other hand, even when a lexical entry is present, it is possible to use a productive morphological pattern to obtain a nonce word differing greatly from its lexicalized homonym in meaning, but which, because of its ease of interpretation, hardly attracts notice, as we saw with German *abwandern*. At the

same time, the syntactic attributes predicted by the derivational pattern appear to be unswervingly respected; we do not find that speakers are as generous with argument structures and syntactic categories of derivatives as they are with meanings.

These observations seem to require that the interface from the lexicon to syntactic structure be in some way insulated from direct contact with semantic structure, so that a word's syntactic properties are forced to interact directly with those of other words in the sentence, while the semantic representation remains largely underspecified until some later point, when word meaning must be integrated into the larger narrative or argumentative context. In other words, the part of semantics that is directly involved in syntax is limited to the argument list preceding the semantic formula; the attachment of arguments to specific conceptual structures is probably not mediated in syntax but in some other computational structure. Again, this separation speaks for loose coupling of the semantic and syntactic representations belonging to a lexical entry.

4.6 Requirements of Linguistic Analyses

This chapter has shown that a model of word processing probably needs to provide separate generative rule systems for word formation and sentence formation to accommodate the differing requirements of the two systems and to take the observations associated with the Impoverishment of Morphology and Lexical Integrity into account, as well as differences in the ways semantics are computed. Operations roughly like those of syntax must be present in word formation, and these must be available during understanding and production of speech, not as historical processes, but 'on-line' in exactly the same way as the operations of syntax, without being part of the syntactic rule system.[17] At the same time, the interaction between the two systems must be more elaborate than a simple 'feeding' relation or rule ordering relation, and it has been hinted that the sort of control structures used to implement exception and error handlers will be necessary to account for important data like phrasal constituents in complex words. A related aspect of this interface problem concerns the representation of the lexical entries themselves, and it has been suggested that the continuum from transparent to idiosyncratic and opaque meaning in word formation requires some sort of flexible access to the range of meanings that can be attached to a complex word, as well as to the entry itself and its morphological constituents. It is perhaps fitting that this central requirement for a simulation of word processing is similar to the one which Descartes identified as the defining characteristic of human language: its ability to respond "en toutes les occurrences de la vie, de même façon que notre raison nous fait agir" — flexibly, intelligently, not like the stupidly fixed apparatus of mechanical systems or parrots that merely imitate the patterns of human language [Des62, 57].

In addition to these outer-level requirements, we have seen some representative examples of inner-level requirements that would need to be formulated for a simulation of a particular language. These have sketched how the lexical semantics, syntactic category, and argument structure of a derived word can be predicted. To the extent that moderately formal linguistic analyses are already available for many of the examples that have been presented, the step to formal specifications for the simulation of the

[17] This does not exclude their being able to account for historical change, which can be seen as resulting from the accumulation of effects due to word formation rules.

derivation of many complex words will not be large or difficult. However, the formal analyses we have seen leave many issues unresolved. The rules involved in derivation must be allowed to have access to conceptual schemata and knowledge of salient or prototypical characteristics of objects and events. For multi-stage derivations, there must be a representational format that provides the semantic and conceptual structure necessary for the following stage of derivation but does not automatically create a potentially unacceptable word (like **capare* to get *capata*). Finally, the rule systems must also allow for features like gender, inflectional paradigm and morphological category to be predicted along with the phonological or graphemic form of the new word. The next chapter will introduce some devices for formal specification with which these requirements can be met.

Chapter 5

Tools for Formal Specification of Morphology

The previous three chapters have gathered constraints or requirements for a model of word processing from data on corpus statistics, experimental word recognition phenomena, and structural linguistics. In the terminology of application-oriented software engineering, these research disciplines represent "stakeholders" in the model. Each stakeholder discipline is in a position to stake claims on how a model of word processing must behave, but no discipline is currently in a position to say by itself how the model should be specified. The only *valid* models (in the sense introduced in chapter 1) will be those that satisfy the claims of *all* stakeholders. Nevertheless, as was pointed out in chapter 4, the claims of structural linguistics do have a privileged status, insofar as, virtually by definition, linguistic analysis alone is in a position to define what is and is not knowledge of a given language, and thus to provide the "linguistic Turing-test", mentioned in section 4.1, that identifies whether or not a given model simulates the tacit knowledge speakers have of a particular linguistic system.

This chapter turns to questions concerning which implementable formalisms might be appropriate for specifying a model of word formation and processing, and of what needs to be specified. As argued in chapter 1, an implementable formal specification plays an important role in showing that a cognitive model does not derive its explanatory power from obscure factors, principles or psychological "humunculi". On the other hand, it is desirable that a requirements analysis avoid any commitment to specific formal devices while the requirements data are being gathered and evaluated. As application programmers well know, formalisms have a way of preempting requirements data and of sometimes hiding obvious relationships, while sometimes exaggerating the importance of unimportant relationships. For this reason, I have tried to refrain up to this point from stating requirements from the three stakeholder disciplines in terms of formal systems. Purely descriptive presentations of data are, however, often less informative and more difficult to understand than graphs, diagrams or formulas, and to some extent my presentation has already taken steps toward formal specifications of various aspects of the KLU model, especially as regards the linguistic requirements. Further techniques that have been developed in computational and descriptive linguistics can also help to formulate the inner-level requirements for a model a particular language, in the form of formal specifications of the language's lexical semantics

and grammar. Six kinds of formalization that have been employed in the KLU model and that will be introduced or elaborated here as specification formalisms are:

- Representation of sentence and word structure in terms of tree graphs and systems of annotated production rules (e.g., (19), chapter 4);
- Representation of allophonic and allomorphic relations in terms of context-sensitive rewrite rules, implemented as finite state transducers;
- Use of feature structures and equations for mapping between components of the grammar, as defined in the formalism of Lexical Functional Grammar;
- Representation of word and sentence meanings in terms of semantic functions and lambda abstractions, introduced in section 4.4.2;
- Modular factorization of data structures and data flow patterns along the lines of the Augmented Address Model of Caramazza et al. and the morphological Meta-Model of Schreuder & Baayen (Figs. 3.3, 3.4), with a detailed modular specification of the lexicon and an additional Indexical Structure;
- Detection of formal errors and the transfer of control to an error handler, introduced in section 1.6.

In addition, the KLU model makes use of a hierarchical segmentation procedure prior to word analysis, but a justification for this approach will not be introduced until the next chapter.

5.1 Specification via Spreading Activation

Before turning to a discussion of the formal resources that have been chosen for KLU's specification, I want to address briefly an issue that has been raised in the linguistics and psycholinguistics literature of recent years, an issue that has cast doubt on the validity of many of the formal methods computational linguistics has made available. Recent psychological discussions, e. g., [MR86], have raised doubts about symbolic computation a plausible description of cognitive processing. One important line of argument is that neural firings in the brain are far too slow (in the range of kilohertz) to implement the kinds of algorithms proposed in symbolic theories and implemented on sequential computers, while the complexity of the connections among neurons far exceeds that of computers, suggesting that other forms of computation are being used in the brain. Moreover, while brain injuries can impair various cognitive functions, the size of the impairment is typically related to the size of the injury; the death of a neuron does not produce the kind of massive failure that would result from breaking a single data or control line in a digital computer. These incontrovertible facts imply that cognitive computations must be massively parallel and slow rather than fast and serial. Moreover, the functions neurons implement cannot be expressed by the logic formulas used to describe the gate circuits of digital computers; rather, their firing is controlled by statistical summing and inhibition functions. Many psychologists and some linguists have therefore concluded that formalisms appropriate for modeling

linguistic systems must be massively parallel and statistical in character rather than discrete, symbolic operations.

A second and probably less serious argument is that the complex symbolic operations proposed in theories of grammar give little evidence of themselves in either external behavior nor in the kinds of data that can be gathered about activation areas in the brain. Objections of this form have been raised most vociferously against the more abstract operations posited in many versions of transformational grammar. Such objections cannot be entertained at all levels of linguistic theory, and chapter 3 presented considerable evidence that many units of symbolic linguistic analysis like phonemes, morphemes, words and semantic relationships do in fact have some sort of empirically visible representation in the linguistic processing system implemented in the brain. Phonemic and morphemic units, for example, have apparently been witnessed in the production of speech errors, and priming data give evidence of both morphological constituents in the surface representations of words and at a deeper level that may correspond to lexical semantics (cf. sections 3.5, 3.6.1, and 3.6.3).

A strong version of the symbolic approach to linguistic modeling would claim that evidence of *all* symbols and operations posited in linguistic theories can be discovered in the operations of the brain by experimental investigation. It would require a one-to-one mapping between all data structures and operations postulated in linguistic theory and all entities that can be independently identified in experiments. It is thoroughly possible that a such a detailed mapping has no chance of ever being verified empirically, however, and it is unreasonable, in any case, to think that a computational symbolic model must imply such a relation. As was pointed out in section 1.5, in computational systems it is often the case that the representations used in computer program sources are reorganized, sometimes drastically, to obtain optimizations for speed, compactness or reliability in the executable object code, and the original representations of a program and its data often cannot be reconstructed (decompiled) post-hoc. Yet surely no one will want to say that a piece of optimized computer runtime code is not a symbolic system, nor that its operation is not correctly described by the program from which it was compiled. A weaker, and more promising, version of the symbolic modeling approach would claim only that its representations *correspond* to those of the actual neural system via some behavior-preserving transformation. In the best of all worlds, the weak version of the symbolic approach would be able to specify this transformation to the cognitive apparatus; in effect, it would be able to tell us which patterns of neuron-firings correspond to, e.g., the event of a noun being recognized as the head of a noun phrase. Failing this, the theory might specify a transformation to some other apparatus, say, a computer, whose equivalence to the human cognitive apparatus could be verified via the linguistic Turing-test described in chapter 4. In neither case would the theory need to postulate that the same symbols and symbolic operations used to formulate the theory are also empirically visible in the implementation.

What a symbolic account of language processing does not explicitly describe, however, is language acquisition, unless the account is augmented by specific learning algorithms. The recent revival of interest in computational or "neural" networks as engineering tools is traceable to the discovery of a number of powerful learning algorithms that can be used to obtain analyses of complex problems automatically (cf. [Kha90]); these have been used most successfully for developing networks to recognize complex

patterns, as in optical character recognition. Their value as a tool for linguistic simulation, however, is less clear. Often cited as evidence in their favor is a simulation by D. Rumelhart and J. McClelland [RM86] that "learned" the rules of English past tense inflection from a corpus of present and past tense verbs and then was able to change present tense verbs that were not part of the corpus to past tense forms. This simulation has since been roundly criticized by others, e.g., [PP88], on many grounds which are not of interest here, but representational formalisms based on notions of controlled spreading of neural activation have been taken up by a number of researchers, and the Meta-Model of Schreuder & Baayen [SB95], introduced in chapter 3, is meant to be specified in a neural network formalism. By means of activation levels, thresholds and feedback functions, the Meta-Model proposes to model factors that may determine morphological productivity and degree or strength of lexicalization, variables which play a large role in the statistical and psycholinguistic studies surveyed in chapters 2 and 3, and which also play a role in studies of linguistic change. By representing lexical items as threshold-sensitive nodes in the network, the model will presumably also be able to simulate the statistical distribution of types and growth curves of corpora that were discussed in chapter 2.

Joining the psychologists, some linguists have also proposed to abandon the symbolic approach in favor of one more like the neural network models. On historical grounds the linguist J. Bybee has championed analyses of word formation and of morphological change in which the relationships between related word forms are formulated in terms of associative graphs whose nodes are individual phonemic segments [Byb88]. In Bybee's model, the phonemic representation of each lexical item is associated with related items by edges of the graph called LEXICAL CONNECTIONS that are annotated with the strengths of the phonemic associations. Such associations are recorded in memory at the level of phonemic representation. For example, in the lexical representations of the English past tense verbs *started* and *waded*, the phonemes making up the past tense morpheme *-ed* are strongly connected because they recur in many similar contexts, while the phonemes of the stems are not connected. This explains the similarity in the tense feature of the two verbs as well as the dissimilarity of other semantic features, without requiring independent lexical representation of the stem and past tense morphemes. Bybee claims that the associative graph can also represent lexical connections among irregular and suppletive forms such as *go* and *went*, and it can be used to give a formal account of the historical process leading to the suppletion. Because historically the verb *wend*, meaning 'proceed, travel', fell out of use, its past tense form *went* lost its semantic connection with its own root. The lexical connections of its phonetic form were then remapped to the much more frequently used phonetic cluster of strings *go*, *goes*, *gone*, and it took on their semantics [Byb95, 238]. Bybee claims that by viewing phonological relations from this perspective, semantic change can be described within the same formal apparatus used to describe phonology, whereas analyses based solely on morphemic segments are unable to display such associations explicitly.

In Bybee's associative graph approach, phonemes connected with a given word representation are marked with another parameter, LEXICAL STRENGTH, indicating how frequently the word in which they appear is used and how easily it can be recognized, and there is a reciprocal relation between the strength of a phoneme's lexical connections and the lexical strength of the word in which it appears. Experimental ev-

idence for such connections in the form of the "gang effect" has been found for several kinds of perception, as reported by Stemberger & McWhinney [SM88] (section 3.5.3). Bybee proposes that this relationship can account for the synchronic and diachronic relationships between competing regular and irregular forms like *wept-weeped* and *slept-*sleeped*. While both *wept* and *weeped* are felt to be acceptable in current English, **sleeped* is unacceptable on Bybee's account because of the high lexical strength of the competing representation of *slept*. Because the irregular form *slept* is frequently used, its phonemic segments are activated frequently (bold face letters in Fig. 5.1), and the resulting high lexical strength results in weak lexical connections to the corresponding segments in *sleep*, shown as dashed lines. In the representation of **sleeped*, the phonemes making up the past tense marker *-ed* are strongly connected to other regular past tense forms, meaning that **sleeped* can be recognized as a past tense form, but the connections to the stem *sleep* are too weak to let **sleeped* be recognized as an acceptable word. The irregular form *wept*, however, is too infrequently used in current English to maintain high lexical strength in most speakers's representations, which permits a stronger lexical connection from the representation of the form *weep* to *weeped* [Byb95]. The associative graph thus is able to describe suppletion as a graded preference relationship among two simultaneously available forms, rather than simply as a lexical substitution.

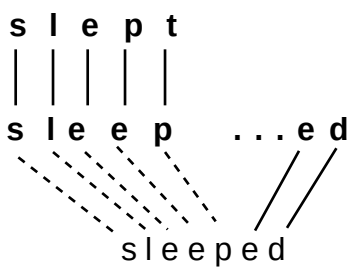


Figure 5.1: Lexical Strengths and Connections for *sleep*. Bold face: high lexical strength; Dashed line: weak lexical connection.

A further property of Bybee's associative graph is that it can account for suppletive paradigms, such as the ablaut pattern *i-u* found in *sting*, *stung*; *hang*, *hung*. Although the words belonging to this irregular pattern are used frequently and thus have high lexical strength, the shared vowels are associated strongly enough to make the incorrect conjugation *drag*, *drug* interpretable to most speakers of English; Bybee notes that this conjugation in fact appears in some American dialects [Byb95, 240]. These facts would not be explainable on a symbolic account that had no way of representing variable and competing strengths of representations and associations.

Formalisms based on weighted networks of associations thus appear to offer a kind of representation that would easily capture important linguistic observations about word formation. They also suggest a means of describing the variable productivity of word formation rules that would be necessary in a model of text production. In the form presented here and as used by Bybee, they are still insufficiently worked out to be considered an implementable formalism; whether other attempts, like that of Schreuder & Baayen [SB95] will be successful remains to be seen. Complex, sta-

tistically weighted networks can, in any case, pose considerable problems for an implementation because their expected behavior results from the sometimes inscrutable interactions of the weighting values. Which behaviors these interactions determine is not directly visible in the formal representation, and what such a system actually represents can sometimes only be discovered empirically, by testing its behavior.

5.2 Symbolic Formalisms for Word Formation

Fortunately, the phenomena described by Bybee are not exclusively amenable to connectionist explanations. They can also be formalized symbolically in terms of probabilistic production systems and finite state transducers. In fact many of the phenomena that seem to require representations in terms of associative or neural networks can be described adequately by discrete symbolic formalisms, which have the enormous advantage that they have well-defined mathematical properties and can explicitly document what the model 'knows' about the domain modeled. Accepting the weak symbolic approach to computational modeling, the question of whether psychological representations in the brain function in the same way becomes largely immaterial, for as was argued above, a computer implementation will probably also exist only as a transformation of the formal specification. Its validity will rest ultimately on its ability to pass the linguistic Turing-test, i.e., to behave in ways linguistically indistinguishable from those of a human speaker of the modeled language.

5.2.1 Probabilistic Production Systems

The possibility of giving a probabilistic description of the system of word formation rules in a language was hinted at in chapter 2, equation (15). Here it was shown that the "relative pragmatic potentialities" of all word formation patterns postulated for an arbitrarily large corpus sums to 1, which suggests that the type frequency distribution and growth curve of a corpus could be simulated by an appropriate set of stochastic rules. W. Levelt [Lev74, chapt. 3] shows that a system of stochastic production rules can be used to describe accurately the statistical distribution of sentence types in a corpus so long as each derivation defined by the rule system is unique (no sentence can be derived in more than one way). It is reasonable to think that a system of probabilistic word formation rules could be induced from the distribution of words in a corpus in the same fashion.

To take a simple example, if the statistical distribution of verbs in a corpus is such that one in every thousand verbs is *talk*, one in every five thousand is *govern*, and one in every 2500 is *discuss*, then the stochastic LEXICAL-INSERTION RULES (83) would generate exactly this distribution. I use a left directed arrow for stochastically weighted rules to indicate that their meaning is only defined for word generation, i.e., as a substitution from right to left, and I assume that the rule probabilities for a given left-hand symbol always sum to one.

(83) verb $\overset{0.001}{\longleftarrow}$ *talk*
verb $\overset{0,0002}{\longleftarrow}$ *govern*
verb $\overset{0,0004}{\longleftarrow}$ *discuss*
verb $\longleftarrow$ ...

Similarly, a set of stochastic CORPUS INSERTION RULES like (84) for all lexical categories in a language would produce a stream of words *W* (devoid of syntactic structure) having the same distribution of lexical categories found in a corresponding corpus.

(84) W $\overset{0.3}{\longleftarrow}$ noun
W $\overset{0.2}{\longleftarrow}$ verb
W $\overset{0.1}{\longleftarrow}$ pronoun
W $\overset{0.1}{\longleftarrow}$ preposition
W $\overset{0.1}{\longleftarrow}$ determiner
W $\overset{0.1}{\longleftarrow}$ adjective
W $\overset{0.1}{\longleftarrow}$ adverb

The growth curve for a corpus generated by these rules, however, would be asymptotic because the rules are only able to insert items from a finite lexicon, and at some point all words in the lexicon will have been introduced into the corpus.

To obtain an unbounded growth curve of the type usually observed in corpora and described by Kalinin's equation ((9), chapter 2) the rule system must, additionally, define a source of words not found in the lexicon, i.e., a source of spontaneous formations based on patterns like the WORD-FORMATION RULES (85 b) and (85 c). These two rules describe describe possible, but perhaps not yet lexicalized nouns. Rule (85 b) describes a null derivation, that is, the simple conversion of a verb to a new noun, like *listen* in *give the song a listen*, and (85 c) describes the creation of new nouns from verbs by the addition of suffixes. Rule (85 a) on the other hand describes the introduction of N(ouns) from the finite set of lexical nouns stored in the static lexicon (i.e., from lexical insertion rules like (83)). Together, the rules (85 a to c) document the fact that both lexical insertion from the category noun and word formation on various patterns produce members of a single, distributionally equivalent class N. As required, the probabilities of the N-producing rules (85 a to c) and the N/verb rules (85 d to g) both sum to 1.0. For clarity I shall name categories of items actually stored in the lexicon with lower case names (e.g., noun), while categories that appear in sentences, and which may originate in both word formation and in the static lexicon, are capitalized (e.g., N). Category-changing affixes are named by the resulting category followed by the base category, e. g., N(oun)/verb.

(85) a. N $\overset{0.95}{\longleftarrow}$ noun i.e., direct insertion from lexicon
b. N $\overset{0.03}{\longleftarrow}$ verb i.e., via conversion
c. N $\overset{0.02}{\longleftarrow}$ verb N/verb i.e., via derivation

d. N/verb $\overset{0.2}{\longleftarrow}$ *-er* e.g., *talker*, via derivation
e. N/verb $\overset{0.1}{\longleftarrow}$ *-or* e.g., *governor*, via derivation
f. N/verb $\overset{0.1}{\longleftarrow}$ *-ion* e.g., *discussion*, via derivation
g. N/verb $\longleftarrow$ ...

Among the N(ouns) produced by the rule system (85), it is evident that 95 percent will be lexical nouns (85 a), while 3 percent will be actively created conversions from

verbs (85 b) and 2 percent will be actively derived from verbs by suffixation (85 c). The verbs used as bases for the conversions will show a distribution based on their lexical insertion probabilities as defined in (83). Multiplying out the insertion probabilities, we would expect to see that $0.03 \times 0.001 = 0.00003$ of the produced nouns are tokens of *talk*, and $0.03 \times 0.0004 = 0.000012$ are nominal conversions of *discuss*. Alas, although *a talk* is well-formed as a nominalization from the verb *to talk*, *a discuss* is felt to be ungrammatical, perhaps because of the competing noun *discussion*. Evidently there are important restrictions on the applicability of word formation rules that are not yet described in this formalism.

N(ouns) formed from verbs via derivational suffixes are described by the rule (85 c). Again, this rule is evidently too primitive to simulate accurately the distribution of derived nouns we could expect to find in a corpus, since it produces not only *talker* and *governor*, but also unacceptable words like **talkor*, **governer* and **talkion*. It, too, would need to be elaborated with restrictions so that it would produce only well-formed nouns, but at the rates required by the corresponding growth curves in the corpus.

In fact, however, even words like *governor* are most unlikely to be produced as spontaneous formations because they are familiar and are probably already present in the lexicon. This means that many word formation rules are likely to be partially redundant with some of the lexical insertion rules. To avoid inflating the insertion probabilities of lexicalized derivatives like *governor*, some additional apparatus would need to prevent the word formation rules from producing words that already exist in the lexicon. Alternatively, we might think of the word-production rules as responsible for producing only the very infrequent words that appear in large corpora, the words like *abroadly*, *unabove*, and *joyness* that Baayen & Renouf [BR6b] found with very low frequencies in their large English corpus (section 2.8). More frequently occurring complex types are probably best represented directly in the lexicon, in the form of lexical insertion rules, annotated with their frequency of occurrence relative to other words of the same category, as in (83), especially when they have a high extent of use V but a low degree of productivity $\mathcal{P}$ (cf. section 2.9). On the other hand, word formation patterns that continue to produce hapaxes limitlessly (high $\mathcal{P}$-type productivity, as for the formation of nouns in *-er*, adverbs in *-ly* and of adjectives in *un-*; cf. chapter 2, section 2.8) produce growth curves like that of Fig. 2.4c, and they must be described by active word formation rules like (85). The rate of hapax insertion will then be controlled by the probability weightings of the rules, and the distribution of types within the pattern will be controlled by the insertion probabilities of the bases and of the affixes. To the extent that the bases, too, may be derived by further rule application, the vocabulary produced by the rule system will be virtually unbounded.

Fine tuning the probabilities of the lexical insertion rules and the word formation rules is likely to be more difficult that it might seem at first, however. When types are classified in detail, the growth curves of individual types can vary considerably according to complex factors. Recall that Baayen & Sproat [Baa92] found that derivations of adjectives in *un-* had, in general, an appreciable degree of productivity $\mathcal{P}$, but the similar pattern of deriving adjectives with *in-* actually had a low value of $\mathcal{P}$. If we determine the stochastic coefficients I and U for the English adjective-generating rules

(86) a. Adj $\overset{I}{\longleftarrow}$ *in-* adjective
b. Adj $\overset{U}{\longleftarrow}$ *un-* adjective

from a small English corpus, we get a much larger ratio of I to U than for a very large corpus, as Baayen & Sproat [BS6a] report. Now, many of the adjectives produced by rules like (86 a) are likely to be classed as lexicalized. The others are the less frequent 'virtual' words that have not been recorded by lexicographers but that, if we wait long enough, will eventually appear in a very large corpus. Both the lexicalized and the virtual forms of adjectives prefixed with *in-* can be described succinctly by just the word formation rule (86 a), which would save much space in the lexicon by avoiding a long list of lexical insertion rules for individual adjectives. Furthermore, although many adjectives formed with *in-* are probably lexicalized, a large corpus will occasionally reveal new adjectives formed with *in-*, and a rule like (86 a) must be present in any case to account for these. The difficulty with this solution is that, since prefixation with *in-* has ceased to be used frequently to produce new adjectives, it has a high V-type productivity, but a relatively low $\mathcal{P}$-type productivity, which produces a nearly asymptotic growth curve like that shown in Fig. 2.4a. To obtain the large initial growth rate for this curve, I must have a relatively large value. Since the value of I is fixed, however, something else must be done to restrict the growth rate later on. Formally, we could constrain the ultimate productivity of *in-* by adding a selectional restriction in the rule (86 a), limiting it to a smaller number of (perhaps mainly Latinate) bases than the more promiscuous *un-*. This would have the effect of generating all of the allowed *in-* derivatives fairly quickly, but it would restrict the number of distinct types ultimately produced, allowing it to reproduce the growth curve correctly, as in Fig. 2.4a (p. 56). Other significant restrictions on derivation have been described in section 4.5.3; formal devices for implementing them are likely to be complex.

It is likely that the same effect would be obtained by giving I in (86 a) a low value and placing the known instances of adjectives prefixed with *in-* redundantly in the static lexicon. This redundant approach, which previous chapters have called DUAL REPRESENTATION, is open to the criticism that it makes the lexicon unnecessarily complex, since some words will be represented both as lexical entries and in a 'virtual form' described by a production rule. Dual representation also appears to complicate the problem of assigning probabilities, since the active word formation rules can generate words whose probabilities are already accounted for by lexical insertion rules. Dual representation is needed, however, to satisfy the requirement, proposed in the AAM model of Caramazza et al., that inflected forms of frequently used words must have two forms of representation in memory. Recall that psychological evidence for dual representation was found both in pathological reading impairments (section 3.7.1) and in experimental data that implied the existence of two separate pathways for the identification of complex lexical items (sections 3.7.2 and 3.7.3). (Further linguistic evidence from semantically ambiguous derived words was presented in chapter 4.) In the AAM model, one form of representation pairs an affix directly with a limited set of stems or bases, and is equivalent to placing each of the combinations whole in the lexicon, while the other representation results from a rule-like relation that only expresses the possibility of combing a stem with an affix of the same inflectional paradigm (cf. Fig. 3.3, p. 92).

The problem remains of how to assign production probabilities to the lexicalized

words and word formation rules of a complex type so that the probabilities for all members of the type add up to one. This problem would disappear if it could be shown that psychologically distinct forms of representation are also always distinguished by meaning, and thus by type, when the corpus data are gathered. This would amount to showing that a stored representation in the AAM model always has a meaning distinguishable from the meaning of the freely formed representation. In other words, in all cases where a derivative exhibits both a lexicalized and a freely derivable semantics, as for Italian *sbobinare* with the senses 'transcribe a dictation' and 'take from a spool'((52), p. 126), or for German *abwandern* in the sense of 'emigrate' and 'use up by hiking' ((67), p. 132), the lexical decision times and priming effects of the lexicalized semantics would be similar to those for a monomorphemic stem, while the spontaneously formed homonym would show the effects expected for a complexly represented stem. Such a demonstration might well lie beyond the reach of current experimental techniques.

In any case, if we assume that words having dual representations are always semantically ambiguous, with both opaque and transparent meanings, then the opaque meaning will be attached to the full-form lexical item, and it will be inserted with that item into sentence structure by rules of the form (83). Active word formation rules like (85), however, have no access to the full-form meaning; they could only insert the meanings of the individual morphological constituents into sentence structure. Making both modes of lexical insertion available would explain how both kinds of meaning can be associated with a complex word. A high insertion probability associated with the full-form rule, which results from the growth curve of the morphological type, would explain the fact that for many complex words, the lexical meaning appears to be the one that is used most frequently. However, for morphological types having a high $\mathcal{P}$-type productivity, such as English nouns formed with *-ness*, the insertion probabilities of the word formation rules will be relatively high, and one would expect the commonly used semantics of such words to be the transparent meanings that can be inferred directly from the words' constituents — which appears generally to be the case. In fact, when the derivative has no technical or specialized meaning variants, there may be no full-form representation in the lexicon at all, even when the derivative has a high surface frequency.

5.2.2 Probabilistic Recognition

The stochastic production systems described in the previous section ought to be able to model accurately the production of large corpora in such a way that the vocabulary growth curves for individual word formation patterns would be correctly simulated. The relevance of stochastic production systems for word recognition is less apparent, since recognition is not obviously probabilistic — we either recognize a word or we don't. The stochastic weightings used to simulate lexical insertion into a corpus must, however, somehow be involved in recognition as well. For complex words with high surface frequencies we have seen that lexical decision times vary as an inverse function of the logarithm of the word's corpus-statistical frequency (cf. section 3.2.1, p. 71), which in a rule system like (83) + (84) would be reflected in the product of the production probabilities of a word's lexical insertion rule (83) and its corpus insertion rule (84). For less common complex words, where the stem frequency effect

starts to dominate (cf. section 3.5.1, p. 78), the decision latency appears to correlate with the frequency of the stem rather than that of the surface (affixed) representation, when the stem also appears frequently in a non-affixed form. For words that appear most often with affixes, word frequency lists do not provide a good measure of the stems' insertion probabilities, since the available lists count only surface forms. But recall that Bradley [Bra80] (section 3.8, p. 95) found evidence that the "cluster stem frequency" for frequently affixed stems correlated well with the lexical decision times for words derived from them on highly productive patterns, such as nominalization in *-ness*. This suggests that the stochastic factors used in the statistical model of corpus generation for lexical insertion directly to the corpus (83) + (84) and for lexical insertion to word formation (83) are closely related to the representational structures that determine recognition latency. The stochastic rules for word formation like (85 b and c) appear, however, to have no observed psychological correlates.

As mentioned in section 5.1, Bybee has drawn on the spreading activation model to explain why during recognition of a poorly perceived sound or orthographic form, the alternative with the highest lexical strength will be recognized. A symbolic word recognition system based on the same stochastically weighted rules used for simulating corpus generation can give an account of recognition preferences, and it again has the advantages of a simple and clear representational format. In such a recognizer, competing lexical alternatives for matching a given input can be weighted for 'goodness of fit' according to their insertion probabilities. When uncertainties exist about individual characters or phonemes, the recognizer will favor the interpretation with the highest lexical or word formation probability. Because insertion rules will usually weight the lexicalized forms much more heavily than the freely produced forms (as do the rules (85)), they will usually favor interpreting an unclear pattern as an item drawn directly from the lexicon rather than as a combination of smaller morphemes.

The stochastic production rule formalism with dual representation therefore meets a main requirement for representing the concept of lexical strength. Dual representation can probably be further justified by the ways in which words are generated and stored in short-term and long-term memory, as Schreuder & Baayen have proposed in their Meta-Model (section 3.9.3). The simultaneous use of atomic lexical entries and probabilistic word formation rules thus helps a model of word production to simulate the differing growth curves found for different word formation patterns, meeting one of the major requirements found in chapter 2. Incorporated into a model of word recognition, a probabilistic lexical insertion formalism with dual representation provides the data required to account for variations in lexical decision times, and it conforms to the psychological evidence presented in chapter 3 that words are sometimes recognized via full-listing representations as well as by decomposition into their constituent morphemes. Finally, it also opens the possibility of explaining the simultaneous availability of opaque and transparent semantics for complex words that was demonstrated in chapter 4. It is safe to say that dual representation is in fact the most plausible architecture for lexical access, despite the apparent redundancy that it introduces in the model.

5.2.3 Finite-State Transducers

Stochastic rule systems are thus able to capture the idea of competing lexical strengths, as proposed by Bybee [Byb95] in connectionist terms, in a symbolic formalism having a mathematically simple and tractable form. If we assume that the lexical strength expressed in the insertion rule for an irregular form like *slept* is high (insertion via rules like (83)), it will make the production or recognition of the competing rule-based form *sleeped* (rules like (84)) unlikely because the insertion probabilities for both forms must add to one (in sets of rules like (85)). By contrast, if the insertion probabilities are not greatly different for a listed form and its corresponding word formation rule, both forms may be found acceptable. So it is that *weeped* may be produced or accepted by speakers despite the existence of suppletive *wept*: the insertion probability for *wept* is relatively low and thus it does not compete so strongly with production of *weeped* via the regular word formation rule. Likewise, using the same rule, the word parser will give some credibility to *weeped* during recognition.

What the stochastic rule formalism cannot describe, however, is the relatedness of irregular paradigmatic relations like *dig-dug* and *drag-*drug*. It is safe to assume (as Bybee in fact does) that these irregular forms are rarely generated actively but are rather stored in the static lexicon. The irregular paradigms rarely lead to nonce formations in the way that productive affixation patterns do, and so they should not require active word formation rules like (85). Nevertheless, speakers are aware that many irregular paradigms are rule-governed, and they can apply vowel-changing rules to create recognizable though incorrect conjugated forms like **drug* and **thunk* (from *drag*, *think*), etc. Speakers can also compare the relative strengths of phonemic connections, as when the connection strength between the root vowel of *weep* and that of *wept* is judged to be weaker than the corresponding connection between *sleep* and *slept*. These facts are easily accounted for in a network of phoneme associations like that proposed by Bybee (Fig. 5.1), but symbolic systems have been used by linguists only to describe non-probabilistic phonology. Can the notion of lexical connections, as associations of varying strengths between phonemic representations, be captured in an implementable symbolic formalism that can be integrated with the stochastic rule formalism? In fact phonemic connections of varying strength can be modeled symbolically, in a manner similar to that proposed for modeling lexical strengths, using stochastically weighted, finite-state transducers to implement the associations.

Phonological relationships like those between the vowels of *weep* and *wept* have traditionally been described by linguists as context-dependent rewrite rules, in the form of (87), a rule meaning that the phoneme /i/ changes to /e/ when it occurs before the phonemic string /pt/.

(87) $\mathrm{i} \longrightarrow \mathrm{e} \,/\_\_\, \mathrm{pt}$

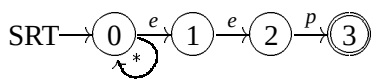
A rule of this form does not, of course, express any further relationship, such as the change in tense that it brings about, and it must be embedded in further sets of rules to describe additional changes in grammatical and semantic features that accompany the phonetic changes. But an important result in computational linguistics has shown that most systems of context-dependent phonological re-writing rules can be translated to finite state transducers.[1] If the phonological rewrite rules are given proba-

[1]This discovery has a long history, recounted by R. Kaplan and M. Kay in [KK94]. To these authors

bilistic weightings, they can express the relative strengths or tendencies of phonological changes to take place, and a system of such rules can be translated to a single stochastic transducer, which has a simple and efficient computational implementation. Without going much into detail, and eschewing mathematical rigor, I shall outline here how this might be accomplished. I shall explain briefly how automata can be used as morphological recognizers, then how in the form of transducers, they can map surface representations of words to more abstract lexical representations. To prepare the discussion of word segmentation in the next chapter and of the transducer implementation in chapter 7, I shall use orthographic rather phonological examples.

A finite-state automaton is simply a kind of formal machine that consumes a (possibly infinite) string of input symbols, switching among a finite number of states as it consumes each new symbol. It can be specified as a table of states and possible input symbols, with each input symbol effecting a transition to a new state of the machine. An equivalent and more easily understood representation is in the form of a transition graph, like (88).

(88)



This automaton describes a device that can recognize words ending in "...eep". The graph is read from left to right, starting at the symbol *SRT*, which shows that before the automaton does anything else, it is placed in a initial state 0, represented by the circle containing a 0. In this state the automaton can consume any character from the input, represented by "*", and return to state 0. Alternatively, it can accept an "e" and change to state 1. Here it can only accept another "e" and then a "p". At this point it is in state 3, which is marked by the double circle as a possible end state. Reaching the end state can be thought of as successful recognition of the input. If after accepting the first "e" any characters other than "ep" are consumed, the automaton blocks and can do nothing further. This can be construed as rejecting the input.

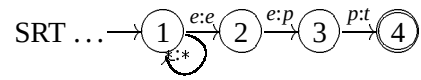
The automaton begins to suggest that one could construct a device to recognize the base forms of just those words belonging to the paradigm of *weep* and *sleep*. This automaton is not yet so clever. It accepts any string that ends with "eep", like "weep", "creep", "sleep", but unfortunately also "beep" and "xqeep". Clearly more would be needed to restrict it to just those words that genuinely belong to the paradigm; but filled out with enough additional states, the automaton could be made to recognize just these strings.

Even more useful would be a device that recognizes orthographic patterns as related, as in *sleep* and *slept* in Fig. 5.1. A device that can do this is the transducer, which functions identically to the automaton, except that it consumes *pairs* of characters rather than single characters. If the pairs are presented to a transducer in two parallel strings, the transducer can decide whether or not the two strings are related. The transducer (89), for example, recognizes any pair in which one member ends with "eep" as being related to a similar string ending in "ept"; for example, it accepts the pair of parallel strings "sleep" : "slept".

---

is due a proof that any finite system of context-dependent rewrite rules can be translated to a symmetric finite-state transducer if the non-contextual part of each rule is not allowed to apply to its own output.

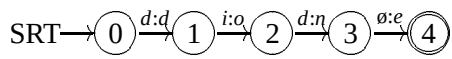
(89)



Augmented with the additional states that would be needed to let (88) recognize just the words belonging to the paradigm of *sleep*, this transducer would be able to recognize the past tense form of any word in the paradigm when it was presented simultaneously with its present tense form. It thus gives a formal, symbolic representation of the kind of relation shown in Fig. 5.1.

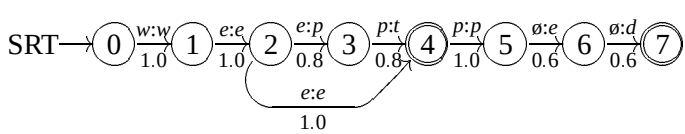
Strings associated with each other in the lexicon are not always of the same length. To allow for shifts in alignment of the two strings, the transducer can define some of its pairs as accepting an empty character, such as "ø". This is merely another way of saying that, while from one of the strings a character is consumed, no character is consumed from the other. The transducer (90), for example, recognizes *did* and *done* as a pair by pairing the "e" in "done" with an assumed empty character "ø" at the end of "did".

(90)



As presented so far, the transducers either accept or reject a pair of strings; they do not tell whether they are strongly or weakly related, and they do not express connections of variable strength between individual characters in the strings, as would be required by Bybee's associative graph model. To do this, we would need to annotate each of the pairs accepted by a transducer with a transition probability. For example, to show that *weep* is strongly paired with *wept* and less strongly paired with *weeped*, the transducer (91) contains appropriate weightings of its transitions. The probability of reaching any end state is the product of the probabilities of all of the nodes traversed, and the transition probabilities have been chosen here so that the pair *weep* : *wept* yields an acceptance probability of 0.64 and the pair *weep* : *weeped* yields 0.36, so that probability for both past tense forms of *weep* sums to 1.0. (This automaton also pairs *wee* with itself.)

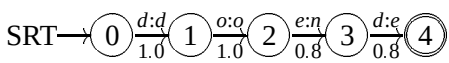
(91)



In the forms shown here, the transition graphs describe non-deterministic finite state automata, but it is well known that any non-deterministic finite state automaton or transducer can be expanded into a deterministic finite state device. Further, any deterministic finite state device can be elaborated as a probabilistic automaton or transducer, by describing each transition as a transition to a vector of states weighted by their transition probabilities [Lev74, chapt. 4], which allows a simple and efficient computational implementation.

If only surface forms of words are paired, one or another form must arbitrarily be chosen as the lexical citation form from which all related forms are derived. In many cases, however, it is more convenient to represent the lexical citation form abstractly, using an underspecified phonemic or orthographic representation. A further possibility is to transform an irregular surface form into a fictive regular form. For example, *done* can be mapped to an abstract representation as **doed*, by a transducer like (92).

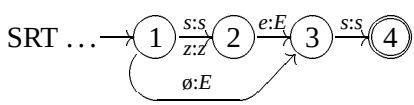
(92)



This representation can then be parsed by a word formation rule like (85 c). In this way the semantic features of the root *do* and the features of the tense inflection are factored into two separate, abstract units that can be parsed and unified, using the same formal apparatus that describes regular inflection. In most case I shall assume that this step is not necessary, since irregular inflection has virtually no $\mathcal{P}$-type productivity and the psycholinguistic evidence suggests full-form storage of irregular forms. However, in unusual cases, such as the dialect use of *drug*, instead of *dragged*, as the past tense of *drag*, the possibility of a phonological analysis must be provided to obtain a reasonably complete simulation of inflectional processing.

Phonological or orthographic accommodations are also necessary in many purely regular inflectional paradigms. Production rules by themselves are therefore not entirely adequate for specifying even concatenative morphological rules. Consider a word formation pattern as simple as English plural formation: orthographically, a word ending in "s" or "z" takes "es" as its plural morpheme; otherwise "s" is sufficient e.g., *masses* vs. *mats*. To avoid needing both "es" and "s" in the word formation rules, one can use a transducer to turn final "es" and "s" into a single abstract plural morpheme, say "Es", as in (93).

(93)



This transducer will turn "mats" into "matEs" and "masses" into "massEs", both of which then contain the identical string as the plural morpheme. This string can be further analyzed by a single word formation rule that recognizes the abstract plural morpheme "Es".

Describing all of the phonology or spelling variations of a language with transducers would, of course, be frighteningly difficult. Fortunately, well developed techniques exist for compiling phonological rules like (87) into transducers. A theorem of Kaplan & Kay [KK94] shows that any number of transducers can be combined into a single transition graph, and techniques have been developed to compile composite transducers from orthographic rule systems that can be easily written and maintained by linguists [DKK$^+$87]. Because transducers are exceedingly simple to implement on digital computers, transducer technology has established itself as a kind of panacea for computational morphology. However, the applications where transducer techniques

have been used most efficiently deal mainly with electronically stored text. In situations where the analyzer must deal with uncertain inputs, as in speech and optical text recognition, the rules of the recognizer's grammar must be formulated probabilistically to let the grammar evaluate various alternative readings of the input data. Needless to say, choosing the transition probabilities for the lexical insertion rules and for the phonological transducers in any large grammar is likely to be a daunting task. While the KLU model makes the perhaps wishful assumption that this could be accomplished satisfactorily, the implementation that will be described in chapter 7 makes no use of either probabilistic word formation rules nor probabilistic transducers. In fact, for reasons that will be discussed at greater length in the next chapter, the KLU model assumes that phonological or orthographic mapping is used only to a very limited extent in on-line processing, and the KLU model deliberately tries to minimize its reliance on a phonological component. Devices like probabilistic transducers have, however, been implemented with some success in large commercial systems for speech and optical text recognition.

5.3 Feature Structures and the LFG Formalism

Probably the single most important class of formal devices that have emerged from computational linguistics are the various unification formalisms. These have been applied both to syntactic analysis [Shi88], [Abe93] and to semantics [Als92]. The KLU model has made use of one of these, the feature unification formalism of Lexical Functional Grammar (LFG), to represent word structure, as well as constituent structure and functional structure in sentences.

Again I shall avoid a mathematical exposition, since from the point of view of a modeling project, a formalism is interesting only for its ability to capture the model's requirements and permit a reasonably efficient translation to the machine language of some computer. Both of these capabilities of LFG have been documented well in the literature (e.g., [DKMZ95]); here I shall merely summarize some of the important properties of the formalism from the perspective of the KLU model's requirements.

5.3.1 Constituent or c-structure

One component of the LFG formalism is a notation for recursive production rules that are used to formalize the kinds of facts about constituent structures of a language that can be induced by the classical techniques of structuralist analysis (cf. [Gle61]). This level of description is called C- (for constituent) STRUCTURE), and the rules are called C-STRUCTURE RULES. As shown above, probabilistic weightings can easily be added to c-structure rules for constituent analysis, in such a way that every sentence can be assigned a unique probability if the grammar observes the restriction that every sentence has a unique derivation. Most important for the KLU model, the same formalism could also be used to describe word formation probabilistically with rules of the kind shown in (83), (84) and (85). This would give a formal representation of the relative pragmatic potentiality (defined in section 2.9) of any given word formation pattern in the form of symbolic rules, and as was shown earlier it would create a word formation grammar that would simulate the type distribution and growth curves of the corpus

from which the grammar was derived.[2]

Briefly, the LFG notation describes constituent structure in sentences with production rules of the familiar form, e.g., NP $\longrightarrow$ Det (AP) N, for "noun phrase NP consists of determiner Det followed by an optional adjective phrase AP followed by noun N". Functional relations like subject and object, as well as the concept of headedness, can be shown as annotations to the production rules, e.g.,

(94) S $\longrightarrow$ NP VP
$(\uparrow \text{SUBJ}) = \downarrow$ $\uparrow = \downarrow$

reads, "head ($\uparrow$=$\downarrow$) of S(entence) is in verb phrase (VP), and the SUBJect of the sentence ($\uparrow$SUBJ)= $\downarrow$ is noun phrase (NP)". Of anything in the LFG notation, the arrows $\uparrow$ and $\downarrow$ present the most initial difficulty. $\uparrow$ always identifies a property of the left-hand constituent in the production rule, and $\downarrow$ refers to the constituent appearing directly above the arrow, so in (94) the annotation ($\uparrow$SUBJ) = $\downarrow$ reads, "sentence's ($\uparrow$) SUBJect is this constituent ($\downarrow$), namely, the NP". Control relations, as between the explicit subject of *promised* and the implicit subject of *sing* in *John promised to sing*, are expressed in a similar functional notation.

The observation that many words require the presence of other constituents has been elaborated in theories of dependency or valence grammar (cf. [Dür91, chapt. 4]); in LFG this is notated by placing a list of subcategorized arguments next to the predicate,[3] e.g.,

(95) /love/ verb, VReg, ($\uparrow$PRED) = 'Love< ($\uparrow$SUBJ), ($\uparrow$OBJ) >'

means "the verb *love* takes a subject and an object, and the corresponding constituents furnish the arguments to the semantic predicate *Love*". *verb* identifies the lexical category, and *VReg*, the inflectional class, here the regular English verb conjugation taking *-s* in the third person singular and *-ed* in the past and past participle. Although there are also notations for anaphora, long-distance dependencies (movement) and barriers, as well as other theory-specific aspects of LFG, the notational idioms given here are sufficient for describing word formation (neglecting semantics). The equal sign (=) is a unification operator whose semantics will be discussed in more detail in connection with the LFG implementation in chapter 7.

[2]Because it has not been tested in the implementation, I shall not present a stochastic LFG rule formalism. In KLU, word formation rules do not allow constraint equations. It is easy to verify in principle that such rules can be compiled to a consistent set of stochastically weighted, non-annotated rules. A compiler can reformulate any word grammar by re-encoding feature annotations as additional morphological categories, in such a way that no word form has more than one derivation. With such an encoding, probabilities can be so distributed that the total probability of each morphological category (start symbol) is 1.

[3]More correctly, in LFG the predicate's arguments specify grammatical functions, not constituents or semantic roles. The mapping to constituents is not one-to-one, since in a sentence constituents can appear, in principle, without any marking for grammatical function, and several constituents can be mapped to a single, shared grammatical function (assuming they are compatible). Since the model I am presenting here is only marginally concerned with syntax, this overview does not go into other aspects of the notation that have significance only for syntactic processing; these are described in [KB95] and most fully in [KM96].

5.3.2 Functional or f-structure

LFG differs from other unification formalisms most in its explicit notation for grammatical functions, like subject, object, modifier, etc. A central linguistic contention of LFG is that grammatical functions are used in all languages as a means of matching argument positions of lexical items' semantic structures to corresponding constituents in c-structure [BK82]. The syntax of a language either provides explicit function marking in the form of case morphemes (e.g., as in Latin) or other particles (such as meaningless prepositions, like *of* in *I'm thinking of my refrigerator*), or it marks constituents for function implicitly by syntactic position (e.g., in English, only *Mary* can be the subject of *Mary sees Jane*). There are obvious regularities in the MAPPING of semantic arguments to grammatical functions, e.g., semantic agents tend to be subjects, recipients tend to be indirect objects. But exceptions are easy to find; in *I received a present* it is the subject that maps to the role of recipient. The mapping patterns that have been discovered in most languages to date appear too unreliable to be implemented as rules, and accordingly LFG requires each lexical entry to specify explicitly how each of its semantic arguments is assigned to grammatical functions of the sentence. Nevertheless, many spontaneous derivations, especially of verbs, cannot be carried out without making mapping decisions, and derivation can be used to test how well proposed mapping rules function in practice. A model of derivation, must of course, provide such rules or at least an indication of how mapping is to be performed when a neologism is encountered.

LFG assumes that at the level of c-structure constituents receive markings for the grammatical functions they assume in the sentence [Bre82b]. For each non-terminal or pre-terminal symbol, a c-structure rule can specify a grammatical function associated with the constituent, either on the basis of position or on the basis of other explicit markings of the constituent, such as case. In (94) the first phrase of the sentence is automatically marked as SUBJect, on the basis of its position ahead of the verb. In a language with free word order like Latin, c-structure can mark almost any NP containing nominative case, regardless of position, as a possible SUBJect.

Some non-terminal symbols in the rule system may be marked with no explicit grammatical function, as is the symbol VP in (94). If a non-terminal is completely unmarked, all functional information attached to it is discarded. If the symbol is marked with the so-called PROJECTION EQUATION ($\uparrow$=$\downarrow$), all functional information attached to the symbol is passed to the left-hand side of the rule.[4] Thus, for the Latin sentence *cogito* 'I think', the required c-structure rules do nothing more than project the lexical description of *cogito* from the lexicon to the f-structure for the sentence.

(96) a. S $\longrightarrow$ VP
$\uparrow$=$\downarrow$

b. VP $\longrightarrow$ Verb
$\uparrow$=$\downarrow$

[4]More correctly, the functional information attached to the node is UNIFIED with, rather than passed to, the functional information attached to the constituent node identified by the head of the rule. A purely declarative formalism, LFG does not provide an assignment operator, and it makes no distinction between analysis and generation of sentences.

The lexical entry for the first person singular form of *cogito* is

(97) /cogito/ Verb, NIL, (↑PRED)='Think< (↑SUBJ) >'
(↑SUBJ PRED)='PRO'
(↑SUBJ NUM)=SG
(↑SUBJ PERS)=1
(↑TENSE)=PRES

This consists, first, of the citation form of the word.[5] Next appears the entry's lexical category, here Verb, followed by its inflectional class. Here, the symbol NIL specifies a full-form entry, i. e., that no inflection is allowed, since *cogito* is a fully inflected verb form. The rest of the entry consists of functional equations that project attributes of the word and their values into the syntactic structure of the sentence. Rule (96 b) projects these attributes to the constituent VP, (96 a) projects them to S(entence). The attribute PRED defines the main predicate of the sentence, 'Think'. This item, with its included argument structure, is called a LEXICAL FORM. It is not identical with the lexical semantic description of the predicate, which will be described later. The notation <(↑SUBJ)> specifies that some semantic role must be realized as the SUBJ(ect) function in the f-structure in which the verb *cogito* occurs. In this case, however, since Latin does not always require an explicit subject, the verb's inflection (-*o*) has already been assumed to project a default subject via the equation (↑SUBJ PRED)='PRO', with the further attributes NUM(ber) as SG (singular) and PERS(on) as 1 (for first).

The functional or F-DESCRIPTION for the sentence S is projected whole from the non-terminal VP, whose f-description in turn is projected whole from V. The set of equations shown in the lexical entry (97) is thus identical to the f-description. This set of equations is usually represented as a directed acyclic graph, as in (98), in which the terms of each equation are chained together from left to right to give a corresponding path through the graph. As mentioned above, the graph representation is, strictly speaking, only a visualization tool, since f-structure is defined in terms of sets of equations, and some equations are conventionally not expressed in the graphic visualization. The sentence's f-description is identical to that of the lexical entry for its verb *cogito*, as in

(98)
$$\begin{bmatrix} \text{PRED} & \text{'Think< (↑SUBJ) >'} \\ \text{SUBJ PRED} & \text{'PRO'} \\ \text{SUBJ NUM} & \text{SG} \\ \text{SUBJ PERS} & 1 \\ \text{TENSE} & \text{PRES} \end{bmatrix}$$

The chain of projection equations (↑=↓) realizes an operation called STRUCTURE COLLAPSING that can make a constituent marked with the projection equation into

[5] In the KLU notation that will be elaborated in chapter 7, the citation form is placed between slashes to indicate that it corresponds to an abstract morphemic level, removed from the surface representation of the word. This representation is thus not necessarily identical to the input string; segmentation and rules for allophonic variation, implemented as transducers like those described in section (5.2.3), can alter the inputs before they are referred to the lexicon.

something like the head of the phrase, as described in X-bar theory [FF90]. Because the projection equation unifies the attributes of the head with the attributes of the phrase, it becomes impossible to distinguish the two at the level of f-structure. Unlike X-bar theory, the LFG notation allows for more than one head within a phrase. As many constituents of the phrase can be marked with the projection equation as will unify. For example, noun phrase syntax can describe both the determiner and the noun as heads.

In the LFG notation, syntax imposes two further well-formedness conditions, called COMPLETENESS and COHERENCE. Completeness requires that each grammatical function named in the argument structure of the lexical item, e.g., SUBJ, must appear within the functional scope of the verb; completeness is thus roughly similar to the notion of c-command in government-binding theory [FF90]. Since the f-description (98) contains the SUBJ required by Think, it passes the completeness test. Coherence requires that no governable grammatical functions may occur that are not explicitly licensed by some constituent. Since (98) contains no governable grammatical functions (such as subject, object, modifier) not called for by Think, it is coherent. An f-description that passes these tests is called an f-structure. Nevertheless, f-descriptions that have not passed these tests are often carelessly called f-structures as well by LFG theoreticians.

The arrows $\uparrow$ and $\downarrow$ actually represent variables that refer to levels within the f-structure; they are therefore bound by the context of the equation in which they occur. While references to f-structure are the default, the LFG formalism allows the construction of any number of parallel unification structures, such as structures for resolving anaphoras or for constructing semantics. The reference to f-structure can be made explicit by inserting the symbol ϕ before the arrow. References to attribute-value pairs in other structures must be prefixed with a Greek letter explicitly identifying the structure.

5.4 Morphology in the LFG Formalism

The LFG formalism defined in [KM96] does not describe morphology explicitly. Rather, it assumes a separate "phonological" (actually orthographic) formalism to describe allomorphy in such a way that morphological features can be attached to the variant forms of a word, using transducers compiled from the orthographic rules. Static lexical relations among words are described by rules in a special notation within the lexical module. For languages with complex morphological systems, however, there is reason to think that this approach might be cumbersome to use and linguistically inadequate for describing derivational morphology; it also fails to allow for spontaneous derivations. As documented in [DKK$^+$87], "two-level", transducer-based morphology can identify and lemmatize morphemes in a word, but it does not provide the analysis of word structure necessary for constructing the meanings and other attributes of non-lexicalized derivatives. Approaches that combine two-level phonology with a unification-based word grammar were introduced by J. Bear [Bea86] and elaborated by H. Trost [Tro91]. More recently, Ch. Schwarze [Sch99] and K. Börjars et al. [BVC97] have proposed that the formal apparatus of feature unification combined with recursive constituent analysis, as used to describe sentence grammar in LFG, of-

fers a more adequate framework for formal morphological analysis. This approach provides a structure in which both inflectional and semantic attributes of a complex word can be computed. It does not, however, explicitly address the problems posed by segmentation. As described in chapter 6, the KLU model treats these in a separately defined, lexically steered segmentation module, using transducers like those described above in (91) to (93).

As was shown in chapter 4, there are many indications that word structures can be described with rule systems similar to those used for syntax, but having a range of expression restricted by the Impoverishment of Morphotactics. In particular, morphology appears to lack many devices like anaphora, deixis, extraction and conjunction; and the kind of embedded recursion that would produce *anti-anti-missile-missile* or *scal-in-in-o* (cf. p. 118), is probably not available in morphology. Nevertheless, even when these restrictions are accepted, doing morphology in the same formalism as syntax misleadingly suggests that, contrary to the demands of Lexical Integrity, the morphological system is simply an extension of syntax, and that there is no boundary between the two.

To clarify the boundary between the two systems, Schwarze [Sch99] has proposed a convention that defines separate notations for "grammatical words" and "lexical words". Schwarze's motivating observation is that not all lexical items can participate in syntactic constructions, as is especially evident in strongly inflecting languages. In Italian, for example, most nouns and verbs must carry an inflectional affix before they can be used in sentences; by themselves the stems are not possible constituents. The fact that the syntax of languages like English and French appears to admit stems without inflectional affixes in sentences does not mean that these languages admit words without inflection. As the agreement relations in these languages show, even the forms that do not carry overt inflectional markings are specified for inflectional attributes by some sort of default assignment. Thus the lexical item *thing* can be specified as plural by adding *-s*, but agreement facts show that *thing*, in a syntactic context, must also be inflectionally specified, in this case as singular. As the base for inflection with *-s* (or for a derivation), however, it makes little sense to say that *thing* is inherently singular; it would be preferable to have a morphologically unmarked entity *thing*, to which inflectional attributes can be added. Lexical items that are in one way or another incomplete with regard to the demands of syntax Schwarze calls LEXICAL WORDS. Until word formation attaches the attributes required by syntax, lexical words cannot be used in sentences. Thus, the lexical word *thing* is not specified for number and cannot be inserted in a sentence. Words that are well-formed from the standpoint of syntax Schwarze calls GRAMMATICAL WORDS. They carry all of the morpho-syntactic attributes required by the syntactic system for integration into sentence structure, and one of the tasks of the word-forming component is to ensure that the well-formedness constraints of syntax are fulfilled. Thus while *thing* as a lexical word cannot be used in a sentence, the grammatical word *thing* created in word formation is specified as singular, and it can be inserted in a sentence.

A grammatical word is not necessarily the product of a word formation pattern. Many words require no inflectional or other kinds of marking to be used in sentences. For example, prepositions, conjunctions, interjections, and in some languages pronouns, are well-formed 'as-is'. They can be entered in the lexicon as grammatical words, and they can be inserted directly into syntax, without word formation. On the

other hand, words that require inflection, like nouns and verbs in English, must in most cases be carried in the lexicon as lexical words, even though they can appear unaltered in sentences. So long as the possibility exists for varying an inflectional attribute, lexical economy requires that these words be stored without commitment to one of the possible inflectional values. The lack of a required attribute like inflectional specification forces a lexical word to pass through word formation before insertion into syntax. To emphasize this distinction, Schwarze [Sch99] capitalizes the categories of grammatical words, as in *Verb*. The categories of lexical words are written without capitalization, e.g., *verb*. If, however, dual representation of a word is required, both grammatical and lexical variants can be present, e. g., inflected grammatical Nouns *thing* and *things* can co-exist with an inflectionally unspecified lexical noun *thing*.

5.4.1 Word Formation Rules for Inflection

The lexicon thus appears to contain both lexical and grammatical words. Grammatical words can be inserted unaltered into the syntactic structure of a sentence, but lexical words must first pass through a word formation process that ensures syntactic well-formedness. Inflectional morphemes, too, can be considered lexical entries of a special kind that carries only inflectional attributes. The inflected form of a lexical word that requires inflection can then be described by a production rule that unifies the attributes of the stem with those of the inflectional morpheme, as in (99). When an overt inflectional morpheme is missing, the default attributes could be attributed to a null morpheme, appended in the same manner as other, visible inflections. However, a null morpheme cannot be introduced at the end of every noun stem, since e.g., *sheep* does not participate in the alternation <*null*>/*-s*. To be sure, a default singular attribute could be attached directly to those lexical entries that allow plural marking, and this attribute could be tested as a condition for adding the plural morpheme. In formalisms like LFG that do not provide explicitly for the overwriting of defaults, it is simpler to add the default attribute via a word formation rule that maps an unmarked lexical word into a grammatical word marked with the default attributes. To illustrate, (99) describes the formation of *thing* and *things* with two word formation rules. The first leaves the form of the noun unchanged, but it adds to the noun's lexical attributes the default syntactic attribute NUM(ber) as SG (singular). The second projects the attributes of the lexical noun (100 a), adding inflectional attributes from the plural morpheme *s* (100 b).

(99) a. Noun $\longrightarrow$ noun
$\uparrow=\downarrow$
$(\uparrow \text{NUM})=\text{SG}$

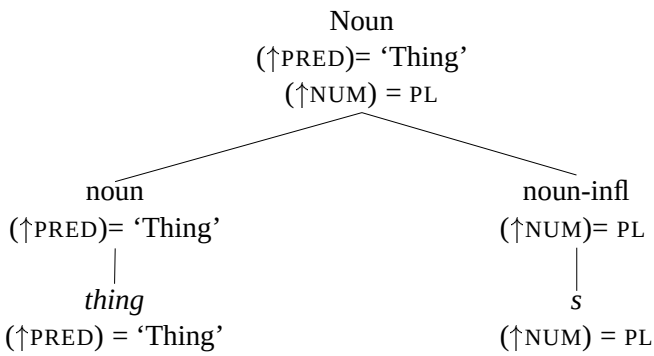
b. Noun $\longrightarrow$ noun noun-infl
$\uparrow=\downarrow$ $\uparrow=\downarrow$

(100) a. /thing/ noun, NREG, $(\uparrow \text{PRED})=$'Thing'

b. /s/ noun-infl, NREG, $(\uparrow \text{NUM})=\text{SG}$

An analysis of *things* starts by segmenting the word into constituent morphemes *thing* and *s*; then the analysis constructs a constituent tree as in (101) and projects the attributes of the stem and of the suffix to the categories noun and noun-infl (equation ↑=↓). The production rule (99 b) projects the attributes of both constituents to the category Noun. The tree (101) illustrates derivation of the final f-description for Noun by showing the full set of equations attached to each node in the tree.

(101)



5.4.2 Morphological or m-Structure

Languages with more elaborate inflectional systems than English require that inflectional morphemes be matched to the inflectional class of their stems. Thus, in French, *chose* 'thing' forms its plural by adding *-s*, but *château* 'castle' adds *-x*, giving *châteaux*. To prevent illegal morphological combinations like **châteaus*, one could introduce a feature to specify inflectional class, say INFCL, in the lexical entries for stems and affixes. The presence of the projection equation for both constituents of the grammatical word in (99 b) ensures that these would have to unify, i.e., the inflectional class of stem and affix would have to be the same. Some derivational patterns also seem to reference markings like the distinction Latinate-Germanic that sometimes governs the combination of affix and base in English, so that Latinate *in-* may combine with Latinate *possible*, letting Germanic *un-* combine with words not so restricted. The restrictions governing learned and unlearned derivations in French [SD92, 56] and Italian [Sch95a, 554] seem to presume a similar sharing of features at the word level.

There are reasons to think, however, that the attribute which controls how a stem refers to its system of affixes ought not to have a representation in f-structure. Unlike other kinds of grammatical features, inflectional features tend be spread over a small range of values distributed in the same regular fashion to all of the words to which they apply. Traditional descriptive accounts of inflectional morphology call these regular patterns paradigms and represent them as densely-filled matrices organized along the dimensions of the attributes the inflection can specify [Wun96]. A paradigm can describe regular inflection, but it can also describe irregular allomorphic variation, such as occurs with German umlaut in the second and third person singular of a verb like *laufen* 'run' (1st sing. = *laufe*, 2nd sing. = *läufst*, 3rd. sing. = *läuft*). Further, unlike other f-structure attributes, the inflectional class carries information that has no relevance beyond the word boundary and has no effect in syntax. There is therefore no reason to project the inflectional class further into the f-structure of the sentence.

To avoid cluttering f-structure with word-internal features, we can postulate an attribute-value structure that is available only to word formation and is not projected

into syntax. A matrix-like structure for representing paradigms is not available in the formal apparatus of LFG, but a largely equivalent mechanism has been implemented in the KLU model as an attribute-value structure called m-structure. Unlike the traditional paradigm matrix, however, this structure is used only to describe regular, concatenative inflection. By means of orthographic substitutions, irregular forms can be transformed into regular, concatenative forms for lexical look-up within a paradigm, as was shown for *done* ⟶ *doed* (92), but to avoid the enormous on-line computational cost that would result from reducing all irregular to regular forms, the KLU model assumes that these are lexically stored in such a way as to have precedence before orthographic analysis. In functional equations, references to m-structure are prefixed with μ, and at the point where grammatical words are inserted into syntax, the m-structure is discarded. This is seen as one of the defining boundaries between syntax and word-formation: word-formation possesses m-structure, but syntax does not. In the KLU formalism for lexical entries, the inflectional class following the lexical category is compiled to a functional equation referring to m-structure. This equation inserts the inflectional class, e.g. NREG in (99), into m-structure as the value of the predefined attribute INFCL, as in (102).

(102) $(\mu\uparrow$INFCL$)=$NREG

Lexical entries for categories that have no inflection, like conjunctions, or lexical entries that for one reason or another are listed with their inflection, as is (97), are given the inflectional class NIL. Having an INFCL attribute available, it is possible to describe the inflection of English *sheep* without having it default to singular, as would result from the rule (99). Nouns like *sheep* can be specified in the lexicon as belonging to an inflectional class UNSPEC. A third rule (103) can be added to (99) to create grammatical from lexical nouns that are not allowed to specify number. This rule attaches an attribute NUM to the grammatical Noun, as required by syntax, but it leaves the value unspecified.[6] The other rules then require an INFCL different from that of *sheep*.

(103) Noun ⟶ noun
$\uparrow=\downarrow$
$(\mu\downarrow$INFCL$)=$UNSPEC
$(\uparrow$NUM$)=\alpha$

5.4.3 Derivation Rules

Using the LFG formalism, it is relatively simple to give an account of inflectional morphology, as we have seen. Derivation, however, is by far more difficult, and it cannot be described entirely within the LFG formalism. As this point I shall show how far one can get using the LFG apparatus. A suggestion for formalizing the semantic side of derivation with recursive applications of lambda functions was presented in section 4.4.3; formal descriptions of other necessary operations will come later.

[6] Strictly speaking, unbound attribute values are not allowed in a well-formed f-structure [Kap95a, 57], which means that NUM must still be given a value somewhere in the analysis of the sentence. The KLU implementation of the formalism does, however, allow unbound attributes. This allows *the sheep slept* to have the attribute NUM without specifying whether one or several sheep are meant.

Like those of inflected words, the constituent structures of derived words are easily described in the LFG formalism. If their word formation rules are given probabilistic weightings, the rule system should also simulate the relative productivities of the derivational patterns, as has been shown above. If, in addition, the known lexicalizations belonging to a pattern are entered in the lexicon (making use of dual representation), it should be possible to simulate the growth curve of each derivational type separately, and idiosyncratic meanings of the lexicalizations will not need to be accounted for in the derivation rules. For those words created actively, however, a word formation rule must ultimately lead to a grammatical word that, like any other, is fully well-formed and can be inserted into sentence structure. This structure must have all the attributes of any grammatical word of its category, including the new word's phonological form, its lexical category, its inflectional class, its lexical semantics, as well as other syntactic attributes, like lexical form, control relations, and other selectional restrictions. Computationally, the most recalcitrant difficulties arise from the complex and sometimes non-compositional nature of the new derivative's semantics, as was demonstrated earlier with examples like French *publiciste* 'specialist for news or advertising', whose compositional derivation seems to require an invisible morpheme *-ité* (section 4.4.1), and Italian **bottigliata* 'the amount held by a bottle', which seems well-formed with reference to nouns built on the same word formation pattern, but which is felt to be ill-formed for complex conceptual reasons related to an "action scheme" in which the base of the derivation must be situated (section 4.5.4). Nevertheless, the Italian data presented in sections 4.5.3 and 4.5.4 show that even such computationally difficult derivations must be represented in the productive, actively used set of word formation rules. It would be an exaggeration to say that the KLU model provides a comprehensive solution to this problem. Here I propose to describe how the semantics of a simple derived word might be computed within the LFG formalism. I shall sketch proposals for some remaining problems in derivation that require additional apparatus in the following section.

To illustrate how word structure and semantics can be assembled using the LFG formalism, let me take a simple Italian diminutive derivation, using a pattern that is very productive in current Italian and hence a thoroughly realistic example of the kind of word that would need to be described by an active word formation rule. We start from the noun *fata* 'fairy', which is a feminine of the *a-e* paradigm, meaning its singular ends in *a* and its plural in *e*. A diminutive form meaning 'little fairy' is *fatina*, and it is also in the *a-e* paradigm. It is accepted by speakers without question as a word with unambiguous semantics, but it is not listed in the comprehensive dictionary [PF92]. The constituent structure of *fatina* includes the feminine inflection *a*, the diminutive suffix *in*, and the base, *fat*. Its plural form is *fatine*. Its semantics can be given as LITTLE(Fairy). Lexical entries for these items in the LFG format are listed in (104).

(104)

a.	/fat/	noun, A-E,	(↑PRED)='Fairy' (↑GEN)=FEM
b.	/a/	noun-infl, A-E,	(↑GEN)=FEM (↑NUM)=SG
c.	/e/	noun-infl, A-E,	(↑GEN)=FEM (↑NUM)=PL
d.	/in/	noun-suffix, α,	(↑PRED)=LITTLE(↑ARG) (μ↑INFCL)=(μ↑ARG INFCL)

Note that the stem *fat* is marked for grammatical gender. This is not to indicate its sex (which is not a grammatical attribute) but to ensure that it combines only with affixes having feminine gender. The masculine counterpart *fato* means 'fate', not 'male fairy' or 'gnome', and it would have to be formed from a stem having not only masculine gender but also a different PRED.

Two stages of word formation need to be described; one produces the new stem *fatin* derived from *fat* (105 b); the other adds the inflectional suffix *a* or *e* (105 a).

(105) a. Append inflection to a noun, obtaining grammatical Noun.

Noun ⟶	noun	noun-infl
	↑=↓	↑=↓
	μ↑=μ↓	μ↑=μ↓

b. Append derivational suffix to a noun, obtaining derived noun.

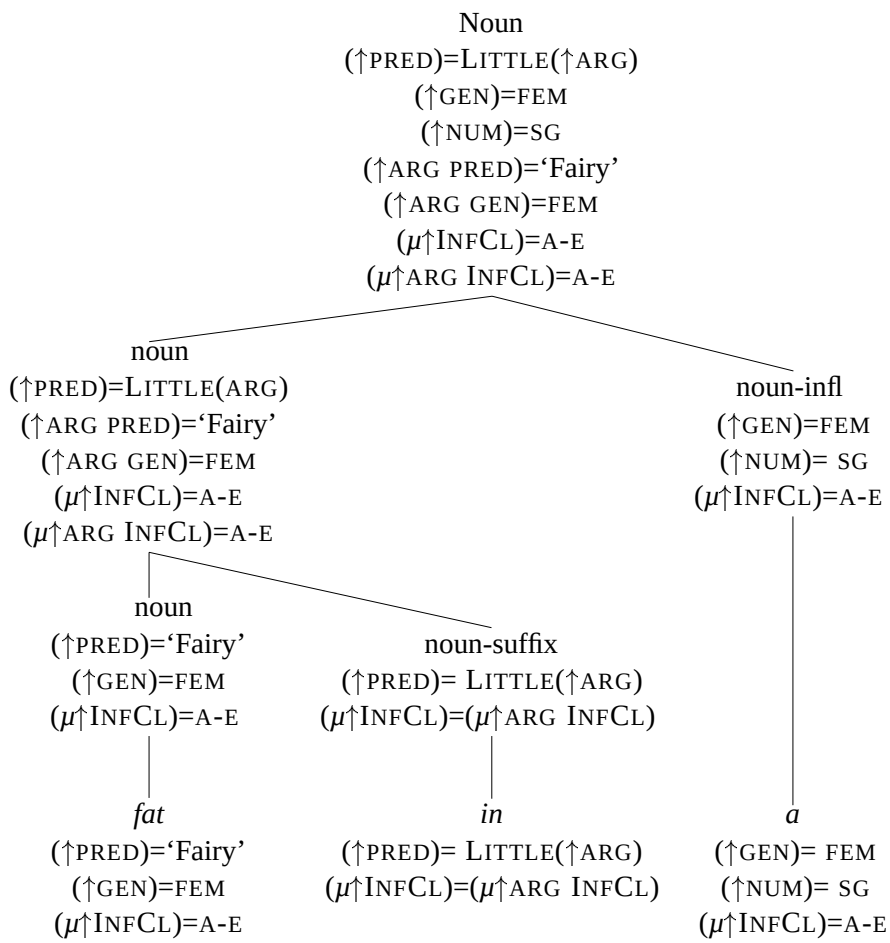
noun ⟶	noun	noun-suffix
	(↑ARG)=↓	↑=↓
	(μ↑ARG)=μ↓	

As is easy to verify, the string of morphological segments *fat in a* will produce the annotated constituent tree (106). As in (101), I again show the annotating equations that become attached to each node in the tree as the derivation proceeds. These equations originate from the lexical entries (104) and they are gathered together by the word formation rules (105). The equations μ↑= μ↓ in (105 a) ensure that the inflectional classes match (unify) between the noun stem and the inflectional affix.

In the derivational rule (105 b) simple unification of the f- and m-structure attributes between the base and the affix would not be possible. Instead, this rule describes the suffix *-in* as a semantic head, taking the base as its specifier or argument. Semantically, *-in* can be thought of as a function that will be applied to the base Fairy to give LITTLE(Fairy) when the argument in (104 d) is evaluated. With respect to inflectional class, however, the derivational suffix is not a head. It does not fix the inflectional class of the derivative — *fatina* is feminine, but *alberino* 'little tree' has to be masculine like its base *albero*. Hence, the position that names the inflectional class in the lexical entry for *-in* simply carries a variable α. Nevertheless, a word derived with *-in* does have an inflectional class (which is needed for the singular–plural inflection). To ensure that the m-structure resulting from the derivation specifies the same inflectional class as the base, the equation (μ↑INFCL)=(μ↑ARG INFCL) says that the m-structure attribute INFCL of the derivative has the same value as the base, or ARG(ument), of *-in*. The resulting set of equations for Noun in (106) can be represented as two attribute-value structures, in which the name of each structure (f- or m-structure) is the root

node of a directed acyclic graph. One of these structures (107) represents the equations referring to f-structure. The other (108) shows the m-structure, displaying the m-structure equation ($\mu\uparrow$INFCL)=A-E. This equation results from substituting for the expression ($\mu\uparrow$ARG INFCL) in the previous stage of the derivation and simplifying. It shows that the derivative has the same inflectional class as the base, and it ensures that it combines only with affixes from this class. After lexical insertion, the m-structure for the word is discarded and does not appear as part of the sentence's description.

(106)



Since the argument ARG of LITTLE has the value PRED = 'Fairy', this word structure seems close to the desired semantics, LITTLE(Fairy). In fact, at the sentence level f-structure is often seen a something akin to an underspecified semantic structure. With a little cleaning up, (107) could be used as a description of the lexical semantics of *fatina*.[7] This raises the false hope that a word grammar like that formalized in (105 b) and (104) can come close to saying all that needs to be said about derivation.

[7]I have deliberately omitted a number of fine points from this discussion to avoid clutter. I call (107) a word rather than an f-structure because there is no evidence that grammatical functions serve within word structure to map word constituents to predicates within the word; in fact, it seems unlikely that a derivational morpheme ever takes more than one argument, and the position of every argument is fixed (cf. "Extraction Impoverishment", section 4.3.2). The PRED of the base, Fairy, is a lexical form, not a semantic predicate, meaning that the semantics are found elsewhere in the lexicon. For predicates having argument structure (like verbs and prepositions), there exists a possibly arbitrary mapping between the grammatical functions of the lexical form and those of the semantic predicate (as with the mapping

(107)
$$\begin{bmatrix} \text{PRED} & \text{LITTLE}(\uparrow\text{ARG}) \\ \text{ARG} & \begin{bmatrix} \text{PRED} & \text{'Fairy'} \\ \text{GEN} & \text{FEM} \end{bmatrix} \\ \text{GEN} & \text{FEM} \\ \text{NUM} & \text{SG} \end{bmatrix}$$

(108)
$$\begin{bmatrix} \text{INFCL} & \text{A-E} \\ \text{ARG INFCL} & \text{A-E} \end{bmatrix}$$

For a more realistic example of the difficulties a formal account of derivation can present, let me take a spontaneously formed verb built on the reversative pattern described in section 4.5.3. Recall that these Italian derivations take a telic action like *bardare* 'to put a saddle on e.g., a horse' and turn it into a reversal of the action, as if playing a film of the action backwards. The derivative *sbardare* therefore means 'unsaddle'. Unlike the *undo* verbs in English, the many recent neologisms on this pattern suggest that the Italian reversative derivation is genuinely productive. Consider again a constructed example, introduced in section 4.5.3:

(109) iscrive 'register, matriculate' $\longrightarrow$ disiscrive 'ex-matriculate'.

Again, it is simple to construct a lexical fragment and a set of word formation rules like (110) to (112) to arrive at a word structure that comes close to describing the meaning of *disiscrive* (m-structure equations, which I have omitted, would be similar to those in (104) and (105)). Note that in the lexical form for 'Register', the third argument is an optional OBLQ (oblique) with a type specification. As an optional functional argument, it is enclosed in braces. If realized in the sentence, this oblique argument must be a phrase having semantics belonging to the type LOC(alization), like 'at the castle', as in *la fata iscrive il cavaliere al castello* 'the fairy registers the knight at the castle'. (In English, one registers *at* the school, *with* the authorities, *in* the telephone directory, but never *via*, *from*, or *during* the entity where the registration is held.)

relations of *receive* and *send*). Thus a lexical form is only an index to a word's semantics, and it is written differently than a semantic formula. The PRED-value of the derivational morpheme *in*, however, *is* a semantic predicate, namely LITTLE, because mapping relations do not exist for derivational morphemes. It is written in small capitals, following the convention introduced in section 4.4.3. In LFG the quote marks surrounding a lexical form denote an instantiation operator that has meaning for the construction of discourse objects. Since derivational predicates are not instantiated (cannot be quantified or referred to anaphorically), they appear without quote marks.

(110) a. /iscriv/ verb, VERE,
(↑PRED)= 'Register<(↑SUBJ), (↑OBJ), {(↑OBLQ$_{LOC}$)} >'

b. /e/ verb-infl, VERE, (↑GEN)=FEM
(↑NUM)=SG
(↑PERS)=3

c. /dis/ verb-suffix, α, (↑PRED)= REVERSE<(↑ARG)>

(111) a. Verb ⟶ verb verb-infl
↑=↓ ↑=↓

b. verb ⟶ verb verb-suffix
(↑ARG)=↓ ↑=↓

The word formation grammar will produce the following structure from the string of morphemes *dis iscrive e*.

(112)
$$\begin{bmatrix} \text{PRED} & \text{REVERSE<(↑ARG)>} \\ \text{ARG} & \begin{bmatrix} \text{PRED} & \text{'Register<(↑SUBJ), (↑OBJ), \{ (↑OBLQ}_{\text{LOC}}\text{) \} >'} \end{bmatrix} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix}$$

This structure does not, however, describe a grammatical word. The PRED that it makes available to syntax at the top level contains has a value REVERSE that is not a lexical form, and it has an argument, ARG, that is not a grammatical function, since ARG is a word structure function not defined in f-structure. The nested PRED's arguments are not the same as those required for *disiscrive* because *disiscrive* must take an oblique with a different semantic restriction. Whereas *la fata iscrive il cavaliere al castello* describes a localizing relation, the reverse — 'un-registering' — is a source relation, expressed in Italian with a preposition like *da* 'from'; hence *la fata disiscrive il cavaliere dal castello* 'the fairy un-registers the knight from the castle'. Formal techniques that might serve to construct the lexical semantics of *disiscrivere* were discussed in section 4.4.3 (cf. the discussion of Dutch *kopen* 'buy' – *verkopen*), where it was found that the reversative derivation is computationally more difficult than the incorporating derivations Stiebels has described. But many other features must be computed, as well, like *disiscrivere*'s grammatical functions and other special subcategorization requirements. If, has been proposed earlier, the derived form can eventually be stored and made available for access without word parsing, a phonetic or orthographic form also needs to be represented somewhere. Ultimately, the analysis must lead to a lexical description like (113), and a lexical semantics for UNREGISTER must somewhere come into being, along with the required argument mappings between the semantics and the lexical form 'Exmatriculate'.

(113) /disiscriv/ verb, VERE,
(↑PRED)= 'Exmatriculate< (↑SUBJ), (↑OBJ), {(↑OBLQ$_{SOURCE}$)} >'

For each of these tasks, it will be possible at least to sketch specifications for a solution, as we shall see in coming sections. A larger problem is that of coordinating word analysis with syntax. On one hand, Lexical Integrity requires that the system describing word formation operate separately from that of syntax. But, on the other hand, neologisms do not appear as isolated words; they appear *in* sentences and are analyzed simultaneously with sentence structure. How is the analysis of the word to be coordinated with the syntactic analysis? There is, I believe, no perspicuous declarative formalism that can account for this relation; rather, an adequate account cannot avoid resorting to the kind of complex control structure that any description of cooperating parallel processes requires. A solution will be proposed in the last section of this chapter; the next section addresses some serious problems in constructing the semantics and argument structure of a newly derived word.

5.5 Formalisms for Semantics and Mapping

Many scholars working in the LFG framework have adopted the convention of describing semantic structures in the same attribute-value notation used for functional structure, following a convention adopted by P-K. Halvorsen [Hal83] and elaborated in [HK95]. For two reasons, this convention has not been followed in the KLU model. For one, semanticists are most likely to be familiar with a lambda-function formalism like that introduced in section 4.4.2, and it was felt that KLU as a modeling tool would not serve its purpose if it required its users to re-orient themselves to an unfamiliar formalism. Software engineering has not found formal consistency across domains to be highly desirable (despite what many computationally oriented linguists, e. g., [PS87], appear to think); rather, experience seems to show that each sub-domain of a complex project should be specified in a formalism that is as close as possible to the conventions and formalisms already in use in the domain. A second, practical reason for adopting a simple lambda-function notation was that such functions can easily be reformulated as Horn clauses, which can be processed directly by the Prolog system in which KLU is implemented, making Prolog's apparatus for logical inference immediately available.

5.5.1 Semantic Formulas and Lexical Forms

The previous section mentioned that in the LFG formalism most syntactic lexical entries, e. g., (110 a), contain a lexical form as the value of a PRED feature. The lexical form of LFG does not itself represent the item's semantics, but is rather a kind of index to a semantic formula. The semantic formula can, in turn, be represented by a lambda expression of the form introduced in section 4.4.2. The lexical form displays no internal details of the PRED's semantics, and its argument list contains only grammatical functions, which map to but are not identical with the arguments of the semantic formula.[8] This means, of course, that the semantic description is, strictly speaking, not part of the syntactic lexical entry; it is stored in a separate collection of data that is only linked to the data structures which immediately participate in syntax. This explicit separation of syntactic and semantic components thus differs from the lexical

[8] A further difference to the semantic formula is that the lexical form can specify "non-thematic" grammatical functions that are linked to *no* semantic arguments, like the subject *it* in *it is raining*.

structures often presupposed in the psychological literature, e. g., Fig. 3.3.

The lambda expressions of section 4.4.2 describe each lexical concept in terms of further concepts that must be available at deeper levels of cognition. In the simplest cases, the lexical concept can be thought of as nothing more than a pointer to a separate item in the cognitive system connected to and used by other systems like vision, touch and locomotion.[9] The lexical concept RED:COLOR merely points to a cognitive concept outside of the linguistic system. More complex lexical concepts like NOVEL:BOOK can reflect at least a part of the non-linguistic cognitive knowledge about books in a language-specific representation. The lexical-conceptual representation of *novel*, for example, seems to include a property AUTHOR that is a defining attribute of the concept (a book without an author is not a novel). In general, the semantic formula contains variables for each of the parameters of variation of an object, as well as for each of the participants in a relation or action. These can be syntactically 'invisible' in the sense that they cannot be specified by specific syntactic arguments (like the matter investigated by an *investigante*, p. 123), or they can be 'projected' or mapped to constituents of an embedding phrase or sentence (like the AUTHOR attribute of the noun *novel*, or the agent of an action like *send*).

A characteristic of the semantic formula is that it gives a description of an object or an event independent of the syntactic representation. On the semantic level, for example, the active and the passive forms of a verb have the same logical implications and the same truth conditions. Many pairs of verbs, like *send–receive*, *teach–learn*, *buy–sell* also refer to the same event structure, so that sentences built with one verb can be immediately translated into similar (though not entirely equivalent) sentences using the other.

(114) a. John sent a package (to me).
b. I received a package (from John).
c. A package was received (from John) (?by me).
a'. Monika taught (me) yoga.
b'. I learned yoga (from Monika).
c'. Yoga was learned (by me) (?from Monika).

Nevertheless, the lambda expression contains more than conceptual event structure of the sort described by Jackendoff [Jac90]. It is evident that both *send* and *receive* denote events in which a sender initiates an action that causes some entity to be transferred to a receiver, but they make the role participants in the event visible to syntax in different ways. In their passivized forms, they present yet other views of the action. These differences are displayed by corresponding differences in the variable lists in (115).

(115) a. *send*: λy λT λA CAUSE(A, GO(T, FROM(A, TO(y))))
b. *receive*: λa λT λY CAUSE(a, GO(T, FROM(a, TO(Y))))
c. *receive* (pass.): λy λa λT CAUSE(a, GO(T, FROM(a, TO(y))))

[9] A variety of arguments for distinguishing lexical concepts from general cognitive concepts are presented in [Bie83], [Lan87] and [Sch95b].

In the formalism defined in section 4.4.2, the place a variable holds in the variable list is not necessarily determined by its place in the semantic structure. The variable list is rather built up in order of ascending thematic 'prominence'. The variables most deeply nested in the semantic structure tend to be listed first, while argument(s) to the top-level semantic predicate (e.g., CAUSE) often appear last. However, as is evident in the contrast between *send* and *receive* (115 a and b), a single concept can be presented at the interface to syntax in different ways. In the case of *send*, the 'causer' *A* is presented as the most prominent argument, and syntactically it is realized as the subject. For *receive*, the recipient *Y* is most prominent. In other words, *receive* describes the same event as *send*, but from the ROLE PERSPECTIVE of the recipient rather than the agent.

Each variable in the semantic formula can be marked as obligatory or as optional: if capitalized, the argument must be specified in the syntactic environment, but if lower case, it is optional. Hence, the argument list for *send* shows that the agent *A* and the moved object or theme *T* must be specified in the syntactic environment, but the recipient *y* is optional. The argument list for *receive* shows that an agent *a* can be missing or named, but a recipient *Y* must be named. The passive form of *receive* demotes its semantically top level argument to an optional and less prominent position in the argument structure, and it requires only the thematic argument *T*. It presents the role perspective of the theme i. e., the transferred object. Furthermore, each variable of the semantic structure belongs to a type that can be specified either in the semantic structure or inherited from a deeper, conceptual level. If present in the semantic structure, the type is named by a conceptual symbol attached to the variable by a colon, as in *a*:AUTHOR (cf. (56), p. 128).

Under the variable binding conventions of the lambda-calculus, this ordering of arguments generally ensures that in derivation incorporated word constituents bind to the left-most and therefore least prominent arguments (as Stiebels shows for the derivation of German *aufsetzen* 'place on', p. 131). For a verb the most prominent, right-most argument tends to be reserved for the syntactic subject or 'external' argument, and intermediate variables in the argument structure are made available for other grammatical functions. Unfortunately, this scheme does not suffice to predict a unique role perspective for the result of a derivation, as was illustrated by Stiebels' account of German *einölen* 'to oil something' (p. 133), and it is less than completely reliable in predicting how arguments will be assigned to constituents in syntax.

5.5.2 Mapping Relations

According to the Grammatical Function Impoverishment, section 4.3.2 (p. 117), word formation does not make use of grammatical functions. For this reason, the argument (↑ARG) of a derivational predicate like REVERSE<(↑ARG)> in (112) can be seen as identical with the corresponding argument λ*A* appearing in its lambda expression. Within f-structure, however, LFG distinguishes between the functional arguments presented in a lexical form and the bound arguments of the lambda expression. I shall use the term ARGUMENT STRUCTURE henceforth to refer only to the lambda-bound arguments of the lambda expression. As described in the previous section, argument structure is marked for semantic prominence, mandatory or optional subcategorization, and thematic type, but not for syntactic realization, i. e., grammatical function. The arguments appearing in a lexical form (the value of the PRED attribute) I shall call

syntactic or FUNCTIONAL ARGUMENTS. Functional arguments are specified for grammatical function with names of grammatical functions like SUBJ(ect) and OBJ(ect), or in a more recent LFG notation as sets of abstract thematic-syntactic attributes [BZ90], but they are not marked in the lexical form for semantic prominence or thematic type. While the connections between functional arguments and the variables displayed in argument structure are to some extent predictable (e. g., the most prominent semantic argument of a verb is tends to be realized as a subject in the lexical form), they also show arbitrary variation. For this reason LFG theoreticians have generally held that the mapping between argument structure and functional arguments is too unreliable to be simply deduced from conceptual and argument structures. In the formalism of the LFG Grammar Writer's Workbench [KM96], individual mappings from grammatical functions to semantic arguments are notated explicitly with functional equations in the lexicon. The semantic predicates and their arguments do not appear in f-structure but rather in a separate attribute-value structure called semantic or s-structure [HK95]. A less consistent but simpler convention has been used to specify mapping relations in KLU, one that does not express mapping relations explicitly. Sentence semantics in KLU does not make use s-structure;[10] rather it expresses semantics in formulas that, like the lambda-expression notation, supply lists of prominence-ranked arguments. To specify a mapping relation, the KLU notation requires that a lexical form display its functional arguments in exactly the same order as the corresponding variables in the argument structure of the semantic description. The mapping relations are implicit in the ordering of semantic and functional arguments in the lexical items, and the semantic prominence hierarchy of the argument structure remains visible in the ordering of functional arguments in the lexical form (examples follow).

The lambda-function notation introduced in chapter 4 has been altered somewhat for the KLU implementation. Instead of lambda-expressions, KLU uses a notation for mapping and semantics based on Horn clauses, reflecting the underlying Prolog implementation, in which the head of the clause (the functor) is the name of a lexical form. The functor gives the semantic expression a symbolic name, a label by which the syntactic entry and derivational operations can refer to it. The prominence hierarchy of the argument structure is reversed with respect to the lambda expression, so that the most prominent argument appears first, as shown in (116).

(116) a. Lambda expression for *send:*
λy λT λA CAUSE(A, GO(T, FROM(A, TO(y))))

b. KLU semantic lexicon entry:
Send(A, T, y) ↦ CAUSE(A, GO(T, FROM(A, TO(y))))

The head of the semantic entry presents the semantic argument structure plus a name, and it is what the lexical form in a syntactic entry refers to. The right-hand side of the semantic entry specifies the lexical-conceptual structure, which I shall call the SEMANTIC FORMULA. A variable appearing within the argument structure is called an argument, but when the same variable appears in the semantic formula I shall continue

[10] Sentence semantics in KLU are in fact constructed as a logical data base of Prolog facts representing discourse objects and relations implied by sentence semantics. This makes the sentence's implications immediately available for logical queries that can be formulated in Prolog or derived from natural language questions.

to call it a parameter. The argument structure specifies which parameters can or must be syntactically bound, and its argument ranking specifies a role perspective. For each position in the argument structure of the semantic entry, the corresponding lexical form substitutes an expression containing the grammatical function that realizes the argument in syntax. A list of these substitutions would constitute the mapping relation for the lexical item. For documentation, I shall sometimes present this list as a relation $\mathcal{M}$; this is entirely redundant and is not written explicitly in the KLU lexical entries. In on-line derivation, however, $\mathcal{M}$ must be computed before a new lexical form can be constructed. The mapping relation from a grammatical function to a corresponding semantic argument is notated with a double-headed arrow, as in (117 c). The equivalence between a semantic argument like *A* in Send(A ...) and a parameter of the same name internal to the semantic formula, like *A* in CAUSE(A ...), is implied by a single-headed arrow $\mapsto$, as in (117 a). Parameters that that cannot be specified by syntactically specified constituents, like the object under investigation by an *investigante* (p. 123), may appear in the semantic formula, but they do not appear in the argument structure.

(117) a. Send(A, T, y) $\mapsto$ CAUSE(A, GO(T, FROM(A, TO(Y))))
b. /send/ noun, NREG,
(↑PRED)= 'Send< (↑SUBJ), (↑OBJ), {(↑OBLQ$_{to}$)} >'
c. $\mathcal{M}$: *A* $\leftrightarrow$ SUBJ, *T* $\leftrightarrow$ OBJ, *y* $\leftrightarrow$ OBLQ$_{to}$

The lexical semantic entry for *send* (117 a) contains three parameters, *A*, *T*, and *y*. Each of these parameters appears in the argument structure, with *A* in the most prominent ('external') and *y* in the least prominent position. The required semantic arguments *A* and *T* map to the grammatical functions SUBJ, OBJ in the syntactic lexical entry (117 b). The optional argument *y* maps to an optional OBLQ function, marked with the preposition *to* and placed between braces to show that its presence in the sentence is not required. In this well-behaved example, the semantic arguments can be assigned one after the other to grammatical functions according to an intrinsic ranking SUBJ, {OBJ, OBLQ}, OBJ2, as proposed by Bresnan and Zaenen [BZ90], to give the mapping $\mathcal{M}$ (117 c). However, English often (but not always) allows the OBLQ$_{:to}$ function to be realized as a simple indirect object, so an additional mapping relation must also be recorded for *send*, as in (118 c). The two alternative syntactic entries might at first appear redundant, insofar as (118 c) could be derived from (117 c) simply by remapping OBLQ$_{:to}$ to OBJ and OBJ to OBJ2. However this remapping rule has exceptions, like *donate*, which does not allow **we donated the school a microscope*, in contrast to *give*, which is all but identical to *donate* at the level of semantic structure. The existence of such arbitrary variations in mapping is often cited in the LFG literature as a reason for requiring that each semantic argument be explicitly paired with its grammatical function in each item of the lexicon.

(118) a. (Lexical semantics identical with (117 a))
b. /send/ noun, NREG,
(↑PRED)= 'Send< (↑SUBJ), (↑OBJ2), (↑OBJ) >'
c. $\mathcal{M}$: *A* $\leftrightarrow$ SUBJ, *T* $\leftrightarrow$ OBJ2, *y* $\leftrightarrow$ OBJ

Argument structure and mapping relations for *send* and *receive* show again more clearly that the difference between two lexical items can result mainly from differences in the ways a shared semantic formula interfaces to syntax.

(119) a. Receive(Y, T, a) $\mapsto$ CAUSE(a, GO(T, FROM(a, TO(Y))))
b. /receive/ noun, NREG,
(↑PRED)= 'Receive< (↑SUBJ), (↑OBJ), {(↑OBLQ$_{from}$)} >'
c. $\mathcal{M}$: $Y \leftrightarrow$ SUBJ, $T \leftrightarrow$ OBJ, $a \leftrightarrow$ OBLQ$_{from}$

The variations evident here indicate that an underlying lexical-conceptual structure does not uniquely determine argument structure or mapping. Exactly how mapping relations arise is still a topic of considerable interest. One important study of argument structure [Gri90] has shown that both nesting levels within the semantic decomposition and the prominence rankings of the arguments constrain the possible mapping relations; other studies, e. g., [BZ90], have postulated a calculus of thematic role features that interact with argument prominence to determine the mapping relation. The crucial point is that both conceptual prominence (position of a parameter within the semantic formula) and argument prominence appear to be involved in fixing mapping relations. To account for a mapping relation, both of these prominence orderings must therefore have separate, distinguishable representations.

As formulated so far, mapping rules are simply generalizations about fixed structures stored in the lexicon, and as such they would not need to be implemented as part of a model of word processing. However, in order to process spontaneous derivations like Stiebels' German nonce formation *abwandern* 'hike off' (section 4.4.3, p. 132), means must be provided to compute at run time a mapping that cannot be fetched from the lexicon.

(120) a. Abwandern(A, p, U) $\mapsto$ HIKE(A) & p:PATH &DECREMENT(U)
b. /abwandern/ noun, NREG,
(↑PRED)= 'Abwandern< (↑SUBJ), (↑OBJ), {(↑OBLQ$_{LOC}$)} >'
c. $\mathcal{M}$: $A \leftrightarrow$ SUBJ, $U \leftrightarrow$OBJ, $p \leftrightarrow$ OBLQ$_{LOC}$

It has been suggested that each language defines a set of default mappings between argument structure and functional structure based on argument rankings, and this might be enough to account for the mapping relations in nonce forms. But close examination of (120) shows that its semantic structure is too flat to allow an unambiguous ranking of its arguments. Since conjunction (&) is commutative, either the parameter of HIKE or of DECREMENT could assume the most prominent position in argument structure. Nevertheless, speakers rarely seem to have problems in identifying the functional assignments for spontaneous derivations like *abwandern*, and a semantic formalism for derivation that produces both semantic structure and an argument ranking suitable for mapping ought to be within range of current theory. Unfortunately, the data in current mapping studies does not appear to have been controlled specifically for morphological productivity. A clearer picture might emerge if mapping studies attempted to define the rules that would be needed to account only for the mapping relations of newly constructed words rather than for words that may have been subject to additional, arbitrary forces in the course of lexicalization. For the purposes at hand, the important point is that the formalism used to model derivational semantics must furnish not only

the derivative's semantics, but also a means of computing the relation $\mathcal{M}$ from the argument structure and semantic formula of the derivative.

5.6 Representing Derivational Semantics

As was mentioned above, the KLU implementation represents semantic formulas as Horn clauses rather than in a lambda-calculus, and it carries out semantic computations in Prolog. This is partly a matter of convenience, but derivation actually imposes a number of requirements for which a formalism based on a lambda-calculus is less than ideally suited. The example derivations presented in the last chapter led to new semantic formulas for derivatives (e. g., (70)), but not to new lexical entries. In fact, however, the hypothesis of dual representation requires both: it must be possible to analyze a well-formed sequence of morphemes, and it must be possible to store the result of the analysis so that subsequent accesses do not analyze it again. This requires not only a semantic formula as a result of the analysis, but also a new lexical entry, containing an orthographic access form, a lexical form, a conceptual predicate or a semantic formula and a mapping relation. Furthermore, as was shown in sections 4.4.1 and 4.4.4, the meanings of complex derivations often do not follow straightforwardly from applying the semantics of the affix to the semantics of the base or from incorporating an affix into an argument position of the base. Conceptual factors can also play a large role in identifying the meaning finally attached to a derivation, as was shown for some Italian nominalizations in *-at(a)* in section 4.5.4. Hence, a formal apparatus for describing derivations must provide the possibility of giving the derivative a new lexical semantics that is different in form from the compositionally derived semantic formula.

Prolog offers two formal devices that can be used to construct such an apparatus: one is a pattern-matching query mechanism; the other is the primitive *assert*. By matching a semantic formula to a data base of conceptual knowledge, a new semantic formula can be found that does not necessarily result directly from the computed semantics of the derivative. Once all of the elements of a new lexical entry have been assembled, the static lexicon can be expanded by using the operation *assert* to place the new item in the lexicon. To clarify these steps, I shall complete the sketch of a formal derivation of Italian *disiscrivere* that was begun in (110) to (113), and I shall propose a similar sketch for Italian *librata* 'a blow with a book'. An implementation of this derivation of *disiscrivere* in the KLU system will be presented in chapter 7.

5.6.1 Formal Derivation of *disiscrive*

Using lexical items like (121) and word formation rules (111), section 5.4.3 suggested we could construct the word structure (122) for the inflected derivative *disiscrive* 'unregister, ex-matriculate' in a manner similar to the derivation of the diminutive *fatina* 'little fairy'.

(121) a. /iscriv/ verb, VERE,
(↑PRED)= 'Register<(↑SUBJ), (↑OBJ), {(↑OBLQ$_{LOC}$)} >'

b. /e/ verb-infl, VERE, (↑GEN)=FEM
(↑NUM)=SG
(↑PERS)=3

c. /dis/ verb-suffix, α, (↑DPRED)= REVERSE<(↑ARG)>

(122)
$$\begin{bmatrix} \text{DPRED} & \text{REVERSE}\langle(\uparrow\text{ARG})\rangle \\ \text{ARG} & \begin{bmatrix} \text{PRED} & \text{`Register}\langle(\uparrow\text{SUBJ}),(\uparrow\text{OBJ}),\{(\uparrow\text{OBLQ}_{\text{LOC}})\}\rangle\text{'} \end{bmatrix} \\ \text{NUM} & \text{SG} \\ \text{PERS} & 3 \end{bmatrix}$$

Then by some sort of derivational magic, it was assumed that a new syntactic lexical item like (123) would come into being.

(123) /disiscriv/ verb, VERE,
(↑PRED)= 'Exmatriculate< (↑SUBJ), (↑OBJ), {(↑OBLQ$_{\text{SOURCE}}$)} >'
(↑OBLQ PRED) =$_c$ 'Da<>'

It was pointed out that the structure (122) is not well formed from the perspective of syntax because it contains an unmapped predicate, REVERSE<(↑ARG)> with an argument that is not a grammatical function. In fact, it may be the case that all derivations lead at first to this kind of ill-formedness, even for simple diminutives like *fatina*. To make it easier to recognize an ill-formed word structure resulting from derivation, I shall use the function DPRED instead of PRED for derivational predicates. The presence of a DPRED in a word structure is taken to mean that the word structure cannot be inserted into syntax; it must be analyzed further until it meets the well-formedness conditions of syntax. Since only DPREDs specify unmapped predicates rather than mapped lexical forms, a well-formed f-structure can be obtained by eliminating DPREDs. Additionally, a lexical access form must be created, and a mapping relation must be established between a new lexical form and a corresponding semantic concept. This measure in effect realizes most of the requirements imposed by Lexical Integrity, as developed in sections 4.3.2 and 5.4.

In addition to a PRED with arguments mapped to an argument structure of some semantic formula, the derivation must also define other lexical attributes, such as grammatical gender and inflectional class (as in the derivation of *fatina*). Some words, especially verbs, can require equations to express control relations. For example, if it were necessary to derive a verb *misadvise* meaning 'advise badly', then the new verb would need a control relation that equates its object *you* $_i$ with the missing subject *ø* $_i$ of the controlled infinitive, as in *I misadvised you* $_i$ *to ø*$_i$ *visit Alaska*.[11] In case marking languages, a derived verb must be outfitted with case markings for each of its functional arguments. To specify the rules governing each of these steps in detail for a given language would entail a large empirical investigation; here I can only propose

[11] Syntactic control relations and their representation in LFG are described in [Bre82a].

solutions based on impressions of what might be necessary for a few Italian derivations like those studied in Mayo et al. [MSSZ95]. I shall sketch each of the steps in enough detail to at least suggest the principles involved. Details related to how the KLU system implements these propasals are presented in chapter 7.

The processing used in the KLU model to obtain (123) from (122) first carries out a morphotactic analysis of the word, using an LFG-like word grammar, that results in an attribute-value structure like (122). It then assembles a new derived lexical entry (123) by means of the following steps:

(124) a. From the head category of the word formation rule, assign the lexical category of the derivative to a variable *Cat*.

b. Search the conceptual data base for a concept named by a semantic formula *Concpt* with a role perspective and an argument structure *Args* that matches the expression shown as the value of the top-most DPRED.

c. Apply mapping rules to this item to obtain a preferred mapping relation $\mathcal{M}$. In addition, gather all functional argument constraints like case markings, selectional restrictions and control relations into a list *Eqns*.

d. Using $\mathcal{M}$ and *Args*, construct a lexical form *Form*.

e. Observing orthographic and phonological constraints, create an access form /*Acc*/ from the access forms of the derivative's constituent morphemes (excluding inflectional morphemes).

f. Observing any special restrictions, assign the value of ($\mu\uparrow$INFCL), obtained from the word formation rule, to $Infl$.

g. Create (assert) a lexical semantic item
$Form \mapsto Concpt$.

h. Create (assert) a lexical syntactic item
/*Acc*/ $\quad \begin{matrix} Cat, Infl, (\uparrow \text{PRED}) = Form \\ Eqns \end{matrix}$

Lexical category of the derivative *Cat*. The derivative's category is not available in the word's attribute-value structure (122), although the word formation rules could add it as the value of an attribute like CAT. As a matter of convenience, in the KLU implementation *Cat* is simply copied from the head of the word formation rule.

Search for a lexical-conceptual item $Args \mapsto Concpt$. Using a formal apparatus like the one introduced in the last chapter, it is possible to derive semantic formulas for some derivations on the basis of semantic structures alone, without reference to conceptual or 'encyclopedic' knowledge. Nevertheless, it was shown that this apparatus is inadequate by itself in cases like German *einölen* 'place oil in something' (p. 133) and Italian **bottigliata* 'amount contained in a bottle' (p. 142). Specific aspects of argument structure and functional assignment, as well as the conceptual well-formedness of the derivative, can evidently only be obtained by reference to knowledge of things

that the derived semantic formula might name. Moreover, in cases like *librata* 'blow with a book' (p. 142), only a restricted set of entities named by a simple and general semantic formula actually belong to the derivative's meaning. These findings indicate that conceptual knowledge must play a large role in determining the semantics of a derived word.

The KLU model assumes the existence of a knowledge representation or conceptual data base module which is able to return complexly structured concepts when presented with a corresponding semantic formula.[12] For example, presented with the formula YOUNG(CAT), it would return a pointer KITTEN to the concept satisfying the query. The model assumes, however, that the lexicon includes not only symbolic references to concepts like KITTEN but more complex semantic structures like those introduced in section 4.4.2. In addition, it includes names of complex functions that can be applied to semantic structures to derive new semantic structures. In the derivation of Italian *disiscrive* (121) to (123) this conceptual data base must be queried with

REVERSE(REGISTER(l:LOC, T:THEME, A:AGENT))

in order to return a corresponding conceptual predicate

UNREGISTER(l:LOC, T:THEME, A:AGENT)

Whether the REVERSE function is truly a conceptual, rather than a semantic, operation in the case of reversative derivations is a question that has no simple answer. On one hand, operations like forming a diminutive (as in *fatina*), reorganizing argument focus (as in pairs like Dutch, *kopen* 'buy' – *verkopen* 'sell', (74) p. 136) or reversing a telic action (as in *disiscrivere*) appear in many languages. The affixes naming these operations are language-specific, but the operations themselves are probably too general to be considered part of the knowledge acquired in learning a specific language. On the other hand, it seems possible to paraphrase the meaning of a word like *unwipe* as something like 'undo the action of wiping', even though it is hard to figure out what an example of this action might be. Unlike the paraphrase of *kitten* as *young cat*, this 'undo the action of wiping' is not a description of some easily identifiable concept; it is at best a kind of semantic riddle that may or may not refer to a concept. This would imply that at least some derivations take place by simply restructuring the semantic formula, without much regard to conceptual well-formedness. In unusual contexts, a semantic formula for a word like *unwipe* may find a useful meaning (cf. the nonce formation *uncaress*, documented by Baayen & Renoulf [BR6b], p. 58), but in the absence of a widely known, corresponding conceptual structure, the word is not likely to be remembered or learned by others, as Schwarze [Sch97b] has shown in the context of tense inflection.

Some derivations, like passivization, do little more than alter the role perspective of the argument structure by rearranging the prominence hierarchy. In the form that has been assumed in the KLU model, both argument restructuring and conceptual

[12] Most of the required queries could probably be answered by a suitably constructed knowledge base like KL-ONE, as described in [BS85]. A modern view of knowledge representation systems is presented in [Bib93].

search are described outside the language-specific lexicon. Whether or not this validly represents the underlying processes will have to remain a topic for further research.

Obtain mapping relation $\mathcal{M}$ and equations *Eqns*. The model assumes that for a given language a set of rules can be specified that assign variables in the argument structure to grammatical functions. These rules presumably take account of the prominence hierarchies in both argument structure and in the semantic formula, as well as semantic variable type and the marking obligatory vs. optional (upper or lower case variable). In addition, the mapping rules are responsible for identifying syntactic constraints like case and prepositional markings, and for specifying control relations that may be associated with a grammatical function.

The thematic types of the variables are used to specify additional constraints, such as case or prepositional markers *C*, and these are entered in the list of equations *Eqns* in the form

$$(\uparrow GF) ==_c C$$

(The operator $==_c$ is a KLU-specific variant of the LFG constraint equation operator which does not fail if one its terms is not present. This equation simplifies the treatment of optional functions in the lexical form by checking only functions that are actually present in the sentence. The presence of obligatory functions is checked by a separate mechanism that ensures completeness of the f-structure.)

From $\mathcal{M}$ and *Args*, construct a lexical form. A purely formal requirement for creating a new lexical entry is a name for the new semantic functor, e. g., 'Exmatriculate' in (123) and for the corresponding lexical form. Note that the semantic functor is not a semantic formula, nor does it point directly to anything in conceptual structure. Thus in a semantic entry like

$$\text{Red(X)} \mapsto \textsc{Red}\text{(X)}$$

Red(X) merely provides a link to a lexical form 'Red' (which can appear as a PRED-value attached to an access form /red/), while RED(X) is a predicate found in conceptual structure.[13] Thus any arbitrarily generated symbol can be used as a functor and as a lexical form, without reference to either the access form or to the names of conceptual predicates, but for readability the KLU program generates a symbol *Form* based on the access form of a derivative, e. g., "Dis_iscriv".

Using $\mathcal{M}$, each of the upper case variables in the argument structure *Args* is replaced by an expression *FArg* of the form $(\uparrow GF)$, where *GF* is the grammatical function corresponding to the variable. Lower case variables are replaced by $\{(\uparrow GF)\}$. All of these are assembled into a lexical form

$$\text{'}Form < FArg_1, FArg_2, \ldots, FArg_n, > \text{'}.$$

[13] Thus the PRED feature in KLU has a slightly different meaning than is assumed in much of the LFG literature because its attribute, the lexical form, is not itself a predicate but merely a link to one.

Create an access form /*Acc*/. In many cases a new lexical access form can be created simply by concatenating the access forms of the derivative's constituents: /dis/ + /inscriv/ $\longrightarrow$ /disiscriv/. In other cases orthographic and phonological rules must be consulted first (consider English /in/ + /politic/ $\longrightarrow$ /impolitic/). In the KLU program, orthographic transducers, as described in section 5.2.3, are provided to adjust the access string to orthographic and phonological constraints. During recognition, this operation follows one in which phonology or orthography must be undone in order to identify the morphemic constituents, e. g., *impolitic* $\longrightarrow$ /in/ + /politic/, since a segment /im/ will not be present in the lexicon.

Create (assert) a lexical semantic item. Using the attribute-value structures for a derived word (computed by word formation rules), procedures implementing the functions specified above will compute the elements of a new semantic entry. Specifically, the value of the attribute DPRED is presented as a query to the knowledge base and returns an argument structure *Args* and a semantic formula or predicate *Concpt*. The semantic item is assembled as

$$Form(Args) \mapsto Concpt.$$

A further procedure (Prolog *assert*) allocates a corresponding block of space in the lexicon and copies this structure to it.

Create (assert) a lexical syntactic item. The category of the derived word, *Cat*, its inflectional class, $Infl$, and its lexical access form, *Acc*, are available at this point, as well as a symbolic name, *Form*, for the lexical form. Applied to both *Args* and *Concpt*, the mapping rules have returned a mapping relation $\mathcal{M}$, and a subsequent procedure has constructed the lexical form. Control and constraint equations will have been supplied by the mapping rules in a list of equations Eqn_n. These components are now assembled as

$$\begin{array}{ll} /Acc/ & Cat, Infl, (\uparrow \text{PRED}) = \text{`}Form < FArg_1, FArg_2, \ldots, FArg_n, > \text{'} \\ & \quad Eqn_1 \\ & \quad Eqn_2 \\ & \quad \ldots \\ & \quad Eqn_n \end{array}$$

A further procedure allocates a corresponding block of space in the lexicon and copies this structure to it. Additionally, the access string *Acc* is entered in the index table used by the word segment recognizer.

Implementation details. Formal specification of the details of all operations required to produce full-blown lexical entries for a variety of derivational patterns is beyond the scope of the present study. As was shown in the last chapter, the reversative derivation is one of the more complex patterns in Italian and Dutch. Derivations for Italian diminutives like *fatina* and reversative verbs like *disiscrive* have been worked out in some detail. 'Proof of principle' derivations on this pattern can be fully

specified and implemented in the framework provided by the KLU model, and this at least lends weight to the claim that an implementation like the KLU program could provide a comprehensive simulation of morphological derivation. Implementations of the derivations of *fatina* and *disiscrive* are described in section 7.6.3.

5.6.2 Formal Derivation of *librata*

Two-stage, recursive derivations involving intermediate forms were presented in chapter 4. Examples were German *einrahmen* 'frame', which appears to presuppose a verb *rahmen* 'bring in proximity to a frame' ((70), section 4.4.3); Italian *sbarcare* ((80), section 4.5.3), which appears to require an intermediate verb meaning 'do something with a ship'; and *librata* 'blow given with a book' (cf. (81), section 4.5.4), which requires an intermediate meaning 'to hit with a book'. Although it was argued that these derivations are most plausibly described in two derivational steps, it was also noted that speakers sometimes judge the intermediate verbs as non-existing or unacceptable. Nevertheless, a recursive derivation could be proposed if the intermediate form were supposed to exist merely as a "ghost verb", i. e., as some form of incomplete entity that must not meet all well-formedness restrictions for grammatical or lexical words. Building on the proposal for single-step derivations of the last section, I shall show that this is in fact possible. All that is actually required of the intermediate derivative is a semantic formula with argument structure, a lexical category, and possibly an inflectional class. Neither a lexical form, a mapping relation nor an access form is required for the intermediate derivation; hence the intermediate can be seen as a semantic rump, lacking most syntactic attributes. Moreover, there is no need to see even this semantic rump as a lexical entity; it is probably simply an intermediate computational result that is discarded as soon as the derivation is completed.

Following the arguments of Samek-Lodovici [SL6b] (cf. section 4.5.4), I shall assume that the semantics of *librata* is characterized by the formula TYPICAL-EVENT-WITH(B:INSTRUMENT:MOVABLE OBJECT):STRIKE, but that the noun shows remnants of an argument structure that must have been present in an intermediate verb. The derivation must therefore proceed in two steps. The first step builds a semantic structure on the framework of an "action scheme" for quick, thrust-like actions [Sch95a, chapt. 3], with the book in the role of instrument. The second step identifies a corresponding nominal event, carries out mapping, creates a lexical form, an access form, and finally a lexical entry.

The fragment of word grammar required for the derivation comprises three lexical items:

(125) a. noun stem

/libr/	noun, O-I,	(↑PRED)= 'Book'
		(↑GEN)=MASC

b. inflection

/a/	noun-infl, A-E,	(↑GEN)=FEM
		(↑NUM)=SG

c. derivational suffix

/at/	verb/noun, α,	(↑DPRED)= EVENT-OF<(↑ARG)>
		(μ↑INFCL) =A-E

and three word formation rules,

(126) a. Derive a constituent meaning 'to hit with <base noun>'.

ghostverb ⟶ noun
(↑DPRED) = THRUST-HIT<(↑ARG1:AGENT),
(↑ARG2:PATIENT),
{(↑ARG:INSTR)} >
(↑ARG) = (↓PRED)

b. Derive a noun from a suffixed verb.

noun ⟶ ghostverb verb/noun
(↑ARG) = ↓ ↑=↓
$\mu\uparrow=\mu\downarrow$

c. Append inflection to a noun, obtaining a grammatical Noun.

Noun ⟶ noun noun-infl
↑=↓ ↑=↓
$\mu\uparrow=\mu\downarrow$ $\mu\uparrow=\mu\downarrow$

Applied to the string of segments *libr at a*, rule (126 a) first converts the noun *libr(o)* to a "ghost verb" meaning 'carry out a thrust or hit with a book', based on a lexicalized action scheme that requires some object that can be used as an instrument for such an action. Note that this 'verb' has no lexical status. Distributionally it appears at a position in word structure where lexical verb stems can appear, but it does not have the features of a verb stem that could be inflected and inserted, by itself, into a sentence (therefore the category ghostverb, to preclude a lexical search for the constituent.) Note also that the action scheme is not part of the word grammar; it is rather a conceptual structure referred to by the predicate THRUST-HIT. (Other action schemes assumed for bases of the *-at(a)* derivation can, e. g., measure out a quantity of something, as for *secchiata* in (82), p. 142.) The resulting attribute-value structure is:

(127)
$$\begin{bmatrix} \text{DPRED} & \text{EVENT-OF}<(\uparrow\text{ARG})> \\ \text{ARG} & \begin{bmatrix} \text{DPRED} & \text{THRUST-HIT}<(\uparrow\text{ARG1:AGENT}), \\ & \quad (\uparrow\text{ARG2:PATIENT}), \\ & \quad \{(\uparrow\text{ARG:INSTR})\} > \\ \text{ARG} & \text{'Book'} \end{bmatrix} \\ \text{NUM} & \text{SG} \\ \text{GEN} & \text{FEM} \end{bmatrix}$$

The nested DPRED expression in (127) is first reduced by substituting the semantics of 'Book' in the argument position for instrument. As defined in the recursive search algorithm, this reduction is completed by a query to conceptual knowledge that returns a conceptual predicate THRUST-HIT having two unbound arguments and BOOK as an

internalized instrument, as shown in (128).[14] If at this point the arguments of this predicate were mapped to grammatical functions, we could expect the first, most prominent argument to be a subject, and the second to be an object; but such a mapping would serve no purpose, since a further DPRED remains to be evaluated in (128).

(128)
$$\begin{bmatrix} \text{DPRED} & \text{EVENT-OF}\langle(\uparrow\text{ARG})\rangle \\ \text{ARG} & \begin{bmatrix} \text{DPRED} & \text{THRUST-HIT}\langle(\uparrow\text{ARG1:AGENT}), \\ & \quad (\uparrow\text{ARG2:PATIENT}), \\ & \quad \text{BOOK}\rangle \end{bmatrix} \\ \text{NUM} & \text{SG} \\ \text{GEN} & \text{FEM} \end{bmatrix}$$

The second DPRED evaluation serves roughly to turn an action concept into an object concept in (129). The instrument argument remains unchanged, but the obligatory participants are returned as optional arguments, one still providing for an agent, the other for a patient. The event concept, which I have not given a specific type, is probably referenced when the arguments are mapped to grammatical functions. That is, the AGENT of an event will likely be a subject, but as an argument of an object concept it will map, in Italian, to a prepositional phrase headed by *di*.

(129)
$$\begin{bmatrix} \text{DPRED} & \text{BLOW-WITH-BOOK}\langle \{(\uparrow\text{ARG1:AGENT})\}, \\ & \quad \{(\uparrow\text{ARG2:PATIENT})\}\rangle \\ \text{NUM} & \text{SG} \\ \text{GEN} & \text{FEM} \end{bmatrix}$$

This structure (129) is not yet well-formed from the standpoint of sentence syntax because it still contains an unmapped predicate rather than a lexical form. However, because no further, embedded DPREDs remain, the final steps of the derivation can be initiated to create a mapping relation, a lexical category, an inflectional class, and an access form, finally leading to a new lexical item (130). In addition, constraint equations must be added to indicate that the agent, if present, must be in a prepositional phrase marked with the preposition *di*, and the patient must be in a phrase marked with *a*. Likewise the conceptual predicate BLOW-WITH-BOOK must be entered in the linguistic lexicon to provide a point of reference for the lexical form 'Blow-with-Book', as in (131). (This semantic formula is probably more complex than I have represented it here.)

(130) /librat/ noun, A-E,
($\uparrow$PRED)= 'Blow-with-Book$\langle\{(\uparrow\text{OBLQ}_{:di})\}, \{(\uparrow\text{OBLQ}_{:a})\}\rangle$'
($\uparrow\text{OBLQ}_{:di}$ PRED) $=_c$ 'Di$\langle\rangle$'
($\uparrow\text{OBLQ}_{:a}$ PRED) $=_c$ 'A$\langle\rangle$'

[14] For this case, certainly it appears that little more is required than simple argument substitution, but the contention here is that conceptual knowledge must be consulted as well, if for nothing else than to verify that a book would be suitable as an instrument in a hitting action. Other derivations examined in chapter 4 make it clear, I hope, that conceptual knowledge can play a large role in selecting predicates and in resolving argument relations, and that it would be invalid to describe the reduction simply as a form of variable substitution.

(131) Blow-with-Book(a, p) $\mapsto$ BLOW-WITH-BOOK(a:AGENT, p:PATIENT)

Syntactic analysis of a sentence containing *librata* would have previously failed because morphotactic analysis of *librata* did not return a well-formed grammatical word. Now that the syntactic and semantic items (130) and (131) are available in the lexical buffer, sentence analysis can proceed as usual. Lexical Integrity is preserved because syntactic analysis triggers word analysis but it does not directly incorporate its results. Some computational advantages of this separation have already been addressed in section 1.5.2. The implementation via an error handler is specified in section 5.9.1.

5.7 Some Validity Considerations

Semantic decomposition of derived words using a small number of primitive logical predicates has been proposed in similar forms by a number of semanticists, e.g., [Dow79], [Jac90], and there is in fact surprisingly little disagreement about the primitives and the basic structure of semantic formulas for a wide range of words, mainly verbs. The painstaking logical investigations on which the semanticists' analyses rest, however, provide little direct evidence that semantic formulas have psychological reality in the sense required by a strong version of symbolic modeling. The logical incompatibility, for example, of *judge* and *misjudge* (which cannot be predicated over the same objects simultaneously) can be modeled symbolically by formulas in which these verbs share a common logical primitive *judge* and another primitive that, in the case of *misjudge*, negates the common element. This representation is not, however, the only one that could lead to the required inferences and logical incompatibility; these could follow from any number of other logical structures. One could postulate, for example, that the verbs' semantics are recorded in the lexicon as atomic predicates that inherit features from two categories described as incompatible on a much higher level of conceptual structure.

Thus, on the basis of the semantic and mapping data alone, representations of the lexical structures like (119) do not have to be psychologically 'real' in the sense of strong symbolic modeling. The cross-modal priming data of W. Marslen-Wilson et al. [MWTWO94], discussed in section 3.6.3, suggest that semantically transparent words are in fact, however, represented in semantically decomposed forms at a point in word processing very close to perception, a point where deliberative logical reasoning can play no role. The priming pairs in their data, like *unwind* – REWIND, *govern* – GOVERNOR, *misjudge* – JUDGE (section 3.6.3), are exactly those whose semantic decompositions would reveal common primitives. Their experiments seem to allow no other explanation for the observed priming than that activation of one or more primitives by the prime prepares access to that same primitive when the target is presented, and it lends psychological plausibility to a form of lexical representation favored on structural and logical grounds by linguistic analysis. However, it runs counter to a model of derivation like the one presented above, which effectively erases the predicate-argument structure built up by word formation, replacing it with an equivalent conceptual primitive. Whereas the derivation of *disiscrive* replaces the DPRED REVERSE(REGISTER) with an atomic pointer to the concept UNREGISTER, the results of Marslen-Wilson et al. predict that the derivation of *disiscrive* should lead to a se-

mantic structure in which the semantics of the base *iscrivere*, namely REGISTER, are still visible.

On the assumption that semantic representations are specific to individual languages, it might be possible to resolve this issue by repeating the experiments of Marslen-Wilson et al. with bi-lingual subjects, using primes and targets from different languages. Results similar to those reported in [MWTWO94] would open the possibility that the priming effect is not a result of shared semantic primitives but rather shared elements of a deeper conceptual representation. A negative result would indicate that primitives are shared at a level internal to a specific language, but not at a higher, conceptual level.[15]

5.8 Modularization and Encapsulation

Previous chapters have provided several indications that the mental lexicon is not a single block of similar items. Speculative explanations of the Zipf-Mandelbrot word-frequency distribution (section 2.3) suggest that the lexicon may contain a small number of frequently used, relatively simple items, while less frequently used items contain increasingly more differentiating features (section 2.4). Coding theory points out advantages that natural languages would obtain from providing small sets of symbols bearing little or no proper meaning (section 2.6.3). Descriptive linguists have, of course, long been aware that words come with many sizes and functions. A relatively small number can be characterized as FUNCTION WORDS, serving mainly to establish connections among the much larger class of CONTENT WORDS that refer more directly to things and events. Linguists have also frequently noticed that some classes of words, like conjunctions and verb auxiliaries, belong to CLOSED CLASSES, in the sense that new words are very rarely added to the class, while others, like nouns and verbs, belong to OPEN CLASSES. Lexical statistical studies, as we have seen, show that most of the vocabulary growth in a corpus occurs in these open classes, and that much of the growth can be described by the operation of productive word formation rules (section 2.7).

The data from aphasias have often been taken as an indication that major parts of the linguistic system may be to some extent both functionally and physically separated in the brain [Cur88]. Many of the results from psycholinguistic studies presented in chapter 3 indicate that affixes, as a kind of subrange of lexical entities, are processed at times and in ways different from those of stems. Particularly striking is the result of Drews' priming study [Dre89], which shows that inflectional affixes and stems may be processed at different, non-overlapping stages of the recognition process (section 3.5.2. Moreover, the fact that only a few, open categories of words can be expanded by adding elements composed from other areas of the lexicon suggests that these other areas may be accessed prior to the areas containing the open classes. These and similar observations led early on in the development of the KLU model to an effort to determine first, how the lexicon could best be divided into subclasses, and second, how word and sentence processing interact with the various classes.

[15] A clear distinction between semantic and conceptual structure is not accepted by all semanticists. However, clear differences in semantic structure between lexical items that pass as translations of each other in French and German have been demonstrated in [Sch85].

In software engineering, modularization has long been promoted as a 'divide and conquer' strategy to help organize large programs and reduce the burden of coordinating masses of detail, but it also provides a means for formally specifying the organization and data flow patterns within a cognitive model. Many formal tools are available for modularization; all involve notations for controlling the use and visibility of symbolic names, as well as notations for defining interfaces, and many are intimately tied to the definitions of particular programming languages. The KLU program makes use of controlled name spaces implemented in Prolog II+ [Pro96]. This mechanism enforces a limited form of data hiding, in which symbols belonging to a declared module can be made visible or invisible to other modules both at run-time and at compile-time by activating an execution context that specifies intervisibility among modules. Context declarations can be used to define an inheritance hierarchy for name resolution. Names that have already been used in a visible, higher-level module are compiled as references to the higher module rather than as belonging to the module in which they appear. Functions of the system like word segmentation, lexical access and parsing can be constructed so as to have access only to restricted name spaces, making them in effect members of a module or of an intermodule interface.[16]

5.8.1 Lexical Structure in the KLU Model

The data presented in chapter 2 on vocabulary growth in large corpora suggest that some sort of extensible data structure should be chosen for representing the lexicon. As has also been frequently observed in the semantics literature, the set of words is not unstructured but can be organized into subsets or classes, some of which share syntactic and semantic properties. A representation of the lexicon as an inheritance lattice would therefore be advantageous, and the LFG formalism implemented in the Xerox PARC Grammar Writer's Workbench [KM96] provides something similar by means of a macro notation. For a number of practical reasons, a different approach was taken in KLU. Using a form of controlled identifier search with defaults, the Prolog II+ environment makes it possible to define operations applying only to a chosen module or to a group of modules. Search for a lexical item, for example, can be restricted to a module, implemented as a name space, known to contain only a particular kind of item (say, inflectional morphemes). While the module mechanism does not define inheritance of structured entities (which would be necessary for inheritance of syntactic and semantic properties of lexical items), it lets one subdivide the lexicon into regions that can stand in an inheritance hierarchy with respect to atomic symbols. Thus, if a grammatical feature NUM(ber) and its possible values are defined within the highest region of the lexicon, they can be inherited by all lower regions. This means that an entry in a lower region can assign a value to its feature NUM, and this will unify with the NUM value in a lexical entry in a higher region. It is, however, also possible to block inheritance for symbols that are not meant to be shared. This has been done in KLU for lexical categories and lexical access forms. This makes it possible to define homonyms in different regions of the lexicon and ensure that they can be distinguished according to their region type. For example, a preposition *by* defined

[16] Prolog II+ does not provide an explicit object model. It does not permit hiding or inheritance of anything other than identifiers, and the modules of KLU should not be understood in terms of the object model of programming languages like C++ or Smalltalk.

as a spatial relation in a region reserved for content words can be easily distinguished from a different *by* defined as a passive marker in a region reserved for function words. Specifically, the lexicon of existing items is divided into the two sub-regions LEXICAL CONCEPTUAL STRUCTURE and OPERATIVES. A region MORPHOTACTICS describes word formation, the set of 'virtually' existing words beyond those explicitly listed. These regions correspond roughly to the traditional distinctions among content words, function words and virtual words, but with a number of finer stipulations that will be described below.

Additionally, to carry out a full simulation of word formation in the context of sentence processing, the KLU system requires a COGNITIVE CONCEPTS module containing conceptual predicates and relations, as well as a SYNTAX module that describes sentence structure. Additionally, an ALLOMORPHY module defines the transducers used to accommodate orthographic variation and allomorphy. Most of these module files are compiled from LFG-like notations to Prolog; Cognitive Concepts and Allomorphy are written directly in Prolog. The modules are ranked in the order: Cognitive Concepts, Lexical Conceptual Structure, Operatives, Syntax, and Morphotactics, as shown in Fig. 5.2. The order of these modules represents their precedence in an inheritance hierarchy: symbols are inherited downwards but not upwards. This means that a conceptual predicate defined in Cognitive Concepts can be referred to by a lexical entry in Lexical Conceptual Structure, but symbols defined in Lexical Conceptual Structure have no meaning in Cognitive Concepts. With additional notations it is possible to override the default name inheritance, but in the grammars that have been constructed with KLU this step has seldom been necessary.[17] Curiously, this module ordering was originally specified differently, but as experience in writing grammar fragments accumulated it became clear that most symbol references in fact follow in the directions implicit in the current ordering.

5.8.2 Cognitive Concepts Region, CPT

This module is meant to stand in for a knowledge representation system. Its primary function is to provide inferences about concepts and role relations, and to provide the conceptual primitives like CAUSE, BECOME, etc. from which lexical conceptual structures are built. By convention, the symbols and relations in this module are language-independent, which means that — in principle — this module ought to be the same for all languages described in the KLU framework. Names of conceptual predicates like CAUSE are defined in this region, and they are inherited by semantic formulas appearing in a language-specific lexicon module. In the current implementation this module is written in Prolog to simulate some of the operations that would be available from a genuine knowledge representation system, operations like checking and inheriting properties of concepts, as well as drawing logical inferences from conceptual premises and structures. The abbreviation Cpt is used to refer to this region.

[17] These grammars include small fragments for Spanish and Italian and a larger, fairly representative grammar for French, constructed by Ch. Schwarze and V. Knüppel, described in [Sch96].

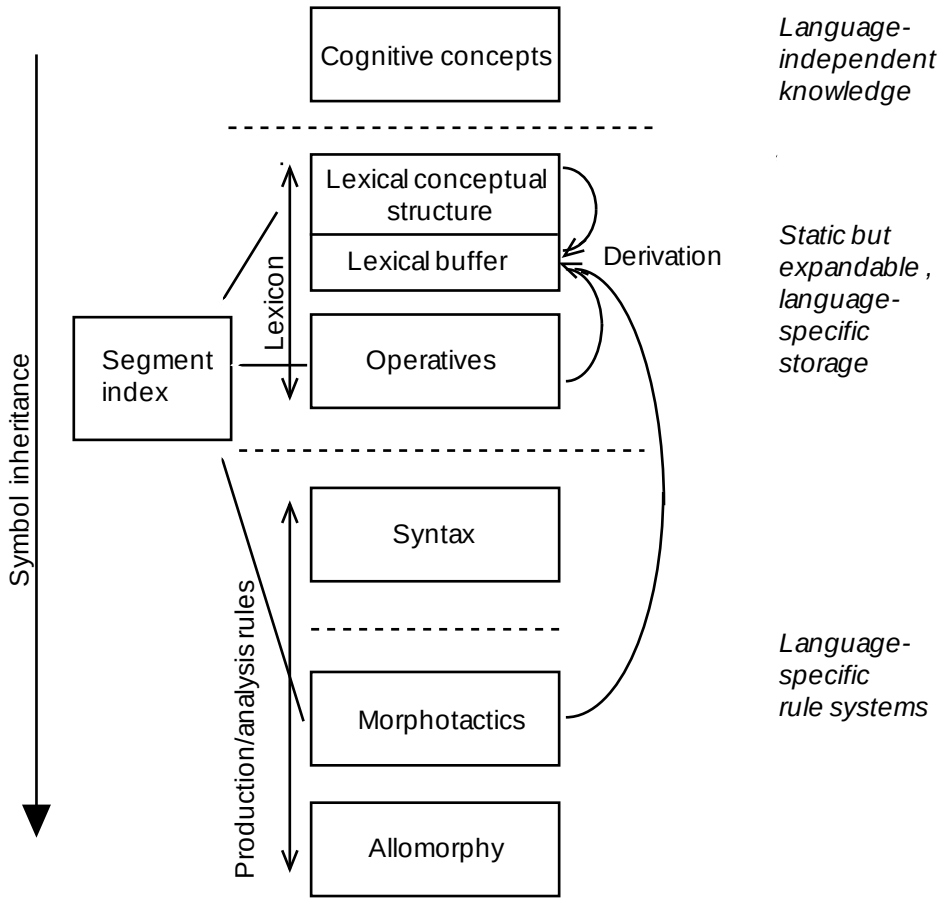


Figure 5.2: Module Hierarchy in the KLU Model of Word Formation.

5.8.3 Static Lexical Storage

Lexical Conceptual Structure Region, LCS

The largest region of the lexicon is the one containing the content words of a language. The name Lexical Conceptual Structure is meant to recall a similar cognitive region proposed by Jackendoff [Jac90], but it is in reality somewhat different from Jackendoff's Lcs. The KLU model proposes that the semantic formulas entered in Lcs are distinct from purely conceptual structures in that they are learned as part of a specific language, and as a rule they are probably much simpler and more abstract than cognitive knowledge structures. Knowing the strictly semantic properties of a concept like *computer* — its hyponyms, hyperonyms, its implicit role relations, its classification as count rather than mass noun — is much different from substantive knowledge about computers. However, the entries in Lcs can be thought of as implementing something similar to the tripartite lexical forms described by Jackendoff in [Jac97]: they contain an abstract orthographic citation form that can be internally structured and phonologically underspecified (Jackendoff's Lexical Phonological Structure), a syntactic description (Jackendoff's Lexical Syntactic Structure), and a semantic description (Jackendoff's Lexical Conceptual Structure). In agreement with Jackendoff's arguments

that these structures are only loosely tied to one another, KLU allows several syntactic descriptions to refer to the same semantic description (as with German *Apfelsine* and *Orange* both referring to the semantics ORANGE:FRUIT). The reverse, a phonological access form or lexical form without a matching lexical semantic formula, is not allowed in Lcs. In KLU's Lcs region all entries must include a syntactic description with a PRED attribute having a mapped lexical form as its value. Mapping relations are prohibited in the other regions. This is the true reason this region is designated as Lexical Conceptual Structure in the KLU model: it contains just those lexical entries that map lexical semantic formulas to syntactic entities.

Lexical semantics in KLU can, in principle, be formulated simply in model theoretic terms and evaluated against a fixed model. In the currently implemented KLU language descriptions, semantics are stated in the form of inferences that will be drawn at run time to create discourse objects in a separate discourse structure. The exact form of a lexical semantic entry depends on the working assumptions and primitives used for the semantic and conceptual analysis.

A further characteristic of KLU's Lcs predicates is that when they are used in sentences, they instantiate lexical forms as objects that can be referred to by pronouns or other deictic predicates like *the first one, this stuff, the second time, that relationship*. Two predicates of the same name do not unify. Thus, *Mary saw Mary* (or *the soldier saw the soldier*) means something different from *Mary saw herself* because the two predicates 'Mary' are instantiated to two distinct discourse objects. The LFG notation specifies that instantiation takes place for PRED values that appear in quotation marks. In the machine readable notation used by the KLU program for creating lexicon files, described in chapter 7, the user is responsible for introducing an instance variable in the semantic formula; in a correct implementation the quotation marks would be compiled to this instance variable automatically. The syntactic description of an item is otherwise largely equivalent to the standard LFG notation, as in (95).

Discourse semantics is built up largely from instatiations of concepts named by the "content words" of a sentence. A "content word" in the KLU model is an Lcs item, linked to a lexical semantic description via a fixed mapping relationship. Nouns without any form of subcategorization are a limiting case in which no argument mapping is visible. Nevertheless, the link from the lexical form, e. g., 'Truth', to a conceptual predicate TRUTH qualifies the item for inclusion in Lcs. Should an item lose its mapping relation, it would no longer contribute content to the sentence, although it would presumably still perform some other grammatical function. Since the regions below Lcs are seen as containing function words and items involved in word-formation, an implementation of the function responsible for such shifts would presumably entail a formal account of semantic bleaching and grammaticalization.

Anticipating the discussion of segmentation in the next chapter, it is worthwhile noting that it is the content words in the Lcs region that typically lie at the core of segmentable words. Because the Lcs region is by far the largest part of the static lexicon, a segmentation structure that avoids searching this region during early stages of morphological segmentation would be faster than one that searched everywhere. The segmentation algorithm described in chapter 6 is in fact so defined that an allowable segmentation hypothesis can draw only its innermost, last-found segment from the Lcs region.

A further, important characteristic of the Lcs region is that the results of deriva-

tional processes always share the characteristics of Lcs entries, having, in particular, a semantic predicate and a set of mapping relationships to the syntactic predicate (lexical form). Lcs is therefore the open portion of the lexicon in which synchronic vocabulary growth, of the type described in chapter 2, takes place, and it must therefore be implemented with a dynamic data structure like the one Prolog systems provide for facts asserted at run-time.

Operatives Region, OPR

The second region in the lexical hierarchy is Operatives, abbreviated Opr. This region contains entries for FUNCTION WORDS, words that have predominantly marking, structure building or deictic functions, but do not contribute predicates directly to sentence semantics. Operatives must be single morphemes, and they cannot be created by word formation rules. The Operatives region is therefore closed. Typical Operatives are non-predicative prepositions, like the English passive marker *by*, auxiliaries, pronouns, expletives and other particles.[18] Although minimal lexical predicates are allowed in Opr, no mapping from semantic to lexical forms is permitted. Thus, pronouns can be described in Opr with the conventional LFG lexical form (↑PRED) = 'PRO', since 'PRO' has no argument structure and no conceptual description, but a PRED value that could require any kind of explicit or implicit mapping relation (as in adjectives or event nouns) is forbidden. On many accounts, simple nouns, too, have an implicit argument, although the exact distinction from pronouns is probably fluid. To illustrate, *that* is clearly a demonstrative pronoun without specific predicative content, but for noun phrases like *a first*, *the older (of the two)*, *a winner*, *a conductor*, the deictic function seems to give way progressively to specific conceptual structure, implying specific properties and specific though implicit role relations to other concepts, that must be mapped to syntax by syntactic devices like the pronoun *by* in *the novel by Jane Austen*.

The stipulation that operatives must be monomorphemic follows from the way morphological structure is reflected in the structure of the lexical regions postulated in KLU, but as an empirical matter, this relation has not been exhaustively investigated. One finds at any rate in English, German and French that argument-marking prepositions like the passive markers *by*, German *von/durch*, French *par*, are typically monomorphemic, and that poly-morphemic prepositions like *by virtue of*, German *in bezug auf* 'referring to', French *à cause de* 'by virtue of' appear always to be attached to semantic predicates.

Derivational morphemes have also been permitted as a kind of morphological operator within the Opr region, although it is evident that they differ from syntactic operators in important ways, the most important being that they are bound morphemes and not directly insertable into syntax. Unlike syntactic operators, it appears necessary to assign lexical predicates to derivational morphemes; however, these predicates differ from those found in Lcs in two ways. First, they cannot be quantified or referred to by anaphoric or other means; hence, there is no reason to think that they are instantiated or introduce discourse objects. In the KLU notation, the value of a derivational predicate is therefore written without quotation marks (cf. the lexical fragment for *fatina*

[18]The distinction between case marking and predicative prepositions is discussed in detail in [Sch92], and is summarized in [May96a]. The non-predicative character of auxiliaries is demonstrated in [Sch99].

(104 d)). A second apparent characteristic of derivational predicates is that they appear to be limited to a single argument. Studies of Italian derivations, e. g., [Sch97a], have not uncovered dyadic or triadic derivational morphemes for Italian, and more than one argument in a word constituent would in fact be surprising. It would require the existence of specific syntactic functions at the level of word structure, which would be necessary to ensure unambiguous assignment of word constituents to the argument structure of the derivational morpheme. However, we see no evidence for syntactic functions within word-structure. Case marking as a means of argument assignment is of course missing, and positional marking appears to be absent because the position of a base is fixed relative to the derivational morpheme; we do not find cases where a base can be placed both in front of and after the derivational morpheme in order to distinguish two derivational meanings. (cf. Grammatical Function Impoverishment, p. 31. Anderson [And92, 37-42] presents similar conclusions based on a survey of other languages.) A consequence is that mapping is unnecessary, since there are no syntactic functions to which semantic argument positions need to be assigned. Mapping is purely a relation between semantic argument structure and grammatical functions in the Lcs region of the lexicon, and the existence of mapping relations in the lexical entries of the Lcs region is therefore one of the distinguishing differences between Lcs and Operatives. This means that a derivational operative has a kind of PRED feature, but, in contrast to an Lcs entry, this predicate involves no mapping relation, which prohibits the kind of connection between syntactic and semantic structure defined for lexical forms in f-structure. Thus, a derivational predicate cannot belong to a well-formed f-structure. In word structure, by contrast, there are no mapping relations; the value of the derivational PRED *is* its semantics rather than a lexical form pointing to a semantics held elsewhere. Interestingly, the division between Lcs and Operators in KLU is quite similar to the lexical structure that Caramazza et al. [CMSL85] inferred from certain acquired dislexias (p. 91). In the KLU implementation, derivational morphemes carry an attribute DPRED rather than PRED, but this is simply an implementational convenience.

At the interface to phonology, derivational operatives are assumed to lie at the intermediate level (or levels) of segmentation, which helps the segmentation algorithm restrict its search during the analysis of complex words. Since syntactic operatives are monomorphemic, they will never be found in segmentable structures and can be excluded from segmentation and lexical phonology altogether. Therefore, segmentation can save additional time by checking the Operatives region for whole-word matches before attempting any segmentation (more on this topic in chapter 6).

5.8.4 Lexical and Grammatical Words in the Static Lexicon

A further feature that the lexical regions Lcs and Operatives share is that they both impose an internal distinction between grammatical and lexical words. In the KLU implementation the distinction between grammatical and lexical words is enforced by the compiler, which permits only grammatical word categories to be mentioned in syntactic rules, and these category names must start with a capital, as in Noun or Verb.

In the Operatives region, the distinction between grammatical and lexical words is simple. Unbound Operatives are grammatical words like pronouns and case-marking prepositions that can be directly inserted into sentence syntax. Bound, hence non-

grammatical, Operatives are derivational morphemes that enter only into word formation. In Lcs, however, the distinction is a bit subtler. A grammatical word is defined as one that is syntactically well-formed according to a number of criteria that were discussed in section 5.6.1.

5.8.5 Morphotactics Region, MRPH

Morphotactics is the name given to a KLU module that contains, partly for convenience, both word-formation rules and inflectional morphemes. Its abbreviation is Mrph. Like derivational morphemes, inflectional morphemes are bound and can only be described as lexical, since they never enter c-structure directly; however, semantically they are so impoverished that their lexical entries can contain no predicates. This stricture requires close attention to the properties of the affixes in cases where they may have a derivational function as well. Attaching the feminine noun inflection *-a* to Italian *magino* 'little (male) magician' for example, has semantic as well as syntactic consequences, because *magina* can only be 'little female magician'. This introduces the possibility of inferences like MOTHER-OF and deletes the possible inference FATHER-OF for *magino*. Attaching *-a* to an adjective, however, has no consequences not already implied by the adjective's argument. Thus, the feminine inflection for Italian nouns has a derivational aspect, and it would probably need to be described as an Operator, whereas the *-a* that attaches to adjectives would require an additional, separate entry as an inflectional suffix in the Morphotactics region.

As a practical matter, the inflectional lexicon was combined into a single module with the set of word formation rules in KLU because neither set is likely to be very large by itself, and because they are so closely intermeshed. One might doubt, as some linguists have, whether it is correct to describe inflectional affixes as lexical items at all, since it is only their derivational counterparts, like the noun suffix *-a*, that contribute information to the sentence, and since their only function seems to be that of helping build up f-structure by adding to lexical words features for agreement among grammatical words, like number, tense, mode, etc. Because the set of inflectional morphemes is closed and has no semantics, i.e., no argument structures, and because the word formation rules are on the same side of the syntax-lexicon barrier as these morphemes, one could write the word formation rules in a form that simply stated the morphemes as symbols in the rules, without explicit lexical entries.

On the other hand, inflectional affixes, unlike constituent categories, have phonological reality; they are learned and can be identified by theoretically unspoiled speakers in a way that constituent categories cannot. Again following [Jac97], it is probably a mistake to see the lexicon as a list of tightly knit, uniformly structured entities. The absence of argument structure and meaning is not proof that an item is not in the lexicon of a language.

5.8.6 Syntax Region, STX

Syntactic rules for a language are defined in the KLU model as annotated production rules along the lines defined in [KB95], and as illustrated briefly in (94), p. 163. Departing from the standard LFG notation, the KLU compiler enforces the distinction between grammatical and lexical words by insisting that category names for lexical

words begin with a small letter, while the category names for syntactic words begin with a capital. The syntax region inherits the names of grammatical functions and their values, grammatical categories, and lexical forms from Lcs and Opr.

5.8.7 Allomorphy Region, ALM

The Allomorphy Region contains transducers implementing context-sensitive orthographic rules of the sort described in section 5.2.3. The rules must be translated by hand to individual transducers, and the transducers must finally be combined to a single transducer by combining the necessary states and transitions. An algorithm for compiling transducers from rules is given in [DKK+87], but in fact the grammars that have been written for KLU have not required large transducers, and the segmentation algorithm, described in the next chapter, is meant to reduce the amount of phonology in the grammar to a minimum.

This is accomplished by dividing the orthographic rules into three distinct sets corresponding to the morphological levels proposed by Kiparsky in [Kip82b]. The first set of rules deals mainly with the orthography of inflection and highly productive derivation (Kiparsky's level 3), the second deals mainly with less productive derivation (level 2), and the third can include rules for umlaut and ablaut (level 1). Each level has its own transducer, which means that it is not necessary to combine the entire rule system into a single transducer. In accord with Kiparsky's Elsewhere Principle, the segmentation algorithm always gives lexical entries precedence before orthographic rules. This means that the lexicon writer can avoid formulating complex rules simply by placing highly irregular variations in the lexicon.

Additionally, each transition in a transducer table can separately specify an inflectional class, allowing the grammar writer to restrict an orthographic rule to a single inflectional class, if so desired.

5.8.8 Indexical Structure

Previous discussions of word analysis have tacitly assumed that word parsing can proceed from a string of morphemic units like *libr at a*. During compilation of the lexical and morphotactic modules, KLU constructs a further module, called Indexical Structure, that contains exactly those sub-strings of words that can correspond to the morpheme units required for word analysis. Indexical structure plays a crucial role in the low level processing that makes dual representation of complex words possible, and it accounts for the role of pseudo-affixes in recognition.

Each lexical access string in each of the modules Lexical Conceptual Structure, Operatives, and Morphotactics can lead to one or more index strings in Indexical Structure. For example, the lexical entries for *thing* and its plural marker *s* in (100) produce index strings "thing" and "s". These index strings allow the KLU segmentation processor to divide an input word "things" into two substrings "thing" and "s" and then to identify the corresponding lexical entries. If the lexicon writer wishes to prevent morphological decomposition of a complex word, he or she can write its access form in the lexicon as a single word between slash marks. That is, creating an entry with an access form /things/ will prevent the input "things" from being segmented into "thing" + "s", even though it contains recognizable morpheme units. This precludes dual rep-

resentation of *things*, i. e., accessing both the full-form representation *things* and its morphological decomposition *thing s*. It amounts to a linguistic claim that speakers no longer make active use of the individual morphemes within the word, as is undoubtedly the case for many Greek and Latinate derivatives in English (like *rejuvenate*). On the other hand, to retain the possibility of dual representation of a lexical item, the lexicon writer must indicate possible internal segment boundaries with the symbol °. This is most obviously necessary when a complex word has a lexicalized meaning that must co-exist with the possibility of morphological analysis, in order to account for non-lexicalized meanings, as was shown for German *abwandern* (68), p. 132. The lexicalized meaning 'depart' can be placed in a lexical entry with an access form written as /ab°wand/; this allows an input string "abwand" to reference this lexical entry, but it also presents the morphemes *ab* and *wand* separately to morphotactic analysis. If both *ab* and *wand* have compatible lexical entries, a word structure and semantic analysis can be carried out to derive a non-lexicalized meaning such as 'hike off'.

A segmentation boundary in the access form is not equivalent to a morphemic boundary, however. It merely licenses a *possible* morpheme boundary; whether the segments actually have morphemic status depends on whether or not they are represented separately in the lexicon. Segmentation of a word leads to on-line morphological analysis only when all found segments correspond to existing lexical words (stems or roots) and affixes. This does not have to be the case. If the lexicon writer indicates a segmentation boundary for a pseudo-affix like like *re* in *rejuvenate*, an input string "rejuvenate" will be segmented into "re" + "juvenate", and this pair of strings will immediate match the lexical access form /re°juvenate/, but it will not induce a morphological analysis because *juvenate* does not exist as a separate morpheme (cf. section 3.3). Dual representation is therefore not possible, despite the internal segment boundary. Morphologically, internal segment boundaries serve no purpose in such cases, but the data of Taft & Forster [TF75] indicate that such segmentation boundaries are often present even when they are not reflected in the semantic structure of a word. For matching keyboard input from the computer to lexical access forms, they provide no computational advantages. Inputs from the auditory and visual systems, or from speech and optical character recognition programs, are very often ambiguous, however. It is possible that matching the input to the lexical access form /re°juvenate/ via two separate strings will be faster than it would be via a single string "rejuvenate" because it allows earlier decisions about the set of correct recognition alternatives. [19] The use of pseudo-prefixes may also help to optimize the coding density of a language, as was explained in section 2.6.2.

Indexical Structure is not explicitly provided in any of the cognitive models that have been discussed previously. There are ample empirical reasons for assuming separate semantic, syntactic, and access form representations in the lexicon. But apart from the still controversial data on pseudo-prefixes, there are no statistical, experimental, or linguistic data that require a fourth module which represents morphological strings in a form distinct from those directly associated with lexical items. Nevertheless, it appears nearly impossible to specify a detailed account of word recognition and analysis

[19] Section 6.2 will show that on average the ambiguity increases exponentially in the length of the ambiguously recognized input string; reducing a long string to short substrings can thus greatly reduce the number of lexical look-ups required to identify the word.

without such a module, and such a module explains compactly how pseudo-affixes and dual representation might function in recognition. Indexical Structure could only be discovered from the effort to integrate a wide range of requirements into a functioning computational model, and it may in fact be the single most important innovation resulting from the KLU implementation effort. How Indexical Structure is used during segmentation will be described in the next chapter; the Prolog data structure that implements it is described in chapter 7. Other arguments favoring the existence of Indexical Structure have to do with its role in word segmentation, and these will be presented in the next chapter.

5.8.9 Transparency and Opacity

An interesting consequence of a set of insertion rules like (85), p. 153, is that they allow a given noun to be inserted twice. The first insertion simply substitutes the item from lexical storage, while the second insertion can analyze the same noun by combining its individual morphemic segments. So long as both insertions lead to the same results, the duplication is descriptively correct and leads only to an annoying repetition. But we have seen cases in chapter 4, like the German verb *abwandern* 'hike away, depart' (section 4.4.3) and the Italian neologism *sbobinare* 'transcript dictation, de-spool', (section 4.4.1, p. 122) that in fact require two (or more) alternative semantic descriptions. To explain the availability of both senses of e.g., *sbobinare*, the KLU model specifies the possibility of dual representation for complex words. Both a lexicalized semantics and a semantic representation derived on-line can be presented to sentence analysis. Whether both senses should be made available at once, or whether the second should be available only later, after a kind of back-tracking, is an interesting issue but one that would need to be empirically decided. In any case, the model predicts that both meanings become available at times set by the recognition latencies, which are an inverse logarithmic function of the frequencies of the directly inserted word, in the first case, and of its constituent morphemes in the second.

5.8.10 Lexical Insertion and Buffering

To solve the problems raised by Lexical Integrity, the KLU model separates the rule systems describing sentence structure from those that describe word structure. The word structure rules are not an extension of the syntactic system, but they can be activated when the lexicon does not contain the required item. The separation is aided by a distinction that must be maintained in the lexicon and in the rule systems between grammatical words and lexical words (cf. section 5.4). In the KLU program, this distinction is enforced by a lexical insertion procedure. Each terminal category of sentence syntax, like Noun, Preposition, etc. is realized as a call to a complex procedure for lexical insertion, which attempts to unify the syntactic category with one or more symbols in the input stream. This procedure implements the interface from the syntactic system to the word-formation system and thus represents KLU's most significant departure from other architectures that have been based on unification formalisms. The complexity of lexical insertion arises from the diversity of the items that terminal symbols of syntax can describe: these can be single, atomic lexical items, but they can also be collocations consisting of several input words, bracketed strings of

morphemes listed in the lexicon or merely 'virtual words' permitted by the rules of word formation. Furthermore, a given terminal symbol can sometimes be unified in more than one way with the input stream. For this reason, the various modes of lexical access must be seen as operating in parallel and offering all possible unifications to syntax for evaluation.

An additional task of lexical insertion is word buffering. While we can think of syntactic operations largely as involving local storage in combinatorial operations, lexical access involves fetching complex data structures from the vast storage area that is the mental lexicon. From times that subjects in word recognition experiments require to determine whether or not a morpheme is in their mental lexicon, we can be sure that considerable time, tens of milliseconds, is consumed after the phonological or visual form is analyzed and presented to the lexicon. The information from the lexicon must, however, unify with information that is often not yet present in the input stream and may not appear for some time. For example, in *Try to look at least one word up*, the intransitive verb *look* cannot be excluded as a candidate for insertion into syntax until *word* appears, and the sentence's predicate, LOOK-UP, cannot be determined with certainty until *up* appears. This seems to require that the information required for both the intransitive and the transitive readings of *look*, fetched with the initial lexical access, must remain in some sort of short-term working storage until the end of the sentence is reached; only at this point can the irrelevant information be thrown away.

Thus, like many kinds of storage access devices, it seems probable that lexical insertion is outfitted with a buffer so that it does not have to initiate a word access more than once during sentence processing. In KLU a "Temporary concepts" region of the lexicon, TpLex, is meant to represent such a buffer. In the operation of a purely unification-based linguistic description, the presence of such a buffer would, of course, have no formal consequences; it would be merely an implementation device meant to speed the calculations by keeping fetched data easily available, like the cache storage of a computer. We know that the human memory system is not so exact as that of the computer, however; data moved from short-term to longer-term memory can be simplified or altered, which means that we cannot be sure of retrieving the same data from the buffer as we put into it. This, of course, suggests a possible cause of changes that occur in the course of lexicalization.

5.9 Derivation and Error Handing

The formal derivations presented in section 5.6 indicated that the formal apparatus of LFG could be used to describe a large part of what happens in word derivation. However, this apparatus cannot be used to specify how newly coined words are inserted into sentence structure because the word structures it creates must be discarded before lexical insertion, and the syntactic apparatus of LFG is defined as having access only to a static lexicon. This has lead some investigators, like S. Pulman [Pul96, 76], to believe that LFG cannot formally account for the introduction of spontaneously derived words.

It has been suggested a number of times that the interface from word formation to syntax must be mediated by something like an error handling mechanism. Yet it is a curious fact that none of the widely used linguistic or logic formalisms, including LFG, provides an explicit notation for error handling. There is nothing in the concept

that resists formal specification, although error handling does introduce a procedural element that computational linguists have resisted for reasons mentioned in section 5.3. Most modern programming languages, including those with a declarative bias like Prolog, do however provide sets of primitives to implement error detection and error recovery, and large application programs typically devote a significant portion of their code to error handling, as pointed out in section 1.6. From the perspective of software engineering, intelligent error recovery is in fact often the single most important characteristic distinguishing a valid application from one that is merely correct.

I venture here that the paradoxes which have haunted investigations of Lexical Integrity and of the simultaneous availability of opaque and transparent senses of complex words are not paradoxical at all. They are rather an artifact of the formal systems in which most theoretical linguists have attempted to describe these paradoxes, and in fact they demonstrate excellently how commitment to a formal system can inhibit the correct perception of a modeling requirement. Common sense suggests that when we do not immediately know the meaning of a word, we try to figure it out, by whatever means we can find. Our syntax analysis does not simply crash; we fetch a dictionary, ask someone what the word means, or try to figure it out by linguistic or non-linguistic means. Readers of the public-relations slogan "I ♡ New York" must stop at ♡ and use their extra-linguistic intelligence to find something associated with the icon that could be inserted into the syntax under construction. They then return to the linguistic system and let the analysis run further. Yet this very simple transfer of control has no equivalent in any of the formalization tools discussed so far.

Similarly, word analysis can stop and generate an error when it encounters a morphotactic anomaly, such as a phrasal constituent appearing within a word. Compounds like *ate-too-much-headache* ((35), p. 119) or *c'est-la-vie-attitude* suggest that often-heard phrases become 'listemes' in the hearer's lexicon and are thus available for insertion as constituents into words, without any analysis by rules of phrasal syntax. But phrasal constituents can be processed even when they are unlikely to be listed. To analyze a compound like *ate-too-much-prosciutto-headache*, word processing can turn to syntactic knowledge to make sense of the anomalous constituent *ate-too-much-prosciutto*. But this action is not itself an automatism built into the linguistic system. As pointed out in a footnote, page 121, word constituents can appear as objects that can hardly be present in any linguistic system, e. g., ♡*-less villain* or χ^2-test. Our ability to process such items requires more than linguistic intelligence to handle the morphotactic error; and even in the case of a phrasal constituent, syntactic analysis of the phrase occurs only after a higher level of intelligence has realized that morphotactic processing might be able to provide the item being sought to fill the place of a word-level constituent.

5.9.1 Error Handling at Lexical Insertion

In the KLU model, a higher level of error handling intelligence has been formulated using an error-handling "environment" of the sort that has long been used in functional and block-structured programming languages like LISP and Ada. (The Prolog II+ code is displayed in section 7.5.5.) In KLU the entire lexical insertion procedure is enclosed in such an environment. Each candidate for lexical insertion to syntax is checked for unifiability with the surrounding syntactic structure, i. e., for word-grammaticality in

the sense of Schwarze [Sch99]. If the check fails, lexical insertion is aborted, an error is raised, and the ill-formed string or icon is passed to an error handler that tries to resolve the object into a syntactically acceptable constituent along the lines described in section 5.6. (For ease of implementation, the KLU insertion error is generated simply by checking for the presence of a DPRED function in the word's attribute-value structure, rather than by examining specific well-formedness conditions individually.) If this attempt succeeds, a new lexical item is placed in a lexical buffer, and lexical insertion is called again. Since this second attempt to insert the word will now encounter a well-formed lexical item in the buffer, syntactic analysis can continue. If it fails, a further error is raised, and higher-level error handlers must deal with the anomaly. In a Pascal-like pseudo-code:

```
{Unify a syntactic terminal category S with a word candidate s}
procedure: LexicalInsertion(S, s)
  catcherror
    On Error: RepairWord(S, s)
    Unify(S, s)
  end catcherror{End LexicalInsertion}
procedure: RepairWord(S, s)
  Abort(LexicalInsertion(S, s))
  if Abbreviation(s) then
    ExpandString(s, t)
    Buffer(s, t); LexicalInsertion(t)
  else if Icon(s) then
    IconToString(s, t)
    Buffer(s, t); LexicalInsertion(t)
  else if IncompleteMapping(s) then
    Derivation(s, t)
    Buffer(s, t); LexicalInsertion(t)
  else
    RaiseError(s)
  end if
  Restart(LexicalInsertion(S, s)) {End RepairWord}
```

5.9.2 Lexicalization of Buffered Words

The most important operation, from the standpoint of the modeling problem posed by derivational morphology, is the operation Buffer(*s*, *t*). Formally it is not necessary, since to let the syntactic analysis proceed, the error handler needs only to pass the new item back to lexical insertion, but allocating space for a new item in a lexical buffer goes a long way toward explaining many difficulties raised in other models of derivational processing, such as the Meta-Model of Schreuder & Baayen [SB95], described in section 3.9.3.

Possibly the most comprehensive psychological model in the current literature, the Meta-Model proposes to model lexicalization by attaching to lexical representations statistical weightings, like the lexical strengths assumed by Bybee [Byb95] (cf. section

5.1), that can change dynamically. As was noted in section 3.9.3, what the Meta-Model does not explain is how frequently occurring combinations of morphemes come to be recognized as single lexical entries. As long as the constituent morphemes of a complex word retain high activation levels, they will be recognized quickly and will favor analysis of the constituents at the expense of whole-word recognition. The only possible escape from this dilemma is to assume some additional factor that would favor recognition of the unanalyzed word despite the disadvantage imposed by its low initial lexical strength. In the KLU model this is achieved by ordering lexical search in such a way that the lexical insertion procedure favors entries in the lexical buffer. It is psychologically plausible to assume that recently created or recognized items have a high activation level that quickly decays to their inherent, stored level; for a newly derived word this level will be very low, but at the point where the error handler restarts lexical insertion, just after creation of the new item in the buffer, the activation will be high, and it will overshadow the lexical strengths of the component morphemes. In the KLU implementation, a similar effect is achieved by forcing lexical insertion always to check the buffer for a recently saved, non-lexicalized word before starting a morphotactic analysis.

After being heard or read repeatedly, a newly formed word will acquire a high residual activation of its own, and morphotactic analysis will only be activated when the lexicalized meaning is not compatible with the rest of the sentence. This can be seen as a shift from storage in the lexical buffer to storage in the permanent lexicon. The KLU model assumes that frequent accesses to the lexical buffer will increase the inherent lexical strength of the whole-word representation, while at the same time weakening the morphemic boundaries. This process is undoubtedly gradual, but in the KLU implementation it must be described by transferring the item from the lexical buffer to the permanent static lexicon. At some point, the item's morphemic boundaries may even cease to be recognized, word segmentation will no longer take place, and the item may be fetched as a single symbol directly from the static lexicon. In other cases, we have seen evidence (cf. section 3.3) that even when a morphemic segment like *re-* in *repertoire* has lost all connection to a semantic representation of it own, its morphemic boundaries can remain intact for some early stage of recognition. The distinct roles of the morpheme as a semantic entity and as a unit of perceptual recognition require a more careful examination of the problems involved in word segmentation, however, to which I turn in the next chapter.

Chapter 6

Segmentation

An important requirement for any model of word processing is that during recognition the processor must be able to identify boundaries separating words and word constituents. The writing conventions of modern European languages conveniently mark word boundaries with empty spaces, but the morpheme boundaries within words are generally not marked, which means that these must be identified by some means before subunits can be compared to a morphological lexicon. Continuous speech often appears to disregard word boundaries altogether, and prosodic structure can obscure morpheme boundaries by placing syllable boundaries between morpheme boundaries, e. g., for morphological analysis, *universal* must segmented as *univers al*, but it is probably syllabified as *univer.sal*. Nevertheless, several studies of processing and perception of metrical structure and suprasegmental phonology have hypothesized that intonation, stress, and syllabic structure are used to identify possible word boundaries in speech.[1]

Alphabetically written language, on the other hand, provides few clues of any kind about morpheme boundaries within a word. Yet as the psychological studies surveyed in chapter 2 show, it is unlikely that the cognitive system is only able to recognize whole words, and in any case some languages, like Finnish and Turkish, present too many distinct forms to be held in memory. S. Pinker [Pin95, 128] reports that verbs in Kivunjo, a Bantu language, have segments marking focus, subject agreement, tense, object agreement, benefactive agreement, and aspect, in addition to the verb's root segment. Multiplied out, these give about half a million possible forms for *each* verb in the language. Hence, the apparatus that processes written languages with such rich morphological structure must also provide some means of finding segment boundaries not marked in the surface representation.

In many cases there is more than one possible morphological segmentation of a word, and when individual phonemes or characters are not recognized with certainty, the complexity of word segmentation and recognition can become formidable. It is all the more astonishing, therefore, that many of the segmentation ambiguities that arise in recognition can only be resolved by relatively late stages of grammatical processing, e. g., at the syntactic or semantic level. Any consideration of segmentation must therefore ask what strategies the cognitive system might employ to keep the combinatorial

[1] This point is defended persuasively by D. Norris et al. in [NMC95]. J. Meinschaefer [Mei98] gives an overview of a number of studies pointing in the same direction.

explosion of possible inputs under control. For various reasons that will be developed in this chapter, the KLU model proposes that word recognition tries to avoid morphological segmentation as far as possible, and that where several segments must be identified, the recognition strategy assumes they are likely to be found at the periphery of the word. The model further assumes that segmentation is based on a hierarchical word structure of the sort that has been postulated in studies of lexical phonology. KLU's segmentation is steered not by phonological or orthographic rules, however, but directly by the lexicon, using a list of possible word segments stored in the Indexical Structure that was introduced in section 5.8.8, p. 200.

6.1 Why Segmentation is a Problem

Morphological analysis of any kind presupposes a method for finding segment boundaries within words and of matching these to lexical items to verify that the segments correspond to known lexical entities. Under the influence of phonological or orthographical rules, however, valid segments may differ from those stored in the lexicon. To segment *impolitic*, for example, the segment *im* might have to be recognized as a variant of *in*, in which the consonant /n/ has assimilated to the following /p/. In German comparatives like *länger*, a vowel of the root can undergo umlaut under the influence of the comparative suffix *er*, and this must be reversed before the stem *lang* can be found in the lexicon. In some languages such effects can extend over several segments, and the application of phonological rules can be intimately intertwined with the segmentation. S. Anderson cites as an example the Finnish word *karahka* 'stick' which appears as *karahkoja* in the partitive plural form. Three interdependent phonological rules must be applied before the constituent morphemes can be identified in the lexicon; comprehensive phonological systems can require up to fifteen sequential rule applications of this kind [And92, 378-379].

(132)

surface form:	/karahko + j + a/
undo glide formation, i → j:	/karahko + i + a/
undo t → Ø, applied after a weak syllable before a short vowel:	/karahko + i + ta/
undo a → o before i:	/karahka + i + ta/
lexical morphemic string, 'stick' + 'plural' + 'partitive':	/karahka/ + /i/ + /ta/

The effort to find computational devices capable of carrying out such analyses led to the development of MONOSTRATAL systems in the 1980's, in which the phonological rules were implemented in systems of transducers, as described in section 5.2.3. Instead of producing several intermediate forms in the course of a derivation, as in (132), K. Koskenniemi [Kos83] showed that a system of mutually constraining, parallel transducers could mediate directly between the surface form on one side and the lexicon on the other side, producing a "two-level" morphology without intermediate derivations. Every match of a surface form to the lexicon produced a lexical root and a set of features associated with inflectional and derivational features found in the surface form. It was quickly found, however, that mapping between an unsegmented surface form

and the lexicon was still too complex a task to be implemented efficiently, and subsequent elaborations of two-level morphological analysis broke the lexical matching step into a series of matches. Instead of matching the entire surface form, an initial substring, corresponding, say, to a derivational morpheme, is paired against a subset of the lexicon. The lexicon is partitioned into "continuation classes", and each such substring in the lexicon is marked with the the class or morphological category or categories that may follow it [Kar83], [DKK+87]. The classes thus correspond roughly to morphological constituent categories. The analyzer in essence carries out a kind of morphotactic analysis of the word in addition to the phonological analysis. The output consists, again, of the root plus a list of the features associated with each of the derivational and inflectional morphemes, but without constituent structure. While this is quite adequate for analyses of inflection and some simple forms of derivations (e. g., the diminutive derivation of *fatina* 'little fairy' discussed in section 5.4.3), the meanings of more complex derivations, like that of *librata* 'blow with a book', (cf. section 5.6.2, p. 188), cannot be expressed as unstructured sets of attributes collected from the constituent morphemes. Instead, a word grammar must identify derivational predicates and their arguments, and it must construct the meaning by a form of recursive functional evaluation. To steer this evaluation, morphological analysis must build up a more complex, hierarchical attribute-value structure.

Computational approaches that explicitly utilize morphological structure to constrain phonological mappings were devised as COMBINED PHONOLOGICAL-MORPHOLOGICAL ANALYZERS by Bear [Bea86] and Trost [Tro91]. These approaches interleave phonological mapping with morphological constituent analysis in systems of mutual constraints, so that the applicability of phonological rules can be restricted to substrings within the word. Like the monostratal systems, these systems do not explicitly acknowledge that phonological rules applied within the root of a word have a much different status than those applied at the periphery. As was noted in section 5.2.3, speakers of English know that some verbs like *hang* have past tense forms in which the vowel /a/ appears as /u/, as in *hung*. However, this is not equivalent to saying that speakers analyze *hung* by reducing /u/ to /a/ + PAST. Because the pattern /a/ to /u/ is hard to predict (*stand* becomes *stood*; *drag* becomes *dragged*), speakers must learn to associate this pattern with just those verbs that exhibit the change /a/ to /u/, and they are sometimes uncertain which verbs these are. By analogy they can recognize *drug* as a possible past tense form of *drag* (cf. section 5.1), but the phonological rule does not tell them whether or not this form is generally accepted. Thus, there are reasons to think that phonological changes in the root or stem of a complex word are usually stored in the lexicon and rarely need to be generated or analyzed by on-line word processing. At the periphery of the word, however, where inflectional and some derivational affixes appear, phonological changes are more predictable, and can be invoked in on-line processing.

Unlike the mono-stratal morphology, segmentation and phonology based on a stratified or LEVEL-ORDERED system of phonological rules is able to spare much effort by taking advantage of structural principles common to all words. Like monostratal analyzers with continuation classes, it can also constrain the search for possible segments by recognizing that affixes tend to appear at the periphery, and roots near the middle of a word. Algorithmically, this approach is potentially more complex than either two-level morphology using continuation classes or mutually constraining

phonological-morphological analysis, but in practice its run-time complexity is probably lower than either. Like the mono-stratal analysis, KLU's level-ordered algorithm first identifies a linear sequence of lexical substrings, but it takes advantage of level-ordering to restrict the search for most word segments to the periphery of the word. A second stage of analysis constructs a constituent structure for the word, based on a word grammar, as in the combined phonological-morphological approach. Unlike its predecessors, KLU's MORPHOTACTIC ANALYSIS, described in the previous chapter, section 5.4, strictly separates morphological constituent analysis of words from phonology and segmentation. The next paragraphs will examine some of the linguistic and psychological data that support the validity of an approach based on level-ordering with a strict separation of segmentation from morphological analysis.

6.1.1 Segmentation vs. Morphological Analysis

Psycholinguistic research has not emphasized the role that a separate segmentation processor might play in word recognition. In the opinion of Drews [Dre89], a stumbling block in the psycholinguistic research on morphology has been the psychologists' fixation on stimulus-response data, which undoubtedly obscures many relations that might be accessible if linguistically informed models had been used to guide the investigations. Drews calls particular attention to the confusion between morphological analysis, which in linguistics has a structural and semantic dimension, and simple segmentation.

> ...einige empirische Ergebnisse [scheinen] schon eher mit der Annahme eines von links-nach-rechts operierenden Parsings vereinbar zu sein, der eine visuell präsentierte Wortform, angefangen vom ersten Graphem, solange mit internen Repräsentationen vergleicht, bis ein adäquater Eintrag gefunden ist. Der Vorteil dieser Hypothese liegt darin, daß sie keine Berücksichtigung wortinterner Strukturen durch einen morphologischen Dekompositionsmechanismus verlangt, sondern von lexikalischen Einträgen ausgehen kann, in denen sowohl gebundene als auch freie Stamm-Morpheme oder komplette Wortformen repräsentiert sind [Dre89, 72].[2]

If, on the other hand, recognition were based on simultaneous morphological analysis and segmentation, we would expect to find effects from derivational affixation and inflection in all experimental paradigms. However, as we have seen in chapter 3, some effects, like those from pseudo-prefixes, are sometimes evident only in early stages of word recognition. In visual recognition, early recognition effects like those found for the derivational pseudo-prefix *re* in *reproach* by Taft [Taf79a] (section 3.3, p. 74) have not been found at later stages of recognition. Instead, in the cross-modal priming of W. Marslen-Wilson et al. [MWTWO94], which can only be sensitive to later stages of recognition of the visual target (cf. section 3.6.3), only genuine derivational morphemes showed an effect.

[2] ...some empirical results [appear] more compatible with the assumption of left-to-right parsing, in which, starting with the first character, a visually presented word is compared to internal representations until an adequately matching entry is found. The advantage of this hypothesis is that it requires no mechanism to discover word-internal structures via morphological decomposition; rather, it can work with a lexicon that includes both bound and free stem morphemes, as well as complete word forms.

For spoken word recognition, the evidence for a segmentation step separate from morphological analysis is weaker. The data from gating studies like those used by Marslen-Wilson and his collaborators to support the cohort model of word recognition, e. g., in [MWZ89], are often interpreted as showing that word recognition and segmentation coincide (cf. Gating, p. 72). However, a study by Taft provides a finding that seems to contradict this implication of the cohort model. If spoken words were presented directly to the lexicon as strings of phonemes, one would expect them to be recognized at precisely the point where they were uniquely identifiable, i. e., at their uniqueness points. Thus, both *tension* and *intention* have their UPs at the phoneme /ʃ/ because of competitors like *tent* and *intensive*, and they should both be recognized at this point. Taft [Taf88] found, however, that *intention* is recognized sooner, which seemed to indicate that the prefix was being identified separately and was somehow speeding recognition of the rest of the word (cf. section 3.6.2). The segment *in*, however, is not a morpheme here. While in other words like *include*, *in* demonstrably contributes to word meaning, exchanging *intention* for *extension* (in speech) does not produce the kind of related change in meaning seen in the contrast *include – exclude*. It appears that at an early stage of recognition *in* is being recognized separately, as if it could be a morphemic unit, even in cases where it is demonstrably not a morpheme and could not participate in morphological analysis. In other words, it appears that the segment "in" is identified as a separate unit in blind preparation for a later morphological analysis. For formations that clearly require morphological analysis, say *inadmission*, this step is unavoidable; in the case of *intention* it at least does no harm. [3]

If segmentation is taking place even for segments that do not represent morphemes within the word, one might ask whether segments, as distinct from meaningful morphemes, have any lexical status at all. The segment boundaries might be found by a procedure that simply looks at transition probabilities between phonemes or characters, without consulting the lexicon at all. It might place segment boundaries at points where the transition to the next character or phoneme is unlikely to occur within a morpheme. This could explain the segment boundary between "in" and "tention", because in many cases the transition from "in" to a following consonant is in fact a morpheme boundary, as for *incapable*, *inside*, or *include*. Asking the same question, Taft & Forster [TF76] did not, however, find that the rejection of a non-word containing some lexical material was aided by the presence at a segment boundary of a letter transition that would not occur within a word or morpheme, such as the transition from "t" to "m" in **footmilge* (cf. p. 75). They concluded that segmentation is entirely lexically controlled, even when the segments only have morphological status in words other than the one being recognized.

In sum, the experimental evidence appears to show that word segments are identified and processed even when they are not morphemes, but that segments are identified by their similarity to genuine morphemes in the lexicon.

[3] It should be noted that pseudo-prefixation effects like these may not be visible in all languages. Burani et al. [BDTL97] have argued that the more pronounced experimental evidence for morphological analysis of derivatives in Italian by comparison with English has to do with the relative lack of pseudo-prefixed words in Italian. This does not exclude the possibility that segmentation precedes morphological analysis in Italian. Rather, most of the early recognition effects associated with blind segmentation would again be evident in later stages of recognition, simply because most segments are in fact morphemes (cf. section 3.5.5).

6.2 Uncertainties in Segmentation

Although there are adequate reasons for assuming that segment boundaries are determined independently of morphological analysis, the computational problems posed by BLIND SEGMENTATION are overwhelming. Without some knowledge of word structure, the segmentation algorithm is forced to search the entire word representation for all possible affixes in the lexicon, at all positions. Blind segmentation will rarely find a single, unique set of segment boundaries, and it will not be in a position to assign the found segments to a possible word structure. Further, some segments may not belong to the word at all. In written Italian, for example, clitics are concatenated with some verb forms, and in connected speech clitics of many languages merge seamlessly with their hosts.

6.2.1 Recognition Uncertainties

In spoken or visual word recognition the problem is compounded by the uncertain nature of the word's representation. The sounds or shapes presented to the earliest stages of recognition will often be ambiguously interpreted. Given that each uncertain phonetic segment in the auditory input may correspond to phonemes in several different and unrelated lexical representations, the task of matching an input string to a lexical entry explodes in complexity with the length of the string. Assuming — optimistically — that each phonetic segment is on average two-ways ambiguous, i.e., is interpretable as two phonemes, for one segment there will be two possible ways of entering the lexicon, for two segments there will be four, for three, eight and so on. Let us assume that, in context, the sounds in "did" are easily confused as follows: [d] = [n], [i] = [e], [d] = [t]. Then the recognition of "did" will require comparing all of the following phonemic strings against the lexicon:

(133) "did" $\longrightarrow$ [d] or [n] + [i] or [e] + [d] or [t]
$\longrightarrow$ /did/, /dit/, /ded/, /det/
/nid/, /nit/, /ned/, /net/

Among the auditory recognition alternatives for *did* fancifully postulated here, *dead*, *knit*, *Ned* and *net* are all entries in the English lexicon that would need to be fetched and compared to the input for a best match.

Recognition ambiguity thus increases exponentially in the length of the string. Additional ambiguity can arise if the lexicon does not explicitly list the surface forms of all allomorphic or orthographic variants. A maximally compact lexicon will contain a single citation form for each lexical item, and variants will be mapped to their surface forms by chains of phonological or orthographic rules which distinguish each variant from the base form. If we construct a lexicon in which a number of morphologically related forms like /do/, /does/, /done/ are stored as a single string, a surface form like /did/ would be stored as /do/ + PAST. *Each* of the alternative recognition candidates in (133) would have to be passed through the phonological component in order to match the candidate /did/ to a possible underlying /do/. While, as Kaplan & Kay [KK94] have demonstrated, this phonological mapping can be implemented as a finite-state transducer that requires time only linear or polynomial in the length of the string, phonological rules are often not unambiguously reversible. Each of the recognition

alternatives could therefore produce more than one possible abstract lexical look-up. For a string of length n, the number of lexical searches N will thus be roughly

(134) $$N = (PR)^n,$$

where P is the average ambiguity of the phonological rules and R is the ambiguity of the phoneme (or character) recognition component. If speech were to be recognized in this way, for all but the shortest words lexical lookup would need to be exceedingly fast, and one would expect the speed with which words can be recognized to decrease drastically with increasing word length — an effect that simply is not observed. (Cf. section 3.2.1, p. 69.)

By dividing the word into segments, the exponential growth in the number of look-up candidates can be restrained. For a two-ways ambiguous recognition results like (133), a 6-character word generates $2^6 = 64$ search alternatives that have to be fetched and compared to the input, but segments of 4 and 2 characters lead only to $16+4 = 20$ alternatives. Assuming that the fetched alternatives can be buffered long enough to avoid refetches, a search for a best match can be carried out within each segment, and with high probability the sequence of best matches will match the lexical entry (cf. [SHC85]). Thus, it would appear advantageous to store words in an internally segmented form, so that lexical matching could take place by searching for smaller segments — and in fact, long monomorphemic, i.e., morphologically unsegmentable, words are relatively rare, and it even seems doubtful that long, monomorphemic words such as *Vladivostok* could be stored in the English lexicon in an unsegmented form.

Unrestricted segmentation, however, presents problems of its own. The number of possible segment boundaries in a string of length n is $n-1$, and the number of possible segmentations is the sum of the binomial coefficients for m, with $m = n-1$.

(135) $$\binom{m}{0} + \binom{m}{1} + \binom{m}{2} + \ldots + \binom{m}{m} = 2^m$$

For a word of length 4 there are $1+3+3+1 = 8$ possible segmentations; for $n = 10$, there are 512. Because of the exponential growth in the number of possible segmentations, unrestricted segmentation would again drastically slow the search times for longer inputs. The existence of transparent neologisms as well as the priming effect that word constituents produce suggest that the access mechanism takes advantage of segmentation of both transparent and opaque derivatives and compounds to help find optimal lexical matches quickly. On the other hand, a search strategy that segmented beyond the morpheme level (to individual characters or phonemes) would require that access units be recombined at a later point in order to obtain access to the morpheme entries that psycholinguistic results appear to uncover in the lexicon.

6.2.2 Clitics

Clitic particles attached to words are an additional source of ambiguity in segmentation. From the perspective of phonology, M. Nespor and I. Vogel [NV86] have argued that clitics deserve to be regarded as word segments; and within syntax, clitics have properties that other categories generally do not have with regard to their linear order. In French, for example, the clitic group in front of the verb is ordered by rules that cannot be neatly expressed as rules of constituency. Nevertheless, in the KLU model

an arbitrary decision was made not to regard clictics as incorporated word constituents but rather to strip them from the word structure and present them to syntax, which must define their possible positions and syntactic functions as best it can.

Separating clitic particles from their host is potentially a much more difficult problem than is at first apparent. Starting from the end of the word (and/or beginning, depending on the language), segmentation must split any clitic particles from their host so that they are represented to syntax as separate words. Sometimes this step is not unambiguous, as for Italian *calmati*. As an imperative form meaning 'calm yourself',*calmati* has an attached clitic pronoun *ti*, which would need to be separated before syntactic analysis could recognize the verb and its object. However, *calmati* can also be a past participle meaning 'calmed', with a masculine plural inflection. Italian in fact presents a large inventory of such ambiguities, describedby R. Delmonte in [Del93]. Because segmentation has no access to the grammatical context needed to disambiguate these cases, it must place all alternative segmentations in a list that replaces the original, single string in the sentence. Hence, *calmati* must be presented to morphotactic analysis as a list of lists. Each nested list is a possible word formation, except that the pronoun *ti* is not presented as an input to word analysis.

(136) "calmati" $\longrightarrow$ [[/calm/ /at/ /i/], [/calm/ /a/] /ti/]

6.3 Segmentation Based on Level-Ordering

The previous section has presented some reasons for thinking that word analysis keeps segmentation and phonology separate from genuine morphological analysis. It has argued that segmentation is lexically controlled, in the sense that the search for segments is carried out by matching possible substrings to strings in the lexicon rather than by searching for combinations of characters or phonemes that mark segmentation boundaries. Keeping segmentation completely separate from morphological analysis seems to miss something, however. We may recognize *in* as a segment at the beginning of a word, even when it has no morphological status, but it seems highly unlikely that *in* will be identified as a segment within the root of a word like *combination*, giving a segmentation like *comb in ation*. On the other hand, it is to be expected that word analysis will try to segment a nonsense word like *inbinker* into *in bink er*, but not into *inb in ker*. In other words, it seems reasonable to expect that segments corresponding to possible affixes will be isolated at the beginning and end of the word, but not in the middle. Naive, mono-stratal segmentation without a continuation lexicon or other means of constraining lexical search, would be forced to search for such segments everywhere. But these examples suggest that segmentation does make use of some strategy to limit the search ranges for various classes of segments to specific areas of word structure. Morphological analysis would be one such possible strategy, but morphological analysis requires morphemes, which of necessity have meanings. Segmentation is evidently oblivious to meaning, as has been shown experimentally by the segmentation effects found with pseudo-prefixed words like *intention* and *rejuvenate*. A segmentation algorithm thus requires a strategy for finding a word structure that is not tied to the meaning structure of the word.

Such an approach is implicit in the theory of lexical phonology formulated by P. Kiparsky [Kip82a], [Kip82b] and others. Using examples from English, Kiparsky ar-

gues that the phonology of complex words displays a layered structure, building on a lexical root that undergoes phonological changes and affixation in three layers. New lexical material in the form of affixes is added only at the ends of the word, and rules of phonology do not, in general, apply over the whole word. Instead, the scope of a phonological rule tends to be restricted to a single layer of affixation, and inner layers are not visible to rules of outer layers. In other words, the rules apply only within a given level of affixation, and once all affixes and phonological changes have applied to the level, the internal structure of the layer becomes invisible to higher layers (Kiparsky's Bracketing Erasure Convention). A final set of postlexical phonological rules then applies to the word as a whole. Most importantly for segmentation, affixes can be assigned to layers on the basis of their phonological properties. English affixes that force changes in stress, e. g., *-ous, -ity*, can only appear at level 1, whereas phonologically unrestricted affixes like *non-* and *-ist* tend to appear at level 2. Most unrestricted are the inflectional affixes like *-s, -ed,* and *-t* at level 3. The distribution of an affix within the phonological levels of word structure is unrelated to the affix's semantic function. The stress-changing derivational suffix *-ity*, for example, belongs to level 1, while the phonologically neutral but semantically similar suffix *-ness* is assigned to level 2. This means that the level-ordering distribution of affixes (or pseudo-affixes) within a word can be specified or at least circumscribed independently of morphological structure. By referring to something like phonological structure and the corresponding level assignments of affixes, segmentation can therefore limit its search for various classes of segments, without making reference to semantics or to a morphological structure that is tied to semantics.

Affixes are apparently marked in the lexicon for the level-ordering context in which they may apply, which restricts their distribution within the word. This means that inflection (level 3) cannot appear inside of derivation (levels 2 and 1), and a level 2 affix must lie closer to the word's periphery than a level 1 affix. Kiparsky [Kip82b] illustrates this pattern with constructions like $[[[Marx_{\,0}]ian_1]ism_2]$, where *-ian* as a level 1 prefix may precede the level 2 suffix *-ism*, but where $*[[[Marx_0]ism_2]ian_1]$ violates the level ordering sequence.

Kiparsky made two further important observations that can have consequences for a word analysis algorithm. One is that, like the derivational affixes found at level 1, the phonological rules applying at level 1 tend to be idiosyncratic and unproductive, whereas at the outer levels affixes and phonological changes apply more systematically. This suggests that speakers have little active access to either the phonological or word formation rules on the lowest levels. The second observation (Kiparsky's Elsewhere Condition) is stated as a rule about phonological rules: the less general the formulation of a rule (the more context dependent it is), the higher its precedence of application, and if two rules produce effects that cannot be unified, the rule with higher precedence wins. The lexical entry itself can be considered to be a kind of rule that has precedence higher than all rules of phonology — in other words, the lexical entry can, in principle, arbitrarily overrule all rules of the phonology.

6.3.1 Experimental Evidence of Level-Ordering

As a purely descriptive hypothesis, level-ordering may help to describe speakers' implicit knowledge of their language, but it does not necessarily have consequences for

a cognitive model of word processing. Psycholinguistic experiments have, however, given empirical evidence of level-ordering in word processing. Some experiments with young children have born out the prediction that irregular plurals (at level 1) can be the input to compounding (at level 2), but that these language learners do not produce compounds containing regular plurals (formed at level 3 and thus not available as inputs to level 2). Presenting children with unfamiliar puppets said to like to eat various kinds of other creatures, Gordon [Gor85] found that his subjects avoided incorporating regular plurals (even overregularized, like *mouses*) in compounds of the form X-*eater*, but the children readily produced compounds with irregular plurals, e. g., *mice-eater*. Pluralia tantum forms, like *clothes-eater* appeared, but *glasses* and *scissors* were often reduced to the singular, e. g., *glass-eater*, mainly by younger children. Gordon noted that Dutch compounds like *muizenvanger* 'mice-trap' appear to contradict this principle, but he hypothesized that the existence of inflectional classes in Dutch may require speakers to store the plural as a semi-lexicalized, level 1 formation. This suspicion was confirmed by Clahsen et al. [CMBW96], who found that German children select one of the available plural formation patterns as a default rule for building plurals of unfamiliar words (58.5 per cent select *-s*; 26.2 per cent, *-e*; 14.6 per cent, *-(e)n*, 0.8 per cent, *-er*). They also avoid building compounds containing plurals built on the default pattern they have selected. In other words, it appears that the German children accept plurals as inputs to compounding so long as the plural form is somehow present in the lexicon; if their internalized word grammar lets them build the plural freely, without special lexical marking, only the singular form will be used in the compound. This confirms the prediction that genuinely rule-based plural formation must follow compounding.

Another important constraint imposed by the theory of level ordering is that predictable inflection and derivational markings tend to appear at the periphery of the word, relegating the lexical content to the innermost segment. Recall that Bradley [Bra80], section 3.8, found a clear stem-frequency effect for suffixes that are statistically productive, like *-ness*, *-ment*, and *-er*, but the effect was less evident for relatively unproductive suffixes like *-ion* [Bra80]. These affixes, which level-ordering assigns for phonological reasons to an intermediate level of word structure, are evidently bound more tightly to their roots, so that the roots by themselves are not strongly activated.

6.3.2 The Challenge of Templatic Morphology

A cognitive model has to distinguish the architecture of the language faculty from the structure of specific languages that it can process. A segmentation strategy based on level ordering makes strong assumptions not only about the phonological and segmental structure of a language, but also about cognitive mechanisms involved in word recognition. The validity of these assumptions would be seriously compromised by any language that exhibited inflection or highly productive derivation in the middle of the word. The templatic morphology of Semitic languages presents such a challenge.

Like irregular conjugations and declinations in Germanic, Semitic languages exhibit morphological patterns in which variation (derivation and some forms of inflection) appears to take place not at the periphery of the root but in a very pronounced way in central segments. Both noun and and verb roots are typically formed around a skeleton of consonants, and morphologically related forms arise by varying the vow-

els. For example, in her study of word formation in modern Hebrew, O. Bat-El [BE89] observes that the verb *gadal* 'he grew' can be changed to the verb *gidel* 'he raised' or the noun *gódel* 'size' merely by replacing vowels within the consonantal skeleton *g-d-l* [BE89, 8]. Causative verbs are related to inchoatives in a somewhat similar fashion, e.g., *katav* 'he wrote' becomes *hixtiv* 'he dictated' ('cause to write') [BE89, 19].

The crucial question is, however, whether such patterns of relatedness among words represent actively used word formation rules. Superficially it appears that the consonantal skeleton is a root, identifying the semantics of a word, with the vowels adding additional semantic properties in a compositional way. If this were the case, it would be a strong argument against level-ordering. Bat-El [BE89] concludes, however, that templatic morphology does not contribute to the freely usable word formation patterns. In the case of the causative derivation (*katav* $\longrightarrow$ *hixtiv*), she observes that many verbs built on the pattern of *hixtiv* are not derived from any other verbs. The only semantic generalization possible is that when such pairs are found, it is the verb built on *hixtiv* that is the causative. The examples she cites support the conclusion that "the [consonantal] root does not always have a consistent and plain meaning, and therefore every stem must be considered individually, and this can be achieved only under the word-based view", which does not see the consonantal skeleton and the vowels as separate, morpheme-like elements [BE89, 20]. Noun formation is even less systematic in modern Hebrew, although again there is an inventory of fixed consonantal patterns that can be identified to some extent with semantic classes, like place or collective entity. There are a number of patterns that can be used to derive nouns from verbs, e.g., *kicer* 'he shortened' leads to *kicur* 'shortening'. Nevertheless, the patterns are inconsistent, and for some verbal nouns there is no corresponding verb, indicating that these nouns are not the products of active derivation [BE89, 24].

Thus, templatic morphology does not appear to contradict the thesis that derivations at level 1 are inactive relics of older morphological processes that have become lexically fixed. When they are used to form new words, it appears that this is more likely to be a process of deliberate imitation rather than of spontaneous production.

6.4 The Roles of Lexical and Indexical Structure

The advantage of performing segmentation before morphotactic analysis is that segments — as opposed to morphemes — can be characterized independently of word semantics, and their distribution in the word can be characterized by rules independent of the word's morphological structure. A level-odered theory of lexical phonology can provide the clues necessary for finding possible segment boundaries. Nevertheless, some constraints postulated in lexical phonology require considerable knowledge about the word's phonological structure, like the requirement that the English comparative suffix *-er* attaches only to bases of one or two syllables. Using level-ordered morpho-phonology to steer segmentation would thus require us to postulate an additional stage of analysis, prior to segmentation, to obtain stress and intonation contours, even during fast, silent reading. Clearly it would be advantageous if the search for segments were steered not by phonological structure directly but by information that was already available, prior to the start of the segmental analysis. Such information is available in the layered structure of the lexicon.

As chapter 2 has argued, the lexicon must be structured in some way that places the frequently used, low-information inflectional affixes at one, quickly accessible end of its structure, and the low-frequency, high-information content words at the other, less easily accessed end. The affixes which level-ordering assigns to level 3 are just the inflectional affixes found in KLU's Morphotactics region, the affixes of levels 2 and 1 are by and large the derivational morphemes found in KLU's Operatives region, [4] and the underived roots of level 0 are found only in Lexical Conceptual Structure. To the extent that the hierarchy of lexical regions roughly mirrors the phonological grouping of affixes, it could help segmentation restrict the search for affixes at each of the segmentation levels, without the effort of matching their phonological properties to the phonological structure of the word. Minor discrepancies would not matter, since lexical structure is being used only to restrict the number of segmentations that must be explored at each level of analysis, not to obtain the actual word structure.

A major problem for a lexically controlled segmentation strategy is the number of accesses it must make to lexical storage. Especially in the early stages, segmentation will be checking a large number of recognition candidates to see whether they can be found in the region associated with each of the phonological levels. Fortunately, segmentation needs relatively little information about a hypothesized segment. It does not need the syntactic category, the lexical form or the subcategorization information that is stored along with the access form of a syntactic entry. In fact, it does not even need full access forms when they contain internal segments, like *re°sale* or *re°pertoire*, since internally segmented forms can be matched segment-wise instead of word-wise. What segmentation does need is a quickly accessible compilation of all word segments found in the access forms of the lexicon, devoid of the baggage that only morphology and syntax will require. The KLU model postulates that such a data structure, called Indexical Structure, exists separately from the lexical regions (cf. section 5.8.8).

While I assume that the lexical index structure contains mainly orthographic (or phonological) information, it probably needs to make a small amount of additional information available. To limit the search space for segment candidates, the segment indices need to be grouped according to lexical region. Lexical region is perhaps not the only principle that is invoked to limit the search space for segments, however. The inflectional classes of Latin or of German, for example, can be used in some cases as a kind of preliminary filter to break off the investigation of segmentation alternatives that do not belong to the same class. Consider the segmentation of a German verb formed with the participle-forming circumfix *ge- ... -t*. When the left-hand search finds an initial segment *ge-*, it can note the inflectional class of the morpheme. The right-hand search can then reject all segment candidates that do not belong to the same class as *ge-*. Because the inflectional class is an atomic data type limited to a small range of values, it adds little to the quantity of information that must be fetched from the lexicon during segmentation, yet it permits a very general kind of syntactic constraint among the proposed segments. In their model of affix processing, Badecker & Caramazza [BC89] propose a similar sort of access index table which includes gender markers in addition to inflectional class (cf. section 3.7.3).

[4] Aditi Lahiri has pointed out to me that is doubtful the distinction between levels 2 and 1 can be found in all languages (personal communication). The distinction seems to result from the peculiar mixture of Romance and Germanic morphology present in English.

A grammar writer using the KLU program does not write an index table. Instead, the lexicon compiler gathers a list of all segments appearing in the access forms of each of the three lexical modules described in section 5.8.1, and it appends this list to the compiled lexicon module. Rather than searching the lexical entries themselves, which are comparatively large data structures, the segmentation algorithm searches only the table or tables likely to contain strings belonging to a given level of word structure. The tables contain compact descriptions of lexical segments. For atomic access forms, like /that/ or /the/, the tables will contain strings identical to the access forms. For complex words containing internal segment boundaries, like /re°sale/ but also /re°pertoire/, the tables contain only the individual segments, not the full access forms, e. g.,/re/, /sale/ and /pertoire/, but nerither /resale/ nor /repertoire/. The tables are distinguished by the name space modularization used to encapsulate other elements of the KLU system (cf. section 5.8). (The Prolog representation of the index lists is displayed in chapter 7.)

The KLU model assumes that the index tables include not only the lexical segments but also their inflectional class, as well as the lexical category, the lexical symbol and the lexical form of each word with which they are associated. Segmentation, as mentioned above, can check the compatibility of affixes with their bases to break off segmentation attempts at an early point. Segmentation's use of the index tables may also explain early priming effects like those reported by Taft and Forster [TF75] and Taft [Taf79b] for pseudo-affixed pairs like *reproach* and *approach* (cf. section 3.6.1, p. 85). While these words do not share any elements at the level of morphemic structure, it is possible that their access strings are internally segmented. In the KLU lexicon, their access forms would be written as /re°proach/ and /ap°proach/. If the lexical entries are written this way, the KLU index tables will contain lexical segments /re/, /ap/ and /proach/. Recognition of both words takes place by identifying the segments and matching the list of segments directly to the list in the lexical entry's access form (i. e., without reference to any rule of word formation). Therefore, if /proach/ has been activated during the recognition of *reproach*, it can prime recognition of /proach/ when *approach* is segmented into /ap/ + /proach/. The model thus explains why *reproach* primes the recognition of *approach* without being required to list either /ap/ or /proach/ as a morpheme in the lexicon.

Of course, neither of these segmentations leads to morphological analysis, since neither the pseudo-prefix /re/ nor /ap/ nor /proach/ is a morpheme. Instead, the chains /re/+/proach/ and /ap/+/proach/ match directly to the lexical access forms /re°proach/ and /ap°proach/ of the corresponding entries.[5]

6.5 The Role of Allomorphy

Corresponding to each of the lexical strata, the KLU model provides a set of orthographic rules that map surface segments to lexical items. Thus, like segment search, the scope of orthographic rules in KLU is controlled by the lexical regions rather than

[5]The productive derivational morpheme *re-* will be represented in the index table as the string "re" with a morphological class. The pseudo-prefix *re-* is represented by the same string, but its entry in the index table has no morphological class. The prefix *ap-* occurs in a large number of opaque Latinate derivatives, but it is not productive and can probably not be assigned a well-defined meaning. It would therefore be represented as a pseudo-prefix, with no corresponding, independent entry in the lexicon.

by the history of a phonological derivation. If a grammar writer were to introduce rules that had genuinely phonological level-ordering dependencies, then the rule system would have to be expanded to discover the phonological bracketing independently of segmentation.

No attempts have been made to write large phonological rule systems with the KLU program, which makes it hard to say at this point whether lexical control can be used effectively in a segmentation algorithm. In any case, a working assumption of the KLU model is that the majority of phonological rules are not involved in on-line processing. Especially at the level of roots and stems, it is to be expected that variant forms are found in the lexicon, and that during segmentation they will be matched directly. Hence, for most purposes, there is probably seldom any reason for constructing the lexicon so abstractly that it requires rules operating below the post-lexical level, since both the linguistic and the experimental evidence suggest that variation at lower levels is stored rather than being computed on-line. For the German non-default plurals like those investigated by Clahsen et al. [CMBW96] (cf. section 6.3.1) the lexicon would therefore contain distinct stems for the root and umlaut forms; e. g., for *Maus* 'mouse' and its plural *Mäuse* the lexicon ought not to contain /mAus/ with a umlaut rule applying to "A" , but rather /maus/ 'mouse' and /mäus/ as separate stems. This approach appears to mirror the knowledge that speakers actually use to analyze such word formation patterns, and it helps to make lexically controlled segmentation computationally tractable.

Nevertheless, as was argued in section 5.2.3, speakers can produce and understand non-lexicalized forms on the basis of phonological rules belonging to lower-levels, as when *drug* is interpreted as a past tense form of *drag* by analogy to the pattern *hang – hung* (cf. section 5.1, p. 151). A comprehensive morphological description of a language would therefore need to include its seldom used phonological rules, implemented as low probability paths through the transducer graph. During recognition, this would have the effect that other interpretations of the string would be dealt with first. Using probabilistic word recognition, lexical insertion would first return *drug* as a noun with a high degree of recognition confidence. Because the ablaut rule for "a" to "u" has such a low probability of use in production, in recognition the recognizer would return *drug* as a past tense verb with a very low degree of recognition certainty. Probabilistic weighting of segmentation alternatives has not been implemented in the KLU program, but the required elaboration appears to be straightforward.

6.6 The Segmentation Algorithm

The KLU model allows for altogether four levels of orthographic rules and segmentation. The entire sentence is submitted to suprasegmental rules; the output of these rules is then the input to word segmentation. The first stage of segmentation separates clitics, using a special list of orthographic clitics declared in the region Morphotactics. The second stage searches for inflectional morphemes in the index table for Morphotactics; the third stage searches for derivational morphemes defined in Operatives. The final stage must find a root segment in Lexical Conceptual Structure. At each stage of segmentation, a candidate segment from the input string is submitted to orthographic rules, implemented as a transducer in the KLU module Allomorphy, specific to a lex-

ical region. The output of the rules is a string that is compared directly to the index table for the region. If a segment is found, a mark is set and the remainder of the word is investigated.

The orthographic rules simply mediate between candidate substrings of the word and corresponding tables of the lexicon, except in the case of suprasegmental orthography, for which there is no lexical search. To gain an overview of how segmentation and orthography are interleaved, consider the case of a nonce word like *undirtied*. This word is synonymous with *cleaned*, so it is probably not lexicalized and must be analyzed to its root *dirty*. At each of the lexical layers a segment is found, but the root segment "dirti" is not in the lexicon. It must be mapped by an orthographic rule to the abstract lexical segment /dirtY/.

The analysis proceeds as in (137), with a string in quotation marks, "...", representing a string awaiting segmentation and lexical look-up, while a string between slashes, /.../, is a lexical segment found in an index table.

(137) a. Orthography, post-lexical: no changes
Segmentation at Morphotactics region
"undirtied" → ["undirti" /ed/]

b. Orthography, level 2: no changes
Segmentation at Operatives region
["undirti" /ed/] → [[/un/ "dirti"] /ed/]

c. Orthography, level 1: Y → i / __ e
"dirti" → "dirtY"
Lexical look-up at Semantics region
[[/un/ "dirtY"] /ed/] → [[/un/ [/dirtY/]] /ed/]

Note that this final bracketing will be erased before the string of lexical segments is presented to Morphotactics for morphological analysis.

Consider now how the candidate segments are being compared with the lexical segments in the index tables. In many cases the two forms can be identical, but as is evident for the candidate segment "dirti", sometimes orthographic rules will intervene to match surface and lexical forms. The orthographic rules are grouped according to the regions of the lexicon in which they apply.

Suprasegmental rules are provided to deal with characters like the apostrophe. When the French article *la* appears before a vowel, "a" is elided, giving *l'*. Rather than trying to reconstruct the missing vowel, the current KLU grammars simply concatenate the apostrophe to the previous word. The French singular definite article must therefore appear in the lexicon as *la*, *le*, and as *l'*. Another suprasegmental rule must map an upper case letter at the beginning of a sentence to lower case. The rule that recognizes the full stop, exclamation or question mark as the end of the sentence might also be seen as part of suprasegmental processing, although KLU grammars are forced to treat these marks simply as lexical items belonging to a special category in the grammar.

Post-clitic rules are defined, in principle, over a level of representation intermediate between words and sentence structure called the CLITIC GROUP, roughly as described by M. Nespor [Nes85]. This comprises a verb and its cliticized particles, and it is possible that some phonological rules are specific to this level. General rules with this scope cannot be defined in the KLU program because many clitics appear orthographically as separate words. Without some special apparatus for identifying the entire clitic group beforehand, rules specific to the clitic group cannot be applied. Written Italian, however, sometimes concatenates clitic pronouns to the end of a verb or participle, so at least in these strings orthographical rules for the clitic group can be applied before the clitics are separated from the verb. For its Italian grammar, the KLU program takes advantage of this convention and defines a layer of POST-CLITIC orthographic rules whose scope is exactly a word and any clitics that are immediately concatenated to it. The module Morphotactics must list explicitly which particles can be concatenated to their hosts in this way.

Post-lexical rules apply to an entire string after clitic particles have been stripped; they are not applied to the clitics, however. Most of the rules that have been implemented in KLU grammars are at this level; an example is the rule for Italian that often must insert "h" after "g" when "o" or "a" changes to "i" or "e". These rules are applied after clitics are stripped, and before affixes are separated from the word. Affixes are thus allowed to alter the stem via the post-lexical rules before they are separated (and thus made invisible to lower level rules).

Level 2 lexical rules apply to both the affixes and the stem of an inflected word, but separately. That is, their scope is such that a rule which affects the stem cannot specify a context lying within the affix, and vice versa.

Level 1 lexical rules apply to the root of a derived word. Their scope is such that a rule which affects the root cannot specify a context lying within the derivational affix, and vice versa. However, like the rules of other levels, they can be made contingent on m-structure features like inflectional class.

6.6.1 Bracketing Paradoxes

As segmentation and phonology work their ways through the levels of word structure, they build up a bracketing structure that is used to restrict the search for segments and the scope of the phonological rules, as in (137). This bracketing sometimes differs, however, from the bracketing that morphotactic rules will define. Such discrepancies have been called bracketing paradoxes in the morphological literature, and the theory of level-ordering has attempted to resolve them by distinguishing phonological from morphological structure.

An ambiguous bracketing can be found in a word like *unhappier*. The English comparative suffix *-er* attaches only to words of one or two syllables, but the prefix *un-* is indifferent to the prosody of its base. This apparently imposes a segmentation structure in which *-er* is added before *un-*, giving [un [[happy] er]]. If the meaning of the word were built up according to this bracketing, we would obtain

NOT(MORE(HAPPY)) as the semantics, but in fact unhappier means MORE(NOT(HAPPY)). This and similar examples suggest that the bracketing structure obtained by level-ordering is not directly related to the bracketing implicit in the word's morphological derivation, and that some sort of reanalysis may be required, as Kiparsky [Kip82b] has proposed. KLU's morphotactic analysis algorithm in fact simply discards the lexically conditioned bracketing obtained by segmentation. In any case, the ease with which bracketing paradoxes are resolved by this strategy would appear to be a further argument in favor of separating segmentation from morphological analysis.

In the KLU model, many of these bracketing paradoxes, like that of *unhappier*, do not surface explicitly because level 2 and level 1 derivational affixes are both stored in the region Operatives, and KLU's segmentation does not recognize them as belonging to separate bracketing levels. Hence, instead of producing the phonologically required bracketing

[/un/ [[/happY/] /er/]]

KLU's segmentation simply produces

[/un/ [/happY/] /er/].

Whichever bracketing segmentation furnishes, the rules of Morphotactics will in any case reorganize the unbracketed lexical items in a way that allows a correct derivation of the semantics, namely as

((UN (HAPPY)) ER).

6.7 Segmentation in Detail

To make the following, more detailed discussion of KLU's segmentation algorithm precise, it is necessary to define some terminology that I have used rather informally in the previous sections.

6.7.1 Terminology

(138) a. A SURFACE STRING is the unsegmented string of characters lying between two spaces in the input, roughly, a word of input text, written as e.g., "cookable". It can contain no abstract, underspecified characters, like the "Y" in the access form /dirtY/, but it can include concatenated clitics.

b. An ABSTRACT STRING is the representation of an input word or a portion of an input word after segmentation and orthographic analysis. It will not, however, necessarily be found in the lexicon. It can have the same form as the surface string, "cook" $\leftrightarrow$ "cook", or (as for Italian *fungo*, 'mushroom') it can contain abstract characters , e.g., "funG", corresponding to possible surface strings "fung" and "fungh". Like a surface string, it is identified by double quote marks.

c. A LEXICAL SEGMENT is an abstract string that has a representation in the lexicon but that is not yet specified for lexical region. "funGi" is an abstract string because it is an output of orthographic analysis, but it is not likely to be a lexical segment, if only the underspecified stem segment /funG/ appears in the lexicon. All known lexical segments are contained in the index tables. All segments passed on after segmentation are lexical segments, meaning that they are known to be in the lexicon, but it is not known to which region they belong. A lexical segment like /is/ is unique, but it may map to more than one distinct item in the lexicon, e. g., the main verb *is* and the auxilliary verb *is*.

d. A LEXICAL SYMBOL is the representation corresponding to a lexical segment found at a particular region in the lexicon and attached to a lexical entry. A single lexical segment can correspond to many lexical homonyms. For example, the verb *is* will be represented after segmentation as the lexical segment /is/. The corresponding main, copula verb, found in the Semantics region, is the symbol Lcs:/is/, but the auxiliary will be found in the Operatives region and is thus the symbol Opr:/is/. At segmentation, however, it is not possible to know whether sentence syntax will turn out to require a main verb or an auxiliary. Lexical segments can only be resolved into appropriate lexical symbols at the point of syntactic or morphological analysis. This may seem redundant at the word level, where segmentation is forced by level ordering to search for segments in restricted regions of the lexicon, but it is required by the bracketing reanalysis discussed earlier. It results from the facts that a segment can appear in the same form in different regions, and that segmentation only tries to identify segments, not lexical entries. Segmentation cannot correctly disambiguate lexical segments into lexical symbols. Instead, the lexical segments in the segmentation alternatives must be disambiguated by Morphotactics or at later stages by syntactic or semantic processing.

e. A LEXICAL CHAIN is a linked list of lexical symbols, in which each symbol carries a pointer to either the next symbol or an end mark. It can occur in the representation of a collocation, like *bed and board*, or in the representation of an internally segmented word, like *cook°able*. While both forms are lexical chains, the collocation has internal word boundaries and is represented as /bed and board/. An internally segmented word has only two word boundaries, but it has internal morpheme boundaries, as in /cook°able/.

6.7.2 Strings, Segments, Symbols

When a user of KLU types a sentence, blanks, tab and some punctuation characters are interpreted as boundary markers for word-like inputs strings that are the input to segmentation. Segmentation is thus a procedure that works through a sentence, represented as a list of surface strings. In some cases a string can actually represent a word with concatenated clitic particles. For each single surface string, segmentation returns a list of one or more segmentation alternatives in the output. Each segmentation alternative is in turn a list of lexical segments, i. e., references to lexical items that can

be grammatical words, lexical words, derivational affixes, inflectional affixes or parts of words in the region Lcs. It does not necessarily represent a list of morphemes (lexical symbols) awaiting morphological analysis: if the segments have been drawn from a monomorphemic word like *rejoice* that contains an internal segment boundary justified by its recognition behavior, its access form /re°joice/ will be matched to the lexical segments /re/ and /joice/ directly, without morphotactic analysis.

The segmentation algorithm is actually rather complex, since it requires several nested recursions. At the outermost level is the descent through layers of word structure correspoding to the lexical regions described above. At each stage, the algorithm first checks the regions Lcs and Opr for a corresponding unsegmented lexical access form, using the orthographic rules for the corresponding lexical region. This step in effect implements Kiparsky's Elsewhere Condition (cf. section 6.3) by giving the lexicon precedence before any rule-based analysis. If no full-form entry is found, the algorithm can start searching from both ends of the surface string for one or more lexical segments found in the index table for the region. It sets a segment boundary, initially to the left of the last character or to the right of the first character. Using a special sub-table in Opr, it first removes clitic segments from both ends of the surface string. Then, comparing candidate segments to entries in the index table for Mrph, the algorithm identifies possible inflectional affixes and places them in working storage. After a parameterizied limit (e. g., maximally two inflectioal affixes) is reached or no more candidate segments are found, level 3 is closed and the search continues in the index table for Opr (e. g., derivational morphemes), and finally in Lcs (for stems and roots).

When a lexical segment is found at the beginning or end of the string, the algorithm calls itself recursively, with the remaining surface string, the name of the next lexical region, as well as additional information like the inflectional class of any marked affixes, as arguments. If the final recursion to the region Lcs fails to find a lexical segment, the algorithm must backtrack and test the next alternative segmentation by moving segment markers toward the middle of the word. Segmentation keeps a record of its progress though the lexical regions, which produces a kind of bracketing structure for the word. This is not the level-ordering structure that lexical phonology would produce but merely the history of the segmentation analysis, since, strictly speaking, not level-ordering but lexical structure is used to steer the segmentation, and segmentation is used only to reduce the computational complexity of the segmentation task, not to analyze the word. (Morphological analysis is the responsibility of the word formation rules in the module Morphotactics.) Thus, while *run* as the root of a possible derivation like *runable* ought to be found at level 1, segmentation will not assign it to a specific bracketing level. However, *runable* will nest *run* inside a structure containing *able* because the two segments belong to distinct lexical levels.

6.7.3 Segmentation Example "uncorkers"

Let us take an analysis of the surface string "uncorkers", which I assume must be fully analyzed to the root *cork*, as an example. The algorithm, as shown in (139), proceeds recursively through the layers of word structure, as was described in the previous section.

(139) a. A surface string is mapped to an abstract string by the post-clitic transducer and segmented into either
A lexical segment in Lcs + any clitic particles. Finished.
ELSE
a post-clitic string S_p (not found in lexicon).
Return any clitic particles.

b. S_p is mapped to an abstract string by the post-lexical transducer and segmented into either
Lexical segments found only in Opr. Finished.
ELSE
Lexical segments in Lcs or Opr + any inflectional affixes. Finished.
ELSE
a post-lexical string S_1 (not found in lexicon).
Return any inflectional affixes

c. S_1 is mapped to an abstract string by the level 2 transducer and segmented into either
Segments in Lcs + any derivational affixes. Finished.
ELSE
a level 1 string S_0 (not found in lexicon).
Return any derivational affixes.

d. S_0 is mapped to an abstract string by the leval 1 transducer to a
Root in Lcs. Return the root.

e. OTHERWISE the segment boundary is rejected.
Move segment markers toward middle of the word.
Restart the search.

As shown in Fig. 6.1, the unsegmented surface string “uncorkers” is presented to a transducer implementing the post-lexical orthographic rules. The output of the transducer is an abstract string. It is checked against the index tables for a matching lexical segment in either Opr or Lcs, but in this case /uncorkers/ is not present, so segmentation must proceed to the next level. Starting from the right end of the string (or from the left, or from both ends — this is a language-specific parameter), first one, then two, then three and more characters are presented to the transducer for lexical level 2. The abstract string coming from the lexical side of the transducer, “s”, matches a lexical segment /Mrph:s/ in the index table for Morphotactics, which is inserted as an abstract symbol /s/ in the segmentation structure.[6] The search for further level 2 segments continues to the left, until a maximal number of level 2 segments (two for the current grammars), or no more segments are found. At the same time, each remaining abstract string, e. g., “uncorker”, “uncorke”, “uncork”, etc., is compared to the index for Lcs, on the assumption that it might be the stem of the inflected word. If a stem is found, processing is done.

[6]To help users testing a word grammar, segmentation documents where the abstract symbol was found by actually inserting the lexical symbol, e. g.,/Mrph:s/. Later, Morphotactics discards the region identity of a segment and replaces it with the identity of a symbol having the lexical attributes required by the word formation rules.

Since no stem can be found, the remaining surface string is passed to the next lower level of segmentation. All affixes found here will be nested inside the structure built to contain the stem of the level 2 (inflectional) affixes. The search for abstract segments begins from the right of the surface string “uncorker”. Again, up to two affix segments are permitted at the word’s periphery. This stage finds /Opr:er/ and places a further abstract symbol, /er/, in the segmentation hypothesis, with the root still not found. Then the search for level 1 prefixes is taken up from left to right. The surface string “u” is not found, but the level 2 transducer maps“un” is to a lexical symbol /Opr:un/ found in Operatives. Segmentation deposits the symbol /un/ in the structure and tries to find an item matching “cork”. /Lcs:cork/ is found, and /cork/ is placed at the lowest nesting level in the structure. The nesting structure contains a level of bracketing under the inflectional segment /s/; within this a further bracketing for the affixes /er/ and /un/, and yet a further bracketing for the root /cork/.

Since, as was explained above, the nesting that results from segmental analysis sometimes contradicts the nesting required by morphological structure, this nesting is discarded. It is needed to keep track of the recursion through the lexical levels, but it serves no further purpose. At this point morphological analysis of a linear list of specific lexical symbols can begin, as described in section 5.6.1.

6.7.4 Segmentation Ambiguities

The level-ordering hypothesis predicts that once a lexicalized segment is found, further segmentation should not be necessary. In fact, however, level ordering is not quite as effective as it might seem at first, because homonymous forms can require different segmentations, and segmentation has no way of knowing which form it is dealing with. As was shown in section 6.2.2, Italian imperatives with an attached clitic can be homonymous with a past participle, as for *calmati*. To deal with such cases, after it has created a segmentation alternative by stripping a clitic, the LKU segmentation algorithm automatically submits both the entire string and the substring without the clitic to further segmentation. At levels one and two, this is not done, although it is likely that this would be necessary for a large lexicon. A thorough investigation of the problem in the framework of our implementation would have required a large Italian lexicon and word grammar, which we did not have available, and it would need to be made in conjunction with other decisions about the internal representations of stems and roots. Given these uncertainties, some aspects of the algorithm can be modified by the grammar writer, by setting parameters in a KLU initialization file. The main escape from orthographic and segmentation problems, however, is to place difficult variants in the lexicon. If a lexicon writer decides, for example, that certain stems need no internal segmentation, then the stem of e. g., *servire* can be /servi/ rather than /serv°i/, and the segmentation algorithm can stop at level 2.

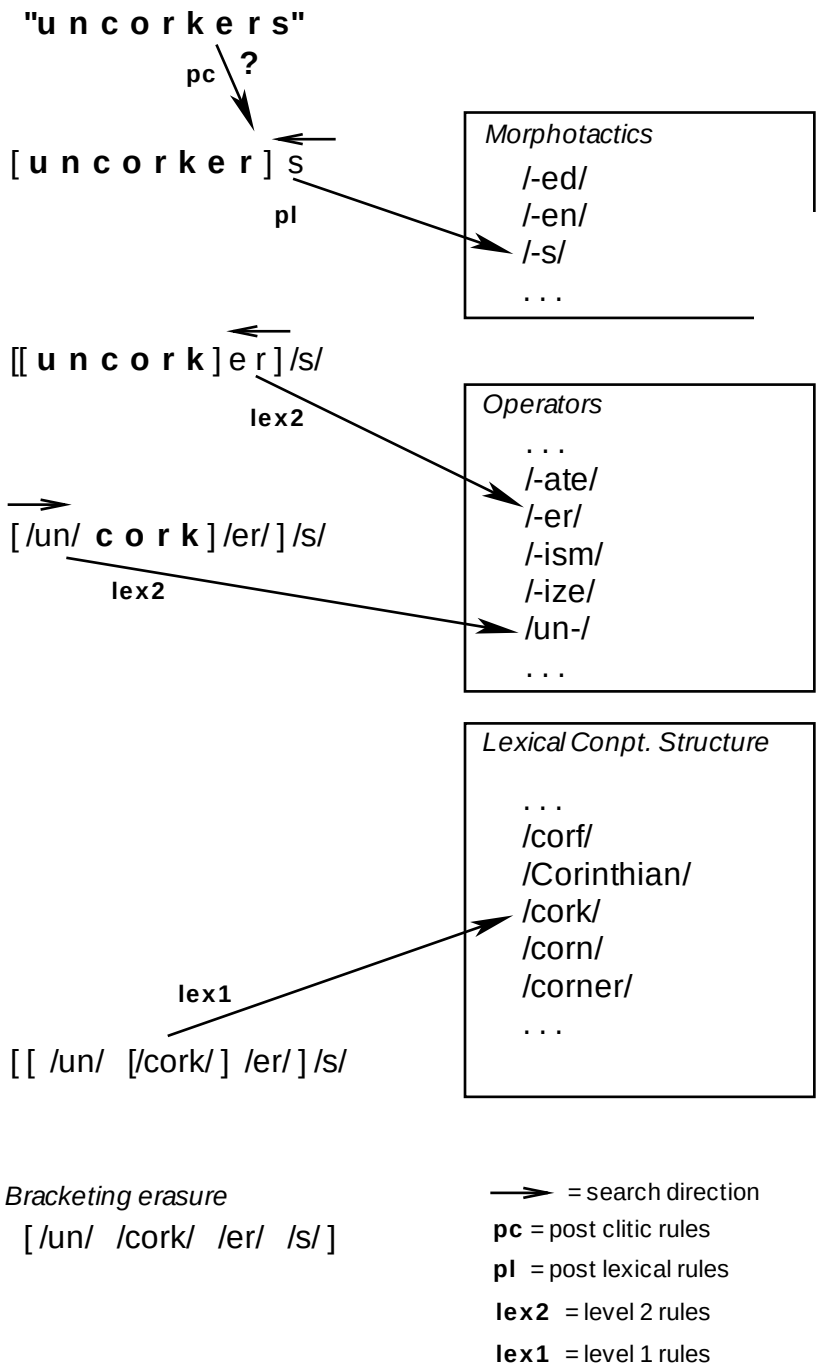


Figure 6.1: Stages of Segmentation in the KLU Model

Chapter 7

The KLU Implementation

Chapters 4, 5, and 6 have have attempted to give an account of how words are represented in memory, how they are introduced into discourse, and how they are analyzed, based on data about word formation drawn from statistical studies of text corpora, experimental data, and linguistic investigations of word structure. As a step toward a coherent formalization of this account, these chapters have presented sketches of specifications for various parts of a computational model of word formation and its interface to sentence structure. This model elaborates on the traditional factorization of the language processing system into lexicon and syntax. It represents the lexicon as a complexly structured system of modules, rules and controlled interfaces. It restricts the combinatory rule system traditionally called syntax to operations over grammatical words, which contain only fully specified inflectional attributes and argument mapping relations. Morphological relations are described by word formation rules, which do not interact directly with syntax. Instead, the results of word formation are passed to syntax via a lexical buffer, and syntax is shielded from anomalies in word formation by a lexical insertion procedure and by an error handler. Moreover, unlike syntax, word formation is often not invoked to process the objects it describes. Instead, complex words are often drawn directly from the lexicon. Morphology is actively invoked on a regular basis only to process inflectional affixes.

Many features of KLU's abstract model of word processing, described in the previous chapters and sketched in Fig. 5.2, have been implemented and tested as a Macintosh application program written in Prolog II+ [Pro96]. In addition to analyzing words, this program can analyze single sentences into discourse structures, represented as sets of Prolog facts inferred from the sentence and from a knowledge base. With its user interfaces, this program compiles from 557 kilobytes of Prolog source code. The largest French grammar that has been constructed for KLU, described in detail in [Sch96], contains an additional 74 kilobytes of material in the LFG and Prolog formalisms. Smaller Spanish and Italian grammars have also been prepared.[1] Derivational rules for Italian that have been implemented in KLU were presented in chapter 4. Specific features of the implemented KLU model are summarized in (140). The user interface and other program design issues have been described in [May96b].

[1]These grammars are largely due to Ch. Schwarze and V. Knüppel, with additional help from M-Th. Schepping, N. Schpak-Dolt and many students at the University of Konstanz who have used the KLU system.

(140) a. The overall structure of the system is highly modular. The implementation makes heavy use of a namespace mechanism of Prolog II+, described in section 5.8, that allows name resolution at both compile-time and run-time. Accordingly, the writer of a language description must break down the description into three lexical modules, plus a syntactic module, a morphotactic module, and a module for phonology and allomorphy. Each module of the description is represented in a text file in appropriate formalisms based on LFG (except for allmorphy). Each lexical module has a module-specific interface to syntax and lexical insertion. The modules are arranged in a precedence hierarchy, and the compiler enforces module dependencies by allowing the user to compile modules lower in the hierarchy only after superior modules have been compiled. A limited amount of compile-time checking is carried out to catch formal errors and potentially unbounded recursions. During grammar development, detailed traces of all derivations can be enabled, and constituent categories at all levels can be independently tested. When provided for in the grammar, neologisms can be processed within a sentence, producing a functional and semantic analysis of both the word and the matrix sentence.

b. The LEXICON comprises three 'region' modules. A fourth region, Cpt for Concepts, describes extra-linguistic knowledge. At least in theory, Cpt seldom needs to be edited because it does not vary from one language to another. Semantic, syntactic, and morpho-phonological attributes are all involved in assigning items to the lexical regions. Items from the three regions do not combine freely. Inflectional and derivational affixes combine mainly with content words, but not with each other alone, and infrequently with words lacking lexical semantic predicates. The largest region, containing 'content' words, is expandable; the other regions are closed. As described in chapter 6, word segmentation refers to the lexicon by region as it tries to identify possible segments of the word.

LEXICAL CONCEPTUAL STRUCTURE, Lcs is an open (expandable) module that contains semantic and syntactic entries linked by mapping relations. From the set of syntactic items it supplies grammatical words to a syntactic component and lexical words to a word formation component. The semantic entries are not supplied directly to syntax or word formation but are mapped to lexical forms in the syntactic entries. This allows two syntactic items like *car* and *automobile* to share a single lexical semantic representation. Lcs notation enforces a distinction between grammatical and lexical words.

OPERATORS, Opr is a closed module containing function words supplied to syntax and derivational morphemes supplied to word formation. It contains no mapped predicates, but it does contain deictic predicates (as in pronouns) and derivational predicates. Derivational morphemes are regarded as lexical items, since like the lexical items of Lcs, they can only be inserted as bound morphemes into sentence structure.

MORPHOTACTICS, MRPH is a closed module that supplies only inflectional

affixes to word formation. It contains no entries having predicates of any kind. It also contains the rules of inflection and word derivation, described below.

c. The COMBINATORY APPARATUS comprises two distinct components. Sentences are described by a syntax component; actively constructed words are described by a word formation component called morphotactics. Syntax and morphotactics are specified in a modified LFG formalism, introduced in section 5.3, and they are compiled to elaborated Definite Clause Grammars (DCGs), implemented in Prolog.

   SYNTAX, Stx, describes the rules of sentence structure. It receives grammatical words directly from Lcs and Opr. Complex words that have been morphologically analyzed do not come from the lexicon but from a lexical insertion procedure. Thus, syntax respects the principle of Lexical Integrity (cf. section 4.3.2) and receives no input directly by substitution from morphotactic rules. The output of syntax is an f-structure that is evaluated by a semantic processor.

   MORPHOTACTICS, Mrph, describes the rules of inflection and derivation. It receives its inputs as lexical strings after naive segmentation and allomorphy. These potentially ambiguous strings must be resolved by morphotatic rules into lexical words or morphemes from Lcs, derivational affixes from Opr, and inflectional affixes from Mrph. The output of word formation is a lexical entry that is placed in a lexical buffer attached to Lcs. If well-formed, the item is substituted directly to a pre-terminal node of syntax.

   ALLOMORPHY, Alm, describes allomorphic and orthographic variation. It receives inputs as strings separated by white spaces when the user types a sentence at the program console for testing or analysis. Together with Segmentation, which is controlled lexically and contains no user-defined rules, Allomorphy maps surface strings to lexical access forms.

d. LEXICAL INSERTION to syntax functions as a gate-keeper or barrier between the syntactic component and morphotactics. A syntactic rule cannot insert a lexical item directly from the lexicon or from word formation; instead, it calls lexical insertion to find the item, carry out any required morphological analysis, and ensure that it is a grammatically well-formed word. If after inflectional analysis the item cannot be accepted as a grammatical word, an error is raised, and an error handler must attempt to make sense of the input being presented to syntax (cf. section 5.9). This requiries a derivational step to generate a lexical item matching the required category. Lexical insertion is a crucial device for enforcing Lexical Integrity.

   Coupled with the lexical insertion procedure, the KLU model postulates a lexical buffer and a garbage collector to simulate the decay of memory traces of newly formed words. Frequently used, difficult derived words should be purged last, and infrequently used, simple inflected forms are purged first.

e. SYNTACTIC ENTRIES in the lexicon consist of access forms (as lexical symbols in the terminology of section 6.7) annotated with functional equations.

They contain no semantic information. In Lcs, the equations must include a PRED feature with a lexical form as its value, which points to a semantic entry containing the lexical semantics of the item. In the regions Opr and Mrph PRED is not permitted. DPRED features are required for bound (derivational) morphemes in Opr. The syntactic entries are specified in a modified LFG formalism and compiled to Prolog rules. The citation or access form of a lexical entry can contain internal morphological or pseudo-morphological segmentation, and it can contain a collocation (lexical chain) composed of more than one word.

For each lexical module the compiler produces an INDEX TABLE. The index table contains strings representing only atomic segments appearing in the lexical access forms; these are required by word segmentation. The index table also contains the inflectional class of the word from which the segment is taken and the associated lexical form. The lexical form in the index table is required during derivation.

f. SEMANTIC ENTRIES are permitted only in Lcs; derivational semantic rules are named but not specified in Opr (derivational operators are assumed to name purely conceptual operations, like 'find the corresponding feminine entity' or 'find the opposite'). They are specified as Horn clauses and compiled to Prolog rules. Mapping relations to corresponding syntactic entries are specified implicitly by the correspondence between thematic argument variables in the semantic entries and functional arguments in the lexical forms of syntactic entries having the same functor (cf. section 5.5.1). The semantic representation is thus not part of the data structure of its corresponding syntactic entry, and the two items are only loosely coupled. It is possible in principle to write semantic entries that have no matching syntactic entries.

g. The module SEGMENTATION is steered by index tables organized according to the modular structure of the lexicon (140 b) and by a set of level-ordered orthographic rules (cf. chapter 6). The orthographic rules are implemented by the user in the Allomorphy file, Alm, as state transition tables for transducers, as described in section 5.2.3. Each transition is written as a Prolog fact.

h. The module COGNITIVE CONCEPTS, CPT contains conceptual facts and relations expressed in Prolog. This module is merely a stand-in for an assumed knowledge representation system, and it will not be presented in this chapter. It is consulted to check conceptual attributes of arguments, to find conceptual predicates, and to find derivational predicates.

7.1 Overview of the KLU Implementation

The main components of the KLU program are a user interface, an LFG and Horn clause compiler, and a run-time system. The user interface is meant to assist grammar writers in organizing the components of a grammar by checking module dependencies, providing various test and trace functions, error messages, help pages and the like. A

large part of the compiler is devoted to the LFG formalism for syntax, which will not be treated here in detail; the full implemented formalism has been described in [May97]. LFG production rules are compiled to Definite Clause Grammars, which can be executed in an extended Prolog run-time environment. This environment includes interpreters for graph unification, semantic interpretation of f-structures, orthographic transducers, the segmentation algorithm, the lexical insertion procedure and the error handler that carries out derivation, as well as a menu and window-based user interface.

7.2 The LFG Compiler

To specify the grammar of a language, the user of KLU must write formal descriptions of the language for each of the modules Lcs, Opr, Stx, Mrph, and Alm. The descriptions in Lcs, Opr, Stx and Mrph are written in a slightly altered LFG formalism. This formalism requires that grammatical functions, lexical categories, and clitic particles be explicitly declared, and grammatical functions must specify their permitted values. The declarations allow the compiler to carry out a number of consistency checks to catch user errors, and they ensure that if a symbol is declared in a higher module, lower modules will inherit its definition rather than redefine the same symbol.

A language module is divided into PARAGRAPHS, or sections containing various kinds of information. The paragraphs are identified by DIRECTIVES that always begin with the symbol #. Each directive changes the state of a control variable in the compiler, causing the compiler to translate the syntax proper to the paragraph. The changes must follow a sequence defined by a state machine, so that each module can contain only the allowed paragraphs, in the required order. The declarations are translated to internal lists that can be consulted by the compiler when it is translating syntax rules and lexical items. The recursive descent compiler translates the contents of each paragraph using a definite clause grammar (described below) that gathers symbols in variables and writes the translation to an output file. It is based on straightforward techniques described in the compiler literature, e. g., N. Wirth [Wir84], to which the reader is referred for details.

Fig. 7.1 shows a small Lcs module for an Italian grammar containing the lexical noun *fata* ‘fairy’ and the verb *iscrivere* ‘register’. In KLU’s lexical modules, each grammatical function must be explicitly declared with its range of permissible values; grammatical functions are thus seen as strongly typed. The type declarations are used by the compiler for some static checks. Run-time errors, i.e., attempts to unify a feature with a value for which it is not defined, are unfortunately not found, although this would often be useful during grammar debugging. The governed functions in the figure are declared with a range &, which means that they can take any value, including a set of functional equations. The non-governed functions take only scalar values. LFG defines the quotation marks placed around the predicate of a lexical form like “Fata” as generating a distinct instantiation of the predicate, but the quotation marks are ignored by the KLU compiler. The semantic formula for Fata corresponds to the notation NYMPHA: OBJECT that was introduced in chapter 4, and the additional variable *u* is supplied to allow creation of an instantiation of the concept as a discourse object or

situation. The syntactic entry corresponds to (104 a), p. 172.

The lexical semantic entry for the verb Iscrivere 'register' presents a redundancy: it defines the meaning of the verb with both a conceptual predicate IMMATRICULO(S, O1, O2) and with a lexical semantic formula

CAUSE(S, CHANGE(NOT(LISTED(O1,O2)), LISTED(O1,O2))).

The redundancy is not required, since it is assumed that the lexical formula is simply a lexicalized simplification of a unique conceptual structure. Having an atomic conceptual predicate available in the semantic lexical merely simplifies the query that must be presented to the conceptual data base when the semantics have to be interpreted, e. g., during morphological derivation.

The compiler translates the semantic entry for Fata

```
§ Fata  => object(NYMPHA, u).
```

to a nearly equivalent Prolog representation,

```
 Lcs:Fata -> object(NYMPHA, u) ;
```

(Note that the syntax is the now antiquated syntax of Prolog II, which had to be retained for compatibility reasons; user-defined modules can be written for KLU in ISO (Edinburgh) syntax, if desired.)

The syntactic entry for *fata* is translated to a Prolog rule in which the terminal category noun is the rule's functor, and the lexical symbol representing the stem Lcs:fat is the first term of a Prolog difference list.

(141)

```
Lcs:noun(Lcs:fat.s0, s0, _, m, f, c, _done) ->
       merge(m, (INFLCLASS.NDa).n0)
       merge(f, (PRED.w1).r1)  merge(w1, Fata)
       merge(f, (GEN.w2).r2)  merge(FEM, w2)
    word_protk(" Lcs:noun ", nil, Lcs:fat, m, f, c); %ENDE Lcs:fat
```

```
# Description Italiano.
-- This is a comment: Note that all grammatical categories
must be explicitly declared.

#  Governed functions are.
SUBJ { & },   OBJ { &},   OBL { & }.

#  Non governed functions are.
NUM  { SG, PL },   GEN  { FEM, MAS },
PERS { 1, 2, 3 },     PCASE { Ad }.

#  Lexical categories are.
Prep,  N,   V.

#  Morphological categories are.
noun,   verb,   adjective.

# Inflectional classes are.
INFLCLASS  {NDa, NDo, NDe, VKare, VKere, AdjDg, AdjDog }.

# Lexical concepts are.
-- First we see the lexical semantic entry for Fata, a subclass of
object.

§ Fata  => object(NYMPHA, u).

-- Then we see the corresponding syntactic entry.
/fat/  noun, NDa,   (^ PRED ) = "Fata"
                    (^ GEN ) = FEM.
-- Here an item with argument structure; the mapping relation
is SUBJ <=> S; OBJ <=> A1; OBL <=> a2.

§ Iscrivere(S, A1, a2)  =>
          event(e, IMMATRICULO(S,  A1, a2))
          agent(e, S)
          theme(e, A1)
          loc(e, a2)
          phasesemantics(e, 1, CAUSE(SS, CHANGE(
                          NOT(LISTED(A1,a2)), LISTED(A1,a2)))).

/iscriv/ verb, VKere, (^ PRED) = "Iscrivere<(^SUBJ),(^OBJ), {(^OBL)}>"
                      (^ AUX) = AVERE
                      (^ OBL  PCASE) =cc Ad.
# End Italiano.
```

Figure 7.1: Source Module for Lexical Conceptual Structure, Lcs.

The corresponding Prolog translations for the syntactic entry /iscriv/ and the semantic entry Iscrivere are (details are presented later):

(142)

```
Lcs:verb(Lcs:iscriv.s0, s0, _, m, f, c, _done) ->  %SYNTACTIC ENTRY
       merge(m, (INFLCLASS.VKere).n0)
       merge(f, (PRED.w1).r1) merge(w1, Iscrivere(^ SUBJ, ^ OBJ,  OBL))
       merge(f, (AUX.w2).r2)  merge(AVERE, w2)
       add_condition(f,  OBL.PCASE, Ad, _done)
       cond_merge(c, (Cid1. <'=cc', '^'.OBL.PCASE, Ad>).r3)
    word_protk(" Lcs:verb",nil,Lcs:iscriv,m,f,c); %ENDE Lcs:iscriv

Lcs:Iscrivere(s, o1, o2) ->  %SEMANTIC ENTRY
      ereignis(e, IMMATRICULO(s, o1, o2) )
      agens(e, s)
      thema(e, o1)
      loc(e, o2)
      phasensemantik(e, 1, CAUSE(s, CHANGE(NOT(LISTED(o1, o2) ),
                                    LISTED(o1, o2) ) ) ) ;
```

The entries in the index table for *fat* and *iscriv* are

```
Lcs:lex_idx("fat",NDa,noun,Lcs:fat,Fata)->;
Lcs:lex_idx("iscriv",VKere,verb,Lcs:iscriv,Iscrivere(^SUBJ,^OBJ,OBL))->
```

Rules of syntax and word formation are translated in a similar way. For examble, the syntactic rule (appearing in the module Stx, not shown)

(143)

```
NP ->   Det      ^ = v
        N        ^ = v
              (^PERS) = 3 .
```

is translated to a Prolog rule

```
Stx:NP(s1, s99,  k, f, _tiefe, _done) ->
    searchdepth(_tiefe, t2, _done)
    lex_insertn(:SRCH_CTX, "Det",s1, s2, k1, _, f1, _, _done)
        merge(f, f1)
    lex_insertn(:SRCH_CTX, "'N'",s2, s3, k2, _, f2, _, _done)
        merge(f, f2)
        merge(f, (PERS.w1).r1)  merge('3', w1)
    eq(s99, s3)
    eq(k, <NP, k1, k2>)
   pars_protk(" Stx:NP ", s1, s99, _, f, _); %ENDE NP
```

How the KLU run-time system evaluates these structures is explained in the following paragraphs.

7.2.1 Definite Clause Grammars

The formalism of Lexical Functional Grammar, defined by Kaplan & Bresnan in [DKMZ95], represents rules of syntax as rewrite rules augmented with systems of functional equations that are unified at some unspecified time. The rewrite rules provide a means of formulating generalizations about the surface constituent structure or C-STRUCTURE of a language, and such rules can be easily implemented in Prolog.

Like other versions of Prolog, Prolog II [GKPvC85] defines a 'logic language' over facts and rules of inference. By including a list data type in the kinds of objects about which logical inferences can be stated, Prolog can express the rewrite rules of a grammar as a set of logic statements about sentences, represented as lists of lexical symbols. The lexicon of the grammar can be similarly formulated as a set of Prolog facts about terminal categories of the grammar. If a list of strings or lexical symbols is presented as an argument to the category rule for Sentence, the Prolog system can draw the inference that the list is or is not a sentence described by the grammar. A separate parser is not necessary because the list data type can be presented directly to the rule.

A lexical entry is expressed as a fact. Such a fact can be, for example, that *horse* appearing by itself in a sentence is a noun. This can be expressed in Prolog II by writing

```
noun(horse.S, S) -> ;
```

The two arguments of the functor `noun` can be read as saying that a noun can be the word *horse* followed by a list (possibly empty) of further words, *S*. Together the two arguments constitute a DIFFERENCE LIST saying that two lists are identical except that one starts with the symbol *horse* while the other does not. When unified with the variables in other rules of the grammar, the two expressions have the effect of saying that a noun is present at the first position in a list of words and is followed by *S*.

A rewrite rule of the grammar is similar to a lexical fact. Its functor names a constituent category of the grammar and its arguments define a difference list over a portion of the sentence. It is stated, however, not as a fact but as an inference from further statements about adjoining constituents. A query to the grammar takes the form of a proof that a certain constituent spans a certain range of symbols in a list. For the category Sentence, of course, the category spans the entire list, so in the query, the second argument must be the symbol *nil*, which marks the end of a list.[2] The main constituents, e. g., NP and VP, span adjoining sublists. Consider the following grammar:

```
sentence(S1, S3) -> np(S1, S2)  vp(S2, S3);
np(S1, S3) -> det(S1, S2) noun(S2, S3) ;
vp(S1, S2) -> verb(S1, S2) ;

det(the.S, S) ->;
noun(horse.S, S) ->;
verb(sings.S, S) ->;
```

[2] A KLU grammar can also define the sentence as the list of words preceding the full stop or question mark; in this case the second argument would be the end mark followed by *nil*.

The query `verb(sings.nil, nil)` is true, or 'succeeds', because the difference list between `sings.nil` and `nil` is a `verb`. Similarly true are `np(the.horse.nil, nil)` and `vp(sings.nil, nil)`, from which the deduction system can infer `sentence(the.horse.sings.nil, nil)`. (Prolog inference is from right to left, in the *opposite* direction of the arrow.)

Definite Clause Grammars [PW80] extend the power of rewrite systems to systems that can make more specific inferences than simple judgments of grammaticality. They can, for example, deduce the constituent structure of the sentence or draw semantic inferences. They can also introduce context dependencies like those imposed by agreement relations, e. g., between determiner and noun in a language like French. This is can done by declaring additional variables in the rules that must unify as Prolog terms. For example, in the following fragment, the gender attributes of the determiner `det` and of the noun `nn` are unified via the variable *A*. If they are not the same, the unification, and therefore the inference, fails. Hence, *la chaise* will be accepted as a noun phrase `np`, but *le chaise* will not.

```
np(S1, S3, A) -> det(S1, S2, A) noun(S2, S3, A) ;

det(la.S, S, fem) ->;
det(le.S, S, masc) ->;
noun(fauteuil.S, S, masc) ->;
noun(chaise.S, S, fem) ->;
```

Because the unification mechanism can be extended to lists and directed acyclic graphs (DAGs), it provides a tool for implementing complex and powerful grammars. Relying on unification rather than concatenation to test well-formedness of sentences, Definite Clause Grammars can describe relations like agreement, argument dependencies, and non-configurational phrase structures.

7.2.2 Unification of f- and m-Structures

In the LFG formalism of Kaplan & Bresnan [KB95], systems of equations over feature structures are unified. The implementation technique chosen in KLU for implementing LFG is not entirely correct with respect to this definition. Instead of unifying functional equations, the code produced by the KLU compiler represents systems of functional equations as directed acyclic graphs (DAGs), embedding the unification operations and constraint tests in definite clause grammars, as first described by A. Eisele and J. Dörre in [ED86]. For most purposes, KLU's graph unification can be seen as equivalent to the equational unification of the LFG formalism, however.

In each production of the grammar, clauses are added to the DCG rules to represent LFG's functional equations. In 143, for example, the simple functional projection equation `^ = v`, representing the usual $\uparrow = \downarrow$, compiles to `merge(f, f1)`. The unification of a specific functional attribute with an atomic value, `(^PERS) = 3` leads to two predicates, `merge(f, (PERS.w1).r1)` and `merge('3', w1)`. The first contains a 'locator' expression that unifies a local variable `w1` with the current f-description in `f`. The second predicate unifies this variable *w*1 with the constant on the right-hand side of the equation, namely 3.

In the syntax module Stx, terminal symbols are treated differently from non-terminals. A syntactic rule is allowed to insert only grammatical categories as terminal symbols, whose names must begin with an upper case letter (cf. section 5.8.4). However, instead of writing `Noun(S2, S3)` at the point where a grammatical Noun is inserted in the c-structure, the compiler writes

```
lex_insertn(:SRCH_CTX, "Noun",s2, s3, k2, _, f2, _, _done).
```

This clause calls KLU's lexical insertion procedure, which is responsible for testing the grammatical well-formedness of all lexical items presented to syntax and thus for enforcing Lexical Integrity. It receives the name `:SRCH_CTX` of a list of lexical modules that should be searched for the item (here Lcs and Opr but not Mrph), the symbolic name of the category (`Noun`), as well as variables for the difference list that the item spans (`s2`, `s3`) and for the c- and f-structures (`k2` and `f2`). Since c-structure is being built on the Prolog call stack, the c-structure built in `k2` is redundant; it is used solely for creating traces. Between `k2` and `f2` an unnamed variable is supplied for m-structure, which unifies within the lexical item but is not passed on to the syntactic node, since m-structure is presumed to exist only within words. In rules of word structure, this position is created with a named variable.

The syntax rules also allow for constraint equations, also implemented as rules. Unlike functional equations, constraint equations cannot be evaluated until their arguments have been fully instantiated. In some cases, this is possible at the run-time point where the clause is called; in other cases the evaluation must wait until the f-structure is complete. The variable `_done` is used as a synchronization flag for a waiting evaluation of the constraint.

Unification of the functional structure is carried out by a destructive unification of Prolog list structures that has a certain tradition in the Prolog literature (cf. e. g., [GM89, 236]). In KLU's compiled code, unification is carried out by the rule `merge`.

```
% *** merge/2
% Graph-Unification of 2 DAGs
%  Arguments must be DAG-Structures
% merge(1.(2.3).4, 1.y);  --> {y=(2.3).4}
%
% 1. Unification of structurally identitical lists
% 2. Unification of f with a partial struktur a.w1
% N.B.: the "open-list" variables of the 2 lists must be different!

merge(x,x) -> !;
merge((a.w1).r1,f) ->
del(a.w2,f,r2)
merge(w1,w2)
merge(r1,r2);
```

```
% *** del/3
% search for path x in x.y; fetch value in/from y; or delete x from y
%  del(2, 1.2.3, x); -->   {x=1.3}
%    del(2, 3.a, x); --> {a=2.v1,x=3.v1}
%    del(2.w2, 1.(2.3).r, x); --> {w2=3,x=1.r}
del(x,x.y,y) -> !;
del(x,e.r,e.s) -> del(x,r,s);
```

Variants of the `merge` predicate are used in KLU to implement a 'negative' unification (`=/`), which fails when two structures unify, and a 'restricted' unification, which can ignore specified features.

7.2.3 Constraint Testing and Parallelism

The LFG formalism defines the well-formedness of a sentence in terms of a so-called context-free skeleton (represented implicitly by the Prolog call stack); the unification of functional equations (in KLU, these are m- and f-structure equations at the word level and f-structure equations at the sentence) level; and constraints on functional equations. In syntax, an equation of the form $A =_c B$ does not represent a unification. The operator $=_c$ is a boolean function that fails if the condition is not found to be true. Constraint equations thus allow conditions to be tested without introducing new information into f-structure. Since we have found no evidence of constraint relations at the level of word structure, KLU does not provide constraint equations in morphotactics.

The definition of the LFG formalism makes no statements about the order in which c-structures, f-structures and constraints must be evaluated; however, dependencies among the three forms of constraints do exist to the extent that functional equations require information about the c-structure nodes to which they are attached, and constraints cannot be tested until all variables to which they apply have been bound. This still allows a great deal of freedom in choosing a parsing strategy, and [MK95] report benchmarks for parsers that evaluate c-structure, unification structures and constraints in sequential order, in parallel, and in interleaved fashion. A compiler can, in addition, optimize a grammar in many ways, for example by reformulating c-structure rules to allow compilation to LR-style tables, and by turning simple functional attributes into additional category names (e.g., functional gender attributes for nouns and adjectives can be turned into categories like N-MASC, N-FEM; ADJ-MASC, ADJ-FEM, etc.). While unification operations are in general slow, an optimization of this sort can create fast parsers even for some grammars that are heavy with functional information.

Sequential evaluation, however, makes it difficult to extract useful trace information during a parse, which was felt to be especially important for an experimental grammar development program. Functional and constraint equations can make the majority of the context-free parses invalid; hence, for a grammar developer a parser is more useful if it tries to invalidate a branch of the parse tree as soon as possible; i.e., if it does not explore lots of low-order branches of the parse tree that the grammar in fact does not permit. For this reason, the KLU compiler produces DCG rules that carry out f-structure unification simultaneously with the c-structure parse (e.g., via the `merge` clauses in the noun phrase rule (143)). Constraint equations are also tested to some degree in parallel with the parse, by means of predicates that implement the con-

straint equations. However, before a constraint test can be performed, all variables in the equation must be bound. The constraint predicates are so implemented that they execute immediately if both sides of the equation have defined values; if not, a Prolog II synchronization predicate (`freeze`) is invoked to delay evaluation of the constraint until the full f-structure is available. Tests of a more elaborate synchronization strategy, in which a constraint could be evaluated as soon as all of its variables were bound, actually produced worse parse times than this somewhat simple-minded approach.

7.2.4 Index Table

The KLU model requires access to morphemic segments in during segmentation, when possible segments must be tested for presence in the lexicon. To provide quick access to just the information about segments that the lexicon contains, KLU postulates a kind of index table, as suggested, for example, by the logogen model of Morton [Mor69], discussed in section 3.2.1. During compilation of the lexicon, the KLU compiler builds an index table, as shown below for a small Italian lexicon:

```
Lcs:lex_idx("a",nil,Prep,Lcs:'a',AD(^ ARG1, ^ ARG2))->;
Lcs:lex_idx("in",nil,Prep,Lcs:in,IN(^ ARG1, ^ ARG2))->;
Lcs:lex_idx("con",nil,Prep,Lcs:con,CON(^ ARG1, ^ ARG2))->;
Lcs:lex_idx("funG",NDo,noun,Lcs:funG,Fungo)->;
Lcs:lex_idx("arriv",Vare,verb,Lcs:arriv,Arrivare(^ SUBJ, ^OBL))->;
Opr:lex_idx("il",nil,Det,Opr:il,v1956)->;
Opr:lex_idx("le",nil,Det,Opr:le,v1972)->;
Mrph:lex_idx("o",NDo,nsufx,Mrph:'o',v1400)->;
Mrph:lex_idx("i",NDo,nsufx,Mrph:'i',v1400)->;
```

Each record of the index table, represented by a Prolog fact, contains an access form (lexical string), plus additional information like inflectional class (paradigm), and properties of the lexical entry from which the record was compiled.[3] Monomorphic words, like the Italian prepositions *a*, *in*, *con* have complete access forms, but for segmentable words like the nouns *fungho*, *arrivare*, etc. we see only the stem or root. The suffixes for these words appear in the index table of the region Mrph. Exactly how much segmentation should be present in a lexicon is not entirely clear, but it seems reasonable to assume that at least the majority of prefixes in a language will be segmented in the access forms, as well as the inflectional suffixes, based on the psycholinguistic evidence of chapter 3. As its primary use is for segmentation, the index consists entirely of such segments, and it never contains complex, morphologically segmentable words.

Altogether there are three such tables. For each of the three lexical regions Lcs (semantics), Opr (operatives), and Mrph (morphotactics) the compiler produces a separate index table, so that segmentation can restrict its search at each level to the corresponding region(s) of the lexicon. Candidate segment strings presented by the orthographic transducers (next section) are compared to the tables by the Prolog system, which uses hash-table indexing for look-up.

[3] For segments compiled from the region Lcs, the compiler also inserts a copy of the lexical form, which would be needed for language production.

Records in the index tables contain some additional data that can be useful during segmentation. These include the inflectional class (second argument position), which can help limit the search for segments, the syntactic category (third argument position), the lexical symbol (fourth position) and the lexical predicate (fifth position). Although no implementation has been attempted, I assume that speech or text generation will also require a form of rapid indexing from lexical semantics and syntactic categories to access strings, and derivation, as a kind of generation, requires this also. As noted briefly in section (5.8.1), an interesting side-effect of this organization is that it accounts for the priming effects that arise with complex words e.g., *cook°able* priming *cook*. Since segmentation must access each of the morphemic segments individually, before lexical access takes place, the access to the segment /cook/ will entail an access to the entire index table entry and thus also to the lexical form Cook(↑SUBJ, ↑OBJ). When *cook°able* is recognized as a prime, it prepares recognition of the segment /cook/ by accessing the index table, and when a target *cook* is presented, access to the lexicon entry Lcs:/cook/ is faster because the recognition segment has already been activated, and less time will be required for subsequent analysis.

A further use of the segment table arises in derivation, where it may be necessary to map from a lexical form like Cook(↑SUBJ, ↑OBJ) to the root node of the data structure for the entry in the lexicon. The Prolog II+ representation does not allow this easily, so this is aspect is only an implementation convenience whose theoretical implications are unclear. I assume that in the cognitive representation direct mapping from a lexical form to the access form is possible, and that lexical forms are probably not present in the index tables.

7.3 Orthography

Since KLU works only with typed, that is to say, graphemic input, and since phonological representations would be much more complex than they are for text processing, no serious attempt has been made to take account of phonology as such. Nevertheless, the string representation of an item in the lexicon sometimes needs to allow for a certain amount of allomorphic variation. To the extent that orthographic variation sometimes reflects phonology and in any case poses many similar problems, the mechanisms required for dealing with orthography can borrow from phonology, and the solutions found may be of interest for phonological theory. (Although transducer-based solutions for orthographic variation in text processing systems have been presented as representations of phonology, these solutions may well be much less suited for the phonetics-phonology interface in a speech processing system, as I have argued in section 6.2.1.)

7.3.1 Orthographic Rules

As noted in chapter 6, frequently used allomorphs like *dig* and *dug* are probably present separately in the lexicon, but in some cases, especially where speakers are not sure of the correct forms, as with *weep* vs. *wept* and *weeped* (section 5.1), or with the German plural formation *Nuss* 'nut' $\longrightarrow$ *Nusse* or *Nüsse*, a rule-based representation of the possible allomorphs captures the ambiguity better — especially when possible forms

that speakers would accept have not yet been encountered. For cases like *Nuss* we need to represent the access form with an underspecified grapheme in the access form that can be mapped both to u and ü in the input or output via a phonological rule (another rule recognizes that surface N must be capitalized, cf. (148) below). As this mapping is only possible for stems of a certain inflectional class that builds its plural with *-e*, the rule requires an additional notation, restricting the rule's application to this class, as in (144). Expressed as context-sensitive rules in the formalism that I call SPE rules (from [CH68]):

(144) U $\longrightarrow$ u | ü. / InflClass = NDe

In other cases a rule's applicability may be dependent on the position of the underspecified grapheme, as with consonant doubling in English before suffixes beginning with a vowel, which requires a context specification, e. g., that the consonant appear at the end of the stem. In Italian, the consonants g and c represent two different sounds, depending on whether or not i or e follows. When the allomorphy requires the obstruent sound [g] or [k] to be retained before i or e, Italian orthography inserts the character h. This requires an abstract representation that leads to a digraph representation gh or ch when i or e is affixed, which again requires SPE rules.

(145) C $\longrightarrow$ ch / $\_\_$ e | i ; as well as
C $\longrightarrow$ c / $\_\_$ $\neg$e & $\neg$i.

(146) G $\longrightarrow$ gh / $\_\_$ e | i ; as well as
G $\longrightarrow$ g / $\_\_$ $\neg$e & $\neg$i.

In rule (145), C represents the abstract character for the obstruent /c/, while c and ch represent the corresponding surface characters. The context specification, following the character / uses an underline $\_$ to denote the position at which the mapping from C to c or ch applies, in this case in the position before the characters e or i. The second rule defines the complementary mapping in contexts where neither e or i follows. Additionally, the character ♯ is used to define a morpheme boundary; in KLU this means the beginning or end of an access form to a lexical segment or word defined in a lexical entry.

For the two-level morphology described in [DKK$^+$87], an elaborate rule formalism was developed that allowed considerably more precise descriptions of orthographic variation than SPE-style rules. The formalism was compiled to a single, large, determinate finite-state transducer that mediated between the lexicon and the input. For KLU a much simpler approach has been taken that (so far as it has been tested) nevertheless seems to be manageable because it takes advantage of level-ordering, as described in the previous chapter, so that a transducer maps only a limited class of word segments to one or more regions of the lexicon. For each level of segmentation the user must formulate orthographic rules that operate only at that level; these are then translated by hand into a table for an indeterminate finite-state transducer (introduced in section 5.2.3) that mediates between the segmentation alternatives at the given level and the lexicon. This of course runs much more slowly than a compiled, determinate finite-state transducer would run, but run times for our small grammars have remained quite acceptable. It is perhaps interesting that the orthographic rule compiler of the Alvey

Natural Language Tools also produces tables for non-determinate automata [RRBP92, 151], despite the run-time advantage of a determinized automaton. The Alvey automaton must be restarted at each character position in the word. In the KLU system, a similar effect arises as a hypothetical segment boundary moves through the word: each new position of the boundary produces two segment candidates that must be presented to the appropriate transducers (cf. Fig. 6.1).

Because each level of segmentation has its own rule system, the individual transducers are not collapsed into a single large transducer; rather they apply successively, at each level of segmentation. In a character-based system, this reduces storage space for the transducers but not necessarily run time; it is only in the context of ambiguous input, as in speech or optical character recognition, that level-ordering of the transducers appears to offer a run-time advantage. The main run-time advantage of level-ordering in KLU is that the segment searches can be restricted to sub-regions of the lexicon, corresponding to the morphological levels, and they are usually limited to the word's periphery.

The uppermost level of segmentation as defined in KLU is the word as it appears in the input, which for a language like Italian can contain cliticized pronouns that will be first be stripped. At this level in English it is sometimes necessary to allow for capitalization as a kind of emphasis, as in *the absolute Truth*, by reducing the upper case letter to the lower case letter of the lexical access form.

(147) $X \longrightarrow x \,/\, \sharp \_\_ .$

With X standing for a capital and x standing for a lower case character, this means that a lower case letter replaces a capital in the input at the position following the word's left-hand boundary, or at the beginning of the word. This rule has exceptions, however; proper names must always remain capitalized in the text. Since capital letters are used by convention to designate underspecified characters in the access representations, one can use an additional character, "$", to mark capitals that are not to be treated as underspecified. One writes the entry for "Paris" thus as /$Paris/.

(148) $\$X \longrightarrow X \,/\, \sharp \_\_$

If a grammar-writer wants to make use of null morphemes, it may be necessary to define underspecified characters in the lexical access forms that can be realized as a special, pre-defined null character "Ø". "Ø" can be defined as a lexical entry that can match an empty position in the input. For example, a null-morpheme account of English plurals would require that *dog* be presented to word formation as /dog/ + /Ø/. This allows the singular attribute to be added to the stem from the lexical entry for Ø. The lexical entry must specify the Ø/*-s* alternation as an additional, abstract character such as "F" (for inflection), placed at the end of each stem on which it may appear. This character expands to both the null morpheme *-Ø* and to the inflectional ending *-s* and is then separated from the stem during segmentation.

As indicated above, the segment transducers apply to segment candidates of a given level without regard to the segment's context. A limited kind of context dependency is possible, however. Each transducer specifies a variable for inflectional class (first argument of the Prolog fact) to implement context conditions like the one shown in (144). See the description of Argument 1 of the transducer table below.

7.3.2 Transducer Tables

Following the usual notation, transducer pairs in the KLU tables are separated by a colon, as in `"A":"a"`, with the lexical character appearing on the right side and the surface character on the left (cf. section 5.2.3, p. 160). Some other notational devices have been introduced to reduce the size of the transducers. For a transition that does nothing, one does not need to write `"A":"A", "B": "B", ..., "z":"z"`; the special notation `<"?">:<"?">` is equivalent, meaning "same character in lexicon and in input". The pairs `<"?">:"C"` and `"C":<"?">` have the meaning, "accept any character as the variant of the character `"C"`". 'Null' characters can be notated as "ø", meaning that nothing corresponds to the other character of the pair at the given position. For fixed sets of pairs, like the one containing upper case paired with lower case characters (`"A":"a", "B":"b", ..., "Z":"z"`) one can define a two-place Prolog predicate that defines the mapping and place it, instead of the individual character pairs, in the transducer table.

Corresponding to each level of segmentation, transducers are defined as Prolog clauses `postklitik`, `postlex`, `lex2`, `lex1`. Each of the transducers maps between an as yet unsegmented surface string and the index tables of one or more regions of the lexicon. The lexical regions to which the transducers' mappings refer are defined in a startup file for KLU. A transducer table is written as a set of facts defining a clause. The clause represents the automaton for a given level of orthographic analysis. Each fact within the clause represents a possible transition of the automaton. Fields within a transition fact are defined as follows:

Argument 1: an inflectional category. If bound, the transition is allowed only if the inflectional category of a segment searched for at the corresponding level in the lexical index table is the same. That is, if an umlaut rule `"ü":"U"` specifies `NDe` (for nominal-e-declination in German), then the umlaut will only apply when the found lexical item has the inflectional class NDe (cf. (144)). If this argument is a variable, it has the effect that if two or more segments are found marked in the lexicon for inflectional class, then the class of both must be the same, so long as they belong to the same level, which could help segment circumfixes like the German /ge . . . t/ and /ge . . . (e)n/. If this argument is nil, inflectional classes of the segments found in the lexicon are disregarded.

Argument 2: The initial state of the transition.

Argument 3: The character mapping. The first character is from the surface string; the second character must match the lexicon.

Argument 4: The state resulting from the transition.

Argument 5: `"SRT"` marks the single start state of the transducer, equivalent to / ♯ __ in SPE notation.

`"END"` marks a possible end state of the transducer. In the post-clitic transducer, this can be the end of the orthographic word. At the post-lexical level it is the end an

inflected word. At the lowest segmentation level, it is the end of the root. Equivalent to / ___ ♯ in SPE notation.

`nil` marks a state that is not a possible end state, i.e. further characters must follow.

To translate the rules (148) and (146) to equivalent entries in the transducer table for post-clitic allomorphy, we define an initial state for the transducer, say 0, with the entry (`"SRT"` = Start), and we create further transitions to account for all allowable mappings, as shown below:

```
% Start state = 0
postklitik(nil, 0, 0, 0, "SRT") -> ;

% In the start state, lower-case letters remain unchanged.
%      smallChar is a 2-place predicate defininig x : x for
%      lower-case letters.
% This is a possible END state of the transducer.
%      (e.g., for a word of one character like "I")
postklitik(nil, 0, smallChar, 2, "END") ->;


% Starting at the word's left boundary, look for a lexical "$".
% ø -> $ / # _
postklitik(nil,0, "ø" : "$" 1, nil) ->;

%  If found, the next pair must be capitals on both sides.
% X -> X / $ _
postklitik(nil,1, onlyCapital, 2, "END") -> ;

% After the rule for capitalization, further surface
%  and lexicon characters are identical.
% (i. e., <"?">:<"?">)
% ? -> ? / x _
% ? -> ? / $X _
% Since state does not change, this transition is valid up to the
% end of the word. Loop till end reached in state 2.

postklitik(nil, 2, <"?">:<"?">, 2, "END") ->;
```

The transducer table entries for introducing a null morpheme at a stem ending in the lexicon in `"F"` are

```
% Loop with 3 possible terminations:
% For the null-allograph of "s"
% "attend" -> "attendFØ" -> "attendF"+"Ø"
% "attends" -> "attendFs" -> "attendF"+"s"
```

```
% ? -> ? / _ #
% ? -> s /  F _ #
% ? -> Ø /  F _ #
postlex(nil, 0, <"?">:<"?">, 0, "END") -> ;
postlex(nil, 0, "s" : "F", 1, "END") -> ;
postlex(nil, 1, "ø" : "s", 5, "END") -> ;
postlex(nil, 0, <"s"> : <"s">, 2, nil) -> ;
postlex(nil, 2, "ø" : "F", 3, nil) -> ;
postlex(nil, 3, "ø" : "Ø", 5, "END") -> ;
```

Finally, as an example of a more general kind of orthographic rule, here are table entries that translate the abstract characters "G" and "C" to "gh" and "ch" before "i" or "e", and to "g" and "c" elsewhere.

```
postlex(nil, 0, <"?">:<"?">, 6, "END") -> ;
postlex(nil, 6, "g" : "G", 7, nil) -> ;
postlex(nil, 7, "ø":"ø", 8, "END") -> ;
postlex(nil, 7, "a":"a", 8, "END") -> ;
postlex(nil, 8, <"?">:<"?">, 8, "END") -> ;
postlex(nil, 6, "g":"G", 9, nil) -> ;
postlex(nil, 9, "h":"ø", 10, nil) -> ;
postlex(nil, 10, "i":"i", 11, "END") -> ;
postlex(nil, 10, "e":"e", 11, "END") -> ;
postlex(nil, 11, <"?">:<"?">, 11, "END") -> ;
```

Even this very small amount of orthographic mapping produces a great deal of search at the lexical interface. For example, the Italian surface string "funghi" 'you function' or 'mushroom + PLURAL', produces these search strings

(149)

```
"funghi" ->
       "funghi"
       "funghiFØ"
       "funghi"
       "funGi"
```

None of these orthographic alternatives, however, corresponds to a lexical item, which will provoke segmentation. All of the alternatives have known suffixes, *-i* or the null suffix *-Ø*, but only one segmentation terminates with an Lcs access form, which gives a list of lexical segments. (Note that the F-alternation is probably not required for Italian.)

7.4 Segmentation

The segmentation algorithm was described in detail in chapter 6. The segmentation procedure is applied successively to each of the strings presented in an input sentence.

In all cases, the result for each input word is either a single lexical segment or a list of alternative segmentations, with each alternative represented as a list of lexical segments. For the non-listed verb *disiscrive*, segmentation produces a nested structure corresponding to three levels of level-ordering analysis, with the inflectional suffix *-e* at the outermost level and the derivational prefix *dis-* one level deeper (the bottom level being not bracketed):

```
Prego >La fata disiscrive il cavaliere.
Segmentation results:
≪    la
         {  [ fat 'a'  ]
           }
         {  [
               [ dis iscriv  ]
            'e'  ]
           }    il
         {  [ cavalier 'e'  ]
           }       ≫
```

Note that the segmentation of "fata" does not have the form that lexical phonology would predict. Since the inflectional affix /a/ is a level 3 segment and the stem /fat/ is a level 1 (possibly level 2) segment, the result ought to be [[[fat]] a]. The segmentation algorithm tries, however, to be as non-committal about level-ordering as possible, and it introduces brackets only at boundaries where a new lexical context is entered. The last level boundary (for /fat/) is not bracketed.

The level-ordering hypothesis predicts that once a lexical segment has been found, further segmentation is not necessary. In fact, however, it is easy to find cases where this is trivially false, as was shown in section 6.7.4. The segmentation strategy of KLU can be parameterized in various ways, to determine whether segmentation proceeds from right to left only or in both directions, and whether each level is allowed to see the whole string or only a substring left over from a higher level. Setting these parameters correctly, it turns out, is not easy.

7.5 Lexical Insertion

To avoid potential confusion about the various forms in which lexical items can be represented at various places in lexical structure, section 6.7 introduced some important distinctions which I recapitulate here. When a user of the KLU program types a word as input, e. g., *uncorkers*, it is first represented internally as the SURFACE STRING, "uncorkers". This string is not presented directly to the lexicon; instead it must pass through levels of segmentation and matching orthographic rules. At each stage of segmentation, a transducer presents an ABSTRACT STRING to one or more index tables of the lexicon, which may contain a matching LEXICAL SEGMENT. Lexical segments are unique, but the lexicon itself can contain homonyms of the same lexical segment as distinguishable LEXICAL SYMBOLs, e. g., *by* as a preposition in Lcs having a spatial predicate and as a preposition in Opr having a passive case marking. Lexical access

forms can be lists of lexical symbols (/Lcs:bed . Lcs:and . Lcs:board/), and lexical symbols can themselves be internally segmeted (Lcs:/re°joice/). Both kinds of LEXICAL CHAIN are represented as Prolog lists. Thus, lexical insertion must be considered relative to three essential cases: monomorphemic access, collocations, and internally segmented words.

7.5.1 Monomorphemic access

A monomorphemic word like *is* presents the simplest form of lexical access. The result of its segmentation is the lexical segment /is/, which must be searched for in the lexical index tables, possibly leading to two lexical symbols, `Opr:/is/` and `Lcs:/is/`. Which of the attached sets of lexical attributes fits into sentence structure will, of course, need to be sorted out by syntax.

Being monomorphemic has nothing to do with the distinction between grammatical and lexical words introduced in section 5.8.4 (p. 198). The verb forml /is/ is a grammatical Verb that can be inserted directly into syntax. In the case of a verb like *run*, however, we have systematic variation between *run* and *runs*, and it is more plausible to describe *run* as a lexical verb that must receive marking for person and number from an inflectional affix. Thus, *is* can be inserted directly into syntax via the production rule Verb $\longrightarrow$ *is*, but *run* must be inserted via a detour through a word-formation rule.

7.5.2 Access to collocations

Fixed strings of words, like *bed and board*, which I shall call collocations to distinguish them from idioms and lexicalized constructions, are represented in the lexicon as lexical chains, and they are inserted into syntax in the same way as monomorphemic syntactic words, except that each lexical symbol in chain (except the last) carries a pointer to the next. (Idioms and constructions are formally difficult to describe in LFG and have no representation in KLU.) When "bed and board" is presented as input, the access mechanism thus finds two lexical symbols for *bed*. One is the atomic lexical noun `Lcs:/bed/`; the other is `Lcs:/bed/`, chained to the symbols `Lcs:/and/` followed by `Lcs:/board/`, with the whole chain being the access form for a grammatical Noun attached to a single lexical description with the semantics 'lodging-with-meals'. Collocations that might require inflection, e.g., French *travaux publics*, cannot be represented as lexical chains in KLU.

Segmentation cannot recognize *bed and board* as a single lexical unit, and it will simply pass three symbols to lexical insertion as

(150) `<< . . . bed and board . . . >>`

where $\ll$ $\gg$ mark the sentence boundaries. When the Noun /bed/ is fetched from the lexicon, it will be marked as being attached to further lexical symbols in its chain, and lexical insertion will try to match the lexical chain to further symbols in the input stream (via Prolog list unification). When an end-marked symbol is reached, the match is complete, and the symbols extending from *bed* to *board* in the input will be consumed as a single lexical Noun, contributing a single set of syntactic and semantic features to the sentence.

Of course, individual lexical entries for the nouns *bed* and *board* can also be accessed via the mechanism for monomorphemic access. This requires additional lexical entries (lexical symbols) not chained to further symbols but linked to the syntactic and semantic descriptions of these words. Members of chains are not required, however, to have any representation as separate lexical symbols. In the French collocation *au fur et à mesure*, *fur* is not a current lexical item. It has no lexical entry of its own. It is linked only to the symbols `Lcs:/au/` and `Lcs:/et/` and not to any syntactic or semantic description. It has no syntactic category, and it is not independently visible to syntax. It functions solely as a link in the chain, as a lexical atom without any lexical attributes. But as a lexical segment it is represented in the index table, and thus segmentation can recognize it as a word-like lexical symbol.

7.5.3 Access to Internally Segmented Words

An internally segmented word like cook°able is also represented in KLU by a lexical chain, but the chain is nested inside a pair of word delimiters, corresponding to the word delimiters in the input structure. Thus `A cook able to stew clams` in the input produces no match to `cook°able`. At a deeper level, however, the access to /cook°able/ proceeds in the same way as with collocations — the lexical segment `/cook/` is identified as matching the lexical symbol `Lcs:/cook/`, but since this `Lcs:/cook/` is linked to a further symbol, `Lcs:/able/`, the match continues until a word boundary in the lexical entry matches a corresponding word boundary in the input.

Internally segmented items may also be found within a collocation, leading to a more complex representation of the item such as

(151) `<<. . . { [ field ing ] } average . . . >>`

where { } corresponds to word boundaries where the input word was segmented (except where clitics have been stripped and resegmented as independent words). The square brackets [ ] indicate individual segmentation alternatives within the input word. The collocation must be represented in the lexicon as a lexical chain having as its first symbol a nested lexical chain. The entire tree-like structure is the access form for a single set of lexical attributes of category Noun, which in this case leads to a single, obscure semantic predicate known mainly to baseball fanatics.

While in human recognition we would expect *bed and board* to prime recognition of *bed* and of *board*, the KLU model accounts for the priming effect not by semantic connections from the collocation (which could have a completely unrelated meaning) to the lexical entries Lcs:/bed/ and Lcs:/board/, but rather by the activation of the corresponding entries in the index table during segmentation, before *bed and board* is recognized together as a lexical unit.

7.5.4 The Insertion Algorithm

The implementation of lexical insertion is rather complex (nearly 29 kilobytes of Prolog source), but in outline it is as follows:

(152) 1. A monomorphemic word (e. g., *word*):
From a difference list *word.xxx.yyy.zzz.nil*, lexical insertion returns the lexical features for *word* to f-structure by copying them from the lexicon and leaves *xxx.yyy.zzz.nil* as the remainder for further analysis.

2. A 'linear' collocation (e. g., *au fur et à mesure*):
From a difference list *au.fur.et.à.mesure.xxx.yyy.zzz.nil* lexical insertion matches the lexical chain starting with *au* to an access form, returns the lexical features of *au.fur.et.à.mesure* to f-structure, and leaves *xxx.yyy.zzz.nil* as the remainder for further analysis.

3a. A word that has undergone segmentation because of morphemic or pseudo-morphemic segments in the access form:

The word *intention* in an input might produce three alternative segmentations

```
((in.tention.nil).
(in.tent.ion.nil).
(in.ten.tion.nil).nil).xxx.yyy.zzz.nil
```

One of these, *in tention*, will match the access form /in°tention/, represented internally as (in.tention.nil). For this case lexical insertion returns the lexical features of *in°tention* to f-structure, ignores the other segmentations, and leaves *xxx.yyy.zzz.nil* as the remainder for further analysis.

3b. An inflected word that has undergone segmentation because of morphological segments in the access form:

The word *admits* in an input might produce the segmentation

```
((admit.s.nil).nil).xxx.yyy.zzz.nil
```

This will have an analyzable morphological structure that leads to a unified f-structure description. Lexical insertion returns this description to the lexical buffer and to the sentence's f-structure, leaving *xxx.yyy.zzz.nil* as the remainder for further analysis.

3c. A word with an unknown, derived stem that has undergone segmentation because of morphological segments in the access form:

The word *inadmit* in an input might produce the alternative segmentations

```
((in.admit.nil).(in.ad.mit.nil).nil).xxx.yyy.zzz.nil
```

One of these, *in admit* could have an analyzable morphological structure that leads to a DPRED in the f-structure description. In this case lexical insertion raises an error and halts. The error handler attempts a derivation. If

the semantic-conceptual derivation is successful, it deposits the result in the lexical buffer TpLex, and restarts lexical insertion. Now, on the second attempted insertion, the derived word is found as /in°admit/ by step (152 3a), which returns the lexical features to f-structure and leaves *xxx.yyy.zzz.nil* as the remainder for further analysis.

Because 1. is a special case of 2., it has no separate implementation. A segmented word fetched from the lexical buffer TpLex must be found as a full-form entry, so word-syntactic analysis is not allowed in this region. This conforms to the strict reading of Lexical Integrity, which requires even inflection to be analyzed outside of the syntactic rule system. However, it might be more plausible to place only the stem in the lexical buffer to avoid filling the buffer with endless numbers of inflected forms that must be immediately purged.

7.5.5 C-Structure and f-Structure

When lexical insertion returns features from a segmented word to syntax, it also returns a constituent structure from the morphotactic analysis for debugging purposes. This structure is carried into the c-structure trace of the sentence, and it can be displayed during traces of sentence analyses. In the first example (153) an inflected Italian Noun *funghi* 'mushrooms' is shown with the features that would be inserted to syntax. The second case (154) shows the homographic inflected Italian Verb *funghi* 'you function'. The c-structure representations returned from word analysis play no role in sentence analysis, however, since, according to Lexical Integrity, the c-structure of a word is invisible to syntax.

(153)

```
  Noun      [PRED: fungo]
   |        [GEN: MAS]
   .        [NUM: PL]
 +-^--+
funG  i
```

(154)

```
                [PRED: Fungere(^ SUBJ, OBL)]
  Verb          [AUX: AVERE]
   |            [SUBJ:
   .               [NUM: SG]
 +-^--+            [PERS: 2]
funG  i         [MODE: IND]
                       PRES]
                [INF: ]
```

7.5.6 Error Handler and Derivation

When lexical insertion cannot pull a lexical item, such as a Verb like /dis°iscriv°e/, directly from the lexicon, insertion fails at step (152 3a). Using the inflection rule

$$\text{Verb} \longrightarrow \text{verbstem verbsuffix}$$

lexical insertion indeed finds /e/ as a verbsuffix via step (152 3b), but it cannot find /dis°iscriv/ as a verb stem, neither via (152 3a) nor via further inflectional analysis (152 3b). However, in addition to being a lexical category, verbstem is also a word derivation rule

$$\text{verbstem} \longrightarrow \text{verbprefix verb}$$

Invoking this rule (as an alternative to insertion of an internally segmented word), word-formation produces a word-level f-description as in (155) and, finding a DPRED, which is a feature not permitted in syntax, lexical insertion declares an error in trying to insert a Verb.

(155)

```
      [DPRED: NegReverse(^ ARG)]
      [ARG [PRED: Iscrivere<(^ SUBJ),(^ OBJ), {(^ OBL)}>]
       [AUX: AVERE]
       ]
   Derivation trap, -92  "Verb"
```

Simply to show how the control structure for error handling and derivation is realized, without going into detail, below is shown the rule `lex_insertn` that realizes lexical insertion in KLU. The predicate `default` realizes a control structure somewhat like a **case** statement, which investigates all insertion possibilities realized by `lex_insertn_all`. If nothing is found here, `default` calls `lex_insertn_tplex` to see whether a previous derivation might have left something in the lexical buffer, TpLex.

The crucial structure here is the `block` predicate, which creates an error-handling environment. If during the search for an insertable word an error occurs (e. g., resulting from an attempted conversion of a monomorphic noun to a verb), rule 1. raises the error and passes control to the error handler, `lex_ins_errh`, with information about the error in the parameter list. Rule 2. examines the lexical buffer after insertion at 1. has failed: at this point there may be a fresh entry in TpLex which can be returned to satisfy the request from syntax.

```
lex_insertn(_kontx, _kat, s0, s1, k, m, f, c, _done) ->
   default(lex_insertn_all(_kontx, _kat, s0, s1, k, m, f, c, _done),
     % Only a total failure of all search modalities
     % leads to a search of TpLex for a previous derivation
     lex_insertn_tplex(_kontx, _kat, s0, s1, k, m, f, c, _done) );
```

```
%1. Insertion attempt from all lexical regions
lex_insertn_all(_kontx, _kat, s0, s1, k, m, f, c, _done) ->
   :KLU_MODULES(_kontx, _,_, <_, _, _mods, _>)
   block(e1, _msg, lex_insertn_1(_mods, _kat, s0, s1, k, m,
                                 f, c, _done))
   lex_ins_errh(e1, _msg, _, _kat) ;

% 2. If nothing was found at 1., check TpLex only.
lex_insertn_tplex(_, _kat, S.s9, s9, k, m, f, c, _done) ->
   block(e1, _msg, lex_insertn_rk("TpLex".nil,
                  _kat, S, nil, nil, k, m, f, c, _done))
   lex_ins_errh(e1, _msg, _, _kat) ;
```

7.6 Program Traces

The KLU program can provide detailed traces of a word's segmentation and morphotactic analysis. This section first shows how word segmentation can proceed. A trace of the analysis of the Italian diminutive derivation of *funghini* 'little mushrooms' serves as an example that shows inflection, an orthographic mapping during segmentation, and a derivation after morphotactic analysis. Finally, a detailed trace of the derivation of the neologism *disiscrive* 'ex-matriculate' in the context of a sentence is shown.

7.6.1 Trace for *funghini*

Orthographic analyses and segments can be seen for each of the lexical levels by choosing a corresponding menu option in the program. After the user types the surface string, followed by a full stop, the program displays all abstract strings produced by the transducer and all segmentations proposed for the level.

For `funghini` the post-clitic transducer does nothing:

```
String>funghini.
Alm:postklitik -> funghini
```

The post-lexical transducer used in this trace contains a rule that can add an abstract inflection `F` followed by a null morpheme Ø, as was described in section 7.3, as well as a rule that maps surface `g` to lexical `G`. At this level, segmentation removes affixes that it finds in the region Mrph. The found affix can now be represented by a lexical symbol (`Mrph:'i'`), but the remainder is still a surface string, appearing in square brackets to show that it furnishes input for the next lower level. Because of backtracking possibilities, some analyses are traced several times.

```
String>funghini.
Alm:postlex -> funghini
    Next level segments:  [funghin] Mrph:'i'
    Next level segments:  [funghin] Mrph:'i'
    Next level segments:  [funghin] Mrph:'i'
```

```
    Next level segments:  [funghin] Mrph:'i'
Alm:postlex -> funghiniFØ
Alm:postlex -> funghini
    Next level segments:  [funghin] Mrph:'i'
    Next level segments:  [funghin] Mrph:'i'
    Next level segments:  [funghin] Mrph:'i'
    Next level segments:  [funghin] Mrph:'i'
Alm:postlex -> funGini
    Next level segments:  [funGin] Mrph:'i'
    Next level segments:  [funGin] Mrph:'i'
    Next level segments:  [funGin] Mrph:'i'
```

The transducer for lexical level 2 contains no applicable rules. Segmentation finds the segment /in/ in the region Opr. This time the remainder can be found as a lexical symbol, which is deposited at the next lower level of the segmentation structure. The presence of a lexical symbol here blocks further orthographic processing and further segmentation.

```
String>funGin.
Alm:lex2 -> funGin
    Next level segments:  [Lcs:funG] Opr:in
```

Simple morphotactic analysis for a word of the grammatical category 'N' traces the results of segmentation and then the search for lexical symbols and word constituents. The segmentation shows the lexical symbols found by segmentation with their lexical nesting. In the following case, the stem `/funG/` and its inflection `/o/` are found as lexical symbols, and the functional attributes are unified to an f-description that can be inserted to syntax.

```
N>fungo.
Segmentation results:
  <<
         {  [ funG 'o'  ]
           }     >>

1:- Word (re)accepted

      [PRED: Fungo]
      [GEN: MAS]
      [NUM: SG]
```

Since this analysis builds a grammatical word, it also tests the resulting f-description for constraints, coherence and completeness, as if it had resulted from the syntactic analysis of a sentence.

```
 - Testing remaining constraints
 - Constraint equations satisfied
```

Derivational word analysis is shown here in a more detailed trace. Again the segmentation shows the lexical symbols found by segmentation with their lexical nesting; the detailed trace mode chosen here displays the region identifier for each symbol to indicate where the symbol was foud. These identifiers, however, are still regarded by morphotactics as lexical segments, i. e., as unspecified for region. Therefore, as morphotactics attempts to identify each of the word's constituents, it must check all of the lexical regions for possible matches. Before analyzing the word via word formation rules, it first searches the lexicon for a match to the lexical chain made up of the segments /funG/+/in/+/i/. This step (152 3a) fails, leading to an attempt to interpret the input as a string of morphotactic constituents (152 3b).

```
N>funghini.
Segmentation results:
<< { [ [ Lcs:funG Opr:in ] Mrph:'i' ] } >>

Trying to construct N -> funG in i
Searching TpLex  for N  -> funG in i
Searching Opr  for N  -> funG in i
Searching Lcs  for N  -> funG in i
Trying to construct N -> funG in i
Searching TpLex  for noun  -> funG
Searching Mrph  for noun  -> funG
Searching Opr  for noun  -> funG
Searching Lcs  for noun  -> funG
 Lcs:noun  -> /funG/
      [PRED: Fungo]
      [GEN: MAS]

Searching TpLex  for ndsufx  -> in
Searching Mrph  for ndsufx  -> in
Searching Opr  for ndsufx  -> in
 Opr:ndsufx  -> /in/

      [DPRED: Small(^ ARG)]
      [GEN: ]
      [ARG:
         [GEN: ]
      ]

 Mrph:nsufx  -> /i/
      [NUM: PL]
```

At this point in the trace, all of the input strings have been matched to lexical symbols, and Morphotactics can attempt to find a fitting constituent analysis. Applying word formation rules that add the diminutive suffix and then the inflection to the root produces the following f-description (cf. the derivation of *fatina*, p. 174), which is immediately recognized as ill-formed because of its DPRED (represented as a PRED in

(107)). It is therefore a candidate for derivational analysis, and if this is successful, for buffering in the lexical buffer, TpLex. This triggers step (152 3c).

```
* Analyzing 'N' for TpLex .  . .

      [ARG:
         [PRED: Fungo]
         [GEN: MAS]
      ]


      [DPRED: Small(^ ARG)]
      [GEN: MAS]
      [NUM: PL]
```

Substituting to the argument of DPRED, the semantic processor obtains Small(Fungo). Mapping the semantics of Fungo to conceptual structure, it obtains Small(FUNGUS). Submitting Small(FUNGUS) as a query to a conceptual data base (implemented as a simple Prolog rule package) returns a pointer to the node FUNGINUS in conceptual structure.[4] Lexical insertion creates Fung_in_i as a new lexical predicate for the derived item, maps its meaning to FUNGINUS, adds a lexical access form, a declination class, number and gender attributes, and finally reports a successful lexicalization.

```
* TpLex:'N' was stored.
```

Now the lexical buffer contains a grammatical word with the access form /funG°in°i/, with a mapped predicate, and no DPRED. Lexical insertion, which was aborted after this entry was created, is restarted. Since the buffer region TpLex is always searched before a list of segments is analyzed, the insertion candidate /funG/ + /in/ + /i/ matches directly to the access form, and the word is accepted as a grammatical category N.

```
Trying to construct N -> funG in i
Searching TpLex  for N  -> funG in i
 TpLex:'N'  -> /TpLex:'N'/
      [PRED: FunG_in_i]
      [NUM: PL]
      [GEN: MAS]


1:- Word (re)accepted
```

[4]For want of a better idea, KLU's concept module follows the precedent of the Grimms' *Deutsches Wörterbuch* in using Latin names as 'universal' keywords to designate items in conceptual structure.

```
      [PRED: FunG_in_i]
      [NUM: PL]
      [GEN: MAS]

 - Testing remaining constraints
 - Constraint equations satisfied
 - Testing completeness, coherence
```

7.6.2 Trace for *iscrive*

Taking up again the neologism *disiscrivere* that was worked through abstractly in section 5.4.3, we shall see how a sentence containing this non-lexical verb is processed by the KLU program. 'Reversative' derivations of this type were discussed at length in chapters 4 and 5, where it was shown that they represent one of the more challanging patterns for any comprehensive theory of morphological derivation. To ease the plunge into this rather complex example, however, I first show how sentence analysis proceeds from morphology to semantics for a sentence having a lexicalized verb *iscrivere* as its predicate. Hopefully, many of the points raised in the discussion so far will coalesce around this example.

An example of a simple sentence containing the lexicalized verb *iscrive*, which does not require any derivation, with its subcategorized arguments, could be (156).

(156) La fata iscrive il cavaliere al castello.
'The fairy registers the knight at the castle'

The main predicate *iscrivere* has as its semantics 'cause a change in the relation between the object, and some entity in which one can be listed or not, to the state of being listed'. In this case, the sentence apparently means

(157) CAUSE(Fairy, CHANGE(NOT(LISTED(Knight, Castle)),
LISTED(Knight, Castle)))

and this is roughly what we should expect an analysis of the sentence to produce (cf. section 5.4.3).

The first stage of processing is word segmentation. Half of the words carry inflectional endings and do not appear in their inflected forms in the lexicon. Since they cannot be substituted directly into the sentence's syntactic structure, lexical insertion must attempt to parse their word structures and unify the inflectional affixes with their stems, as in the examples (153) and (154) above. In each case this step yields a virtual grammatical word, which can be substituted into sentence structure as if it had been present in the lexicon. Syntactic analysis produces an f-structure for the sentence. Finally, the PRED values of functional arguments required by Iscrivere((↑ SUBJ),(↑ OBJ),{(↑ OBL)}) are bound to its functional argument positions. The PRED values are passed to the argument structure of the associated semantic predicate Iscrivere(S,O,L), and its implications, defined by the semantic lexical entry in Fig. 7.1, are written into discourse structure as accomplishment number 411.

A rule of sentence syntax leads in the usual way to the f-structure

```
._____________________________________________
| SUBJ:
|    ._________________________
|    | NUM: SG
|    | GEN: FEM
|    | SPEC: DEF
|    | PRED: Fata
|    | PERS: 3
|    |_________________________
|
| PRED: Iscrivere<(^ SUBJ),(^ OBJ), {(^ OBL)}>
| AUX: AVERE
| MODE: IND
| TENSE: PRES
| OBJ:
|    ._________________________
|    | NUM: SG
|    | GEN: MAS
|    | SPEC: DEF
|    | PRED: Cavaliere
|    | PERS: 3
|    |_________________________
| OBL:
|    ._________________________
|    | PCASE: Ad
|    | NUM: SG
|    | GEN: MAS
|    | SPEC: DEF
|    | PRED: Castello
|    |_________________________
|
| FORCE: ASSERT
|_____________________________________________
```

Substituting the semantic values of the arguments to *inscrive* and calling the rule defining its lexical semantics produces the semantic implications of the sentence.

```
Iscrive(Fata,Cavaliere,Castello) =>
          ACCOMPLISHMENT(411)
          AGENS(411,Fata)
          THEMA(411,Cavaliere)
          LOCUS(411,Castello)
          PHASE(411,
             CAUSE(Fata, CHANGE(NOT(LISTED(Cavaliere,Castello),
             LISTED(Cavaliere,Castello))))
```

7.6.3 Trace for *disiscrive*

Consider now what must happen with a sentence containing the non-listed verb *disiscrive*. Segmentation produces a nested structure reflecting the level-ordering analysis of the unknown verb, with the inflectional suffix *-e* at the outermost level and the derivational prefix *dis-* one level deeper (the bottom level again being not bracketed):

```
Prego >La fata disiscrive il cavaliere.
```

Segmentation results in:

```
<<    la
       {  [ fat 'a'  ]
         }
       {  [
             [ dis iscriv  ]
          'e'  ]
         }    il
       {  [ cavalier 'e'  ]
         }     >>
```

KLU displays the nested level-ordering structures, e. g., [[dis iscriv] 'e'] only as a matter of record, as noted earlier. The input to word formation is a flattened structure in which none of the word's lexical bracketing is visible (cf. Bracketing Erasure, section 6.6.1). The attempt to find an internally segmented verb /dis°iscrive°e/ fails, so a Verb cannot be inserted directly from the lexicon. Using the word formation rule

$$\text{Verb} \longrightarrow \text{verbstem verbsuffix}$$

lexical insertion indeed finds /e/ as a verb suffix, but it cannot find /dis°iscriv/ as a verbstem in Lcs. However, in addition to being a lexical category, verbstem is also a word formation rule in Mrph:

$$\text{verbstem} \longrightarrow \text{verbprefix verb}$$

Invoking this rule, word-formation produces a word-level f-description and, finding a DPRED, which is a feature not permitted in syntax, lexical insertion declares an error in trying to insert a grammatical Verb.

```
[DPRED: NegReverse(^ ARG)]
[ARG [PRED: Iscrivere<(^ SUBJ),(^ OBJ), {(^ OBL)}>]
      [AUX: AVERE]
]
Derivation trap, -92  "Verb"
```

This structure[5] of course suggests that word-formation has described the derivative as the reverse of *iscrivere*, but lexical insertion cannot yet pass the structure obtained by

[5] I hesitate to call this an f-description; strictly speaking, word formation has produced a proto-semantic rather than a functional structure, since we do not find functional relations among internal constituents in word formation.

unification on to the Verb node in sentence structure because it contains a DPRED. DPRED is a warning of potential problems. One of these is that the semantics and argument structure of NegReverse(Iscrivere) are not yet known, which means that syntax cannot yet know for sure that it must deal with a three-place predicate requiring a SUBJect, an OBJect and an optional OBLique argument, taking the auxiliary *avere*. A further problem is that for its OBLique argument *iscrivere* requires a localizing preposition like *a* 'at', but *disiscrive* will require a different preposition meaning 'out of' or 'from'. These are results that cannot be obtained simply by unifying features of *dis-* with its base.

Lexical insertion declares an error, triggered by the DPRED in the word's functional description, and it appeals to the semantics of NegReverse to create a new, syntactically acceptable lexical item. The rather schematic representation we see here in word structure is not really an adequate indication of the operation that has to be carried out, however (cf. section 5.6.1). NegReverse must

(158) 1) define a new lexical access form /dis°iscriv/
2) identify a new semantic predicate and argument structure
3) define a new mapping of semantic to functional arguments
4) redefine the subcategorization of the optional oblique argument
5) possibly redefine the f-structure attributes, like AUX (as might be required in passivization)
6) possibly redefine a new inflectional class as an m-structure attribute (as in the conversion of a verb to a noun)

Thus, much more information must in fact be passed to the operation NegReverse than the attributes PRED and AUX. The representation in the word structure is in fact at best a sketch of what the derivation actually entails, and it would be pointless to try to carry out the operation in the LFG formalism.[6] In KLU the semantics of NegReverse are defined by a Prolog predicate that implements a mapping from the m-, f- and constraint-structure found by word formation to the corresponding structures for the new lexical item.[7] These are passed to NegReverse (160) by the derivational error handler, in the first tuple <a, m, f, c>, with the following assignments:

(159) a = lexical chain from the input
m = m-structure (containing inflectional class of affix and stem)
f = word structure (including DPRED)
c = constraint list (from all constituents)

The new lexical entry is thus a function of all these arguments, not merely of the PRED of the base. The derivational function must determine the corresponding structures M, F, C, for the new entry, and in addition it must return, in S, the new predicate that will define the semantics of the new lexical entry. The lexical chain can be passed without changes, since the entry in TpLex will remain morphologically segmented.

[6]Certainly this would not be impossible, since the full LFG formalism is computationally quite powerful — but the operations involved are not easily comprehensible with the notational devices it presents.

[7]Because it represents f-structure with feature graphs rather than with sets of equations, KLU must place the constraint equations in a separate data structure.

(160)

```
NegReverse("Verb",  <a, m, f, c>, <a, M,F, C, S>) ->
          fetch_arg(f, NegUmkehr, A)
          find(reverse_of, "Verbal",  A, K, S)
          mapping("'V'", K, <_, m, f, c>, <_, M,F, C>);
```

The operation `find` must be defined so as to take a lexical semantic formula like (157), returning a formula (161) that has the 'reversed' meaning of *iscrivere*

(161) CAUSE(A, CHANGE(LISTED(O, L), NOT(LISTED(O L))))

and it must also check this formula for conceptual plausibility. The function `mapping` must then change all of the features in the m-, word and constraint-structures in accord with rules provided by mapping theory.

Having derived these structures, lexical insertion writes a new lexical entry to the Temporary concepts region of the lexicon. In the KLU program this entry can be examined via a menu function that lists the contents of the TpLex region.

(162)

```
 / dis°iscriv°e /  ================================
§ Dis_iscrivere(S,O,L)  =>
  accomplishment(v1)
  agent(v1,S)
  theme(v1,O)
  localization(v1,L)
  phase(v1,CAUSE(S,CHANGE(LISTED(O,L),NOT (LISTED(O,L)))))

     TpLex:Verb  -> /TpLex:Verb/
      [PRED: dis_iscriv((^ SUBJ),(^ OBJ),{(^ OBL)}]
      [AUX: AVERE]

      [μ^ INFLCLASS: VKere]

   (^ OBL PCASE ) =/cc Da
=====================================================
```

Lexical insertion is restarted for the lexical chain /dis°iscriv°e/, and the analysis can run to completion.

```
1:- New c/f description completed.

     KLUsys:LFGPHRASE
   +--------^--------+
   NP                VP
 +-^-+       +-------^----+------------+
Det 'N'  TpLex:'V'        NP           PP
 |   |       |          +-^-+       +--^--+
 la  .       .         Det 'N'   PrepArt 'N'
   +-^-+    +^-+        |   |       |     |
  fat 'a'  dis .        il  .      dal    .
            +--^--+      +--^---+      +--^--+
          iscriv 'e'  cavalier 'e'  castell 'o'


._____________________________________________________
| SUBJ:
|   ._____________________
|   | NUM: SG
|   | GEN: FEM
|   | SPEC: DEF
|   | PRED: Fata
|   | PERS: 3
|   |_____________________
|
| PRED: dis_iscriv<(^ SUBJ),(^ OBJ),{(^OBL)}>
| AUX: AVERE
| OBJ:
|   ._____________________
|   | NUM: SG
|   | GEN: MAS
|   | SPEC: DEF
|   | PRED: Cavaliere
|   | PERS: 3
|   |_____________________
|
| FORCE: ASSERT
|_____________________________________________________
```

giving the semantic analysis,

```
The relation dis_iscriv(Fata,Cavaliere,Castello) =>
     ACCOMPLISHMENT(411)
     AGENS(411,Fata)
     THEMA(411,Cavaliere)
     LOCUS(411,Castello)
     PHASE(411, CAUSE(Fata,
             CHANGE(LISTED(Cavaliere,Castello),
             NOT(LISTED(Cavaliere,Castello)))))
```

I have not said much about the functions *find* and *mapping* that do most of the work in the derivational predicate NegReverse. The operation of finding a possible concept from a given semantic formula, while crucially important for the derivation, is really an issue for knowledge representation rather than linguistics, and in the KLU concepts module it is implemented simply by pairing semantic formulas with their corresponding concepts in a table.

Mapping is more of a linguistic issue; given a concept, the derivational operation must be able to assign thematic roles and grammatical functions, but it must also take care of many other details required by real lexical entries. These include defining the lexical access form; defining morphological attributes like gender and inflectional class; and defining additional syntactic requirements like control relations and constraints on arguments. The current interest in argument structure and mapping has suggested ways of creating this part of the lexical entry. Other issues, like the assignment of inflectional class and gender, while not as exciting, may pose far-reaching theoretical questions.

For the few cases that have been worked out in detail, the mapping operation can be implemented as tables of ready made results. These tables give us at least an idea of what derivational semantics and mapping theory need to do and where they can be incorporated into a revised formal architecture of Lexical Functional Grammar. The theory of derivational semantics can be seen in this context as just the error handling function that takes derivational predicates and bases into semantic and conceptual descriptions; mapping is the theory that takes those same descriptions back into m-, f- and constraint structures for a function `make-entry` that creates new entries in a buffer region of the lexicon.

Chapter 8

Summary and Outlook

The appropriateness of declarative, unification-based approaches to problems in natural language processing has become almost a dogma of computational linguistics, although most researchers acknowledge that the boundary between the logic of unification and the control processes that implement it is fluid. This dogma is undoubtedly due in part to the good match which declarative programming formalisms give to many constructs of structuralist linguistics. The work reported here has shown that neglect of issues of control, however, can be held responsible for difficulties purely unification-based computational approaches have had in accounting for phenomena arising in derivational morphology.

Data that would help to specify the control component of a linguistic model are hard to obtain with the methods of structuralist linguistics alone. It was argued in chapter 1 that the goal of constructing a cognitive model cannot be merely that of correctly implementing the formal specification of a descriptive linguistic theory, any more than an application program can be seen as finished when it correctly implements a functional specification. In the terminology of software engineering, linguistic analyses can provide the basis for formally correct models, but not for valid models, because the discipline of structuralist linguistics is not the only "stakeholder" in the cognitive domain of language. In the end, even several cognitive disciplines taken together cannot name all the requirements for a valid model of language processing, because validity can ultimately only be decided by a linguistic Turing test — a test of whether the model's behavior is indistinguishable from that observable in the domain it models. But until such a test becomes possible, it seems advisable to draw on the lessons of software engineering for application programs, letting all known stakeholders contribute what they can to the model.

This study has drawn requirements for a model of word formation from three main stakeholder disciplines, namely corpus statistics, psycholinguistic studies of word recognition, and linguistic studies of derivation. The following sections summarize how data from these disciplines have contributed to the KLU model of word formation.

8.1 Contributions to the Model from Corpus Statistics

Statistical studies of large corpora have shown that some classes of words are freely generated, and that the number of words in these classes increases without limit as a function of the size of the corpus. The limitless generation of hapax legomena in large text corpora is not explainable without some sort of word-generating system, and this system could be seen as a subsystem of the sentence-generating system. Although the productive generation of words and of sentences can be described by similar formal devices, the productivity distributions at the word and sentence levels differ greatly. While sentences are rarely used twice, most words appear again and again, with hapaxes appearing only in narrowly circumscribed classes. These data have suggested that both the sentence and word-producing components of the grammar could be modeled by stochastically weighted production rules, but they have also suggested that the two sets of rules may function separately.

In the KLU implementation, rules similar to those used for sentence analysis in Lexical Functional Grammar, but without an explicit, word-internal functional structure, have been used to analyze word structures. I have further claimed that systems of stochastically weighted production rules could accurately simulate the vocabulary growth curves of large corpora, but no test of this hypothesis has been carried out. The hypothesis of dual representation, which states that a given complex word can be represented in the lexicon by both a word-formation rule and a lexical item, could be helpful in simulating the growth curves for certain classes of complex words that already contain a large number of lexicalized items.

Not all productive word formation patterns can be described by production rules that concatenate fixed word segments. Many languages show a certain amount of analogical productivity of irregular forms, like the use of **rept* as a past tense of *reap* instead of *reaped*, on the analogy of *keep – kept*. In the KLU model, irregular orthographic variants of a form can be reduced (lemmatized) to a lexical form via rules, implemented as transducers, that mediate between the lexicon and word segment candidates. A further, untested claim of the model is that adding stochastic weightings to the transducers could account for speakers' abilities to generate such analogical forms; incorporated into a stochastic model of word production, it might permit the growth curves of irregular forms to be simulated. Taken together, stochastic production rules and transducers appear to allow a symbolic treatment of important statistical properties of large corpora, and they suggest ways of modeling certain aspects of long-term changes in languages without resort to the less-informative spreading activation models that have found favor with many cognitive scientists.

8.2 Contributions from Psycholinguistics

Experimental psychological findings relating to the recognition and production of words have identified effects suggesting that some complex words can be represented in memory both as whole entities and as constituent parts that must be split and recombined during recognition. In particular, inflectional affixes often appear to be processed separately from their stems; similar but less pronounced effects have been

identified for derivational prefixes. At least one study indicates that storage of derived stems in decomposed form is most strongly evident in just those classes of words that show the highest degree of productivity in corpus-statistical studies. Other findings suggest that at a very early stage of processing pseudo-prefixes are being used separately to help identify a word, even though later stages of processing seem to regard them as part of the stem.

Taken together, these findings appear to require rather complex representations of lexical items in the lexicon. For early stages of recognition, where pseudo-prefixes can play an important role, the KLU lexicon contains an Indexical Structure used solely to steer word segmentation and to explain some morphological priming effects. Segmentation is computationally intensive and can require the examination of large numbers of candidates. The KLU model assumes that the cost of accessing a segment rises with the size of the data structure in which it is represented, and for this reason segments are represented in Indexical Structure by simple tuples, containing simple scalar features like morphotactic category, inflectional class, and pointers to semantic predicates, when they exist. KLU compiles the segments in Indexical Structure from the lexical entries, taking into account both atomic access strings and subsegments in lexical chains like /re°proach/. The segments do not contain representations of semantic or sentence-level functional information. Thus, whether an affix has morphological reality (is a unit of word meaning) or is merely a pseudo-prefix plays no role at the level of word segmentation. The pseudo-prefix "re" in /re°proach/ will be identified as a lexical segment, but it will used in recognizing the word merely as an access key to the lexical chain, while the segment "re" in *re-caress* will be identified in later processing as a genuine prefix that can enter into morphotactic analysis and derivation.

Rather than continually consulting the lexicon, one could imagine that the segmentation algorithm places possible segmentation marks between pairs of characters having a low transition probability. However, one result in psychological studies of visual word recognition indicates that segmentation is not influenced by transition probabilities between characters. This implies that the model cannot specify an independent algorithm for finding segment boundaries; rather, it should rely entirely on the access strings of the lexical entries. Naive search for an indefinite number of substrings, however, is computationally very expensive. To avoid the cost of naive search, a segmentation algorithm has been proposed that restricts the possible substrings according to the structure of the lexicon, which roughly reflects the ordering of levels in the level-ordering hypothesis of lexical phonology (which has been confirmed experimentally). It searches from the periphery of the word towards the middle, looking first for inflectional, then derivational affixes, and finally for stems and roots. Level-ordering makes it possible to reduce the search space for segment candidates because at each level only candidates from the corresponding lexical region are considered. Level-ordering also predicts that at lower levels segmentable items, like derived stems, are likely to have lexical idiosyncrasies and be represented as wholes in the lexicon. This means that once a lexical segment is found, no further segmentation needs to be attempted. This saves work because not all possible segments need to be found. At each each level a language-specific parameter defines whether the search is bi- or unidirectional.

8.3 Contributions from Linguistic Analyses

Three linguistic issues have played a large role in shaping the KLU model. These are the Lexical Integrity hypothesis, the compositionality of derivation, and the polysemy of derived words.

The Lexical Integrity hypothesis states, in essence, that word-formation does not interact directly with syntax, i. e., that the rules of word formation cannot be implemented as an extension of the syntactic system, as has been proposed for unitary, unification-based accounts of word and sentence formation. A number of syntactic relations like extraposition, anaphoric reference and coordination appear to have no counterparts in word structure. The appearance of syntactic fragments within words, as in *a c'est-la-vie-attitude* does not necessarily show that the rules of syntax and of word formation work together. It has instead been argued that these fragments are first lexicalized and then inserted as single lexical items at word formation, in some cases via the same lexical buffer that is postulated for inserting newly formed derived words.

For some newly derived words it is possible to describe an analysis entirely within the syntactic system, as has been shown for some derived verbs in German (like *aufsetzen* 'put on') that incorporate prepositions as prefixes. Other data, particularly Italian 'reversative' derived verbs and nominalizations in $-at(a)$, show that it can often be difficult to derive the semantics and argument structure of a derived word in the same framework used for analyzing sentence structure. Moreover, these data indicate that most derivations must be computationally complex and slow. Certain derivational steps appear to require particularly elaborate conceptual operations that make reference to prototypical action schematas, as in Italian *gomitata* 'blow with the elbow', and some seem to involve general, script-like knowledge as in Italian *spaghettata* 'meal with spaghetti'.

It is thus evident that many derived words, especially denominal verbs, inherit few features directly from the base. Features like inflectional class, arity, mapping of syntactic to semantic arguments, semantic structure and special subcategorization requirements may all need to be given new definitions, and these cannot always be obtained by unifying features of the base with the derivational morpheme in the way that an inflectional morpheme can be unified with its stem. Instead, costly references to conceptual knowledge are often involved. Furthermore, it appears that operations like these are less frequently used in constructing sentence meanings, and that some operations required for derivation are specific to a distinct lexical module. No general formal account has been offered for these derivational operations, although this study has been shown how two simple derivations (Italian *funghini* 'little mushrooms and *disiscrive* 'ex-matriculates') can, in principle, be implemented with queries to a conceptual data base and with substitutions to appropriately selected semantic schemata.

The meaning that can be constructed in a rule-based fashion for a derived word sometimes coexists with another, sometimes arbitrary, lexicalized meaning, as has been illustrated by German *abwandern* 'depart', which can be used to mean 'hike off' in an appropriate context. The Italian neologism *sbobinare* can be interpreted transparently as 'take from a spool', but it has been used to mean 'transcribe dictation'. This meaning, however, presumes knowledge about spools on dictation machines, which may not be available to all users of the word. The existence of such polysemy suggests that access to word meanings is not simply a matter of access to the lexicon or

to word structure. The KLU model accounts for polysemy, like other aspects of dual representation, by giving a complex, procedural specification of lexical insertion that can return several alternative representations of a word to syntax.

8.4 Conclusions and Outlook

Like the data showing that word generation has statistical characteristics much different from sentence generation, the linguistic data appear to require a separate logical module for describing word structure, and at the control level they require a procedure that mediates between syntax and the lexical module. Like the psychological data supporting dual representation, linguistic data also show that derived words can exist simultaneously in relatively fixed, lexicalized form and in a decomposed form that can be freely created and understood.

While much of word formation by itself can be described in a simple declarative fashion, for example in the LFG formalism, requirements from statistics on word formation, from experimental results in word recognition and production, and from linguistic analyses indicate that there is a barrier between word formation and syntax, and that it would be inaccurate to model both in a single system of rules. This barrier has been modeled in KLU with a procedurally defined lexical insertion algorithm, in the form of a search strategy that starts with simple items in the lexicon, continues with attempts to unify features of word segments, and can end with the generation of an insertion error for a new, derived word, which cannot be unified to an insertable structure. The apparatus that resolves the error is not a component of the linguistic system, but it can submit the non-unifiable structure of a derived word to deeper levels of the linguistic system that include rules for conceptual search, argument mapping, determination of inflectional class and gender, and the like. That at least some of these rules may be seated in a region of the cognitive system sometimes called “general intelligence” is possible; in any case, the same error handler is held responsible for figuring out abbreviations, correcting spelling errors — and for sending its owner to look the word up in a dictionary when all else fails.

A second, crucial component of the model is the lexical buffer. A newly derived word is not substituted directly to the corresponding non-terminal symbol of the syntax; rather, it is placed in a lexical buffer or cache, and it may remain there for some time. Lexicalization is thus explained as a learning effect that arises from repeated access to the lexical buffer.

It would have been difficult to specify this model in the framework of a single set of declarative rules, although, of course, as an exercise it could be given a largely logical formulation. A declarative, logic-oriented specification would, however, obscure the nature of the relation between the systems responsible for sentence and word structure, and it would violate the maxim of software engineering that one should choose one’s formalisms to suit the data (rather than choose the data to suit one’s favorite formalism). The KLU implementation is, to be sure, entirely in Prolog, but functions like lexical insertion and segmentation make considerable use of the non-declarative Prolog ‘assert’, ‘cut’, and error-handling constructs; and in fact it appears that an entirely correct implementation of the KLU model would require a threshold-sensitive cut in order to retain highly improbable but still possible alternatives, e.g., in segmentation.

What remains as a gap in the implementation is the probabilistic model of word production and recognition that was proposed in chapter 5. Simply providing for statistical weightings in the word formation rules and transducers would probably not be a large undertaking, but these would be meaningless without appropriate numerical values, and these could only be obtained by attempting to match the rules' behaviors empirically to the growth curves in appropriately tagged corpora. Many other details of the theoretical model that have been discussed in the text have no specific representation in the KLU program. In particular, the model suggests that inflected forms are likely to acquire independent lexical representations later than derived forms of the same frequency because the cost of recomputing inflectional attributes is much lower than for recomputing derived stems. To implement this aspect of the model, the KLU program needs an intelligent purging or garbage collection mechanism that would tend to purge first those items least recently used, but would also factor in the cost of analysis relative to the cost of storage. The current implementation, however, merely places every word that is analyzed in the lexical buffer, and it counts accesses to the new word as long as it remains there. When the access count exceeds a certain threshold, the word can be moved to the permanent lexicon. Another relatively unexplored issue is the actual usefulness of level-ordering in segmentation. It appears to offer a large advantage, but this may be compromised in practice by segmentation ambiguities, like that for Italian *serviti*, which can be segmented both as an imperative with a cliticized pronoun meaning 'serve yourself' and as a masculine plural participle meaning 'served'.

What the KLU program hopefully demonstrates is that such problems in word formation *can* be modeled symbolically, largely in the framework of a unification grammar, and that the modeling effort can make a serious contribution to our understanding of the underlying cognitive processes.

Bibliography

[Abe93] Anne Abeillé. *Les nouvelles syntaxes: grammaires d'unification et analyse du français*. Colin, Paris, 1993.

[AK80] Mark Aronoff and Mary-Louise Kean, editors. *Juncture: A Collection of Original Papers*. Studia Linguistica et Philologica 7. Anma Libri, Saratoga, Ca., 1980.

[Als92] Hiwan Alshawi, editor. *The Core Language Engine*. ACL-MIT Press Series in Natural Language Processing. MIT Press, Cambridge, MA, 1992.

[And86] Sally Andrews. Morphological influences on lexical access: Lexical or nonlexical effects? *Journal of Memory and Language*, 25:726–740, 1986.

[And92] Stephen R. Anderson. *A-Morphous Morphology*. Cambridge Studies in Linguistics. Cambridge U. Press, Cambridge, 1992.

[Aro76] Mark Aronoff. *Word Formation in Generative Grammar*. MIT Press, Cambridge, MA, 1976.

[Aro84] Mark Aronoff. Word formation and lexical semantics. *Quaderni di Semantica*, 5(1):45–49, 1984.

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, Reading, MA, 1986.

[Baa92] Harald Baayen. Quantitative aspects of morphological productivity. In *Yearbook of Morphology* [BvM], pages 109–149. 1991.

[Baa3a] Harald Baayen. Statistical models for word frequency distributions: A linguistic evaluation. *Computers and the Humanities*, 26:347–363, 1993a.

[Bar89] Jon Barwise. Mathematical proofs of computer system correctness. Technical Report CSLI-89-136, Center for the Study of Language and Information, Stanford, CA, 1989.

[BBS97] Harald Baayen, Cristina Burani, and Robert Schreuder. Effects of semantic markedness in the processing of regular nominal singulars and plurals in Italian. In *Yearbook of Morphology* [BvM], pages 13–33. 1996.

[BC89] Willian Badecker and Alfonso Caramazza. Priming homographic stems. *Journal of Memory and Language*, 28:531–546, 1989.

[BCF89] Avron Barr, Paul R. Cohen, and Edward A. Feigenbaum, editors. *The Handbook of Artificial Intelligence*. Addison-Wesley, Reading, MA, 1989.

[BDTL97] Cristina Burani, Francesca M. Dovetto, Anna M. Thornton, and Alessandro Laudanna. Accessing and naming suffixed pseudo-words. In *Yearbook of Morphology* [BvM], pages 55–72. 1996.

[BE89] Outi Bat-El. *Phonology and Word Structure in Modern Hebrew*. Ph. D., University of California at Los Angeles, Los Angeles, 1989.

[Bea86] J. Bear. A morphological recognizer with syntactic and phonological rules. In *COLING-86*, number 1986, pages 28–32, Bonn, BRD, 1986.

[BF95] Robert C. Berwick and Sandiway Fong. A quarter century of computation with transformational grammar. In *Linguistics and Computation*, CSLI Lecture Notes, pages 103–143. CSLI, Stanford, 1995.

[Bib93] Wolfgang Bibel. *Wissensrepräsentation und Inferenz: Eine grundlegende Einführung*. Artificial Intelligence/Künstliche Intelligenz. Vieweg, Wiesbaden, 1993.

[Bie83] Manfred Bierwisch, editor. *Semantische und konzeptionelle Repräsentation lexikalischer Einheiten*. Studia Grammatika 22. Akademie-Verlag, Berlin, 1983.

[BK82] Joan Bresnan and Ronald M. Kaplan. Introduction: Grammars as mental representations of language. In Bresnan [Bre82b], pages xvii–lii.

[Bla91] Richard E. Blahut. *Principles and Practice of Information Theory*. Addison-Wesley, Reading, MA, 1991.

[Blu94] Bruce J. Blum. A taxonomy of software development methods. *Communications of the ACM*, 37(1):82–94, 1994.

[BM95] Joan Bresnan and Samuel A. Mchombo. The lexical integrity principle: Evidence from Bantu. *Natual Language and Linguistic Theory*, 13:181–254, 1995.

[BM98] Ira D. Baxter and Michael Mehlich. Reverse engineering is reverse forward engineering. *Software Engineering Technical Council Newsletter*, 15(3-4):Reeng–7–14, 1998.

[Boe86] Barry W. Boehm. A spiral model of software development and enhancement. *IEEE Computer*, 21(5):61–72, 1986.

[Boi88] Patrice Boizumault. *Prolog: l'implantation*. Etudes et recherches en informatique. Masson, Paris, 1988.

[BR6b] R. Harald Baayen and Antoinette Renoulf. Chronicling the times: Productive lexical innovation in an English newspaper. *Language*, 72(1):69–96, 1996b.

[Bra80] Dianne Bradley. Lexical representation of derivational relation. In Aronoff and Kean [AK80], pages 37–55.

[Bra86] Ivan Bratko. *Prolog Programming for Artificial Intelligence*. International Computer Science Series. Addison-Wesley, Wokingham, 1986.

[Bre82a] Joan Bresnan. Control and complementation. In *The Mental Representation of Grammatical Relations* [Bre82b], pages 282–390.

[Bre82b] Joan Bresnan, editor. *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA, 1982.

[Bre82c] Joan Bresnan. The passive in lexical theory. In *The Mental Representation of Grammatical Relations* [Bre82b], pages 3–86.

[BS85] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216, 1985.

[BS6a] Harald Baayen and Richard Sproat. Estimating lexical priors for low-frequency morphologically ambiguous forms. *Computational Linguistics*, 22(2):155–166, 1996a.

[BVC97] Kersti Börjars, Nigel Vincent, and Carol Chapman. Paradigms, periphrases and pronominal inflection: a feature-based account. In *Yearbook of Morphology* [BvM], pages 155–180. 1996.

[BvM] Gert Booij and Jaap van Marle, editors. *Yearbook of Morphology*. Kluwer, Dordrecht.

[Byb88] Joan Bybee. Morphology as lexical organization. In Hammond and Noonan [HN88], pages 119–141.

[Byb95] Joan Bybee. Diachronic and typological properties of morphology and their implications for representation. In Feldman [Fel95], pages 225–246.

[BZ90] Joan Bresnan and Annie Zaenen. Deep unaccusativity in LFG. In *Grammatical Relations: A Cross-Theoretical Perspective*, pages 45–57. CLSI, Stanford, 1990.

[CH68] Noam Chomsky and Morris Halle. *The Sound Pattern of English*. Studies in Language. Harper & Row, New York, 1968.

[Cho57] Noam Chomsky. *Syntactic Structures*. Mouton, The Hague, 1957.

[Cho65] Noam Chomsky. *Aspects of the Theory of Syntax*. MIT Press, Cambridge, MA, 1965.

[Cho70] Noam Chomsky. Remarks on nominalization. In *Readings in English Transformational Grammar*, pages 184–221. Ginn, Waltham, MA, 1970.

[Cla97] Harald Clahsen. The representation of participles in the German mental lexicon: Evidence for the dual-mechanism model. In *Yearbook of Morphology* [BvM], pages 73–95. 1996.

[CLR88] Alfonso Caramazza, Alessandro Laudanna, and Cristina Romani. Lexical access and inflectional morphology. *Cognition*, 28(3):297–332, 1988.

[CMBW96] Harald Clahsen, Gary Marcus, Susanne Bartke, and Richard Wiese. Compounding and inflection in German child language. In *Yearbook of Morphology* [BvM], pages 115–142. 1995.

[CMSL85] Alfonso Caramazza, Gabriele Miceli, M. Caterina Silveri, and Alessandro Laudanna. Reading mechanisms and the organization of the lexicon: Evidence from acquired dyslexia. *Cognitive Neuropsychology*, 2(1):81–114, 1985.

[Cor90] Danielle Corbin. Associativité et stratification dans la représentation des mots construits. In *Contemporary Morphology*. de Gruyter, Berlin, 1990.

[Cur88] Susan Curtis. Abnormal language acquisition and grammar: Evidence for the modularity of language. In Larry M. Hyman and Charles N. Li, editors, *Language, Speech and Mind: Studies in Honour of Victora A. Fromkin*, pages 81–102. Routledge, London, 1988.

[Dav93] Alan M. Davis. *Software Requirements: Objects, Functions, and States*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[Del93] Rodolfo Delmonte. Computational morphology for Italian. Technical Report 87.01681.93, Unipress, 1993.

[Den80] Peter J. Denning. Working sets past and present. *IEEE Transactions on Software Engineering*, SE-6(Jan.):64–84, 1980.

[Des62] René Descartes. *Discours de la méthode [1637]. Texte et commentaire*. Librarie Philosophique. J. Vrin, 1962.

[Dij68] Edsger Dijkstra. The structure of the "THE" multiprogramming system. *Communications of the ACM*, 11(55):341–346, 1968.

[DKK+87] Mary Dalrymple, Ronald M. Kaplan, Lauri Karttunen, Kimmo Koskenniemi, et al. Tools for morphological analysis. Technical Report CSLI-87-108, Center for the Study of Language and Information, 1987.

[DKMZ95] Mary Dalrymple, Ronald M. Kaplan, John T. III Maxwell, and Annie Zaenen, editors. *Formal Issues in Lexical-Functional Grammar*. CSLI Lecture Notes 47. CSLI Publications, Stanford, 1995.

[Dow79] David Dowty. *Word Meaning and Montague Grammar: The semantics of verbs and times in generative semantics and in Montague's PTQ*. Reidel, Dordrect, 1979.

[Dre86] Hubert L. Dreyfus. *What Computers Can't Do*. Harper & Row, N.Y., 1986.

[Dre89] Etta Drews. *Die Bedeutung von Morphemen für die Sprachanalyse: Zur mentalen Verarbeitung lexikalischer und grammatischer Morpheme*. Psycholinguistische Studien. Westdeutscher Verlag, Opladen, 1989.

[Dro83] Günther Drosdowski, editor. *Duden Deutsches Universalwörterbuch*. Dudenverlag, Mannheim, 1983.

[Dür91] Christa Dürscheid. *Modelle der Satzanalyse: Überblick und Vergleich*. Kölner Linguistische Arbeiten - Germanistik 16. Gabel, Köln, 1991.

[DW87] A. M. DiSciullo and E. Williams. *On the Definition of Word*. MIT Press, Cambridge, MA, 1987.

[ED86] Andreas Eisele and Jochen Dörre. A Lexical Functional Grammar system in Prolog. In *Proceedings of Coling '86, 11th International Conference on Computational Linguistics*, number 1986, pages 551–553, N.Y., 1986. Association for Computational Linguistics.

[EPS+95] Urs Egli, Peter E. Pause, Christoph Schwarze, Arnim v. Stechow, et al., editors. *Lexical Knowledge in the Organization of Language*. Current Issues in Linguistic Theory 114. Benjamins, Amsterdam, 1995.

[Fel95] Laurie Beth Feldman, editor. *Morphological Aspects of Language Processing*. Erlbaum, Hillsdale, 1995.

[FF90] Gisbert Fanselow and Sascha W. Felix. *Sprachtheorie: Eine Einführung in die Generative Grammatik*. Franke, Tübingen, 1990.

[Fod81] Jerry A. Fodor. *Representations: Philosophical Essays on the Foundations of Cognitive Science*. MIT Press, Cambridge, MA, 1981.

[Fro71] Victoria A. Fromkin. The non-anomalous nature of anomalous utterances. *Language*, 47:27–52, 1971.

[Fro87] Victoria A. Fromkin. The lexicon: Evidence from acquired dyslexia. *Language*, 63(4):1–22, 1987.

[FS92] Uli H. Frauenfelder and Robert Schreuder. Constraining psycholinguistic models of morphological processing and representation: the role of productivity. In *Yearbook of Morphology* [BvM], pages 165–183. 1991.

[GA82] Henri Guiter and M. V. Arapov, editors. *Studies on Zipf's Law*. Quantitative Linguistics 16. Brockmeyer, Bochum, 1982.

[GKPvC85] F. Giannesini, H. Kanoui, R. Pasero, and M. van Caneghem. *Prolog*. InterÉditions, Paris, 1985.

[Gle61] H. A. Gleason, Jr. *An Introduction to Descriptive Linguistics*. Holt, Rinehart & Winston, New York, 1961.

[GM89] Gerald Gazdar and Chris Mellish. *Natural Language Processing in Prolog: An Introduction to Computational Linguistics*. Addison-Wesley, Wokingham, 1989.

[Gor85] Peter Gordon. Level-ordering in lexical development. *Cognition*, 21:73–93, 1985.

[Gri90] Jane Grimshaw. *Argument Structure*. Linguistic Inquiry Monographs 18. MIT Press, Cambridge, MA, 1990.

[GT91] Tiziana Gatti and Lucia Togni. A proposito dell'interpretazione dei derivati in *-ata* e in *s-*. Technical Report 30, FG Sprachwissenschaft, University of Konstanz, 1991.

[Gün88] Hartmut Günther. Oblique word forms in visual word recognition. *Linguistics*, 26:583–600, 1988.

[Hal83] Per-Kristian Halvorsen. Semantics for Lexical-Functional Grammar. *Linguistic Inquiry*, 14(4):567–615, 1983.

[Han92] Jorge Hankamer. Morphological parsing and the lexicon. In Marslen-Wilson [MW92b], pages 392–408.

[Har95] Trevor A. Harley. *The Psychology of Language: From Data to Theory*. Erlbaum(UK) Taylor & Francis, Hove, UK, 1995.

[Hau98] Roland Hausser. Häufigkeitsverteilung deutscher Morpheme. *LDV-Forum*, 15(1):6–28, 1998.

[HBL83] K. Hess, J. Brustkern, and Winfried Lenders. *Maschinenlesbare deutsche Wörterbücher*. Niemeyer, Tübingen, 1983.

[HJ80] K. Hofland and S. Johansson. *Word Frequencies in British and American English*. Longman, London, 1980.

[HK95] Per-Christian Halvorsen and Ronald M. Kaplan. Projections and semantic description in Lexical-Functional Grammar. In Dalrymple et al. [DKMZ95], pages 279–292.

[HMT89] Robert M. Haralick, Alan K. Mackworth, and Steven L. Tanimoto. Computer vision update. In Barr et al. [BCF89], pages 521–582.

[HN88] Michael Hammond and Michael Noonan, editors. *Theoretical Morphology: Approaches in Modern Linguistics*. Academic, San Diego, 1988.

[Jac90] Ray Jackendoff. *Semantic Structures*. MIT Press, Cambridge, MA, 1990.

[Jac92] Ivar Jacobson. *Object-Oriented Software Engineering*. Addison-Wesley, Reading, MA, 1992.

[Jac97] Ray Jackendoff. *The Architecture of the Language Faculty*. Linguistic Inquiry Monographs 28. MIT Press, Cambridge, MA, 1997.

[JC87] Marcel Adam Just and Patricia A. Carpenter. *The Psychology of Reading and Language Comprehension*. Allyn & Bacon, Newton, MA, 1987.

[Kan85] Siegfried Kanngießer. Strukturen der wortbildung. In *Handbuch der Lexikologie*, pages 134–183. Athenäum, Königstein/Ts., 1985.

[Kap95a] Ronald M. Kaplan. The formal architecture of Lexical Functional Grammar. In Dalrymple et al. [DKMZ95], pages 7–27.

[Kap95b] Ronald M. Kaplan. Three seductions of computational psycholinguistics. In Dalrymple et al. [DKMZ95], pages 339–367.

[Kar83] Lauri Karttunen. Kimmo: A general morphological processor. *Texas Linguistic Forum*, 2:165–189, 1983.

[KB95] Ronald M. Kaplan and Joan Bresnan. Lexical-Functional Grammar: A formal system for grammatical representation. In Dalrymple et al. [DKMZ95], pages 29–135.

[KF67] H. Kučera and W. N. Francis. *Computational Analysis of Present-day English*. Brown Univ. Press, Providence, 1967.

[Kha90] Tarum Khanna. *Foundations of Neural Networks*. New Horizons in Technology. Addison-Wesley, Reading, MA, 1990.

[Kip82a] Paul Kiparsky. From cyclic phonology to lexical phonology. In Harry van der Hulst and Norval Smith, editors, *The structure of phonological representations*, Linguistic Models, pages 131–175. Foris, Dordrecht, 1982.

[Kip82b] Paul Kiparsky. Word-formation and the lexicon. In Frances Ingemann, editor, *1982 Mid-America Linguistics Conference*, number 1982, pages 3–29, Lawrence, KS, 1982. Linguistics Dept., Univ. of Kansas.

[KJ94] Jean-Pierre Koenig and Daniel Jurafsky. Type underspecification and on-line type construction in the lexicon. In Raul Aranovich, Willian Byrne, Susanne Preuss, and Martha Senturia, editors, *Thirteenth West Coast Conference on Formal Linguistics*, number 1994, pages 270–285, University of Califorina at San Diego, 1994. Stanford: CLSI.

[KK94] Ronald M. Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, 1994.

[KM96] Ronald M. Kaplan and John T. III Maxwell. Grammar writer's workbench. Technical Report 3.1, Xerox Corp., 1996.

[Kos83] Kimmo Koskenniemi. Two-level morphology: A general computational model for word-form recognition and production. Technical Report 11, Department of General Linguistics, University of Helsinki, Helsinki, 1983.

[Kow74] Robert A. Kowalski. Predicate logic as a programming language. In *Proceedings IFIP 1974*, number 1974, pages 569–574. North Holland, 1974.

[Kow79] Robert Kowalski. Algorithm = logic + control. *Communications of the ACM*, 22(7):424–436, 1979.

[KP74] Brian W. Kernighan and Kenneth Plauger. *The Elements of Programming Style*. McGraw-Hill, N.Y., 1974.

[KR93] Hans Kamp and Uwe Reyle. *From Discourse to Logic*. Studies in linguistics and philosophy 42. Kluwer, Dordrecht, 1993.

[Kuh62] Thomas S. Kuhn. *The Structure of Scientific Revolutions*. Univ. of Chicago Press, Chicago, 1962.

[KuMvBF93] Jürgen Kunze and unter Mitarbeit von Beate Firzlaff. *Sememstrukturen und Feldstrukturen*. studia grammatica 36. Akademie Verlag, Berlin, 1993.

[Kun95] Jürgen Kunze. Reflexive Konstruktionen im Deutschen. *Zeitschrift für Sprachwissenschaft*, 14(1):3–53, 1995.

[Lan87] Ewald Lang. Semantik der Dimensionsauszeichnung räumlicher Objekte. In *Grammatische und konzeptuelle Aspekte von Dimensionsadjectiven*, studia grammatica, pages 287–458. Akademie-Verlag, Berlin, 1987.

[LB93] Rochelle Lieber and Harald Baayen. Verbal prefixes in Dutch: a study in lexical conceptual structure. In *Yearbook of Morphology* [BvM], pages 51–78. 1992.

[Lev74] Willem J. M. Levelt. *Formal Grammars in Linguistics and Psycholinguistics*. Mouton, The Hague, 1974.

[Lev89] Willem J. M. Levelt. *Speaking: From Intention to Articulation*. ACL-MIT Press Series in Natural Language Processing. MIT Press, Cambridge, MA, 1989.

[Lie92] Rochelle Lieber. *Deconstructing Morphology: Word Formation in Syntactic Theory*. University of Chicago Press, Chicago, 1992.

[Lin94] Richard C. Linger. Cleanroom process model. *IEEE Software*, 11(2):50–58, 1994.

[LKFT83] G. Lukatela, A. Kostic, Laurie B. Feldman, and M. T. Turvey. Grammatical priming of inflected nouns. *Memory and Cognition*, 11(1):59–63, 1983.

[LP81] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall Software Series. Prentice-Hall, Englewood Cliffs, 1981.

[Mar82] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman, San Francisco, 1982.

[May93] Bruce Mayo. Eine Werkbank für Linguisten: Die Konstanzer LFG-Umgebung. Technical Report 55, FG Sprachwissenschaft, University of Konstanz, 1993.

[May95] Bruce Mayo. Describing verbs of motion in Prolog. In Egli et al. [EPS+95], pages 203–243.

[May96a] Bruce Mayo. Computer simulation as a tool for descriptive linguistics. In Peter Blumenthal, Giovanni Rovere, and Christoph Schwarze, editors, *Lexikalische Analyse romanischer Sprachen.*, Linguistische Arbeiten 353, pages 73–83. Niemeyer, Tübingen, 1996.

[May96b] Bruce Mayo. Die Konstanzer LFG-Umgebung. *LDV-Forum*, 13(1/2):31–53, 1996.

[May97] Bruce Mayo. Die Konstanzer LFG-Umgebung. Technical Report 82, FG Sprachwissenschaft, University of Konstanz, Mai 1997.

[Mei98] Judith Meinschaefer. *Silbe und Sonorität in Sprache und Gehirn*. Dr. Phil., Ruhr-Universität Bochum, Bochum, FRG, 1998.

[MJL76] George A. Miller and Philip N. Johnson-Laird. *Language and Perception*. Harvard, Cambridge, MA, 1976.

[MK95] John T. III Maxwell and Ronald M. Kaplan. The interface between phrasal and functional constraints. In Dalrymple et al. [DKMZ95], pages 403–429.

[Mor69] John Morton. Interaction of information in word processing. *Psychological Review*, 76:165–178, 1969.

[Mor79] John Morton. Facilitation in word recognition: Experiments causing change in the logogen model. In *Processing of Visible Language*, pages 259–268. Plenum, New York, 1979.

[MR86] James L. McClelland and David E. Rumelhart, editors. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Computational Models of Cognition. MIT Press, Cambridge, MA, 1986.

[MSSZ95] Bruce Mayo, Marie-Theres Schepping, Christoph Schwarze, and Angela Zaffanella. Semantics in the derivational morphology of Italian: Implications for the structure of the lexicon. *Linguistics*, 33:883–938, 1995.

[MT77] L. Manelis and D. A. Tharp. The processing of affixed words. *Memory and Cognition*, 5:690–695, 1977.

[Mul77] Charles Muller. *Principes et méthods de statistique lexicale*. Langue, linguistique, communication. Hachette, Paris, 1977.

[MW92a] William Marslen-Wilson. Access and integration: Projecting sound onto meaning. In *Marslen-Wilson 1992* [MW92b], pages 3–24.

[MW92b] William Marslen-Wilson, editor. *Lexical Representation and Process*. MIT Press, Cambridge, MA, 1992.

[MWTWO94] William Marslen-Wilson, Lorraine Komisarjevksy Tyler, Rachelle Waksler, and Lianne Older. Morphology and meaning in the English mental lexicon. *Psychological Review*, 101(1):3–33, 1994.

[MWZ89] William Marslen-Wilson and Piene Zwitserlood. Accessing spoken words: The importance of word onsets. *Journal of Experimental Psychology*, 15(3):576–585, 1989.

[Nes85] Marina Nespor. The phonological word in Italian. In *Advances in Nonlinear Phonology*, pages 193–204. Foris, Dordrecht, 1985.

[NMC95] Dennis Norris, James M. McQueen, and Anne Cutler. Competition and segmentation in spoken-word recogniton. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 21(5):1209–1228, 1995.

[NV86] Marina Nespor and Irene Vogel. *Prosodic Phonology*. Studies in Generative Grammar 28. Foris, Dordrecht, 1986.

[Orl82] Ju. K. Orlov. Ein Modell der Häufigkeitsstruktur des Vokabulars. In Guiter and Arapov [GA82], pages 154–233.

[Par90] Helmut. A. Partsch. *Specification and Transformation of Programs: A Formal Approach to Software Development*. Texts and Monographs in Computer Science. Springer, Berlin, 1990.

[PF92] Fernando Palazzi and Gianfranco Folena. *Dizionario della lingua italiana*. Loescher, Turin, 1992.

[Pin95] Steven Pinker. *The Language Instinct*. HarperPerennial, New York, 1995.

[PP88] Steven Pinker and Alan Prince. On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. In *Connections and Symbols*, Cognition Special Issues, pages 73–193. MIT Press, Cambridge, MA, 1988.

[Pro96] PrologIA. *Prolog II+*. Marseille, 1996.

[PS87] Carl Pollard and Ivan A. Sag. *Information-Based Syntax and Semantics*. CSLI Lecture Notes 13. CSLI, Menlo Park, CA, 1987.

[PTA94] Colin Potts, Kenji Takahashi, and Annie I. Antón. Inquiry-based requirements analysis. *IEEE Software*, 11(2):21–32, 1994.

[Pul96] Stephen G. Pulman. Unification encodings of grammatical notations. *Computational Linguistics*, 22(3):295–327, 1996.

[Pus95] James Pustejovsky. *The Generative Lexicon*. MIT Press, Cambridge, MA, 1995.

[PW80] Fernando C. N. Pereira and David H. D. Warren. Definite clause grammar formalisms for language analysis — a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13:231–278, 1980.

[Qua87] Claudio Quarantotto. *Dizionario del nuovo italiano: 8000 neologismi della nostra lingua*. Manuali moderni 24. Newton Compton, Rome, 1987.

[Rad11] W. Radlof. *Versuch eines Wörterbuches der Türk-Dialecte*. Glasunoff, St. Petersburg, 1892-1911.

[Rai66] Gordon Raisbeck. *Information Theory: An Introduction for Scientists and Engineers*. MIT Press, Cambridge, MA, 1966.

[Rap82] Anatol Rapoport. Zipf's law re-visited. In Guiter and Arapov [GA82], pages 1–28.

[Red90] James W. Redhouse. *A Turkish and English Lexicon*. Boyajian, Constantinople, 1890.

[RM86] David E. Rumelhart and James L. McClelland. On learning the past tenses of English verbs. In Jerome A. Feldman, Patrick J. Hayes, and David E. Rumelhart, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Computational Models of Cognition, pages 216–271. MIT Press, Cambridge, MA, 1986.

[RRBP92] Graeme D. Ritchie, Graham J. Russell, Alan W. Black, and Stephen G. Pulman, editors. *Computational Morphology: Practical Mechanisms for the English Lexicon*. ACL-MIT Press Series in Natural Language Processing. MIT Press, Cambridge, MA, 1992.

[RS93] Gert Rickheit and Hans Strohner. *Grundlagen der kognitiven Sprachverarbeitung: Modelle, Methoden, Ergebnisse*. Uni-Taschenbücher 1735. Francke, Tübingen, 1993.

[San94] Dominiek Sandra. *Morphology in the reader's mental lexicon*. Duisburger Arbeiten zur Sprach- und Kulturwissenschaft 21. Peter Lang, Frankfurt, 1994.

[SB95] Robert Schreuder and R. Harald Baayen. Modeling morphological processing. In Feldman [Fel95], pages 131–154.

[Sch85] Christoph Schwarze, editor. *Beiträge zu einem konstrastiven Wortfeldlexikon Deutsch-Französisch* . Narr, Tübingen, 1985.

[Sch92] Marie-Theres Schepping. Zur Grammatik der Präpositionen. Technical Report 51, FG Sprachwissenschaft, University of Konstanz, 1992.

[Sch95a] Christoph Schwarze. *Grammatik der italienischen Sprache*. Niemeyer, Tübingen, 1995.

[Sch95b] Christoph Schwarze. Semantic and conceptual structures in word formation. In Manfred Bierwisch and Peter Bosch, editors, *Semantic and Conceptual Knowledge*, number 1995 in Arbeitspapiere des Sonderforschungsbereichs 340, Sprachtheoretische Grundlagen der Computer-Linguistik, pages 211–221, Stuttgart, December 1995. University of Stuttgart.

[Sch96] Christoph Schwarze. Lexikalisch-Funktionale Grammatik. Eine Einführung in 10 Lektionen mit französischen Beispielen. Technical Report 76, FG Sprachwissenschaft, University of Konstanz, Konstanz, April 1996.

[Sch97a] Marie-Therès Schepping. Die negation der wortbildung am beispiel der italienischen präfix *s-*/*dis-*. FG Sprachwissenschaft, University of Konstanz, Aug. 1997.

[Sch97b] Christoph Schwarze. Repräsentation und Variation: Zur Entwicklung der romanischen Auxiliarsyntax. Technical Report 85, FG Sprachwissenschaft, University of Konstanz, Konstanz, 1997.

[Sch99] Christoph Schwarze. Lexical-functional morphology and the structure of the lexicon. In Lunella Mereu, editor, *Boundaries of Morphology and Syntax*, number 1997 in Current Issues in Linguistic Theory, pages 73–95, Amsterdam, 1999. Benjamins.

[SD92] Nikolaus Schpak-Dolt. *Einführung in die französische Morphologie*. Romanische Arbeitshefte 36. Niemeyer, Tübingen, 1992.

[Sha48] Claude E. Shannon. A mathematical theory of information. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[Sha81] Roger C. Shank. *Inside Computer Understanding: Five programs plus miniatures*. Erlbaum, Hillsdale, NJ, 1981.

[SHC85] Sargur N. Srihari, Jonathan J. Hull, and Ramesh Choudhari. Integrating diverse knowledge sources in text recognition. In *Tutorial: Computer Text Recognition and Error Correction*, pages 68–87. IEEE Computer Society, Los Angeles, 1985.

[Shi85] Stuart Shieber. Criteria for designing computer facilities for linguistic analysis. *Linguistics*, 23:189–211, 1985.

[Shi88] Stuart M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Lecture Notes. CSLI, Stanford, 1988.

[SL97] Vieri Samek-Lodovici. A unified analysis for Italian nominalizations in *-ata*. FG Sprachwissenschaft, University of Konstanz, Oct. 1997.

[SL98] Vieri Samek-Lodovici. A unified analysis of noun- and verb-based Italian nominalizations in *-ata*. Technical Report 80, FG Sprachwissenschaft, University of Konstanz, Konstanz, 1998.

[SL6a] Vieri Samek-Lodovic. Complex words: Nominalizations in -ATA. Technical report, FG Sprachwissenschaft, University of Konstanz, Konstanz, July 1996a.

[SL6b] Vieri Samek-Lodovici. On *-ata* nominalization. FG Sprachwissenschaft, University of Konstanz, Aug. 1996b.

[SM88] Joseph Paul Stemberger and Brian MacWhinney. Are inflected forms stored in the lexicon? In *Theoretical Morphology: Approaches in Modern Linguistics*, pages 101–116. Academic, San Diego, 1988.

[Sol86] Elliot Soloway. Learning to program = learning to construct mechanisms and explanations. *Communications of the ACM*, 29(9):850–858, 1986.

[Spr88] Richard Sproat. On anaphoric islandhood. In *Theoretical Morphology: Approaches in Modern Linguistics*, pages 291–301. Academic, San Diego, 1988.

[Ste95] Joseph Paul Stemberger. Phonological and lexical constraints on morphological processing. In Feldman [Fel95], pages 247–267.

[Sti96] Barbara Stiebels. *Lexikalische Argumente und Adjunkte*. studia grammatica 39. Akademie-Verlag, Berlin, 1996.

[Sti6b] Barbara Stiebels. Complex denominal verbs in German and the morphology-semantics interface. Technical Report 78, Heinrich-Heine-Universität, Düsseldorf, March 1996b.

[SZR91] Herbert Schriefers, Pienie Zwitserlood, and Ardi Roelofs. The identification of morphologically complex spoken words: Continuous processing or decomposition? *Journal of Memory and Language*, 30:26–47, 1991.

[Taf79a] Marcus Taft. Lexical access via an orthographic code: The basic orthographic syllabic structure (BOSS). *Journal of Verbal Learning and Verbal Behavior*, 18:21–39, 1979.

[Taf79b] Marcus Taft. Recognition of affixed words and the word frequency effect. *Memory and Cognition*, 7(4):263–272, 1979.

[Taf88] Marcus Taft. A morphological-decomposition model of lexical representation. *Linguistics*, 26:657–667, 1988.

[TF75] Marcus Taft and Kenneth I. Forster. Lexical storage and retrieval of prefixed words. *Journal of Verbal Learning and Verbal Behavior*, 14:638–647, 1975.

[TF76] Marcus Taft and Kenneth I. Forster. Lexical storage and retrieval of polymorphemic and polysyllabic words. *Journal of Verbal Learning and Verbal Behavior*, 15:607–620, 1976.

[Tro91] Harald Trost. X2morph: A morphological component based on two-level morphology. Technical Report RR-91-04, Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslauten, FRG, January 1991.

[Tur63] Alan M. Turing. Computing machinery and intelligence. In *Computers and Thought*, pages 11–35. McGraw-Hill, N.Y., 1963.

[Win84] Patrick Henry Winston. *Artificial Intelligence*. Addison-Wesley, Reading, MA, 1984.

[Wir71] Niklaus Wirth. Program development by stepwise refinement. *Communications of the ACM*, 14(4):221–227, 1971.

[Wir84] Niklaus Wirth. *Compilerbau: Eine Einführung*. Teubner, Stuttgart, 1984.

[WKNW72] William A. Woods, Ronald M. Kaplan, and Bonnie Nash-Webber. The lunar sciences natural language information system: Final report. Technical Report 2378, Bolt, Beranek and Newman, Inc., Cambridge, MA, 1972.

[Wun96] Dieter Wunderlich. Minimalist morphology: the role of paradigms. In *Yearbook of Morphology* [BvM], pages 93–114. 1995.

[Zip65] G. K. Zipf. *The Psychobiology of Language: An Introduction to Dynamic Philology.* M.I.T. Press, Cambridge, MA, 1965.

[Zwi92] Arnold Zwicky. Some choices in the theory of morphology. In Steven Davis, editor, *Formal Grammar: Theory and Implementation*, Vancouver Studies in Cognitive Science, pages 327–371. Oxford, N.Y., 1992.