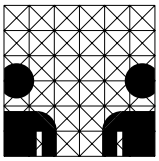


Maschinelles Dolmetschen mit Mehr-Ebenen-Charts

Dissertation

zur Erlangung des Grades eines
Doktors der Naturwissenschaften
am Fachbereich Informatik
der Universität Hamburg

von
Jan Willers Amtrup

aus
Hamburg

9. Juni 1998

Danksagung

Diese Arbeit ist am Arbeitsbereich Natürlichsprachliche Systeme des Fachbereichs Informatik an der Universität Hamburg entstanden. Ohne die mir dort gebotenen Möglichkeiten wäre sie nicht fertig geworden. Ich danke Wolfgang Menzel für die Betreuung meiner Arbeit und seine stetige Bereitschaft, Teile davon zu diskutieren. Walther von Hahn hat mich während meines Studiums mit dem Gebiet der Sprachverarbeitung bekannt gemacht, was mich trotz einer nicht-linguistischen Vorbildung dazu veranlaßt hat, meine Spezialisierung auf diesem Gebiet zu suchen. Günther Görz schließlich hat einen wesentlichen Teil meiner Ausbildung bestritten, indem er mich für Strukturanalyse und deren architektonische Aspekte interessiert hat. Christopher Habel hat dankenswerterweise die Anfertigung des Drittgutachtens übernommen.

Dreieinhalb Jahre meiner Zeit als Mitarbeiter in Hamburg war ich im Projekt Verbmobil beschäftigt. In diesem Rahmen habe ich sehr viele mir nützliche Gespräche führen können und sehr viel von anderen gelernt, insbesondere von Hans Weber. Für meine Dissertation unmittelbar wichtig ist der Hamburger Worterkenner und die Arbeit, die Uwe Jost und Henrik Heine investiert haben, ihn mir zugänglich zu machen. Die in dieser Arbeit beschriebenen Experimente sind zum großen Teil auf Maschinen des Computing Research Labs der New Mexico State University gerechnet worden.

Den intensivsten Kontakt zu meiner Arbeit haben Susanne und Volker gehabt. Ihnen beiden danke ich ausdrücklich.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Algorithmen

Kapitel 0

Einleitung

Das menschliche Sprachverstehen arbeitet inkrementell. Dies bedeutet, daß Personen Teile der akustischen Eingabe bereits verarbeiten, bevor diese noch vollständig vorliegt, und sie z.B. in einem Dialog den Redebeitrag des Partners antizipieren können, ehe er ausgeredet hat. Simultandolmetscher — um ein hier relevantes Beispiel zu nennen — arbeiten inkrementell, indem sie mit geringer Verzögerung Äußerungen in die Zielsprache übersetzen.

Die Einführung inkrementeller Strategien in die maschinelle Verarbeitung von Sprache ist folglich bereits aus dem Gesichtspunkt der Verarbeitungsadäquatheit sinnvoll; sobald von Maschinen ähnlich natürliche Leistungen erwartet werden wie der Mensch sie erbringen kann, müssen auch dort zwangsläufig inkrementelle Algorithmen verwendet werden.

Allerdings geschieht dies bisher nur zögerlich. Das liegt vor allem daran, daß Inkrementalität zunächst eine Erhöhung des Aufwands zur Bearbeitung einer Eingabe mit sich bringt, da durch den fehlenden rechten Kontext weniger Sicherheit bezüglich des Ganges der Verarbeitung besteht. Eine damit verbundene Erhöhung der Verarbeitungszeit läßt sich erst durch die Ausnutzung der nun bestehenden Möglichkeiten zur Parallelisierung vermeiden, eine qualitative Verbesserung der einzelnen Arbeitsschritte erst durch die Konstruktion alternativer Informationspfade zur gegenseitigen Beeinflussung erzielen.

In den letzten Jahren wird jedoch zunehmend versucht, die Einsetzbarkeit inkrementeller Verfahren innerhalb sprachverarbeitender Systeme zu prüfen; vornehmlich geschieht dies im Bereich der Systeme zur Verarbeitung spontan gesprochener Sprache, und hier insbesondere im Rahmen von Dolmetschsystemen. Erste richtungweisende Ergebnisse liegen bereits vor.

Das in der vorliegenden Arbeit entwickelte System MILC (*M*achine *I*nterpreting with *L*ayered *C*harts, Maschinelles Dolmetschen mit Mehr-Ebenen-Charts) stellt den Versuch dar, ein durchgängig inkrementelles System zur Übersetzung spontan gesprochener Sprache zu entwerfen und prototypisch zu implementieren. Jede Komponente des Systems beginnt mit der Bearbeitung seiner Eingabe, bevor letztere abgeschlossen ist. Dabei werden Teile der Eingabe strikt in der Reihenfolge der Sprechzeit angenommen. Daraus resultiert ein insgesamt inkrementelles Verhalten der Anwen-

dung, die bereits zielsprachlichen Text generieren kann, bevor der quellsprachliche Dialogpartner seinen Redebeitrag beendet hat.

Um ein solches System in integrierter und uniformer Weise konstruieren zu können, wurde eine neue Datenstruktur, die Mehr-Ebenen-Chart, entwickelt, die — analog zur herkömmlichen Betrachtungsweise einer Chart während der Strukturanalyse oder der Generierung — zur Speicherung von Zwischenergebnissen auf allen Ebenen benutzt wird. Außerdem wurde ein getypter komplexer Merkmalformalismus implementiert, der einerseits die Wartbarkeit der Wissensquellen verbessert und andererseits den Austausch linguistischer Objekte zwischen Modulen des Systems erleichtert. Beide Maßnahmen erlauben eine homogene Sicht auf den Gesamtzustand des Systems zu jedem Zeitpunkt der Verarbeitung und sichern die Möglichkeit zur Untersuchung nichttrivialer Interaktionsmuster zwischen den Komponenten.

Durch die erfolgreiche Implementation wird nachgewiesen, daß ein konzeptionell stringentes, integriertes System zur inkrementellen Übersetzung spontan gesprochener Sprache für ein eingeschränktes Korpus realisierbar ist. Zusätzlich ist hierdurch die Grundlage für weitergehende Untersuchungen zur Architektur inkrementeller sprachverarbeitender Systeme gegeben, etwa durch die Neueinführung weiterer Komponenten. Exemplarisch ist dies gezeigt durch die Einbeziehung der inkrementellen Erkennung und Übersetzung von idiomatischen Ausdrücken.

Diese Arbeit ist wie folgt organisiert:

- Kapitel 1 führt in detaillierter Weise in den Gegenstandsbereich der Untersuchung ein. Die inkrementelle Arbeitsweise des menschlichen Sprachverstehens und die Anwendung in der Sprachverarbeitung werden vorgestellt.
- Kapitel 2 gibt einen Überblick über den Teil der Graphentheorie, der in der Verarbeitung gesprochener Sprache eine besonders große Rolle spielt und faßt einige Ergebnisse zur Evaluation von Spracherkennung zusammen. Außerdem werden Hypergraphen präsentiert, als grundlegende effiziente Datenstruktur zur Repräsentation hochgradig redundanter Erkennungsergebnisse.
- Kapitel 3 führt in die Verwendung unifikationsbasierter Formalismen in der Sprachverarbeitung, insbesondere in Übersetzungssystemen, ein. Der für die vorliegende Arbeit implementierte, getypte Merkmalformalismus wird vorgestellt.
- Kapitel 4 beschreibt die Architektur und Implementierung des Systems MILC. Dabei wird näher auf die globale Architektur, die Kommunikation von Komponenten sowie auf ausgewählte Eigenschaften einzelner Module eingegangen.
- Kapitel 5 stellt die zur Evaluation des System verwendeten Experimente vor. Ebenfalls werden die unterschiedlichen Wissensquellen, die für MILC eine Rolle spielen, präsentiert.
- Kapitel 6 faßt die zentralen Ergebnisse dieser Arbeit zusammen und versucht, einen Ausblick auf sich anbietende weitere Forschungsvorhaben im gewählten Zusammenhang zu geben.

Kapitel 1

Einführung

In diesem Kapitel wird zunächst eine formale Definition von Inkrementalität vorgenommen. Davon ausgehend werden die Auswirkungen des Einsatzes in sprachverarbeitenden Systemen diskutiert. Als grundlegende Motivation wird die Fähigkeit des Menschen, Sprache inkrementell zu verstehen, genauer ausgeführt. Schließlich wird die grobe Architektur des dieser Arbeit zugrunde liegenden Systems dargestellt.

1.1 Inkrementelle Sprachverarbeitung

Drei Eigenschaften sprachverarbeitender Systeme haben in den letzten zehn Jahren immer mehr an Bedeutung gewonnen: Modularität, Parallelität und Inkrementalität. Erst die Anwendung softwaretechnischer Prinzipien wie der Modularität erlaubt den Bau großer Anwendungen (Sommerville, 1996), auch in der Sprachverarbeitung. Sie fördert die Wiederverwendung, die Stabilität und langfristig ebenfalls die Effizienz einzelner Module und erlaubt grundsätzlich erst die Skalierung von Forschungsprototypen in Richtung auf kommerziell einsetzbare Systeme (Zajac, Casper, und Sharples, 1997). Die Verteilung der eingesetzten Algorithmen auf mehrere Prozessoren durch die Inkorporation paralleler Verfahren ist u.E. essentiell für die Entwicklung genügend performanter umfangreicher Systeme. Die Implementierung paralleler Module zur Sprachverarbeitung ist allerdings eher eine Randerscheinung, obwohl es gerade im Bereich der massiv parallelen Systeme lohnende Ansätze gibt (Kitano, 1994). Die Anwendung inkrementeller Verfahren in großem Maßstab befindet sich allerdings trotz fortgesetzter Bemühungen immer noch im Anfangsstadium, insbesondere bezüglich der Auswirkungen auf die Architektur sprachverarbeitender Systeme.

Abgrenzungs- und Definitionsunterschiede der Begriffe zur Einteilung von Systemen sind in vielen Fällen Ursache von Mißverständnissen. Dies gilt insbesondere, wenn Zusammenhänge zwischen verschiedenen architektonischen Facetten gezogen werden können. Im folgenden wird deshalb zur

akkuraten Eingrenzung des Gegenstands eine Definition einiger Begriffe gegeben. Dabei spielt naturgemäß die Definition der Inkrementalität die entscheidende Rolle, sie wird formal genau ausgeführt.

Zunächst muß zwischen einem *monolithischen* und einem *modularen* System unterschieden werden. Monolithische Systeme besitzen keine softwaretechnische Trennung nach funktionalen oder anderen Gesichtspunkten. Sie erscheinen dem Betrachter ohne innere Struktur auf einer höheren als der atomaren Ebene zu sein. Modulare Systeme hingegen besitzen mehrere Funktionseinheiten, die weitgehend voneinander isoliert arbeiten und Informationen einzig über definierte Schnittstellen austauschen. In dem Schaubild in Abb. 1.1 wird eine solche Funktionseinheit gezeigt (anhand derer wir unten die Eigenschaft Inkrementalität formal definieren), ein größeres, dann modulares System wird durch Komposition derartiger Einheiten modelliert. Alle relevanten Systeme, die in dieser Arbeit betrachtet werden, sind (natürlich) modular.

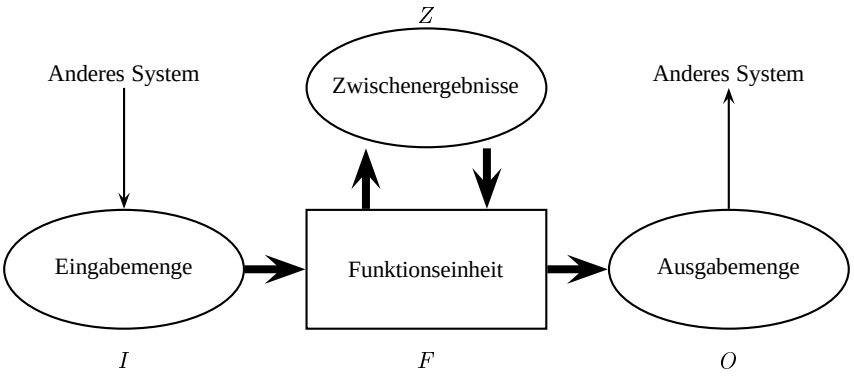


Abbildung 1.1: Die Architektur eines inkrementellen Systems

Während die Modularität ein strukturelles Kriterium zur Betrachtung und Klassifikation von Systemen darstellt, ist die Frage, wieviele Eingabeelemente zur Zeit bearbeitet werden, ein Aspekt, der Relevanz für das Systemverhalten besitzt. Ein *sequentielles* System hat in dieser Dimension die Eigenschaft, daß zu jedem Zeitpunkt höchstens ein Eingabedatum bearbeitet wird. Ein *paralleles* System hingegen ist dazu in der Lage, mehr als einen Eingabewert zu betrachten, es gibt also mindestens einen Zeitpunkt, zu dem zwei Eingabewerte gleichzeitig prozessiert werden. Die simultan durchgeführte Arbeit in einem System kann sich in einer *intramodularen* Parallelität äußern. Dabei ist die Funktionseinheit, die das Modul repräsentiert, in mehrere Bearbeitungseinheiten unterteilt, die teilweise autonom arbeiten können. Grundsätzlich muß hier unterschieden werden zwischen einer Parallelität, die durch algorithmische Mittel erbracht wird und datengetriebener Parallelität. Ein Beispiel für die Einbeziehung algorithmischer Bezüge ist die parallele Unifikation von Merkmalstrukturen (Hager und Moser, 1989). Datengetriebene Parallelität entsteht dadurch, daß mehrere Bearbeitungseinheiten vorgesehen werden, die unterschiedliche Ausschnitte der Eingabedaten und Zwischenergebnisse bearbeiten. Die Aufteilung einer Äußerung in einzelne Teile oder die Parti-

tionierung einer Grammatik sind Beispiele hierfür (Amtrup, 1992), ebenso wie eine Klasse von Algorithmen, die auf dem Vergleich der Eingabe mit bereits vorher gesehenen Beispielen beruht (sog. *memory-based* oder *example-based* Systeme).

Im Gegensatz zu dieser Funktionseinheit-internen Parallelität steht die *intermodulare* Parallelität. Sie entsteht, wenn zwei Funktionseinheiten (Module) gleichzeitig Eingabedaten bearbeiten (z.B. wenn die syntaktische und die semantische Analyse in separaten Komponenten erfolgen). Um dies zu gewährleisten, müssen gleichzeitig für mehrere Module Eingabedaten vorliegen. Dies ist im allgemeinen Fall nur dann möglich, wenn inkrementell gearbeitet wird.

Inkrementalität bedeutet umgangssprachlich, daß die Eingabedaten nicht als Gesamtheit betrachtet werden, sondern "stückchenweise" bearbeitet werden, also mit der Verarbeitung einzelner Teile der Eingabedaten begonnen wird, bevor der gesamte Input in der Funktionseinheit bekannt ist. Die im Zusammenhang mit der vorliegenden Arbeit wichtigste Form ist die *zeitliche Inkrementalität* (auch Links-Rechts- oder LR-Inkrementalität). LR-inkrementelle Systeme operieren auf Daten, die zeitlich ausgedehnt sind, wie es in sprachverarbeitenden Systemen durch die zeitliche Folge von Eingabewörtern bzw. bei der Analyse gesprochener Sprache durch die zeitliche Abhängigkeit des Schalldrucks gegeben ist. Prinzipiell denkbar sind ebenfalls andere Aufteilungen der Eingabedaten (z.B. nach strukturellen Gesichtspunkten), hier werden diese Ausprägungen jedoch keine große Rolle spielen.[1]

Die Maxime inkrementeller Sprachverarbeitung für die Analyse ist, die Beschreibung der Eingabe in Intervalle uniformer oder verschiedener Länge aufzuteilen. Auf der Signalebene ist die Einteilung z.B. durch die Abtastrate der Vorverarbeitung gegeben, im Normalfall 10 ms. Auf höherer Ebene ist eine nichtuniforme zeitliche Ordnung z.B. durch die Endzeitpunkte von Worthypothesen oder phrasalen Hypothesen gegeben.[2] Ein System wird dadurch LR-inkrementell, daß die Eingabedaten in strenger zeitlicher Ordnung verarbeitet werden und ein Zurückgehen in der Zeit nicht oder nur über sehr eng begrenzte Zeitfenster möglich ist.

Aus Gründen der Relevanz soll nun der Begriff der Inkrementalität und seine Unterteilung in die Inkrementalität der Eingabe, der Ausgabe und des Gesamtsystems formal dargelegt werden. Als generelles Modell, anhand dessen wir die Begrifflichkeiten erklären wollen, diene die in Abbildung 1.1 angegebene Systemarchitektur (vgl. Amtrup, 1997c). F sei eine Funktionseinheit, die ein Resultat durch Anwendung gewisser Operationen auf Eingabewerte produziert. Die Eingabedaten für F seien gegeben durch eine Menge I von Eingabeelementen $A_i, i \in \{1, \ldots, n\}$. Jedes der A_i sei unabhängig von anderen Elementen der Eingabemenge, die Ankunftsreihenfolge ist also nicht durch inhaltliche Abhängigkeiten vorherbestimmt, die in der Arbeit von F begründet liegen.[3]

[1] Im Compilerbau wird Inkrementalität verwendet, um jeweils nur die sich ändernden Programmteile neu umzuwandeln, vgl. z.B. Marchetti-Spaccamela, Nanni, und Rohnert (1992).

[2] Hierdurch wird in vielen Fällen den Eingabedaten zusätzlich eine strukturelle Inkrementalität aufgeprägt, die u.a. auf der Kompositionalität einiger Verfahrensweisen beruht.

[3] Selbstverständlich kann eine externe Abhängigkeit durch die Methodik der vorhergehenden Funktionseinheit gegeben sein, die in diesem Modell aber irrelevant ist.

Die Menge O enthält die Resultate, die von F produziert wurden. O wird analog zu I konstruiert und besteht aus Ausgabeelementen $B_j, j \in \{1, \ldots, m\}$. Zwischenresultate $C_k, k \in \{1, \ldots, p\}$ werden in der Menge Z gespeichert. Wir nehmen an, daß keine Operation von F auf I oder O ausgeführt wird, daß also die Verarbeitung entweder den Zustand von Z ändert oder ein neues Resultat in O ablegt. Diese Vereinfachung kann ohne Beschränkung der Allgemeinheit des Modells angenommen werden. Neue Zwischenergebnisse werden durch die Anwendung einer Funktion f produziert, während eine andere Funktion g für die Erzeugung von Resultaten zuständig ist:

$$C_k := \quad f(D_k), \quad D_k \in I^* \times Z^* \tag{1.1}$$

$$B_j := \quad g(D_j), \quad D_j \in I^* \times Z^* \tag{1.2}$$

Typischerweise ist g eine einfache Auswahlfunktion, die Zwischenergebnisse aus Z als Resultate lizensiert und in O ablegt. Wir modellieren das inkrementelle Eintreffen von Eingabedaten dadurch, daß in Abb. 1.1 die A_i von einem anderen System in einer arbiträren Reihenfolge geliefert werden und in I erscheinen, d.h., es gibt einen Strom von Eingabewerten, der I füllt. Jedes von F produzierte Resultat B_j wird unmittelbar nach Auftreten in der Ausgabemenge O an ein folgendes System weitergegeben, das an den Ausgabekanal von F angeschlossen ist. Eine LR-inkrementelle Verarbeitungsweise ergibt sich durch die zusätzliche Eigenschaft des Eingabestroms, die Hypothesen in zeitlicher Ordnung zu liefern.[4]

Der Zustand des Systems F kann nun durch den Inhalt der drei Mengen I, O und Z beschrieben werden:

> Zu jedem Zeitpunkt t mit $0 \leq t < \infty$ sei der Systemzustand von F durch die aktuellen Inhalte von $I(t)$, $O(t)$ und $Z(t)$ definiert.

Für den Verarbeitungszustand von F gibt es zwei Extremsituationen. Im Anfangszustand sind alle drei Mengen leer:

$$I(0) = \emptyset \wedge O(0) = \emptyset \wedge Z(0) = \emptyset$$

Die Bearbeitung der Eingabedaten ist beendet, wenn alle Ausgabeelemente produziert wurden, d.h.,

$$t_{\text{End}} := t : I(t) = \{A_1, \ldots, A_n\} \wedge O(t) = \{B_1, \ldots, B_m\} \wedge Z(t) = \{C_1, \ldots, C_p\}.$$

Nun lassen sich drei verschiedene Arten von Inkrementalität unterscheiden[5]:

[4] So sind z.B. die Worthypothesen eines Wortgraphen üblicherweise nach ihren Endzeitpunkten sortiert (die Worterkennung hypothetisiert Wortenden), also partiell geordnet.

[5] Zu einer informellen Beschreibung dieser Typen vgl. Kilger (1994)

- *Inkrementelle Eingabe*. Das System beginnt mit der Bearbeitung von Eingabewerten, ohne auf das vollständige Vorliegen des gesamten Inputs zu warten.

  $$\exists t : \#(I(t)) \neq n \wedge Z(t) \neq \emptyset$$

  Eine mögliche Anwendungsform für diese Art der inkrementellen Verarbeitung wäre ein System zur Beantwortung von Ja/Nein-Fragen. Die linguistische Analyse der Anfrage kann bereits beginnen, bevor die Eingabe komplett ist. Die Beantwortung kann allerdings vom letzten Eingabeelement abhängen, so daß eine Reaktion nicht vor Betrachtung aller Eingabedaten möglich ist.

- *Inkrementelle Ausgabe*. Einige Elemente der Ausgabemenge werden weitergegeben, bevor die gesamte Verarbeitung abgeschlossen ist, d.h. bevor I und Z ihren endgültigen Inhalt besitzen:

  $$\exists t : \#(Z(t)) \neq p \wedge O(t) \neq \emptyset$$

  Als Beispiel für ein System mit derartigem Verhalten diene eine Vorlesemaschine, die einen ganzen Satz einliest, und schritthaltend mit der Berechnung des Signalverlaufes bereits einzelne Wörter verliest.

- *Inkrementelles System*. Ein solches System bearbeitet Teile der Eingabe und beginnt bereits mit der Ausgabe von Resultaten, bevor der gesamte Satz von Eingabedaten im System vorliegt:

  $$\exists t : \#(I(t)) \neq n \wedge O(t) \neq \emptyset.$$

  Ein inkrementeller Parser, der Wort für Wort die zu analysierende Äußerung einliest und partielle Hypothesen über deren grammatische Struktur ausgibt, stellt einen Vertreter dieser Art von Systemen dar.[6]

Inkrementalität bedeutet also zumindest, daß die Eingabedaten in kleinen Portionen bearbeitet werden, kann allerdings im Extremfall auch bedeuten, daß Eingabe und Ausgabe eines Systems überlappen. Ein wesentliches Beschreibungsmaß für inkrementelle Systeme ist die *Inkrementgröße*, also die Fraktion der Gesamteingabedaten, die als Portion für die Verarbeitung dient. Dies kann von sehr kleinen (Abtastzeitpunkten) bis hin zu sehr großen Einheiten (vollständigen Redebeiträgen in Dialogen) variieren. Dabei kann die faktische zeitliche Ausdehnung eines Inkrements selbstverständlich unterschiedlich groß sein. Auch wenn die Ausgaben eines Moduls in streng zeitlicher Reihenfolge initiiert werden, so ist es durchaus möglich, daß die Hypothesen unterschiedliche Zeitintervalle abdecken, so etwa im Falle von variabel langen Konstituenten, die von einem Parser geliefert werden. Eine gut aufeinander abgestimmte Kombination von Inkrementgrößen innerhalb eines komplexen, insgesamt inkrementell arbeitenden, Systems ist essentiell für das Gelingen eines solchen Unterfangens. Eine einfache Methode, die Kombination von Komponenten und Inkrementgrößen zu bewerten, besteht in der Messung des Nachlaufes. Unter dem Nachlauf zwischen zwei Modulen

[6] Es ist möglich, inkrementelle Eingabe und inkrementelle Ausgabe so miteinander zu kombinieren, daß sich daraus kein inkrementelles System ergibt. In solchen Systemen gibt es einen Zeitpunkt t, so daß $\#(I(t)) = n \wedge \#(Z(t)) \neq p \wedge O(t) = \emptyset$. Eingabe und Ausgabe überlappen sich also nicht.

wird die Differenz der augenblicklichen Verarbeitungszeitpunkte verstanden (Pyka, 1992c). Im Idealfall (wenn jedes Eingabeelement unmittelbar und schnell bearbeitet werden kann) ist dies durch die Inkrementgröße des vorgeschalteten Moduls bestimmt. Der Gesamtnachlauf des Systems besteht in diesem Fall aus der Summe der einzelnen Nachlaufzeiten.[7] Dieser Idealfall tritt allerdings in praktischen Anwendungen nicht auf, die Bestimmung des Systemnachlaufes reicht allenfalls zur qualitativen Abschätzung (vgl. Weber, Amtrup, und Spilker, 1997). Im Regelfall ist die Verarbeitungszeit (und die Übermittlungszeit), die eine Komponente für ein Inkrement aufwendet, nicht vernachlässigbar, sondern hat gewichtige Auswirkungen auf das Verhalten der Anwendung. Zu klein gewählte Inkrementgrößen können dazu führen, daß der Kontext so stark vernachlässigt werden muß, daß der Erfolg der Bearbeitung insgesamt in Frage gestellt werden kann — z.B. bei einer Übersetzung, die Wort für Wort stattfindet, ohne einen größeren Zusammenhang herzustellen. Zum anderen werden Module, die einen hohen Aufwand in die Verarbeitung von Eingaben investieren, durch eine Flut von Hypothesen nicht mehr dazu in der Lage sein, zeitlich adäquat zu reagieren. Zu große Inkremente auf der anderen Seite bringen eventuell eine mangelnde Auslastung von Modulen in einem verteilten, parallelen System mit sich. In der vorliegenden Arbeit wird versucht, die Größen von Inkrementen möglichst klein zu halten, um den Nachlauf zu minimieren, ohne dabei Qualität und Performanz wesentlich zu beeinträchtigen.

Wünschenswert ist eine Ausdehnung der obigen Zusammenhänge, die von einer endlichen Eingabemenge ausgingen, auf kontinuierlich arbeitende Systeme, da dadurch eine begrenzte Analogie zum Wesen des menschlichen Sprachverstehensapparates möglich wird, der während der Lebensspanne einer Person ständig arbeitet. Allerdings erweist sich die Erweiterung als nur bedingt sinnvoll. Zunächst gilt es nämlich festzuhalten, daß ein kontinuierlich arbeitendes System gemessen an der umgangssprachlichen Definition “Produziere Ausgabedaten, während die Eingabe noch nicht vollständig ist” trivialerweise inkrementell ist, da die Eingabe nie abgeschlossen ist.

Eine korrekte Handhabung erfordert denn auch die Identifikation zusammenhängender Bereiche in den Ein- und Ausgabedaten. Die Betrachtung des Systemverhaltens wird auf Intervalle eingeschränkt, die innerhalb des Stroms von Eingabewerten einen Bereich bilden, der logisch zusammenhängende Daten enthält. Innerhalb dieser Bereiche können dann die hier präsentierten Definitionen angewendet werden. Wo die Grenzen solcher Intervalle zu ziehen sind, hängt dabei im Einzelfall stark von der Intuition des Betrachters ab. Wenn formale Systeme gegeben sind, lassen sich die Intervalle durch den symmetrischen Abschluß von f und g bezüglich der Eingaben und der Zwischenergebnisse konstruieren. Ein Zeitintervall ist dann lokal zusammenhängend, wenn f und g innerhalb des Intervalls operieren und zusätzlich das Intervall minimal ist. Auf einen solchen Bereich lassen sich dann die oben getroffenen Definitionen anwenden. Bei der Betrachtung biologischer Systeme wie dem Menschen ist eine Fassung der Intervallgrenzen schwierig. Auch offensichtliche Abgrenzungen können hier täuschen. So wird man z.B. Schlafperioden zunächst als sichere Grenzen für den Prozeß der Spracherkennung des Menschen ansehen. Allerdings ist dieser Prozeß durchaus von Erfahrungen abhängig, die sehr viel weiter zurückreichen, z.B. von der Frequenz, mit der Wörter bisher gehört wurden (vgl. Shillcock, 1990).

[7] Es sei hier erneut angemerkt, daß wir ausschließlich Inkrementalität zeitlicher Ausprägung betrachten. In auf strukturellen Gesichtspunkten basierenden Systemen sind Konzepte wie der zeitliche Nachlauf nicht sinnvoll.

Mithin wird eine Eingrenzung auf bestimmte zeitliche Abschnitte stets eine Abstraktion bezüglich gewisser Eigenschaften der zu untersuchenden Realitäten vornehmen und die Granularitätsfrage zusätzlich komplizieren. So ist im vorliegenden Fall die obere Grenze der Verarbeitung der Redebeitrag eines kooperativen Sprechers innerhalb eines Dialogs zur Termin-Vereinbarung. Alle Wissensquellen, die strukturell größere Zusammenhänge beschreiben, wie etwa eine Dialogmodellierung, ein weit zurückreichendes Weltwissen etc., werden nicht betrachtet, obwohl auch sie einen Einfluß auf die Auswertung von Redebeiträgen haben.

Ein System inkrementell zu gestalten, bringt zunächst nur Nachteile mit sich (Ausiello et al., 1991). Abgesehen von der Tatsache, daß inkrementelle Algorithmen vielfach komplizierter sind als nichtinkrementelle[8], fehlt die Möglichkeit zur globalen Optimierung, da kein rechter Kontext vorliegt. Für die in der vorliegenden Arbeit relevante Form der linguistischen Verarbeitung von Wortgraphen (vgl. Abschnitt 2.2) heißt dies, daß der akustisch bestbewertete Pfad nicht bekannt ist, ja im Extremfall noch nicht einmal vorliegt.[9]

Eine weitere Folge von Inkrementalität kann eine starke Vergrößerung des Suchraumes einzelner Module und des Gesamtsystems sein. Dies zeigt sich erneut bei der Erzeugung von Wortgraphen in einem Worterkenner und deren Weiterverarbeitung in einem syntaktischen Parser. Wenn Wortgraphen inkrementell erzeugt werden, ist die Eliminierung von "toten Enden", Knoten des Graphen also, von denen ausgehend keine ausreichend gut bewerteten Worthypothesen mehr gefunden werden können, nicht möglich; diese Operation wird in konventionellen Worterkennungssystemen in einem Rückwärtsschritt nach Erkennung der vollständigen Äußerung durchgeführt. Dies hat zur Konsequenz, daß der nachgeordnete Parser bei der Verfolgung von Hypothesen nicht im Voraus entscheiden kann, ob ein Pfad innerhalb des Graphen jemals zu einer vollständigen Hypothese ausgebaut werden kann. Es wird also Energie in die Bearbeitung von Teilpfaden investiert, die möglicherweise nicht zu einem globalen Endzustand führen.

Isoliert betrachtet ist Inkrementalität folglich unerwünscht. Es ergeben sich hingegen gewichtige Vorteile ihres Einsatzes, wenn man die sich durch die Kombination von Modulen in einem System ergebenden Alternativen und die Auswirkungen der inkrementellen Arbeitsweise in Rechnung stellt. Die Inkrementalität eröffnet architektonische und algorithmische Möglichkeiten, die das Potential haben, sowohl die Qualität als auch die Performanz von Verarbeitungsprozessen zu erhöhen. Aus im Rahmen dieser Arbeit durchgeführten Experimenten ergibt sich, daß durch die Ausnutzung derartiger Methoden die inkrementelle Bearbeitung von Sprache in Performanzbereiche vordringen kann, die sich an die herkömmlicher Verfahren annähern (vgl. Abschnitt 5.3).

Eine unmittelbare Folge inkrementellen Arbeitens in einem modularen System ist die Gelegenheit zur Einführung intermodularer Parallelität, welche direkt zur Steigerung der Performanz genutzt

[8] Zumindest ist es nicht immer leicht, die Aktualisierungsalgorithmen effizient zu konzipieren, vgl. Ramalingam und Reps (1992)

[9] Nicht bekannt bedeutet in diesem Zusammenhang, daß zwar der Präfix des bestbewerteten Pfades in der Ausgabe eines Erkenners vorhanden ist, andere Präfixe aber in der aktuellen Situation noch besser sind; es kann bei sehr langen Wörtern jedoch auch ein Zustand eintreten, der zur Folge hat, daß der bestbewertete Pfad noch nicht bis zum aktuellen Verarbeitungszeitpunkt konstruiert werden konnte.

werden kann. Nur wenn eine Funktionseinheit Ausgabedaten produziert, solange die Bearbeitung insgesamt noch nicht abgeschlossen ist, und die folgende Funktionseinheit beginnt, inkrementell auf diesen Daten zu arbeiten, entsteht die Möglichkeit, daß beide Module echt parallel arbeiten.[10]

Dies bedeutet allerdings weder, daß modulare inkrementelle Systeme grundsätzlich parallel ablaufen, noch daß ein nichtinkrementelles System immer auf Parallelität verzichten müßte.[11] Inkrementalität und Sequentialität sind in weiten Bereichen orthogonal.

Die zweite Dimension architektonischer Modifikationen, die durch inkrementelle Verarbeitung eröffnet wird, ist die der Interaktivität. Interaktive Systeme weisen die Fähigkeit auf, zusätzlich zu dem Hauptdatenstrom zwischen Modulen, der in einer Richtung fließt (*feed forward*), ebenfalls solche Datenströme vorzusehen, die entgegengesetzt zu dieser Richtung operieren. Das Motiv für die Durchführung einer solchen Interaktion ist stets die Einflußnahme eines Moduls auf die Arbeitsweise anderer Module, die an der Berechnung beteiligt sind (u.a. sog. *Top-Down* Interaktionen). Ein Einfluß kann aber natürlich nur dann stattfinden, wenn die Bearbeitung der Eingabedaten im Zielmodul der Interaktion noch nicht abgeschlossen ist. Wesentlich hierbei ist die Wahl der Inkrementgröße; so kann selbst ein System zur Verarbeitung von gesprochenen Dialogen, das die Analyse in Einheiten von Redebeiträgen der Dialogpartner (*Turns*) durchführt, als inkrementell auf Turnebene betrachtet werden, solange z.B. die Dialogverarbeitung Hinweise auf die mögliche Struktur des nächsten Turns geben kann. Durch die Rückkopplung zwischen Modulen kann dabei zusätzliche Funktionalität gewonnen werden, so ist es etwa möglich, Hypothesen zurückzuweisen oder Vorhersagen zu produzieren.[12]

Bezogen auf das Beispiel inkrementeller Erkennung mit Wortgraphen könnte z.B. der Suchraum des Erkenners reduziert werden, indem Worterkennung und syntaktische Analyse verschränkt ablaufen. Dies ist immer dann der Fall, wenn der syntaktische Parser mit dem Worterkenner Schritt hält. Falls ein Teilpfad innerhalb des Graphen nicht syntaktisch analysiert werden kann, darf der sich an die letzte Worthypothese anschließende Teilsuchraum abgeschnitten werden. Dies kann zu einer Performanzsteigerung des Worterkenners verwendet werden (Hauenstein und Weber, 1994; Fink, Kummert, und Sagerer, 1994).

Generell gilt, daß durch die frühe Hypothesenbildung und -verifikation in Komponenten Teilbereiche der Suchräume vorgeschalteter Module abgeschnitten werden können. Außerdem ist nur bei verschränkter Arbeitsweise und enger Interaktion die Auswertung von Prädiktionen möglich. Prädiktionen liefern dabei nicht nur binäre Entscheidungen, die die Weiterverfolgung von bestimmten Hypothesen verhindern, sondern sie können gezielt inhaltliche Vorgaben für andere Komponenten sein, so daß die Struktur des jeweiligen Suchraumes verändert wird und Ergebnisse in anderer Ordnung produziert werden können. Im hier gewählten Beispiel ist es denkbar, die Menge an

[10] Die einzige Ausnahme hiervon ist der Spezialfall, daß ein Modul Ausgabedaten produziert und zwei identische Kopien an zwei weitere Module liefert, die unabhängig voneinander auf diesen Daten operieren.

[11] Intramodulare Parallelität ist immer möglich, in der Tat werden heutzutage fast alle Programme auf sehr tiefer Ebene (Prozessor-Pipelining) bereits parallel ausgeführt.

[12] Selbstverständlich kann eine Rückkopplung auch zu Schwingungen und unerwünschten Resonanzen führen, welche ein durchaus schwerwiegendes Problem darstellen.

syntaktisch zulässigen Folgewörtern zu jedem Zeitpunkt zu generieren, um dem Worterkenner ein eingeschränktes Vokabular zu liefern. Die Folge davon ist, daß nicht lediglich bestimmte Teile des Suchraums abgeschnitten werden, sondern daß die Gestalt des Suchraumes selbst verändert wird (Weber, 1995, S. 64ff.).

Im vorliegenden System wird keine Interaktion unter Benutzung von Prädikitionen durchgeführt. Der intermodulare Einfluß, den Komponenten bei inkrementeller Arbeitsweise aufeinander ausüben können, wird durch die Analyse und Übersetzung von Idiomen demonstriert. Sie wird separat durchgeführt, indem schnelle Graphensuchalgorithmen verwendet werden und führt zur Modifikation von Bewertungen von Worthypothesen und den darauf aufbauenden phrasalen Kanten in den Modulen zur tiefen linguistischen Verarbeitung.

1.2 Inkrementelles Sprachverstehen

Wir haben bereits erwähnt, daß sich diese Arbeit an dem Ziel orientiert, analog zur menschlichen Sprachverarbeitung inkrementelle Verfahren für das Dolmetschen zu konzipieren und zu implementieren. Die praktischen Auswirkungen, die die Inkrementalität auf die eigene Art, Sprache zu hören und zu verarbeiten hat, kann jeder anhand eigener Anschauung feststellen, indem er Wörter bereits erkennt, bevor sie vollständig vom Dialogpartner ausgespro...[13]

Eine Theorie, die diese frühe Phase der auditiven Spracherkennung erklärt, ist das sog. Kohortenmodell von Marslen-Wilson (Marslen-Wilson und Welsh, 1978; Marslen-Wilson und Tyler, 1980). Es geht von der kompetitiven Auswahl eines Wortes aus einer Menge von zahlreichen Kandidaten aus. Das hauptsächliche Kriterium, nach dem ausgewählt wird, ist das eingehende akustische Signal. Bereits mit den ersten 150 Millisekunden Sprachsignal wird eine initiale *Kohorte*, eine Menge von Wörtern, die akustisch mit dem Signal verträglich sind, gebildet. Im weiteren Verlauf des Hörens werden Mitglieder aus der Kohorte eliminiert, da sie mit dem akustischen Input nur noch schlecht übereinstimmen. Bleibt schließlich lediglich ein Element in der Kohorte bestehen, so wird angenommen, dies sei das geäußerte Wort. Die lexikalische Entscheidung wird gefällt und das Wort erkannt. Diese Art der Worterkennung führt dazu, daß im Einzelfall Wörter bereits antizipiert werden, bevor der Hörer das akustische Signal vollständig aufgenommen hat. Mit Hilfe von *Cross-modal Priming*-Experimenten läßt sich dies gut nachweisen. Dazu werden Versuchspersonen Stimuli (hier: Wortfragmente) angeboten und überprüft, ob der Stimulus zu einer besseren (d.h. in der Regel schnelleren) Erkennung von ausgewählten Zielkonzepten führt. Um gegenseitige Einflüsse von Stimulus und Zielkonzept auszuschließen, werden die Zielkonzepte in einer anderen Modalität (z.B. visuell auf einem Bildschirm) angeboten. Falls ein Zielkonzept, das eine große semantische Nähe zu einem Stimulus aufweist, schneller erkannt wird als ein unabhängiges, so kann geschlossen werden, daß die Bedeutung des Stimulus aktiviert wurde. Zwitserlood (1989) benutzt als Stimulus z.B. das Fragment [capt] und weist eine bessere Reaktion auf das semantisch verwandte [ship] nach, selbst wenn der Stimulus nur unvollständig dargeboten wurde (eigentlich: [captain]).

[13] Hier ist der auditive Kanal selbstverständlich nur simuliert.

Diese frühe Fassung des Kohortenmodells ist bezüglich einiger Fragestellungen allerdings problematisch. Die Betonung der Bedeutung der wortinitialen Kohorte sowie der unbedingte Ausschluß von Kandidaten aus der Menge potentiell erkannter Wörter sind die Ursachen hierfür. In der Theorie führt ein sehr früher Abschnitt des akustischen Signals zur Konstruktion der Kohorte (*lexical access phase*, lexikalischer Zugriff). Dies impliziert prinzipiell, daß vor der Worterkennung eine Segmentierung des Sprachsignals in einzelne Wörter vorgenommen worden sein muß. Die Einteilung der akustischen Eingabe in Wörter ist jedoch keine triviale Aufgabe, nicht für artifizielle Systeme zur Worterkennung, deren Leistung beim Übergang von isolierten Wörtern mit deutlichen Pausen zu kontinuierlich gesprochener Sprache stark sinkt, und ebenfalls nicht für Menschen, die mitunter statt

> It's easy to recognize speech **(1.1)**

verstehen

> It's easy to wreck a nice beach[14] **(1.2)**

Eine angenommene Vorsegmentierung ist ebenfalls unverträglich mit der Erkennung von Subwörtern in Stimuli. Damit ist gemeint, daß Versuchspersonen Teile von Wörtern erneut als eigenständige Wörter hypothetisieren. So führt z.B. die Präsentation des Stimulus [trombone] dazu, daß [bone] als eigenständiges Wort aktiviert wird, und das, obwohl es zum einen eingebetteter Bestandteil des vollständigen Wortes ist, mithin also bei vorliegender Segmentierung nicht zur Erzeugung einer wortinitialen Kohorte führen dürfte, und zum anderen, obwohl [trom] kein für sich genommen eigenständiges Wort darstellt.

Das zweite Problem betrifft die Modellierung der nächsten Stufe der Worterkennung. Diese Phase, *lexical selection* (lexikalische Auswahl), besteht darin, den Umfang der Kohorte aufgrund akustischer (und anderer) Evidenz zu reduzieren, um schließlich eine Auswahl treffen zu können. Nach Marslen-Wilson und Welsh (1978) ist die Zugehörigkeitsentscheidung eine binäre, bei Deviation der Modelle im Lexikon von der tatsächlichen akustischen Realisierung werden Kandidaten aus der Kohorte entfernt. Dieses Verfahren hat zur Folge, daß Erkennungsfehler eigentlich zum Scheitern der Worterkennung führen müßten. Solche Fehler können etwa auf mangelhafter Artikulation beruhen oder aufgrund von Störgeräuschen auftreten. In der Tat ist es allerdings so, daß selbst unvollständige Wörter noch gut erkannt werden, bei deren akustischer Präsentation vor dem Entscheidungspunkt (dem Zeitpunkt, zu dem die Anzahl der Kandidaten in der Kohorte auf eins reduziert wurde, vor dem folglich noch Ambiguität besteht bezüglich der potentiell erkannten Wörter) Teile durch Rauschen überlagert oder gar ersetzt wurden (*phoneme restoration effect*, vgl. Warren, 1970).

Eine Modifikation der Theorie, die von Marslen-Wilson (1987) vorgenommen wurde, beschreibt die menschliche Performanz der Worterkennung in den Phasen des lexikalischen Zugriffs und der

[14] Zit. nach Shillcock (1990).

Auswahl besser: Die Kompetition bezüglich der Mitgliedschaft in einer Kohorte ist nun gegeben durch kontinuierliche Bewertungen von Worthypothesen, die nicht ausschließlich zum Zeitpunkt des Wortanfangs initialisiert werden können. So läßt sich hier schlüssig erklären, warum richtig [cigarette] erkannt wird, auch wenn [shigarette] artikuliert wurde.

Anhand des *phoneme restoration effect* kann ebenfalls geschlossen werden, daß die Perzeption akustischer Signale stark durch lexikalisches Wissen unterstützt wird. In *signal detection tests* klassifizieren Versuchspersonen Stimuli danach, ob eine Ersetzung oder Überlagerung eines Phonems durch Rauschen vorlag. Durch die Art der Versuchsanordnung wurden postperzeptive Prozesse unterdrückt, so daß der Einfluß des Lexikons auf die Signalerkennung ermittelt werden konnte. Dieser manifestiert sich durch die schlechtere Diskrimination von Ersetzung und Überlagerung beim Hören von Wörtern im Gegensatz zum Hören von Nichtwörtern. Samuel (1990) folgert, daß entweder akustische Perzeption und Lexikon eine funktionale Einheit bilden (ein Modul) oder beide Module stark interagierend angelegt sind (und im Sinne der hier verwendeten Nomenklatur Top-Down-Interaktionen ausführen).

Linguistisch höhere Ebenen (Syntax, Semantik) haben dagegen i.A. keinen Einfluß auf die Perzeption. Dies gilt ebenso für die Phase des lexikalischen Zugriffs im Kohortenmodell. Es läßt sich nicht generell nachweisen, daß syntaktische Einschränkungen bei der Einführung der in eine Kohorte zu integrierenden Wörter eine Rolle spielen. Hier scheint die Modularitätshypothese von Fodor (1983) zu gelten. So werden z.B. bei der Präsentation von

They all rose **(1.3)**

sowohl die verbale als auch die nominale Lesart von rose aktiviert, was mit cross-modal priming gezeigt wurde (Tanenhaus, Leiman, und Seidenberg, 1979, zit. nach Shillcock und Bard, 1993). Allerdings gilt dieses Resultat lediglich für offene Wortklassen. Falls im aktuellen syntaktischen Kontext eine geschlossene Wortklasse als Kategorie für das nächste zu hörende Wort hypothetisiert wird, so werden auch nur die zu der entsprechenden Klasse gehörenden Wörter aktiviert (Shillcock und Bard, 1993). So wird z.B. bei Betrachtung der Homophone would und wood die nominale Bedeutung in Kontexten wie

John said that he didn't want to do the job, but his brother would **(1.4)**

unterdrückt. Dieser Effekt scheint in der Tat durch die Zugehörigkeit zu einer offenen oder geschlossenen Wortklasse begründet zu sein. Auch wenn Ying (1996) unterschiedliche Artikulationen von Homophonen findet, die in Abhängigkeit vom Kontext variieren, so argumentieren Shillcock und Bard (1993), daß die verschiedenen Realisierungen von Phonetikern isoliert nicht klassifiziert werden konnten, mithin die Beispiele gleich artikuliert seien.[15]

Die Phase der lexikalischen Auswahl ist in wesentlich stärkerem Maße abhängig von Evidenz aus anderen Ebenen als es der lexikalische Zugriff ist.[16] So kann bei der Auswahl ein semantischer

[15] Eine noch stärkere Position wäre durch den Austausch der Realisierung in den Präsentationen möglich.

Einfluß, der durch den vorangegangenen Kontext erzeugt wird, als maßgeblich für die Modifikation von Bewertungen unterschiedlicher Worthypothesen in einer Kohorte angesehen werden. Dazu wird in Priming-Experimenten die höhere Aktivierung eines semantisch durch den Kontext gestützten Wortes in einer Kohorte nachgewiesen. Angenommen, Versuchspersonen wird der Kontext

The men stood around the grave. They mourned at the loss of their ... **(1.5)**

präsentiert. Wird daraufhin der Stimulus /cap/ angeboten, so zeigt sich eine Bevorzugung für /captain/ gegenüber /capital/, obwohl beide denselben akustischen Präfix besitzen (Zwitserlood, 1989). Auch hier zeigen sich erneut Unterschiede der Verarbeitung von Wörtern offener und geschlossener Wortklassen (vgl. Friederici, 1992).

Die bis hierhin dargelegten Ergebnisse psycholinguistischer Forschung haben als Konsequenz der Inkrementalität menschlichen Sprachverstehens eine Beschleunigung der Worterkennung und eine Reduktion der lexikalischen Ambiguität gezeigt. Der Einfluß inkrementellen Verstehens erstreckt sich jedoch auch auf höhere Ebenen. Zwischen den hier angeordneten Prozessen scheint es Interaktionen zu geben, die es dem Menschen ermöglichen, dem extrem hohen Grad an Ambiguität zu begegnen, indem frühzeitig Alternativen nicht bedacht, Analysepfade beherzt beschritten und sich ohne abschließende Kenntnis der Sachlage entschieden wird.[17]

Um hier nur ein Beispiel zu nennen: Niv (1993) argumentiert, daß gewisse syntaktische Entscheidungen über die Funktionen von Konstituenten aufgrund nichtsyntaktischer Information gefällt werden. Ein von ihm zur Illustration dieses Effekts eingeführtes Prinzip lautet: *Avoid new subjects*. Es behandelt die Ursachen von Bevorzugungen prinzipiell gleichwertiger syntaktischer Einbettungen. Am Beispiel:

$$\text{The maid disclosed the safe's location} \begin{cases} \text{a) to the officer} \\ \text{b) had been changed} \end{cases} \qquad \textbf{(1.6)}$$

Die Fortsetzungsvariante b) erfordert (neben der Erkennung eines reduzierten Nebensatzes) die Kategorisierung des Geldschrankes als Subjekt, während er in a) als Objekt fungiert. In Versuchen zeigt sich eine Präferenz für b), was Niv (1993) so interpretiert, daß eine Subjekt-Lesart vermieden wird, wenn die in Frage kommende NP bisher noch nicht in den Diskurs eingeführt worden war.

Die Bedeutung inkrementeller Verarbeitung für das menschliche Kommunikationsverhalten zeigt sich deutlich bei der Untersuchung von Verhaltensmustern in Dialogsituationen. Oft wird es vor-

[16] Klassische Modelle zur Worterkennung wie das Kohortenmodell oder TRACE (McClelland und Elman, 1986) gehen meist von einer Unterteilung des zur Erkennung notwendigen Wissens in Ebenen (phonologisch, syntaktisch, semantisch) aus. Gaskell und Marslen-Wilson (1995) schlagen hingegen die Inkorporation aller möglichen Einflüsse in einen einzelnen Verarbeitungsmechanismus, einem neuronalen rekurrenten Netz, vor.

[17] Nicht umsonst haben auch Menschen große Probleme mit der Verarbeitung von Garden-Path Sätzen wie The horse raced past the barn fell. Die Untersuchung deterministischer Methoden der Sprachverarbeitung ist deswegen lange prominent (Marcus, 1980). Das Nachvollziehen etwa syntaktischer Ambiguitäten durch den Entwickler sprachverarbeitender Systeme stellt diesen manchmal vor schier unlösbare Aufgaben.

kommen, daß ein Dialogpartner vom anderen unterbrochen wird, etwa, weil dieser eine Präsuppositionsverletzung erkannt hat. Diese Unterbrechungen sind nur durch die inkrementellen Eigenschaften des menschlichen Sprachapparates möglich. Besonders deutlich wird dies beim Simultandolmetschen, etwa auf Konferenzen (Künzli und Moser-Mercer, 1995). Hierbei kann ein Dolmetscher es sich aus Zeitgründen und aus Gründen seiner Arbeitslast einfach nicht leisten, die Vervollständigung von Redebeiträgen abzuwarten, bevor er mit der Übersetzung beginnt (Anderson, 1994; Klein, Jekat, und Amtrup, 1996). Dolmetscher führen also nicht nur zwei Aufgaben parallel aus (Textverstehen und -produktion), beide sind zusätzlich stark korreliert. Chernov (1994) berichtet, daß Dolmetscher etwa zu 70 % der Zeit, die sie zuhören, auch sprechen. Der EVS (*Ear-Voice-Span*, der zeitliche Versatz zwischen Hören und Übersetzen) beträgt typischerweise zwei bis sechs Sekunden (Gerver, 1997). Er beruht auf der Ansammlung von quellsprachlichen Einheiten in einer Art Puffer, während das Sprachzentrum noch mit anderen Aufgaben beschäftigt ist (Anderson, 1994). Dabei haben Dolmetscher unterschiedliche Strategien des Vorgehens, je nach struktureller Komplexität der Quellsprache und den Unterschieden in der Struktur der Zielsprache. Im Regelfall scheint ein Dolmetscher zu warten, bis vom Sprecher ein satzäquivalentes Gebilde geäußert wurde. Erst dann — z.B. nach einer Nominalphrase und dem Prädikat — beginnt er zu übersetzen (Goldman-Eisler, 1972). Hier wird eine Verzögerung versucht, bis Syntax und Semantik vollständig vorliegen.

Bei strukturell stark unterschiedlichen Sprachen kann sich diese Strategie als nicht durchführbar erweisen. Beim Dolmetschen von einer Sprache mit Verbletztstellung in eine Verb-Zweit-Sprache beispielsweise stehen bereits Zeitgründe einer solchen Strategie entgegen.[18] Darüberhinaus wächst die Belastung des Dolmetschers, der lange Strukturen memorieren muß. In solchen Fällen weichen Übersetzer auf die offene Satzplanung aus, um auch dann erfolgreich mit der Übersetzung beginnen zu können, wenn die Rollenanbindung an ein Prädikat (oder gar das Prädikat selber) noch nicht klar sind. Um die funktionale Struktur der zielsprachlichen Äußerung möglichst lange offen zu halten, kann ein Dolmetscher eine Kopula verwenden, um strukturellen Restriktionen der Zielsprache zu genügen; danach können die Argumente gedolmetscht werden (vgl. Kitano, 1994, S. 97).

1.3 Inkrementelle Architekturen und die Architektur von MILC

Keines der im Moment existierenden Systeme zum automatischen Dolmetschen führt Simultandolmetschen durch, wenige haben dieses explizite Ziel.[19] Tatsächlich arbeiten die weitaus meisten

[18] Matsubara und Inagaki (1997b) präsentieren ein Übersetzungssystem, daß inkrementell von Englisch nach Japanisch übersetzt. Es nutzt die Eigenschaft aus, daß Englisch eine Verbzweitsprache, Japanisch hingegen eine Sprache mit Verbletztstellung ist. Zur Übersetzung der Komplemente eines englischen Verbs wird zunächst ein kurzer japanischer Satz (bestehend aus dem Subjekt und dem Prädikat des englischen Satzes) übersetzt. Die noch fehlenden Komplemente werden in einem zweiten Satz nachfolgend übersetzt. Solchermaßen angefertigte Übersetzungen wahren einen relativ guten Stil, da im Japanischen Nullsubjekte auftreten können, die Übersetzung des zweiten Satzes mithin keine vollständige Wiederholung des ersten darstellt (Matsubara und Inagaki, 1997a; Mima, Iida, und Furuse, 1998). Diese Art der Verarbeitung ist jedoch sehr stark auf die speziellen Eigenschaften der gewählten Sprachstellung ausgerichtet; eine Umkehrung der Übersetzungsrichtung ist nicht möglich.

[19] Eine Ausnahme bildet das hochgradig parallele System ΦDMDIALOG (Kitano, 1990; Kitano, 1994), das Beispielorientierte Übersetzung durchführt.

nicht-inkrementell.[20] Auch das vorliegende System kann nicht simultan dolmetschen. Da jedoch Inkrementalität eine grundlegende Eigenschaft menschlichen Sprachverstehens ist und in unseren Augen notwendig für erfolgreiche, adäquat kommunizierende technische Systeme zur Sprachverarbeitung, stellt die Inkrementalität Ausgangspunkt und Motivation für diese Arbeit dar.

Bis heute sind Dolmetschsysteme mit großer Abdeckung (z.B. Verbmobil (Wahlster, 1993), Janus (Waibel, 1996)), aber auch umfangreiche sprachverarbeitende Systeme anderer Ausprägung, durch eine Reihe von architektonischen Eigenschaften gekennzeichnet:

- Sie sind modular und sehen unterschiedliche Softwarekomponenten für verschiedene Teilbereiche der Bearbeitung einer Eingabe vor. Die tatsächlich vorgenommene Modularisierung orientiert sich meist an den angenommenen Ebenen linguistischer Verarbeitung. In Einzelfällen gibt es auch unterschiedliche Einteilungen (so sind bei Verbmobil mehrere Module für die Übersetzung zuständig, die unterschiedliche Ansätze von Übersetzung verfolgen (vgl. Abschnitt 4.8.1). Jedes einzelne Modul für sich ist jedoch monolithisch strukturiert, was die ausgesprochen schlechte Eignung zur intramodularen Parallelisierung derartiger Systeme erklärt.
- Sie setzen viele verschiedene Mechanismen zur Repräsentation linguistischen Wissens sowie zur Speicherung und Übertragung von Zwischenergebnissen ein. Die Folge davon ist normalerweise ein ausgesprochen hoher Speicherverbrauch.
- Sie arbeiten nicht-inkrementell, d.h. sie warten, bis der Sprecher die Eingabe abgeschlossen hat, bevor die Verarbeitung einer Äußerung beginnt.

Ein wesentlicher Schritt, der in den letzten Jahren im Bereich der sprachverarbeitenden Systeme vollzogen wurde, ist die Einbeziehung von Spontansprache, während davor stets gelesene oder kontrollierte Rede akzeptiert wurde (Waibel et al., 1991). Dieser Übergang hat großen Einfluß auf die Gestalt der notwendigen Wissensquellen und Verarbeitungskomponenten, treten doch in Spontansprache sowohl auf der Ebene der Sprachlaute als auch der Sprachstrukturen völlig neue Probleme auf. Neben dem starken Einfluß der Verbundenheit der Sprache, der im Ansatz ebenfalls bei gelesener Sprache auftritt, gelangen bei spontanen Äußerungen akustische Phänomene wie Häsitationen, Abbrüche und Pausen in den Vordergrund. Im Bereich der linguistischen Verarbeitung entstehen nun gehäuft fragmentarische Äußerungen, die im Sinne der Standardsprache oft agrammatisch sind und häufig elliptische Konstruktionen enthalten. Zusätzlich sind alle Redebeiträge in eine Sprechsituation eingebettet — z.B. einen Dialog, wie im vorliegenden Fall — die Ellipsen, Anaphern und deiktische Ausdrücke für den Teilnehmer am Dialog perfekt verständlich werden läßt, der für eine maschinelle Analyse aber ausgesprochen problematisch ist (vgl. Tropf, 1994).

Das hier diskutierte System MILC (*M*achine *I*nterpreting with *L*ayered *C*harts) ist eine prototypische Implementierung eines durchgängig inkrementellen Systems zum Dolmetschen spontaner

[20] Inkrementalität wird auf verschiedenen Ebenen partiell in Systeme integriert (Wirén, 1992; Finkler und Schauder, 1992; Schröder, 1994; Poller, 1994; Kilger, 1994; Milward, 1995; Hanrieder, 1996), ohne dadurch zu einer durchgängig inkrementellen Arbeitsweise zu gelangen.

Sprache. Ausgehend vom Prinzip der Inkrementalität lassen sich konkrete Architekturschemata konstruieren, die den Informationsfluß innerhalb solcher Systeme modellieren. Ein prominenter Vertreter hiervon ist das *kognitiv orientierte Architekturmodell* von Briscoe (1987), das unter anderem als Leitlinie im ASL-Projekt (vgl. Görz, 1993) gedient hat. Es beschäftigt sich primär mit der Modularisierung von Komponenten und den möglichen Interaktionen zwischen Komponenten. Jedes einzelne Modul ist gemäß der *schwachen Modularitätshypothese* weitgehend autonom. Verbindungen zwischen Modulen existieren in zwei Spielarten, deren eine den Hauptpfad darstellt, der im wesentlichen den linguistischen Strata folgt. Zusätzlich dazu gibt es Interaktionskanäle, auf denen Suchraumbeschränkungen und Prädiktionen für andere Komponenten transportiert werden (Hahn, 1992; Pyka, 1992c). Des weiteren können diese zusätzlichen Verbindungen für Frage-Antwort-Sequenzen verwendet werden, die zur Disambiguierung dienen.[21]

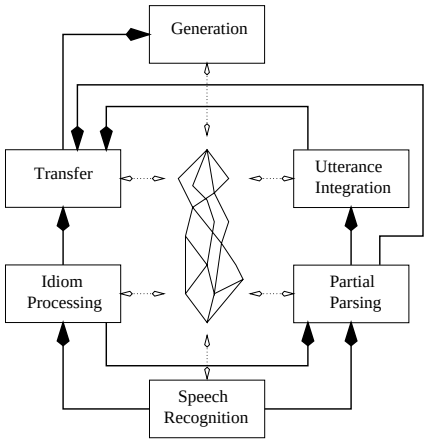


Abbildung 1.2: Das Grobmodell der Architektur von MILC

Die Konzeption von MILC folgt dieser Vorstellung von Architektur, indem sie ein modulares System mit dezidierten Informationskanälen darstellt (Amtrup, 1994b; Amtrup und Benra, 1996)(vgl. Abb. 1.2).[22] Dabei wurde bei der Erstellung des Systems das Prinzip der *Integriertheit* strikt verfolgt. Darunter wird hier verstanden, daß innerhalb des Systems eine einheitliche Datenstruktur zur Speicherung von Zwischenergebnissen verwendet wird. Im vorliegenden Fall wurde dafür eine neue, Mehr-Ebenen-Chart genannte, Struktur konzipiert (Amtrup, 1997b; Amtrup, 1997c). Sie erleichtert die Weitergabe von Resultaten eines Moduls beträchtlich, da Referenzen quasi modulübergreifend valide werden. Der Gebrauch wird in Abb. 1.2 in der Mitte zwischen den Komponenten angedeutet. Eine genaue Beschreibung der Module von MILC und der stattfindenden Interaktionen findet sich in Kapitel 4. Hier sei lediglich erwähnt, daß die Grundlage der Mehr-Ebenen-Chart eine Graphenstruktur ist, die einen direkten Weg zur intramodularen Parallelisierung eröffnet. Darüberhinaus ist stets eine konsistente Sicht auf den Systemzustand durch die Vereinigung der Information

[21] Weitere Architekturschemata sind unter Verwendung anderer Prinzipien natürlich denkbar. Levelt (1989) stellt ein Modell zur Sprachproduktion auf, das fast vollständig auf Interaktion verzichtet.

[22] Die aktuelle Fassung macht jedoch keinen Gebrauch von der Möglichkeit der Nutzung von Interaktionskanälen.

in allen Komponenten gegeben[23], ohne gleichzeitig das Vorhandensein globaler Strukturen zu verlangen, wie dies Boitet und Seligman (1994) tun.

Im Rahmen eines inkrementellen Systems realisiert MILC zusätzlich zur Integriertheit ebenfalls das Prinzip der Uniformität. Das bedeutet den Gebrauch eines einheitlichen Formalismus, wo immer dies sinnvoll möglich ist (Kay, 1984; Weisweber, 1992). Obwohl diese Eigenschaft streng genommen nicht für die Leistungsfähigkeit einer einzelnen Komponente ausschlaggebend ist, so trägt sie zur besseren Verständlichkeit, Wartbarkeit und Erweiterbarkeit eines Systems bei, z.B. durch die Vermeidung von komplexen Umsetzungsproblemen an Modulschnittstellen. Im vorliegenden Fall wurde ein getypter komplexer Merkmal-Formalismus (vgl. Carpenter, 1992) entwickelt, wie er in der modernen Sprachverarbeitung gebräuchlich ist. Dieser wird im vorliegenden Falle in allen Modulen zur deklarativen Formalisierung linguistischen Wissens (vgl. Emele et al., 1991) und dessen Verarbeitung genutzt, von der Erkennung von Idiomen bis zur Generierung. Er trägt durch die automatenorientierte, verschiebungsinvariante Art der Implementierung (vgl. Carpenter und Qu, 1995; Wintner und Francez, 1995b; Wintner und Francez, 1995a und Abschnitt 3.3) viel zur einfachen Art der Verteilung des Systems und zur Effizienz des Einsatzes intermodularer Parallelität bei.

Ein gewichtiger Vorteil daran, Integriertheit und Uniformität zu befolgen, liegt in der Möglichkeit, die Wirkweise komplexer Interaktionsmuster zwischen verschiedenen Komponenten zu explorieren. So ist es z.B. möglich, bei Vorliegen alternativer Analysepfade nicht nur die Endergebnisse der Bearbeitung gleicher Segmente der Eingabe zu vergleichen und das potentiell bessere auszuwählen. Vielmehr ist man dazu in der Lage, bereits während der Erstellung von partiellen Analysen Teilergebnisse auszutauschen und so auf die Arbeit anderer Module einzuwirken. Diese Art der Interaktion und Einflußnahme wird im vorliegenden System MILC durch die Erkennung und Verarbeitung von Idiomen (vgl. Abschnitt 4.5) exemplarisch gezeigt.

Die prädominante Methode zur Strukturierung auf allen Ebenen, die in dieser Arbeit genutzt wird, ist, Graphen als Mittel der Repräsentation zu nutzen. Zunächst bildet natürlich die Mehr-Ebenen-Chart einen verteilt gespeicherten, gerichteten, azyklischen Graph, auf dessen graphentheoretische Grundlagen ausführlich in Kapitel 2 eingegangen wird. Die Eingabe in das System besteht aus Wortgraphen, deren Eigenschaften wir in Abschnitt 2.2 zusammen mit Möglichkeiten zu ihrer algorithmischen Modifikation diskutieren. Die Darstellung aller Arten von intermediären Ergebnissen und schließlich auch der Übersetzung sind in diesem Rahmen graphenartig. Lediglich für die Präsentation der endgültigen Ausgabe wird inkrementell aus dem Generierungsgraphen eine lineare Form erstellt. Daneben bestehen die jeder Kante zugeordneten linguistischen Beschreibungen aus getypten Merkmalstrukturen, die gerichtete, mit Kanten- und Knotenbezeichnungen versehene, allerdings potentiell zyklische Graphen realisieren. Der zugrundeliegende Typenverband ist ein gerichteter, zusammenhängender Wurzelgraph (vgl. Kapitel 3).

Schließlich bildet auf einer gröberen Strukturierungsebene die Anwendung MILC selbst einen Graphen von Komponenten, dessen Kanten Kommunikationswege repräsentieren (vgl. Abschnitt 4.2). Der Komponentengraph kann potentiell zyklisch sein, um die Einführung von Rückkopplungen zu

[23] Natürlich ist der globale Zustand eines verteilten Systems nicht bekannt. Gemeint ist hier, daß die Dateneinheiten, die zu einem bestimmten Zeitpunkt in mehreren Modulen verfügbar sind, sich sinnvoll kombinieren lassen.

erlauben. Von dieser Eigenschaft wird in der aktuellen Fassung von MILC allerdings noch kein Gebrauch gemacht.

Bei MILC handelt es sich um ein Transfer-orientiertes Dolmetschsystem. Der fundamentale Unterschied zwischen Transfersystemen und ihrem Gegenüber, den Interlingua-basierten Systemen, besteht im Status der Information, die als Mediator zwischen Quell- und Zielsprache dient (Hutchins, 1986). Interlingua-Systeme (Nirenburg, 1987; Nirenburg, 1992) stellen eine interne Repräsentation sprachlicher Ausdrücke zur Verfügung, die unabhängig von der konkreten Sprache ist, in der sie geäußert wurden (vgl. Abb. 1.3). Das bedeutet, daß für ein Übersetzungssystem mit n Sprachen $2n$ Komponenten erforderlich sind, pro Sprache ein Analyse- und ein Generierungsmodul.

Auf Transfer basierende Systeme (s. z.B. Nagao und Tsujii, 1986; Luckhardt, 1987; Grabski, 1990) hingegen kennen keine solche einheitliche Repräsentation. Vielmehr existiert für jedes Sprachpaar (und im Extremfall sogar für jede Übersetzungsrichtung) eine spezialisierte Transferkomponente. Obwohl dies im schlimmsten Fall $3n(n \Leftrightarrow 1)$ Komponenten erfordert, scheint sich der Transferansatz zunehmend durchzusetzen. Dies liegt nicht nur an der fast philosophischen Diskussion über die notwendige Mächtigkeit der Interlingua (Hutchins und Somers, 1992; Nirenburg, 1993), die alle Feinheiten sprachlicher und kultureller Dimension auszudrücken imstande sein muß[24], sondern auch daran, daß man eine sehr weitgehende Disambiguierung in vielen Fällen für nicht erforderlich hält (Zajac, 1990; Somers, 1993).

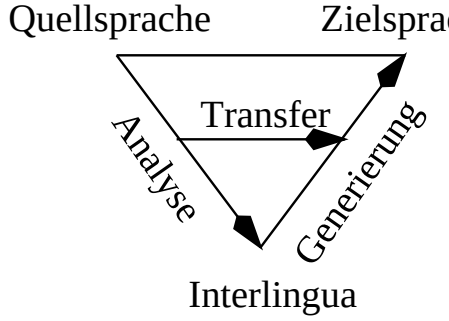


Abbildung 1.3: Interlingua und Transfer

In Transfer-orientierten Systemen kann man sich darauf beschränken, bis zu der Tiefe zu analysieren, die für die Übersetzung im gewählten Sprachpaar ausreicht. Dies bedeutet im Regelfall weniger Aufwand, da z.B. die kulturelle Stellung der Sprachen zueinander klar ist.

Unabhängig davon, welche Art von Analyse man für eine Sprache vornimmt, bevor Eingaben transferiert werden, wird es vorkommen, daß bestimmte Teile einer zu übersetzenden Äußerung mitunter zu weitgehend untersucht werden. So lassen sich für jede Ebene linguistischer Betrachtung Phänomene finden, deren Übersetzung adäquat nur durch Einbeziehung von Wissen der entsprechenden Ebene geleistet werden kann (Kinoshita, Phillips, und Tsujii, 1992). Um nur einige zu nennen:

[24] Es gibt sogar Ansätze, innerhalb von Übersetzungssystemen Esperanto oder Englisch als Interlingua zu benutzen (vgl. Schubert, 1992).

- *Idiome* und *Routineformeln*. Sie lassen sich am einfachsten aufgrund lexikalischer Information übersetzen, da sie vielfach eine nicht-kompositionelle Semantik aufweisen, strukturell aber vergleichsweise fest sind.
- *Head switching*. Dabei handelt es sich um ein oft zitiertes Beispiel für ein strukturelles Transferproblem syntaktischer Art[25]. Es besteht darin, daß bei der Übersetzung in der Quellsprache als Adjunkte dienende Lexeme Köpfe in der zielsprachlichen Realisierung werden. In **(1.3)** muß das deutsche gerne als Kopf der englischen Phrase (likes) benutzt werden.

  Hans schwimmt gerne
  John swims likingly
  "John likes to swim."

  (1.7)

- *Zeit* und *Aspekt*. Sie sind nur aufgrund semantischer Information korrekt zu übersetzen. In vielen Fällen verlangen sie sogar nach extralinguistischem Wissen (Eberle, Kasper, und Rohrer, 1992).

Üblicherweise ist die Transferkomponente nach der syntaktisch/semantischen Analyse angeordnet. Durch die monotone Art der Informationsanreicherung innerhalb linguistischer Objekte, wie sie fast ausschließlich praktiziert wird, kann der Transfer auch auf Ergebnisse vorangegangener Analysekomponenten (etwa der Prosodie) zugreifen, so daß i.a. die Behandlung der eben erwähnten Probleme möglich ist. Ein Verfahren, die Tiefe der Analyse zu regulieren, besteht dagegen in aller Regel nicht. Hat man sich bei der Architektur eines Transfersystems einmal entschieden, welche Ebene der Repräsentation als Grundlage des Transfers dienen soll, so wird normalerweise davon ausgegangen, daß alle Äußerungen bis zu dieser Ebene analysiert werden. Dies ist jedoch nicht zwingend.

Erfahrene Simultandolmetscher etwa kann man dabei beobachten, wie sie gewisse Redewendungen und Passagen fast automatisch übersetzen (Hauenschild, 1985). Hier scheint eine tiefe Analyse des sprachlichen Materials nur dann zu geschehen, wenn dieses komplexer Natur ist oder schwer auflösbare Ambiguitäten enthält. Hauenschild und Prahl (1994) schlagen auf dieser Grundlage die Einführung eines variablen Transfers vor, der die unterschiedlichen Anforderungen berücksichtigt (vgl. Weisweber und Hauenschild, 1990).

Der Vorschlag geht dahin, Äußerungen und ihre Teile nur so tief zu analysieren, wie das für den erfolgreichen Transfer in die Zielsprache notwendig ist.

Der Mehr-Ebenen-Transfer nach dem Prinzip der variablen Analysetiefe postuliert, daß genauso viele Ebenen des Transfers wie Ebenen linguistischer Verarbeitung existieren, daß es folglich mehrere Punkte auf der Analyseseite geben kann, an denen Transfer ansetzt (vgl. Abb. 1.4).

Durch die Integriertheit und Uniformität von MILC ist eine optimale Grundlage zur explorativen Untersuchung unterschiedlicher Transfermechanismen gegeben. Exemplarisch ist dies gezeigt

[25] Obwohl Dorr (1993) derartige *mismatches* lexikalisch behandelt.

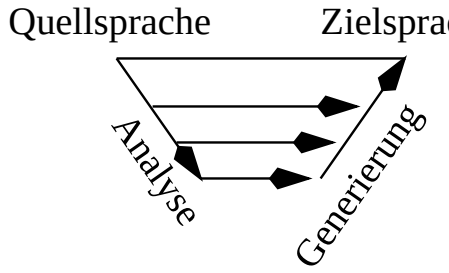


Abbildung 1.4: Mehr-Ebenen-Transfer

durch die Implementierung des Transfers von idiomatischen festen Redewendungen, die durch eine eigene Komponente erkannt und deren nichtkompositionale Analyse direkt an den Transfer weitergeleitet wird. Neben der Vermeidung von Redundanz wird die Verarbeitung in diesem Fall zusätzlich beschleunigt, da die Idiomerkennung wesentlich weniger komplex ist als eine volle linguistische Analyse.

Die Verarbeitungszeit spielt — gerade bei Anwendungen, die gesprochene Sprache als Ein- und Ausgabemodalität nutzen — eine große, ja entscheidende Rolle. Während die Übersetzung eines geschriebenen Textes ohne größere negative Auswirkungen über Nacht ausgeführt werden kann, liegt die Toleranzschwelle beim Warten auf Reaktionen eines interaktiven Systems (hier natürlich gemeint im Sinne eines Dialogsystems) lediglich im Bereich von einigen Sekunden. Aus diesem Grunde ist schon recht früh (Wahlster, 1993) die Forderung nach der Entwicklung von Any-time-Algorithmen (Russel und Zilberstein, 1991; Mouaddib und Zilberstein, 1995) für die Sprachverarbeitung entstanden (Menzel, 1994). Das ultimative Ziel ist selbstverständlich die Konstruktion eines zeitsynchron arbeitenden Systems zur maschinellen Übersetzung gesprochener Sprache. Da dies zum gegenwärtigen Zeitpunkt illusorisch erscheint, kann wenigstens gefordert werden, daß das System eine Art von Bewußtheit über die Beschränkung der zur Verfügung stehenden Zeit entwickelt oder sich vom Benutzer unterbrechen läßt.

MILC implementiert keine Any-time Algorithmen. Wir werden zu einem späteren Zeitpunkt ausführlich auf die Interferenzen zwischen Any-time, komplexen Formalismen, Modularität und Inkrementalität eingehen; hier sei nur angemerkt, daß erneut die Anwendung inkrementeller Verfahren eine notwendige Eigenschaft zur Einführung einer Any-time Eigenschaft in modulare Dolmetschsysteme darstellt.

Die Entwicklung von MILC ist selbstverständlich nicht loszulösen von der Präsenz anderer Systeme. In den letzten Jahren sind eine ganze Reihe interessanter, fortgeschrittener Anwendungen zur Übersetzung geschriebener und gesprochener Sprache entwickelt worden. Wir werden hier nur auf vier für den Gang der Arbeit relevante Vertreter eingehen: SLT, Trains, TDMT und Verbmobil.

Der *Spoken Language Translator* (SLT) stellt die Forschungsumgebung für Übersetzungssysteme von SRI in Cambridge dar. Die Ausgangsbasis für weitergehende Entwicklungen besteht in einem aus vorhandenen Einzelkomponenten komponierten System zur Übersetzung gesprochener Sprache in der ATIS-Domäne (Flugplan-Information) (Agnäs et al., 1994). Die Sprachstellung war ursprünglich Schwedisch-Englisch, inzwischen sind Module zur Verarbeitung französischer Sprache hinzugekommen. Außerdem ist der anfänglich isoliert anmutende Charakter einzelner Module einer zunehmend integrierteren Prozessierung gewichen.

Die Eingabe in das System besteht in der Liste der fünf besten Ketten, die ein Erkenner für eine Äußerung liefert; diese werden in einen kleinen Wortgraphen transformiert. Zwei Instanzen der *Core Language Engine*, einer unifikationsbasierten linguistischen Maschine, dienen zur Analyse und Generierung. Der eigentliche Übersetzungsschritt ist transferbasiert und benutzt sowohl eine tiefe linguistische Analyse als auch eine an Oberflächenrepräsentationen orientierte, flache Übersetzung (Rayner und Carter, 1997). Obwohl der SLT insgesamt nichtinkrementell arbeitet, wird ständig die beste Kette von Übersetzungen bereitgehalten, die aus Kombinationen tiefer und flacher Analysen bestehen kann.

Eine Methode, die zur Verbesserung der Qualität der Analysen und zur Steigerung der Effizienz ausgenutzt wird, besteht in der Spezialisierung einer recht allgemeinen Grammatik durch domänenspezifisches Wissen. Die Ambiguität, die durch lexikalische Mehrdeutigkeiten auf der Basis syntaktischer Kategorien eingeführt wird, kann durch die Benutzung eines Sprachmodells stark reduziert werden. Diese Methode wird erweitert durch die Elimination der Ambiguität auf Konstituentenbasis, indem ein überwachtes Trainingsverfahren angewendet wird. Dazu werden sog. Diskriminanten aus Konstituenten extrahiert, die eine stark verkürzte Beschreibung enthalten. Auf diesen wird erneut ein Sprachmodell trainiert, daß als zusätzliche Einschränkung während der Bearbeitung dient. Zusätzlich wird die Tatsache ausgenutzt, daß, abhängig von der Domäne, bestimmte Grammatikregeln häufiger benutzt werden als andere. Die generelle Grammatik wird auf die für die Domäne relevanten Regeln reduziert, was zu einer deutlichen Effizienzsteigerung bei milder Übergenerierung führt (Rayner und Carter, 1996). Der SLT stellt mithin eine erfolgreiche Demonstration des Einsatzes hybrider symbolischer und statistischer Verfahren dar.

Trains (Allen et al., 1994) ist kein Übersetzungssystem. Vielmehr handelt es sich hierbei um ein Dialogsystem für einfache Routenplanungen. Ein Aspekt ist jedoch für die vorliegende Arbeit von Interesse: Die Korrektur von Spracherkennungsfehlern. Trains verwendet einen beste-Kette Erkenner und ist deswegen anfällig für Fehler, die während der Erkennungsphase auftreten können. Die dort praktizierte Lösung besteht darin, die Ausgabe des Spracherkenners einer Nachverarbeitung zu unterziehen, um häufige Fehler zu korrigieren. Dazu wird die erkannte Kette Modifikationen unterworfen und die Wahrscheinlichkeit der resultierenden Wortkette mit einem Sprachmodell approximiert. Die Modifikationen bestehen im wesentlichen in der Vertauschung von einzelnen Wörtern des Lexikons (Allen et al., 1996).

TDMT (*Transfer Driven Machine Translation*, Furuse, 1994) stellt den Transfer in das Zentrum der Verarbeitung. Ein an vorher gesehenen Übersetzungen orientiertes Transfermodul erhält die in das System eingehenden Oberflächenrepräsentationen. Die Transferregeln bestehen aus Mustern, die häufig vorkommende Kombinationen von Wörtern und Phrasen miteinander in Beziehung setzen

(Sobashima et al., 1994). Aufgrund der in den Regeln enthaltenen Muster können Auswertungsvorgänge angestoßen werden, die unterschiedliche Ebenen linguistischer Verarbeitung betreffen, z.B. lexikalischer Art zur Analyse von Komposita oder syntaktischer Art zur Behandlung von Kombinationen auf phrasaler Ebene. Aufgrund der Art der Tansferregeltabelle eignet sich TDMT gut für eine parallele Bearbeitung von Übersetzungen, die sich aus der Partitionierung des Transferwissens ergibt (Oi et al., 1994); durch die Einführung von Grenzmarkierungen zwischen Konstituenten kann außerdem eine inkrementelle Verarbeitung erreicht werden (Furuse und Iida, 1996). Die Einführung dieser Grenzen beruht jedoch initial auf einer Annotierung der Eingabewörter mit syntaktischen Kategorien, um die auf Mustern basierenden Regeln zu instantiieren.

Verbmobil schließlich (Kay, Gawron, und Norvig, 1991; Wahlster, 1993) liefert wesentliche Impulse für die Anfertigung dieser Arbeit; der Autor war einige Jahre Mitglied des Projektes. Verbmobil ist ein sehr großes, aus etlichen Modulen bestehendes System zur Übersetzung spontansprachlicher Dialoge in der Domäne der Terminvereinbarung. Es verarbeitet die Sprachen Deutsch, Englisch und Japanisch, geht jedoch davon aus, daß alle Partizipanten in einem Dialog wenigstens passives Wissen von Englisch besitzen. Die Eingabe besteht aus Wortgraphen, die zunächst mit prosodischer Information angereichert werden; die Ausnutzung derartigen Wissens ist zum ersten Mal in diesem Kontext erfolgreich in großem Maßstab angewendet worden. Die erste Phase des Projektes unterwarf die Wortgraphen einer tiefen linguistischen Analyse und Übersetzung sowie unabhängig davon zwei unterschiedlichen Arten flacher Übersetzung aufgrund von oberflächenorientierten Analysen und statistischer Information. Mittlerweile werden die verschiedenen Ansätze zur Erstellung einer Analyse des Inhaltes jedoch holistisch miteinander kombiniert, um auch äußerungsintern von der Möglichkeit des Resultataustausches Gebrauch machen zu können (Worm, 1998). Die Modellierung des Dialogwissens und ein Teil des Transfers sind dabei an Dialogakten (Jekat et al., 1995) orientiert. Verbmobil erreicht mehr als 70% approximativ korrekte Übersetzungen (Wahlster, 1997).

Obwohl einige Komponenten innerhalb des Verbmobil-Systems inkrementelle Verarbeitungsmechanismen erlauben (z.B. Finkler, 1996), stellt Verbmobil bislang kein durchgängig inkrementelles System dar (Hahn und Amtrup, 1996). Insbesondere besteht die Eingabe in einem Wortgraphen, der nichtinkrementell erzeugt wurde. Ein Teil der linguistischen Analyse (Caspari und Schmid, 1994) basiert auf der nichtinkrementellen A*-Suche. Das im architektonischen Teilprojekt von Verbmobil entwickelte System INTARC (Görz et al., 1996) dagegen realisiert einen inkrementellen Ansatz. Eine bedeutsame Einschränkung für die Inkrementalität stellt jedoch die Restriktion auf die Übersetzung von am Anfang der Äußerung beginnender, satzwertiger Konstituenten dar. Eine fragmentarische Analyse ist nicht vorgesehen. Die wesentlichen Eigenschaften sind

- Die inkrementelle Erzeugung von Wortgraphen (Huebener, Jost, und Heine, 1996) und deren Weiterverarbeitung in einer strikt von links nach rechts operierenden Art.
- Die Erzeugung prosodischer Information ohne Rückgriff auf die Präsenz von vorher erkannten Wörtern (Strom und Widera, 1996).
- Die Aufteilung der syntaktisch-semantischen Analyse in die inkrementelle Suche nach Äußerungshypothesen und deren anschließende Verifikation aufgrund einer Unifikationsgrammatik (Kasper et al., 1996).

- Eine dialogaktbasierte Übersetzung, die im Falle des Scheiterns der tiefen linguistischen Analyse auf rudimentäre Information über fokussierte Wörter und deren zugeordnete Dialogakte zurückgreifen kann (Elsner und Klein, 1996).
- Die Möglichkeit zur Einführung von Interaktionen zwischen Modulen zur linguistischen Analyse und der Worterkennung (Hauenstein und Weber, 1994).

1.4 Zusammenfassung

Die Inkrementalität von sprachverarbeitenden Systemen scheint der nächste logische Schritt in der Entwicklung großer, effizienter, natürlich operierender Systeme. Das in der vorliegenden Arbeit entwickelte System MILC ist eine Implementierung eines inkrementellen Systems zum Dolmetschen spontan gesprochener Sprache. Inkrementalität läßt sich als "stückchenweise" Bearbeitung der Eingabedaten ansehen, was z.B. dazu führt, daß ein System reagieren kann, noch bevor die Eingabe komplett vorliegt. Eine solche Art des Verhaltens ist an den Eigenschaften des menschlichen Sprachverstehens orientiert; sie eröffnet neue Möglichkeiten der Einschränkung von Ambiguität (und damit der Steigerung der Effizienz) durch die Ausnutzung von Interaktionen und die Einführung von intermodularer Parallelität. Simultandolmetscher nutzen inkrementelle Verarbeitungsstrategien u.a. zur Lastreduzierung mittels offener Satzplanung.

MILC ist modular. Die Komponenten sind durch einen Kommunikationsmechanismus miteinander verbunden, der kanalorientiert arbeitet. Sie bilden eine integrierte Architektur, indem eine neuartige Datenstruktur, die Mehr-Ebenen-Chart, zur Speicherung von Resultaten und Zwischenergebnissen verwendet wird. Ein automatenorientierter, getypter Merkmalformalismus, der einen effizienten Austausch von Merkmalstrukturen zuläßt, dient der Darstellung linguistischen Wissens. Die Art der Verarbeitung wurde an fünf Dialogen eines Korpus spontan gesprochener Sprache in der Domäne der Terminvereinbarung demonstriert, die zu 60,4% approximativ korrekt übersetzt wurden.

Kapitel 2

Sprachverarbeitung und Graphentheorie

Die Graphentehorie bildet eine der zentralen Säulen für die vorliegende Arbeit. Alle relevanten Datenstrukturen und Mechanismen lassen sich auf Operationen auf Graphen zurückführen. In diesem Kapitel werden deshalb einige Begriffe aus der Graphentheorie eingeführt, die für die Betrachtung von Anwendungskonzepten für uns relevant sind. Wir beginnen bei allgemeinen Betrachtungen über Graphen und enden schließlich bei inkrementellen Suchverfahren.

Graphen sind als fundamentale Datenstruktur für die Sprachverarbeitung von zentraler Bedeutung. Im Rahmen dieser Arbeit sind mindestens vier Aspekte zu betrachten: Zum einen sind sämtliche Algorithmen, die zur linguistischen Verarbeitung der Eingabe dienen, auf dem Konzept einer Chart aufgebaut, die einen Graphen darstellt. Zum anderen besteht die Eingabe für das System aus Wortgraphen, die einem akustischen Erkenner entstammen. Drittens bestehen die Annotationen an Kanten aus Merkmalstrukturen, die graphenförmig sind. Und schließlich formt die Anwendung selbst einen Graphen, dessen Knoten Komponenten und dessen Kanten Informationspfade sind.

2.1 Allgemeine Definitionen

Wir beginnen damit, einen Graphen zu definieren. Für manche Begriffe werden in Klammern hinter den deutschen Bezeichnungen die jeweiligen englischen Entsprechungen angegeben. Zu einer Einführung in die Graphentheorie s. z.B. Chen (1971) oder Harary (1974).

Definition 2.1 (Graph)
Ein Graph G ist ein Paar $G(\mathcal{V}, \mathcal{E})$, bestehend aus einer endlichen Menge $\mathcal{V}$ von Knoten (vertices,

nodes) sowie einer endlichen Menge $\mathcal{E}$ von ungeordneten Paaren (v, v'), $v, v' \in \mathcal{V}$, die Kanten (edges) heißen.

Der Graph $G(\emptyset, \emptyset)$ heißt leerer Graph. Die Kanten in dieser Definition sind zunächst ungerichtet, die beiden Endpunkte v und v' einer Kante (v, v') sind gleichberechtigt. Ausdrücklich zugelassen sind Kanten der Form (v, v), sogenannte Schleifen (*loops*).

In dieser Arbeit werden fast ausschließlich Graphen genutzt, bei denen nicht nur die Identität der durch eine Kante verbundenen Knoten eine Rolle spielt, sondern für die auch die Richtung der Kante wesentlich ist. Dies liegt vor allem daran, daß eine der charakteristischen Eigenschaften von Sprache ihre Linearität ist. Eingabeäußerungen werden auf der untersten Ebene, die für diese Arbeit relevant ist, als zeitlich aufeinanderfolgende Hypothesen über gesprochene Wörter repräsentiert. Linguistische Beschreibungen beziehen sich stets auf (zeitliche) Ausschnitte einer Äußerung. Ebenso greift die formale Beschreibung linguistischen Wissens (vgl. Kapitel 3) auf Kanten mit einer Richtung zurück, indem hierarchische Zusammenhänge dargestellt werden müssen. Wir werden folglich im weiteren stets sogenannte *gerichtete Graphen* betrachten und kurz Graphen nennen.

Definition 2.2 (Gerichteter Graph, zugehöriger ungerichteter Graph)
Ein gerichteter Graph (directed Graph) G ist ein Paar $G(\mathcal{V}, \mathcal{E})$, bestehend aus einer endlichen Menge $\mathcal{V}$ von Knoten sowie einer endlichen Menge $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ von Paaren von Knoten, die gerichtete Kanten heißen (geschrieben (v, v') oder vv'. Der ungerichtete Graph, der durch Herauslassung der Richtung von Kanten entsteht, heißt der zu G gehörige ungerichtete Graph.

Im Gegensatz zu ungerichteten Graphen gilt in gerichteten Graphen i.A. nicht $\forall_{v,v'} : vv' \in G \Longrightarrow v'v \in G$. Oft ist es sinnvoll, zwischen zwei Knoten eines Graphen mehr als eine Kante zuzulassen: Dies dient in der Anwendung der Sprachverarbeitung z.B. zur Darstellung von Ambiguität bzw. zur Repräsentation von Wissen unterschiedlicher Art über einen Ausschnitt der Eingabe. Wir lassen deshalb eine Indizierung von Kanten zu, so daß Kanten, die identische Paare von Knoten miteinander verbinden, unterscheidbar werden. Da wir über eine Indizierung mit natürlichen Zahlen hinaus weitere Information über Kanten vorhalten wollen, verwenden wir Kantenbezeichnungen (*labels*). Als Bezeichnungen werden im vorliegenden System einerseits Wörter benutzt (zur Darstellung der Eingabeäußerung), andererseits werden Kanten innerhalb linguistischer Verarbeitungsstufen mit Merkmalstrukturen annotiert.

Definition 2.3 (Graph mit Kantenbezeichnungen)
Ein Graph mit Kantenbezeichnungen (labelled graph) G ist ein Tripel $G(\mathcal{V}, \mathcal{E}, \mathcal{L})$, bestehend aus einer endlichen Menge $\mathcal{V}$ von Knoten, einer Menge $\mathcal{L}$ von Bezeichnungen (labels) sowie einer endlichen Menge $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{L}$ von Kanten, die jeweils eine Bezeichnung tragen.

Zusätzlich kann einer Kante ein Gewicht beigemessen werden, etwa die akustische Bewertung (*score*) als Maß für die Übereinstimmung eines Signalausschnittes mit dem Modell eines Wortes. Kantengewichte dienen insbesondere zur Bewertung der Relevanz einer Kante für den Gang der Ver-

arbeitung. Mit Hilfe dieser Information wird die Reihenfolge der Anwendung von Bearbeitungsschritten gesteuert.

Definition 2.4 (Graph mit Kantengewichten)
Ein Graph mit Kantengewichten (weighted graph) G ist ein Tripel $G(\mathcal{V}, \mathcal{E}, \mathcal{W})$, bestehend aus einer endlichen Menge $\mathcal{V}$ von Knoten, einer Menge $\mathcal{W}$ von möglichen Gewichten (weights) sowie einer endlichen Menge $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} \times \mathcal{W}$ von gewichteten Kanten. Normalerweise werden die Gewichte aus $\mathbb{R}$ entnommen.

Um auf die einzelnen Komponenten eines Graphen und seiner Bestandteile zugreifen zu können, führen wir folgende nützliche Funktionen ein:

Definition 2.5 (Zugriffsfunktionen)
Sei $G(\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{L})$ ein gerichteter Graph mit Kantenbezeichnungen und Gewichten. Sei e eine Kante des Graphen mit $e = (v, v', w, l)$. Wir definieren

- *α als Zugriffsfunktion auf den Startknoten einer Kanten*

$$\alpha : \mathcal{E} \Leftrightarrow \mathcal{V}, \alpha(e) := v \tag{2.1}$$

- *β als Zugriffsfunktion auf den Endknoten einer Kanten*

$$\beta : \mathcal{E} \Leftrightarrow \mathcal{V}, \alpha(e) := v' \tag{2.2}$$

- *l als Zugriffsfunktion auf die Bezeichnung einer Kante*

$$l : \mathcal{E} \Leftrightarrow \mathcal{L}, l(e) := l \tag{2.3}$$

- *w als Zugriffsfunktion auf das Gewicht einer Kante*

$$w : \mathcal{E} \Leftrightarrow \mathcal{W}, w(e) := w \tag{2.4}$$

Vollständige linguistische Analysen beschreiben immer eine Traversierung des die Eingabe repräsentierenden Graphen. Wir benötigen deswegen eine Charakterisierung des “Wegs” durch einen Graphen oder Teile desselben. Zusätzlich führen wir den Begriff der Erreichbarkeit ein, der angibt, ob man von einem Knoten ausgehend zu einem anderen Knoten gelangen kann, indem man einer Reihe von Kanten folgt.

Definition 2.6 (Adjazenz, Erreichbarkeit)
Zwei Knoten v und v' eines Graphen $G(\mathcal{V}, \mathcal{E})$ heißen adjazent ($v \to v'$), falls $vv' \in \mathcal{E}$:

$$\forall v, v' \in \mathcal{V} : v \to v' \Leftrightarrow \exists e \in \mathcal{E} : \alpha(e) = v \wedge \beta(e) = v' \tag{2.5}$$

Die transitive Hülle der Adjazenz $\to$ heißt Erreichbarkeitsrelation und wird mit $\overset{*}{\to}$ geschrieben.

Definition 2.7 (Kantenfolge, Kantenzug, Weg, Kreis)
Sei $G(\mathcal{V}, \mathcal{E})$ ein Graph, $e_{[1;n]}$ eine endliche Folge der Länge n von Kanten des Graphen. Wir schreiben $e^{(i)}_{[1;n]}$ für die i-te Komponente der Folge, $i \in \{1 \ldots n\}$.

Die Folge heißt Kantenfolge (edge sequence), wenn alle Kanten ineinander übergehen, d.h. falls gilt:

$$\forall_{i=1\ldots n-1} \beta(e^{(i)}_{[1;n]}) = \alpha(e^{(i+1)}_{[1;n]}) \tag{2.6}$$

Eine Kantenfolge heißt geschlossen, falls $\alpha(e^{(1)}_{[1;n]}) = \beta(e^{(n)}_{[1;n]})$, sonst offen. Eine Kantenfolge heißt Kantenzug (edge train), falls alle Kanten des Zuges unterschiedlich sind, also

$$\forall_{i,j \in \{1\ldots n\}} e^{(i)}_{[1;n]} = e^{(j)}_{[1;n]} \Longrightarrow i = j \tag{2.7}$$

Eine offene Kantenfolge heißt Weg (path), falls kein Knoten mehrfach vorkommt, d.h.

$$\forall_{i,j \in \{1\ldots n\}} \beta(e^{(i)}_{[1;n]}) = \alpha(e^{(j)}_{[1;n]}) \Longrightarrow j = i + 1 \tag{2.8}$$

Eine geschlossene Kantenfolge heißt Kreis oder Zyklus, falls außer $\alpha(e^{(1)}_{[1;n]}) = \beta(e^{(n)}_{[1;n]})$ kein Knoten mehrfach vorkommt.

Definition 2.8 (Länge einer Kantenfolge, Abstand)
Die Länge einer Kantenfolge ist definiert als die Anzahl der Kanten in ihr. Als Abstand $\delta(v, v')$ zwischen zwei Knoten ist die Länge des kürzesten Weges zwischen ihnen definiert. Falls es keinen solchen Weg gibt, definieren wir $\delta(v, v') = \infty$.

Zwei weitere grundlegende Begriffe der Graphentheorie, Zusammenhang und Planarität, sind im Rahmen dieser Arbeit lediglich von untergeordnetem Interesse. Der Zusammenhang der verwendeten Graphen wird in späteren Definitionen implizit gefordert, so daß nichtzusammenhängende Graphen nicht auftreten; die Graphen sind außerdem regelmäßig so dicht, daß sie nicht planar sind.

Definition 2.9 (Zusammenhang)
Ein Graph heißt zusammenhängend, wenn je zwei Knoten durch einen Weg verbunden sind. Ein gerichteter Graph heißt stark zusammenhängend, falls je zwei Knoten gegenseitig erreichbar sind. Er heißt einseitig zusammenhängend, falls zwischen zwei Knoten mindestens ein Weg besteht. Er heißt schwach zusammenhängend, falls der zugehörige ungerichtete Graph zusammenhängend ist.

Definition 2.10 (Planarität)
Ein Graph heißt planar, falls er sich in einer Ebene zeichnen läßt, ohne daß sich zwei Kanten schneiden.

Charts, die für die natürlichsprachliche Verarbeitung verwendet werden, sind i.A. nicht planar. Als Beispiel diene der geschriebene Satz

Der Peter singt mit Freude. **(2.1)**

Die syntaktische Strukturanalyse mit üblichen Methoden führt auf eine Chart ähnlich der in Abb. 2.1. Sie enthält mindestens präterminale Kanten, die aus einem Lexikonzugriff resultieren, sowie einige inaktive Kanten, die drei unterschiedlichen Satzinterpretationen entsprechen, und eine inaktive Kante, welche die gesamte Verbphrase erklärt. Diese Chart ist isomorph zu dem $K_{(3,3)}$-Graphen in Abb. 2.2. Jener ist ein vollständiger bipartiter Graph mit sechs Knoten, die in zwei Mengen von je drei Knoten partitioniert werden können. Nach dem Theorem von Kuratowski (Aigner, 1984, S. 74) ist der $K_{(3,3)}$ und damit die vorliegende Chart nicht planar.

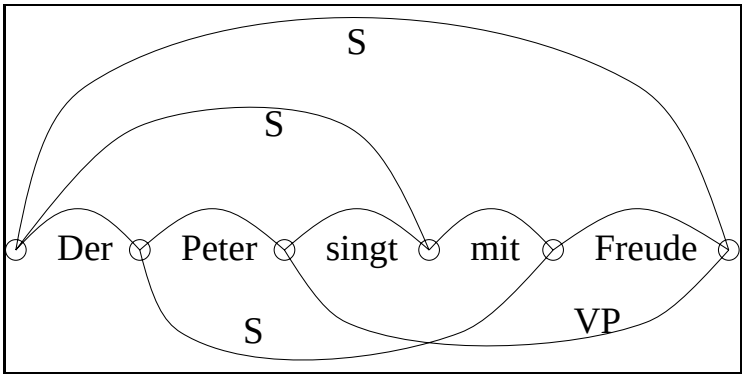


Abbildung 2.1: Eine Chart für „Der Peter singt mit Freude“

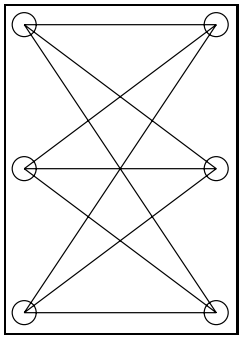


Abbildung 2.2: Der Graph $K_{(3,3)}$

Im Normalfall einfacher ist der Nachweis der Nichtplanarität, indem man ein Theorem über die lineare Beschränkung der Anzahl der Kanten in einem Graphen benutzt (vgl. McHugh (1990, S. 38)), das auf die Eulersche Formel zurückgeht. Insbesondere bei Wortgraphen ist die Nichtplanarität dann einfach zu zeigen.

Die Bezeichnung Chart[1] für eine Datenstruktur, die vollständige und unvollständige Hypothesen über einen gewissen Ausschnitt der Eingabe speichert, hat — abhängig vom Anwendungsfall — Konsequenzen für die Form und die Eigenschaften des Graphen, den sie bildet. Betrachtet man geschriebene Eingabe, so bilden die Zwischenräume zwischen den Wörtern die Knoten des Graphen (und der Chart).[2] Je zwei benachbarte Knoten sind durch (mindestens) eine Kante verbunden, die das eingegebene Wort repräsentiert. Dadurch ist der Graph einseitig zusammenhängend, der Abstand zweier beliebiger Knoten ist endlich ($\forall_{v,v' \in V} : \delta(v, v') \neq \infty$). Die Erreichbarkeitsrelation in dem gerichteten Graphen bildet also eine totale Ordnung auf den Knoten des Graphen.

Verwendet man eine Chart für die Darstellung von Strukturen, bei denen Alternativen auf der niedrigsten Ebene (i.A. Wörter) entstehen, so gilt dies nicht mehr. Die Ausgabe eines Worterkenners kann z.B. aus einem Wortgraphen bestehen, der mehrere alternative Pfade vorsieht. Dies bedeutet, daß Knoten entstehen, die nicht mehr durch einen Weg verbunden sind, deren Abstand also ∞ ist. Die Erreichbarkeitsrelation bildet nurmehr eine Halbordnung. Alle Wege vom Anfangsknoten des Graphen zum Endknoten des Graphen bilden dagegen einseitig zusammenhängende Graphen, sie repräsentieren gerade alternative Hypothesen über die möglichen Gesamt-Wortfolgen.[3]

Wichtig ist die folgende Definition eines gerichteten, azyklischen Graphen, da praktisch alle in der Computerlinguistik vorkommenden Graphen diese Gestalt haben. Charts und Wortgraphen haben neben ihrer Gerichtetheit die Eigenschaft, einen linearen Charakter zu besitzen, d.h. sie enthalten keine Kanten, die in der Zeit rückwärts weisen: sie sind sogar azyklisch.[4]

Definition 2.11 (DAG)
Ein gerichteter, azyklischer Graph (directed, acyclic graph, DAG) ist ein gerichteter Graph, der keine gerichteten Kreise enthält.

2.2 Wortgraphen als Eingabe für sprachverarbeitende Systeme

Systeme, die geschriebene Sprache als Eingabe erhalten, sind in der glücklichen Lage, keinerlei Zweifel über die Identität der Eingabe zu haben.[5] Besteht die Eingabe dagegen aus gesprochener Sprache, taucht eine ganze Reihe von Problemen auf, die einerseits aus den inhärenten Eigenschaf-

[1] Später (s. Def. 4.1) wird dieser Begriff formal näher erläutert.

[2] Boitet und Seligman (1994) reservieren den Begriff Chart für diesen einfachen Fall. Wir nutzen jedoch die allgemeinere Definition (s.u.).

[3] Wir werden später im Zusammenhang mit Hypergraphen (vgl. Abschnitt 2.6) eine (weitgehend natürliche) totale Ordnung über die Knoten einführen, die auf einer Abbildung auf die Zeitachse beruht.

[4] Für die Chartanalyse verwendet man manchmal leere aktive Kanten, die zwar nicht rückwärts in der Zeit weisen, aber trotzdem Zyklen bilden. Wir verzichten auf diese, indem wir in jedem Fall eine Left Corner Analyse nach Kilbury (1985) durchführen (s. Kapitel 4).

[5] Wirén (1992) modelliert die Rücknahme von Wörtern der Eingabe, indem er Korrekturen von Tippfehlern bei der Eingabe erlaubt.

ten gesprochener Sprache, andererseits aus den verfügbaren Mitteln zur Spracherkennung resultieren:

- Aufnahmebedingungen: Der wichtigste Ursprung einer suboptimalen Akustik liegt in der Präsenz von Störgeräuschen wie Hintergrundlärm etc. Solange die Störungen im wesentlichen stationär sind, können sie recht gut ausgefiltert werden. Problematischer sind nichtstationäre Geräusche wie Radiosendungen, Türenklappen usw.
- Unvollständige Erkennung: In bestimmten Fällen kann nicht garantiert werden, daß der Eingabepegel über die Zeit konstant ist. Dies kann etwa durch Bewegungen des Sprechers relativ zum Mikrophon hervorgerufen werden. Durch geeignete Aufnahmemodi (z.B. mit Nahbesprechungsmikrophonen) kann diese Art von Störungen allerdings minimiert werden.
- Sprechervariation: Neben dem schon klassischen Unterschied der Erkennungsgüte zwischen den Geschlechtern können dialektale Unterschiede sowie Unterschiede im Sprechtempo die Erkennung wesentlich erschweren.
- Verbundenheit: Der Mangel an klar definierten Wortzwischenräumen bei der Erkennung kontinuierlicher Sprache läßt den Suchraum, den ein Erkenner traversieren muß, explodieren. Ein Rückgriff auf Einzelworterkennung, die wesentlich sicherer ist, scheidet in der vorliegenden Anwendung aus.
- Performanzphänomene: Hier sind im wesentlichen Wortabbrüche, Häsitationen und Sprechergeräusche zu nennen. Performanzphänomene treten insbesondere gehäuft bei Spontansprache auf, während etwa bei vorgelesener Sprache deren Auftreten nicht sehr stark ins Gewicht fällt.
- Stochastikbasierung der vorherrschenden Erkenner: Moderne Erkenner sind fast ausschließlich HMM-basiert[6]. HMMs sind endliche Automaten, deren Zustandsübergänge und Emmissionen durch Verteilungen modelliert werden. Diese Verteilungen werden mit Hilfe von Lernverfahren auf der Basis einer großen Menge von Trainingsdaten geschätzt. Die Auswahl und Abdeckung der zum Training verwendeten Äußerungen determiniert maßgeblich die Performanz der Erkennungsvorgangs bei Bearbeitung bisher unbekannter Anwendungseingaben.

Grob gesagt, ist man nie sicher, daß ein Erkenner genau das erkennt, was gesagt wurde. Die Ausgabe eines Erkennungssystems spiegelt diese Situation wider. Prinzipiell werden drei Ausgabeschnittstellen vom Erkenner zu dahintergeschalteten linguistischen Modulen verwendet:

Die einfachste Form ist die der **besten Kette**. Der Erkenner liefert die mögliche Folge von Wörtern, die die beste akustische Bewertung erhält[7]. Der Vorteil dieses Verfahrens liegt darin, daß die Module zur Verarbeitung geschriebener Sprache in unveränderter Form auch gesprochene Eingaben

[6] *H*idden *M*arkov *M*odel, s. z.B. Woodland et al. (1995)

[7] D.h. die Wortkette, deren aufgrund der Trainingsdaten produzierte akustische Signale am besten mit der tatsächlichen Eingabe übereinstimmt.

akzeptieren können. Zudem lassen sich, genügend kleine Vokabularien vorausgesetzt, mit heutigen Erkennern erstaunliche Worterkennungsraten (90% und mehr) erzielen. Die beiden Hauptnachteile liegen in der mangelnden Skalierbarkeit beim Übergang auf größere Wortschätze und in der geringen Satzerkennungsrate. Überspitzt formuliert: Obwohl fast alle Wörter richtig erkannt werden, wird fast kein ganzer Satz richtig erkannt.

Eine naheliegende Erweiterung ist, nicht nur die beste Kette auszugeben, sondern mehrere Hypothesen, für den Fall, daß die akustisch beste Kette nicht die tatsächlich gesprochene Äußerung darstellt (**n-beste Hypothesen**) (vgl. etwa Tran, Seide, und Steinbiss, 1996). Man kann dies auf die Hoffnung stützen, daß die Wahrscheinlichkeit, eine verarbeitbare Äußerung zu erhalten, höher ist, wenn zehn oder zwanzig Möglichkeiten bestehen. Allerdings ist dies nur eine marginale Verbesserung, da die Anzahl der im Erkenner vorgehaltenen Hypothesen um Größenordnungen über der Zahl der ausgegebenen n besten Ketten liegt. Zudem erinnert die repetierende Bearbeitung von Ketten, die sich vielfach nur um ein einziges Wort unterscheiden, sehr an einen Batch-Betrieb.

Die Repräsentationsform, die sich in den letzten Jahren zunehmend durchsetzt, sind sog. **Wortgraphen** (Oerder und Ney, 1993; Aubert und Ney, 1995). Ein Wortgraph ist ein gerichteter, azyklischer Graph, dessen Kanten mit Bezeichnungen versehen sind, die in orthographischer Form Wörter denotieren. Zusätzlich sind Kantenbewertungen annotiert, die die akustische Bewertung[8] der entsprechenden Worthypothese modellieren.[9] Der Graph enthält ausgezeichnete Start- und Endknoten, die den Anfang bzw. das Ende der Äußerung kennzeichnen. Jeder Pfad durch den Graphen vom Anfangs- in den Endknoten stellt eine mögliche Äußerungshypothese dar. Jeder Knoten eines Wortgraphen steht zudem für einen Zeitpunkt während der Sprechzeit der Äußerung. Zur formalen Definition eines Wortgraphen benötigen wir zunächst den Begriff des Grades eines Knotens.

Definition 2.12 (Inzidenz, Grad eines Knotens)
Sei $G(\mathcal{V}, \mathcal{E})$ ein Graph. Eine Kante $vv' \in \mathcal{E}$ heißt inzident von v und inzident zu v'. Für einen Knoten v eines Graphen G ist der Eingangsgrad als Anzahl der Kanten inzident zu v definiert:

$$\#_{in}(:)\mathcal{V} \Leftrightarrow\!\!\rightarrow \mathbb{N}, \#_{in}(v) := \#\{e \in \mathcal{E} | \beta(e) = v\} \tag{2.9}$$

Der Ausgangsgrad eines Knotens v ist als Anzahl der Kanten inzident von v definiert:

$$\#_{out}(:)\mathcal{V} \Leftrightarrow\!\!\rightarrow \mathbb{N}, \#_{out}(v) := \#\{e \in \mathcal{E} | \alpha(e) = v\} \tag{2.10}$$

[8] Diese Bewertungen werden hier dargestellt als negative Logarithmen von Dichten, es gilt folglich, daß Kanten mit geringerer Bewertung besser sind.

[9] Spracherkenner bedienen sich häufig statistischer Sprachmodelle, die Übergangswahrscheinlichkeiten zwischen Wörtern modellieren. Dadurch wird die Anzahl der Wörter, die einem bestimmten Wort in einer Äußerung folgen können, eingeschränkt. Diese Übergangswahrscheinlichkeiten werden jedoch im resultierenden Wortgraphen nicht dargestellt, sie dienen aber zur Suchraumbeschränkung innerhalb des Erkenners. In Modulen zur linguistischen Verarbeitung können die auf einem Sprachmodell beruhenden Wahrscheinlichkeiten wieder benutzt werden.

Definition 2.13 (Wortgraph)
Ein Wortgraph ist ein gerichteter, azyklischer Graph $G(\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{W})$ mit Kantenbezeichnungen und Kantengewichten, für den gilt:

- *Der Graph besitzt einen ausgezeichneten Startknoten. Dieser stellt den Anfang der Äußerung dar.*

$$\exists v^{(r)} \in \mathcal{V} : \#_{in}(v^{(r)}) = 0 \wedge \forall v \in \mathcal{V} : \#_{in}(v) = 0 \Longrightarrow v = v^{(r)} \tag{2.11}$$

  Aus der Azyklizität und der Existenz von $v^{(r)}$ folgt bereits der schwache Zusammenhang von Wortgraphen.

- *Der Graph besitzt einen ausgezeichneten Endknoten. Dieser repräsentiert den Endpunkt der Äußerung.*

$$\exists v^{(f)} \in \mathcal{V} : \#_{out}(v^{(f)}) = 0 \wedge \forall v \in \mathcal{V} : \#_{out}(v) = 0 \Longrightarrow v = v^{(f)} \tag{2.12}$$

Jeder Knoten ist darüberhinaus mit einem Zeitpunkt annotiert, mit dessen Hilfe die Knoten linear entlang der Zeitachse angeordnet werden können. Die Zeit wird hierbei stets diskretisiert in Form von Zeitabschnitten der Länge 10ms, den sog. Frames.[10] *Wir definieren hierzu eine Funktion, die Knoten auf Frame-Nummern abbildet:*

$$t : \mathcal{V} \Leftrightarrow\!\!\!\rightarrow \mathbb{N} \tag{2.13}$$

t ist weder injektiv, da nicht garantiert ist, daß zu jedem Frameende ein Knoten existiert, noch surjektiv, da es unter bestimmten Voraussetzungen mehrere Knoten für einen Zeitpunkt geben kann.

Die Kantenbezeichnungen denotieren Wörter, die Kantengewichte akustische Bewertungen.

Ein Beispiel für einen typischen Wortgraphen ist in Abb. 2.3 zu sehen. Dieser Wortgraph ist allerdings nichtinkrementell erzeugt. Dies ist einfach zu sehen: Ein inkrementell erzeugter Wortgraph würde etliche "tote Enden" enthalten, die dadurch entstehen, daß ein Pfad ab einem bestimmten Punkt nicht weiter verfolgt werden kann, da keine weiterspannenden Worthypothesen mit genügender Konfidenz gefunden werden konnten.[11] In diesem Sinne muß die Forderung (2.12) modifiziert werden: Es kann beliebig viele Endknoten geben (Knoten, deren Ausgangsgrad Null ist), nach Ende der Bearbeitung stellen jedoch nur Wege durch den Graphen vollständige Äußerungen dar, die am Anfangsknoten beginnen und an dem Endknoten enden, der am weitesten rechts steht.

Wir tragen dieser Eigenschaft durch die folgende Definition Rechnung, die inkrementell erzeugte Wortgraphen (bei Weber (1995) linksverbundene Wortgraphen genannt) beschreibt.

[10] Diese Bezeichnung entstammt der Sprachsignalverarbeitung. Im Zuge der Vorverarbeitung werden die Schalldruckdaten auf sie beschreibende Merkmalvektoren abgebildet. Dies geschieht mit einem Zeitfenster, das jeweils um 10ms weitergeschoben wird.

[11] Die graphische Darstellung inkrementeller Wortgraphen ist weitgehend sinnlos, da die Verzweigungsdichte normalerweise so hoch ist, daß lediglich eine breite, geschwärzte Fläche entsteht (vgl. aber Abschnitt 4.10).

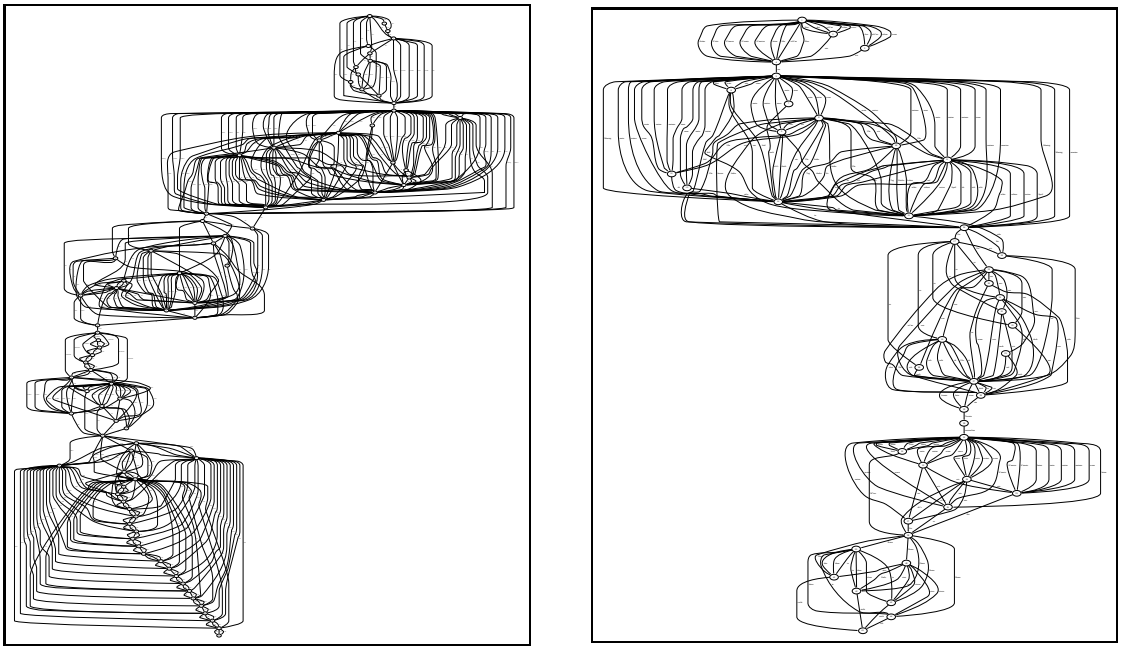


Abbildung 2.3: Wortgraphen zur Äußerung n002k000: “schön hervorragend dann lassen Sie uns doch noch einen Termin ausmachen wann wäre es Ihnen denn recht” Der linke Graph stellt die nichtinkrementelle Ausgabe des Worterkenners dar. Der rechte Graph ist auf eindeutige Wortsequenzen reduziert.

Definition 2.14 (Linksverbundener (Inkrementeller) Wortgraph)
Ein linksverbundener Wortgraph ist ein gerichteter, azyklischer Graph mit Kantenbezeichnungen und Kantengewichten, der einen ausgezeichneten Startknoten besitzt.

Durch die Möglichkeit, daß partielle Wortketten im Graphen geteilt werden (Teilpfade können identische Präfixe oder Postfixe besitzen), entsteht eine sehr kompakte Darstellung einer großen Menge alternativer Äußerungshypothesen. Der Graph zum Turn n002k000, der in Abb. 2.3 gezeigt ist, besitzt z.B. 461 Kanten, enthält aber $1,2 \cdot 10^{23}$ verschiedene Pfade. Allerdings besitzen auch Graphen einen hohen Grad an Redundanz dadurch, daß ein und dieselbe Wortkette auf verschiedenen Wegen durch den Graphen produziert werden kann, indem von einem Knoten zwei Kanten ausgehen, die mit demselben Wort annotiert sind, die aber in unterschiedlichen Knoten (z.B. zu unterschiedlichen Zeitpunkten) enden. So enthält der eben beschriebene Graph lediglich $8,6 \cdot 10^8$ unterschiedliche Wortsequenzen. Dieser reduzierte Graph ist im rechten Teil der Abb. 2.3 gezeigt[12].

Eine wichtige Eigenschaft von Wortgraphen ist die Möglichkeit zur Einführung einer topologischen Ordnung auf den Knoten des Graphen.

Definition 2.15 (Topologische Ordnung eines DAG)
Die topologische Ordnung eines gerichteten, azyklischen Graphen $G(\mathcal{V}, \mathcal{E})$ ist eine Abbildung $\tau : \mathcal{V} \leftrightarrow\!\!\!\rightarrow \mathbb{N}$ mit der Eigenschaft, daß gilt

$$\forall e \in \mathcal{E} : \tau(\alpha(e)) < \tau(\beta(e)) \tag{2.14}$$

Die topologische Ordnung eines DAG kann in der Zeit $O(|\mathcal{V}| + |\mathcal{E}|)$ berechnet werden. Algorithmus 1 beschreibt die Durchführung der topologischen Sortierung für Wortgraphen. Er ist geringfügig einfacher als der allgemeine Fall (vgl. Cormen, Leiserson, und Rivest, 1990, S. 486), da Wortgraphen schwach zusammenhängend sind. Insbesondere durch die topologische Sortierung von Wortgraphen werden viele der im folgenden dargestellten Algorithmen weniger komplex.

Die nächsten Abschnitte dieses Kapitels widmen sich allgemeinen Eigenschaften von Wortgraphen und Methoden zu deren Evaluation sowie Verfahren zur Größenreduktion. Wir werden dabei nicht näher auf linksverbundene Graphen eingehen, die uns erst später wieder beschäftigen.

2.3 Evaluation von Wortgraphen: Größen- und Gütemaße

Die Einführung von Wortgraphen in die maschinelle Sprachverarbeitung hat neue Fragenkomplexe aufgeworfen, was die Evaluation unterschiedlicher Verfahren und Systeme angeht. Insbesondere in der Spracherkennung spielen quantitative Evaluationen eine bedeutende Rolle, weist doch eine hohe Worterkennung auf die Güte des erkennenden Systems hin. War diese Angabe relativ einfach zu

[12] Der Algorithmus zur Reduktion wird weiter unten beschrieben.

---

begin
[1] Stack s;
Initialization
[2] **for** each vertex $v \in \mathcal{V}(G)$, **do**
[3] $inE_v := \#_{in}(v)$
[4] s.Push($v^{(r)}$)
[5] order $\leftarrow$ nil

Simplified Depth First Search
[6] **while** $v \leftarrow$ s.Pop() **do**
[7] order $\leftarrow$ order.v
[8] **for** each edge $e = (v, v')$ starting in v **do**
[9] $inE_{v'} \leftarrow inE_{v'} \Leftrightarrow 1$
[10] **if** $inE_{v'} == 0$ **then**
[11] s.Push(v')

[12] **return** order
end

Algorithmus 1: Berechnung der topologischen Ordnung eines DAG

machen, falls die Ausgabe des Erkenners lediglich aus einer Kette von Wörtern bestand, so kann man bei Wortgraphen deren Topologie nicht ganz außer acht lassen. Betrachtet man etwa die beiden Graphen aus Abb. 2.3, so erkennt man, daß sich die "Kompliziertheit" nicht isomorph auf den kompletten Graphen erstreckt, sondern es sind Bereiche zu erkennen, innerhalb derer offensichtlich eine hohe Ambiguität in Form der Erkennungsunsicherheit besteht, während einige Bereiche eher einfach erscheinen. Wesentlich sind außerdem die Verjüngungen des Graphen, an denen sich fast alle Pfade begegnen. Wir werden in diesem Abschnitt zunächst auf Größenmaße für Wortgraphen eingehen und ein neues Maß vorschlagen, daß unabhängig von bestehenden Transkriptionen ist und u.E. stärker auf die Einbindung eines spracherkennenden Systems in ein Gesamtsystem zur natürlichsprachlichen Verarbeitung eingeht. Daran schließt sich eine kurze Diskussion von Gütemaßen für Wortgraphen an, in deren Verlauf wir mögliche Alternativen vorstellen.

Zu Art und Form der Wortgraphen lassen sich unterschiedliche Anforderungen stellen, die leider im Allgemeinen gegenseitig unverträglich sind:

- Der Wortgraph soll klein sein, damit die linguistische Verarbeitung in kurzer Zeit Resultate liefern kann. Im Extremfall entspricht diese Forderung der lange üblichen Repräsentation durch die beste Kette.
- Der Wortgraph soll groß sein, damit die gesprochene Äußerung mit hoher Sicherheit im Graphen enthalten ist. Schon aus statistischen Gründen ist mit steigender Anzahl von Äußerungshypothesen mit einer höheren Worterkennungsrate zu rechnen.

Als Größenmaß für Wortgraphen sind bisher kantenbezogene Angaben gebräuchlich, so etwa die Dichte, die als Anzahl der Kanten pro Referenzwort definiert ist. Wir definieren hier zunächst zusätzlich eine Dichte, die unabhängig von einem Transkript ist:

Definition 2.16 (Dichte, Transkriptunabhängige Dichte)
Die (transkriptabhängige) Dichte eines Wortgraphen ist der Quotient aus der Anzahl der Worthypothesen in ihm und der Anzahl von Wörtern, die in der Referenztranskription enthalten sind.

Die transkriptunabhängige Dichte eines Wortgraphen $G = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{L})$ ist die mittlere Anzahl von Kanten, die einen Zeitpunkt überspannen.

Diese Dichte kann mit Hilfe von Algorithmus 2 in $O(|\mathcal{V}|)$ ermittelt werden, vorausgesetzt, die Knoten des Graphen sind topologisch sortiert.

Das in der Worterkennung gebräuchlichste Gütemaß ist das der Wortakkuratheit (*word accuracy*).

Definition 2.17 (Wortakkuratheit)
Gegeben eine von einem Worterkenner erkannte Sequenz von Wörtern und eine Referenztranslitteration der gesprochenen Äußerung, ergibt sich als Fehlermaß die Wortakkuratheit als

$$\textit{Wortakkuratheit} \quad = \quad 100 \Leftrightarrow 100 \times \frac{\#\textit{Fehler}}{\#\textit{Referenzwörter}} \tag{2.15}$$

begin
Initialization
[1] dens $\leftarrow$ 0
[2] totalDens $\leftarrow$ 0

Consider each vertex
[3] **for** each vertex $v \in \mathcal{V}$, taken in topological order, **do**
[4] totalDens $\leftarrow$ totalDens + dens
[5] dens $\leftarrow$ dens - $\#_{in}(v)$ + $\#_{out}(v)$

[6] **return** totalDens / $(|\mathcal{V}| \Leftrightarrow 1)$
end

Algorithmus 2: Berechnung der transkriptunabhängigen Dichte eines Wortgraphen

Als Fehler werden dabei Vertauschungen, Einfügungen und Auslassungen von Wörtern gezählt:

$$\#\textit{Fehler} \quad = \quad \#\textit{Vertauschungen} + \#\textit{Einfügungen} + \#\textit{Auslassungen} \tag{2.16}$$

Die analoge Übertragung der Wortakkuratheit von der Evaluation von Ketten auf die Bewertung von Erkennungsleistungen an Wortgraphen ist zumindest problematisch. Üblicherweise wird als Resultat dabei das Maximum der Wortakkuratheiten aller im Graphen vorhandenen Wortketten angenommen. Dazu wird per dynamischer Programmierung der Pfad durch den Graphen gesucht, auf dem die kleinste Anzahl von Fehlern auftritt. Um verschiedene Größen von Graphen zu berücksichtigen, wird die Akkuratheit als Funktion der Dichte angegeben.

Dabei wird größtenteils vernachlässigt, daß eine Erkennung von Wortgraphen i.a. nur sinnvoll ist in Zusammenhang mit einer nachgeschalteten linguistischen Analyse. Dies ist bei Erkennern, die lediglich die beste Kette abliefern, nicht unbedingt erforderlich, etwa bei “Diktaphon”-Anwendungen. Einer isolierten Evaluation kommt u.E. folglich nur eine untergeordnete Bedeutung bei (vgl. Amtrup, Heine, und Jost (1996), Amtrup, Heine, und Jost (1997)).

Eine realitätsnähere Evaluation bezieht das Verhalten von Modulen zur linguistischen Analyse von Wortgraphen ein. Der erste Schritt zur Etablierung eines Maßes für die Größe von Wortgraphen besteht darin, statt der Dichte eines Graphen die Anzahl der Pfade in ihm zu benutzen. Dies gründet in der Annahme, daß eine linguistische Analyse (wir werden im folgenden immer einen hypothetischen Parser annehmen) potentiell jeden Pfad im Graphen untersucht.

Die Anzahl der Pfade innerhalb eines Graphen wächst exponentiell mit der Anzahl der Knoten

in ihm (Oerder und Ney, 1993); wenn die Kanten gleichmäßig verteilt sind, enthält ein Graph $(\frac{|\mathcal{E}|}{|\mathcal{V}|})^{|\mathcal{V}|-1}$ Pfade. Der Algorithmus 3 berechnet die Anzahl der Pfade in einem Graphen, $p^{(G)}$. Die Komplexität des Algorithmus beträgt $O(|\mathcal{E}| + |\mathcal{V}|)$. Obwohl zwei Schleifen benutzt werden, so bestimmt die äußere (Zeile [1]) doch lediglich die Reihenfolge, in der die Kanten bearbeitet werden. Dabei wird keine Kante zweimal betrachtet.

---

begin
Handle each vertex
[1] **for** each vertex $v \in \mathcal{V}(G)$, taken in topological order, **do**

Compute the number of paths ending at v
$p^{(v)} := \sum p^{(w)} : wv \in \mathcal{E}$
[2] $p[v] \leftarrow\!\!\!\!\Leftrightarrow 0$
[3] **for** each edge $e = wv$ ending in v **do**
[4] $p[v] \leftarrow\!\!\!\!\Leftrightarrow p[v] + p[w]$
The number of paths in the graph is the number of paths up to the final vertex
$p^{(G)} \leftarrow\!\!\!\!\Leftrightarrow p^{(v^{(f)})}$
[5] **return** $p^{(G)}$
end

Algorithmus 3: Berechnung der Anzahl der Pfade in einem Graphen

Der nächste Schritt motiviert sich durch die Beobachtung, daß viele unterschiedliche Wege durch einen Wortgraphen dieselbe Wortsequenz erzeugen. Folglich ist eine weitergehende Einschränkung, lediglich die Anzahl der unterschiedlichen Pfade zu zählen. Um dies zu erreichen, wird der Graph einer Transformation unterzogen, die die Topologie verändert. Dies ist allerdings nicht übermäßig kritisch, da sich zwei gleich bezeichnete, aber unterschiedliche Pfade durch einen Graphen nur in zwei Aspekten unterscheiden können:

- Die Menge der Knoten, die besucht werden. Diese ist für einen Parser normalerweise irrelevant, da die genaue Abbildung der Wörter in die Zeit keine große Rolle spielt. Zu beachten ist dabei allerdings, daß bei Einbeziehung anderer Wissensquellen, die zeitlich genau limitierte Information liefern — z.B. prosodische Information über Wort- oder Phrasengrenzen bzw. Akzentuierung oder Eingaben in anderen Modalitäten wie Zeigegesten, Lippenbewegungen u.ä. — auf einer Invarianz der Wortsequenzen bezüglich der Zeit bestanden werden muß.
- Die akustischen Bewertungen, mit denen die Worthypothesen annotiert sind. Idealerweise sollte der Algorithmus, der einen Wortgraphen auf eindeutige Sequenzen von Bezeichnungen reduziert, die relativen Bewertungen aller Wege erhalten. Kann dies nicht geschehen, wie dies für unseren Algorithmus gilt, so sollte die Bewertung eines Weges wenigstens nicht schlechter werden.

Algorithmus 4 beschreibt das Verfahren der Einschränkung auf eindeutige Bezeichnungs-Sequenzen. Er stellt sicher, daß kein Knoten von zwei Kanten verlassen wird, die dieselbe Bezeichnung tragen. Diese lokale Bedingung garantiert, daß aus globaler Sicht keine unterschiedlichen Pfade dieselbe Sequenz von Worthypothesen tragen können. Das Verfahren entspricht dem aus Hopcroft und Ullman (1979) zur Umwandlung eines nichtdeterministischen in einen deterministischen endlichen Automaten (vgl. auch Seligman, Boitet, und Hamrouni, 1998).

---

begin
[1] **for** each vertex $v \in \mathcal{V}(G)$, taken in topological order, **do**
[2] **for** each pair of identically labeled edges e_1, e_2 **do**
Perform edge merging and create new vertex
[3] Create a new vertex v having
$t(v) := \min(t(\beta(e_1)), t(\beta(e_2)))$,
inserting v into the topological order
Copy all edges incident from $\beta(e_1)$ to v
[4] **for** each edge $e = (\beta(e_1), w, s, y)$ **do**
[5] Create a new edge $e' := (v, w, s, y)$
Copy all edges incident from $\beta(e_2)$ to v
[6] **for** each edge $e = (\beta(e_2), w, s, y)$ **do**
[7] Create a new edge $e' := (v, w, s, y)$
Delete e_1, e_2
end

Algorithmus 4: Beschränkung eines Graphen auf eindeutige Bezeichnungs-Sequenzen

Die Komplexität des Algorithmus ist im schlimmsten Fall exponentiell in der Anzahl der Knoten des Graphen. Abb. 2.4 zeigt einen Graphen, an dem dies anschaulich gemacht werden kann. Jeder Knoten außer den letzten beiden wird von zwei identisch bezeichneten Kanten verlassen, von denen eine zum nächsten Knoten, eine zum übernächsten Knoten verläuft. Für jedes Paar von solchen Kanten muß bei Anwendung des Algorithmus ein neuer Knoten erzeugt werden, der zweimal soviele ausgehende Kanten besitzt wie der ursprüngliche. Wenn wir etwas allgemeiner d paarweise gleich bezeichnete Kanten annehmen, die einen Knoten verlassen, enthält der Graph $2\,|\mathcal{V}| \Leftrightarrow 1$ Kanten.

Bei Bearbeitung des ersten Knotens müssen $\frac{d}{2}$ neue Knoten erzeugt werden, jeder mit $2\,d$ ausgehenden Kanten. Wird einer dieser neuen Knoten betrachtet, entstehen erneut $\frac{d}{2}$ Knoten, diesmal jedoch mit jeweils $4\,d$ Kanten. Dieser Zyklus kann $\frac{|\mathcal{V}|}{2} \Leftrightarrow 1$ mal wiederholt werden, da bei jedem Schritt zwei Knoten fortgeschritten wird. Währenddessen werden

$$\sum_{i=1}^{\frac{|\mathcal{V}|}{2}-1} 2^i\,d \tag{2.17}$$

neue Kanten erzeugt, die alle vom Algorithmus bearbeitet werden müssen. Dies resultiert in mindestens

$$d(2^{\frac{|\mathcal{V}|}{2}} \Leftrightarrow 2) \tag{2.18}$$

Operationen, die Komplexität des Algorithmus ist folglich $\Theta(\sqrt{2^{|\mathcal{V}|}})$.

Der hier vorgeschlagene Algorithmus ist nicht auf reale Wortgraphen angewendet worden, dazu ist die Komplexität zu hoch. Eine modifizierte Version, die im folgenden vorgestellt wird, gehört zwar in dieselbe Komplexitätsklasse, durch einige Optimierungen konnte aber eine akzeptable Laufzeit erreicht werden, so daß Evaluationen von Wortgraphen möglich wurden.

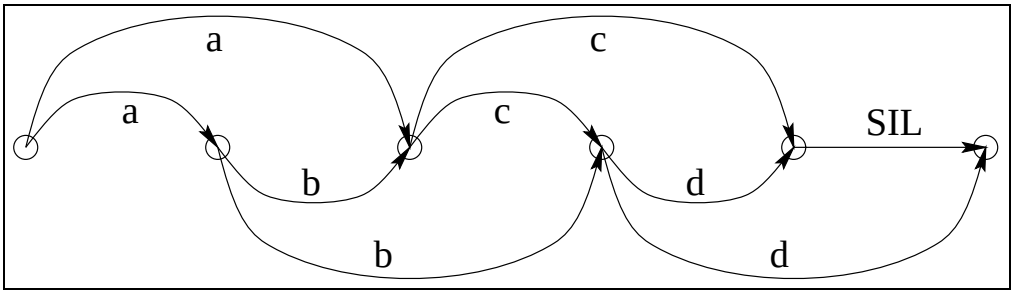


Abbildung 2.4: Ein schwieriger Graph bezüglich der Reduktion auf eindeutige Bezeichnungssequenzen

Die Verschmelzung von Knoten hat hierbei den größten positiven Effekt auf die Laufzeit. Die in Algorithmus 5 dargestellte Verschmelzung kopiert Kanten von einem Knoten zu einem anderen und führt dabei einen Redundanztest durch. Doppelte Kanten werden nicht eingesetzt, lediglich die akustische Bewertung wird ggfs. modifiziert. Um die Integrität des Wortgraphen nicht zu beschädigen, dürfen ausschließlich gegenseitig nicht erreichbare Knoten verschmolzen werden.

Das Verschmelzen von Knoten kann unter verschiedenen Bedingungen stattfinden:

- Zwei Knoten werden verschmolzen, wenn sie dieselben Mengen von Kanten inzident zu ihnen und inzident von ihnen besitzen. Dies kann allerdings zu einer leichten Erhöhung der Anzahl der Pfade im Graphen führen, da neue partielle Pfade der Länge 2 durch das Verschmelzen eingeführt werden können.
- Zwei Knoten werden verschmolzen, wenn sie dieselben Mengen von Knoten inzident von ihnen besitzen.
- Zwei Knoten werden verschmolzen, wenn sie zum selben Zeitpunkt gehören. Diese Situation kann entstehen, wenn im Laufe der Anwendung des Algorithmus neue Knoten erzeugt werden.

Das Verschmelzen von Knoten wird immer dann ausgeführt, wenn ein neuer Knoten betrachtet wird. Zusätzlich kann ein Verschmelzen stattfinden, wenn die Kanten, die einen Knoten verlassen,

procedure Merge(v, w)

Merge two vertices v and w
First, copy w's ingoing edges
[1] **for** each edge $e = (x, w, s, l)$, **do**
[2] **if** $\exists e' = (x, v, s', l)$, **then**
[3] $s' \leftarrow\!\!\!\!\!\!\Leftrightarrow min(s, s')$
[4] **else**
[5] Create a new edge $e'' = (x, v, s, l)$

Now, copy w's outgoing edges
[6] **for** each edge $e = (w, x, s, l)$, **do**
[7] **if** $\exists e' = (v, x, s', l)$, **then**
[8] $s' \leftarrow\!\!\!\!\!\!\Leftrightarrow min(s, s')$
[9] **else**
[10] Create a new edge $e'' = (v, x, s, l)$
end

Algorithmus 5: Verschmelzen zweier Knoten

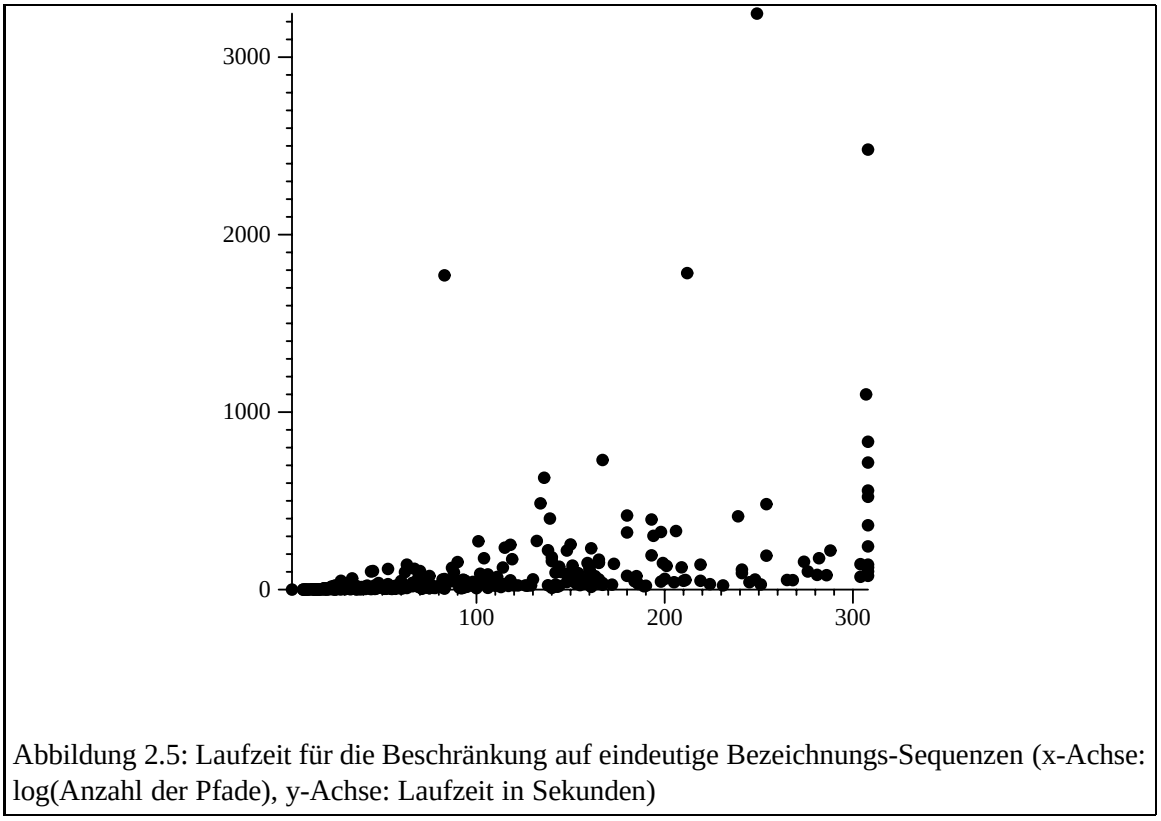


Abbildung 2.5: Laufzeit für die Beschränkung auf eindeutige Bezeichnungs-Sequenzen (x-Achse: log(Anzahl der Pfade), y-Achse: Laufzeit in Sekunden)

abschließend betrachtet wurden. Die Knoten, die dadurch neu erzeugt wurden, besitzen dieselbe Menge von Knoten inzident zu ihnen (nämlich den gerade bearbeiteten Knoten) und sind deshalb gute Kandidaten für das Verschmelzen.

Durch die Modifikationen des ursprünglichen Algorithmus konnte eine akzeptable Laufzeit erreicht werden, allerdings um den Preis, daß die Bewertungen von Pfaden durch den Graphen verändert (allerdings nur verbessert) wurden, möglicherweise sogar zusätzliche Pfade eingeführt wurden. Abb. 2.5 zeigt eine Übersicht der Laufzeit des Algorithmus bei Anwendung auf die Graphen der Verbmobil-Evaluation 1996 (Reinecke, 1996).[13]

Bisher ist bei der Annäherung an ein realitätsnahes Maß für die Größe von Wortgraphen auf die Anzahl der Pfade eingegangen worden. Allerdings muß beachtet werden, daß kein existierendes linguistisches System (z.B. ein Parser) alle Pfade aufzählen wird. Die letzte Erweiterung, die wir hier vornehmen werden, zielt folglich auf das Verhalten eines vernünftigen, komplexen Parsers ab.

Der fiktive Parser, den wir als Abnehmer von Wortgraphen betrachten, benutzt einen unifikationsbasierten Merkmalformalismus. Dies verhindert eine kubische Komplexität des Parsers, da wir für je zwei Kanten, die von einer Regel der Grammatik erzeugt werden, unterschiedliche Merkmale konstruieren können (etwa die Konkatenation der annotierten Worthypothesen) (vgl. Weber (1995) zur Komplexität kontextfreien Parsings von Wortgraphen, das in die Klasse $O(|\mathcal{V}|^3)$ gehört). Wir beschränken uns dabei auf Grammatiken mit einem kontextfreien Regelgerüst sowie annotierten Merkmalstrukuren, und fordern zusätzlich die Chomsky-Normalform, d.h. eine Regel besitzt höchstens zwei Nichtterminale auf der rechten Seite.[14] Wir machen jedoch keine weiteren Annahmen, z.B. über das Suchverhalten des Parsers.

Als Maß für den Aufwand, den ein Parser zur Bearbeitung eines Graphen treiben muß, sehen wir die Anzahl der Bearbeitungsschritte an, die er zur vollständigen Analyse des Graphen benötigt. Betrachtet man die Anzahl der Operationen $d^{(p)}$, die zur Analyse einer Äußerungshypothese p erforderlich sind, ist

$$d^{(p)} = \frac{n^3 \Leftrightarrow n}{6} \tag{2.19}$$

, wobei n die Länge des Pfades denotiert. Diese Zahl korrespondiert zur Anzahl der Schritte zur Füllung der Ableitungsmatrix des CKY-Algorithmus (vgl. Mayer (1986, S. 107)).

Wenn man annimmt, daß alle Pfade innerhalb eines Graphen voneinander unabhängig sind, so ergibt sich für die Anzahl der benötigten Ableitungsschritte für den Graphen insgesamt

[13] Es handelt sich um 305 Äußerungen ohne Buchstabiereinheiten aus Dialogen der Verbmobil CD 14.

[14] Da eine Grammatik Ambiguitäten enthalten kann, wird es passieren, daß bei Kombination zweier Kanten mehr als ein Resultat erzeugt wird. Wir abstrahieren hiervon ebenso wie von dem Umstand, daß aufgrund von Grammatikrestriktionen zwei Kanten möglicherweise nicht kombiniert werden dürfen.

$$d^{(G)} = \sum_{p \in G} d^{(p)} \tag{2.20}$$

Dies legt eine lineare Abhängigkeit zwischen der Anzahl der Pfade in einem Graphen und der Anzahl der Operationen zur Bearbeitung desselben nahe. Allerdings ist eine Grundeigenschaft von Wortgraphen, daß partielle Pfade geteilt werden, 2.20 ist folglich nur eine obere Abschätzung. Um gemeinsam genutzte Teilgraphen einzubeziehen, muß Algorithmus 6 angewendet werden.

Neben den bereits erwähnten Annahmen über die Grammatik (fehlende Ambiguität und keinerlei Restriktionen über die Anwendbarkeit von Regeln) gehen wir bei diesem Verfahren zusätzlich davon aus, daß es an einem Knoten beliebig viele partielle Analysen geben darf, was jeder "vernünftige" Parser für gesprochene Sprache durch die Anwendung von Beschränkungsmechanismen (*Pruning*) verhindern würde.

Die Komplexität des Algorithmus 6 ergibt sich zu $O(|\mathcal{E}||\mathcal{V}|)$. Die äußere (Zeile [2]) und die innerste (Zeile [5]) Schleife garantieren die Bearbeitung aller Kanten und regeln die Reihenfolge der Behandlung. Die Schleife in Zeile 4 erhöht die Anzahl der Ableitungen bis zum gegenwärtigen Knoten. Dabei werden alle kürzeren Ableitungen bis dahin verwendet. Diese Sequenzen können maximal eine Länge von $|\mathcal{V}|$ besitzen, was die Komplexität ergibt.

2.4 Evaluation von Wortgraphen: Gütemaße

Im vorhergehenden Abschnitt haben wir verschiedene Größenmaße für Wortgraphen betrachtet sowie eine wichtige Modifikation vorgestellt, die entscheidenden Einfluß auf die Größe eines Graphen haben kann. Auf Qualitätsmaße war dabei nur implizit eingegangen worden. In diesem Abschnitt werden wir kurz andere Qualitätsmaße einführen, die allerdings sämtlich auf der Wortakkuratheit basieren, indem sie stets die "beste Kette" sowie deren Eigenschaften in Bezug auf den Wortgraphen untersuchen. Sich als Qualitätsmaß ausschließlich auf die Akkuratheit zu verlassen, kann problematisch sein, da sie (wie die bisherigen kantenbasierten Größenmaße) die Form des Graphen nicht berücksichtigt. Prinzipiell sind andere, integrierte Maße denkbar, die die Topologie eines Wortgraphen in Rechnung stellen. Als Güte eines Wortgraphen bezogen auf eine Referenzäußerung könnte gelten:

- Die Differenz der akustischen Bewertungen zwischen der akustisch besten Kette durch den Graphen und der richtigen Kette[15].
- Die Wahrscheinlichkeit der richtigen Kette bezogen auf den Eingabegraphen.

[15] Wir bezeichnen hier als "richtige Kette" diejenige mit der höchsten Wortakkuratheit bezogen auf die Referenztranskription.

---

begin

Initialization

[1] totalderiv $\leftarrow\!\!\Leftrightarrow 0$

Handle each vertex

[2] **for** each vertex $v \in \mathcal{V}(G)$, taken in topological order, **do**

Adjust the number of rule applications for this vertex and the total number of derivations so far.

[3] deriv$_v[1] \leftarrow\!\!\Leftrightarrow \#_{in}(v)$

[4] **for** all $i \in \{2, \ldots, |\mathcal{V}|\}$ do

[5] **for** each edge $e = (w, v, x, y)$ ending in v **do**

[6] deriv$_v[i] \leftarrow\!\!\Leftrightarrow$deriv$_v[i]$ + deriv$_w[i \Leftrightarrow 1]$

[7] totalderiv $\leftarrow\!\!\Leftrightarrow$totalderiv + deriv$_v[i]$

totalderiv holds the number of derivations

[8] **return** totalderiv

end

Algorithmus 6: Berechnung der Anzahl der Ableitungsschritte eines fiktiven Parsers für einen Wortgraphen

- Die Anzahl der Verarbeitungsschritte, die ein fiktiver Parser benötigt, um die richtige Kette zu finden.
- Der Rang der richtigen Kette, bezogen auf die akustische Bewertung.

Die Motivation, den Rang der richtigen Kette bezogen auf die akustische Bewertung als Maß für die Güte eines Graphen anzunehmen, beruht erneut auf der Vorstellung, ein nachgeschalteter Parser werde im schlimmsten Fall alle Pfade absuchen, bis er den richtigen gefunden hat. Dies abstrahiert natürlich von der Möglichkeit, daß bereits ein akustisch besserer Pfad, der eine höhere Anzahl von Fehlern enthält, aufgrund von Eigenschaften der Grammatik als korrekt erkannt wird. Das Verfahren zur Bestimmung des Ranges einer Kette ist in Algorithmus 7 angegeben. Genauer gesagt, ermittelt der Algorithmus den Rang einer Kette mit einer bestimmten Gesamtbewertung, gemessen an den Bewertungen aller anderen möglichen Pfade durch den Graphen. Die Eingabe für den Algorithmus besteht deswegen aus einem Wortgraphen und einer Referenzbewertung s_{ref}, deren Rang zu ermitteln ist.

---

begin

Compute minimum and maximum prefix scores

[1] **for** each vertex $v \in \mathcal{V}(G)$, taken in topological order, **do**

[2] $s_{v,min} \leftarrow\!\!\!\!\Leftrightarrow \infty$

[3] $s_{v,max} \leftarrow\!\!\!\!\Leftrightarrow 0$

[4] **for** each edge $e = (w, v, s_e, y)$ ending in v **do**

[5] $s_{v,min} \leftarrow\!\!\!\!\Leftrightarrow \min(s_{w,min} + s_e, s_{v,min})$

[6] $s_{v,max} \leftarrow\!\!\!\!\Leftrightarrow \max(s_{w,max} + s_e, s_{v,max})$

Compute rank of path with score s_{ref} recursively, starting at the final vertex

[7] $r \leftarrow\!\!\!\!\Leftrightarrow$ **ComputeRank (**v_f, s_{ref}**)**

r is the rank of a path with score s_{ref} through the graph

[8] **return** r

end

Algorithmus 7: Ermittlung des Ranges eines Pfades

Eine naive Suche zählt alle Pfade auf, bis einer gefunden ist, der die gesuchte Bewertung trägt. Der hier dargestellte Algorithmus basiert hingegen darauf, Kanten am Ende des Pfades zu entfernen und dabei minimale und maximale Bewertungen an Knoten zu vergleichen. Dadurch können in günstigen Fällen Bündel von Pfaden in einem Schritt betrachtet werden; Dies sollte für viele Fälle effizienter sein als eine einfache akustische Bestensuche.

Trotzdem hat der Algorithmus im schlechtesten Fall eine exponentielle Komplexität in Abhängigkeit von der Anzahl der Knoten im Graphen. Das liegt daran, daß bei ungünstiger Verteilung der

Bewertungen doch der gesamte Graph pfadweise abgesucht wird. Nehmen wir an, ein Graph besitze $|\mathcal{V}|$ Knoten und $n(|\mathcal{V}| \Leftrightarrow 1)$ Kanten für ein ungerades n. Abbildung 2.6 zeigt einen derartigen Graphen mit vier Knoten und $n = 5$.

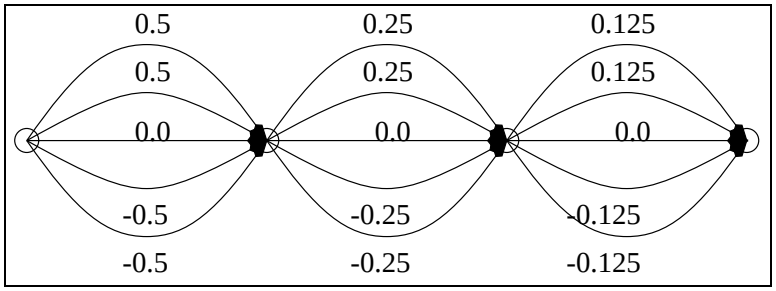


Abbildung 2.6: Ein komplexer Graph zur Bestimmung des Ranges einer Kette

Die Kanten sind gleichmäßig zwischen den Knoten verteilt, jeweils n Kanten verbinden zwei benachbarte Knoten. Die Knoten seien von 1 bis $|\mathcal{V}|$ durchnumeriert, gemäß der topologischen Sortierung des Graphen. Die Gewichte der Kanten zwischen dem Knoten i und dem Knoten $i + 1$ seien wie folgt vergeben:

- Die Kante, die zum Referenzpfad gehört, besitze das mittlere Gewicht, z.B. 0.
- Die Hälfte der restlichen Kanten habe das Gewicht $\frac{1}{2^i}$.
- Die restlichen Kanten haben das Gewicht $\Leftrightarrow\frac{1}{2^i}$.

Durch die Zeilen [1] bis [6] des Alg. 7 werden die Unter- und Obergrenzen möglicher Bewertungen von Teilpfaden von der Wurzel des Graphen bis zu jedem Knoten ermittelt. Danach stellt die Funktion **ComputeRank()**, die als Algorithmus 8 gezeigt ist, das Kernstück der Ermittlung des Ranges einer Kette dar. Falls die gesuchte Bewertung kleiner als die minimale oder größer als die maximale Bewertung bis zu einem Knoten ist, brauchen alle Knoten links davon nicht betrachtet zu werden, die Lösung (wieviele Pfade links von dem Knoten sind besser bzw. schlechter als der Referenzpfad) läßt sich anhand von $s_{v,max}$ bzw. $s_{v,min}$ ablesen. Fällt die Referenzbewertung hingegen in $]s_{v,min}, s_{v,max}[$, so wird rekursiv weiterverfahren.

Angewendet auf den Beispielgraphen führt das dazu, daß die Funktion zunächst am Endknoten mit einer Referenz von 0 aufgerufen wird. Da diese weder kleiner als die minimale noch größer als die maximale Bewertung bis dahin ist, resultieren n weitere Aufrufe von **ComputeRank()**. Da kein partieller Pfad, der von einem internen Knoten des Graphen zum Endknoten läuft, eine kumulierte Bewertung haben kann, deren Absolutbetrag größer ist als die einer Kante, die an dem Startknoten des partiellen Pfades endet, folgt, daß an jedem Knoten erneut n Aufrufe von **ComputeRank()** generiert werden.

function ComputeRank (v, s_{ref} **)**

If the reference score is lower or equal than the minimum score, all paths left from here are worse and need not be considered

[1] **if** $s_{ref} \leq s_{v,min}$

[2] **return** 0

If the reference score is greater or equal than the maximum score, all paths left from here are better and need not be considered

[3] **if** $s_{ref} \geq s_{v,max}$

[4] **return** $p^{(v)}$

Now comes the hard part. Recursively consider all vertices incident to v

[5] $r_{tmp} \leftarrow\!\!\!\!\!\Leftrightarrow 0$

[5] **for** each edge $e = (w, v, s_e, y)$ ending in v **do**

[6] $r_{tmp} \leftarrow\!\!\!\!\!\Leftrightarrow r_{tmp}$+**ComputeRank (**$w, s_{ref} \Leftrightarrow s_e$**)**

[7] **return** r_{tmp}

end

Algorithmus 8: Ermittlung des Ranges eines Pfades (Teil 2)

Insgesamt werden folglich

$$\prod_{i=1}^{|\mathcal{V}|-1} n \quad = \quad n^{|\mathcal{V}|-1} \tag{2.21}$$

Aufrufe getätigt, so daß die Komplexität im schlechtesten Fall $\Theta(n^{|\mathcal{V}|})$ beträgt.

Alle angedeuteten Maße sind von der Wortakkuratheit abgeleitet. An dieser Stelle wurde lediglich die letzte Möglichkeit näher erläutert, zur Evaluation von Graphen werden wir aber weiterhin ausschließlich die Wortakkuratheit benutzen. Der grund dafür liegt einerseits an der Vergleichbarkeit mit der für beste-Kette Erkenner verwendeten Evaluationsmetrik, die erhalten werden soll, und andererseits an den inhärenten Problemen, die jegliche Art von Graphenevaluation bietet.

Sämtliche alternativen Evaluationsmaßstäbe, die wir hier eingeführt haben, versuchen ein Maß für den Aufwand zur Auffindung des besten Ergebnisses zu geben. Alleine diese Angabe zu liefern, führt jedoch dazu, daß alle Erkenner, die ausschließlich die beste Kette ausgeben, identisch bezüglich dieses Maßes sind. Eine Evaluation müßte folglich mindestens ein Paar von Werten als Ergebnis haben, nämlich ein Qualitätsmaß (wie die Wortakkuratheit) und ein Aufwandsmaß (wie

den Rang der richtigen Kette).

Dieses zweite Maß gibt jedoch nicht unbedingt ein korrektes Bild davon, wie hoch der Aufwand zur Erzeugung einer Analyse in nachgeschalteten Analysemodulen tatsächlich ist. Abhängig von der Art dieser Module, der Durchlässigkeit der assoziierten Grammatiken etwa, wird nämlich nicht der gesamte Suchraum bis zur richtigen Kette aufgespannt. Vielmehr sind moderne Systeme häufig so konstruiert, daß sie extrem generelle Wissensquellen entalten, so daß im Extremfall bereits der erste betrachtete Pfad eine erfolgreiche Bearbeitung ergibt, selbst wenn die Wortfehlerrate wesentlich höher als möglich ist. Auf der anderen Seite sagt die Wortakkuratheit nichts darüber aus, welche Wörter mit der Transkription übereinstimmen. Das kann im ungünstigsten Fall dazu führen, daß die wesentlichen Inhaltswörter sämtlich fehlen oder falsch erkannt werden, und lediglich irrelevante Anteile der Äußerung richtig erkannt wurden.

Ein integriertes Maß für die Qualität von Graphen zu erhalten, das die "Erfolgschancen" für linguistische Analysen und den zu ihrer Erstellung notwendigen Aufwand korrekt abbildet, ist so u.E. nahezu ausgeschlossen. Die Konsequenz daraus ist, daß wir an der Wortakkuratheit als Gütemaß festhalten und eine lediglich eine Änderung herkömmlicher Größenmaße für Wortgraphen favorisieren.

2.5 Weitere Operationen auf Wortgraphen

Kanten, die als Annotation Stille ($<$ sil $>$) tragen, sind eine gesonderte Betrachtung wert. Dies liegt daran, daß meistens angenommen wird, sie trügen keine Bedeutung und würden in Modulen zur linguistischen Verarbeitung ignoriert. So wird das Einfügen von Stille-Kanten in Wortgraphen beispielsweise nicht als Fehler bei der Evaluation von Wortgraphen gewertet.[16] In diesem Sinne kann man zur Verbesserung der Leistung eines Erkenners danach trachten, möglichst viele Stille-Kanten sus einem Graphen zu entfernen. Damit verbessert man zwar nicht die Wortakkuratheit, reduziert aber die Dichte des Graphen (durch die Entfernung von Kanten) und dadurch die Leistung des Erkenners relativ zur Dichte. Wir werden hier zunächst drei Methoden vorstellen, mit Stille zu verfahren, und danach einen Algorithmus präsentieren, der durch ein unorthodoxes Ausnutzen der gängigen Evaluationsmaße die Erkennung scheinbar verbessert.

2.5.1 Entfernung einfacher Stille

Der erste Algorithmus, den wir hier vorstellen wollen, entfernt einfache Stille-Kanten, die zwei Knoten verbinden. Er geht davon aus, daß die Stille die einzige Kante zwischen zwei Knoten darstellt. Dieser Fall ist selten, er tritt normalerweise nur am Anfang und Ende eines Wortgraphen auf,

[16] Pausen innerhalb einer Äußerung sind aus linguistischer Sicht sehr wohl relevant. Sie können z.B. dazu verwendet werden, die Anwendung gewisser syntaktischer Regeln zu steuern, etwa im Fall von Grammatiken, die Turns mit mehreren Teiläußerungen beschreiben (Kasper und Krieger, 1996). In diesen Fällen sollte jedoch ein spezialisierter Erkenner für prosodische Ereignisse verwendet werden (Strom und Widera, 1996).

wo sich per definitionem Stille-Kanten befinden sollen. Die Anzahl dieser Fälle kann allerdings ansteigen, falls durch andere, oben beschriebene Methoden, die Form des Wortgraphen verändert wird.

---

begin
[1] **for** each vertex $v \in \mathcal{V}$, taken in topological order, **do**
Search for silences leaving v
[2] **for** each edge $e = (v, w, s, < SIL >)$ **do**
[3] **if** $v \overset{*}{\not\rightarrow} w$, disregarding e, **then**
Adjust scores
[4] **for** each edge $e' = (v, w', s', l'), e \neq e$, **do**
[5] $s' \leftarrow\!\!\!\!\leftrightarrow s' + s$
Merge vertices v *and* w
[6] **Merge(**v, w**)**
[7] Delete w
end

Algorithmus 9: Entfernung isolierter Stille-Kanten

Die Grundidee des Algorithmus 9 ist, zwei Knoten zu verschmelzen, falls sie ausschließlich durch eine Stille-Kante verbunden sind. Natürlich muß, um Zyklen zu vermeiden, darauf geachtet werden, daß die in Frage kommenden Knoten nicht anderweitig gegenseitig erreichbar sind. In einer Version des Algorithmus wurde ein zusätzliches Constraint eingeführt, das die zeitliche Integrität des Wortgraphen erhält. Das Verschmelzen von Knoten wurde unterbunden, falls dadurch Kanten entstünden, die rückwärts in der Zeit weisen. Diese Einschränkung ist allerdings nicht wesentlich für Algorithmus 9 und dort auch nicht vermerkt. In jedem Fall muß jedoch eine Neubewertung der kopierten Kanten erfolgen, um die akustische Bewertung der entfernten Stille-Kante zu berücksichtigen; die Bewertungen von Pfaden sollen durch die Operation nicht verändert werden.

2.5.2 Entfernung zusammenhängender Stille

Die einzige Worthypothese, deren Repetition ohne Bedeutung bleibt, ist die Stille. Falls zwei hintereinanderliegende Stille-Hypothesen auftreten, so können diese verschmolzen werden. Die einfachste Situation ist die in Abb. 2.7 a) gezeigte. Die beiden Kanten können durch eine einzige ersetzt werden. Allerdings tritt dieser Fall selten auf, etwa, wenn am Verbindungsknoten Kanten entfernt wurden.

Die interessantere Situation zeigt Abb. 2.7 b). Hier müssen Kanten, die an dem Verbindungsknoten beginnen oder enden, geeignet behandelt werden, damit keine größeren Modifikationen am Graphen auftreten. Wir beschränken die Verschmelzung auf solche Fälle, bei denen es keine Kanten gibt, die innerhalb des interessanten Zeitintervalls liegen, um eventuelle Zyklenbildung zu verhindern. Für

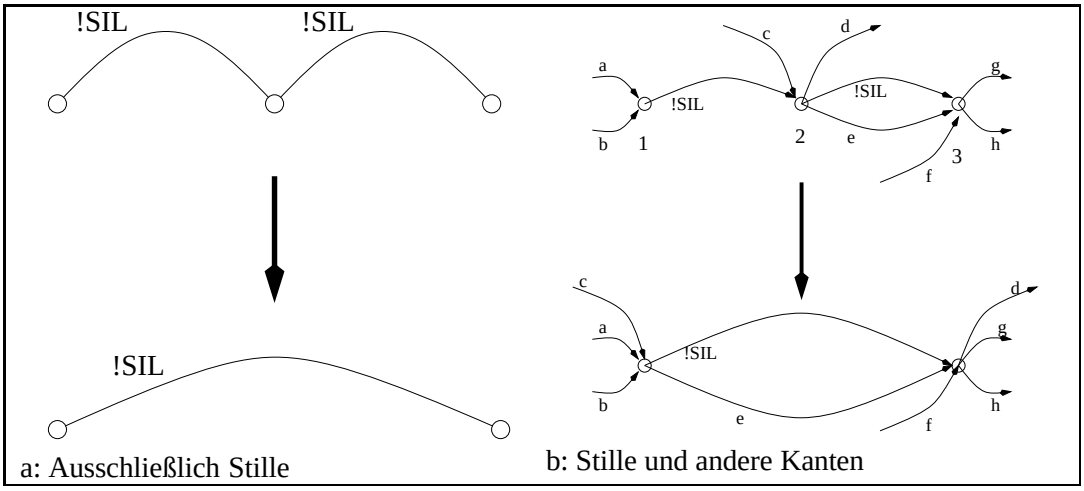


Abbildung 2.7: Verschmelzung von Stille

die Verlagerung von Kanten, die am Verbindungsknoten der Stille-Kanten beginnen oder enden, gibt es mehrere Möglichkeiten. In der in Abb. 2.7 b) gezeigten Situation werden die Kanten inzident zu Knoten 2 in den Knoten 1 verschoben, die Kanten inzident von Knoten 2 in den Knoten 3. Dadurch werden eventuell neue partielle Pfade in den Graphen eingeführt (hier die Kantenfolge $c \Leftrightarrow e \Leftrightarrow d$). Falls andererseits d an Knoten 1 und c an Knoten 3 gebunden wird, verschwinden Pfade.[17]

Dadurch, daß jetzt jedoch eine größere Anzahl von Kanten zu den Randknoten adjazent ist, kann durch die Entfernung identischer Worthypothesen die Anzahl der Pfade normalerweise merklich eingeschränkt werden. Das Verfahren zur Entfernung zusammenhängender Stille ist in Algorithmus 10 angegeben.

2.5.3 Entfernung aller Stille-Kanten

Die radikalste Form des Umgangs mit Stille-Kanten ist, sie generell aus einem Wortgraphen zu entfernen. Algorithmus 11 beschreibt das Vorgehen. Alle Kanten, die an dem Endknoten einer Stille-Kante enden, werden an den Anfangsknoten der Stille kopiert, und ihre Bewertungen werden korrigiert. Dies darf jedoch nicht am Ende des Graphen stattfinden, um mehrfache Endknoten zu vermeiden.

Durch die Kopieroperationen wird zwar u.U. die Anzahl der Kanten in einem Graphen erhöht, die Anzahl der Pfade wird jedoch abnehmen, da einerseits identische Kanten, die während des Kopierens auftreten, nicht in den Graphen eingesetzt werden, andererseits durch die Anwendung anderer

[17] Die beiden restlichen Möglichkeiten werden hier nicht diskutiert.

begin
[1] **for** each vertex $v \in \mathcal{V}$, taken in topological order, **do**
Search for two silences
[2] **if** $\exists e_1 = (v_1, v, s, < SIL >), e_2 = (v, v_2, s', < SIL >) \in \mathcal{E}$, **then**

Check applicability
[3] **for** each edge $e = (w, v, x, y) \in \mathcal{E}$, **do**
[4] **if** e falls into the range $[t(v_1), t(v_2)]$, **then**
[5] **continue** with loop in line **[2]**

Move edges incident to v along
[6] **for** each edge $e = (v', v, x, y) \in \mathcal{E}, e \neq e_1$, **do**
[7] **if** $v' = v_1$, **then**
[8] Create new edge $e' = (v_1, v_2, x + s', y)$
[9] **else**
[10] Create new edge $e' = (v', v_1, x \Leftrightarrow s, y)$

Move edges incident from v along
[11] **for** each edge $e = (v, v', x, y) \in \mathcal{E}, e \neq e_1$, **do**
[12] **if** $v' = v_2$, **then**
[13] Create new edge $e' = (v_1, v_2, x + s, y)$
[14] **else**
[15] Create new edge $e' = (v_2, v', x \Leftrightarrow s', y)$

[16] Delete v and all incident edges
end

Algorithmus 10: Entfernung zusammenhängender Stille

begin
[1] **for** each vertex $v \in \mathcal{V}$, taken in topological order, **do**
[2] **for** each edge $e = (v, w, s, < SIL >)$, **do**
Copy edges from the endvertex
[3] **for** each edge $e' = (w, x, t, l)$, **do**
[4] **if** $\exists e'' = (v, x, t', l)$, **then**
[5] $t' \leftrightarrow min(t + s, t')$
[6] **else**
[7] Create a new edge $e'' = (v, x, t + s, l)$
[8] Delete e
end

Algorithmus 11: Entfernung aller Stille-Kanten

Operationen (z.B. das Reduzieren auf eindeutige Bezeichnungssequenzen) eine hohe Wahrscheinlichkeit dafür besteht, die Anzahl der Pfade zu verringern.

2.5.4 Verschmelzung gegenseitig nicht erreichbarer Knoten

Die Auswahl von Bewertungskriterien für Worterkenner treibt manchmal seltsame Blüten. Im Zuge der Erkennungs-Evaluation innerhalb des Verbmobil-Projektes 1996 wurde als Kriterium für die Größe von Wortgraphen die Dichte, also die Anzahl der Kanten pro Wort in der Referenzäußerung gewählt. Innerhalb einiger so definierter Klassen wurde dann die Wortakkuratheit als Gütemaß angenommen.

Das Ziel der Graphenverarbeitung war mithin, eine möglichst hohe Anzahl von Pfaden mit einer möglichst geringen Anzahl von Kanten zu erreichen. Eine Methode hierzu ist, Knoten zu verschmelzen, wann immer dies machbar ist. Die Verschmelzung kann die Anzahl der Kanten reduzieren (falls durch sie Kanten mit derselben Bezeichnung auftreten), der Haupteffekt wird jedoch durch die Erhöhung der Anzahl der Pfade erreicht. Bereits aus rein statistischen Erwägungen muß hier die Wortakkuratheit steigen. Neben den früher erwähnten Algorithmen zur Reduzierung eines Graphen durch Verschmelzung bzw. Beseitigung von Stille bietet es sich an, gegenseitig nicht erreichbare Knoten zu verschmelzen. Dadurch wird die Topologie des Graphen nicht gestört, die Kantenzahl reduziert und die Pfadanzahl drastisch erhöht. Im Zuge der erwähnten Evaluation konnte mit diesem Verfahren eine Verbesserung der Worterkennungsrate um ca. 1,5% erreicht werden. Dabei kann natürlich der Fall auftreten, daß zwei Knoten, die zu völlig unterschiedlichen Bereichen des Signals gehören, verschmolzen werden, Auf die Realität der gesprochenen Äußerung wird mithin keine Rücksicht genommen.

2.6 Hypergraphen

Das größte Problem bei der Bearbeitung von Wortgraphen in sprachverarbeitenden Systemen ist natürlich deren Größe (unabhängig von der konkreten Wahl des Größenmaßes *sind* Wortgraphen, insbesondere inkrementelle Wortgraphen, extrem umfangreich und entsprechend aufwendig zu behandeln). Eine wesentliche Ursache hierfür ist das Vorhandensein vieler annähernd identischer Worthypothesen. Mit "annähernd" identisch ist hierbei gemeint, daß sich zwei Kanten bei gleicher Bezeichnung in ihren Anfangs- und Endknoten nur unwesentlich unterscheiden. Abbildung 2.8 zeigt einen (fiktiv kleinen) Ausschnitt aus einem Wortgraphen, der etliche Worthypothesen zu den Wörtern und und dann enthält. Die Start- und Endknoten unterscheiden sich lediglich um einige hundertstel Sekunden.

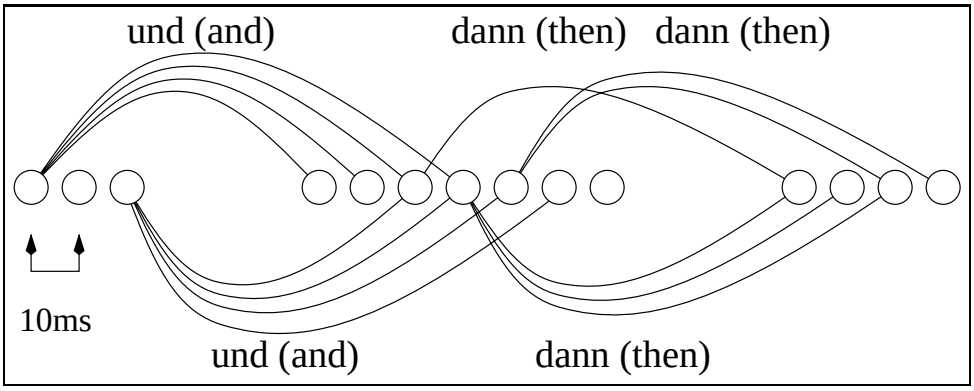


Abbildung 2.8: Zwei Familien von Kanten in einem Wortgraphen

Die Existenz derartiger *Familien* von Kanten (Weber, 1995) resultiert aus mindestens zwei Gründen:

- Worterkenner, die mit Hidden Markov Modellen arbeiten — wie der Hamburger Erkenner (Huebener, Jost, und Heine, 1996), mit dem die in dieser Arbeit untersuchten Wortgraphen erstellt wurden — versuchen zu Beginn jedes Zeitfensters, Wortmodelle zu starten bzw. zu beenden. Da die zeitliche Auflösung sehr fein ist (üblicherweise werden *Frames* mit 10ms Abstand verwendet), resultiert dies in vielen Fällen in einer ganzen Schar von Kanten in aufeinanderfolgenden Knoten.
- Gesprochene Sprache tendiert dazu, Wortgrenzen zu verschmieren. Dieser Effekt ist zum Teil für den dramatischen Abfall in der Worterkennungsrate verantwortlich, der beim Übergang von kontrollierter Rede mit deutlichen Pausen zwischen den einzelnen Wörtern auf kontinuierliche Sprache zu verzeichnen ist. Insbesondere gilt dies für spontan gesprochene Sprache. Abb. 2.8 demonstriert die Ungenauigkeit durch die Existenz zahlreicher Verbindungsknoten zwischen den Hypothesen und und dann.

Folglich werden von einem Spracherkenner im Regelfall Scharen von Worthypothesen ausgegeben, die sich in mehreren Knoten berühren. Beide Merkmale sind extrem hinderlich für die linguistische Verarbeitung. Eine hohe Anzahl von Worthypothesen resultiert in einer hohen Anzahl von

Lexikonzugriffen und grundlegenden Operationen, z.B. dem Vorschlagen syntaktischer Kategorien während eines *Bottom-Up Parsings.* Viele Berührungspunkte führen zu einer unnötig hohen Anzahl komplexer Operationen, etwa Unifikationen, mit denen die linguistischen Beschreibungen von Kanten kombiniert werden.

Es sind mehrere Ansätze denkbar, dieses Problem anzugehen:

1. Eine offensichtliche Methode ist, die zeitliche Auflösung des Spracherkenners innerhalb der Module zur linguistischen Verarbeitung zu reduzieren (vgl. Weber, 1992). Wann immer eine Worthypothese von ihnen gelesen wird, werden die Start- und Endzeiten auf eine gröbere Auflösung abgebildet; ein Redundanztest wird ausgeführt, um multiple Kopien von Kanten zu vermeiden. Dieser Ansatz ist zwar einfach, er führt allerdings zu zusätzlichen Pfaden durch künstlich eingeführte Berührungspunkte von Kanten, die zuvor nicht existierten. Außerdem ist die Wahl einer Auflösung schwierig, da die Längen der Intervalle, die bei Wortanfängen und -enden notwendig werden, abhängig von unterschiedlichen Wörtern sind.

2. Eine direkte und konsistentere Wahl der Repräsentation ist die Benutzung von Intervallgraphen zur Speicherung von Worthypothesen. Kanten verbinden nicht mehr zwei Knoten, sondern es werden zeitliche Intervalle zur Darstelllung der Anfangs- und Endzeiträume verwendet. Abbildung 2.9 zeigt einen derartigen Graphen. Das Hauptproblem bei der Bearbeitung von Intervallgraphen besteht in der Komplexität des Kantenzugriffs. Allerdings sind einige der Eigenschaften, die wir im folgenden verwenden, einfacher und ohne Verlust der Allgemeinheit mit Hilfe von Intervallarithmetik zu zeigen.

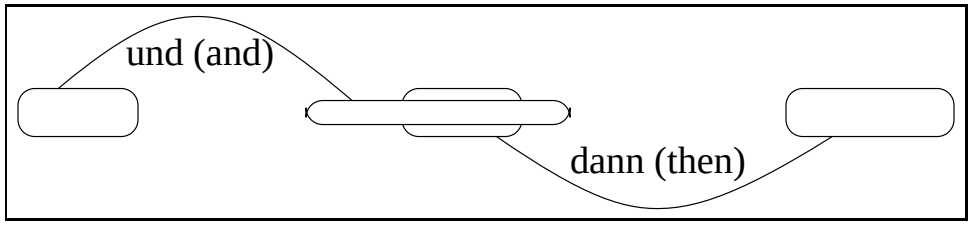


Abbildung 2.9: Ein Intervall-Graph

3. Das Verfahren, das wir für am geeignetsten halten und im Rahmen dieser Arbeit[18] folglich anwenden, benutzt Hypergraphen, um Worthypothesengraphen und Interpretationsgraphen zu repräsentieren. Es beruht auf der Tatsache, daß Operationen ein einziges Mal durchgeführt werden sollen, und bei Auftreten von nahezu identischen Hypothesen schlicht vermerkt wird, daß eine Kante mehrere Start- und Endknoten haben kann. Weber (1995) führt die Bezeichnung *Familie von Kanten* ein für eine Schar von Kanten mit unterschiedlichen Endknoten, aber identischem Startknoten. Im Rahmen dieser Arbeit wurde sein Verfahren auf Kanten mit unterschiedlichen Startknoten erweitert, was eine Verbesserung der Kantenreduktion um

[18] Die Formalisierung von Hypergraphen und deren algorithmische Behandlung sind zusammen mit Volker Weber entstanden, vgl. (Weber, forthcoming).

6% erbrachte (Amtrup und Weber, 1998). Abbildung 2.10 zeigt den bereits bekannten Graphenausschnitt als Hypergraphen. Zur Verarbeitung adaptieren wir das Verfahren von Weber (1995) zur Modifikation von akustischen Bewertungen von Kanten.

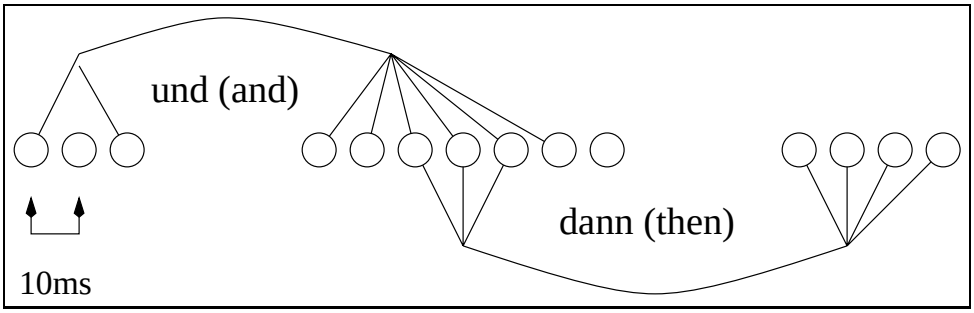


Abbildung 2.10: Zwei Familien von Worthypothesen als Hyperkanten

2.6.1 Formale Definition von Hypergraphen

Hypergraphen (vgl. Gondran und Minoux, 1984, S. 30ff) entstehen aus ungerichteten Graphen dadurch, daß als Kanten Mengen von Knoten angesehen werden. Wir benutzen im folgenden gerichtete Hypergraphen (Gondran und Minoux, 1984, S. 33), verwenden aber die laxe Sprechweise Hypergraph als Synonym hierzu. Wir beschreiben zunächst Hypergraphen als Erweiterung von Wortgraphen; allerdings läßt sich die Hypergraphen-Eigenschaft ohne weiteres auf linksverbundene Wortgraphen und Charts erweitern.

Definition 2.18 *Hypergraph*
Ein Hypergraph ist ein 4-Tupel $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{W})$ *mit*

- *einer Menge* $\mathcal{V}$ *von Knoten wie in Definition 2.13,*
- *einer Menge* $\mathcal{L}$ *von Kantenbezeichnungen wie in Definition 2.13,*
- *einer Menge* $\mathcal{W}$ *von Kantengewichten wie in Definition 2.13,*
- *sowie einer Menge* $\mathcal{E}$ *von Hyperkanten mit*

$$\mathcal{E} \subseteq \mathcal{V}^* \backslash \emptyset \times \mathcal{V}^* \backslash \emptyset \times \mathcal{W} \times \mathcal{L} \tag{2.22}$$

Einige Bezeichnungen und Funktionen müssen für die Benutzung von Hypergraphen angepaßt bzw. erweitert werden. Sei dazu $e = (V, V', w, l)$ *eine Hyperkante.*

- *Die Zugriffsfunktionen für Start- und Endknoten von Hyperkanten geben Mengen von Knoten zurück:*

$$\alpha : \mathcal{E} \Leftrightarrow\!\!\rightarrow \mathcal{V}^*, \alpha(e) := V \tag{2.23}$$
$$\beta : \mathcal{E} \Leftrightarrow\!\!\rightarrow \mathcal{V}^*, \beta(e) := V' \tag{2.24}$$

- *Zwei Hyperkanten e und e' sind adjazent, wenn sie mindestens einen End- bzw. Startknoten gemeinsam haben, d.h., wenn gilt:*

$$\beta(e) \cap \alpha(e') \neq \emptyset \tag{2.25}$$

- *Die Erreichbarkeitsrelation wird zu*

$$\forall v, w \in \mathcal{V} : v \rightarrow w \Leftrightarrow \exists e \in \mathcal{E} : v \in \alpha(e) \wedge w \in \beta(e) \tag{2.26}$$

Zusätzlich definieren wir Zugriffsfunktionen für die zeitlich extrem liegenden Start- und Endknoten, sowie für die von ihnen abgedeckten Zeitintervalle:[19]

$$\alpha_< : \quad \mathcal{E} \Leftrightarrow\!\!\rightarrow \mathcal{V}, \quad \alpha_<(e) := \textit{arg min}\{t(v)|v \in V\} \tag{2.27}$$
$$\alpha_> : \quad \mathcal{E} \Leftrightarrow\!\!\rightarrow \mathcal{V}, \quad \alpha_>(e) := \textit{arg max}\{t(v)|v \in V\} \tag{2.28}$$
$$\beta_< : \quad \mathcal{E} \Leftrightarrow\!\!\rightarrow \mathcal{V}, \quad \beta_<(e) := \textit{arg min}\{t(v)|v \in V'\} \tag{2.29}$$
$$\beta_> : \quad \mathcal{E} \Leftrightarrow\!\!\rightarrow \mathcal{V}, \quad \beta_>(e) := \textit{arg max}\{t(v)|v \in V'\} \tag{2.30}$$
$$\alpha_{[]}(e) := [t(\alpha_<(e)), t(\alpha_>(e))] \tag{2.31}$$
$$\beta_{[]}(e) := [t(\beta_<(e)), t(\beta_>(e))] \tag{2.32}$$

Die obige Definition von $\alpha_<(e)$ etc. beruht darauf, daß jeder Knoten einem Zeitpunkt zugeordnet ist. Obwohl es prinzipiell nicht notwendig ist, eine totale Ordnung auf der Menge der Knoten einzuführen, geschieht dies implizit durch die Definition von Hypergraphen. Da aber Sprache zeitlich linear geäußert wird, erscheint dies natürlich und nicht als zu große künstliche Einschränkung.

Es erweist sich als sinnvoll, die zu einer Wortkante $e = (v, v', w, l)$ gehörende Hyperkante zu definieren. Wir schreiben dafür $\mathcal{H}e$ und setzen $\mathcal{H}e = (\{v\}, \{v'\}, w, l)$.

Im Gegensatz zu Intervallgraphen wird von Hypergraphen nicht verlangt, daß die Mengen von Start- oder Endknoten zusammenhängend sind, es wird also weder

$$\forall e \in \mathcal{E}, v \in \mathcal{V} \quad : \quad \alpha_<(e) \leq t(v) \leq \alpha_>(e) \Longrightarrow v \in \alpha(e) \quad \text{gefordert, noch} \tag{2.33}$$
$$\forall e \in \mathcal{E}, v \in \mathcal{V} \quad : \quad \beta_<(e) \leq t(v) \leq \beta_>(e) \Longrightarrow v \in \beta(e) \tag{2.34}$$

[19] Die Funktion arg min gibt das zu einem Minimum gehörende Argument zurück, nicht das Minimum selbst. Es gilt z.B. $\arg\min\{x^2|x \in \{2, 3, 5\}\} = 2$, nicht 4.

Falls im folgenden die einzelnen Knoten aus $\alpha(e)$ oder $\beta(e)$ nicht von Belang sind, benutzt die Argumentation implizit Intervallgraphen.

Hypergraphen sollten, um vernünftig verarbeitbar zu sein, ebenso wie Wortgraphen azyklisch sein, also muß mindestens gelten:

$$\forall v \stackrel{*}{\rightarrow} w : v \neq w \tag{2.35}$$

Wir fordern sogar

$$\forall e : t(\alpha_>(e)) < t(\beta_<(e)) \tag{2.36}$$

Dies ist eine schärfere Forderung als 2.35, sie motiviert sich aber aus der zeitlichen Ordnung von Sprache und verhindert, daß eine Hyperkante etwa rückwärts in der Zeit verläuft.

2.6.2 Mischen von Hyperkanten

Eine Worthypothese zu einem Hypergraphen hinzuzufügen, stellt einen Spezialfall des Mischens von Hyperkanten dar, da jede Worthypothese als Hyperkante mit genau einem Start- und Endknoten aufgefaßt werden kann. Hier wird zunächst der allgemeine Fall behandelt. Um zwei Hyperkanten ohne Verlust an linguistischer Information zu mischen, müssen drei Bedingungen erfüllt sein:

- Die lexikalischen Einträge, die zu den Hyperkanten gehören, müssen identisch sein,
- Die Kantengewichte (z.B. akustische Bewertungen) müssen sinnvoll kombiniert werden, und
- Die Mengen von Start- und Endknoten müssen kompatibel und ohne Einführung von Zyklen kombinierbar sein.

Definition 2.19 *Mischung von Hyperkanten*
Seien $e_1, e_2 \in \mathcal{E}$ zwei Hyperkanten eines Hypergraphen G mit $e_1 = (V_1, V_1', w_1, l_1)$ und $e_2 = (V_2, V_2', w_2, l_2)$. Dann können beide kombiniert werden, wenn

$$l(e_1) = l(e_2) \tag{2.37}$$
$$\min(t(\beta_<(e_1)), t(\beta_<(e_2))) > \max(t(\alpha_>(e_1)), t(\alpha_>(e_2))) \tag{2.38}$$

gilt. Aus der Kombination entsteht eine neue Hyperkante $e_3 = (V_3, V_3', w_3, l_3)$ mit den neuen Komponenten

$$l_3 = l_1 \quad (= l_2) \tag{2.39}$$
$$w_3 = \textit{scorejoin}(e_1, e_2) \qquad (s.u.) \tag{2.40}$$
$$V_3 = V_1 \cup V_2 \tag{2.41}$$
$$V_3' = V_1' \cup V_2' \tag{2.42}$$

e_1 und e_2 werden aus G entfernt, e_3 neu eingefügt.

Die zwei angegebenen Bedingungen sind hinreichend, um eine konsistente Mischung von Hyperkanten zu erlauben. (2.37) verhindert (natürlich) die Mischung von Hyperkanten, die zu unterschiedlichen Worthypothesen gehören, (2.38) regelt, welche Hyperkanten kombiniert werden dürfen und verhindert Zyklenbildung. Eine Analyse der auftretenden Fälle zeigt dies. Ohne Beschränkung der Allgemeinheit nehmen wir an, daß $t(\beta_>(e_1)) \leq t(\beta_>(e_2))$, d.h. e_2 hinter e_1 endet.

- $\alpha_{[]}(e_1) \cap \beta_{[]}(e_2) \neq \emptyset \quad \vee \quad \alpha_{[]}(e_2) \cap \beta_{[]}(e_1) \neq \emptyset$
  Die Anfangsknoten der einen Kante überlappen die Endknoten der anderen. Eine Kombination dieser Kanten ergäbe eine Kante e_3, bei der $t(\alpha_>(e_3)) \geq t(\beta_<(e_3))$ gilt. Da dies Zyklen in den Graphen einführen könnte, verhindert (2.38) die Kombination.
- $\alpha_{[]}(e_1) \cap \beta_{[]}(e_2) = \emptyset \quad \wedge \quad \alpha_{[]}(e_2) \cap \beta_{[]}(e_1) = \emptyset$
  Dies ist der komplementäre Fall.
  - $t(\alpha_<(e_2)) \geq t(\beta_>(e_1))$
    Die Kante e_1 liegt vollständig vor e_2. Eine solche Situation kann auftreten, wenn dieselbe Worthypothese mehrfach im Graphen auftritt, etwa weil der Sprecher ein Wort mehrfach verwendet. Derartige Kanten sollten nicht vermischt werden, sie werden durch (2.38) getrennt gehalten.
  - $t(\alpha_<(e_2)) < t(\beta_>(e_1))$
    Dies ist der komplementäre Fall.
    * $t(\alpha_<(e_1)) \geq t(\beta_>(e_2))$
      Dieser Fall tritt wegen der Konsistenz der Kanten und der zusätzlichen Annahme, daß $t(\beta_>(e_1)) \leq t(\beta_>(e_2))$ ist, nicht auf.
    * $t(\alpha_<(e_1)) < t(\beta_>(e_2))$
      Dies ist der komplementäre Fall. Die beiden Hyperkanten liegen so zueinander, daß $t(\alpha_>(e_1)) < t(\beta_<(e_2))$ und $t(\alpha_>(e_2)) < t(\beta_<(e_1))$ gelten, mithin $\forall t_\alpha \in \alpha_{[]}(e_1) \cup \alpha_{[]}(e_2), t_\beta \in \beta_{[]}(e_1) \cup \beta_{[]}(e_2) : t_\alpha < t_\beta$. Dies ist genau die in (2.38) gestellte Bedingung. Beide Kanten können gemischt werden.

Die Frage der Kombination der Gewichte zweier Hyperkanten im Zuge des Mischens kann nur anwendungsabhängig beantwortet werden. Falls man, wie hier sinnvoll, annimmt, daß lediglich Wortgraphen u.Ä. betrachtet werden und im wesentlichen von akustischen Bewertungen ausgeht, dann kann die Funktion scorejoin() wie folgt definiert werden. Das Resultat ist hier, daß jeweils die Bewertung übernommen wird, die den besten (kleinsten) Wert pro Zeit hat. Die Bewertung der Kante ist vom frühesten Startknoten bis zum spätesten Endknoten gemessen.

$$\begin{aligned} \text{scorejoin}(e_1, e_2) \quad := \quad & min\left(\frac{w_1}{t(\beta_>(e_1)) \Leftrightarrow t(\alpha_<(e_1))}, \frac{w_2}{t(\beta_>(e_2)) \Leftrightarrow t(\alpha_<(e_2))}\right) \cdot \\ & \cdot(t(\beta_>(e_n)) \Leftrightarrow t(\alpha_<(e_n))) \end{aligned} \tag{2.43}$$

Das Einfügen von Worthypothesen in einen Hypergraphen stellt, wie bereits erwähnt, einen Spezialfall der beschriebenen Situation dar. Die Aufgabe besteht darin, eine Worthypothese mit einer bestehenden Hyperkante zu mischen. Die relevanten relativen Positionen sind in Abb. 2.11 dargestellt.

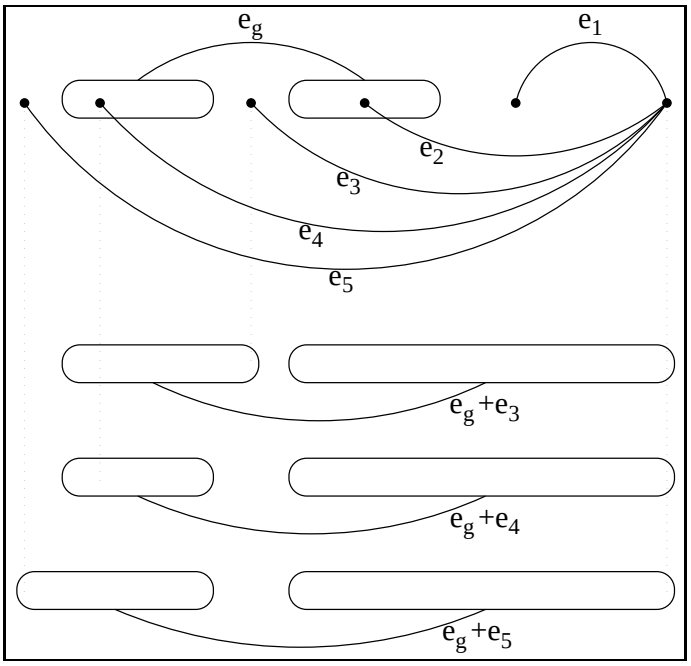


Abbildung 2.11: Hinzufügen von Worthypothesen zu einer Hyperkante

Die bestehende Hyperkante wird durch e_g bezeichnet, e_1 bis e_5 sind Kandidaten, die evtl. hinzugefügt werden. e_1 und e_2 können nicht eingesetzt werden, da sonst Zyklen im Hypergraphen entstehen. Die Resultate des Einfügens von e_3 bis e_5 sind in der Abbildung angegeben. Das Verfahren, mit dem Kanten hinzugefügt werden, wird in Algorithmus 12 gezeigt.

Für jede Hyperkante wird ein einzelnes Kantengewicht benutzt. Ausgehend von den akustischen Bewertungen der Ausgangshypothesen bietet sich nach Weber (1995) die relative akustische Bewertung pro 10ms (*Score* pro *Frame*) als längennormalisiertes Maß an. Wir verwenden die minimale Bewertung pro Frame als Gewicht der Hyperkante.[20] Dadurch werden Pfade durch den Hypergraphen höchstens besser als die korrespondierenden Pfade durch den Wortgraphen, der als Basis der Konstruktion diente.

2.6.3 Kombination von Hyperkanten

Die Kombination von Hyperkanten eines Graphen, z.B. zum Zwecke der syntaktischen Strukturanalyse, verläuft vollständig analog zu entsprechenden Operationen auf normalen Wortgraphen. Zwei

[20] Entsprechend der Darstellung von Bewertungen durch negative Logarithmen.

begin
[1] **if** $\exists e_k \in \mathcal{E}$ **with** $l(e_k) = l(e_n) \wedge t(\beta_<(e_k)) > t(\alpha(e_n))$ **then**
Modify edge e_k
[2] $e'_k := (\alpha(e_k) \cup \{\alpha(e_n)\}, \beta(e_k) \cup \{\beta(e_n)\}, \text{scorejoin}(w(e_k), w(\mathcal{H}e_n)), l(e_k))$
[3] **return** $G' := (\mathcal{V} \cup \{\alpha(e_n), \beta(e_n)\}, \mathcal{E} \backslash e_k \cup \{e'_k\}, \mathcal{W} \backslash w(e_k) \cup \{w(e'_k)\}, \mathcal{L})$
[4] **else**
Add edge $\mathcal{H}e_n$
[5] **return** $G' := (\mathcal{V} \cup \{\alpha(e_n), \beta(e_n)\}, \mathcal{E} \cup \{\mathcal{H}e_n\}, \mathcal{W} \cup \{w(e_n)\}, \mathcal{L} \cup \{l(e_n)\})$
end

Algorithmus 12: Hinzufügen einer Worthypothese e_n zu einem Hypergraphen $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{W})$

Hyperkanten e_1 und e_2 können kombiniert werden, falls sie adjazent sind, also

$$\beta(e_1) \cap \alpha(e_2) \neq \emptyset \tag{2.44}$$

gilt. Der symmetrische Fall ($\beta(e_2) \cap \alpha(e_1) \neq \emptyset$) wird analog behandelt. Zusätzlich sind im Normalfall weitere Kriterien einzuhalten, die durch die zugrundeliegenden Wissensquellen (Grammatik etc.) vorgeschrieben sind. Aus den beiden Kanten wird eine neue Hyperkante e_n konstruiert, welche die Startknoten der einen und die Endknoten der anderen Kante erbt:

$$\alpha(e_n) := \alpha(e_1) \tag{2.45}$$
$$\beta(e_n) := \beta(e_2) \tag{2.46}$$

Aufgrund der Konsistenz der Ausgangskanten und dem nichtleeren Schnitt der End- und Anfangsknoten ist gewährleistet, daß die neue Kante ebenfalls konsistent ist. Als Bezeichnung für e_n werden entweder generische Namen oder durch die Grammatik vorgeschriebene Etiketten verwendet.

Fur die Berechnung des Kantengewichts für e_n, in unserer Anwendung primär die akustische Bewertung, wird — analog zur Mischung von zwei Hyperkanten — der kleinstmögliche Wert verwendet. Da $\beta(e_2) \cap \alpha(e_1)$ mehrere Knoten enthalten kann, die sich auf unterschiedliche Zeitpunkte beziehen, muß das Minimum gesucht werden, das sich durch Kombination der Kantengewichte $w(e_1)$ und $w(e_2)$ ergibt:

$$w(e_n) = \min_{\forall v \in \beta(e_2) \cap \alpha(e_1)} \frac{w(e_1) \cdot (t(v) \Leftrightarrow t(\alpha_<(e_1))) + w(e_2) \cdot (t(\beta_>(e_2)) \Leftrightarrow t(v))}{t(\beta_>(e_2)) \Leftrightarrow t(\alpha_<(e_1))}, \tag{2.47}$$

$w(e_n)$ bezeichnet die akustische Bewertung für den Fall, daß die beiden Kanten e_1 und e_2 hintereinander in der Eingabe vorkommen. Aufgrund von Vererbungsmechanismen muß die Bewertung von e_n aktualisiert werden, falls sich eine der Bewertungen der Ausgangskanten ändert bzw. sich die Knotenmengen der Kanten ändern. Diese Art der Berechnung beücksichtigt bisher nicht die Einbeziehung zusätzlicher Evidenz, etwa durch probabilistische Grammatiken, Sprachmodelle etc.; die Adaption auf derartige Fälle ist allerdings unproblematisch.

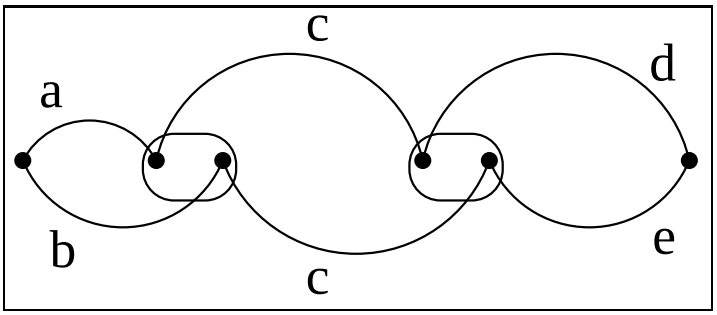


Abbildung 2.12: Die Entstehung zusätzlicher Pfade durch Hypergrapheneinsatz

Die hier vorgestellte Methode der Kombination von Hyperkanten scheint vernünftig, da die Anzahl von Kanten drastisch reduziert werden kann Es ist allerdings möglich, daß durch Ausnutzung von Hypergraphen bei der Kombination von Hyperkanten neue Pfade innerhalb des Graphen entstehen. Abbildung 2.12 zeigt dies. Verarbeitet man den dort gezeigten Graphen als Wortgraphen, so enthält er zwei Sequenzen von Bezeichnungen, nämlich a-c-d und b-c-e. Die Induktion eines Hypergraphen auf den Daten wird eine Hyperkante mit der Bezeichnung c entstehen lassen, die zwei Anfangs- und zwei Endknoten besitzt. Dadurch werden zwei neue Pfade eingeführt, nämlich a-c-e und b-c-d. Der Übergang von Wortgraphen zu Hypergraphen ist folglich nicht informationserhaltend in einem strengen Sinn. Allerdings spielt dies für praktische Anwendungen kaum eine Rolle, da die Situationen, in denen neue Pfade eingeführt werden, selten sind: Wir haben keine wesentlichen Verarbeitungsunterschiede erfahren, bei unseren Experimenten sind lediglich 0,2% zusätzliche Analysen durch die Hypergraphenkonversion eingeführt worden (vgl. Abschnitt 5.1).

2.7 Suchverfahren in Graphen

Das Ziel der Verarbeitung einer natürlichsprachlichen Eingabe besteht prinzipiell darin, eine Interpretation zu finden, die optimal bezüglich eines gewählten Kriteriums ist. Konkretisiert auf die in dieser Arbeit verwendeten Datenstrukturen bedeutet dies, daß ein Weg durch einen Wortgraphen gesucht werden muß, dessen Gewichtssumme optimal ist. Gewöhnlich werden kürzeste Wege gesucht (vgl. z.B. Brandstädt, 1994). Dies entspricht auf Ebene der akustischen Bewertungen von Worthypothesen der allgemein verwendeten Darstellung durch negative Logarithmen von Dichten.

Die konkret zu stellende Frage ist folglich: Welches ist der kürzeste Weg vom Startknoten des Graphen zu seinem Endknoten, und wie hoch ist sein Gewicht? Es zeigt sich, daß die Berechnungen

nicht aufwendiger werden, wenn das *Single source shortest path*-Problem gelöst wird, bei dem für einen ausgezeichneten Startknoten das Gewicht des kürzesten Weges zu jedem anderen Knoten im Graphen berechnet wird. Der allgemeine Fall wird durch den Algorithmus von Bellman-Ford (Cormen, Leiserson, und Rivest, 1990, S. 532) gelöst; beschränkt man sich auf positive Gewichte an Kanten — bei Verwendung des beschriebenen Gewichtsmaßes ist dies hier möglich —, dann ist der bessere Algorithmus von Dijkstra (Cormen, Leiserson, und Rivest, 1990, S. 527) verwendbar, der mit geeigneten Modifikationen eine Komplexität von $O(|V| \log |V| + |E|)$ besitzt.

Die Azyklizität der relevanten Graphen erlaubt sogar die Anwendung eines linearen Algorithmus ($O(|V| + |E|)$) von Lawler (Cormen, Leiserson, und Rivest, 1990, S. 536). Er ist als Algorithmus 13 angegeben.

---

begin

Initialize shortest path estimate and minimal weight

[1] **for** each vertex $v \in V(G)$ **do**

[2] $d[v] \leftarrow\!\!\!\Rightarrow \infty$

[3] $\pi[v] \leftarrow\!\!\!\Rightarrow$ NIL

[4] $d[s] \leftarrow\!\!\!\Rightarrow 0$

Construct shortest paths

[5] **for** each vertex $u \in V(G)$, taken in topological order **do**

[6] **for** each vertex $v \in \text{Adj}(u)$ **do**

Relax the edge

[7] **if** $d[v] > d[u] + w(u, v)$ **then**

[8] $d[v] \leftarrow\!\!\!\Rightarrow d[u] + w(u, v)$

[9] $\pi[v] \leftarrow\!\!\!\Rightarrow u$

end

Algorithmus 13: SSSP für DAGs

Im Rahmen dieser Arbeit findet die Suche nach besten Wegen durch einen Graphen innerhalb der Generierung natürlichsprachlicher Ausdrücke statt. Die Generierung (vgl. Abschnitt 4.9) produziert für erfolgreich transferierte semantische Beschreibungen englische Oberflächenformen. Das primäre Resultat ist ein Hypergraph von Kanten, die mit englischen Äußerungsfragmenten annotiert sind. Innerhalb dieses Graphen ist nun inkrementell der beste Pfad vorzuhalten, um zu jedem Zeitpunkt die gültige Konkatenation ausgeben zu können. Algorithmus 14 zeigt die notwendigen Operationen. Durch die Möglichkeit, eine topologische Ordnung auf den in dieser Arbeit verwendeten Graphen einzuführen, braucht nicht die volle Mächtigkeit von inkrementellen Algorithmen (Cheston, 1976) angewendet zu werden. Insbesondere kann in unserem Zusammenhang keine Löschung von Kanten erfolgen, die besonders behandelt werden müßte (Ramalingam und Reps, 1992). Der hier gezeigte Algorithmus wird für jede in den Graphen neu eingeführte Kante aufgerufen. Er verwaltet zwei Werte für jeden Knoten, nämlich das Gewicht des bisher besten Pfades (b_v), sowie die

Kante, die auf dem besten Pfad zu ihm führt (r_v). Es werden jeweils nur die Knoten betrachtet, die sich rechts von der eingesetzten Kante befinden und deren Gewichtswert für den besten Pfad sich durch das Einsetzen der Kante tatsächlich geändert hat. Der in dem in Abschnitt 4.9 beschriebene Generator benutzt einen modifizierten Algorithmus, da dort ebenfalls das Überspringen von Lücken im Graphen modelliert wird. Zu diesem Zweck werden virtuelle Kanten zwischen je zwei benachbarten Knoten eingesetzt, deren Gewicht einer Übergangsstrafe entspricht.

Update single source shortest information
procedure UpdateSSSP(G, e**)**
Topological order o
Initialization (done only once)
[1] **for** each $v \in \mathcal{V}$, **do**
[2] $b_v \leftarrow\!\!\!\!\!\!\leftrightarrow \infty$
[3] $b_{v^{(r)}} \leftarrow\!\!\!\!\!\!\leftrightarrow 0$

Update immediate end vertices of e
[4] **for** each $v \in \alpha_<(e)$, **do**
[5] **for** each $w \in \alpha_>(e)$, **do**
[6] **if** $b_v + w(e) < b_w$, **then**
[7] $b_w \leftarrow\!\!\!\!\!\!\leftrightarrow b_v + w(e)$
[8] $r_w \leftarrow\!\!\!\!\!\!\leftrightarrow e$
[9] Insert w into o

Update vertices if necessary
[10] **while** $o \neq \emptyset$, **do**
[11] $v \leftarrow\!\!\!\!\!\!\leftrightarrow o.First()$
[12] **for** each e **having** $v \in \alpha_<(e)$, **do**
[13] **for** each $w \in \alpha_>(e)$, **do**
[14] **if** $b_v + w(e) < b_w$, **then**
[15] $b_w \leftarrow\!\!\!\!\!\!\leftrightarrow b_v + w(e)$
[16] $r_w \leftarrow\!\!\!\!\!\!\leftrightarrow e$
[17] Insert w into o
end

Algorithmus 14: SSSP für inkrementelle Hypergraphen

2.8 Zusammenfassung

Wortgraphen bilden augenblicklich die fortgeschrittenste Schnittstelle zwischen akustischer Worterkennung und linguistischer Verarbeitung. Sie haben gegenüber der Repräsentation des Erken-

nungsergebnisses durch Wortsequenzen den großen Vorteil, sehr große Anzahlen von Äußerungshypothesen ($> 10^{30}$) in kompakter Wiese kodieren zu können. Sie sind jedoch nicht völlig frei von Redundanz: Identische relevante Wortsequenzen können auf unterschiedlichen Wegen durch den Graphen erzeugt werden, was die Möglichkeit zur Größenreduzierung (etwa durch die Reduktion auf eindeutige Sequenzen oder durch die Elimination von Stille-Hypothesen) mit sich bringt, die zwar prinzipiell im Laufe der Verarbeitung auch für linksverbundene Wortgraphen möglich wären, jedoch sehr teuer sein können.

Eine außerordentlich effiziente Art der Aufwandsreduktion hingegen stellt der Übergang zu Hypergraphen dar, deren Kanten Mengen von Start- und Endknoten tragen.

Wichtig für den Vergleich verschiedener Sprachverarbeitungssysteme sind schließlich Größen- und Gütemaße für Wortgraphen. Wir sind der Auffassung, daß statt der Dichte, die einerseits abhängig von der Transliteration der Eingabe ist und andererseits die Topologie des Wortgraphen unberücksichtigt läßt, eher ein anwendungsrelevantes Maß für die Größe gewählt werden sollte: Die Anzahl der Operationen, die ein fiktiver, idealisierter Parser zur Verarbeitung des Graphen aufwenden muß. Es lassen sich überdies Alternativen zu dem allgemein üblichen Gütemaß der Wortakkuratheit denken, die Orientierung an der Anzahl der Abweichung von einer Referenz wird aber u.E. noch lange Gültigkeit haben.

Kapitel 3

Unifikationsbasierte Formalismen für die Übersetzung in der maschinellen Sprachverarbeitung

Der Formalismus zur Beschreibung linguistischen Wissens ist ein zentraler Bestandteil jedes natürlichsprachlichen Systems. In diesem Kapitel werden nach einer Einführung in die Unifikation zunächst kurz verschiedene unifikationsbasierte Formalismen und Verfahren beschrieben, die in transferorientierten Systemen zur maschinellen Übersetzung angewendet werden. Danach schließt sich die Präsentation des im vorliegenden System verwendeten Formalismus mit einigen Details zu seiner Implementierung an.

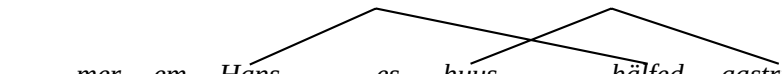
3.1 Unifikationsformalismen in der maschinellen Sprachverarbeitung

Unifikationsbasierte Formalismen erlauben es, deklarativ grammatisches Wissen zu formulieren. Sie verwenden dabei eine knappe, generalisierende Darstellungsweise und sind in der Lage, mit partieller Information umzugehen. Die Entwicklung derartiger Formalismen hat im Gegensatz zu automatenorientierten Ansätzen oder zu nicht primär an der maschinellen Verarbeitung orientierten Formalisierungen (etwa im Bereich der generativen Grammatik, vgl. Chomsky, 1995) zahlreiche Vorteile.

Automaten zur Repräsentation natürlichsprachlichen Wissens sind aus Arbeiten zur Umwandlung von Programmen in Programmiersprachen abgeleitet; dort haben sie sich gut bewährt. Für die Beschreibung natürlicher Sprachen sind vor allem ATNs (*Augmented Transition Networks*) zur Anwendung gekommen (Woods, 1973). Dies sind Transitionsnetzwerke, deren Kanten mit Bedingungen annotiert werden können; zusätzlich werden Register eingeführt, die Werte beliebig lange speichern

und sogar mit Defaults vorbesetzt werden können. Dadurch sind ATNs dazu in der Lage, Typ-0 Sprachen (Chomsky, 1959) zu verarbeiten. Obwohl einige große Systeme mit ATN-Grammatiken versehen sind (Kaplan, 1973), erschwert gerade der Registermechanismus das Verstehen eines konkreten Netzes. Der Ablauf einer syntaktischen Analyse kann im Nachhinein aus den Ergebnissen nicht nachvollzogen werden, da Register nichtmonoton verändert werden können. Zusätzlich erscheint der Charakter der Netzwerkgrammatiken zu prozedural orientiert, die Ebene der Abstraktion zu niedrig.

Eine sehr viel leichter nachvollziehbare, deklarativ orientierte Beschreibungsweise bilden Phrasenstrukturgrammatiken. Schon sehr früh sind kontextfreie Grammatiken (*context free grammar*, CFG), die den rekursiven Transitionsnetzwerken (RTN) äquivalent sind[1], zur syntaktischen Beschreibung eingesetzt worden. Klassische Arbeiten zur effizienten Strukturanalyse verwenden CFGs (z.B. Earley, 1970). Allerdings ist vielfach argumentiert worden, kontextfreie Grammatiken reichten für eine adäquate Beschreibung natürlicher Sprachen nicht aus (vgl. Sampson, 1983 oder Shieber, 1985). Trotzdem kann ein bedeutender Ausschnitt natürlicher Sprachen kontextfrei modelliert werden, so daß die Existenz problematischer Fälle allein nicht ausreichend motiviert, auf CFGs vollständig zu verzichten (Gazdar, 1983). Phänomene wie z.B. *cross-serial dependencies* im Holländischen in **(3.1)** können durch eine geringfügige Erweiterung der Verarbeitungsmechanismen kontextfreier Grammatiken behandelt werden (Vogel, Hahn, und Branigan, 1996). Die Erweiterung ist erforderlich, da prinzipiell beliebig viele Paare von zusammengehörigen Konstituenten paarweise ineinander verschränkt sein können und zusätzlich Kongruenz aufweisen.



…	*mer*	*em*	*Hans*	*es*	*huus*	*hälfed*	*aastriiche*	**(3.1)**
…	we		Hans-DAT	the	house-ACC	helped	paint	

…we helped Hans paint the house.

Die Ursache für die weite Verbreitung unifikationsbasierter Formalismen und deren Verwendung zur Modellierung syntaktischer Strukturen ist denn auch nicht ausschließlich in ihrer theoretischen Notwendigkeit zu suchen. Vielmehr ergeben sich mit Hilfe der Unifikation bedeutende praktische Vorteile bei der Erstellung von Grammatiken großer Abdeckung. Insbesondere im Vergleich zu kontextfreien Grammatiken lassen sich knappe, generalisierende Regeln schreiben. Für die Kongruenz zwischen Subjekt und Verb im Deutschen etwa reicht eine Regel aus, die fordert, daß gewisse Merkmale der Beschreibung der beiden Konstituenten unifizierbar sind; eine kontextfreie Darstellung benötigt für jede mögliche Kombination von Merkmalen eine gesonderte Regel.

Darüberhinaus erlauben Merkmalstrukturen die Formulierung partieller Information, was kontextfrei nur schwer gelingt. Dies ist nützlich — erneut im Falle der Kongruenz —, falls z.B. die morphologische Markierung des Genus ambig ist.

Viele unifikationsbasierte Formalismen erlauben die Darstellung von Einschränkungen über Merk-

[1] D.h., für jede kontextfreie Grammatik gibt es ein rekursives Transitionsnetzwerk, das dieselbe Sprache akzeptiert.

malstrukturen (*Constraints*), die im Laufe der Verarbeitung garantieren, daß die Beschreibungen linguistischer Objekte konsistent bleiben. Dieser constraintbasierte Ansatz ist extrem mächtig und führt im Extremfall zu einer sehr kleinen Menge an Prinzipien und allgemeinen Schemata (z.B. bei HPSG, vgl. Pollard und Sag, 1987;Pollard und Sag, 1994).

In der vorliegenden Arbeit kommt ein Merkmalformalismus zum Einsatz, der jeder einzelnen Merkmalstruktur einen Typ zuweist. Dadurch kann der Raum der potentiellen Beschreibungen weitergehend strukturiert werden. Durch Einführung von Vererbung innerhalb der hierarchisch aufgebauten Typenmenge wird die Beschreibung erneut kompakter. Zusätzlich gewinnt man Effizienz bei der Unifikation von Merkmalstrukturen, da nur solche mit "kompatiblen" Typen miteinander unifizierbar sind. Diese Operation ist durch eine günstige Repräsentation der Typenhierarchie äußerst einfach und effizient.

Der Ursprung der Unifikation liegt in den Arbeiten von Herbrand, die moderne Sicht beginnt mit Guard (1964) und Robinson (1965). Der Unifikationsalgorithmus von Robinson (1965) besitzt eine exponentielle Komplexität und behandelt das Problem der Unifikation zweier Terme der Prädikatenlogik erster Stufe. Das Resultat ist eine Substitution für Variablen, so daß die beiden zu unifizierenden Terme durch Einsetzung der Variablenwerte identisch werden. Knight (1989) stellt die Vorteile des Übergangs zur Graphenunifikation dar, die auf feste Stelligkeit und Anonymität von Unterstrukturen verzichtet und mit dem Mittel der Koreferenz eine elegantere Beschreibung von Identitäten zuläßt. Der verbreitetste Unifikationsalgorithmus ist der von Huet (vgl. Knight, 1989, S. 98), der fast linear ist[2]. Es gibt inzwischen zwar lineare Algorithmen zur Graphenunifikation (Paterson und Wegman, 1978); diese erfordern allerdings einen hohen Verwaltungsaufwand und sind in fast allen praktischen Fällen weniger effizient (vgl. Amtrup, 1992, S. 67). Interessant im Zusammenhang mit dieser Arbeit sind parallele Implementierungen von Unifikationsformalismen wie z.B. Vitter und Simons (1986) oder Hager und Moser (1989). Nach Knight (1989) ist die Unifikation allerdings ein hochgradig sequentieller Prozeß, der nur durch Einführung bedeutender Ressourcen im parallelen Fall merkbar beschleunigt wird. Besonders kompliziert wird die Durchführung der Unifikation, wenn disjunktive Terme betrachtet werden (Eisele und Dörre, 1988). Das Problem liegt in der möglichen Interaktion zwischen disjunktiven Merkmaltermen und Koreferenzen, die nicht auf disjunktive Terme beschränkt sind (Kaplan und Maxwell, 1989). Eine Lösung hierzu verwendet benannte Disjunktionen, die stets auf dieselben ursprünglich verwendeten Teilterme referieren (Backofen, Euler, und Görz, 1991).

Die erste Verwendung von unifikationsbasierten Formalismen innerhalb der Sprachverarbeitung wurde von Kay (1979) getätigt. Seitdem sind zahlreiche hochspezialisierte Theorien zur linguistischen Beschreibung entstanden, z.B. die *Lexical Functional Grammar* (LFG, Bresnan, 1982), *Head Driven Phrase Structure Grammar* (HPSG, Pollard und Sag, 1987l; Pollard und Sag, 1994). Daneben existieren stärker systemorientierte Formalismen und Werkzeuge wie die *Definite Clause Grammars* in Prolog (DCG, Pereira und Shieber, 1984), PATR II (Shieber, 1984) oder *Unification Tree Adjoining Grammars* (UTAG, Joshi, 1985; Harbusch, 1990).

[2] Er weist eine Zeitkomplexität von $O(n \cdot \alpha(n))$ auf, wobei $\alpha(n)$ die Umkehrfunktion der Ackermannschen Funktion ist.

Die bisher angesprochenen Formalismen sind deklarativ in zweierlei Hinsicht: zum einen beschreiben sie linguistische Sachverhalte, ohne auf die Mechanismen zur Lösung einer bestimmten Aufgabe einzugehen. Zum anderen sind Merkmalstrukturen selbst Datenobjekte, deren Behandlung extern definiert wird. Carpenter und Qu (1995) legen jedoch dar, daß Merkmalstrukturen ebenfalls als abstrakte Maschine interpretierbar sind. Mit Wintner und Francez (1995a) und Wintner (1997) liegt sogar eine vollständige Implementierung eines Merkmalformalismus unter dieser Prämisse vor. Auch die vorliegende Implementation orientiert sich stark an dieser Anschauung.

3.1.1 Definition von getypten Merkmalstrukturen mit Appropriateness

In diesem Abschnitt werden die Merkmalstrukturen, die in der vorliegenden Arbeit verwendet werden, definiert. Die Definition ist Grundlage der Implementation des Merkmalformalismus, die in Abschnitt 3.3 beschrieben ist. Das Ziel ist, Merkmalstrukturen wie in Abb. 3.1[3] formal zu definieren. Die Darstellung ist angelehnt an Carpenter (1992).

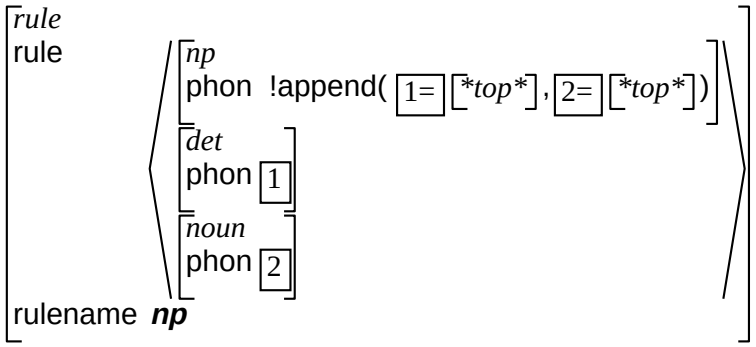


Abbildung 3.1: Merkmalstruktur einer einfachen syntaktischen Regel

Typenverbände

Jede Merkmalstruktur besitzt einen Typ. Der Raum von möglichen Merkmalstrukturen wird dadurch in gewünschte Klassen eingeteilt. Die Menge von Typen ist hierarchisch geordnet, um weitergehende Strukturierungsmöglichkeiten bereitzustellen und einen Vererbungsmechanismus zu ermöglichen. Durch die Hinzunahme eines generellsten Typs *top* ($\top$)[4] und des inkonsistenten Typs *bottom* ($\bot$) wird die Hierarchie zum Verband von Typen.

[3] *Kursiv* gesetzte Namen bezeichnen hier Typen (**top** bezeichnet $\top$, s.u.), serifenlos gesetzte Namen bezeichnen Merkmale, ***fette, geneigte Serifen*** denotieren Zeichenketten.

[4] Die Formulierungen in dieser Arbeit arbeiten mit einer modelltheoretischen Sicht. $\top$ ist der generellste Typ, da er alle Extensionen enthält.

Die Ordnungsrelation innerhalb des Verbandes heißt Subsumption. Ein Typ σ subsumiert einen Typ τ (geschrieben $\sigma \sqsupseteq \tau$), falls er allgemeiner ist, d.h. näher an $\top$ liegt. Die größte untere Schranke von zwei Typen bezüglich der Subsumption ist das Ergebnis der *Unifikation* zweier Typen.

Definition 3.1 (Typenverband)
Sei $D = (T, \sqsupseteq)$ eine Halbordnung, wobei T eine Menge von Typensymbolen bezeichne. $\sqsupseteq$ bezeichne die Ordnungsrelation auf T, d.h. $\sqsupseteq$ ist reflexiv, antisymmetrisch und transitiv.

Sei $D \subseteq T$ eine Teilmenge von Typen. Ein Element $s \in T$ heißt kleinste obere Schranke (Supremum) von D, geschrieben $\sqcup D$, falls gilt:

$$\bigwedge_{x \in D} \quad s \sqsupseteq x \tag{3.1}$$

$$\bigwedge_{s\prime \in T} \quad (\forall x \in D : s' \sqsupseteq x) \Longrightarrow s' = s \tag{3.2}$$

Ein Element $i \in T$ heißt größte untere Schranke (Infimum) von D, geschrieben $\sqcap D$, falls gilt:

$$\bigwedge_{x \in D} \quad x \sqsupseteq i \tag{3.3}$$

$$\bigwedge_{i\prime \in T} \quad (\forall x \in D : x \sqsupseteq i') \Longrightarrow i' = i \tag{3.4}$$

Die Erweiterung $D\prime = (T \cup \{\top := \sqcup T, \bot := \sqcap\emptyset\}, \sqsupseteq)$ bildet einen Verband, den Typenverband der Typen aus T. $\sqsubseteq$ *heißt* Subsumptionsrelation*, das Auffinden des Infimums $\sqcap D$ einer Teilmenge von Typen heißt* Typunifikation.

Merkmalstrukturen

Merkmalstrukturen werden nun definiert als gerichtete Graphen mit Knoten- und Kantenbezeichnungen und einer ausgezeichneten Wurzel. Die Knoten sind mit Typsymbolen markiert, die Kanten mit Namen von Merkmalen. Zyklen innerhalb von Merkmalstrukturen sind ausdrücklich zugelassen.

Definition 3.2 (Merkmalstruktur, Pfade)
Sei $D = (T, \sqsupseteq)$ ein Typenverband, $Feat$ eine endliche Menge von Merkmalnamen, $Feat \neq \emptyset$.

Eine Merkmalstruktur *(feature structure, FS) ist ein Viertupel $F = \langle Q, \overline{q}, \theta, \delta \rangle$. Dabei gilt:*

- *Q ist eine endliche Menge von Knoten, $Q \neq \emptyset$.*
- *$\overline{q} \in Q$ ist die Wurzel der Merkmalstruktur.*
- *$\theta : Q \leftrightarrow\!\!\!\rightarrow T$ ist eine totale Funktion, die Knoten auf Typen abbildet.*
- *$\delta : Q \times Feat \leftrightarrow\!\!\!\rightarrow Q$ ist eine partielle Funktion, die Merkmalwerte bestimmt.*

Die Definition von δ wird auf die Möglichkeit, mehr als ein Merkmal zur Zeit zu traversieren, erweitert. Sei $Path := Feat^*$ die Menge der Pfade über Merkmalen, ϵ der leere Pfad ohne Merkmale. Dann definiert man zusätzlich

$$\delta(q, \epsilon) := q, \textit{ und} \tag{3.5}$$

$$\delta(q, f\pi) := \delta(\delta(q, f), \pi), \textit{ wobei } q \in Q, f \in Feat, \pi \in Path \tag{3.6}$$

Die Funktion δ bildet die Kanten des Graphen, der die Merkmalstruktur repräsentiert. Die Knoten in Q müssen jeweils von $\overline{q}$ aus durch eine wiederholte Anwendung von δ erreichbar sein, d.h. die Wurzel des Graphen muß eindeutig bestimmt sein.

Es kann sein, daß δ nicht injektiv ist, d.h. ein Knoten des die Merkmalstruktur repräsentierenden Graphen besitzt zwei eingehende Kanten ($\exists q1, q2 \in Q \exists f1, f2 \in Feat : (q1, f1) \neq (q2, f2) \wedge \delta(q1, f1) = \delta(q2, f2)$). In diesem Fall spricht man von der *Koreferenz* zweier Merkmalswerte, die in graphischer Form durch koindizierte Kästen notiert wird (s. Abb. 3.1). In Logikprogrammiersprachen entspricht dies der Verwendung identischer Variablen. Zyklen innerhalb von Merkmalstrukturen existieren, wenn einer der koreferierenden Knoten vom anderen aus erreichbar ist ($\exists \pi \in Path : \delta(q1, \pi) = q2$).

Analog zur Typsubsumption kann man nun die Subsumption über Merkmalstrukturen definieren. Eine Merkmalstruktur F subsumiert eine Merkmalstruktur F', wenn F allgemeiner ist als F', dabei aber alle strukturellen Eigenschaften von F auch in F' auftreten. Die Definition ermöglicht im Falle der Repräsentation partieller Information eine Abschätzung darüber, welche von zwei Merkmalstrukturen mehr Information über ein bestimmtes linguistisches Objekt enthält.

Definition 3.3 (Subsumption von Merkmalstrukturen)
Seien $F = \langle Q, \overline{q}, \theta, \delta \rangle$ und $F\prime = \langle Q\prime, \overline{q}\prime, \theta\prime, \delta\prime \rangle$ zwei Merkmalstrukturen über dem Typenverband $D = (T, \sqsupseteq)$ und der Merkmalsmenge $Feat$. Dann subsumiert *F die Merkmalstruktur $F\prime$, $F \sqsupseteq F\prime$, falls es eine Abbildung $h : Q \leftrightarrow\!\!\!\rightarrow Q\prime$ gibt, die folgende Bedingungen erfüllt:*

- *Die Wurzel von F wird auf die Wurzel von $F\prime$ abgebildet:*

$$h(\overline{q}) = \overline{q}\prime \tag{3.7}$$

- *Der Typ jedes Knotens in F subsumiert den korrespondierenden Typ in $F\prime$:*

$$\bigwedge_{q \in Q} \theta(q) \sqsupseteq \theta\prime(h(q)) \tag{3.8}$$

- *Die Kanten von F treten ebenfalls in $F\prime$ auf:*[5]

$$\bigwedge_{q \in Q} \bigwedge_{f \in F} \delta(q,f)_{def.} \Longrightarrow h(\delta(q,f)) = \delta\prime(h(q),f) \tag{3.9}$$

Ebenfalls analog zur Unifikation von Typen wird die Unifikation von Merkmalstrukturen definiert. Die Unifikation ist die kleinste Merkmalstruktur, die von beiden Ausgangs-Strukturen subsumiert wird. Dies ist dadurch möglich, daß mit der Subsumption eine Ordnungsrelation über Merkmalstrukturen eingeführt wurde. Die Durchführung der Unifikation beginnt bei den jeweiligen Wurzelknoten der beiden Operanden. Der Wurzelknoten des Resultats erhält als Typ die Unifikation der Typen der Ausgangsknoten. Danach werden repetierend Knoten identifiziert, die als Resultat identischer Merkmale auftreten, und ein Resultatknoten erzeugt, erneut vom Typ der Unifikation der Ausgangstypen. Die Unifikation endet, wenn alle Knoten betrachtet wurden und scheitert, falls an einer Stelle die Unifikation zweier Typen fehlschlägt (d.h. $\perp$ liefert). Formal benutzt die Definition in Carpenter (1992) Äquivalenzklassen.

Definition 3.4 (Äquivalenzklasse, Quotientenmenge)
Sei $\equiv$ eine Äquivalenzrelation, also eine transitive, reflexive und symmetrische Relation über einer Menge X. Dann heißt

$$[x]_{\equiv} = \{y \in X \mid y \equiv x\} \tag{3.10}$$

die Äquivalenzklasse von x bezüglich $\equiv$.

Die Menge

$$X/_{\equiv} = \{[x]_{\equiv} \mid x \in X\} \tag{3.11}$$

heißt Quotientenmenge von X bezüglich $\equiv$.

Die Definition der Unifikation zweier Merkmalstrukturen definiert eine Äquivalenzrelation, die genau die oben geschilderten Bedingungen erfüllt.

[5] Wir schreiben abkürzend $R(x,y)_{def.}$, falls $R(x,y)$ definiert ist.

Definition 3.5 (Unifikation von Merkmalstrukturen)
Seien $F = \langle Q, \overline{q}, \theta, \delta \rangle$ *und* $F\prime = \langle Q\prime, \overline{q}\prime, \theta\prime, \delta\prime \rangle$ *zwei Merkmalstrukturen über dem Typenverband* $D = (T, \sqsupseteq)$ *und der Merkmalmenge* $Feat$. *Sei außerdem* $Q \cap Q\prime = \emptyset$.[6] *Die Äquivalenzrelation* $\bowtie$ *sei die kleinste Äquivalenzrelation, so daß*

$$\overline{q} \bowtie \overline{q}\prime, \textit{ und} \tag{3.12}$$

$$\delta(q, f) \bowtie \delta\prime(q\prime, f), \textit{ falls beide definiert sind und } q \bowtie q\prime \tag{3.13}$$

Die Unifikation von F und $F\prime$ *ist dann definiert als*

$$F \sqcap F\prime = \langle (Q \cup Q\prime)/_{\bowtie}, [q\prime]_{\bowtie}, \theta^{\bowtie}, \delta^{\bowtie} \rangle, \tag{3.14}$$

wobei

$$\theta^{\bowtie}([q]_{\bowtie}) = \sqcap\{(\theta \cup \theta\prime)(q\prime) \mid q\prime \bowtie q\} \tag{3.15}$$

und

$$\delta^{\bowtie}([q]_{\bowtie}, f) = \begin{cases} [(\delta \cup \delta\prime)(q, f)]_{\bowtie} & \textit{, falls } (\delta \cup \delta\prime)(q, f) \textit{ definiert ist} \\ \textit{undefiniert} & \textit{, sonst} \end{cases} \tag{3.16}$$

falls alle Typunifikationen in $\theta^{\bowtie}$ *existieren.* $F \sqcap F\prime$ *ist ansonsten undefiniert.*

Die Unifikation zweier Merkmalstrukturen ist wiederum eine Merkmalstruktur, falls sie definiert ist (vgl. Carpenter, 1992, S. 47). Insbesondere kann gezeigt werden, daß die Unifikation Information aus beiden Ausgangsstrukturen konjunktiv zusammensetzt. Merkmale, die nur in einer der beiden enthalten sind, werden durch die Reflexivität von $\bowtie$ in die Resultat-Struktur übernommen.

Appropriateness

Die bisherige Definition von getypten Merkmalstrukturen macht zwar während der Unifikation Gebrauch von der Typenhierarchie, sie schränkt jedoch nicht die Verwendung von Merkmalen an Knoten bestimmten Typs ein. Dies ist jedoch genau eine der Eigenschaften, die in verschiedenen Formalismen (z.B. in HPSG, s. Pollard und Sag, 1987) gefordert wird, um nicht zulässige von nicht

[6] Die Unifikation bildet die neue Merkmalstruktur durch Vereinigung der Knotenmengen unter bestimmten Umständen. Daher wird ein leerer Schnitt verlangt, der durch alphabetische Varianten immer erreicht werden kann (vgl. Carpenter, 1992, S. 43).

besetzten Merkmalen zu unterscheiden. Darüberhinaus erlaubt eine Einschränkung der Menge von zulässigen Merkmalen für einen Typ eine Repräsentation von Knoten in Merkmalstrukturen, die eine feste Länge besitzt. Dies wird in der unten beschriebenen Implementation des in dieser Arbeit verwendeten Merkmalformalismus relevant sein.

Carpenter (1992) definiert eine Zulässigkeitsfunktion (*appropriateness specification*) als eine partielle Funktion, die angibt, welche Merkmale für einen Typ zulässig sind und welchen Typ die Werte dieser Merkmale haben müssen.

Definition 3.6 (Appropriateness)
Sei $D = (T, \sqsupseteq)$ ein Typenverband und $Feat$ eine endliche, nichtleere Menge von Merkmalen. Dann heiß eine partielle Funktion $Approp : T \times Feat \leftrightarrows T$ Zulässigkeitsfunktion*, wenn die folgenden Bedingungen erfüllt sind:*

- *Jedes Merkmal wird an genau einer Stelle im Typenverband eingeführt:*
  Für jedes Merkmal $f \in Feat$ gibt es einen allgemeinsten Typ $Intro(f) \in T$, so daß $Approp(Intro(f), f)$ definiert ist
- *Die Zulässigkeit ist nach unten im Typenverband abgeschlossen:*
  Falls $Approp(\sigma, f)$ definiert ist und $\sigma \sqsupseteq \tau$, dann ist auch $Approp(\tau, f)$ definiert und $Approp(\sigma, f) \sqsupseteq Approp(\tau, f)$.

In der Anwendung eines Formalismus mit Appropriateness werden nur diejenigen Merkmalstrukturen betrachtet, die bezüglich der Einschränkungen, welche die Appropriateness erzwingt, wohlgeformt sind. Dies wird durch die Beschränkung auf wohlgetypte Merkmalstrukturen erreicht.

Definition 3.7 (Wohlgetypte Merkmalstrukturen)
Eine Merkmalstruktur $F = \langle Q, \overline{q}, \theta, \delta \rangle$ heißt wohlgetypt*, falls*

$$\bigvee_{q \in Q} \bigvee_{f \in Feat} \delta(q,f)_{def.} \leftrightarrows Approp(\theta(q), f)_{def.} \wedge Approp(\theta(q), f) \sqsupseteq \theta(\delta(q,f)). \quad (3.17)$$

In der vorliegenden Arbeit werden ausschließlich wohlgetypte Merkmalstrukturen verwendet. Allerdings sind die Einschränkungen durch die Appropriateness schwächer als in Def. 3.6 gefordert. Auf die Eindeutigkeit des einführenden Typs $Intro(f)$ wird verzichtet. Das verhindert zwar eine Typinferenz und damit eine Klassifikation von Merkmalstrukturen gemäß des Typenverbandes, diese Operationen werden allerdings hier nicht benötigt. Eine weitergehende Wohlgeformtheitseigenschaft birgt die Definiton von vollständig wohlgetypten Merkmalstrukturen. Neben der Wohlgetyptheit wird hier zusätzlich gefordert, daß jedes zugelassene Merkmal einer Merkmalstruktur auch belegt ist.

Definition 3.8 (Vollständig wohlgetypte Merkmalstrukturen)
Eine Merkmalstruktur $F = \langle Q, \overline{q}, \theta, \delta \rangle$ *heißt* vollständig wohlgetypt, *falls sie wohlgetypt ist und*

$$\bigvee_{q \in Q} \bigvee_{f \in Feat} Approp(q, f)_{def.} \Leftrightarrow\!\!\!\rightarrow \delta(q, f)_{def.} \tag{3.18}$$

gilt.

Funktionen als Werte von Merkmalen

In einigen Formalismen wird die Möglichkeit zur Formulierung von Funktionen und Relationen über Merkmalstrukturen angeboten. Emele und Zajac (1990) beispielsweise benutzen die Annotation von Bedingungen an Typdefinitionen, um rekursive Funktionen zu formulieren. Die HPSG (Pollard und Sag, 1994) definiert Relationen innerhalb von Merkmalstrukturen, die stets erfüllt bleiben. Dort dienen sie unter anderem dazu, Listenoperationen (z.B. für phonologische Repräsentationen oder Subkategorisierungsinformation) vorzusehen.

Wir verwenden in der vorliegenden Implementation keine vollständigen Relationen und ebenso keine Bedingungen für Merkmalstrukturen. Insbesondere die Annotation von Bedingungen verhindert eine feinkörnige Kontrolle über den Ablauf von Unifikationen, da die Erfüllung von Constraints an Merkmalstrukturen ihrerseits erneut Unifikationen bedingen kann. Hingegen definieren wir Funktionen, die unabhängig von der eigentlichen Unifikation ausgeführt werden und eher prozeduralen Charakter haben (vgl. Abschnitt 3.3.2).

3.2 Maschinelle Übersetzung mit Unifikationsformalismen

Bereits kurz nach der Einführung unifikationsbasierter Formalismen in die Sprachverarbeitung von Kay (1979) wurde deren Eignung und Bedeutung für die Anwendung in der Transferphase der maschinellen Übersetzung deutlich. So schlägt Kay (1984) die Modellierung von Übersetzungswissen innerhalb der *Functional Unification Grammar* (FUG) vor, andere Autoren benutzen ähnliche Methoden. Dabei erweist es sich als notwendig, neben der Identität von Merkmalstrukturen, die i.A. durch Koreferenzen ausgedrückt wird, ebenfalls die Anwendung untergeordneter Transferregeln repräsentieren zu können, die kompositionellen Transfer modelliert. Dies wird in vielen Ansätzen durch eine besondere Form von Transferregeln und einen auf diese Form angepaßten Verarbeitungsmechanismus realisiert.

(Kaplan et al., 1989) benutzen im Rahmen der LFG die dort verwendeten Korrespondenzen, die Relationen zwischen verschiedenen Ebenen linguistischer Repräsentation einer Äußerung darstellen. Zwei Korrespondenzen sind bereits in der ursprünglichen Theorie vertreten: Φ vermittelt zwischen der Konstituentenstruktur (*c-structure*) und der Beschreibungsebene syntaktischer Funktionen (*f-*

structure), die Relation σ beschreibt die weitergehende Abbildung auf die Ebene semantischer Repräsentation (vgl. Kaplan und Bresnan, 1982).

---

beantworten V
(↑ PRED) = 'beantworten ⟨ (↑ SUBJ) (↑ OBJ)⟩'
(τ ↑ PRED FN) = repondre
(τ ↑ SUBJ) = τ (↑ SUBJ)
(τ ↑ AOBJ OBJ) = τ (↑ OBJ)

Abbildung 3.2: Eine Transferregel im LFG-Stil

Um die LFG auf Transferprobleme anwenden zu können, führen Kaplan et al. (1989) eine weitere Relation τ ein, die Beziehungen zwischen Strukturen der Quell- und der Zielsprache beschreibt. Durch die funktionelle Komposition der verschiedenen Relationen können Transferregeln auf unterschiedlichem Abstraktionsniveau realisiert werden.[7] So treten in der Transferregel in Abb. 3.2 Kompositionen von ↑[8] und τ in unterschiedlicher Reihenfolge auf. Diese Reihenfolge entscheidet darüber, ob die Elemente der Relationen auf der quellsprachlichen Seite der Äußerung gesucht werden oder auf der zielsprachlichen Seite. Im vorliegenden Beispiel wird bei der Übersetzung des Verbs "beantworten" ins Französische durch die zusätzliche Pfadangabe AOBJ sichergestellt, daß dem Objekt des Satzes ein "à" vorangestellt wird:

Der	*Student*	*beantwortet*	*die*	*Frage.*
Le	étudiant	répond	la	question.

"L'étudiant répond à la question." **(3.2)**

Eine Reihe von Ansätzen verwendet unterschiedliche Pfade in Merkmalstrukturen, um Informationen zu separieren, die unterschiedliche Sprachen beschreiben. Dazu gehören z.B. Noord (1990), Carlson und Vilkuna (1990), Morimoto et al. (1992). Zajac (1992) verwendet einen getypten Merkmalformalismus, was in etlichen Fällen zu einer Reduktion der Transferregeln führt, wenn Vererbung innerhalb des Typenverbandes ausgenutzt werden kann. Zajac benutzt zwei Merkmale, ENG und FR, um Strukturen in Englisch bzw. Französisch darzustellen, und erlaubt Koreferenz über die Sprachgrenze hinweg.

Rekursiver Transfer wird durch Bedingungen innerhalb des Formalismus (TFS, Emele und Zajac, 1990) ausgedrückt, so daß keine weitergehenden formalismusexternen Mechanismen hierfür erforderlich sind. Eine einfache Übersetzung

[7] Allerdings kann diese prinzipiell elegante Lösung nicht mit Einbettungen umgehen, s. (Sadler und Thompson, 1991).

[8] ↑ ist eine Abkürzung für die unmittelbare Dominanz.

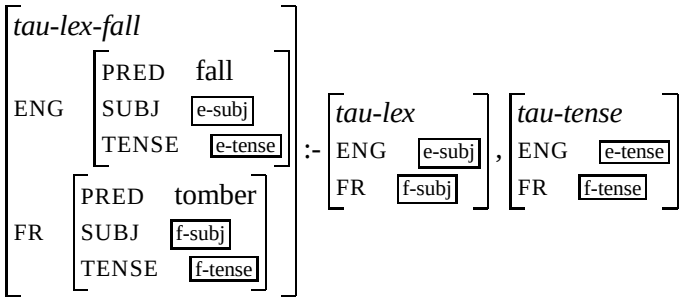


Abbildung 3.3: Eine Transferregel im TFS-Stil

A student falls.
Un étudiant tombe. **(3.3)**

kann erreicht werden durch die Anwendung der in Abb. 3.3 dargestellten Transferregel. Man beachte hierbei, daß tau $\sqsubseteq$ tau-lex $\sqsubseteq$ tau-lex-fall ist.

Die unterschiedlichen Aspekte einer Transferregel werden bei Beskow (1993) durch eine Unterteilung in vier verschiedene Aspekte deutlich gemacht. Als Beispiel diene hier die in Abb. 3.4 gezeigte Regel, welche die Insertion eines Artikels bei der Übersetzung von definiten Nominalphrasen aus dem Schwedischen ins Englische beschreibt.

Die vier Teile einer Transferregel sind:

- Eine eindeutige Kennzeichnung der Regel (`Label`)
- Eine quellsprachliche Struktur, welche die Anwendbarkeit der überprüft (`Source`). Eine Transferregel ist dabei geeignet für die Übersetzung einer Äußerung, wenn die strukturelle Beschreibung mit dem `Source`-Teil unifiziert werden kann. Der Stern (‘`*`’) denotiert dabei die Wurzel des Graphen, Koreferenz wird durch Variablenbindung ausgedrückt (Namen von Variablen beginnen mit ‘`?`’).
- Eine zielsprachliche Struktur, die entweder zur Verifikation einer Übersetzung benutzt werden kann oder diese konstruiert (Das Modell kann für beide Verfahren eingesetzt werden).
- Eine Liste von untergeordneten Transfergleichungen (`Transfer`), die rekursiven Transfer beschreibt. Die einzelnen Gleichungen werden dabei sukzessive abgearbeitet, um die Übersetzung einer komplexen Struktur zu erreichen.

```
Label
    NP-def
Source
    <* cat>     = NP
    <* def>     = DEF
    <* num>     = ?Num
    <* head>    = ?head1
Target
    <* cat>     = NP
    <* def>     = DEF
    <* num>     = ?Num
    <* det lex> = 'the'
    <* head>    = ?head2
Transfer
    ?head1     <=> ?head2
```

Abbildung 3.4: Eine Transferregel von Beskow

Die meisten der vorgenannten Systeme trennen den Kontrollteil des Transfers von der deklarativen Beschreibung kontrastiven Wissens über die beteiligten Sprachen. Anzumerken ist, daß zusätzlich zur Identitätsrelation, die mit der Koreferenz zum Standardinventar aller Unifikationsformalismen gehört, eine Relation des untergeordneten Transfers ausgedrückt werden muß. Dies geschieht durch spezielle Merkmale innerhalb der Strukturen, die von geeigneten Mechanismen ausgewertet werden. Lediglich bei Benutzung von Formalismen, mit denen die Formulierung von Bedingungen möglich ist, kann ein Teilaspekt (die automatische Auswertung von untergeordneten Transferrelationen) formalismusintern durchgeführt werden (Zajac, 1992). Dadurch verliert man allerdings auch die Möglichkeit, spezielle Kontrollstrategien zur Auswahl der anwendbaren Transfergleichungen anzuwenden.

3.3 Architektur und Implementierung des Formalismus

Für die in der vorliegenden Arbeit entwickelten Verarbeitungsmechanismen und Systemmodule wurde eine Implementierung eines merkmalbasierten Formalismus mit Typen und Appropriateness durchgeführt. Die Implementation realisiert die in den vorhergehenden Abschnitten definierten Objekte und auf ihnen operierenden Funktionen und ähnelt daher stark Teilen von ALE (Carpenter und Penn, 1998), mit den bereits erwähnten Beschränkungen, z.B. was die Einführung von Merkmalen angeht.

Die vorliegende Implementierung ist an die Arbeiten von Carpenter und Qu (1995) und Wintner und Francez (1995a) angelehnt, die eine Interpretation von Merkmalstrukturen als abstrakte Maschinen, ähnlich zur *Warren Abstract Machine* für Prolog (Warren, 1983), konzipieren. Eine erste Implementation einer solchen Maschine liegt mit (Wintner, 1997) vor. Wir folgen dieser maschinenartigen Auffassung des Formalismus hier nur insoweit, als zusammenhängende Speicherbereiche zur Darstellung von Merkmalstrukturen verwendet werden, wie dies bereits bei Amtrup (1992) für ungetypte Merkmalstrukturen der Fall war. Die Motivation für eine derartige, im Gegensatz zu üblichen Speicherungsformen als stark verzeigerte Graphen stehende, Modellierung liegt in dem Ziel, einen Formalismus auf die Verwendung in verschiedenen Parallelrechnerumgebungen vorzubereiten. Die Annahme ist, daß in genügend feinkörnigen Systemen der Prozentsatz an Merkmalstrukturen, die zwischen Komponenten eines verteilten Systems ausgetauscht werden müssen, hoch ist. Für konventionelle Implementationen von Merkmalformalismen bedeutet dies, daß die hochgradig diskontinuierliche Repräsentation linearisiert werden muß und diese lineare Form kommuniziert wird. Dies geschieht in den überwiegenden Fällen als Zeichenkette ohne weitere interne Struktur.[9] Auf der Empfängerseite wird nach Erhalt der Zeichenkette diese analysiert (*geparst*), um erneut eine interne Repräsentation der gewünschten Merkmalstruktur zu erhalten, die dann weiter benutzt werden kann.

Im Gegensatz zu diesem Schema sind die Merkmalstrukturen des vorliegenden Formalismus invariant bezüglich Lageverschiebung im Speicher und gegenüber Transport in einen anderen Adreßraum auf einem anderen Rechner. Dies erleichtert die Kommunikation von Merkmalstrukturen enorm, da lediglich zusammenhängende Speicherbereiche verschickt werden müssen, ohne die erwähnte Vor- und Nachbearbeitung. Der Nachteil liegt darin, daß die Routinen zur Unifikation Rücksicht auf die spezielle Art der Kodierung nehmen müssen, was einen erhöhten Verwaltungsaufwand zur Folge hat und verhindert, daß Teilgraphen über mehrere Merkmalstrukturen gemeinsam benutzt werden können (Billot und Lang, 1989). Trotz dieser Tatsache können allerdings effiziente Algorithmen und Darstellungsweisen benutzt werden.

Im Gegensatz zu ALE, TFS (Zajac, 1991) und CUF (Dorna, 1992) ist der hier beschriebene Formalismus nicht als Prolog-Erweiterung konzipiert, sondern als Modul in C++ implementiert. Dies motiviert sich daraus, daß die Möglichkeit zur Parallelisierung von unter Benutzung des Formalismus entstandenen Systemen gefordert wurde. Die Benutzung von C++ garantiert eine Portierungsmöglichkeit auf etliche verfügbare Multiprozessorarchitekturen, da diese ausnahmslos C++ als Sprache zur Verfügung stellen, während “typische” KI-Sprachen (etwa Lisp oder Prolog) im Parallelrechnerumfeld wenig verbreitet sind[10]. Dadurch ist für das in dieser Arbeit entwickelte System MILC nicht nur eine intermodulare Parallelität gegeben (ermöglicht durch die Inkrementalität),

[9] In seltenen Fällen gibt es Kommunikationssysteme, die mit bestimmten komplexen Datentypen umgehen können. Das — ebenfalls in der vorliegenden Arbeit verwendete — Kommunikationssystem ICE (Amtrup, 1994a; Amtrup, 1995b; Amtrup und Benra, 1996) ist z.B. dazu in der Lage, die im Rahmen des Verbmobil-Projektes verwendeten *Verbmobil Interface Terms* (VITs) zu kommunizieren. Sie werden zur Repräsentation flacher semantischer Strukturen benutzt. Aber selbst hier ist eine Analyse der zu übertragenden Datenobjekte notwendig, sie geschieht lediglich transparent für den Benutzer innerhalb der Kommunikationsmechanismen.

[10] Ausnahmen sind etwa die *Connection Machine* (Hillis, 1985), für die ein verktorparalleles Lisp existiert (Steele und Hillis, 1986), sowie einige Implementationen für Transputer (vgl. Kesseler, 1990)

sondern auch eine intramodulare Parallelität.

Der Formalismus sollte gerade mächtig genug sein, daß alle wünschenswerten Eigenschaften modelliert werden können, ohne jedoch einen übermäßigen Aufwand für selten gebrauchte Bestandteile zu treiben. Die Implementation verzichtet deswegen auf die Realisierung von Negation oder komplexer Disjunktion[11] genauso wie auf die Formulierung von Constraints an Merkmalstrukturen wie sie z.B. für TFS realisiert ist. Disjunktion ist ausschließlich für Atome definiert und dient hierbei dazu, Alternativen auf atomarer Ebene darzustellen (z.B. für Mengen unterschiedlicher semantischer Konzepte).

Die Kontrolle über die Verarbeitung von Objekten des Formalismus liegt ausschließlich in der Hand des Benutzers. Das läßt dem Entwickler von Verarbeitungsmodulen eine weitgehende Freiheit über die Wahl und Implementierung von unterschiedlichen Kontrollstrategien, die in einigen anderen Systemen vollständig formalismusimmanent und nur mit großen Schwierigkeiten von außen modifizierbar sind. Für den hier verwendeten Formalismus geht die Kontrollmöglichkeit so weit, daß nicht nur jede Unifikation von Merkmalstrukturen getrennt angestoßen werden kann, sondern auch innerhalb von Unifikationen ablaufende Mechanismen (die Auswertung von Funktionen) getrennt steuerbar sind.[12]

Die Implementierung unterscheidet drei Sätze von Funktionen, die jeweils eine größere Menge an Funktionalität bieten, jedoch auch einen höheren Speicherbedarf haben. Dabei sind die weniger mächtigen Module jeweils vollständig in den mächtigeren enthalten. Das Motiv für diese Dreiteilung liegt erneut begründet in dem Ziel, eine Implementation verfügbar zu halten, die einen Einsatz auf Parallelrechnern erleichtert. Zu diesem Zweck ist es sinnvoll, den Umfang einer Laufzeitversion des Formalismus möglichst klein zu halten, da Parallelrechner normalerweise eng begrenzte Ressourcen pro Knoten besitzen. Die Einschränkungen betreffen zwei Bereiche:

- Die Verfügbarkeit von Namen. Bei der Umwandlung von Merkmalstrukturen von einem externen, lesbaren Format in ein internes Format, das an der Effizienz der Algorithmen orientiert ist, werden sämtliche Namen in eindeutige numerische Bezeichner umgewandelt. Es ist nach dieser Umwandlung nicht mehr erforderlich, alle möglicherweise verwendeten Namen vorzuhalten.
- Die direkte Verfügbarkeit der Typenhierarchie. Die Typen des Typsystems dieser Arbeit werden intern zu Verarbeitungszwecken als Bitvektoren repräsentiert. Für die Umwandlung in dieses Format werden Datenstrukturen und Programmroutinen genutzt, die für eine ablaufende linguistische Anwendung irrelevant sind. Sie können folglich ohne Verlust an Funktionalität aus Laufzeitsystemen ausgeschlossen werden.

[11] Im Rahmen von MILC wird jede Art von komplexer Disjunktion durch die Existenz unterschiedlicher, alternativer Merkmalstrukturen ausgedrückt. Dies erlaubt eine sehr feinkörnige Kontrollsteuerung, die nicht möglich ist, falls Disjunktion innerhalb des Formalismus benutzt wird.

[12] In $\mathcal{TDL}$ (Krieger, 1995), einem ausgesprochen umfangreichen System zur Beschreibung *Constraint*-basierter Grammatiken, kann der Aufruf der Typexpansion individuell gesteuert aufgerufen werden.

Durch den sukzessiven Ausschluß der eben geschilderten Merkmale erhält man drei Ebenen einer Mekmalstrukturmaschine:

- **Eingeschränkte Merkmalstrukturen** (RFS, Restricted Feature Structures). In diesem Modul sind lediglich Ein-/Ausgabefunktionen für bereits in internem Format vorliegende Merkmalstrukturen sowie die notwendigen Unifikationsfunktionen enthalten. Namen von Typen, Merkmalen und Werten können nicht in direkt lesbarer Form ausgegeben werden. Der vornehmliche Einsatzbereich für diese Art von Modul ist die Anwendung in Verarbeitungsteilen im Rahmen einer Parallelrechnerimplementation. Hier wird eine extensive Ausgabe von Objekten des Formalismus nicht benötigt, da etliche Module eines verteilten Systems keinen Zugang zur Außenwelt besitzen. Vielmehr wird es typischerweise ein ausgezeichnetes Modul geben, das als Schnittstelle zwischen dem verteilten System und der Außenwelt fungiert. Lediglich dort sind Ein-/Ausgabefunktionen notwendig, über die der Benutzer mit dem System kommunizieren kann. Jegliche sonstige Kommunikation der Module untereinander kann (und sollte) in interner Form stattfinden.
- **Standard-Merkmalstrukturen** (FS, Feature Structures). Neben den Funktionen für eingeschränkte Merkmalstrukturen werden Funktionen zur Ein- und Ausgabe von Merkmalstrukturen in direkt lesbarer Form geboten. Dies bedingt Analysefunktionen zur Umwandlung in das interne Format. Diese Fassung der Bibliothek muß ebenfalls verwendet werden, wenn Merkmalstrukturen mit Komponenten ausgetauscht werden, die andere Formalismen benutzen.
- **Erweiterte Merkmalstrukturen** (LFS, Large Feature Structures). Zusätzlich zu den bisher erwähnten Möglichkeiten kann mit Hilfe dieser Funktionenbibliothek die Umwandlung von Typenverbänden durchgeführt werden. Das erfordert eine erweiterte Darstellung des im System vorhandenen Namensraumes sowie etliche Funktionen zur Kompilation von Typenverbänden und Analyse der Verbandsdefinitionen.

3.3.1 Definition und Implementation von Typenverbänden

Der vorliegende Formalismus verwendet getypte Merkmalstrukturen. Abb. 3.5 zeigt als Beispiel für einen Typenverband einen kleinen Teil der Strukturierung bezüglich lexikalischer Einheiten. Die zugehörigen Definitionen sind als Abb. 3.6 abgebildet. Das Beispiel zeigt deutlich, wie bestimmte Merkmale sehr früh in der Hierarchie der Typen eingeführt werden. Jedes Zeichen (sign) besitzt die Merkmale phon zur Kennzeichnung der Phonologie, sowie syn und sem zur Repräsentation des syntaktischen und semantischen Gehalts. In der Hierarchie tieferliegende, abgeleitete Typen spezialisieren die Werte bestimmter Merkmale weiter. So wird für den semantischen Gehalt (sem|cont) von Adjektiven gefordert, daß er vom Typ adj-cont sein muß. Diese Einschränkungen können mit der Einführung zusätzlicher Merkmale einhergehen.

Eine solche Definition von Typen resultiert in einem Verband (s. Def. 3.1). Implizit ist stets ein Typ `*top*` ($\top$) definiert, der als Wurzel des Verbandes dient. Weitere Typen werden eingeführt

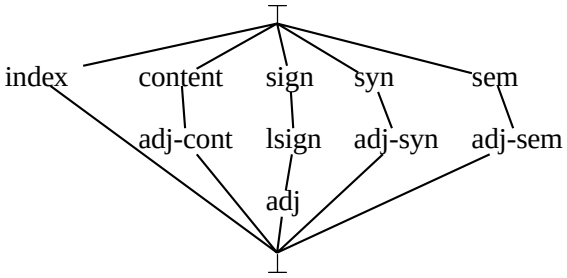


Abbildung 3.5: Ein Ausschnitt aus dem Typenverband

```
type sign isa *top* intro phon:*top* syn:syn sem:sem.
type lsign isa sign.
type syn isa *top*.
type sem isa *top* intro cont:content.
type content isa *top*.
type adj isa lsign intro sem:adj-sem syn:adj-syn.
type adj-sem isa sem intro cont:adj-cont.
type adj-cont isa content intro index:index cond:*top*.
type adj-syn isa syn.
type index isa *top*.
```

Abbildung 3.6: Ein Teil der Strukturierung lexikalischer Einträge

durch die Definition als Subtypen bereits bestehender Typen. Für jeden Typ können zugelassene Merkmale angegeben werden, die Definition der *appropriateness* geschieht parallel zur Definition des Verbandes. Die Syntax für die Definition von Typenverbänden ist in Tab. 3.1 angegeben. Zur einfacheren Strukturierung des Verbandes wurde eine zusätzliche Syntax eingeführt, die ausgehend von einem Typen die Angabe von dessen Subtypen erlaubt.

Typenverband	::=	Eintrag$^+$
Eintrag	::=	Typdefinition | Subtypendefinition **.**
Typdefinition	::=	**type** Typname **[**Supertyp**] [**Appropriateness**]**
Supertyp	::=	**isa** Typname$^+$
Appropriateness	::=	**intro** Approp-Feature$^+$
Approp-Feature	::=	Featurename **:** Typname
Subtypendefinition	::=	**types** Typename **'are'** Typname$^+$
Typname	::=	Symbol
Symbol	::=	Zeichen$^+$
Zeichen	::=	*alle druckbaren Zeichen außer* **[**␣**.:()[]**⟨ ⟩**$,!]**

Tabelle 3.1: Die Syntax des Typenverbandes

Zu den formalen Eigenschaften der Typen und der Appropriateness ist anzumerken, daß im vorliegenden System nicht gefordert wird, daß Merkmale an einem eindeutigen Ort eingeführt werden (s.o.).

Typen liegen in interner Repräsentation als Bitvektoren vor. Diese Darstellungsweise entspricht der in Aït-Kaci et al. (1989) angegebenen kompakten Kodierung für Verbände. Die Wahl von Bitvektoren für die Repräsentation von Typenverbänden hat mehrere Vorteile:

- Die Darstellung von Typen innerhalb einer Hierarchie wird außerordentlich kompakt. Der in dieser Arbeit verwendete Typenverband benötigt für die Darstellung eines von 345 Typen neun Integer-Werte (288 Bits). Eine explizite Repräsentation der Ordnungsrelation auf den Typen ist nicht erforderlich. Innerhalb von Merkmalstrukturen werden Verweise auf den Verband benutzt, nicht die Typrepräsentationen selbst.
- Die Operationen auf dem Typenverband können sehr effizient durchgeführt werden (der Aufwand für eine Typunifikation wird nahezu konstant). Die Prüfung einer Subsumption zwischen Typen und die Unifikation von Typen kann auf ein logisches Und zwischen den Bitvektorrepräsentationen zurückgeführt werden.

3.3.2 Definition und Implementation von Merkmalstrukturen

Die Merkmalstrukturen, die der Formalismus des vorliegenden Systems verarbeiten kann, bilden getypte Merkmalstrukturen mit Appropriateness. Abb. 3.1, die hier als Abb. 3.7 wiederholt wird,

zeigt ein Beispiel für eine Merkmalstruktur.

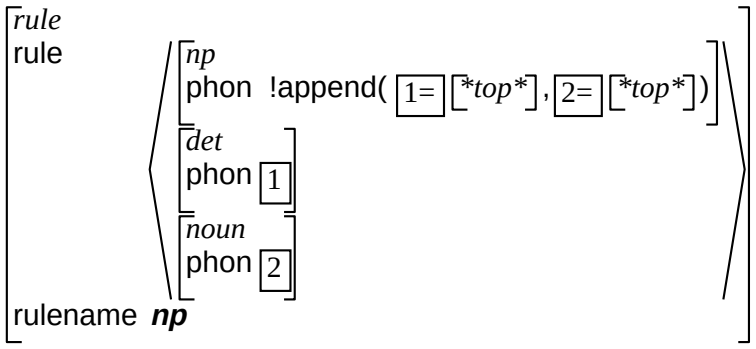


Abbildung 3.7: Merkmalstruktur einer einfachen syntaktischen Regel

Spezielle Eigenschaften der hier modellierten Merkmalstrukturen sind:

- Die Möglichkeit, atomare Disjunktion auszudrücken.
  Dies ist ohne Referenz auf den zugrundeliegenden Typenverband möglich, so daß Mengen gebildet werden können, die außer *top* keinen gemeinsamen Supertyp haben.

- Die direkte Repräsentation von Listen.
  Üblicherweise werden Listen durch Rekurrenz auf die Typenhierarchie gebildet, indem der Typ *list* (Liste) in zwei Subtypen partitioniert wird: *elist* (leere Liste) sowie *nelist* (nichtleere Liste). Der Typ *nelist* besitzt zwei Merkmale, head und tail, wobei die Appropriatenessdefinition fordert, daß der Wert von head vom Typ **top** sei und der Wert von tail vom Typ *list*. Durch Rekursion über den Rest einer Liste können so beliebig lange Listen konstruiert werden. Die im Gegensatz dazu hier gewählte direkte Repräsentation hat den Vorteil, daß die Unifikationsalgorithmen Wissen über die Struktur der beiden in Frage kommenden Merkmalstrukturen haben und diese zur Steigerung der Effizienz ausnutzen können. Darüberhinaus ist so prinzipiell die Einführung von polymorphen Listen möglich, die den Typ ihrer Elemente einschränken (diese Eigenschaft wird im vorliegenden System allerdings nicht ausgenutzt).

- Die Einführung von Funktionsaufrufen.
  Anstelle einer Merkmalstruktur kann der Wert eines Merkmales als Ergebnis des Aufrufs einer Funktion gekennzeichnet werden. Diese, außerhalb der normalen Operationen eines Merkmalformalismus stehende Möglichkeit dient im vorliegenden System vor allem dazu, unterschiedliche phonologische Werte zu konkatenieren.

  Diese Erweiterung des Formalismus ist destruktiv in dem Sinne, daß nicht — wie etwa im Falle der HPSG — Relationen definiert werden, deren Gültigkeit in jedem Stadium einer Merkmalstruktur garantiert wird. Vielmehr wird die Unifikation zweier Merkmalstrukturen zunächst ausgeführt, ohne Rücksicht auf die Funktionsaufrufe zu nehmen. Sie dienen als Platzhalter für zukünftige Werte. Außerdem können sie zunächst nicht innerhalb von Koreferenzen verwendet werden. Nach Durchführung der Unifikation kann der Benutzer ent-

scheiden, Funktionen auswerten zu lassen. Der Funktionsaufruf kann fehlschlagen, was einem Unifikationsfehler entspricht. Gelingt der Funktionsaufruf, so wird der Rückgabewert der Funktion anstelle des Funktionsaufrufes in die Merkmalstruktur eingesetzt; der Funktionsaufruf verschwindet. Sind alle Funktionsaufrufe erfolgreich ausgewertet worden, so ist das Resultat eine "nebeneffektfreie" Merkmalstruktur.

TFS	::=	**[**Typname**]** FS | Liste | OffeneListe | Paar | Koref | KorefDef | Nil | Atom | Menge | String | Funktion
FS	::=	**[(**MerkmalName TFS**)*** **]**
Liste	::=	⟨ FS* ⟩
OffeneListe	::=	⟨ FS* **...** ⟩
Paar	::=	⟨ FS **.** FS ⟩
Koref	::=	**%**Name
KorefDef	::=	**%**Name**=** FS
Nil	::=	**nil**
Atom	::=	TypName
Menge	::=	**(** TypName* **)**
String	::=	**"** Zeichenkette **"**
Funktion	::=	**!**FunktionsName**(** Argument* **)**
Typname	::=	Symbol
Symbol	::=	Zeichen$^+$
Zeichen	::=	*alle druckbaren Zeichen außer* **[**␣**.:()[]**⟨ ⟩**$,!]**

Tabelle 3.2: Die Syntax von Merkmalstrukturen

Die Merkmalstrukturen gehorchen der in Tab. 3.2 angegebenen Syntax. Die Angabe eines Typen für eine Merkmalstruktur ist redundant, falls er aus dem Kontext und der Definition der Appropriateness erschlossen werden kann. Eine Klassifikation von Merkmalstrukturen kann allerdings nicht durchgeführt werden. Dies liegt zum einen daran, daß in der vorliegenden Implementation nicht die Eindeutigkeit des ein Merkmal einführenden Typen gefordert wird, zum anderen daran, daß lediglich wohlgetypte (vgl. Def. 3.7), nicht aber vollständig wohlgetypte Merkmalstrukturen (vgl. Def. 3.8) repräsentiert werden; Merkmale, die von der Appropriateness gefordert sind, können leer bleiben.

Das interne Format von Merkmalstrukturen im Speicher besteht im wesentlichen aus einem Feld von Paaren, die Knoten einer Merkmalstruktur darstellen. Jedes Paar besteht aus der Angabe einer Sorte `sort`, welche die Semantik des direkt darauf folgenden Wertfeldes `value` bestimmt (vgl. Abb. 3.8). Referenzen innerhalb einer Merkmalstruktur werden direkt durch den Index innerhalb des Feldes ausgedrückt, sind also Zeiger relativ zum Anfang des Feldes. Dadurch wird eine zusammenhängende Repräsentation garantiert, die — wie bereits erwähnt — dazu führt, daß Merkmalstrukturen einfach und effizient kommuniziert werden können.

In Tabelle 3.3 wird eine Merkmalstruktur in interner Darstellung gezeigt. Sie entspricht der in Abb.

```
typedef struct FNode {
  int sort;           // Sort of node (Atom, Conj, Disj, etc.)
  int value;          // Value of this node (type, reference, ...)
} FNode;
```

Abbildung 3.8: Definition von Knoten innerhalb von Merkmalstrukturen

3.7 angegeben. Während der Unifikation sind weitere Eigenschaften jedes Knotens zu repräsentieren, etwa, um den Union-Find Algorithmus zu realisieren, der bei dem hier verwendeten Algorithmus von Huet (vgl. Knight (1989, S. 98)) benutzt wird.

3.4 Zusammenfassung

Die Verwendung eines unifikationsbasierten Formalismus in einem System zur Verarbeitung natürlicher Sprache hat bedeutende Vorteile. Die deklarative Art der Beschreibung linguistischen Wissens ist anderen Ansätzen weit überlegen. Darüberhinaus existieren mittlerweile effiziente Algorithmen zur Berechnung der Unifikation zweier Merkmalstrukturen, die auch als Implementation in Form einer abstrakten Maschine vorliegen.

Deartige Formalismen lassen sich gut für den Transferschritt in einem maschinellen Übersetzungssystem einsetzen, wie zahlreiche Beispiele belegen. Wir haben in diesem Kapitel die Konzeption und Implementation des in dieser Arbeit verwendeten Merkmalformalismus dargelegt. Er realisiert komplexe, getypte Merkmalstrukturen mit einer Zulässigkeitsfunktion. Die Darstellung von Merkmalstrukturen in einer kompilierten Form ist sehr kompakt und eignet sich durch eine gegenüber Verschiebungen invariante Semantik gut für die Anwendung in verteilten Systemen.

Knoten Nr.	Knoten Sorte	Knoten Wert	Beschreibung
0	0	12	Zeiger auf Funktionendefinitionen
1	FST_TYPE	rule	Oberster Knoten der Merkmalstruktur
2	FST_FEAT	6	Zeiger auf das Merkmal rule
3	FST_FEAT	4	Zeiger auf das Merkmal rulename
4	FST_STRING	3	Zeichenkette der Länge 3
5	1852833792	0	Zeichenkettenrepräsentation
6	FST_LIST	0	Das Merkmal rule ist eine Liste
7	FST_FIRST	9	Zeiger auf den Kopf der Liste
8	FST_REST	20	Zeiger auf den Rest der Liste
9	FST_TYPE	np	Linke Seite der Regel
10	FST_FEAT	11	Zeiger auf das Merkmal phon
11	FST_EMPTY	0	phon wird durch eine Funktion gefüllt
12	FST_LIST	0	Hier beginnen die Funktionendefinitionen
13	FST_FIRST	15	Zeiger auf die erste Funktion
14	FST_REST	19	Zeiger auf den Rest der Funktionen
15	FST_FUNC	1	Funktion *append()*
16	FST_ARG	11	Zeiger auf den Resultatwert
17	FST_ARG	25	Zeiger auf das erste Argument
18	FST_ARG	31	Zeiger auf das zweite Argument
19	FST_EMPTY	0	Keine weiteren Funktionen vorhanden
20	FST_LIST	0	Fortsetzung der Regeldefinition
21	FST_FIRST	23	Zeiger auf das zweite Element der Regel
22	FST_REST	26	... und weitere Elemente
23	FST_TYPE	det	Definition des Artikels
24	FST_FEAT	25	Zeiger auf das Merkmal phon
25	FST_TYPE	*top*	Leere Merkmalstruktur
26	FST_LIST	0	Fortsetzung der Regeldefinition
27	FST_FIRST	28	Zeiger auf das dritte Element der Regel
28	FST_REST	32	
29	FST_TYPE	noun	Definition des Nomen
30	FST_FEAT	31	Zeiger auf das Merkmal phon
31	FST_TYPE	*top*	Leere Merkmalstruktur
32	FST_NIL	0	Ende der Liste von Regelelementen

Tabelle 3.3: Tabellarische Darstellung einer Merkmalstruktur

Kapitel 4

MILC: Struktur und Implementierung

Diese Kapitel beschreibt die architektonische Struktur und die Implementierung des Systems MILC (Machine Interpretation by Layered Charts).

MILC ist ein System zur inkrementellen Übersetzung spontan gesprochener Sprache unter Berücksichtigung von Integriertheit und Uniformität. Die Motive und wesentlichen Eigenschaften sind bereits in der Einführung in Kapitel 1 erläutert worden, sie seien hier jedoch ein weiteres Mal kurz erwähnt:

- Die Hauptmotivation und das wesentliche Merkmal von MILC ist die inkrementelle Verarbeitungsweise, die durchgehend in allen Komponenten realisiert ist. Ein wesentliches Ziel der Implementation besteht darin, eine Experimentierplattform für weitergehende architektonische Forschungen zur Verfügung zu stellen.
- Die Verteilung von Komponenten auf unterschiedliche Prozesse oder Maschinen verhindert die Existenz eines globalen Systemzustandes. Allerdings ist durch die Implementierung mit Hilfe von Mehr-Ebenen-Charts, die weiter unten erläutert werden, gewährleistet, daß die Vereinigung der zu jedem Zeitpunkt zur Verfügung stehenden Information konsistent ist; sie zeigt einen fast aktuellen Systemzustand.[1]
- Die Einführung von Mehr-Ebenen-Charts etabliert gleichzeitig eine integrierte Art der Verarbeitung. Jede Komponente kann prinzipiell mit jeder anderen Daten austauschen, ohne daß spezielle Schnittstellen eingeführt werden müssen. Der Datentyp, der ausgetauscht wird, beschreibt eine Kante in der Chart mit den ihr innewohnenden Komponenten.
- Die Benutzung eines uniformen Formalismus in möglichst vielen Komponenten erleichtert die Kommunikation verschiedener Komponenten weiter. Der in Kapitel 3 beschriebene Merk-

[1] In einem verteilten System kann es keinen global gültigen Systemzustand geben. Gemeint ist hier, daß eine Komponente, welche die Vereinigung durchführt, Information aus der nahen Vergangenheit der übrigen Module erhält.

malformalismus ist Grundlage der Verarbeitung in allen im Rahmen von MILC implementierten Komponenten.

Wir werden in diesem Kapitel zunächst die zugrundeliegende Struktur, die Mehr-Ebenen-Chart, definieren und erläutern, sowie sie gegenüber anderen möglichen Datenstrukturen abgrenzen. Anschließend wird ein Überblick über die Kommunikationsinfrastruktur gegeben, die das Gelingen einer verteilten Arbeitsweise für MILC garantiert. Nach einer kurzen Einführung in die architektonische Struktur werden wir auf besondere Aspekte der Rolle und des Verhaltens der einzelnen Komponenten eingehen. Der Abschluß dieses Kapitels wird durch eine kritische Würdigung der möglichen und sinnvollen Erweiterungen des Ansatzes gebildet.

4.1 Layered Charts

Das wichtigste, einer Darstellung natürlicher Sprache zugrundeliegende, Ordnungskriterium ist die Zeit. Sprache ist linear in der Zeit geordnet. Im Falle geschriebener Eingabe ist diese Eigenschaft durch die räumliche Anordnung von Buchstaben und Wörtern gegeben; sie kann allerdings aufgeweicht werden, sei es durch den Autor, der durch stilistische Mittel (Strukturierung in Abschnitte etc. oder auch durch die äußere Gestaltung) zusätzliche Bedeutung in den Text einbringen kann, sei es durch den Leser, dem in der Regel freigestellt ist, vor- und zurück zu blättern oder “quer zu lesen”. Nimmt man jedoch wie in dieser Arbeit gesprochene Sprache an, so gilt die Linearität fast vollkommen.[2]

Die Ursache dafür ist natürlich, das gesprochene Sprache in Form von zeitlich sich änderndem Schalldruck erzeugt und wahrgenommen wird. Dieser Tatsache wird im Rahmen einer automatischen Worterkennung dadurch Rechnung getragen, daß deren erste Stufe, die akustische Vorverarbeitung, spektrale Signale innerhalb sich zeitlich verschiebender Fenster berechnet und daraus Merkmalvektoren konstruiert, welche relevant erscheinende Eigenschaften des Sprachsignals im jeweiligen Zeitfenster kodieren. Diese Vektoren werden dann normalerweise in Automaten-basierten Verfahren dazu verwendet, Wörter zu suchen, deren Modelle möglichst gut mit mit der eingehenden zeitlichen Folge von Merkmalvektoren übereinstimmt. Heutzutage werden hierfür fast ausschließlich HMM-basierte Verfahren genutzt (vgl. Hauenstein, 1996). Hier wird nicht weiter auf diese frühen Phasen der Sprachverarbeitung eingegangen, es erscheint jedoch wichtig, die zeitliche Gestalt der Eingabe möglichst früh deutlich zu machen.

Im Rahmen des hier vorgestellten Modells von Sprachverarbeitung ist ebenfalls evident, daß Verbindungen zwischen Modulen zur Spracherkennung und Modulen zur linguistischen Verarbeitung möglich sind, die über die Weitergabe von Worthypothesen hinausgehen. Wünschenswert ist eine engere Integration von *Speech*- und *Language*-Verarbeitung, die eine Verbesserung der Worterkennung aufgrund von linguistischen Tatbeständen ermöglicht. Auch hierfür ist die zeitliche Gestalt der

[2] Wir sehen hier ab von technischen Möglichkeiten der Manipulation, etwa durch Vor- oder Zurückspulen eines Bandes.

Eingabe als Strom von Merkmalvektoren von eminenter Bedeutung. Wir werden auf diese Sichtweisen in Abschnitt 4.11 zurückkommen.

Die initiale Eingabe in das System MILC besteht aus dem Resultat eines automatischen Worterkennungsprozesses. Es ist gegeben durch eine Menge von Worthypothesen, die möglicherweise während eines bestimmten zeitlichen Abschnitts gesprochene Wörter repräsentieren. Konstruiert wird hieraus ein Wortgraph, wie in Definition 2.13 angegeben; genauer gesagt, wird ein Hypergraph von Worthypothesen konstruiert, dies spielt für die augenblickliche Argumentation jedoch keine Rolle. Betrachtet man diesen Graphen nicht als dedizierten Wortgraphen, sondern als *Interpretationsgraphen*, der eine kompakte Art der Darstellung für möglicherweise konkurrierende Interpretationen von Intervallen des Sprachsignals enthält, so kann man jede Worthypothese als Hypothese über den Inhalt eines bestimmten Signalabschnitts ansehen. Ist dies noch trivial, da ja schließlich der Wortgraph in genau dieser Weise konzipiert ist, als Graph von Hypothesen über möglicherweise geäußerte Wörter, so ermöglicht eine abstraktere Sichtweise auf Graphen es, ebenfalls syntaktische Analysen, semantische Beschreibungen, und sogar Oberflächenhypothesen in einer anderen Sprache als Hypothesen über bestimmte linguistische Eigenschaften von Ausschnitten des Eingabesignals anzusehen.

Diese Art der Betrachtung ist der erste Schritt auf dem Wege zur Konstruktion einer Mehr-Ebenen-Chart. Er erlaubt die Einführung einer graphenartigen Struktur zur Repräsentation von Ergebnissen.

Um die im Laufe der Bearbeitung erzielten Ergebnisse zu speichern, kann als einfachste Maßnahme eine monotone Informationsanreicherung angenommen werden, die auf einer uniformen Art der Darstellung basiert. Auf diesem Wege enthalten die Interpretationen, die an Kanten des Interpretationsgraphen annotiert sind, stets alle notwendige Information, die zur weiteren Verarbeitung notwendig ist. Es ist sogar möglich, einen uniformen Verarbeitungsmechanismus zu etablieren, um die Informationsstrukturen zu manipulieren, wie dies in KIT FAST (Weisweber, 1992; Weisweber, 1994) durchgeführt wurde. Allerdings bringt dies mit sich, daß alle verfügbare Information auch in jedem Fall berücksichtigt werden muß, selbst wenn sie im jeweiligen Modul nicht vollständig genutzt wird; zumindest muß sie aktiv ignoriert werden und wird überflüssigerweise über (normalerweise mit Bandbreitenbegrenzungen behaftete) Kommunikationsmedien zwischen Komponenten ausgetauscht.

Das Ziel einer effizienten Repräsentations- und Speicherungsform ist folglich, jeder Komponente nur soviel Information über ein linguistisches Objekt zur Verfügung zu stellen, wie diese zur Bearbeitung benötigt. Diese Art des *information hiding* trägt viel dazu bei, daß Module einfacher strukturiert werden können; zusätzlich kann die Verarbeitungsgeschwindigkeit steigen. Systeme, die auf *Blackboards* (Hayes-Roth, 1985; Engelmore und Morgan, 1988) basieren, implementieren einen derartigen Ansatz der Informationskontrolle, s. z.B. Hearsay II (Erman et al., 1980). Alle Resultate werden zentral verwaltet und typisiert. Eine Komponente erhält folglich nur die für sie relevanten Daten, ohne erkennen zu können, wie die *Blackboard* insgesamt strukturiert ist oder welche Daten in ihr enthalten sind. Bezogen auf die einzelne Komponente ist eine solche Form der Datenhaltung ideal. Es muß allerdings angemerkt werden, daß der Einsatz einer *Blackboard* zur zentralen Speicherung von Daten immer auch den Einsatz einer zentralen Kontrollstrategie impliziert. Dies ist einerseits vorteilhaft, da der Prozeß, der die *Blackboard* kontrolliert, dadurch in die Lage versetzt

wird, die Reihenfolge der Bearbeitung einzelner Daten zu überwachen und eine Präferenzordnung festzulegen (Carver und Lesser, 1994). Nur der Kontrollprozeß hat das hierfür notwendige Wissen, während jede einzelne der Komponenten der eigentlichen Applikation ausschließlich aufgrund lokaler Gegebenheiten Entscheidungen fällen kann. Auf der anderen Seite jedoch stellt gerade diese zentrale Speicherung und Kontrolle eine entscheidende architektonische Hürde dar, indem sie die Möglichkeiten zur Einführung von Parallelverarbeitung zur Performanzsteigerung stark einschränkt (vgl. Kesseler, 1994):

- Der konkurrente Zugriff auf die *Blackboard* durch mehrere Prozessoren (die jeweils eine Komponente des Systems ausführen sollen) kann dazu führen, daß die Speicherbandbreite nicht mehr ausreicht, alle Anfragen gleichzeitig durchzuführen. Die Busschnittstelle muß die Zugriffe verzögern, was einen geringeren Durchsatz als theoretisch möglich nach sich zieht.
- Selbst ohne Berücksichtigung einer beschränkten Busgeschwindigkeit müssen die Zugriffe auf die *Blackboard* partiell serialisiert werden, da jeweils nur ein Prozeß schreibend auf sie zugreifen darf. Im schlimmsten Fall kann dies zu einer vollständigen Serialisierung der Zugriffe führen.

Eine partielle Lösung für dieses Problem stellen verteilte *Blackboards* dar (Jagannathan, Dodhiawala, und Baum (eds.), 1989, Teil II) dar, mit deren Hilfe Engpässe beim Zugriff auf die *Blackboard*, die durch eine zu geringe Speicherbandbreite verursacht werden, beseitigt werden können. Für die Serialisierung der Zugriffe, die aufgrund der Notwendigkeit der Konsistenzerhaltung der Daten entstehen, gilt jedoch nach wie vor, daß eine zentrale Kontrolle im Sinne einer *Blackboard* potentiell Performanzeinbußen nach sich zieht. Denecke (1997) beschreibt ein verteiltes *Blackboard*-System, das die Kontrolle über die Komponentenkommunikation durch eine Menge von Regeln steuert, die an ein Expertensystem erinnern. Sie beschreiben,, wann und unter welchen Umständen die Dienste einer Komponente in Anspruch genommen werden. Letztlich ändert dies natürlich nichts an der Rolle der zentralen Kontrolle, die hier von einer sog. Diskurs-*Blackboard* übernommen wird. Jegliche Aktionen der Komponenten sind serialisiert.

Boitet und Seligman (1994) schlagen einen Ansatz vor, den sie "*Whiteboard*" nennen und der für die vorliegende Arbeit einige Bedeutung hat. Ausgehend von der Suboptimalität von *Blackboards* konstruieren sie ein Architekturschema, das einige der Schwierigkeiten zu lösen verspricht. Sie charakterisieren zwei Hauptprobleme, die ein rein sequentieller Ansatz aufwirft: Informationsverlust und Mangel an Robustheit. Um beide zu demonstrieren, verweisen sie auf *Asura* (Morimoto et al., 1993), eines der Vorgängersysteme. Dort wird nicht alle in einem Modul gewonnene Information weitergegeben, was in vielen Fällen dazu führt, daß im weiteren Verlauf der Bearbeitung reanalysiert werden muß. Außerdem werden partielle Ergebnisse (etwa syntaktische Analysen, die zwar komplett sind, aber nicht wohlgeformte Sätze repräsentieren), zurückgehalten, was in manchen Fällen zum Scheitern des Gesamtsystems führt, obwohl partielle Übersetzungen möglich wären.

Die Anwendung einer *Blackboard*-Architektur löst nach Boitet und Seligman (1994) zwar das Problem des Informationsverlustes, trägt jedoch nicht zur Steigerung der Robustheit bei. Zusätzlich würden weitere Komplikationen entstehen, die im Prinzip mit der Konkurrenz von Komponenten

beim Zugriff auf die *Blackboard* zusammenhängen, z.B. Effizienzprobleme und komplizierte Fehlersuche.

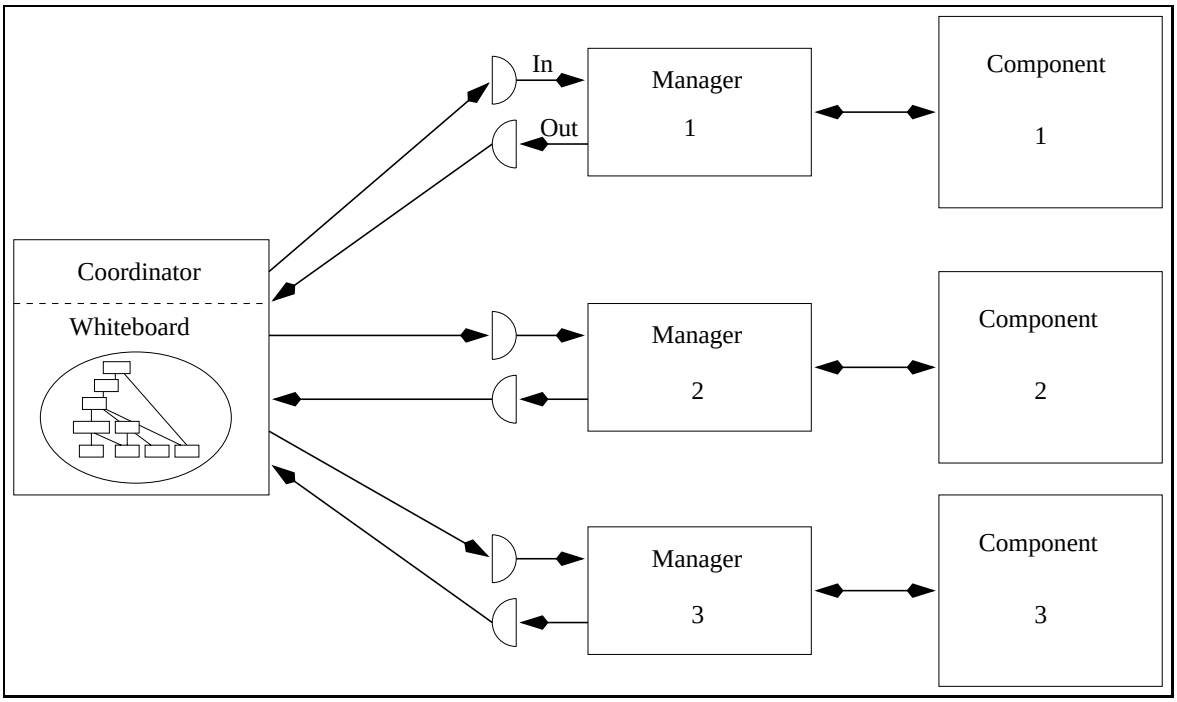


Abbildung 4.1: Die *Whiteboard*-Architektur (nach Boitet und Seligman (1994))

Der *Whiteboard*-Ansatz, den Boitet und Seligman (1994) benutzen, trage zur Beseitigung dieser Schwierigkeiten bei. Das Architekturschema ist das in Abb. 4.1 gezeigte. Eine zentrale Datenstruktur wird von einem Koordinator genannten Prozeß verwaltet. Der Koordinator empfängt Resultate von Komponenten und sendet seinerseits Daten an Module. Gleichzeitig wird alle Information gespeichert, zusammen mit einer Art von Begründungsverwaltung, die auf dem Weg beruht, den ein Datum durch die Applikation nimmt (Doyle, 1979). Die Art der Datenstruktur sei prinzipiell irrelevant, allerdings wird vorgeschlagen, daß eine Verbandsstruktur (*Lattice*) angemessen für die Verarbeitung gesprochener Sprache sei. Die interne Architektur einer Komponente ist ebenfalls nicht von entscheidender Bedeutung, da generell sog. Manager für eine Kapselung der Komponenten sorgen. Diese Manager verwalten Warteschlangen für eingehende und ausgehende Nachrichten und operieren periodisch auf diesen. Dadurch hat das Datenformat, das eine Komponente benutzt, ebenfalls untergeordnete Bedeutung, da Manager Umsetzungsvorgänge initiieren können. Dies trägt zur Wiederverwendbarkeit von existierenden Modulen bei.

Neben der Speicherung von Daten kann der Koordinator der *Whiteboard* ebenfalls Funktionen ausführen, die an die Kontrollprozesse einer *Blackboard* erinnern. Das bedeutet, er kann eine zentrale Kontrolle ausüben und die Verarbeitung der einzelnen Komponenten durch die Art, in der Daten präsentiert werden, steuern.

Boitet und Seligman (1994) geben an, daß mit Hilfe der *Whiteboard* ein inkrementelles Verhal-

ten der Gesamtapplikation simuliert werden könne. Innerhalb des Prototypensystems *KASUGA*, das eine *Whiteboard* benutzt, existieren zwar ausschließlich nichtinkrementelle Komponenten, Inkrementalität wird hier simuliert durch die Zurückhaltung von Ergebnissen einer Komponente durch deren Manager und die stückchenweise Weitergabe an den Koordinator. KASUGA besteht aus

- einem Phonemerkenner, der für das eingehende Sprachsignal einen Graphen erzeugt, der jeweils die drei besten Phoneme für ein Intervall berechnet,[3]
- einer syntaktischen Strukturanalyse, die unter Benutzung einer kontextfreien Grammatik aus dem Phonemgraphen Wörter erzeugt, und
- einem bilingualen (Japanisch-Englisch) Lexikon, das eine Wort-für-Wort Übersetzung ermöglicht. Das System kann 40 elementare Phrasen (*bunsetsu*) übersetzen.

Während jedoch die Konzeption einer *Whiteboard* einen beträchtlichen Fortschritt gegenüber sequentiellen Techniken darstellt, so zeigt sich nach sorgfältiger Analyse, daß sie nicht den Anforderungen genügt, die wir in der Einleitung an ein Architekturschema für die Verarbeitung natürlicher gesprochener Sprache gestellt haben. Selbst wenn die einzelnen Komponenten isoliert voneinander existieren und kein Wissen über die gegenseitige Existenz oder Verarbeitungsstrategie zu haben brauchen, so gibt es doch nach wie vor eine zentrale Kontrollinstanz, die nicht nur alle Arten von Resultaten und Zwischenergebnissen speichert, sondern zudem Kontrollmöglichkeiten besitzt. Das bedeutet, daß auch unter Benutzung einer *Whiteboard* unnötige Serialisierungen von Aktionen auftreten werden.

Die anfangs von uns geforderte Inkrementalität auf allen Ebenen einer Anwendung kann im eben beschriebenen Ansatz lediglich simuliert werden. Das bedeutet zumindest einen Verlust an Performanz, da der Nachlauf einer Komponente gleich ihrer Gesamt-Bearbeitungszeit ist. Durch die Art der Weitergabe wird lediglich eine inkrementelle Ausgabe simuliert; über die Fähigkeit einzelner Module, Eingabe inkrementell zu verarbeiten, ist jedoch nichts gesagt. Wichtiger als das Vorliegen von Performanzeinbußen ist jedoch, daß prinzipiell keine Interaktion zwischen verschiedenen Modulen stattfinden kann, da dasjenige Modul, auf das Einfluß genommen werden sollte, seine Bearbeitung längst abgeschlossen hat. Gemessen an dem Kriterium, Inkrementalität zu explorieren, kann KASUGA daher bestenfalls eine frühe Demonstrationsposition einnehmen.

Aus den bisherigen Ausführungen wird ersichtlich, daß eine Datenstruktur, die für eine inkrementelle Verarbeitung spontan gesprochener Sprache geeignet ist, zumindest folgende Eigenschaften besitzen muß:

- Sie ermöglicht ein dezentrales Schema zur Datenspeicherung, das nicht auf einem zentralen Datenbestand beruht, um Engpässe zu vermeiden, die nicht aus der eigentlichen Funktion der Anwendung hervorgehen.

[3] D.h. offensichtlich, daß ein Phonemgraph der transkriptunabhängigen Dichte drei erzeugt wird.

- Sie läßt die Sicht auf Daten in mehreren Abstraktionsgraden zu, um Auswahl und Überwachung des Bestandes zu erleichtern.
- Zur Kommunikation von Datenobjekten zwischen Modulen des Systems stehen effiziente Kommunikationsmechanismen zur Verfügung, die ohne einen hohen Verwaltungsaufwand operieren.
- Die Datenstruktur selbst sollte von allen Komponenten aus in gleicher Weise zugreifbar sein, so daß der Austausch von Objekten einfach und effizient sowie ohne Verlust von Bedeutung möglich ist.
- Sie sollte die Speicherung von linguistischen Objekten in einem uniformen Formalismus unterstützen, wiederum aus Gründen der Transparenz und Effizienz der Gesamtanwendung.

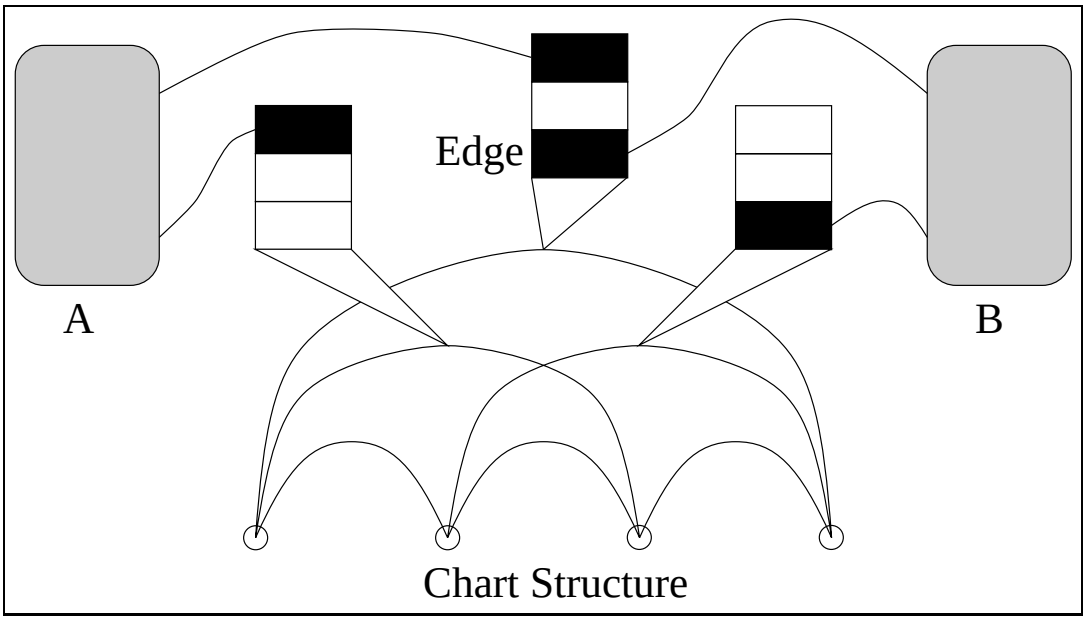


Abbildung 4.2: Der grundsätzliche Aufbau einer Mehr-Ebenen-Chart

Die für MILC konzipierte Datenstruktur der Mehr-Ebenen-Chart realisiert diese Konzepte weitgehend. Das ihr zugrundeliegende Prinzip ist in Abb. 4.2 dargestellt. Jedes Zwischen- und Endergebnis, das von einer Komponente produziert wird, realisiert eine Hypothese über einen bestimmten Ausschnitt des Eingabesignals. In diesem Sinne stellt der Hypergraph, der aus der Worterkennung resultiert, die unterste Ebene der Mehr-Ebenen-Chart dar. Darauf aufbauend, werden zusätzliche Hypothesen durch verschiedene Komponenten erzeugt, die anfänglich lediglich jeweils eine Wortkante überspannen. Im Laufe der Verarbeitung können Kanten kombiniert werden, wie u.a. in Abschnitt 2.6.3 beschrieben. Zu jedem Zeitpunkt bildet die verteilte, dezentrale Mehr-Ebenen-Chart einen Interpretationsgraphen von Hypothesen über die Eingabe. Die wesentlichen Komponenten einer Kante sind:

- **Identifikation**: Eine systemweit eindeutige Identifikation jeder Kante ist essentiell für Referenzen, die über den lokalen Adreßraum einer Komponente hinausgehen.
- **Domäne**: Der Ausschnitt der Eingabe, der durch die Hypothese repräsentiert wird, wird in der Praxis durch die Mengen von Start- und Endknoten der Hyperkante beschrieben. Selbstverständlich müssen dabei die für Hypergraphen anwendbaren Restriktionen eingehalten werden.
- **Beschreibung**: Die Repräsentation des linguistischen Objektes, das durch die Kante überspannt wird. Mit Ausnahme der Hypergraphenerzeugung durch die Worterkennung, für die die orthographische Form der Wörter ausreicht, verwenden wir durchgehend getypte Merkmalstrukturen, die in dem in Kapitel 3 erläuterten Formalismus kodiert sind.
- **Bewertung**: Jede Kante trägt eine Bewertung, die prinzipiell aufgrund unterschiedlicher Evidenz berechnet worden sein kann. In der augenblicklichen Fassung von MILC verwenden wir sowohl akustische Bewertungen als auch solche, die aus einem stochastischen Sprachmodell gewonnen wurden.
- **Vorgänger**: Jede Kante wird mit Informationen zur Begründungsverwaltung ausgestattet. Dies bedeutet, daß Angaben darüber gemacht werden, welche Ausgangskanten zu ihrer Konstruktion geführt haben. Wortkanten besitzen keine Vorgänger. Eine andere Kante kann genau eine Vorgängerkante haben, etwa im Falle des Lexikonzugriffs innerhalb der syntaktischen Verarbeitung. Genausogut sind ebenfalls zwei Vorgängerkanten möglich, wenn eine der Varianten der fundamentalen Regel, welche die einzige Methode zur Verbindung von Kanten und zur Erstellung weiterspannender Analysen darstellt (s.u.), angewendet wird. Da diese Regel stets die Kombination genau zweier Kanten erstellt, sind nie mehr als zwei Kanten Vorgänger einer einzelnen Kante. Neben der Vorgängerrelation wird gleichermaßen die komplementäre Nachfolgerrelation gespeichert, um die Durchführung von Vererbungsoperationen zu erleichtern. Diese können immer dann auftreten, wenn die Worterkennung eine Worthypothese in eine bereits bestehende Hyperkante integriert. Falls die Hyperkante schon irgendwelchen linguistischen Operationen unterworfen war, mag es sein, daß aus ihr neue Kanten entstanden sind. In diesem Fall müssen die durch die Einfügung der Worthypothese modifizierten Mengen von Start- und Endknoten, sowie ggfs. eine modifizierte akustische Bewertung, vererbt werden.
- **Verwaltung**: Zusätzlich wird eine ganze Menge von Verwaltungsinformation gespeichert, die entweder aus den bisher erwähnten Feldern erzeugt werden kann oder die keine unmittelbare linguistische Funktion hat (z.B. Angaben darüber, in welcher Darstellungsart eine Kante ausgegeben werden soll, etc.).

Mehr-Ebenen-Charts sind eine Erweiterung der Charts, wie sie von Kay (1973) ursprünglich für die strukturelle Analyse natürlicher Sprache vorgeschlagen wurden. Der Ausgangspunkt der Überlegungen bestand darin, daß gewisse partielle Analysen mehrfach im Laufe einer Bearbeitung benutzt werden können, so daß es hilfreich ist, gewonnene Analysen festzuhalten (in einer sog. *wellformed substring table*). Eine solche Tabelle kann erweitert werden, indem zusätzlich unvollständige

Analysen dort eingetragen werden. Diese (kleine) Modifikation hat enorme Auswirkungen auf die Komplexität von Algorithmen für die natürlichsprachliche Verarbeitung. So kann der Aufwand für die Strukturanalyse mit Hilfe kontextfreier Grammatiken von exponentieller Komplexität auf $O(n^3)$ reduziert werden (Sheil, 1976).

Zum Zeitpunkt der Einführung der Chart waren unvollständige Analysen mit Phrasenstrukturregeln verknüpft, deren rechte Seiten noch nicht vollständig konsumiert waren. Solche Kanten werden *aktiv* genannt, um anzudeuten, daß sie noch nach weiterem Material suchen, bevor die Analyse komplett ist. *Inaktive Kanten* hingegen repräsentieren vollständig durchgeführte Analysen, die entweder als Resultat ausgegeben werden, oder auf die Inkorporation in größerem Zusammenhang warten.

Definition 4.1 (Chart)
Eine Chart ist ein gerichteter, azyklischer, linksverbundener Graph mit Kantenbezeichnungen. Die Menge von Kanten ist in zwei disjunkte Teilmengen partitioniert:

- *Aktive Kanten repräsentieren unvollständige Analysen*
- *Inaktive Kanten repräsentieren vollständige Analysen*

Zusätzlich wird eine Funktion definiert, die für inaktive Kanten entscheidet, ob diese Zielstatus haben und ausgegeben werden müssen oder nicht.

Zur Durchführung von Analysen mit Hilfe dieser Datenstruktur werden normalerweise drei zentrale Prozeduren definiert: Das Einfügen von Kanten in die Chart (INSERT), das Vorschlagen neuer Hypothesen aufgrund abgeschlossener Analysen (PROPOSE) sowie die fundamentale Regel (COMBINE). Wir werden an dieser Stelle lediglich auf einige Aspekte der fundamentalen Regel eingehen und verweisen für detaillierte Informationen auf die einschlägige Literatur (Winograd, 1983; Thompson und Ritchie, 1984; Allen, 1987; Görz, 1988).

Die fundamentale Regel kombiniert zwei Kanten und konstruiert daraus im Erfolgsfall eine neue Kante. Normalerweise (z.B. in einem Standardsystem zur Strukturanalyse) wird eine aktive Kante mit einer inaktiven Kante kombiniert, wobei der Startknoten der inaktiven Kante gleich dem Endknoten der aktiven Kante sein muß. Daraus wird eine Kante erzeugt, die beide überspannt. Es sind jedoch auch Kombinationen möglich, die die inaktive Kante auf der linken Seite tragen (sog. Inselanalyse, vgl. Abschnitt 4.7) oder die aus zwei aktiven Kanten bestehen (diese Konstellation tritt innerhalb von MILC nicht auf). Es kann sinnvoll sein, die Entscheidung zur Kombination von Kanten nicht anhand eines ausgezeichneten Knotens zu treffen, an dem sich beide treffen. So ist zum Beispiel für den Transfer von semantischen Beschreibungen von Intervallen des Sprachsignals nicht entscheidend, in welcher Oberflächenreihenfolge die einzelnen Kanten zueinander stehen. Relevant ist vielmehr, daß alle semantischen Inhalte einer Kante durch entsprechende Teilanalysen gefüllt sind. Das bedeutet, daß eine aktive Transferkante inaktive Kanten inkorporieren kann, die sie überspannt, unabhängig von deren genauer Position.[4]

Eine weitere Domäne nicht strikt zeitlich gebundener Kombination bildet die Generierung. Kay (1996) beschreibt die Oberflächengenerierung ausgehend von flachen semantischen Beschreibungen von Sätzen, die an die in Verbmobil gebräuchlichen minimalrekursiven Terme (Copestake et al., 1995; Dorna und Emele, 1996) erinnern. Das Ziel der Generierung ist dann, die Sequenz von Einzeltermen durch eine Oberflächenrepräsentation zu erklären. Jede Kante trägt hier als Domäne die Teilmenge von Indizes in die Sequenz, die zu ihrer Konstruktion benutzt wurden. Zwei Kanten können folglich nur dann kombiniert werden, wenn der Schnitt beider Domänen leer ist; das Resultat erhält als Domäne die Vereinigung der beiden Ausgangsdomänen.

Die Datenstruktur der Chart sowie die angedeuteten Prozeduren reichen für sich genommen jedoch noch nicht aus, eine konkrete Komponente zu erstellen, die eine sinnvolle linguistische Bearbeitung vornimmt. Zu diesem Zweck müssen zwei weitere Bestandteile hinzugefügt werden: Eine *Agenda* sowie eine *Bearbeitungsstrategie*.

Eine Agenda ist in diesem Zusammenhang stets ein Warteraum für Aufträge. Jeder Auftrag besteht darin, eine der Chart-Prozeduren mit bestimmten Daten auszuführen. Kay (1980) spricht hier von einem *Algorithmenschema*, das zwar letztendlich die Durchführung aller notwendigen Operationen garantiert, indem alle Aufträge in der Agenda schließlich ausgeführt werden. Die exakte Reihenfolge der einzelnen Aktionen ist jedoch nicht spezifiziert. Erst durch diese Festlegung wird aus dem Algorithmenschema ein Algorithmus, in dem der genaue Verlauf der Analyse festgelegt ist. In Abschnitt 4.6 werden wir einige der denkbaren Strategien beleuchten. Bezüglich der Mehr-Ebenen-Chart kann gesagt werden, daß sie ein globales Algorithmenschema realisiert. Jede einzelne der Komponenten für sich genommen kann allerdings mit unterschiedlichen konkreten Ausprägungen ausgestattet werden, je nach Erfordernis.

Innerhalb von MILC besitzt jedes Modul eine eigene Agenda. Diese ist zunächst global angelegt und wird in regelmäßigen Abständen abgearbeitet. Die Abstände richten sich dabei nach der jeweiligen Verarbeitungsstrategie. Im Regelfall wird immer dann, wenn ein neuer Knoten in die Chart eingefügt wird, ein Bearbeitungszyklus als abgeschlossen angesehen. Dies entspricht der inkrementellen Ausrichtung, die davon ausgeht, daß ein neuer Zeitabschnitt dann in Angriff genommen wird, wenn der vorhergehende vollständig bearbeitet wurde. Bevor also neue Aufträge erzeugt werden, sind die bereits für frühere Zeiten vorliegenden abzuschließen. Allerdings kann die Berücksichtigung von Angaben aus der Begründungsverwaltung dazu führen, daß dieses starre Schema teilweise durchbrochen wird.

Die Ursache dafür ist, daß die MILC-Komponenten grundsätzlich auf Strahlensuche basieren (Steinbiß, Tran, und Ney, 1994). Das bedeutet, daß nicht stets alle Aufträge, die sich auf einer Agenda befinden, ausgeführt werden. Vielmehr wird eine Grenzbewertung definiert, die als Kriterium dient, der Durchführung eines Auftrages “Erfolgschancen” zuzumessen. Alle Aufträge, deren Bewertung oberhalb der Schranke liegt, werden ausgeführt, alle anderen bleiben zunächst unberücksichtigt.[5]

[4] Eine frühere Version des hier vorgestellten Ansatzes (Amtrup, 1994b) basiert auf strukturellem Transfer aufgrund syntaktischer Kriterien; dort ist die oberflächennahe, an der exakten Struktur des Graphen orientierte, Verfahrensweise angemessen.

[5] Etliche Systeme verzichten vollständig auf die schlechtbewerteten Aufträge; dies ist hier nicht möglich.

Die Aufträge werden jedoch nicht nur global vorgehalten, sondern gleichzeitig speichert MILC für jede Kante die potentiell durchführbaren Aufträge.

Eine Neubewertung eines Auftrages ist immer dann möglich, wenn sich die akustische Bewertung einer Kante aufgrund neuer Worthypothesen ändert. Die aktualisierte Bewertung wird vom Worterkenner geliefert. Wir hatten bereits angedeutet, daß ebenfalls Angaben zur Begründungsverwaltung Bestandteil der für jede Kante vorgehaltenen Information sind. Gemäß dieser Daten muß die verbesserte Bewertung an alle Kanten vererbt werden, die aus der zur Worthypothese gehörenden Hyperkante hervorgegangen sind. Dies kann zur Folge haben, daß ein Auftrag, der einen der Nachkommen der Hyperkante als Bestandteil enthält, über die Schwelle gerät, die eine Bearbeitung noch gerechtfertigt hätte. Die Komponenten von MILC führen über diese Vorgänge Buch und initiieren bei Bedarf die Ausführung eines Auftrages, selbst wenn die betreffenden Kanten schon vor dem aktuellen Bearbeitungsabschnitt liegen.

Neben der akustischen Bewertung wird auch die Änderung von Start- und Endknotenmengen als Auslöser für die Vererbung von Informationen zwischen Kanten benutzt. Allerdings kann es hier nicht geschehen, daß der Agendamechanismus rückwärts in der Zeit operieren muß.

Eine weitere wesentliche Eigenschaft einer Mehr-Ebenen-Chart ist, daß die Graphenstruktur, die einer Verarbeitung gesprochener Sprache mit Hilfe von Wortgraphen zugrundeliegt, sehr lange beibehalten werden kann. Jede uns bekannte Implementation eines NLP-Systems gibt diese zu einem bestimmten Zeitpunkt der Bearbeitung auf. Irgendwann entscheidet sich das System für eine Lesart und verzichtet normalerweise ab dann auf eine graphenartige Darstellung. In den bisherigen Prototypen von Verbmobil (Bub, Wahlster, und Waibel, 1997) befand sich diese Grenze recht früh im Verarbeitungsstrom, nämlich nach der syntaktischen Strukturanalyse (Block, 1997). Der architektonische Prototyp INTARC (Görz et al., 1996) verschob diese Grenze bin vor den Transfer. Mit Hilfe von Mehr-Ebenen-Charts ist es im Rahmen von MILC möglich, die Graphenstruktur bis in die Oberflächengenerierung hinein aufrechtzuerhalten. Das hat zur Folge, daß Sequenzen von Oberflächenkonstrukten als Ausgabe präsentiert werden können. Der Benutzer kann dies nur insofern zur Kenntnis nehmen, als daß der Generierungsprozeß selbst inkrementell abläuft und sich zeitlich ändernde Äußerungsteile ausgegeben werden. Eine ähnliche Strategie verfolgt Finkler (1996), dessen Generator die semantische Repräsentation des zu verbalisierenden Inhalts inkrementell verarbeitet. Hier sind Reparaturphänomene und Neuanfänge zu beobachten.

Die vorhergehenden Ausführungen lassen schließlich die (halb-) formale Definition einer Mehr-Ebenen-Chart zu. Wir lassen allerdings sämtliche Angaben zur Durchführung einer Analyse unberücksichtigt.

Definition 4.2 (Mehr-Ebenen-Chart)
Eine Mehr-Ebenen-Chart ist ein gerichteter, azyklischer, linksverbundener Hypergraph[6] *mit Kantenbezeichnungen. Jede Kante trägt als Information:*

[6] Man beachte, daß dadurch u.a. Knoten mit Zeitpunkten assoziiert werden können. Dies wird später (s. Abschnitt 4.8) relevant werden.

- *Eine eindeutige Identifikation, die ein Paar aus der Identifikation der erzeugenden Komponente und einer lokal eindeutigen Identifikation ist,*
- *eine Menge von Anfangsknoten (resultierend aus der Hypergrapheneigenschaft),*
- *eine Menge von Endknoten (resultierend aus der Hypergrapheneigenschaft),*
- *eine Beschreibung (in unserem Falle i.d.R. eine Merkmalstruktur),*
- *eine Bewertung, die prinzipiell aus mehreren Teilaspekten bestehen kann (z.B. akustischer Bewertung und einer Bewertung durch ein Sprachmodell),*
- *die maximal zwei Hyperkanten, aus denen die Kante konstruiert wurde, und*
- *eine lokale Agenda, auf der die Aufträge abgelegt werden, die aufgrund der Strahlensuche nicht sofort ausgeführt wurden.*

4.2 Kommunikation innerhalb der Anwendung

Mehr-Ebenen-Charts stellen eine für die inkrementelle Sprachverarbeitung überaus sinnvolle und nützliche Datenstruktur zur Verfügung. Damit allerdings eine solch komplexe Struktur erfolgreich verteilt werden kann, benötigt man effiziente, an den Notwendigkeiten der Aufgabe orientierte Kommunikationsmechanismen. Auf der einen Seite sollte die Infrastruktur abstrakt genug spezifiziert sein, so daß die auf unteren Ebenen liegenden Kommunikationsprimitive verschattet sind; gleichzeitig muß allerdings auch die Möglichkeit bestehen, Optionen des Kommunikationsverhaltens zu kontrollieren.

Die Realisierung der Infrastruktur sollte auf einem durchdachten theoretischen Fundament aufbauen und gleichzeitig so unabhängig wie möglich von den tatsächlichen Modulimplementationen sein. Auf keinen Fall ist es ratsam, bei jeder neu entstehenden Schnittstelle von neuem über die günstigste Art der Realisierung zu entscheiden; vielmehr muß der kommunikationstechnische Rahmen bereits vor Beginn der Implementierung der Anwendung konzipiert sein. Dies gilt prinzipiell für jede KI-Anwendung nichttrivialer Größe, trifft jedoch in ganz besonderem Maße auf MILC zu, da es sich hierbei im ein komplexes, verteiltes System mit etlichen Komponenten und Verbindungen zwischen ihnen handelt. Das im Rahmen der vorliegenden Arbeit entwickelte Kommunikationssystem ICE (*I*ntarc *C*ommunication *E*nvironment, Amtrup, 1994a) stellt einen Vertreter solchermaßen angemessener infrastrukturellen Subsysteme dar. Es wurde erfolgreich in mehreren Projekten zur Sprachverarbeitung verwendet, u.a. für alle bisherigen Prototypen von Verbmobil (Amtrup und Benra, 1996) sowie für den architektonischen INTARC-Prototypen (Amtrup, 1997a). Viena (Wachsmuth und Cao, 1995) stellt ein Beispiel für den Einsatz außerhalb des engen Bereiches der Sprachübersetzung dar, es handelt sich hier um ein multimodales System, das intelligente Kommunikation mit einem System für virtuelle Realität ermöglicht.

Die theoretische Grundlage für ICE bildet das Kanalmodell CSP von Hoare (1978). CSP (*C*ommunicating *S*equential *P*rocesses) definiert den Aufbau und die Semantik von Punkt-zu-Punkt Verbin-

dungen zwischen Prozessen. Eine erfolgreiche Implementierung des CSP-Modells liegt mit dem Transputer (Graham und King, 1990) als Hardware und der für diese Prozessoren entwickelten Programmiersprache Occam (Burns, 1988) vor. Kanäle sind bidirektionale Verbindungen zwischen Occam-Prozessen, auf denen Nachrichten ausgetauscht werden können. Eine derartige, nachrichtenorientierte Art der Verbindung ist die für den vorliegenden Zweck sinnvollste Art der Kommunikation. Die Verwendung von gemeinsam benutztem Speicher (*shared memory*) scheidet bereits aus weiter oben erwähnten Gründen (Busüberlastung, Serialisierung von Zugriffen) aus. Der Aufruf entfernt vorgehaltener Prozeduren (*remote procedure calls*) ist ungeeignet, da es sich hierbei um ein Kommunikationsmodell mit Rendez-Vous Synchronisation handelt, das aufgrund von Netzwerklatenzen mit unakzeptabel hohen Wartezeiten verbunden sein kann. Aus diesem Grunde wurde ebenfalls die ursprünglich durch Hoare (1978) vorgelegte Modellierung modifiziert. CSP benutzt eine Rendez-Vous Synchronisation, bevor Daten ausgetauscht werden. In der Konzeption von ICE haben wir davon abgesehen und verwenden statt dessen asynchronen Nachrichtenaustausch zur Kommunikation. Die zweite Modifikation, die wir vorgenommen haben, betrifft die Konfiguration von Kanälen, die untereinander nicht gleichwertig sind (s.u.).

4.2.1 Kommunikationsarchitektur einer Anwendung

Die Facette der Gesamtarchitektur einer Anwendung, die Kommunikationsaspekte betrifft, ist in Abb. 4.3 gezeigt. Eine Anwendung, die ICE benutzt, kann aus einer beliebigen Anzahl von Komponenten bestehen. Diese können im Prinzip in unterschiedlichen Programmiersprachen realisiert sein. Die unterstützten Sprachen sind C, C++, Lisp (Allegro, Lucid, CLISP, Harlequin), Prolog (Quintus, Sicstus), Java sowie Tcl/Tk.[7] MILC macht von dieser Heterogenität allerdings keinen Gebrauch, da es vollständig in C++ implementiert ist.[8]

Jegliche Kommunikation zwischen Komponenten findet über Kanäle statt, also bidirektionale, asynchron betriebene Punkt-zu-Punkt Verbindungen. Die zugrundeliegenden Primitive sind dabei nicht von Grund auf neu implementiert worden. Vielmehr wurde auf PVM (*P*arallel *V*irtual *M*achine, Geist et al., 1994) zurückgegriffen, ein Nachrichtensystem, das ausgesprochen weite Verbreitung gefunden hat. Es ist dazu in der Lage, ein heterogenes Netz von Workstations als eine umfangreiche virtuelle Maschine agieren zu lassen. Das bedeutet unmittelbar, daß auch auf ICE basierende Systeme heterogen verteilt werden können. In der Praxis ist dies für Maschinen der Firmen Sun (Solaris), HP (HPUX), IBM (AIX), SGI (Irix) und für PCs (Linux) gemacht worden. Der entscheidende Vorteil einer bereits vorhandenen Kommunikationsschicht niederer Ebene ist jedoch der weitgehende Ausschluß von Fehlern, welche diese Ebene betreffen. Frühere, desillusionierende Erfahrungen mit einem Vorgängersystem (Pyka, 1992a) haben die Notwendigkeit einer stabilen Kommunikationsform deutlich gezeigt.

Oberhalb der PVM-Schicht befindet sich die Kernebene der Kommunikationssoftware (ICE in Abb.

[7] Die Anbindungen für CLISP und Java wurden dankenswerterweise von Volker Weber implementiert.

[8] Die einzige Ausnahme hiervon bildet die Visualisierung (vgl. Abschnitt 4.10). Allerdings sind auch dort die Kommunikationsmechanismen in der Tcl unterliegenden C-Schicht untergebracht.

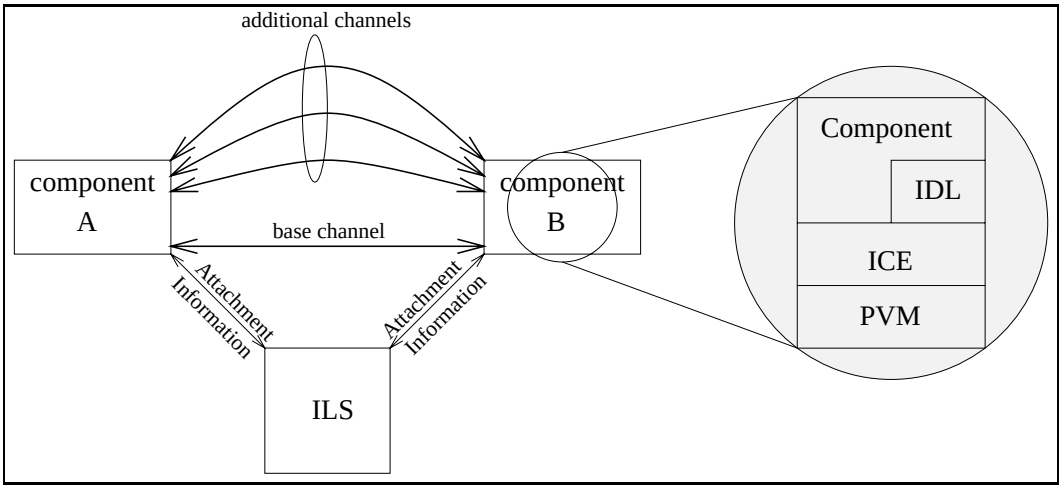


Abbildung 4.3: Prinzipielles Layout von Komponenten

4.3). Sie enthält die notwendigen Funktionen für das An- und Abmelden von Komponenten, das Versenden und Empfangen von Nachrichten sowie eine Reihe von Verwaltungsprozeduren, die das allgemeine Verhalten einer Komponente bestimmen. Außerdem sind hier die Schnittstellen zu den möglichen Programmiersprachen untergebracht. Der oberste, optionale Satz von Funktionen (IDL, *I*ntarc *D*ata *L*ayer) betrifft die Behandlung komplexer Datentypen. Wie bereits erwähnt, können benutzerdefinierte Datenobjekte transparent von ICE vermittelt werden. Dazu ist lediglich die Programmierung von Funktionen zur En- und Dekodierung notwendig, die innerhalb von ICE registriert werden.[9] Der hauptsächliche Gebrauch dieser Schicht besteht innerhalb von MILC darin, daß fast ausschließlich Kanten versendet werden.

Eine spezialisierte Komponente (ILS, *I*ntarc *L*icense *S*erver) operiert als Konfigurationszentrum. Jede Komponente, die Teil einer Anwendung ist, meldet sich beim ILS an und nach Beendigung ihrer Funktion auch wieder ab. Darüberhinaus verwaltet das ILS die Konfiguration aller in der Anwendung benutzten Kanäle. Das ILS stellt jedoch keinen zentralen Datenspeicher dar und übernimmt auch keine globalen Kontrollfunktionen. Nach der Etablierung von Kanälen verläuft die Kommunikation strikt zwischen den jeweils betroffenen Komponenten, ist also streng dezentral.

[9] Ursprünglich war geplant, eine abstrakte Sprache zu definieren, die einen gemeinsamen Satz von Datenprimitiven definiert, die in allen Programmiersprachen zugänglich sind (Pyka, 1992b). Davon ausgehend, sollten Compiler implementiert werden, die die notwendigen Funktionen generieren. Dies erwies sich allerdings als zu restriktiv, so daß letztlich nur die Compiler für C und C++ realisiert wurden.

4.2.2 Kanalmodelle

Der Austausch von Nachrichten zwischen Komponenten findet auf Kanälen statt. Kanäle können in zwei unterschiedlichen Ausprägungen auftreten: *Basiskanäle* und *zusätzliche Kanäle*. Basiskanäle (*base channels*) fungieren als allgemeines Medium für den Transport von Daten zwischen beliebigen Komponenten. Es ist keine bestimmte Absprache oder Lokalisation von Modulen erforderlich. Zusätzliche Kanäle (*additional channels*) hingegen können in mancher Hinsicht konfiguriert werden, z.B. um durch den Verzicht auf eine Hardwarearchitektur-unabhängige Datenkodierung (XDR, vgl. Corbin, 1990) den Durchsatz zu erhöhen. Außerdem ist es auf diese Weise möglich, den Informationsfluß zwischen Komponenten logisch aufzuteilen, um etwa die Separierung von Daten- und Kontrollströmen zu erreichen. Beide Typen von Kanälen sind dazu vorbereitet, sowohl skalare als auch benutzerdefinierte Datentypen tragen. Auch in dieser Weise erinnern sie an Occam-Kanäle, die gleichfalls mit einem Kanalprotokoll ausgerüstet werden können. Interessanterweise wurde aber in Occam diese Art von Datentypen (Verbünde, also *Records*) nicht in die Definition der Sprache aufgenommen, so daß derartige Strukturen zwar verschickt, aber nicht ohne weiteres innerhalb eines Prozesses verwendet werden können.

Kanäle können auf eine weitere, zusätzliche Art und Weise konfiguriert werden: Sie können aufgeteilt werden. Sie fungieren nicht länger als Verbindungen ausschließlich zwischen genau zwei Komponenten. Statt dessen kann definiert werden, daß eine ganze Anzahl von Komponenten Nachrichten, die auf einem Kanal gesendet werden, empfangen sollen. Ebenso ist die Definition multipler potentieller Quellen für eine Nachricht möglich. Die Definition von geteilten Kanälen (*Split Channels*) hat dabei im wesentlichen zwei Motivationen:

- Die Realisierung von Visualisierungskomponenten für die Daten, die auf einem Kanal versendet werden. Dadurch wird zum einen die Fehlersuche stark vereinfacht, zum anderen können partielle Ergebnisse auf einfache Weise dem Benutzer angezeigt werden.
- Die Umwandlung von Datenformaten zwischen zwei Komponenten. Diese Art der Insertion eines Daten modifizierenden Moduls zwischen zwei anderen kann insbesondere dann nützlich sein, wenn Versionsübergänge auftreten, die eine Änderung des Datenformats mit sich bringen.[10]

Abb. 4.4 zeigt ein einfaches Beispiel für eine Konfiguration mit geteilten Kanälen. Zwei Komponenten, **A** und **B**, sind durch einen Kanal miteinander verbunden, hier angedeutet durch eine gestrichelte Linie. Die Kanalendpunkte (in der Abbildung durch graue Kästchen symbolisiert) werden voneinander separiert, um eine Visualisierung der von jeder Komponente gesendeten Daten zu ermöglichen. Diese Funktion wird durch die beiden zusätzlichen Komponenten **UI_A** und **UI_B** wahrgenommen. Außerdem unterlaufen die Nachrichten, die von **A** abgesendet werden, einer Modifikation, bevor sie bei **B** ankommen. Zu diesem Zweck ist die zusätzliche Komponente **C** konfiguriert worden, die in

[10] Aufgrund der Integriertheit von MILC war eine solche Komponente nie notwendig. In anderem Zusammenhang hat sich diese Art des Einsatzes von *split channels* jedoch als ausgesprochen nützlich erwiesen.

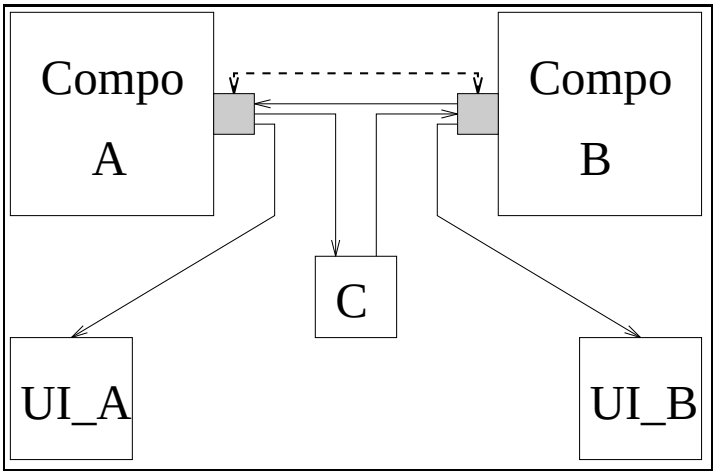


Abbildung 4.4: Konfiguration von *Split Channels*

den Datenpfad zwischen den beiden Ausgangskomponenten eingesetzt wird. Nachrichten hingegen, die von **B** nach **A** gesendet werden, bedürfen keiner Modifikation.

Die Konfiguration von geteilten Kanälen ist vollständig transparent für Komponenten. Das Verhalten wird sich folglich nicht lediglich durch die Aufteilung ändern, obwohl natürlich die in einen Datenpfad eingeschleusten Komponenten dazu in der Lage sind. Mit Hilfe dieses Modells ist die *Breadboard*-Architektur von Komponenten für Verbmobil entwickelt worden (Bub, Wahlster, und Waibel, 1997).

4.2.3 Informationsservice und Synchronisation

Das ILS übernimmt drei Funktionen, die für den Lauf der Verarbeitung in einer verteilten Anwendung wesentlich sind:

- Die Verwaltung der Namen von Komponenten
- Die Konfiguration der Gesamtanwendung
- Die initialen Einstellungen für das Verbreiten von Nachrichten an alle Komponenten (*broadcasting*)

Lediglich das ILS besitzt normalerweise ein vollständiges Bild der Struktur der Anwendung, während einzelne Komponenten nur Informationen über den für sie relevanten Teil haben. Zu diesem Zweck speichert das ILS eine Tabelle mit den Namen und Standorten der Module.

Teil der Initialisierungssequenz eines Moduls ist folglich die Anmeldung beim ILS. Dort wird die Anmeldung registriert und die Identität der Komponente an andere Module weitergegeben, die Kommunikationsbedarf mit ihr angemeldet haben. Der dazu notwendige Synchronisationsaufwand wurde möglichst klein gehalten (s.u.). Analog dazu sollte sich eine Komponente beim ILS abmelden, bevor sie terminiert. Aber selbst wenn eine Komponente aufgrund eines Fehlers abbricht, ohne sich abzumelden, kann das ILS (über PVM) davon erfahren. Auf diese Weise können Komponenten auf die Abwesenheit reagieren; sie werden dazu vom ILS darüber informiert.

Die zweite Hauptfunktion des ILS ist die Verwaltung der Konfiguration von Kanälen. Initial wird eine Konfigurationsdatei gelesen, welche die Konfiguration von geteilten Kanälen beschreibt. Die Konfiguration, die für das oben beschriebene Beispiel notwendig ist, hat die in Abb. 4.5 gezeigte Gestalt.[11]

```
A B BASE
UIG_A BASE -1 1 0
C BASE -1 1 0
B BASE -1 0 1

B A BASE
UIG_B BASE -1 1 0
A BASE -1 1 0
C BASE -1 0 1
```

Abbildung 4.5: Ein Beispiel einer Konfigurationsdatei für geteilte Kanäle

Im Laufe der Verarbeitung werden normalerweise zusätzliche Kanäle auf Anforderung von Komponenten geöffnet. Die einzige Einschränkung hierbei ist, daß geteilte Kanäle nicht dynamisch verwaltet werden können. Deren Konfiguration wird einmal zu Beginn des ILS-Laufes festgelegt.

Die dritte Funktion des ILS ist die erstmalige Information über an der Anwendung teilnehmende Komponenten, wenn das Verbreiten einer Nachricht an alle Komponenten gewünscht ist (*Broadcasting*). Im Normalbetrieb hat jede einzelne Komponente nicht notwendigerweise alle Information über die teilnehmenden Module. Dies ist aber notwendig, wenn eine Nachricht an alle Komponenten verteilt werden soll. Ein zentralistischer Ansatz, der an *Blackboards* erinnert, würde vorschreiben, jede solche Nachricht zunächst an den zentralen Prozeß (in unserem Fall: das ILS) zu senden und von dort aus zu verteilen. ICE hingegen benutzt die Verbindung zum ILS lediglich zum erstmaligen Aufsetzen dieser Funktion. Dies resultiert wiederum in einem geringen Synchronisationsaufwand und verhindert einen Kommunikationsengpaß am ILS.

Die Verbreitung von Nachrichten geschieht in drei Schritten:

[11] Für jeden Kanal wird eine Liste von tatsächlichen Verbindungen definiert, zusammen mit der Richtungsangabe dieser Verbindungen. Zur Syntax der Konfigurationsdateien vgl. Amtrup (1995b).

- Wenn eine Komponente zum ersten Mal eine Nachricht an alle anderen verbreiten will, sendet sie eine Anfrage an das ILS, um über die Identität aller anderen Komponenten informiert zu werden. Das ILS konsultiert dazu die ihm zur Verfügung stehende Konfiguration und sendet entsprechende Antworten an die anfragende Komponente.
- Nachdem diese alle notwendigen Daten erhalten hat, kann die Komponente die gewünschte Nachricht auf allen Kanälen zu anderen Komponenten senden.
- Im zukünftigen Ablauf nimmt das ILS an, daß die Komponente, die eine Nachricht verbreitet hat, dies auch in Zukunft wahrscheinlich tun wird. Deshalb wird beim Anmelden einer neuen Komponente in jedem Fall diese Information zur verbreitenden Komponente weitergegeben, unabhängig davon, ob die Anfrage zur Öffnung eines Kanals zwischen beiden vorliegt. Auf diese Art und Weise braucht die Kommunikation zwischen Komponente und ILS zur Vorbereitung eines Verbreitens von Nachrichten nur einmal durchgeführt zu werden.

Der Ablauf dieser Konfiguration ist vollständig transparent für die verbreitende Komponente. Sie erhält normalerweise nicht einmal den Hinweis, wieviele Komponenten teilnehmen. Dies ist in der ICE-Schicht verborgen und belastet somit nicht den Anwendungsteil des Programms.

Der interessanteste Teil der Kommunikation zwischen Komponenten und ILS ist zweifellos die anfängliche Synchronisation, in der die Identitäten der Module und die notwendigen Kanäle definiert werden. Ein Designziel bei der Erstellung von ICE war, auch hier den Synchronisationsaufwand möglichst gering zu halten. Das motiviert sich zum einen aus der Vermeidung eines Kommunikations-Engpasses beim ILS, andererseits kann aber ein Prozeß beliebig viele Komponenten enthalten, und hier kann eine langwierige Abstimmung mit dem ILS erhebliche Performanzeinbußen mit sich bringen.

Daraus folgt, daß eine Komponente lediglich dann Information über eine andere Komponente fordern sollte, wenn dies unbedingt notwendig ist: dann nämlich, wenn sie selbst eine Nachricht an diese andere Komponente schicken will. ICE implementiert ein Informationsprofil, das mit unterspezifizierten Komponenten- und Konfigurationsdaten auf seiten der Module umgehen kann. Abbildung 4.6 zeigt den Ablauf der Konfiguration vom Anmelden einer Komponente bis zum erstmaligen Nachrichtenaustausch. An den Rändern sind jeweils Komponenten gezeigt, die an der Anwendung teilnehmen, das ILS bildet den Mittelteil.

Zunächst meldet sich Komponente **A** beim ILS an und gibt ihren Namen bekannt. Dazu ruft sie die Funktion `ICE_Attach()` (oder einen Vertreter in einer der anderen unterstützten Programmiersprachen) auf und sendet damit eine Nachricht mit dem Etikett **ILS_ADD** an das ILS.[12] **A** wartet jedoch nicht auf eine Bestätigung, sondern fährt in seiner Bearbeitung fort.

[12] Nachrichten besitzen neben dem Inhalt und der Typinformation zusätzlich ein Etikett, um den Zweck der Nachricht zu kodieren. Die Empfangsfunktionen innerhalb von ICE können so parametrisiert werden, daß sie nur Nachrichten mit bestimmten Etiketten annehmen. **ILS_ADD** ist eines der vordefinierten Etiketten. Der Austausch dieser Systemnachrichten ist transparent für Komponenten. Der Empfang solcher Daten findet priorisiert immer dann statt, wenn ein Modul ICE-Funktionen benutzt.

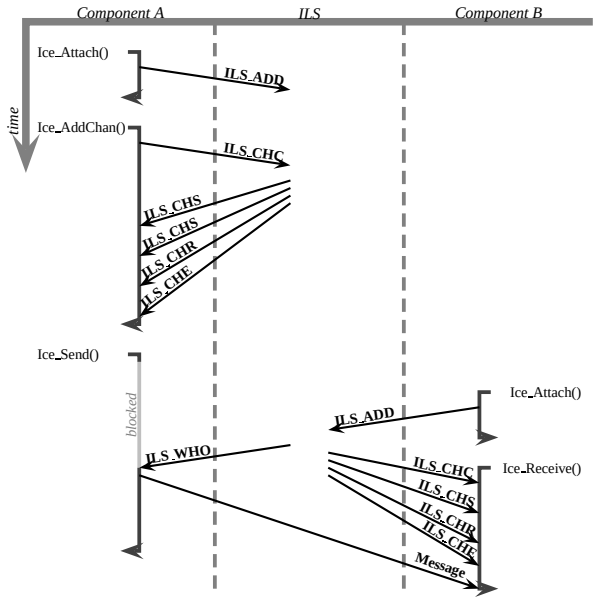


Abbildung 4.6: Der Ablauf der initialen Synchronisation von Kanälen

Anschließend führt **A** die Anforderung nach Öffnung eines Kanals aus, indem `Ice_AddChan()` aufgerufen wird. Eine entsprechende Nachricht mit dem Etikett **ILS_CHC** wird an das ILS geschickt, das durch Angabe der Konfigurationsdaten antwortet. Diese enthalten in Nachrichten mit dem Etikett **ILS_CHS** die echten Kanäle (*real channels*), die zum Versenden einer Nachricht benutzt werden sollen.[13] In diesem Fall sind dies zwei Instanzen. Nachrichten des Etiketts **ILS_CHR** definieren echte Kanäle, von denen Nachrichten empfangen werden sollen. Schließlich wird der Abschluß der Konfiguration mittels einer Nachricht des Etiketts **ILS_CHE** signalisiert.

Nach dieser Konfiguration versucht **A**, eine Nachricht an **B** zu senden, über den gerade konfigurierten Kanal. Da die Zielkomponente **B** noch nicht Teil der Anwendung ist, wird der Aufruf blockiert.[14] Nach einiger Zeit meldet sich die Komponente **B** beim ILS an. Sobald das ILS diese Information besitzt, informiert es die Komponente **A** über den Eintritt der Zielkomponente. Daraufhin kann **A** die Nachricht absetzen, und fährt mit der Bearbeitung fort. Das ILS sendet mittlerweile die für den Kanal notwendigen Konfigurationsdaten an **B**, selbst wenn **B** nicht die Öffnung des Kanals verlangt hat. Sobald **B** eine Empfangsfunktion von ICE (in diesem Fall `Ice_Receive()` aufruft, werden die Konfigurationsdaten zunächst verarbeitet. Danach kann die mittlerweile bei **B** angekommene

[13] Im Falle geteilter Kanäle kann dies mehr als ein Kanal sein. Wir verwenden den Term “echter Kanal”, um anzudeuten, daß auf diesem tatsächlich Nachrichten fließen, während der “Kanal” lediglich eine günstige Abstraktion ist.

[14] Dieses Verhalten ist kann geändert werden. Die Komponente kann entscheiden, nicht zu warten und stattdessen eine Fehlernachricht zu erhalten, wenn eine der betroffenen Zielkomponenten nicht angemeldet ist.

Nachricht normal verarbeitet werden.[15]

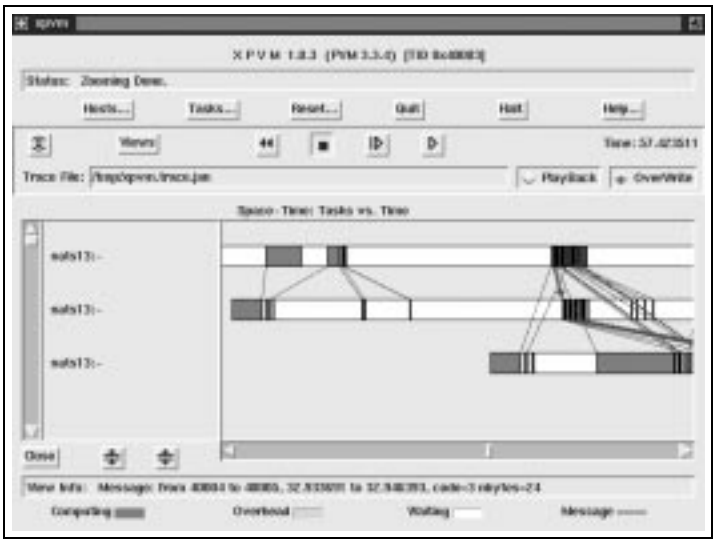


Abbildung 4.7: XPVM-Schnappschuß der initialen Synchronisation

Abbildung 4.7 zeigt einen Ausschnitt aus der tatsächlichen Kommunikation im Programmverlauf. Die Aufnahme erfolgte mit `XPVM`, einem graphischen Werkzeug zur Benutzung von PVM. Der oberste Balken stellt die Aktivität des ILS dar, der mittlere entspricht **A** in unserem Beispiel, der untere der Komponente **B**. Durch einen Pfeil markiert ist die Nachricht, die **A** vom Eintreffen von **B** informiert.

4.2.4 Terminierung

Die Terminierung einer Anwendung ist selbst nicht Bestandteil der Definition von ICE. In diesem Abschnitt werden wir jedoch kurz darlegen, wie die Terminierung innerhalb von MILC gehandhabt wird, da der Kommunikationsstatus eng mit ihr verknüpft ist. MILC ist nicht als fortwährend laufendes System konzipiert, sondern beendet sich nach der Verarbeitung einer Äußerung eigenständig (vgl. dazu Abschnitt 4.11). Daher besteht die Notwendigkeit zu erkennen, wann dieser Zeitpunkt erreicht ist.[16] Aufgrund von möglichen Rückkopplungen kann nicht einfach davon ausgegangen werden, daß die Anwendung einen unidirektionalen Informationsfluß zeigt, und Komponenten schlicht dann beendet werden, wenn sie ihre Eingabe vollständig bearbeitet haben. In diesem Fall müßte lediglich die erste Komponente (in unserem Fall der Worterkenner) einen Hinweis auf das Ende der Eingabe erhalten, was durch die Struktur seiner Eingabe gegeben ist (s.u., Abschnitt 4.4). In der gegenwärtigen Ausbaustufe ist es tatsächlich so, daß die Kommunikationsverbindungen innerhalb von

[15] Die Nachricht von **A** kann theoretisch auch vor der Konfigurationsinformation eintreffen. Sie wird solange gepuffert, bis **B** vollständig konfiguriert ist.

[16] Diese Notwendigkeit besteht übrigens auch für ein kontinuierlich laufendes System, nur daß in diesem Fall nicht die Anwendung beendet wird, sondern einige Datenstrukturen reinitialisiert werden müssen.

MILC einen zyklenfreien Graphen bilden (s. der folgende Abschnitt). Allerdings stellt die aktuelle Fassung u.E. ein Ausgangssystem dar, das in der Zukunft Grundlage für weitergehende Architekturuntersuchungen sein wird. Folglich ist für MILC eine verteilte Terminierung implementiert.

Das Terminierungprotokoll (vgl. Mattern, 1987; Peres und Rozoy, 1991) arbeitet mit Nachrichtenzählern. Diese halten fest, wieviele Nachrichten zwischen je zwei Komponenten ausgetauscht wurden, jede Komponenten verwaltet also pro Verbindung zwei Zähler. Das System kann dann terminieren, wenn die Komponenten paarweise über diese Anzahlen einig sind, d.h. wenn jede gesendete Nachricht auch beim Empfänger abgenommen wurde. Diese Konsistenzbedingung kann periodisch überprüft werden. Im Falle von MILC ist es jedoch sinnvoll, aus Effizienzgründen die Überprüfung der Terminierung nur dann einzuleiten, wenn der Worterkenner, der die Wurzel bildet, seine Arbeit beendet hat.

4.3 Architekturüberblick

Nach der Einführung in die Konzeption von Mehr-Ebenen-Charts und der Erläuterung der Softwareebene, die für die Kommunikation zwischen Komponenten sorgt, kann nunmehr detailliert auf die Anwendungsarchitektur von MILC eingegangen werden (Amtrup, 1998). MILC stellt eine erste Instanz eines Systems dar, das auf Mehr-Ebenen-Charts basiert. Einige aus architektonischer Sicht interessante Aspekte sind in der hier beschriebenen Konzeption aus Aufwandsgründen bisher nicht berücksichtigt worden, die Architektur ist jedoch so ausgelegt, daß die Integration neuer Methoden und das Hinzufügen von Komponenten ohne prinzipielle Probleme vonstatten gehen kann. Die grobe Architektur von MILC ist in Abb. 1.2 gezeigt, die wir hier als Abb. 4.8 wiederholen.

Wie bereits angedeutet, symbolisiert die graphenartige Struktur in der Mitte der Abbildung den Gebrauch einer Mehr-Ebenen-Chart durch die Module des Systems. Daraus folgt, daß stets ein konsistenter Gesamtzustand des Systems verfügbar ist, beschrieben als die Vereinigung aller im System existierenden Kanten. Zu Demonstrationszwecken, und um eine unvermeidliche Fehlersuche zu erleichtern, ist eine Visualisierung implementiert worden, welche die Kanten der Mehr-Ebenen-Chart graphisch darstellen kann. Sie ist in Abschnitt 4.10 beschrieben, in Abb. 4.8 jedoch nicht gezeigt. Die Pfeile in der Abbildung stehen für ICE-Kanäle, die Richtung der Pfeile gibt die Richtung des Informationsflusses an.

Die Wurzelkomponente der Anwendung wird durch die **Worterkennung** gebildet. Wir verwenden Wortgraphen, die vom Hamburger Worterkenner (Huebener, Jost, und Heine, 1996) inkrementell erzeugt wurden, also linksverbundene Wortgraphen. Der tatsächliche Erkennungsprozeß ist jedoch nicht in MILC integriert worden, da mit der Erkennung ein beträchtlicher Zeitaufwand verbunden ist. Insofern ist die Bezeichnung “Worterkenner” für die MILC-Komponente nicht vollständig korrekt. Die Funktion besteht in dem Lesen eines Wortgraphen aus einer Datei und der inkrementellen Konvertierung in einen Hypergraphen.

Für die syntaktisch-semantische Analyse sind drei Komponenten vorgesehen, **Idiomerkennung**, **Partielle Strukturanalyse** sowie **Äußerungsintegration**, auf die wir genauer in den folgenden

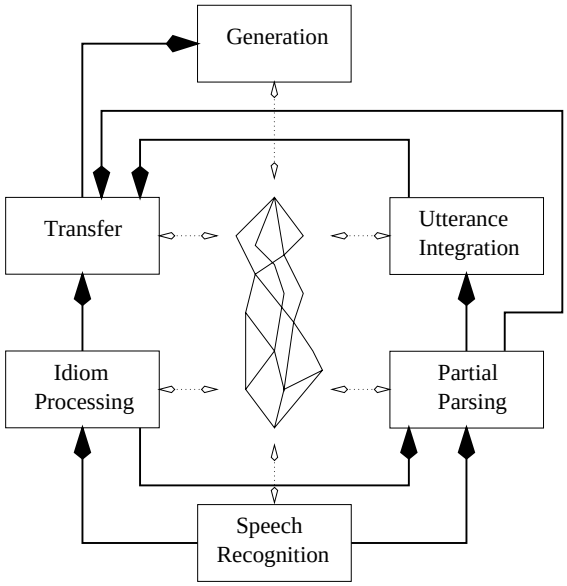


Abbildung 4.8: Das Grobmodell der Architektur von MILC

Abschnitten eingehen werden.

Der **Transfer** sorgt für die Abbildung der semantischen Beschreibung deutscher Äußerungen in eine entsprechende englische Semantik. Auch diese Phase ist Chart-gesteuert, basiert jedoch, anders als in einer Voruntersuchung gezeigt (Amtrup, 1995a), nicht auf syntaktischen Kriterien.

Schließlich wird die (englische) semantische Beschreibung einer **Oberflächengenerierung** unterworfen, deren Ausgabe in einer Sequenz von potentiellen englischen Äußerungsteilen besteht, die den besten Pfad durch den Generierungsgraphen darstellen.

4.4 Worterkennung

Die für Worterkennung verantwortliche Komponente erstellt aus einem linksverbundenen Wortgraphen inkrementell einen linksverbundenen Hypergraphen und gibt die Hyperkanten an die Idiomverarbeitung sowie die partielle Strukturanalyse weiter. Als Beispiel und zur Illustration des verwendeten Formates diene der Wortgraph in Abb. 4.9.

Jede Worthypothese besteht aus einem abstrakten Anfangs- und Endknoten, der orthographischen Transkription des erkannten Wortes, einer akustischen Bewertung (der Logarithmus einer Dichte, der angibt, wie gut das angegebene Segment des Eingabesignals mit einem Modell des hypothe-

tisierten Wortes übereinstimmt; ein kleiner absoluter Wert bedeutet gute Übereinstimmung), sowie der Angabe des überspannten Intervalls des Eingabesignals (Anfangs- und Endzeitpunkt in *Frames*). Worthypothesen sind nach Endzeitpunkten sortiert, sowie innerhalb derer nach Anfangszeitpunkten.

```
%TURN: n002k000
BEGIN_LATTICE
...
1  38 <sil>  -2239.464600 1  38
1  42 <sil>  -2570.498779 1  42
1  46 <sil>  -2914.757568 1  46
1  52 <sil>  -3504.763428 1  52
38 52 okay   -1290.138916 38 52
42 52 sch"on -988.619324  42 52
38 53 okay   -1354.200317 38 53
42 53 sch"on -1059.124756 42 53
46 53 wir    -718.457947  46 53
38 54 okay   -1430.380493 38 54
42 54 sch"on -1127.405151 42 54
46 54 wir    -791.802673  46 54
40 54 Herr   -1312.388672 40 54
38 55 okay   -1516.757080 38 55
42 55 sch"on -1181.626709 42 55
46 55 wir    -874.704224  46 55
42 55 schon  -1215.008667 42 55
...
END_LATTICE
```

Abbildung 4.9: Ein Wortgraph im Verbmobil-Format

Die Operation des Worterkenners besteht nun darin, diesen Graphen zu lesen und die unterste Ebene einer Mehr-Ebenen-Chart zu erzeugen. Der Algorithmus dazu wurde bereits als Alg. 12 auf S. 61 definiert. Der Worterkenner geht initial von einem leeren Hypergraphen aus und fügt sukzessive eine Worthypothese nach der anderen in den Hypergraphen ein. Dabei ist an zwei Punkten die Kommunikation mit der Idiomerkennung und der partiellen Strukturanalyse notwendig:

- Immer dann, wenn eine neue Hyperkante erzeugt wird (Zeile [5] des Algorithmus 12), wird dies bekanntgegeben.
- Immer dann, wenn sich die Repräsentation einer Hyperkante ändert (Zeile [2] des Algorithmus 12), wird dies ebenfalls bekanntgegeben. Diese Änderung kann darin bestehen, daß ein Knoten der Menge der Anfangs- bzw. Endknoten der Hyperkante hinzugefügt wurde, oder aber darin, daß sich die akustische Bewertung der Kante verbessert hat (vgl. Weber, 1995, S. 78).

Die Eingabe des in Abb. 4.9 dargestellten Wortgraphen resultiert folglich in einer Reihe von Nach-

richten. Diese sind schematisch in Tab. 4.1 aufgezeigt.

Tabelle 4.1: Nachrichten vom Worterkenner an folgende Komponenten

Nr.	Art	Inhalt	bezogene Hyperkante
...			
1	Neue Hyperkante	$\mathcal{H}1$:<1–38(sil),2239>[17]	$\mathcal{H}1$
2	Neuer Endknoten	42	$\mathcal{H}1$
3	Neuer Endknoten	46	$\mathcal{H}1$
4	Neuer Endknoten	52	$\mathcal{H}1$
5	Neue Hyperkante	$\mathcal{H}2$:<38–52(okay),1290>	$\mathcal{H}2$
6	Neue Hyperkante	$\mathcal{H}3$:<42–52(schön),988>	$\mathcal{H}3$
7	Neuer Endknoten	53	$\mathcal{H}2$
	Neue Bewertung	1354	$\mathcal{H}2$
8	Neuer Endknoten	53	$\mathcal{H}3$
	Neue Bewertung	1059	$\mathcal{H}3$
...			

4.5 Idiomverarbeitung

Feste Wendungen der Art "tut mir leid" etc. sind kompositionell normalerweise schwer zu analysieren.[18] Das Ziel muß daher sein, diese quasi lexikalisch zu behandeln und der Wendung als Ganzem eine Analyse zuzuordnen. Der üblicherweise verwendete Ansatz ist, entweder lexikalische Einträge zuzulassen, die mehr als ein Wort umfassen, oder syntaktische Regeln zu formulieren, die spezialisierte Subkategorisierungsrahmen für die Wendung aufspannen. Der Nachteil beider Verfahren ist jedoch stets, daß das Analyseverfahren nicht spezialisiert ist, d.h., feste Wendungen werden analog zu anderen syntaktischen Konstruktionen behandelt. Insbesondere wird nicht vermieden, daß trotz des nichtkompositionalen Charakters der Wendung eine syntaktische Analyse der Einzelbestandteile versucht wird. Die Lösung, die wir für MILC favorisieren, besteht deswegen darin, eine spezialisierte Komponente für Idiome vorzusehen, mit deren Hilfe dieser Mehraufwand vermieden wird.

Für die Domäne Terminvereinbarung scheinen Idiome eine wichtige Rolle zu spielen. Sie sind hochfrequent (Schöllhammer (1997) findet einen Anteil von über 30% Äußerungen mit Idiomen) und tragen stark zur Realisierung der kommunikativen Ziele bei, indem sie Dialogakte realisieren (zu Dialogakten vgl. Jekat et al., 1995).

[17] Die angegebene akustische Bewertung (2239) gilt für den Abschnitt, der von der Kante überspannt wird. Als Vergleichskriterium für die Modifikation von Bewertungen von Hyperkanten wird ein zeitlich normalisiertes Maß herangezogen.

[18] Für eine Klassifikation von phraseologischen Wendungen vgl. Schöllhammer (1997) und die dort zitierte Literatur, insbesondere Palm (1995).

Durch deren getrennte Betrachtung kann eine schnelle inkrementelle Suche nach Idiomen implementiert werden, die lediglich an der orthographischen Form der Oberflächenrepräsentation orientiert ist und keinerlei komplexe linguistische Operationen erfordert. Dazu wird neben dem Hypergraphen, der aus der Worterkennung resultiert, gleichzeitig ein Graph erkannter Präfixe von Idiomen konstruiert. Prinzipiell wird dies erreicht, indem zu Programmanfang eine Menge von Idiomdefinitionen der in Abb. 4.10 angegebenen Art gelesen wird. Der Wert des phon-Merkmals besteht dabei aus einer Liste von Wörtern, die das Idiom ausmachen. Anhand dieser Wörter wird ein Suchbaum erzeugt, dessen Kanten mit Teilwörtern von Idiomen markiert sind. An den Knoten des Baumes werden Merkmalstrukturen annotiert, falls ein Pfad von der Wurzel des Baumes zu dem jeweiligen Knoten ein Idiom konstituiert.

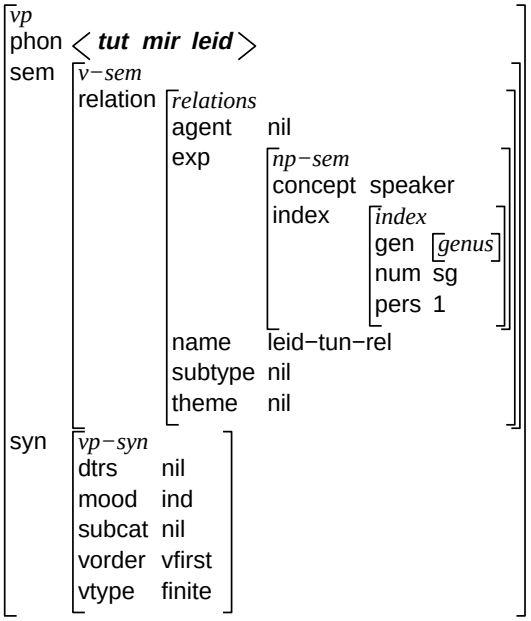


Abbildung 4.10: Idiomdefinition für die Teiläußerung “tut mir leid”

Während der Operation des Moduls wird laufend versucht, einen Schnitt zwischen dem Definitionsbaum und Teilgraphen der Worterkennung zu bilden. Dazu werden im Worthypergraphen Markierungen an Knoten gesetzt, die angeben, ob und welcher Teil welches Idioms an einem bestimmten Knoten endet. Ausgehend von dieser Information kann stets die Vervollständigung einer Wendung geprüft werden, in welchem Falle eine entsprechende Nachricht verbreitet wird.

Die Information über ein gefundenes Idiom besteht dabei aus zwei Komponenten. Zunächst wird natürlich die Merkmalstruktur, die das Idiom repräsentiert, an die partielle Strukturanalyse und den Transfer weitergegeben, damit diese in der Lage sind, die Wendung in ihre Analysen einzuführen. Gleichzeitig werden aber auch inhibitorische Nachrichten ausgesendet. Das Motiv dafür besteht darin, daß möglichst versucht werden soll, die normale, kompositionale Analyse der dem Idiom zugrundeliegenden Worthypothesen zu vermeiden. Die Hintergrundannahme ist, daß wahrscheinlich

die Worthypothesen, die Bestandteil eines Idioms sind, für die Standard-Analyse nur einen begrenzten Wert besitzen.

Wesentlich ist jedoch, daß die betreffenden Worthypothesen nicht eliminiert werden. Dies könnte zu Fehlern führen, etwa für den Fall, daß die Sequenz von Worthypothesen innerhalb einer richtigen Analyse zu zwei wohlgeformten Konstituenten gehört. Deswegen wird lediglich dafür gesorgt, daß die Bewertung der Worthypothesen modifiziert wird, sie erhalten folglich eine "Strafe" dafür, Bestandteil eines Idioms zu sein, die sich auf die Wahrscheinlichkeit der Integration in einen anderen syntaktischen Kontext negativ auswirkt. Die Art der Verbreitung und Vererbung von inhibitorischen Nachrichten wurde bereits weiter oben erläutert.

4.6 Strukturanalyse

Um eine Analyse natürlichsprachlicher Äußerungen durchzuführen, wird in den meisten modernen Systemen eine komplette syntaktisch-semantische Analyse vorgenommen. Ansätze zum partiellen Parsing[19] lassen sich bisher fast ausschließlich im Bereich der Informationsextraktion (*information extraction*) z.B. in den MUC-Systemen (*Message Understanding Conference*) finden; allerdings verspricht die zweite Phase des Verbmobil-Projektes eine weitergehende Verwendung des partiellen Strukturanalyse-Paradigmas (Light, 1996; Worm und Rupp, 1998; Worm, 1998), das in der Form der Reevaluation von Erkennungsbewertungen, die mit Hilfe von Lizensierungen eines *Chunk-Parsers* durchgeführt wird, auch an anderer Stelle Einzug hält (Zechner und Waibel, 1998). Wir werden die monolithische Aufgabe der Konstruktion einer grammatischen Struktur aus Eingabeäußerungen hier ebenfalls aufteilen. Zunächst soll mit Hilfe einer hauptsächlich syntaktisch orientierten Grammatik eine Menge von partiellen Analysen gefunden werden, etwa auf Basis der syntaktischen Kategorien Nominalphrase, Präpositionalphrase, Adverbialphrase etc. Was wir nicht durchführen werden, ist die Strukturbildung für Subkategorisierungsinformation, wie sie bei Verben vorliegt. Desgleichen werden wir auf Anbindungsphänomene verzichten, also z.B. das *Attachment* von Präpositionalphrasen zunächst nicht auflösen. Derartige Ableitungsschritte sind einer zweiten Analysephase vorbehalten, die auf den Ergebnissen des ersten Schrittes arbeitet. Sie hat detailliertere Information zur Verfügung, um Verbkomplexe modellieren zu können.

Die Motivation für einen solchen Ansatz beruht auf Erfahrungen, die während eines Versuchs mit einem spontansprachlichen Dialog (Dialog N002K aus dem Verbmobil-Datenbestand) gewonnen wurden. Wir haben für die Beschreibung dieses Dialogs die notwendigen linguistischen Wissensquellen erstellt, die eine konventionelle, auf Satzkonstituenten beruhende, Analyse erlauben. Innerhalb des bereits beschriebenen Formalismus entstand dazu eine Typenhierarchie mit 345 Typen, die hauptsächlich syntaktische Information enthielt. Die semantischen Anteile beschränken sich auf die Zuweisung von Konzepten, um in bestimmten Fällen Selektionsbeschränkungen formulieren zu

[19] Hier wird unter partiellem Parsing verstanden, daß die zur Verfügung stehende Eingabeäußerung nicht in eine vollständige Analyse eingebunden wird, sondern daß eine Menge von Analysen konstruiert wird, deren Elemente jeweils einen zeitlichen Teilausschnitt der Eingabe beschreiben. Manchmal wird dieses Verfahren auch *Chunk-Parsing* genannt (Abney, 1991; Abney, 1996).

können. Das Lexikon für den Versuch bestand aus 413 Lesarten für Wortformen, die Grammatik besaß 80 Regeln. Ein Teil der Grammatik ist spezialisiert auf Datums- und Zeitausdrücke, die in den zur Verfügung stehenden Dialogen gehäuft vorkommen.

Die Wissensquellen wurden anhand der Transliterationen der Äußerungen des Dialogs N002K erstellt, sie sind aber — abgesehen vom Lexikon — nicht auf diesen spezialisiert. Nach Abschluß der Entwicklungsarbeiten wurden Experimente mit inkrementell erzeugten Wortgraphen durchgeführt. Die Standardeinstellungen des Parsers wurden verwendet. Obwohl dieser Versuchsaufbau natürlich keine allgemeingültigen Aussagen erlaubt, da das Testmaterial sowohl zur Erstellung der Grammatik benutzt wurde, als auch beim Training des Sprachmodells, kann gezeigt werden, daß der vollständige Parsing-Ansatz für spontane Sprache unter den gegebenen Umständen nicht adäquat ist.

Es lassen sich einige prinzipielle Gründe festmachen, die gegen eine vollständige Analyse von Äußerungen sprechen und für einen partiellen oder gestaffelten Ansatz. Die beiden folgenden Abschnitte diskutieren einerseits die unnötige Konstruktion von Komplementkomplexen, andererseits Eigenschaften spontaner Sprache und Wortgraphen, die den vorliegenden Ansatz sinnvoll erscheinen lassen.

4.6.1 Ableitung von Verbkomplexen

Die herkömmliche Art der Ableitung von Subkategorisierungsinformation schlägt in inkrementellen *Speech*-Parsern normalerweise fehl. Diese “herkömmliche Art” besteht darin, beim Verb eine Liste von subkategorisierten Konstituenten (die Komplemente) vorzuhalten und syntaktische Regeln vorzusehen, die ein Komplement nach dem anderen abbinden. Dies funktioniert solange gut, wie Verberst- oder -zweitstellung vorliegen. Bei Verbendstellung, wie im deutschen Nebensatz, tritt die Schwierigkeit auf, daß die Subkategorisierungsinformation *nach* den subkategorisierten Elementen erscheint. Neben der Konstruktion der Konstituenten, die subkategorisiert werden, muß die Einbindung in den Verbrahmen also quasi von rechts her erfolgen. Dies verlangt, daß entweder Cluster von möglichen Komplementen konstruiert werden, die in einem Analyseschritt mit der subcat-Liste des Verbs unifiziert werden, oder daß spezielle Mechanismen vorgesehen werden, die ein Abbinden der subcat-Liste von rechts her ermöglichen.

Die einfachste Möglichkeit, nämlich für jede auftretende Stelligkeit von Verben syntaktische Regeln vorzusehen, welche die jeweils vorhandenen Komplemente abbinden, läßt den Analyseaufwand sprunghaft steigen. In einem Test wurde ein Wortgraph zunächst mit einer vorhandenen Grammatik analysiert. Für den Turn N002K000 wurden 11 Analysen gefunden[20]. Die Bearbeitung erforderte eine Chart mit 1362 Kanten (gesamt: 4,8 MB Speicherbedarf) und dauerte 7,07 Sekunden. Durch Hinzunahme einer einzelnen Regel, die Verbphrasen mit Verbletztstellung für vier Komplemente beschreibt (s. Fig. 4.11), stieg der Speicherbedarf ungefähr auf das 3,5fache (4655 Kanten, 19 MB),

[20] N002K000 ist ein Turn mit mehreren Äußerungen: Schön hervorragend dann lassen Sie uns doch noch einen Termin ausmachen wann wäre es Ihnen denn recht.. Eine Analyse beschrieb den ersten Teil, zehn den zweiten Teil des Turns.

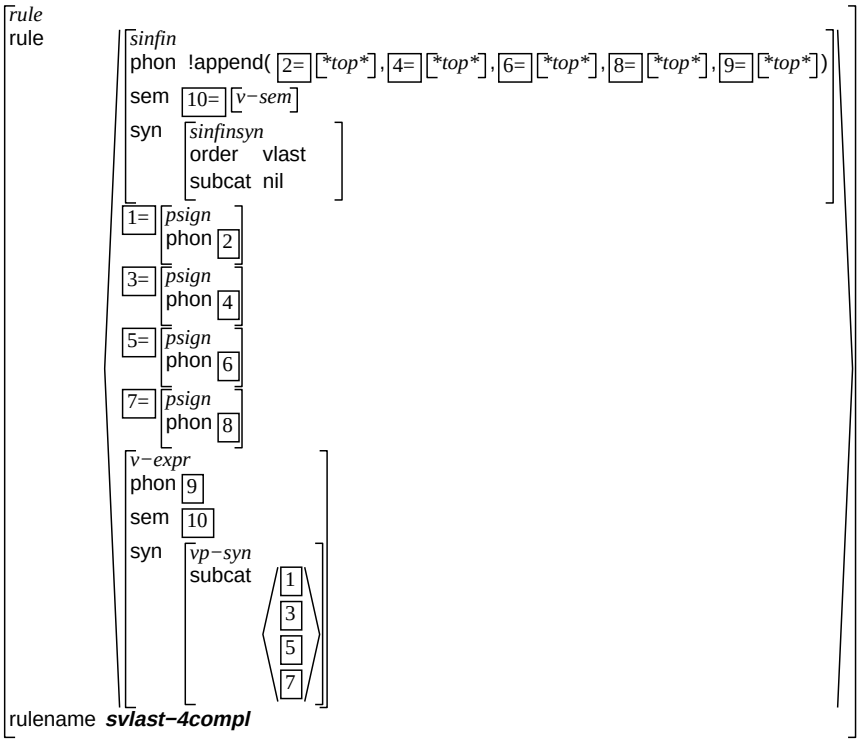


Abbildung 4.11: Eine Regel für Verben in Verbendstellung mit vier Komplementen

die Dauer der Analyse um das 5-fache (35,9 s) an. Im wesentlichen resultiert dieser Anstieg daraus, daß Cluster von potentiellen Komplementen gebildet werden, wo immer dies möglich ist. Mag dies bei der Strukturanalyse geschrieben eingegebener Sprache noch im Rahmen des Machbaren bleiben, findet bei der Verarbeitung gesprochener Sprache eine derartige Erhöhung der Analyselast statt, daß von solchen Methoden abgesehen werden muß.

Eine Lösung für diese Art von Problemen bestünde darin, vom strengen Links-Rechts-Paradigma der Analyse abzuweichen. Jede Grammatikregel könnte mit der Angabe über den "Kopf"[21] der Regel annotiert werden, der als Trigger für die Einführung neuer aktiver Kanten im Bottom-Up Schritt verantwortlich ist. Der Ableitungsalgorithmus wartet daraufhin mit der Anwendung der Regel, bis der Kopf gefunden ist, bevor links und rechts davon die übrigen Elemente gesucht werden. Diese Art von kopfgesteuertem Inselparsing vermeidet den überflüssigen Aufbau von Strukturen. Wir favorisieren jedoch den Ansatz, auf den Aufbau von Komplementkomplexen im ersten Analyseschritt vollständig zu verzichten und das Abbinden von Komplementen einer nachgeordneten Komponente zu überlassen, nämlich dem Integrator (vgl. Abschnitt 4.7).

4.6.2 Spontansprache und Worterkennung

Es wurde bereits an anderer Stelle darauf hingewiesen, daß gesprochene Sprache, und insbesondere spontan gesprochene Äußerungen, sehr oft die Grenzen einer für geschriebene Eingabe noch adäquater Grammatik sprengen (Batliner, Burger, und Kiessling, 1994). Häufig beobachtete Phänomene sind etwa im Bereich der Abbrüche von Sätzen und deren Neuanfängen, aber auch im Bereich ungewöhnlicher Kombinationen von syntaktischen Strukturen zu sehen. Das Design einer Wissensquelle für die Strukturanalyse derartiger Eingaben muß deswegen so gewählt sein, daß auch nicht wohlgeformte (im Sinne einer Standardgrammatik nicht wohlgeformte) Äußerungen analysiert werden können. Falls dazu ein holistischer Ansatz verwendet wird, der mit Hilfe einer einzigen Grammatik die Modellierung durchführt, kann die Folge sein, daß die Explosion der möglichen Kombinationen erheblichen negativen Einfluß auf die Performanz des Systems insgesamt hat. Deshalb werden häufig gestaffelte Ansätze verwendet, so z.B. auch innerhalb von INTARC (vgl. Weber, Amtrup, und Spilker, 1997), für das aus einer großen, merkmalbasierten Grammatik der kontextfreie Anteil für die schnelle stochastische Suche extrahiert wurde (Diagne, Kasper, und Krieger, 1995; Kasper et al., 1996).

Neben einer solchen horizontalen Aufteilung der Wissensquellen ist jedoch auch eine vertikale Aufteilung denkbar, die mehrere Komponenten für Einzelteile einer Äußerung vorsieht. Light (1996) z.B. präsentiert ein System von kaskadierten endlichen Automaten, das sukzessive längere Eingabeteile zu kombinieren versucht. MILC kombiniert beide Methoden, indem einerseits eine vertikale Aufteilung nach Subeinheiten einer Eingabe durchgeführt wird, und andererseits das Hauptgewicht in der Phase der partiellen Strukturanalyse auf syntaktischen Kriterien liegt, während die Konstruktion von Funktions-Argument-Strukturen, welche eine Äußerung semantisch beschreiben, der Integration vorbehalten bleibt.

[21] *Kopf* ist hier nicht ausschließlich im linguistischen Sinne gemeint.

Ein weiterer Aspekt gesprochener Sprache, der in der Repräsentation als Wortgraphen reflektiert wird, ist das Vorkommen von Häsitationen und Pausen innerhalb einer zusammenhängenden Äußerung. Dies macht sich in diskontinuierlichen Ketten von Worthypothesen bemerkbar, die mit Hilfe eines integrierten Syntax-Semantik Ansatzes nur schwer zu beschreiben sind. Denkbare Vorgehensweisen sind die Formulierung von Regeln, die an beliebigen Stellen Häsitationen erlauben, die Einführung von Kategorien, die während der Analyse ohne Modifikation der gewonnenen Repräsentation übersprungen werden dürfen, oder die Modifikation von Analysealgorithmen dahingehend, daß Konstituenten nicht ununterbrochene Pfade innerhalb des Wortgraphen darstellen, sondern Lücken enthalten können. Wir halten die letzten beiden Alternativen für die sinnvollsten. Die Möglichkeit, Kanten zu ignorieren, ist besonders vielversprechend, da nicht nur Pausen und Häsitationen Gegenstand dieser Behandlung sein können, sondern je nach Situation auch in Äußerungen eingestreute Wörter, die keinen relevanten Beitrag zur Bedeutung beisteuern. Für beide Verfahren kann es sinnvoll sein, eine Aufteilung von Modulen der syntaktisch-semantischen Analyse nach der Granularität vorzusehen, sei es, um unterschiedliche Kategorien vorzusehen, die ignoriert werden sollen, sei es, um Lücken in Wortgraphen je nach Umfang von Teilanalysen zu behandeln. Im Rahmen von MILC wurde mit der Definition von zu ignorierenden Kategorien experimentiert, insbesondere, um eingestreute Partikel zu behandeln, wie etwa in "Wir können uns *ja* vielleicht im März treffen".[22]

Die Wirkung der Einführung eines partiellen Strukturanalyseschemas zeigt sich deutlich, wenn der Analyseaufwand in verschiedenen Situationen verglichen wird. Tab. 4.2 enthält einige Angaben zu verschiedenen evaluierten Verfahren im bereits angesprochenen Experiment. Es werden verschiedene relevante Parameter für die vollständige Analyse von Transkripten (T), Wortgraphen (NIG) und linksverbundene Wortgraphen (IG), sowie für die partielle Strukturanalyse linksverbundener Wortgraphen (PP) angegeben. Zur Durchführung der Analyse wurde die oben erwähnte, idealisierte Grammatik verwendet, die u.a. keine diskontinuierlichen Konstituenten modelliert.

Tabelle 4.2: Ergebnisse für Standard- und partielles Parsing des Dialogs n002k

Parameter	Maß	T	NIG	IG	PP
Dichte (s. Def. 2.16)		1	22	181	181
Perplexität[23]		1	3,4	9,7	9,7
Agenda-Aufträge	#	2063	16238	86922	8591
Chart-Knoten	#	8	50	484	484
Chart-Kanten	#	654	2418	4982	909
Analysen	#	4	4	371	160
Unifikationen	#	831	3713	13848	1143
Analysezeit	s	9.37	94.94	113.64	8.56

Der Übergang von Transkripten zu Wortgraphen resultiert in einer ungefähr zehnfachen Verarbeitungszeit bei gleicher Grammatik, linksverbundene Wortgraphen erhöhen die Verarbeitungslast wei-

[22] Ob solche Partikeln vernachlässigt werden können, ist eine Ermessensfrage. Teilweise haben sie erheblichen Einfluß auf die Semantik von Äußerungen, insbesondere im Zusammenhang mit prosodischen Phänomenen, vgl. Niemann et al. (1997).

ter. Eine partielle Strukturanalyse hingegen reduziert den Aufwand wieder auf ein handhabbares Maß. Die Grammatik wurde dazu von allen Regeln befreit, die Komplementkomplexe für Verbalphrasen erstellen sowie die Regeln, die für die Anbindung von Präpositionalphrasen verantwortlich sind.

4.6.3 Struktur und Verarbeitungsstrategien

Die Eingabe, die das Modul zur partiellen Strukturanalyse erhält, besteht aus Worthypothesen, die vom Worterkenner geliefert werden, sowie aus Informationen zu erkannten festen Redewendungen aus der Idiomverarbeitung. Im einzelnen müssen folgende Typen von Nachrichten behandelt werden:

- **Hyperkanten**, die Scharen von Worthypothesen der ursprünglichen Erkennerausgabe repräsentieren. Sie sind dargestellt durch Hyperkanten der Mehr-Ebenen-Chart, die als Bezeichnung orthographische Wortformen tragen. Der Parser führt einen Lexikonzugriff durch, der in etlichen präterminalen Kanten resultieren kann. Für jede lexikalische Lesart wird eine Hyperkante erzeugt, deren Struktur identisch zu der empfangenen Wortkante ist. Die Bezeichnung hingegen besteht fortan aus der Merkmalstruktur, die aus dem Lexikon extrahiert wurde. Für jede solche Kante wird ein Auftrag generiert, sie in die Chart einzusetzen.

- **Modifikationen von Hyperkanten**, die durch die inkrementelle Arbeitsweise der Hypergraphenkonversion entstehen. Hierbei sind zwei Fälle zu unterscheiden: Besteht die Modifikation darin, daß ein zusätzlicher Start- oder Endknoten bekanntgegeben wird, so muß geprüft werden, ob dadurch evtl. neue partielle Pfade durch die bisher konstruierte Chart führen können. Dazu wird versucht, an dem neuen Knoten die fundamentale Regel für alle möglichen Kombinationen von Kanten aufzurufen. Dies ist allerdings nur dann erforderlich, wenn die ursprüngliche Kante bereits in die Chart eingesetzt wurde. Falls das noch nicht passiert war, wird der hinzugekommene Knoten mit den bereits vorhandenen zusammen verarbeitet. Besteht die Modifikation hingegen darin, daß die Bewertung der Kante sich geändert (d.h., verbessert) hat, muß ermittelt werden, ob Aufträge, die früher in der Verarbeitung aufgrund der Strahlensuche nicht berücksichtigt wurden, nun so gut sind, daß sie oberhalb des Schwellwertes liegen. Dann werden die mit der Kante verbundenen Aufträge erneut aktiviert.

  In jedem Fall wird die neue Information über die Kante an alle Kanten vererbt, die mit ihr konstruiert wurden. Ebenso wird, falls notwendig, die Änderung an folgende Komponenten übermittelt.

- **Idiomdefinitionen**, die analog zu Worthypothesen behandelt werden. Der Unterschied besteht lediglich darin, daß kein Lexikonzugriff notwendig ist, da die Idiomerkennung bereits eine Merkmalstruktur liefert, welche die Eigenschaften des Idioms beschreibt.

[23] Die Perplexität eines Graphen ist definiert als $p = 2^{\frac{\sum(\log_2(\#_{out}(v)))}{|\mathcal{V}|}}$.

- **Inhibitionen**, die Worthypothesen betreffen, die in ein Idiom integriert wurden. Die Verarbeitung besteht hier darin, die betreffende Worthypothese mit einer zusätzlichen negativen Bewertung zu versehen, um folgende Kombinationen mit anderen Kanten unwahrscheinlicher zu machen. Bereits im Vorwege konstruierte größere Analysen werden nicht zurückgenommen, hingegen wird in einem Vererbungsmechanismus deren kombinierte Bewertung verschlechtert. Inhibitorische Nachrichten werden, genau wie Verbesserungen von akustischen Bewertungen, an folgende Komponenten weitergegeben.

Abb. 4.12 zeigt ein Beispiel eines Lexikoneintrages für das deutsche Nomen "Arbeitstreffen". Der Eintrag enthält Angaben zu syntaktischen Eigenschaften (Kongruenz etc.) sowie Angaben zur Semantik. Die semantischen Konzepte sind hierarchisch angeordnet, indem sie als Typen des Formalismus repräsentiert werden. Die Supertypen von *"work-meeting"* sind *"meeting"*, *"termin"*, *"n-abstracts"* und *"n-concepts"*. Dadurch ist es z.B. möglich, innerhalb des Integrators Selektionsrestriktionen abhängig von allgemeineren Typen zu formulieren und weder eine vollständige Auflistung von Basistypen vorsehen noch artifizielle Merkmale einführen zu müssen, die gemeinsame Eigenschaften von Basistypen beschreiben.

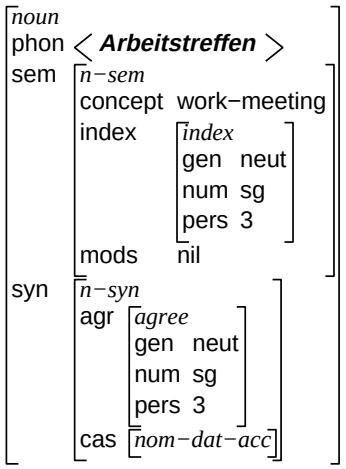


Abbildung 4.12: Einer der Lexikoneinträge für "Arbeitstreffen"

Die Menge von Lexikoneinträgen wird in ihrer Quellform als Liste von Oberflächenrepräsentationen und zugehörigen Merkmalstrukturen kodiert. Die Syntax der Merkmalstrukturen folgt dabei der in der Tab. 3.2 angegebenen. Zusätzlich sind parametrisierbare Makros implementiert, die flexiblen Textersatz realisieren und rekursive Einbettung erlauben. Diese Oberflächenform wird durch einen Übersetzer in eine interne Form umgewandelt, die gleichzeitig schnellen Zugriff über einen mit Hilfe von *Red-Black-Trees* (vgl. Cormen, Leiserson, und Rivest, 1990) implementierten Index erlaubt. Eine wesentliche Eigenschaft aller in MILC verwendeten Wissensquellen ist die Benutzung eines global definierten Typenverbandes, der zur Kompatibilität der Merkmalstrukturen in allen Modulen führt.

Die Verarbeitungsrichtung innerhalb der partiellen Strukturanalyse ist strikt inkrementell von links nach rechts. Das bedeutet, daß jedesmal dann, wenn ein neues Eingabefragment eintrifft, das einen Knoten betrifft, der weiter rechts in der Zeit liegt als der bisher betrachtete, ein Zyklus zur Agendabearbeitung ausgelöst wird. Dieser Zyklus bewertet zunächst die in der Agenda momentan vorhandenen Aufträge und führt die Strahlensuche durch. Abhängig von der Konfiguration des Parsers kann die Breite des Suchstrahls reguliert werden. Dazu stehen neben der direkten Angabe dieser Breite zwei zusätzliche Möglichkeiten zur Verfügung: der Benutzer kann festlegen, daß ein bestimmter Anteil der in der Agenda aktuell vorhandenen Aufträge bearbeitet wird (z.B. 60% der Aufträge), oder er kann die Höchstanzahl der zur Ausführung gelangenden Aufträge angeben (ein typischer Wert ist hier, jeweils maximal 200 Aufträge pro Chartknoten zuzulassen). Nach der Eingrenzung der Agenda werden die als zur Ausführung anstehend markierten Aufträge nacheinander bearbeitet.

Jeder einzelne Auftrag besteht darin, eine der Grundfunktionen der Chartanalyse durchzuführen. Die beiden wesentlichen Operationen, die hier benötigt werden, sind das Vorschlagen neuer Kanten unter Einbeziehung von Regeln der Grammatik sowie die Kombination einer aktiven mit einer inaktiven Kante.

Die Regeln der verwendeten Grammatik sind als Merkmalstrukturen des Typs *rule* formuliert. Das Merkmal rule bildet eine Liste von eingebetteten Merkmalstrukturen, deren erste die linke Seite der Grammatikregel und deren weitere Merkmalstrukturen die rechte Seite bilden. Es handelt sich mithin um ein an Phrasenstrukturregeln orientiertes Analyseverfahren, im Gegensatz etwa zu stark an Prinzipien ausgerichteten Paradigmen (vgl. Konieczny, 1996). In der Tat sind die Grammatiken von MILC lose an PATR II (Shieber, 1984) ausgerichtet, indem überwiegend von einem kontextfreien Gerüst (gebildet durch die Typen der Merkmalstrukturen einer Regel) und zugehörigen Annotationen ausgegangen wird. Abb. 4.13 verdeutlicht das Aussehen einer syntaktischen Regel für die partielle Strukturanalyse.

Durch diese Grammatikregel werden Nominalphrasen lizensiert, die aus einem Artikel (Merkmalstruktur des Typs *det*) und weiteren nominalen Anteilen (Typ *n2*, im einfachsten Fall ein Nomen) bestehen. Die syntaktische und semantische Struktur wird vom Nomen an die NP vererbt, was in diesem Fall etwa der Wirkung des *Head Feature Principle* der HPSG entspricht. Vererbung ist spezifiziert durch die Koreferenzen zwischen den untergeordneten Merkmalstrukturen der rechten Seite der Regel mit der die linke Seite bildenden Merkmalstruktur. Der frontale Artikel bestimmt die Definitheit und muß zusätzlich mit den übrigen Teilen der NP kongruieren.

Die Grammatik wird immer dann konsultiert, wenn eine inaktive Kante in die Chart eingetragen wird. Jede der Regeln wird daraufhin überprüft, ob die inaktive Kante mit der ersten untergeordneten Merkmalstruktur der rechten Seite unifizierbar ist. Im Erfolgsfall wird eine neue Kante erzeugt, die als Beschreibung das Ergebnis der Unifikation trägt. Sie ist aktiv, wenn weitere Konstituenten auf der rechten Seite verlangt werden. Ist hingegen die Regel vollständig konsumiert (unter den hier geschilderten Umständen handelte es sich hierbei dann um eine sog. Kettenregel, die nur ein Symbol auf der rechten Seite trägt), so wird eine inaktive Kante konstruiert. Ein neuer Auftrag zur Insertion der Kante in die Chart wird generiert.

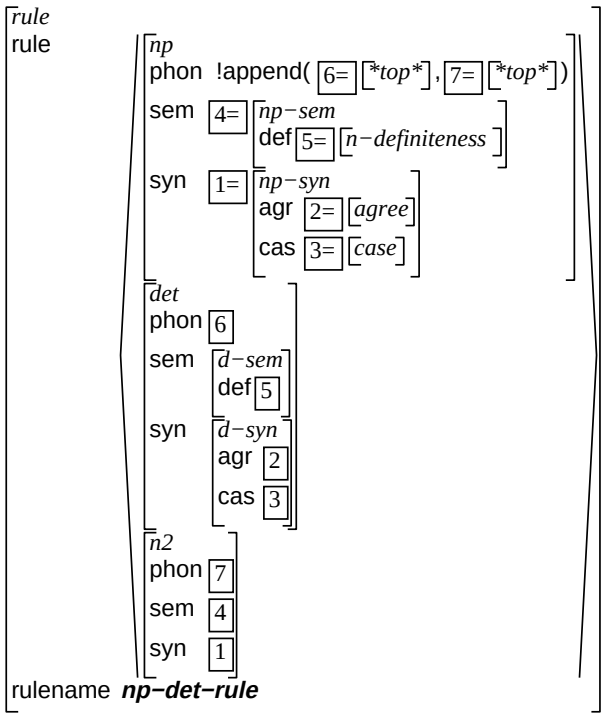


Abbildung 4.13: Grammatikregel für Nominalphrasen mit Artikel

An der Art, in der die Grammatik zur Einführung neuer Kanten benutzt wird, ist ersichtlich, daß MILC im Rahmen der partiellen Strukturanalyse eine Bottom-Up Strategie verfolgt, genauer gesagt, eine von Kilbury (1985) vorgeschlagene Variante, die leere inaktive Kanten vermeidet (und damit dafür sorgt, daß die Mehr-Ebenen-Chart frei von trivialen Zyklen bleibt). Übliche Verarbeitungsstrategien lassen sich in Top-Down- und Bottom-Up-Mechanismen unterscheiden. Einen Überblick über etliche Strategien liefert z.B. Wirén (1988), der in seiner Dissertation (Wirén, 1992) Bezüge zwischen Parsingstrategien und Begründungsverwaltungssystemen in Zusammenhang mit inkrementeller Verarbeitung geschriebener Sprache aufzeigt.

Die zweite Hauptoperation, die wir erwähnen wollen, ist die Kombination von Kanten der Chart. Sie wird immer dann angewendet, wenn eine aktive und eine inaktive Kante einen gemeinsamen Berührungspunkt haben. Im Rahmen der Strukturanalyse bedeutet dies, daß der Schnitt der Menge von Endknoten der aktiven und der Menge von Startknoten der inaktiven Kante nicht leer ist. Daraufhin wird versucht, die Beschreibung der inaktiven Kante mit derjenigen Teilbeschreibung der aktiven Kante zu unifizieren, die das nächste Element der zugehörigen Grammatikregel spezifiziert. Gelingt dies, so kann eine neue Kante erzeugt werden, die, abhängig von der Position innerhalb der Regel, inaktiv oder aktiv wird. Ist sie inaktiv, so wird die Teilmerkmalstruktur extrahiert, welche die linke Seite der Regel enthält und erneut ein Grammatikabgleich durchgeführt. Wie bereits angedeutet, findet die Strukturanalyse strikt inkrementell von links nach rechts statt, die angegebene Bedingung ist folglich die einzige, die zum Erfolg der Kombination von Kanten führen kann.

Die Bewertung der neuen Kante ergibt sich aus der Kombination der Bewertungen der Ursprungskanten. Als akustische Bewertung wird das längennormalisierte Mittel der akustischen Bewertungen der Ausgangskanten verwendet (vgl. Abschnitt 2.6.3). Zusätzlich wird eine modifizierende Bewertung anhand eines statistischen Sprachmodells angenommen. Dafür wird die Sequenz der zugrundeliegenden Wörter hinsichtlich der Wahrscheinlichkeit ihrer Abfolge untersucht. Die Quelle dazu ist das innerhalb von Verbmobil regelmäßig verwendete Modell, das Bigrammwahrscheinlichkeiten berechnen kann. Die Normalisierung dieser Bewertung findet aufgrund der Anzahl der betroffenen Wörter statt.

Die Ausgabe der partiellen Strukturanalyse schließlich besteht im wesentlichen aus inaktiven Kanten, die gemäß der Typenhierarchie Zielcharakter haben. Dazu wird vom Benutzer ein Zieltyp definiert. Alle Kanten, deren Merkmalstruktur von einem Typ sind, der von dem angegebenen subsumiert wird, werden an den Integrator und den Transfer ausgeliefert. Abb. 4.14 zeigt als Beispiel das Resultat der Analyse für die Nominalphrase “das nächste Arbeitstreffen”. Zusätzlich werden, wie bereits vorher erwähnt, Vererbungsinformationen bezüglich der Hyperkanten und erkannter Idiome sowie zugehöriger Inhibitionen weitergeleitet.

4.7 Integration von Äußerungskontexten

Die Aufgabe der Äußerungsintegration von MILC besteht darin, aus den partiellen Analysen, die durch die vorhergehenden Komponenten konstruiert wurden, möglichst vollständige, größere Teilbereiche einer Eingabe überspannende, semantische Beschreibungen zu erzeugen. Dazu zählt ins-

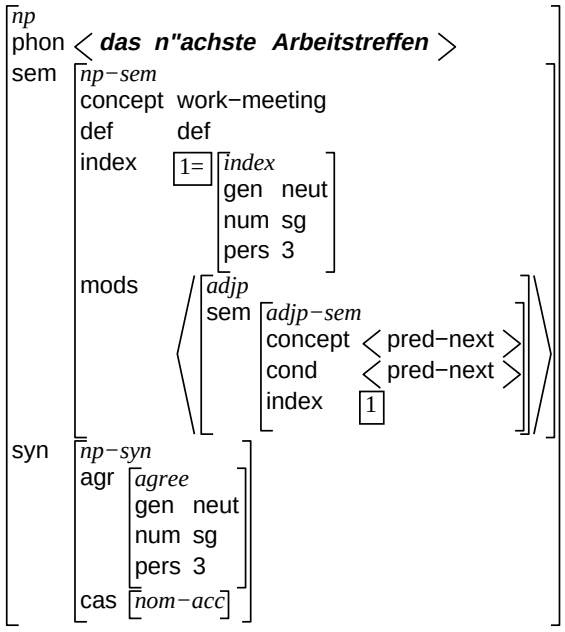


Abbildung 4.14: Eine Nominalphrase aus dem partiellen Parser

besondere der Aufbau von Verbalphrasen, aber auch die Anbindung von Ergänzungen an Nominalgruppen, z.B. die von Präpositionalphrasen. In diesem Sinne wird durch die Äußerungsintegration die syntaktisch-semantische Analyse, die durch den partiellen Parser begonnen wurde, weitergeführt und abgeschlossen.

Die Eingabedaten, die der Integrator verarbeiten muß, enthalten:

- **Inaktive Hyperkanten des Parsers**, die Zielstatus haben. Dazu gehören u.a. vollständige Beschreibungen von Nominalphrasen, Präpositionalphrasen, Adverbialgruppen etc.. Außerdem sind dies Verben, für die der Parser lediglich einen Lexikonzugriff durchgeführt hat, um die zugehörige Merkmalstruktur an den Integrator weitergeben zu können. Diese Kanten werden in die Chart eingetragen und nach dem unten angegebenen Verfahren weiter kombiniert.
- **Modifikationen von Hyperkanten**, die analog zu dem oben beschriebenen Ansatz behandelt werden. Die Modifikationen bestehen aus Änderungen von Knotenmengen, der Verbesserung einer akustischen Bewertung oder der Inhibition aufgrund von Idiomen. Alle drei Typen werden sowohl innerhalb der Komponente vererbt als auch an den Transfer weitergegeben, falls dies notwendig sein sollte.

Die hauptsächlich benutzte Wissensquelle für den Integrator ist seine Grammatik, die vom Parser

gelieferte Analysen zu größeren zu kombinieren versucht. Im Unterschied zu diesem ist der Integrator jedoch in der Lage, eine Inselanalyse (vgl. Steel und Roeck, 1987; Stock, Falcone, und Insinnamo, 1988) durchzuführen. Dieses Verfahren geht davon aus, daß Regeln der Grammatik nicht strikt von links nach rechts abgearbeitet werden, d.h. das erste Symbol auf der rechten Seite muß nicht unbedingt zuerst konsumiert werden, bevor das nächste betrachtet werden darf. Statt dessen wird in jeder Regel eine sog. *Insel* definiert, die als Auslöser für die Anwendung einer Regel dient. Dieses Element (in unserem Fall stets eine Merkmalstruktur) wird in jedem Fall als erstes behandelt. Ist erst einmal eine Ableitung hierfür gefunden, so können Symbole links und rechts davon analysiert werden. Die Regel in Abb. 4.15[24] beschreibt beispielsweise eine Nominalphrase, bestehend aus Artikel, Adjektiv und Nomen; die wesentliche Einschränkung, die hier formuliert ist, sorgt dafür, daß grundsätzlich zuerst das Adjektiv konsumiert wird. Danach ist die Reihenfolge unspezifiziert, abhängig von der Verarbeitungsstrategie wird entweder der Artikel oder das Nomen in der Folge angebunden.

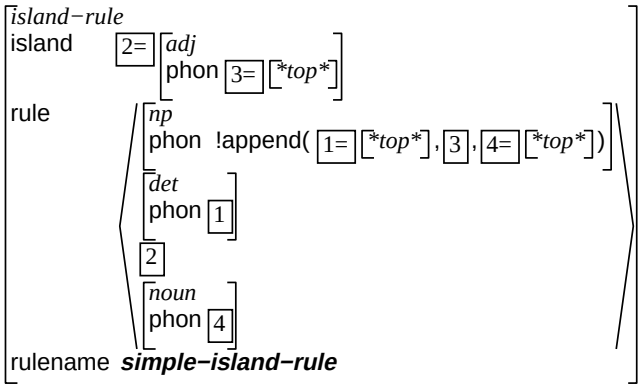


Abbildung 4.15: Inselanalyse: Eine Beispielregel für Nominalphrasen

Die Motivation für einen derartigen Ansatz zur Strukturanalyse liegt ursprünglich in zwei Bereichen:

- Die strukturelle Analyse einer Eingabe sollte mit Teilbeschreibungen beginnen, die möglichst weitreichende Einschränkungen bezüglich der Kombinierbarkeit von Elementen tragen. Das hilft, den Analyseaufwand zu reduzieren und damit effizienter zu verfahren. Dieses Motiv herrscht in Ansätzen zur bidirektionalen Chartanalyse für geschriebene Sprache vor, vgl. z.B. Satta und Stock (1989), Satta und Stock (1994). Insbesondere ist hiermit die Einführung einer kopfgesteuerten Analyse elegant durchführbar (Nederhof und Satta, 1994). Die Definition einer Inselanalyse wird jedoch nach wie vor dieselben Resultate liefern wie eine konventionelle Modellierung. Lediglich die Suche nach den Resultaten verläuft in einer anderen, besseren Art (Stock, 1989).

[24] Diese Regel dient lediglich der Illustration. Adjektivkomplexe werden von MILC nicht innerhalb des Integrators, sondern durch die partielle Strukturanalyse behandelt.

- Die Analyse sollte mit möglichst erfolgversprechenden Eingabeelementen beginnen. Dieses Argument ist speziell auf die Analyse gesprochener Sprache anwendbar. Erfolgversprechend sind hier Worthypothesen, die eine gute akustische Bewertung tragen, also vergleichsweise sicher erkannt wurden. Die grundsätzliche Idee besteht darin, sich diese "sicheren Inseln" herauszugreifen und davon ausgehend, nach links und rechts weitere mit diesem kompatible Wörter zu suchen und in einen Gesamtkontext zu integrieren (vgl. Brietzmann, 1992). Da die Strukturanalyse gesprochener Sprache im Normalfall Effizienz dadurch zu gewinnen sucht, daß Suchräume beschnitten werden, kann der Einsatz einer Inselanalyse hier zu Modifikationen des Suchraums führen, die Äquivalenz der Ergebnisse mit denen einer konventionellen Analyse ist folglich nicht mehr gesichert.

Im Zusammenhang mit dem in dieser Arbeit verwendeten inkrementellen Paradigma scheint die Ausnutzung einer bidirektionalen Analyse zunächst kontraproduktiv, gehen wir doch davon aus, daß wir stets fortlaufend in der Zeit arbeiten wollen. Ein in solcher Weise prinzipielles Argument zu befolgen, läßt jedoch einige Aspekte außer acht. Die Hauptmotivation, im hier beschriebenen System eine Inselanalyse vorzusehen, wurde bereits angesprochen: Zu verhindern, daß unnötigerweise viele Komplementkomplexe aufgebaut werden. Darüberhinaus erscheint es nicht sinnvoll, die Existenz optionaler Elemente von Konstituenten (seien es optionale Komplemente oder Adjunkte) unabhängig von deren tatsächlichem Vorkommen zu hypothetisieren. In diesem Bereich kann eine Inselverarbeitung dazu benutzt werden, nicht mehr zu tun als unbedingt notwendig.

Alle diese Maßnahmen zielen letztlich auf die Steigerung der Effizienz und damit auf auf die Erhöhung der Akkuratheit der Analyse, da innerhalb des gewählten Suchstrahls nun weniger unnötige Aufträge erscheinen und dadurch die akustisch schlechter bewerteten, aber strukturell plausiblen Aufträge Beachtung finden. Letztlich ist aber die Frage, wie gut sich eine Inselanalyse mit einem inkrementellen Paradigma abstimmen läßt. Dazu muß der zeitliche Versatz betrachtet werden, der etwa dadurch entsteht, daß gewartet wird, bis ein Verb vorliegt, bevor die Erzeugung von Komplementkomplexen begonnen wird. Dieser Versatz erweist sich als recht klein. Er liegt im Bereich von einigen Aufträgen, die nach Einsetzung des Verbs in die Chart nachträglich durchgeführt werden müssen. Diese Aufträge bestehen in der Sättigung der Subkategorisierungsliste des Verbs durch links vom Verb auftretende Komplemente. Im Normalfall des deutschen Hauptsatzes ist dies lediglich das Subjekt, für dessen Integration eine Kombination von Kanten notwendig ist. Den Extremfall bildet der deutsche Nebensatz mit Verbletztstellung. Hier ist die Anbindung sämtlicher Komplemente durchzuführen, für jedes Komplement erneut ein Auftrag. Die Grammatikregel, die diesen Fall beschreibt, ist in Abb. 4.16 angegeben. Zusätzlich ist eine Kettenregel zu erfüllen, die für das Einsetzen der kompletten Verbalphrase in die Chart sorgt. Der nachträgliche Aufwand liegt somit praktisch im Bereich von einem bis etwa fünf Aufträgen.

Ein weiterer Effekt der bidirektionalen Analyse ist, daß die Agenda nicht mehr an einen Zeitpunkt gebunden ist. Im Falle der partiellen Strukturanalyse wird zu jedem Zeitschritt ein Bearbeitungszyklus der Agenda ausgelöst. Aufträge für davorliegende Zeitpunkte treten ausschließlich dann auf, wenn durch die Modifikation von akustischen Bewertungen von Kanten bereits bestehende Kombinationen über den Schwellwert der Strahlensuche geraten. Im Integrator hingegen ist die zeitliche Zugehörigkeit eines Auftrages im wesentlichen irrelevant. Die Agenda wird zwar auch hier dann

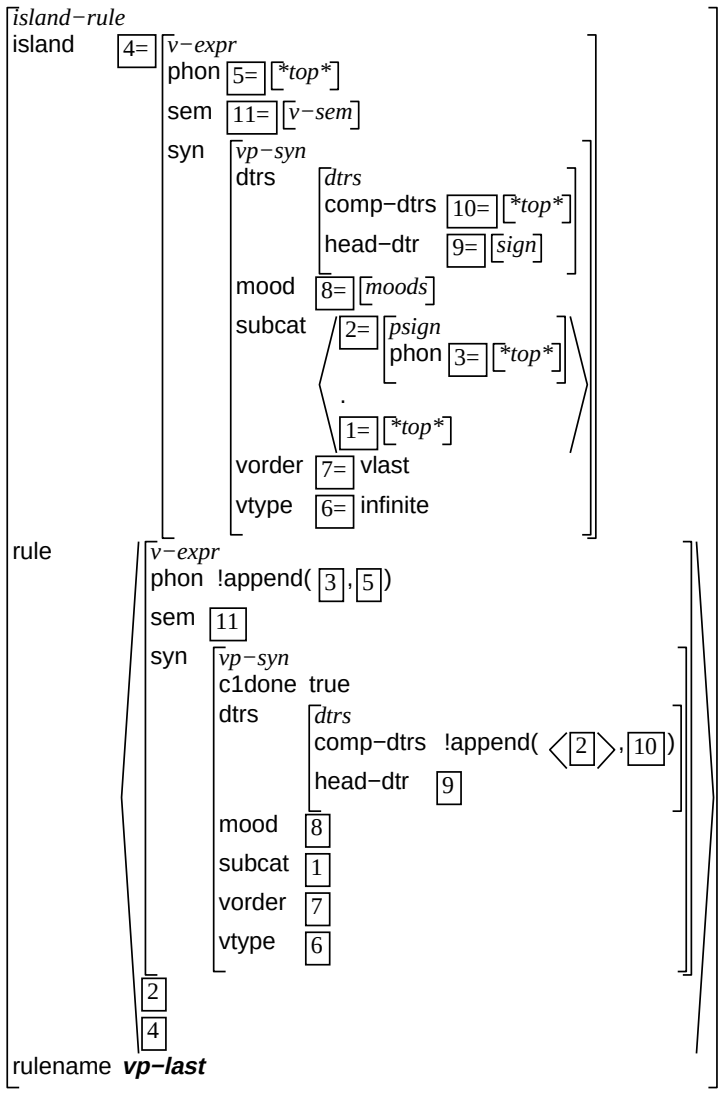


Abbildung 4.16: Eine Regel für Komplemente links vom Verb

bewertet und ihre Aufträge ausgeführt, wenn ein neuer Knoten für die Analyse auftritt, die Aufträge selbst können beliebige andere Ausrichtungen haben.

Um die Funktionsweise der Analyse deutlich zu machen, werden wir hier eine grobe Beschreibung der Analyse von "lassen Sie uns den nächsten Termin ausmachen" geben. Die interessanten Nachrichten, die den Integrator vom partiellen Parser aus erreichen, sind die folgenden:

Nr.	Inhalt	Bemerkung
1	lassen	Verben werden vom Parser an den Integrator durchgereicht
2	Sie	NP
3	uns	NP
4	Termin	NP
5	nächsten Termin	NP
6	den nächsten Termin	NP
7	ausmachen	Verb

Zunächst wird die imperative Lesart von "lassen" in die Chart eingesetzt (die alternativen Lesarten vernachlässigen wir hier). Sie subkategorisiert für drei Komplemente, nämlich das Subjekt, ein direktes Objekt und eine untergeordnete Verbalphrase mit Verbletztstellung im Infinitiv. Daraufhin kann eine Grammatikregel angewendet werden, die Verberststellung für Imperative fordert. Das Ergebnis ist eine aktive Kante vom Typ *v-expr*, die das Verb überspannt und eine weiteres noch nicht erfülltes Element auf der rechten Seite besitzt. Dies rührt daher, daß wir Komplementpositionen der Subkategorisierungsliste nacheinander sättigen. Die nächsten eintreffenden Kanten ("Sie" und "uns") werden in die Chart eingeführt und binden die ersten beiden Komplemente ab. Die längste aktive Kante ist nun eine Beschreibung von "lassen Sie uns", die auf eine VP wartet.

Die folgenden drei Kanten "Termin", "nächsten Termin" und "den nächsten Termin" haben für die anstehende Analyse zunächst keine Bedeutung. Sie werden lediglich in die Chart eingeführt. Die letzte Nachricht, die den Integrator erreicht, ist die Beschreibung von "ausmachen". Die Merkmalstruktur hat die in Abb. 4.17 angegebene Form.

Die in Abb. 4.16 angegebene Grammatikregel produziert aus dem Verb einen verbalen Ausdruck, dessen Subkategorisierungsliste lediglich das Objekt enthält. Das Einsetzen resultiert folglich in einer aktiven Kante. Die annotierte Merkmalstruktur hält den Typ *rule*, da die zugehörige Grammatikregel noch nicht vollständig erfüllt wurde.[25] Die rechte Seite der Regel enthält Strukturen des Typs *np* und *v-island* (*v-island* ist ein Subtyp von *v-expr*). An der Regel ist angemerkt, daß das erste Element noch nicht berücksichtigt wurde.

Daraus ergibt sich die Situation, daß beim Einsetzen nach passenden inaktiven Kanten *links* gesucht wird. Dort existieren drei mögliche Kandidaten, nämlich die eben erwähnten Nominalphrasen. Es

[25] Die Annotationen an aktiven Kanten werden von der angewendeten Regel subsumiert. Inaktive Kanten erhalten lediglich die linke Seite der Regel.

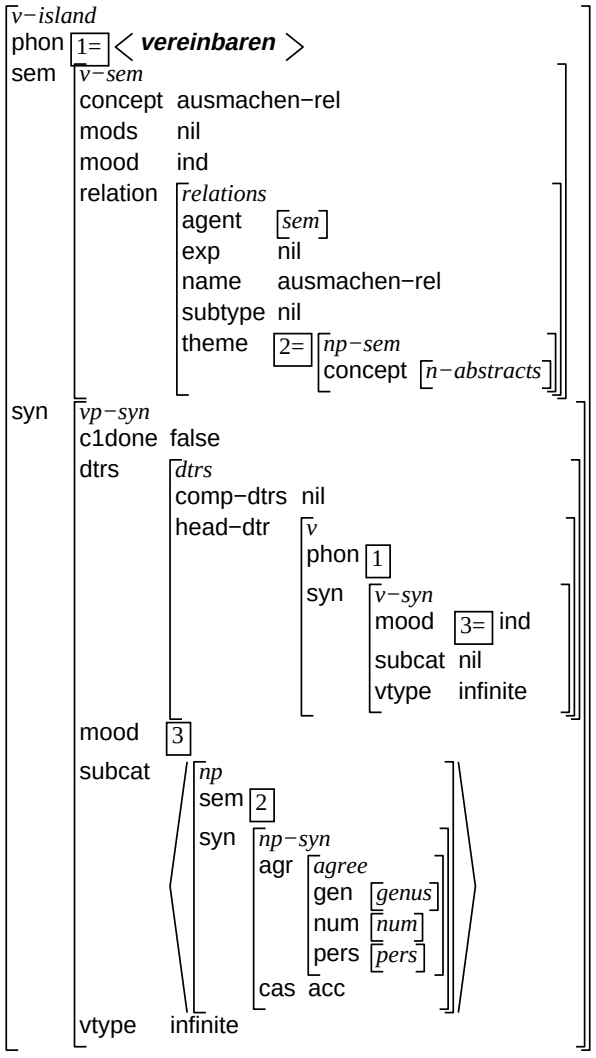


Abbildung 4.17: Lexikoneintrag für “ausmachen”

können also drei inaktive Kanten produziert werden. Lediglich eine dieser Kanten, nämlich die längste, hat jedoch Kontakt zu einer aktiven Kante, mit der sie kombiniert werden kann. Ist dies geschehen, so kann schließlich die in Abb. 4.18 dargestellte inaktive Kante erzeugt werden, welche die gesamte relevante Eingabe überspannt.

Falls diese Kante Zielstatus hat, was wiederum durch Subsumption mit einem vordefinierten Typ geprüft wird, so wird sie an den Transfer weitergegeben. Neben solchen erfolgreich analysierten Konstituenten gibt die Äußerungsintegration — analog zur partiellen Strukturanalyse — ebenfalls Vererbungsinformation weiter.

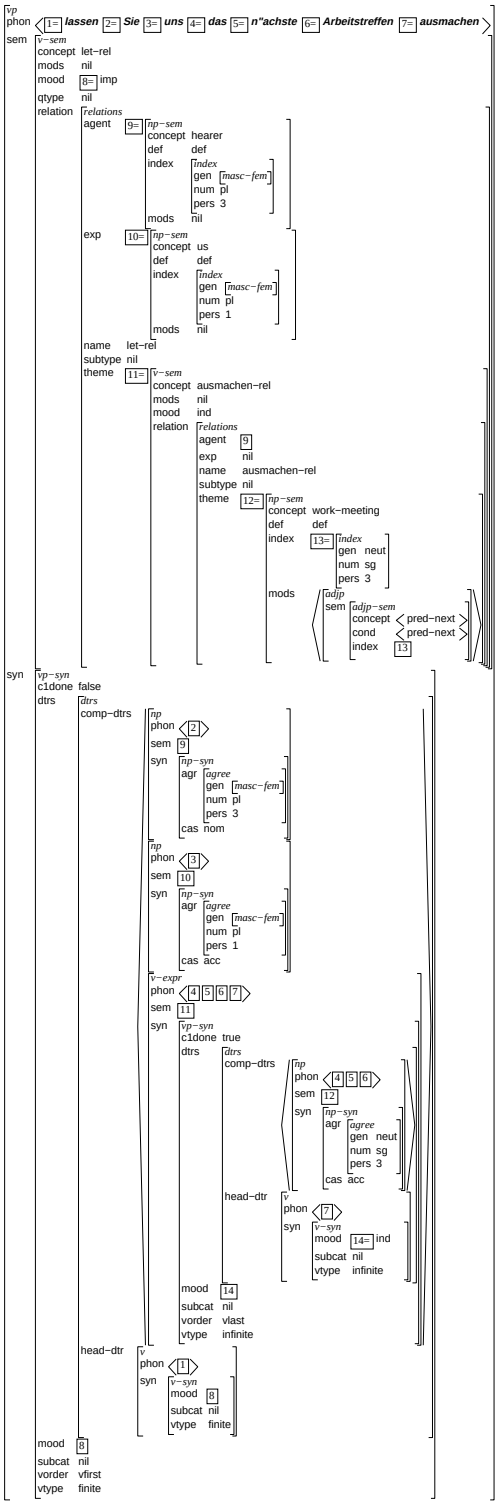


Abbildung 4.18: Eine Verbalphrase aus dem Integrator

4.8 Transfer

Die grundsätzliche Aufgabe, die der Transfer für die automatische Übersetzung von Sprache übernimmt, ist, Verbindungsstelle zwischen linguistischen Beschreibungen unterschiedlicher Sprachen zu sein. Wie bereits in der Einführung erwähnt, findet ein transferorientierter Ansatz zur Übersetzung in immer mehr Systemen Einsatz, insbesondere, wenn ein System nicht nur auf Äußerungen in extrem kleinen, technischen Domänen angewendet werden soll, sondern Alltagssprache verarbeiten muß, die der Beschreibung durch eine Interlingua sehr viel schwerer zugänglich ist (Hutchins, 1994). Außerdem ist ein Transferansatz aus Effizienzgründen selbstverständlich für die Bearbeitung gesprochener Sprache vorzuziehen. Zusätzlich kann einem Prozeßmodell, als das man den Transfer charakterisieren kann, eine aktive Rolle zugesprochen werden, im Gegensatz zur statischen Beschreibung von Sachverhalten mit Hilfe einer Interlingua. Allerdings muß angemerkt werden, daß der Transfer in den meisten Systemen immer noch hauptsächlich Rezipient von Information ist, ohne eine gravierende Einflußmöglichkeit auf deren Produktion durch vorangegangene Arbeitsschritte zu haben. Prinzipiell wäre ein Transfer denkbar, der die zentrale Instanz eines Übersetzungssystems repräsentiert und — angefangen auf einem hohen abstrakten Niveau — Anfragen an andere Wissensquellen stellt. Diese werden sukzessive in einfachere Aufträge dekomponiert und führen schließlich zu Erwartungen, die z.B. innerhalb der Spracherkennung ausgewertet werden.[26]

Dies erscheint psycholinguistisch unplausibel und ist in voller Stringenz sicher nicht mit Erfolg durchführbar. Immerhin gilt, daß auch der Transfer zur Disambiguierung beitragen kann; es sollte möglich sein, von der Verfolgung von Hypothesen abzusehen, die vom Transfer als nicht verarbeitbar kategorisiert wurden, sowie die Generierung von Prädiktionen zuzulassen, die aufgrund kontrastiven Wissens etabliert werden. Zusätzlich ist denkbar, daß ein Transfermechanismus zusätzliche Auswertungen durch andere Komponenten anstoßen kann, wenn dies aufgrund fehlender Information oder mangelnder Übersetzungsqualität notwendig sein sollte. Dies ist zum Beipiel bei Unterschieden im Zeitsystem der beteiligten Sprachen der Fall (Eberle, Kasper, und Rohrer, 1992), oder bei deiktischen Referenzen, die unterschiedliche interpretiert werden können: “Geht es da bei Ihnen” kann als temporale Referenz (“Is that possible for you”) oder als lokale Referenz (“Would it be okay at your place”) aufgefaßt werden (Alexandersson, Reithinger, und Maier, 1997).

Derartige zusätzliche Auswertungen werden bereits heutzutage in großen Übersetzungssystemen eingesetzt, etwa zur Behandlung lexikalischer *Mismatches* (Dorr, 1993) bei übergeneralisierenden Wortkorrespondenzen oder um eine korrekte Übersetzung von Tempus und Aspekt in gewissen Fällen zu garantieren (Buschbeck-Wolf, 1995). Zur Abwicklung derartiger Anfragen bietet sich eine Chart als integrierte Datenstruktur an, da sie den Austausch konsistenter Information ohne den zusätzlichen Aufwand der Einführung eines Frage-Antwort-Protokolls mit komplizierten Verzögerungsmechanismen bereits durch Kantenreferenzen realisiert.

[26] Ein System, das die zentrale Rolle des Transfers hervorhebt, ist TDMT (Sobashima et al., 1994). Dort werden Übersetzungen anhand von vorher gesehenen Beispielen durchgeführt. Die wichtigste Wissensquelle ist eine Tabelle von Transfereinträgen, die mit der Oberflächenrepräsentation der Eingabe verglichen werden. Bei Bedarf werden weitere Auswertungsvorgänge lexikalischer oder syntaktischer Art angestoßen. Vgl. außerdem Kay, Gawron, und Norvig (1991, S. 89ff).

In heutzutage gängigen Übersetzungssystemen ist die Situation, in der eine Komponente zum Transfer zum Tragen kommt, normalerweise diese: Eine quellsprachliche Eingabeäußerung liegt in einer vollständig analysierten Form vor. Das Resultat der Analyse ist eine Struktur, die eine Beschreibung auf syntaktischer oder semantischer Ebene enthält. Im Regelfall wird verlangt, daß die Beschreibung eindeutig ist, dem Vorliegen von verschiedenen, strukturell oder inhaltlich unterschiedlichen Beschreibungen wird durch wiederholte Anwendung des Transferapparates entsprochen. Die Eingabestruktur wird *top-down* traversiert, eine zielsprachliche Beschreibung wird beginnend mit dem obersten Knoten erzeugt. Innerhalb des Suchraumes des Transfers findet eine Breitensuche statt, die nächsttiefere Ebene der Repräsentation wird erst dann verwendet, wenn alle höheren Ebenen komplett bearbeitet sind. Für die Eingabe von geschriebener Sprache reicht ein derartiger konservativer Mechanismus aus, die Eigenarten der inkrementellen Verarbeitung gesprochener Sprache verlangen allerdings ein weitergehendes Vorgehen. Dies ergibt sich bereits aus der Tatsache, daß innerhalb eines inkrementellen Systems erst sehr spät eine vollständige Sicht auf die aus der Eingabe folgenden Information zur Verfügung steht, jede Komponente jedoch schon vorher beginnen muß, partielle Resultate zu produzieren.

Zur Architektur des Transfers ist zunächst anzumerken, daß die Einführung einer separaten Komponente angemessen erscheint. Selbst wenn die prinzipielle Erwägung, die sich aus der in Kapitel 1 erläuterten schwachen Modularitätshypothese und grundsätzliche softwaretechnische Einsichten nicht als ausreichende Begründung angesehen werden, so sprechen weitere Beobachtungen linguistischer Natur nachdrücklich für eine derartige Modularisierung:

- Natürlichsprachliche Systeme sind oft an den klassischen linguistischen Beschreibungsebenen Morphologie, Syntax, Semantik und Pragmatik orientiert, so daß sich für jede Beschreibungsebene ein Modul ergibt. Die Annotierung von Regeln einer anderen Komponente mit Spezifikationen darüber, wie ein Transfer auszuführen sei, ist zwar möglich. Die Ankopplung von Transferwissen an eine ausgezeichnete dieser Ebenen widerspricht aber der Intuition von verschiedenen Verarbeitungsschritten, die zumindest partiell evident scheint, vgl. Kapitel 1. Zudem sind Sprachverstehen und Sprachübersetzung in der menschlichen Informationsverarbeitung getrennt, obwohl sich beide Prozesse überlappen können. Das Konzept der variablen Analysetiefe legt ebenfalls nahe, daß Transferregeln nicht an Regeln anderer Art angekoppelt werden sollten: Eine Annotation z.B. auf syntaktischer Ebene kann weder mit lexikalischen Übersetzungen (Idiomen) noch mit der eine tiefe Analyse erfordernden semantischen Problemen umgehen (Prahl et al., 1995).

- Die Korrespondenz zwischen Transferregeln und Regeln auf anderen linguistischen Ebenen ist nicht symmetrisch. Als Beispiel diene erneut die syntaktische Strukturanalyse: Es ist klar, daß nicht jede strukturbildende Regel im Bereich der Grammatik einer Regel entspricht, die die konstruierte Struktur zu übersetzen in der Lage wäre.[27] So kann z.B. eine Präpositionalphrase im Englischen (wie in “Mounting of the drill” im Deutschen abhängig von Stilfragen wiederum durch eine PP “Montage von dem Bohrer” oder, besser, durch ein Genitivattribut

[27] Trotzdem gibt es Ansätze, die syntaktische Regeln mit Übersetzungen versehen (Katoh und Aizawa, 1994). Allerdings wird hier die Grammatik aufgeteilt, um etwa festen Redewendungen direkt ein Translat zuzuordnen.

"Montage des Bohrers" übersetzt werden[28]. Es ist nicht einzusehen, warum bereits bei der Analyse einer Präpositionalphrase beide Transfermöglichkeiten vorgesehen werden sollten, da die Genitivübersetzung einen Spezialfall darstellt. Zusätzlich gilt, daß syntaktische Strukturbildung und Transfer nicht synchron ablaufen. So kann bei der Analyse von The Big Dog die syntaktische Struktur graduell aufgebaut werden. Die Übersetzung muß in diesem Fall jedoch auf den Kopf der Phrase warten, einerseits um das Genus der NP feststellen zu können und einen angemessenen definiten Artikel zu produzieren, andererseits auch, um eine lexikalisch richtige Übersetzung des Adjektivs zu bewirken (big in der Bedeutung "körperlich groß" vs. in der Lesart "bedeutend").

4.8.1 Chartbasierter Transfer

Charts wurden bisher überwiegend für die Analyse und Generierung natürlicher Sprache angewendet. Es zeigt sich jedoch, daß diese Art der Datenstruktur und damit gleichzeitig die schon erwähnte vorteilhafte Trennung von Algorithmenschema und Suchstrategie auch für den Transfer genutzt werden kann. Die Analogien zwischen dem Transfer in einem hauptsächlich symbolisch arbeitenden, auf Merkmalstrukturen basierendem, Übersetzungssystem und einer syntaktischen Strukturanalyse sind zahlreich. Partielle Übersetzungen einer strukturellen Beschreibung können potentiell an etlichen Stellen während des Transfervorgangs genutzt werden. Die Speicherung in einer Art Tabelle von wohlgeformten Ausdrücken ist deswegen angeraten. Eine konzeptuelle Ausweitung in Richtung auf eine Chart ist ohne Probleme möglich: Inaktive Transferkanten beschreiben dann vollständige Übersetzungen von Konstituenten der Eingabe, aktive Transferkanten gehören zu partiellen Übersetzungen.

Neben solchermaßen streng am symbolischen Paradigma ausgerichteten Ansätzen sind andere Verfahren zur Übersetzung natürlicher Sprache in Gebrauch. Den prominentesten Vertreter stellt die statistische Übersetzung dar (Brown et al., 1990). Man versucht, durch den Vergleich von Texten in unterschiedlichen Sprachen, die denselben Inhalt haben, Wahrscheinlichkeiten für Korrespondenzen von Abschnitten beider Texte zu finden. Die wahrscheinlichste Abbildung eines Quelltextes, gegeben ein Modell für die Entsprechungen, wird als Übersetzung angeboten. Die Schwierigkeit liegt darin, Wörter und Phrasen einander korrekt zuzuordnen (vgl. z.B. Gale und Church, 1991; Fung und Church, 1994). Eine besondere Komplexität bieten dabei Mehrwortlexeme, die u.U. in einer Sprache diskontinuierlich repräsentiert sind; außerdem müssen Unterschiede in der Anzahl der Wörter, durch die eine Phrase ausgedrückt wird, berücksichtigt werden (Wu, 1995a). Melamed (1998) verzichtet deswegen auf jegliche wortübergreifende Korrespondenz und geht davon aus, daß in der Regel ein Wort durch genau ein Wort in einer anderen Sprache zu ersetzen ist. Dadurch unvermeidlich auftretende Fehler werden mit Hilfe eines Geräusch-Modells geglättet. Zusätzlich nutzt er Wissen über Wortklassen (offen bzw. geschlossen) aus, um die Präzision der Übersetzungen zu erhöhen. Er erreicht damit Erfolgsquoten von über 90% für Übersetzungen von Textstellen aus Online-Bibeltexten.

[28] Dieses Beispiel ist von Beskow (1993) adaptiert.

Der bei Melamed (1998) bereits angeklungene Trend, sich nicht ausschließlich auf eine statistische Modellierung von Textäquivalenzen auf der Basis von Oberflächenrepräsentationen zu verlassen, wird ebenfalls in anderen Projekten verfolgt. Dabei reicht das Spektrum von der Benutzung einer Grammatik zur Verbesserung der statistischen Komponente (Wu, 1995b) über die ausgewogene Nutzung von Wissen aus beiden Domänen (Knight et al., 1994; Rayner und Carter, 1997) bis zur Verwendung eines statistischen Sprachmodells zur Verbesserung der Ausgabequalität eines symbolischen Übersetzungssystems (Brown und Frederking, 1995). Seit einiger Zeit werden statistische Ansätze außerdem zur Übersetzung gesprochener Sprache (Vogel, Hahn, und Branigan, 1996; Niessen et al., 1998) erprobt, genauso wie konnektionistische Verfahren (vgl. z.B. Wang und Waibel, 1995).

Ein weiterer korpusbasierter Ansatz besteht darin, das Wissen um übersetzte Äußerungen für zukünftige Bearbeitungen auszunutzen (*memory-* oder *example-based translation*) (Sato und Nagao, 1990). Die wesentliche Wissensquelle besteht hier aus einer großen Datenbasis von Übersetzungen, die im Vorwege aus bilingualen Korpora erhoben werden kann. Anders als rein statistische Modelle erscheinen beispielgetriebene Systeme jedoch flexibler, da einerseits mit einfachen Mitteln Wissen in Form weiterer Übersetzungspaare hinzugefügt werden kann, ohne ein erneutes Training durchführen zu müssen. Andererseits lassen sich *memory*-basierte Systeme in Richtung auf eine abstraktere Modellierung erweitern, indem nicht ausschließlich Paare von Oberflächenrepräsentationen als Grundlage der Übersetzung dienen, sondern z.B. Variablen enthaltende Muster, die dann trivialen, nichtrekursiven Transferregeln entsprechen (Juola, 1994). Außerdem ist auf diesem Wege eine Integration mit anderen linguistischen Verarbeitungsebenen möglich (Furuse und Iida, 1992).

Unserer Einschätzung nach sind hybride Paradigmen zur Übersetzung gesprochener Sprache die erfolgversprechendsten. Sie garantieren durch eine symbolische Komponente, daß auch bisher ungesehene Eingaben verarbeitet werden können, besitzen aber gleichzeitig die Fähigkeit, häufig auftretende kontrastive Kollokationen effizient zu behandeln. Wir werden uns für die Implementation im Rahmen der vorliegenden Arbeit jedoch auf einen rein symbolischen Ansatz beschränken.

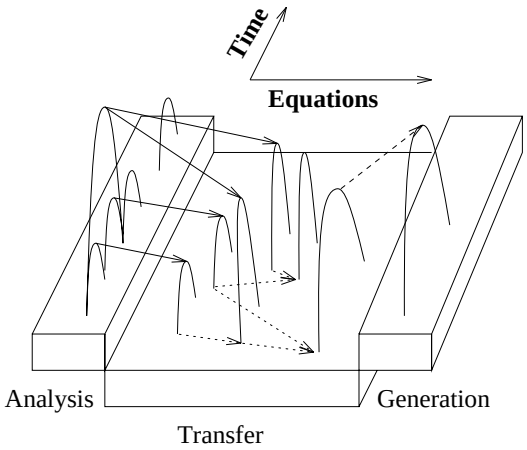


Abbildung 4.19: Eine Transferchart für strukturelle Übersetzungen

Eine Voruntersuchung zu der hier durchgeführten Implementation modelliert einen strukturellen Transfer basierend auf syntaktischen Beschreibungen (Amtrup, 1995a). Die dort verwendete Transferchart ist eine Erweiterung der für die Strukturanalyse genutzten (vgl. Abb. 4.19). Jede inaktive Syntaxkante, die eine vollständige strukturelle Analyse einer Konstituente enthält, fungiert als Wurzel eines Baumes von Transferkanten. Die Knoten des Baumes auf der ersten Ebene entstehen ausschließlich durch die Anwendung von Transferregeln, welche kontrastives grammatisches Wissen repräsentieren. Weitere Ebenen eines Baumes werden durch die Inkorporation bereits bestehender partieller Übersetzungen gebildet; dieser Vorgang wird durch die Existenz untergeordneter Transfergleichungen ausgelöst. Die Kombination konstituiert die fundamentale Regel des Transfers, nämlich die Erzeugung einer neuen Kante aus einer aktiven (die noch ungelöste Anforderungen für rekursiven Transfer enthält) und einer inaktiven (die eine vollständige Übersetzung einer Konstituente darstellt).

Eine solche Transferchart realisiert mithin eine zweidimensionale Struktur: Eine Dimension besteht in der zeitlichen Ausdehnung der zugrundeliegenden Eingabe, die andere in der Anzahl ungelöster Transfergleichungen.

4.8.2 Implementation von Transfer für MILC

Der in der vorliegenden Arbeit gewählte Ansatz für den Transfer verallgemeinert die gerade beschriebene Methodik in zweierlei Hinsicht: Zum einen basiert die Beschreibung von Übersetzungswissen nicht ausschließlich auf syntaktischen Kriterien, sondern bezieht in hohem Maße semantische Anteile mit ein. Zum zweiten wird von der strengen Adjazenzforderung abgesehen, die für eine syntaktisch-strukturell angelegte Transferkomponente angemessen ist, für die in MILC auftretenden syntaktisch-semantischen Beschreibungen jedoch nicht ausreicht.

Die Eingaben für den Transfer bestehen — neben Vererbungs- und Inhibitionshinweisen — aus inaktiven Analysekanten des Integrators. Diese enthalten als Kantenbezeichnungen Merkmalstrukturen, welche den syntaktisch-semantischen Gehalt einer Konstituente der Eingabe beschreiben. Das grundsätzliche Ziel besteht darin, eine entsprechende englische Repräsentation zu finden.

Der erste Schritt dazu ist, anwendbare Transferregeln für die Eingabe-Merkmalstruktur zu ermitteln. Ein Beispiel einer solchen Regel ist in Abb. 4.20 gezeigt. Analog zu einigen der in Abschnitt 3.2 dargelegten Methoden besitzt eine Regel zwei zentrale Merkmale, von denen eines (source) die quellsprachliche semantische Beschreibung enthält, eines (target) die zielsprachliche. Kontrastive Äquivalenzen zwischen beiden werden durch Koreferenzen ausgedrückt, die entweder direkt (Merkmal qtype in Abb. 4.20) oder durch Vermittlung der Merkmale subseq und concept-equivs agieren. Das Merkmal subseq definiert hierbei die notwendigen rekursiv auszuführenden Teilübersetzungen. Der Gebrauch entspricht den σ-Relationen von Kaplan et al. (1989), indem eine Entsprechung auf unterschiedlichen Seiten der Übersetzung gefordert wird. concept-equivs steuert weitgehend den Transfer lexikalischer Konzepte. Aus dem Beispiel ist etwa ersichtlich, daß die Übersetzung der Rollenfüller einer Relation (agent, experiencer, theme) Gegenstand rekursiven Transfers sind, während die des Konzeptes der Relation durch Rückgriff auf ein Transferlexikon

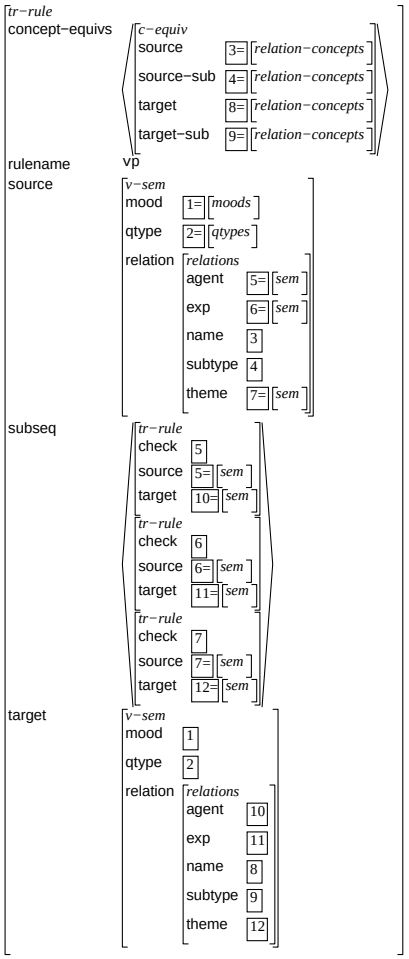


Abbildung 4.20: Eine Transferregel für verbale Ausdrücke

gesteuert wird. Die Elemente dieses Lexikons bilden deutsche semantische Konzepte auf deren englische Entsprechungen ab, was anhand von Abb. 4.21 demonstriert wird.[29]

$$\begin{bmatrix} c\text{-}equiv & \\ \text{source} & \text{sein-rel} \\ \text{source-sub} & \text{pred-recht} \\ \text{target} & \text{suit-rel} \\ \text{target-sub} & \text{nil} \end{bmatrix}$$

Abbildung 4.21: Transferlexikoneintrag für "recht sein" ("suit")

Die Anwendung einer Transferregel besteht zunächst darin, den semantischen Anteil der eingehenden Analysemerkmalstruktur mit dem source-Merkmal der Regel zu unifizieren. Dadurch werden z.B. die wesentlichen Unterscheidungen nach verbalen und nominalen Kategorien getroffen. Zusätzlich werden koreferente Inhalte in der zielsprachlichen Beschreibung und die quellsprachlichen Anteile der konzeptionellen Äquivalenzen und untergeordneten Transfergleichungen besetzt. Das Resultat besteht im Erfolgsfall in einer Transferkante, deren Status (aktiv oder inaktiv) festgestellt werden muß. Als aktiv wird eine Kante — entsprechend der Chartanalyse und der syntaktisch angelegten Vorstudie — charakterisiert, die eine oder mehrere offene, d.h. bislang ungelöste, Transfergleichungen besitzt. Um die Regelmenge für den Transfer klein zu halten und möglichst allgemeine Regeln formulieren zu können, ist jedoch bei der eben beschriebenen initialen Unifikation nicht gefordert, daß jede in einer Regel beschriebene Gleichung des subseq-Merkmals auch gefüllt ist. Vielmehr werden im Anschluß alle Elemente der Liste daraufhin geprüft, ob sie für die aktuell vorliegende Situation von Belang sind. Falls die Quellstruktur eines subseq-Mitglieds nicht verwendet wird, so wird es aus der Liste ausgeschlossen. Die Kategorisierung, ob die so entstandene Kante aktiv oder inaktiv ist, wird erst nachträglich ausgeführt.

Die fundamentale Regel des Chart-Transfers besteht nun darin, eine aktive mit einer passenden inaktiven Kante zu kombinieren. Ein Paar von Kanten muß im wesentlichen zwei Bedingungen erfüllen, um kombiniert werden zu können.

Sie müssen miteinander verträglich sein, was die Topologie des Graphen angeht. Für eine Transferchart syntaktischer Natur bedeutete dies, daß die aktive und die inaktive Kante adjazent sein mußten. Diese Bedingung konnte stets eingehalten werden, da der vollständige, aus Analysekanten gebildete, Teilgraph dem Transfer zur Verfügung stand. Für die vorliegende Implementation stellt sich die Situation jedoch anders dar. Zum einen werden nur solche Kanten an die Transferkomponente übertragen, die nach Kriterien des Integrators Zielstatus besaßen. Dies äußert sich darin, daß für etliche inaktive Kanten keine Subpfade zur Verfügung stehen, mit deren Hilfe ein einfacher kompositioneller Transfer unter Anwendung einer *Bottom-Up*-Strategie durchführbar wäre. Zum anderen werden etliche Modalpartikel während der Analyse als Artefakte spontansprachlicher Äußerungen behandelt, die zwar inkorporiert werden, um weiterspannende Konstituenten zu konstruieren, deren semantischer Gehalt jedoch leer ist.

[29] Exakter formuliert, vermitteln die Einträge zwischen an der deutschen Sprache angelehnten syntaktischen Darstellungen eines semantischen Konzepts und einer eher für englische Generierung geeigneten Form.

Eine offensichtliche Lösung bestünde darin, auf die Anwendung von Chart-orientierten Verarbeitungsmechanismen zu verzichten und einen herkömmlichen Transferansatz zu propagieren. Die Entscheidung hierfür würde allerdings vernachlässigen, daß trotz der eben geschilderten Umstände ein Wiedergebrauch von angefertigten Teilübersetzungen stattfinden kann und wünschenswert ist. Zudem verlöre man durch einen solchen Rückgriff Steuerungsmechanismen im Sinne einer Agenda der Chart.

Wir favorisieren statt dessen einen Ansatz, der die vorteilhaften Charteigenschaften erhält. Statt zu fordern, daß einer der Endknoten der aktiven Kante mit einem der Startknoten der inaktiven Kante identisch ist, und aktive Kanten weiterzuspannen, indem sie inaktive konsumieren, belassen wir die Spannweite der aktiven Kante so, wie dies durch die ihre Konstruktion auslösende Kante angegeben wurde. Das Verhältnis der beiden zu kombinierenden Kanten muß nun so geartet sein, daß die inaktive von der aktiven Kante überspannt wird. Da es sich auch innerhalb des Transfers um Hyperkanten handelt, verwenden wir den kleinsten Anfangsknoten ($\alpha_<(e)$) und den höchsten Endknoten ($\beta_>(e)$) beider Kanten zur Feststellung ihres Verhältnisses. Abb. 4.22 zeigt eine mögliche Anordnung von Kanten. **e1** stellt eine inaktive Analysekante dar, die vom Integrator an den Transfer geliefert wurde. Die Kante **e2** ist durch Anwendung einer Transferregel auf **e1** entstanden und besitzt folglich mit dieser identische Mengen von Start- und Endknoten. **e3** denotiert eine bereits vorher konstruierte inaktive Transferkante. Sie liegt vollständig innerhalb von **e2** und kann somit durch die fundamentale Regel mit ihr kombiniert werden.

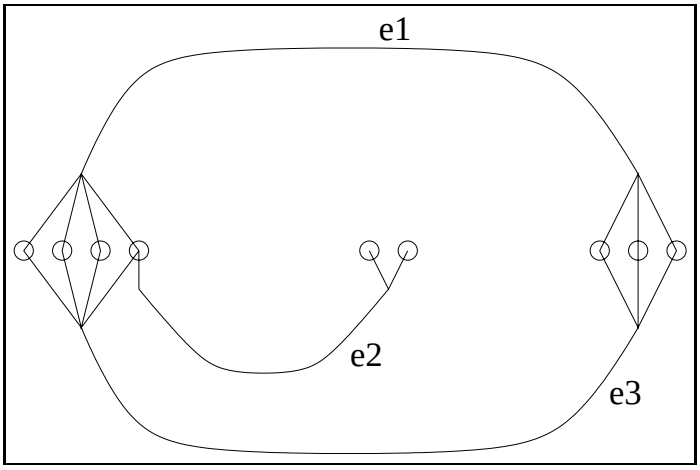


Abbildung 4.22: Topologie der fundamentalen Regel für den Transfer

Diese Art der Prüfung nimmt keine Rücksicht mehr auf die feinere Topologie des Graphen. So kann hier nicht mehr gewährleistet werden, daß eine inkorporierte inaktive Kante auch tatsächlich auf dem Pfad liegt, der durch die aktive Kante beschrieben wird. Dies ist allerdings nicht kritisch, da durch die anschließende Unifikiation, welche die zweite Bedingung zur Kombination darstellt, praktisch alle fehlerhaften Fälle ausgeschlossen werden. Die Merkmalstruktur der inaktiven Kante muß eine der untergeordneten Transfergleichungen der aktiven Kante erfüllen, d.h. sich mit ihr unifizieren lassen. Dadurch wird die Verträglichkeit der semantischen Beschreibungen sichergestellt.

Falls durch dieses Verfahren keine passende inaktive Kante für eine Transfergleichung gefunden werden kann, muß tatsächlich auf den herkömmlichen, rekursiven Transfer zurückgegriffen werden. Dazu werden die unerfüllten Gleichungen als initiale inaktive Kanten aufgefaßt, die der Grammatik zur Suche nach in Frage kommenden Regeln übergeben werden. Sind diese Kanten einmal in die Chart eingesetzt, kann der normale Verarbeitungszyklus wieder greifen.

Um eine inaktive Kante endgültig in die Chart aufnehmen zu können, müssen schließlich noch die Konzeptäquivalenzen aufgelöst werden. Dazu werden alle Kombinationen von Transferlexikoneinträgen gesucht, welche die Elemente des Merkmals concept-equivs erfüllen können. Für jede dieser Kombinationen wird eine neue Kante erstellt, indem die passenden Lexikoneinträge mit den Elementen der Konzeptäquivalenzen unifiziert werden.

Solchermaßen generierte inaktive Kanten sind schließlich auch Kandidaten, die potentiell an die Generierung weitergegeben werden können. Dies geschieht allerdings nur dann, wenn der Ursprung der Kante in der Eingabe durch den Integrator lag. Die Intention dabei ist, innerhalb des Transfers neu vorgeschlagene Analysen zur Durchführung rekursiven Transfers nicht als Zielkanten des Transfers zu kategorisieren, da sie nicht von der Integration als solche markiert wurden. Das Transfermodul liefert folglich vollständige Übersetzungen von vollständigen Analysen, die für das Sprachverstehen relevant waren.

4.9 Generierung

Im Rahmen von MILC wird keine volle Generierung durchgeführt, sondern eine Oberflächenrealisierung, wenn man in der klassischen Unterteilung der Erzeugung einer sprachlichen Form aus kommunikativen Zielen bleibt (vgl. Horacek, 1993). Diese Segmentierung besteht aus der strategischen, inhaltsbestimmenden Textplanung (dem *"What To Say"*), einer taktischen, formbestimmenden Textplanung (dem *"How To Say"*) und der eigentlichen Oberflächengenerierung. Wenn die Eingaben für den Generator von MILC betrachtet werden, ist dies einsichtig, stellen sie doch semantische, bereits recht oberflächennahe, Beschreibungen des Inhalts einzelner Äußerungen dar. Die Planung eines längeren Diskurses ist mithin nicht Gegenstand des hier untersuchten Verfahrens. Auch der formbestimmende Anteil an der Generierung ist innerhalb von MILC bescheiden dimensioniert; Die Eingabe ist zwar unterspezifiziert, was die Auswahl an zur Generierung heranzuziehenden Lexeme angeht, eine echte Wort*wahl* findet jedoch nicht statt, vielmehr werden alle zur Semantik passenden Wörter ausgewählt.

Nachdem ein Generierungssystem den zu versprachlichenden semantischen Gehalt akzeptiert hat, gilt es, die Reihenfolge festzumachen, in der die einzelnen Unterstrukturen behandelt werden sollen; außerdem muß eine geeignete Repräsentation für die im Laufe der Verarbeitung auftretenden partiellen Ergebnisse gefunden werden. Im Prinzip ist folglich die Bestimmung eines Algorithmenschemas (die Struktur) sowie die Festlegung einer Strategie zur Bearbeitung (die aus dem Schema einen Algorithmus werden läßt), notwendig.

Kopfgesteuerte Strategien sind die weitestverbreiteten im Rahmen der Generierung. Es liegt recht nahe, sich zunächst dasjenige Lemma zu suchen, das den Inhalt (*"semantic head driven"*, vgl. Shieber et al., 1989, Shieber et al., 1990, Kikui, 1992) oder die grammatische Struktur der Äußerung (*"syntactic head driven"*, vgl. König, 1994) dominiert. Ausgehend von dieser Information kann weiteres Material aus der Eingabe in die korrekte Position integriert werden. Der Vorgang terminiert, wenn alle Eingabeelemente berücksichtigt wurden (was nicht unbedingt die offene Realisierung in sprachlicher Form impliziert).

Was die Einbettung in eine Datenstruktur zur Generierung angeht, so sind Chart-basierte Verfahren auch hier als günstige Grundlage zur Verwaltung der bis zu einem jeweiligen Zeitpunkt durchgeführten Operationen gebräuchlich. Auf diese Weise kann nicht nur die redundante mehrfache Generierung von Teilstrukturen vermieden werden, auch die Verwendung unterschiedlicher Suchstrategien ist analog zur strukturellen Analyse leicht möglich. So ist beispielsweise die Einführung einer bidirektionalen Generierung denkbar, was die Bildung von bereits realisierten Inseln in der Oberflächendarstellung ermöglicht (Haruno et al., 1993). Zu beachten ist lediglich, daß die direkte Abbildung von Chartknoten auf Zeitpunkte problematisch ist, da die zeitliche Ordnung der zielsprachlichen Struktur erst im Laufe der Anwendung des Generators entsteht und nicht a priori verfügbar ist, wie im Falle der Analyse. Kay (1996) verwendet deshalb Teilmengen von bearbeiteten semantischen Teilstrukturen als Grundlage für die Knotenzuordnung (s.o.).

Die Art und Weise, wie der Generator seine Eingabe konsumiert, stellt ebenfalls ein Unterscheidungsmerkmal für verschiedene Ansätze bereit. Die meisten Systeme vertrauen darauf, daß die

vollständige semantische Repräsentation vor Beginn der Generierung bereit steht. Dies ist besonders für kopfgesteuerte Verfahren (seien sie am syntaktischen oder semantischen Kopf orientiert) einsichtig, da diese ja auf das Vorhandensein des Kopfes angewiesen sind, um den strukturellen Rahmen der Ausgabeäußerung aufzubauen. Selbst chartbasierte und an flachen semantischen Beschreibungen orientierte Methoden wie die von Kay (1996) sind nicht speziell für eine inkrementelle Eingabe ausgelegt. Dennoch ist eine inkrementelle Generierung sinnvoll und durchführbar, wie Reithinger (1992) mit dem System POPEL zeigt (vgl. hierzu auch Finkler und Schauder, 1992, sowie Kilger, 1994).

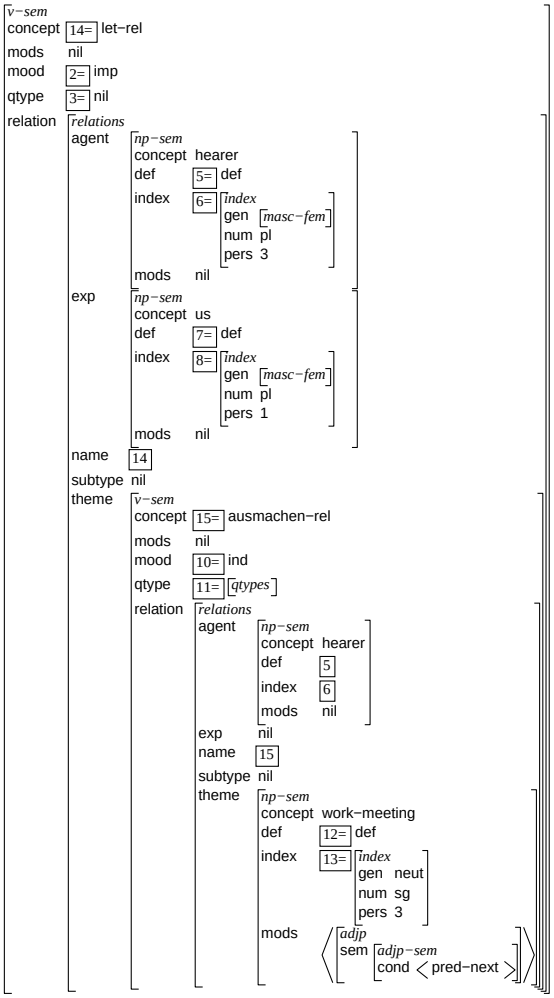


Abbildung 4.23: Ein Teil der Generierungseingabe für “lassen Sie uns das nächste Arbeitstreffen vereinbaren”

Eine interessante Variante von Generierungsalgorithmen bildet die sogenannte *Shake-and-Bake*-Generierung (Beaven, 1992; Brew, 1992; Whitelock, 1992). Hier wird die Eingabe als eine Menge von untereinander unkorrelierten Einheiten angesehen, die solange zu größeren Zeichen kombiniert

werden, bis alle Eingabedaten berücksichtigt wurden und die zugrundeliegende Grammatik, die die Kombinationen lizensiert, dabei nicht verletzt wurde. Der wesentliche Nachteil der frühen Fassung von *Shake-and-Bake* ist die exponentielle Komplexität, welche die Anwendung in Systemen bisher stark eingeschränkt hat. Allerdings ist von Poznański, Beaven, und Whitelock (1995) ein polynomieller Generierungsalgorithmus im Rahmen dieses Ansatzes veröffentlicht worden.

Der Generator von MILC vereinigt einige der bisher geschilderten Eigenschaften: Er ist inkrementell, Chart-basiert und kann kopfgesteuert arbeiten. Wir werden die verschiedenen Verarbeitungsaspekte an dem Beispiel in Abb. 4.23 verdeutlichen, das einen Teil der Eingabe darstellt, die der Generator vom Transfer für den gewählten Satz geliefert bekommt.

Die Merkmalstruktur stellt insofern nur einen Teil der Eingabe dar, als sie die äußerungsüberspannende Kante annotiert. Sämtliche Analysen, die einen kleineren Ausschnitt repräsentieren, sind hier weggelassen. Der Typ der Merkmalstruktur (*v-sem*) instruiert den Generator, in seiner Grammatik nach Regeln zu suchen, die als linke Seite ebenfalls eine Merkmalstruktur dieses Typs tragen.

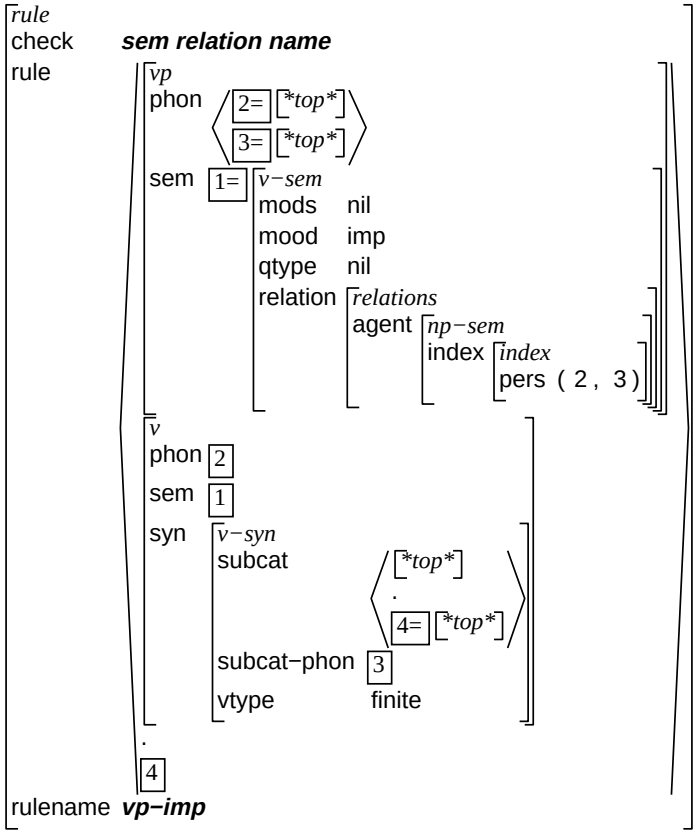


Abbildung 4.24: Generierungsregel für Verben im Imperativ

Das der Ableitung zugrundeliegende Modell besteht wiederum in Phrasenstrukturregeln (vgl. Abb. 4.24), die allerdings in zur Analyse unterschiedlicher Art behandelt werden. Zum einen ist natürlich

die Richtung der Ableitung entgegengesetzt. In der Strukturanalyse werden stets die Merkmalstrukturen der linken Seite aus den untergeordneten Strukturen, die auf der rechten Seite auftreten, komponiert. Für die Generierung ist die Situation die, daß die eine Phrase definierende semantische Beschreibung vorhanden ist, und daß durch die sukzessive Abarbeitung der Elemente der rechten Seite der Regel die syntaktische und phonologische Komponente der linken Seite konstruiert wird. Das ermöglicht eine weitergehende Selektion der anzuwendenden Regeln, als dies ausschließlich durch den Typ der Merkmalstruktur möglich ist. Das Ziel ist dabei, die unnötige Anwendung von Regeln zu verhindern und somit die Anzahl der durchzuführenden Unifikationen zu reduzieren. Das Merkmal check in der in Abb. 4.24 gezeigten Regel definiert einen Pfad (oder eine Liste von Pfaden), deren Werte innerhalb der zu generierenden Struktur nicht leer sein dürfen. Diese Prüfung wird etwa dazu verwendet, den Versuch der Generierung von Modifikatoren von Phrasen zu verhindern, wenn derartige Modifikatoren überhaupt nicht in der semantischen Eingabe präsent sind.

Der zweite Bereich, in dem eine Einflußnahme durch den Autor der Grammatik möglich ist, ist die Reihenfolge der Elemente auf der rechten Seite einer Regel. Diese Elemente bestimmen nicht — wie in der Analyse — die Oberflächenreihenfolge von Wörtern. Diese wird lediglich durch das phon-Merkmal auf der linken Seite einer Regel determiniert. Das führt dazu, daß auf der rechten Seite einer Regel die Reihenfolge nach Ökonomieprinzipien ausgerichtet sein kann. Das bedeutet, daß im Normalfall das Element, daß die stärksten Einschränkungen produziert, als erstes generiert werden wird. So ist es etwa nicht sinnvoll, innerhalb einer verbalen Struktur zuerst das Subjekt aufzubauen und danach das finite Verb, da dieses weitere Einschränkungen für das Subjekt fordern kann. Auf diese Art kann eine Kopfsteuerung flexibel in die Ableitung integriert werden.

Ein weiterer Vorteil dieser Strategie ist, daß Regeln für die Subkategorisierung von Verben auf eine sehr allgemeine, elegante Art formuliert werden können. Der Formalismus ermöglicht die Angabe eines Paares (die den Kopf und den Rest einer Liste symbolisiert, entsprechend einer Lisp cons-Zelle oder der Formulierung [H|T] in Prolog). Damit kann je nach Subkategorisierungsliste des Verbs eine Generierungsregel erzeugt werden, die genau so viele Komplemente enthält wie das Verb sie fordert. Diese Möglichkeit ist ebenfalls in der in Abb. 4.24 gezeigten Regel benutzt worden.

Der hier vorgestellte Generator arbeitet inkrementell. Allerdings ist die Inkrementalität nicht bezogen auf den steten Fluß von Teilen einer semantischen Beschreibung, die nacheinander in das System eingehen, wie dies etwa in dem von Finkler (1996) beschriebenen Ansatz geschieht; vielmehr wird auch in der Generierung eine Inkrementalität aufgrund der zeitlichen Ausdehnung der Eingabeäußerung benutzt. Die semantischen Beschreibungen treffen (in der Regel) in der Reihenfolge ihrer Endzeitpunkte ein. Der Mechanismus der Chart-Analyse wird dann dazu benutzt, bereits vorgehaltene Generierungsabschnitte in weiter gespannte Analysen einzusetzen. Immer, wenn eine Kante die Generierung vom Transfer aus erreicht, wird die Grammatik konsultiert. Sie liefert Regeln, die mögliche Kandidaten für die Oberflächenrealisierung der Eingabe beschreiben. Die Suche nach anwendbaren Regeln wird gesteuert durch den Typ der eingehenden Merkmalstruktur sowie durch die im Vorwege zu prüfenden Merkmale. Wurden passende Regeln gefunden, so wird deren linke Seite mit der Eingabe unifiziert. Die durch diesen Prozeß neu entstandenen Kanten werden in die Chart eingesetzt.

Abhängig von der Anzahl der noch zu bearbeitenden Elemente der rechten Seite einer Generie-

rungsregel werden Kanten als aktiv bzw. inaktiv klassifiziert. Die zu aktiven Kanten gehörenden Regeln wurden noch nicht vollständig befriedigt. Beim Hinzufügen einer solchen Kante zur Chart wird innerhalb der Spanne der Kante nach inaktiven Kanten gesucht, die das nächste zu verarbeitende Element der rechten Seite füllen können; dies wird durch Unifikation überprüft. Im Erfolgsfall wird jeweils eine neue Kante konstruiert, die demselben Zyklus unterworfen wird. Außerdem wird für ein Nichtterminal auf der rechten Seite erneut der Vorgang der Suche innerhalb der Grammatik angestoßen, für den Fall, daß es noch weitere Ableitungen außer den bisher gefundenen geben könnte. So werden nach und nach aktive und inaktive Kanten kombiniert, bis schließlich eine inaktive Kante gefunden wird, die die Eingabesemantik vollständig realisiert. Analog dazu versucht der Generator, inaktive Kanten in größere Kontexte einzubetten. Dies ist so implementiert, daß versucht wird, aktive Kanten zu finden, die eine eingesetzte inaktive Kante überspannen und kompatibel mit dieser sind. In diesem Fall können beide kombiniert werden.

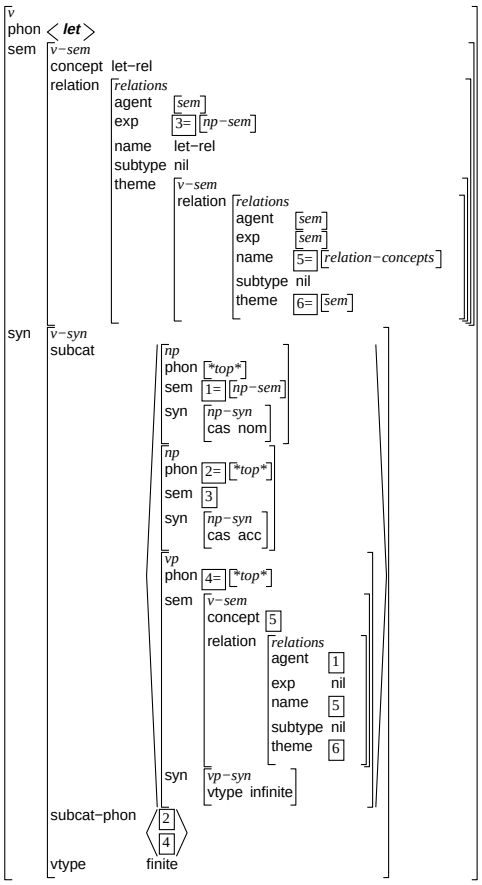


Abbildung 4.25: Generierungslexikoneintrag für “let”

Einen Sonderstatus nimmt die Bearbeitung terminaler Elemente innerhalb von Grammatikregeln ein. Terminale sind mit lexikalischen Typen assoziierte Merkmalstrukturen, wie etwa die Substruktur des Typs *v* in der in Abb. 4.24 gezeigten Regel. Muß ein solches Regelfragment generiert werden,

so findet ein Lexikonzugriff auf der Basis des semantischen Konzepts statt, das in der Beschreibung gefordert wird. Das Generierungslexikon ist mit Hilfe von Vollformen spezifiziert, es wird keine morphologische Generierung durchgeführt. Für jedes passende Wort des Lexikons wird eine neue Kante generiert. Abb. 4.25 zeigt einen Lexikoneintrag, der für das hier gewählte Dialogteilstück relevant ist. Hier ist zugleich die Reihenfolgesteuerung der Phonologie der subkategorisierten Konstituenten sichtbar. Das listenwertige Merkmal subcat-phon beschreibt die Oberflächenabfolge der zum Verb gehörenden Äußerungsteile.

Werden im Laufe der Verarbeitung des Generators inaktive Kanten erzeugt, deren Beschreibung so geartet ist, daß sie ausgegeben werden sollen (der Typ der Merkmalstruktur muß dazu von einem ausgezeichneten Typen *gener-cat* subsumiert werden), dann wird die Kante entsprechend markiert. Die Komponente hält ständig einen Hypergraphen von möglichen Oberflächenrealisierungen von Teilen der übersetzten Äußerung bereit. Wird eine neue Kante hinzugefügt, so wird mit Hilfe des inkrementellen Algorithmus 14 auf Seite 64 der kürzeste Weg durch den Generierungsgraphen gesucht und bei Änderungen zum vorhergehenden Status ausgegeben. Das Gewicht einer Kante wird dabei durch die akustische Bewertung der Eingabewörter bestimmt. Zusätzlich wird ein Übergang von einem Knoten des Graphen zum nächsten mit einer Strafe belegt, falls zwischen beiden keine Kante verläuft. Auf diese Weise wird ein Weg durch den gesamten Graphen garantiert. Eine weitere Modifikation der Bewertung abhängig von der Länge der Oberflächenrepräsentation sorgt dafür, daß möglichst alle Beiträge zur Semantik overt realisiert werden.

Die Art, in der die Suchinformation innerhalb der Generierung vorgehalten wird, bewirkt eine inkrementelle Ausgabe der zielsprachlichen Äußerung im Laufe der Verarbeitung. Tab. 4.3 präsentiert die Ausgabe des Gesamtsystems für die deutsche Eingabe "lassen Sie uns den nächsten Termin ausmachen", unter deren Benutzung alle Beispiele in diesem Kapitel konstruiert wurden.

Tabelle 4.3: Ein Beispiel für die Ausgabe des Generators

you
you |we
you |we |it
you |we |it |work meeting
you |we |it |schedule work meeting
you |we |the next work meeting
you |we |schedule the next work meeting
let us schedule the next work meeting

Man sieht deutlich, wie länger werdende Teile der Eingabeäußerung sukzessive ins Englische übersetzt werden. Das Verb im Imperativ "lassen" etwa wird erst nach Eintreffen aller subkategorisierten Konstituenten realisiert. Erst dadurch wird auch die Information berücksichtigt, daß den nicht demonstrativ zu übersetzen ist, sondern als Artikel.

4.10 Visualisierung

Die Visualisierung der im Laufe eines Verarbeitungszyklus entstehenden Information ist für jedes größere System unerläßlich, sei es, um Fehler in der Programmierung zu finden, oder um die Ergebnisse der Arbeit in eingängigerer Form zu präsentieren, als dies etwa durch seitenlange textuelle Repräsentationen möglich ist. Die besonders dringende Notwendigkeit für MILC, mit einer Visualisierung ausgerüstet zu sein, resultiert aus mehreren Eigenschaften:

- MILC ist ein verteiltes System, bestehend aus im Moment sechs Komponenten zur Analyse des eingehenden Sprachsignals. Die Fehlersuche in solchermaßen verteilten, konkurrenten Systemen ist notorisch schwierig, durch die relative Unsicherheit über die genaue Reihenfolge, in der manche Aktionen ablaufen werden, sind eine ganze Reihe von Fehlerquellen gegeben.
- MILC behandelt gesprochene Sprache. Dies hat eine sehr hohe Anzahl von elementaren Eingaben (Worthypothesen) zur Folge, deren Anzahl durch die Hypergrapheneigenschaften zwar stark reduziert wird, jedoch nicht in einem Bereich angesiedelt ist, der für Tests mit geschriebener Sprache adäquat ist. Darüberhinaus ist die Zuordnung von Kanten zueinander sehr viel einfacher mit Hilfe einer graphischen Schnittstelle zu bewerkstelligen als z.B. durch den Vergleich von Knotennummern der Chart.
- MILC arbeitet mit komplexen Datenstrukturen. Die Entwicklung von Grammatiken und Lexika mit Hilfe von textbasierten Werkzeugen ist schnell und oft ausreichend. Dies liegt zum einen an der Eigenschaft vieler Formalismen (so auch des Formalismus von MILC), knappe Formulierungen in Form von Makros anzubieten, zum anderen ebenfalls daran, daß die erzeugten Strukturen (Lexikoneinträge und Regeln) hochgradig unterspezifiziert sind. Wird jedoch auf eine Ausgabe von Merkmalstrukturen Wert gelegt, die den Endzustand einer Bearbeitung repräsentieren, so ist die simple Textausgabe unbefriedigend. Die graphische Aufbereitung hilft durch unterschiedliche Zeichensätze und durch die graphische Klammerung, schnell Zusammenhänge einordnen zu können. Die Visualisierung von MILC ist jedoch kein Entwicklungswerkzeug für linguistische Wissensquellen, so fehlt die Möglichkeit zur Modifikation genau so wie die Manipulation der Ausgabe durch das Falten von bestimmten Merkmalen, wie sie z.B. in dem System Fegramed (Kiefer und Fettig, 1993) realisiert ist.

Die Wahl der Implementierungssprache für die Visualisierung fiel aus Effizienzgründen (sowohl was die eigentliche Implementierung, als auch, was die Ausführungsgeschwindigkeit angeht) auf Tcl/Tk (Ousterhout, 1994). Diese Sprache bildet das Grundgerüst für das Erscheinungsbild der Visualisierung, das in Abb. 4.26 als eine Momentaufnahme der Verarbeitung dargestellt ist. Gezeigt wird ein Ausschnitt des momentan zu analysierenden Hypergraphen. Die Eigenschaft von Hyperkanten, zahlreiche Start- und Endknoten besitzen zu können, wird durch die strahlenförmige Verbindung von Kanten zu Knoten deutlich gemacht. Eine der Kanten wurde durch Selektion mit der Maus zur detaillierten Darstellung ausgewählt. Die Eigenschaften der Kante sind in der unteren Hälfte des Fensters dargestellt.

Zu diesen Eigenschaften gehört die ursprüngliche Eingabe (“Herr Pfitzinger”), die Angabe des überspannten Anteils der Chart, und die möglichen Annotationen von unterschiedlichen Komponenten. Aus diesen Möglichkeiten (die in der Liste im linken Teil angegeben sind) wurde wiederum eine Beschreibung, nämlich die zugehörige zielsprachliche Oberflächenrepräsentation (“Mr. Pfitzinger”) ausgewählt, was zur Darstellung der gewünschten Merkmalstruktur führt. Die Darstellung der Hyperkanten und der Merkmalstruktur sind nicht direkt mit Tcl/Tk implementiert, sondern vielmehr in einer unterliegenden Schicht in C programmiert.

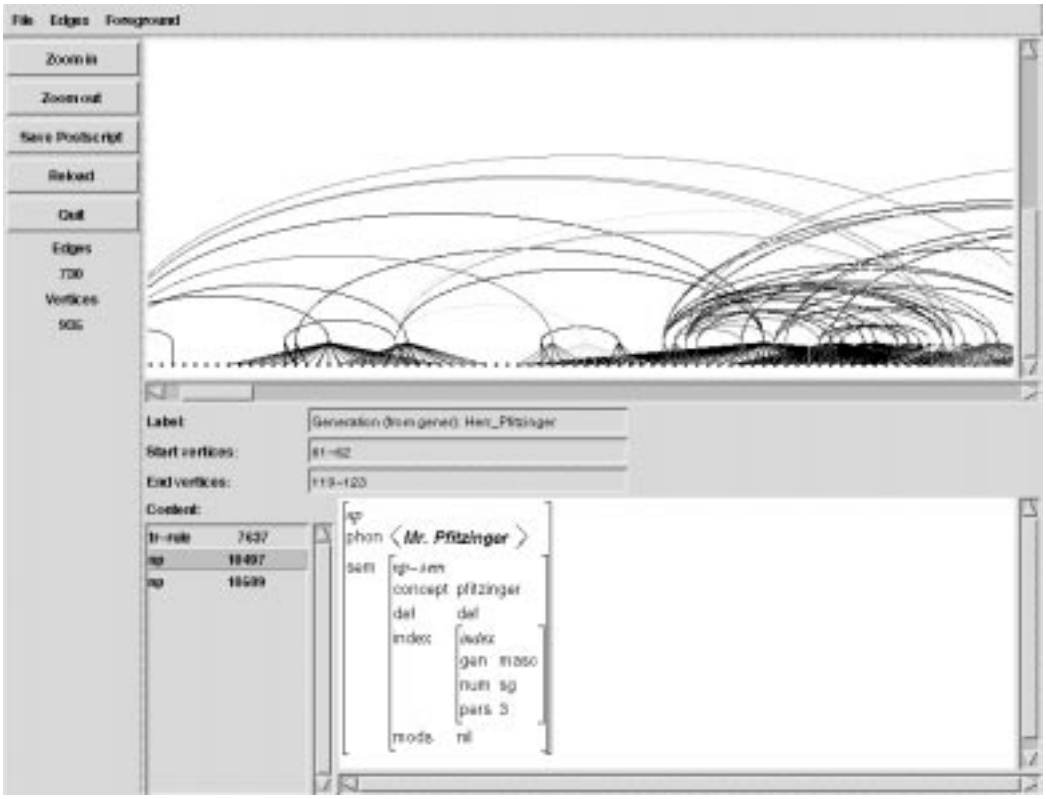


Abbildung 4.26: Ein Schnappschuß der Verarbeitung mit MILC

Die inkrementelle graphische Darstellung von Wortgraphen in übersichtlicher Weise hat sich als erstaunlich schwierig erwiesen (Florey, 1998). Insbesondere die inkrementelle Bildung einer graphischen Repräsentation und die Beibehaltung gewisser Layoutaspekte (weiter rechts in der Zeit liegende Knoten sollten auch in der Visualisierung weiter rechts liegen) bereitet bei Anwendung gängiger Algorithmen zum Layout von Graphen Probleme. Für MILC ergibt sich insofern eine günstige Situation, als durch den Übergang zu Hypergraphen die Anzahl der Kanten drastisch reduziert wird. Insofern ist es hier sinnvoll und vertretbar, eine relativ einfache Zeichnung von Knoten (linear und äquidistant in der x-Achse) und Kanten (als Ausschnitte von Ovalen) vorzusehen.

4.11 Potentielle Erweiterungen

MILC ist ein vollständiges System zur Übersetzung von deutschen Einzeläußerungen spontan gesprochener Sprache in Englisch. Der Umfang und die Arbeitsweise wurden in den vorangegangenen Abschnitten erläutert. Dennoch ist in etlichen Einzelaspekten eine Verbesserung bzw. Erweiterung der Funktionalität denkbar und sinnvoll. Dies entspricht vollkommen der Absicht hinter der Implementierung, die zuerst und vor allem darauf abzielte, den Übersetzungsvorgang insgesamt inkrementell zu gestalten, gleichzeitig jedoch ein Experimentiersystem zur Verfügung zu stellen, an dem weitere Untersuchungen zur Architektur sprachverarbeitender Systeme im allgemeinen und zur Rolle der Inkrementalität und Interaktivität im besonderen durchgeführt werden können; ein in diesem Sinne offenes System also. Wir werden in diesem Abschnitt auf einige offensichtliche, unmittelbar implementierbare, Modifikationen hinweisen. Desgleichen wird eine aus zwei Teilen bestehende Erweiterung vorgeschlagen, die sowohl den Bereich der zu betrachtenden Phänomene innerhalb einer Äußerung als auch die Abdeckung über isolierte Äußerungen hinaus betrifft. Schließlich diskutieren wir einige Fragen, die im Zusammenhang mit der potentiellen Etablierung einer Any-Time Analyse entstehen.

Bei den einfacheren Erweiterungen handelt es sich zumeist um lokale Modifikationen an einzelnen Komponenten, die sich direkt auf die Performanz oder Qualität eines Moduls auswirken können. Hier sind als Beispiele zu nennen:

- **Sprachmodell**. Bei dem im Rahmen von MILC verwendetem Sprachmodell handelt es sich um ein Bigramm, das jeweils lediglich die Wahrscheinlichkeit der Adjazenz zweier aufeinanderfolgender Wörter berechnet. Da jedoch ebenfalls längere Ketten von Wörtern konstruiert werden, ist die Benutzung zumindest eines Trigramms angeraten.
- **Erweiterte Idiomatik**. Die bisher in MILC vorgesehene Behandlung von Idiomen konzentriert sich auf Routineformeln, die quasi lexikalisch verstanden werden, indem sie ohne Flektion oder strukturelle Abweichung auftreten. Gerade im Bereich der Terminabsprache sind jedoch Idiome, die einen gewissen Grad an Freiheit in der Gestaltung besitzen, mit einiger Frequenz vertreten, so sind Terminvorschläge sehr häufig idiomatisch realisiert, sie enthalten jedoch ein Datum und manchmal eingestreute Modalpartikel, die mit einem starr lexikalisch orientierten Ansatz nicht adäquat verarbeitet werden können (Erbach und Krenn, 1994). Die notwendige variable Modellierung von Idiomen kann z.B. mit Hilfe von endlichen Automaten erreicht werden, die Zugriff auf Kategorieninformation für bestimmte Wörter zulassen. Auch optionale Bestandteile, wie Partikel, sind so darstellbar (Zajac, 1998).
- **Grammatiken**. Die Regeln der jeweils verwendeten Grammatiken sollten um Wahrscheinlichkeiten erweitert werden, etwa analog zu der von Weber (1995) durchgeführten Methode. Dieses Verfahren, die Bewertung einer Ableitung auch mit Hilfe statistischer Information über die Grammatik zu bestimmen, ist nicht auf die Strukturanalyse beschränkt und kann folglich auch für den Transfer und die Generierung übernommen werden.
- **Auswahl**. Die Bewertungsfunktion, die im Generierungsgraphen nach der besten Hypothesenkette sucht, um diese auszugeben, kann verbessert werden. Dazu bieten sich zunächst

zwei sinnvolle Aspekte an:

  - Die Bewertung kann zusätzlich ein englisches Sprachmodell benutzen, um zunehmend "glattere" Äußerungen zu produzieren.
  - Die generierten Äußerungen können zu einem gewissen Teil erneut strukturell analysiert werden, um ein höheres Maß an Kohärenz zwischen den Fragmenten zu erzeugen.[30]

- **Generierungsgrammatik**. Die Regeln der für die Generierung erzeugten Grammatik werden in eine Hierarchie integriert. Das resultiert in einer besseren Kontrolle über deren Anwendung und stellt sicher, daß Spezialfälle stets vorrangig behandelt werden. Zugleich kann die Effizienz des Generators gesteigert werden, indem allgemeine Generierungsregeln nur dann in Betracht gezogen werden, wenn keine der Spezialisierungen anwendbar ist.

- **Chartstruktur**. Die Analysealgorithmen zumindest verlangen in der augenblicklichen Fassung des Systems die Adjazenz von Wörtern und Konstituenten, um sie in einen größerem Zusammenhang zu integrieren. In Abschnitt 4.6.2 wurde die Möglichkeit diskutiert, gewisse Kategorien von Kanten zu kennzeichnen, die ohne Modifikation der linguistischen Beschreibung konsumiert werden können, in Abschnitt 2.6 wurde die Einführung von Lücken in der Chart beschrieben. Eine Kombination beider Verfahren könnte dazu führen, daß genügend Material in der Chart übersprungen werden kann, um eine kontinuierliche Analyse zu erreichen, gleichzeitig jedoch den Aufwand nicht übermäßig zu steigern.

- **Suchstrategie**. Linksverbundene Graphen besitzen Knoten, von denen aus keine weitere Worthypothese mit genügend hoher Konfidenz gefunden werden kann. Die Suchstrategie (zumindest innerhalb der Module zur partiellen Strukturanalyse und zur Äußerungsintegration) kann modifiziert werden, um dies auszunutzen. Zu diesem Zweck wird eine verzögerte Auswertung eingeführt, die Aufträge für Kanten nur dann in die globale Agenda einsetzt, wenn dieser Knoten mindestens eine ausgehende Kante hat. Auf diese Weise ist gesichert, daß zumindest die einfachen Sackgassen (solche, die nur aus einer Kante bestehen) nicht berücksichtigt werden. Allerdings muß gesagt werden, daß diese Art der Suche mit der strengen Links-Rechts-Inkrementalität interferiert, da nicht vorhergesagt werden kann, wie lange auf eine Fortsetzung gewartet werden muß.

- **Parallelisierung**. Die Struktur und Implementierung des Systems MILC realisieren eine weitestgehende intermodulare Parallelisierung aller Komponenten. Wir haben jedoch bisher davon abgesehen, genauer auf die Optionen zur intramodularen Parallelisierung einzelner Komponenten einzugehen. Es zeigt sich, daß eine Konzeption mit Mehr-Ebenen-Charts und einem verschiebungsinvarianten Formalismus eine nahezu ideale Grundlage für die parallele Bearbeitung graphenartiger Strukturen ist. Wir haben bereits in einer früheren Untersuchung (Amtrup, 1992; Amtrup, 1995c) gezeigt, daß die parallele Strukturanalyse unter Berücksichtigung eines chartbasierten Ansatzes potentielle Verbesserungen in Hinsicht auf die Performanz eines Strukturanalysemoduls mit sich bringt. Allerdings wurde in Amtrup (1995c) lediglich die

[30] Diese Maßnahme läßt sich in hohem Maße erweitern, indem Rückkopplungen zu anderen Modulen eingeführt werden. Das ultimative Ziel kann hier sein, eine Selbstüberwachung zu modellieren, wie sie Konferenzdolmetscher durchzuführen scheinen, indem sie den von ihnen produzierten Übersetzungen zuhören.

Strukturanalyse geschriebener Eingabe untersucht. Spilker (1995) untersucht das Verhalten bezüglich der Verarbeitung gesprochener Sprache, erreicht jedoch mit einem agendabasierten Parallelisierungsverfahren auf der Grundlage eines speichergekoppelten Mehr-Prozessor-Rechners keine befriedigende Resultate. Von diesem beiden Arbeiten ausgehend, scheint die Verteilung von Bestandteilen der Analyse in Abhängigkeit von Teilpfaden eines Wortgraphen unter Verwendung lose gekoppelter Multiprozessoren angemessener. Eine sich unmittelbar anbietende Variante ist hier, die von einem Knoten ausgehenden Worthypothesenkanten als Initiatoren verteilter Verarbeitung anzunehmen. Bis zu einem gewissen Grade wird der Hypothesengraph in einen Baum aufgefaltet, um von der Präsenz sehr vieler Prozessoren Gebrauch machen zu können. Nachdem Teilpfade auf unterschiedlichen Prozessoren analysiert wurden, werden die Ergebnisse kombiniert, um die ursprüngliche Topologie des Graphen wiederherzustellen.

Die durchgehende Verwendung von chartbasierten Verfahren im Rahmen von MILC führt dazu, daß diese Art der parallelen Verarbeitung nicht auf die syntaktische Strukturanalyse beschränkt ist. Vielmehr können auch andere Module von der Existenz hochgradig paralleler Rechnersysteme (etwa auf der Basis von lose gekoppelten Transputern) profitieren.

4.11.1 Erweiterung der Architektur

Die Erweiterung des Systems MILC, die wir in diesem Abschnitt vorstellen, betrifft zwei Aspekte der Verarbeitung: Zum einen werden zusätzliche Eigenschaften gesprochener Sprache in die Analyse integriert. Dies betrifft vor allem prosodische Phänomene, zu einem Teil jedoch auch die Einführung von Rückkopplungen, die bisher nicht implementiert wurden. Zum anderen wird der Gegenstandsbereich der Betrachtung auf vollständige Dialoge ausgeweitet, um Effekte modellieren zu können, die über den engen Kontext einer isolierten Äußerung hinausgehen.

Beide Richtungen der Erweiterung sind in der Vergangenheit zumindest zu einem gewissen Teil in anderem Zusammenhang untersucht worden, sie stellen folglich nichts genuin Neues dar, das durch MILC möglich würde. Trotzdem erscheint die Exploration des Verhaltens in einem integrierten, uniformen System extrem wünschenswert, u.a. um die gegenseitigen Einflüsse von Architekturentscheidungen verfolgen und bewerten zu können. MILC bietet dafür einen idealen Rahmen.

Der Einfluß der Prosodie auf die Analyse gesprochener Sprache ist erstmals ausführlich im Rahmen von Verbmobil untersucht worden (Noeth et al., 1997). Jeder Worthypothese werden hier Wahrscheinlichkeiten zugeordnet, die Angaben über das Tragen eines Phrasenakzentes und das Vorkommen von Phrasen- bzw. Satzgrenzen sowie die Art des Satzmodus machen. Diese Information wird in den nachfolgenden Modulen zur linguistischen Analyse zur Modifikation von Bewertungen von Regeln benutzt, um die Akkuratheit der Analyse zu steigern (vgl. Amtrup und Jekat, 1995). Mit Hilfe prosodischer Daten konnte z.B. die syntaktische Analyse so weitgehend verbessert werden, daß die Anzahl der Lesarten um über 90% reduziert wurde. Ein anderer Aspekt von Prosodie wurde zusätzlich in dem architektonischen Prototypen INTARC genutzt, um den Fokus einer Äußerung zu berechnen und eine Art von flacher, dialogaktbasierter Übersetzung (vgl. Jekat, 1997) durchzuführen. Eine Analyse zeigt, daß fokussierte Wörter häufig als Grundlage für die Detektion des

Dialogaktes benutzt werden können, daß folglich eine prosodisch motivierte Schlüsselworterkennung bereits den Dialogakt prädiziert (Elsner und Klein, 1996).

Die Integration solcher Daten in eine Mehr-Ebenen-Chart kann grundsätzlich auf zwei Weisen erreicht werden. Die in Verbmobil akzeptierte Methode beruht auf der Annotation von Wörtern mit prosodischer Information nach der eigentlichen Worterkennung, da u.a. die Silbenstruktur der Wörter für die Berechnung der prosodischen Merkmalvektoren bekannt sein muß. Eine solche zusätzliche Annotation an Wortkanten paßt sich nahtlos in die Architektur von MILC ein. Das einzig denkbare Hindernis sind sich gegenseitig widersprechende Informationen über Worthypothesen, die in einer einzigen Hyperkante integriert werden sollen, in welchem Fall eine zusätzliche Hyperkante erzeugt wird. Allerdings existieren Ansätze, die ausschließlich am Signalinhalt orientiert sind und nicht auf die vorhergehende Wortsegmentierung durch die Erzeugung eines Wortgraphen angewiesen sind (Strom und Widera, 1996). Die Ergebnisse liegen auf einem hohen Niveau (die Erkennungsrate für Phrasengrenzen liegt bei 86%) und lassen sich als zusätzliche Kanten in die Mehr-Ebenen-Chart integrieren, um für Ableitungen zur Verfügung zu stehen.

Die zweite Maßnahme zur Verbesserung, die äußerungsintern durchgeführt werden kann, ist die Einführung von Rückkopplungen zwischen Modulen zur linguistischen Verarbeitung und der akustischen Spracherkennung. Ein derartiges Verfahren ist, wie bereits in der Einführung angemerkt, innerhalb von INTARC zur Effizienzsteigerung in der Zusammenarbeit von Worterkennung und syntaktischer Analyse angewendet worden (Hauenstein und Weber, 1994). MILC verwendet derzeit keine Rückkopplungen, das Design und die Implementation sind jedoch speziell auf diese Ziele ausgerichtet. Neben der engen Verzahnung der Spracherkennung mit der Syntax sind jedoch auch Einflüsse höherer Bearbeitungsebenen denkbar und sinnvoll (Menzel, 1994).

Diese höheren Verarbeitungsebenen betreffen den zweiten Aspekt der Modifikationen, nämlich die Verarbeitung vollständiger Dialoge statt einzelner Äußerungen. MILC operiert bisher stets auf einzelnen Redebeiträgen von Dialogpartnern, ohne jedoch das akkumulierte Wissen zu speichern. Diese Erweiterung ist allerdings aus mehreren Gründen wünschenswert. Zunächst ist offensichtlich, daß Textkohärenz sich auch in der gesprochenen Sprache über mehrere Äußerungen erstreckt. Die aus vorangegangenen Redebeiträgen extrahierte Information kann zur Disambiguierung der kommunikativen Ziele des aktuellen Beitrags dienen, manche Äußerungen sind erst aus dem Dialogkontext verständlich und können nur unter Benutzung dieses Kontextes adäquat übersetzt werden (vgl. Alexandersson, Reithinger, und Maier, 1997). Daher ist der Ausbau von MILC um eine Dialogverarbeitung ein wünschenswertes Ziel. Dabei kann offen gelassen werden, ob dieses Modul die Daten über Vorangegangenes intern speichert, oder ob die Mehr-Ebenen-Chart zeitlich expandiert wird, um ganze Dialoge zu speichern. Zur Disambiguierung ist eine explizite Speicherung in der Chart nicht notwendig, diese Funktion kann auf einer Frage-Antwort Basis erfüllt werden, wie dies in Verbmobil geschieht (Alexandersson, Maier, und Reithinger, 1995). Falls die Dialogverarbeitung jedoch in einer Rückkopplungsschleife Einfluß auf andere Komponenten nehmen soll, so ist eine Erweiterung der Chart aus Gründen der Uniformität der Informationsspeicherung sinnvoll. Dazu sind allerdings geringfügige Modifikationen an der Chartstruktur notwendig, um lediglich die relevanten dialogischen Strukturen zu erhalten und alle ausschließlich äußerungsintern verwendeten Daten vor der Analyse des nächsten Redebeitrages zu entfernen.

Der Ausbau der Architektur von MILC in Richtung einer dialogverarbeitenden Maschine ist schließlich auch aus der Perspektive der Generierung nützlich. Erst wenn das System Zugang zu bereits getätigten und realisierten Übersetzungen hat, kann eine Textplanung strategischer oder taktischer Art stattfinden, etwa um definite Deskriptionen, die bereits ausgegeben wurden, durch Pronominalisierung stilistisch zu verbessern.

4.11.2 Any-Time Übersetzung

Any-Time Algorithmen sind bisher hauptsächlich für Planungsprobleme angewendet worden (Russel und Zilberstein, 1991; Boddy und Dean, 1994). Ihre Anwendung auch in der Sprachverarbeitung erscheint hochgradig wünschenswert, wie bereits in der Einführung angemerkt. Die theoretischen und praktischen Probleme beim Einsatz in hochgradig modularen, mit komplexen Formalismen arbeitenden Sprachverarbeitungssystemen sind jedoch mannigfaltig. Zunächst gilt es genau zu definieren, wie sich eine Any-Time Fähigkeit auf die Funktionsweise eines Systems auswirkt, und was Nutzer realistischerweise für eine Leistung erwarten dürfen. Danach erst sind die Auswirkungen auf die Interdependenzen zwischen einzelnen Algorithmen und Modulen mit einiger Sicherheit vorhersagbar.

Das Zeitverhalten eines Systems zur Verarbeitung gesprochener Sprache teilt sich in zwei Bereiche auf, wenn wir weiterhin von der Anwendung des Dolmetschens innerhalb eines Dialogs ausgehen:

- Die erste Phase ist durch die Dauer bestimmt, die der Sprecher benötigt, seinen Redebeitrag zu verbalisieren. Abhängig von der Situation und den Sprechern handelt es sich hierbei um wenige Sekunden bis zu etwa einer Minute.[31]
- Die zweite Phase besteht in dem Warten des Sprechers auf die Übersetzung durch die Maschine. Hat der Benutzer einmal seine Äußerung beendet, so ist seine Aufmerksamkeitsspanne recht kurz, die Relevanz, welche die Übersetzung für ihn besitzt, nimmt rasch ab.[32]

Russel und Zilberstein (1991) unterscheiden zwei Arten von Any-Time Algorithmen: *Unterbrechbare Algorithmen* können zu jedem Zeitpunkt während ihres Laufes unterbrochen werden und sind stets dazu in der Lage, Resultate zu liefern. *Kontraktalgorithmen* werden im Vorwege über die nutzbare Zeitspanne informiert; werden sie vor Ablauf dieser Spanne unterbrochen, so findet keine Ausgabe statt, nach der Laufzeit hingegen sind Resultate vorhanden. Die gemeinsame Eigenschaft beider ist, daß die Qualität der Resultate mit der zur Verfügung stehenden Zeit monoton ansteigt, was durch ein probabilistisches Performanzprofil modelliert ist.

[31] Äußerungen von einer Minute Länge sind selten und im Bereich der Terminabsprache meist experimentell elizitiert. Allerdings spielen hier kulturelle Einflüsse eine gewisse Rolle.

[32] Eine interessante Methode, die Zeit, die einem System für die Bearbeitung zur Verfügung steht, zu verlängern, ist, während der Analyse “gemurmelte” Informationen über deren augenblicklichen Stand zu produzieren (Karlgren, 1994).

Aus dieser Definition ist ersichtlich, daß im hier angenommenen Rahmen der Sprachverarbeitung Kontraktalgorithmen primär nicht in Frage kommen können. Die Länge der ersten oben genannten Phase ist unbekannt, während die zweite Phase sehr kurz ist. Der erste Zeitpunkt, an dem eine Schätzung über die noch zur Verfügung stehende Zeit möglich ist, liegt an der Grenze der beiden Phasen. Prinzipiell scheinen also unterbrechbare Algorithmen das Mittel der Wahl zu sein.[33] Die Frage ist nun, welche Granularität ein Algorithmus besitzt, d.h., in welchen Abständen es sinnvoll sein kann, ein Ergebnis abzufragen, in welchen diskreten Einheiten folglich die Zeit gemessen wird. Görz und Kesseler (1994) untersuchen dies für die Strukturanalyse von INTARC und nennen typische Werte von 10 bis 100 ms. Einzelne Unifikationen sollten nach ihrer Meinung nicht unterbrechbar gestaltet werden, da sie Koreferenzen enthalten können, die andernfalls ungelöst bleiben.[34] Innerhalb eines Chart-basierten Systems bieten sich entweder die Durchführung eines einzelnen Auftrages oder die Abarbeitung einer Agenda zu einem bestimmten Zeitpunkt als atomare Einheiten an.

Weitere Überlegungen zeigen, daß es nicht sinnvoll ist, ein System zur Verarbeitung gesprochener Sprache zu unterbrechen, bevor die Äußerung komplett gesprochen ist. Dies ist zwar trivial, hat jedoch Auswirkungen auf die Messung der Zeit im System. Bisherige Arbeiten zur Planung mit Anytime Algorithmen gehen von einer initial zur Verfügung stehenden Eingabe aus, auf Grund derer die Lösung erzeugt wird. Die Eingabe findet atomar statt und kostet, konzeptionell gesehen, keine Zeit. Unter den hier relevanten Gegebenheiten ist jedoch die Herstellung der Anfangsbedingungen keineswegs instantan. Im Gegenteil, die Zeit, die zur Eingabe notwendig ist, macht den Großteil der insgesamt zur Verfügung stehenden Verarbeitungszeit aus. Insofern muß die Gesamtdauer des Prozesses als Summe der beiden o.a. Phasen angesehen werden; eine Unterbrechung während der ersten Phase ist jedoch ausgeschlossen. Nach Ablauf der ersten Phase kann das System in gewissen Abständen (z.B. 100ms) angehalten werden und liefert ein mit der Zeit monoton besser werdendes Ergebnis.

Die Frage nach der Entscheidung über die Qualität eines Resultats ist weithin unentschieden. Im Rahmen der maschinellen Übersetzung kann die strukturelle Komplexität einer Übersetzung (oder, besser, des sie repräsentierenden Strukturbaumes) als erster Anhaltspunkt dienen, ohne jedoch den Status der Allgemeingültigkeit zu besitzen. Deswegen lassen sich maschinelle Verfahren zur Verarbeitung natürlicher Sprache nur als *schwache* Any-Time Algorithmen (Menzel, 1994) modellieren, für die es kein generelles Performanzprofil gibt, sondern lediglich ein auf der Instanz der Verarbeitung einer gegebenen Eingabe beruhendes. Im Rahmen des Systems MILC wird für die ständige Lieferung von besseren Resultaten durch die Auswahlfunktion innerhalb der Generierung gesorgt. Insofern kann die inkrementelle Ausgabe von Äußerungsfragmenten als erster Schritt angesehen werden, eine mit der Zeit monoton wachsende Qualität der Ausgabe bereitzustellen.

MILC implementiert natürlich kein Any-Time Verhalten. Es ist zwar durchgehend inkrementell

[33] Russel und Zilberstein (1991) zeigen, daß sich jeder Kontraktalgorithmus in einen unterbrechbaren Algorithmus umwandeln läßt. Der so gewonnene Algorithmus benötigt die vierfache Zeit für die Produktion von Resultaten äquivalenter Güte.

[34] Sie erwähnen allerdings die Möglichkeit, Kernmerkmale und weniger relevante Merkmale zu definieren, um so eine geteilte Unifikation zu erlauben.

implementiert, eine Unterbrechung, bevor Generierungshypothesen für einen großen Teil der Eingabe vorliegen, ist jedoch wenig erfolgversprechend. Insofern kann MILC nicht auf zeitliche Einschränkungen reagieren, die unterhalb einer gewissen Schwelle liegen.

Die inkrementelle Arbeitsweise aller Module eines Systems hingegen ist eine notwendige Voraussetzung für die Möglichkeit zur Einführung eines Any-Time Verhaltens. Wenn nämlich die Implementation eines Algorithmus nicht monolithisch ist, wird es bei nichtinkrementeller Arbeitsweise zu einer Situation kommen, in der zum Zeitpunkt der Unterbrechung erst eine Teilmenge der für die Produktion eines Ergebnisses notwendigen Komponenten Ausgaben geliefert haben. Wenn die Unterbrechung eintritt, kann folglich kein Resultat geliefert werden. Selbst wenn Kontraktalgorithmen angenommen werden, so ist die Schätzung über die einer Komponente zur Verfügung stehenden Ressourcen abhängig von anderen Modulen und kann nicht ohne weiteres ermittelt werden.

Inkrementelle, modulare Systeme können prinzipiell mit einer Any-Time Anforderung umgehen. Vorausgesetzt, die Inkremente sind klein genug, ist eine repetitive Bearbeitung und Qualitätsverbesserung denkbar. Die Strategie besteht dann darin, zunächst eine sehr schwache Ausgabe zu produzieren (etwa eine Wort-für-Wort Übersetzung) und in weiteren Wellen erneut die Äußerung zu betrachten und zu versuchen, die Ergebnisse qualitativ zu steigern. Eine Auswahlfunktion bewertet die Ergebnisse des Gesamtsystems und ist dazu in der Lage, zu jeder Zeit die bis dahin beste Analyse auszugeben. Ein solches Verhalten, wie es von MILC gezeigt wird, vermeidet die Festlegung auf Zwischenresultate, die endgültige Auswahl wird solange wie möglich offen gelassen; dies geschieht um den Preis eines erhöhten Aufwandes für die Bearbeitung einer hohen Anzahl von Alternativen. Neben der Frage, was denn qualitativ besser sei, die bereits diskutiert wurde, spielt jedoch auch die Frage eine Rolle, wieviel Aufwand in die Verbesserung eines unbefriedigenden Resultats investiert werden muß, also eine dynamische Kosten-Nutzen-Analyse. Auch hier kann es lediglich Heuristiken geben, deren Wirkung bisher noch nicht weiter exploriert wurde; mögliche erste Ansätze bieten sich hier im Bereich der Beschränkungserfüllung an (Menzel, 1998).

Angewendet auf die oben erwähnte Zweiteilung der Bearbeitungszeit läßt sich der Ablauf des Übersetzungsprozesses im Licht einer Any-Time Analyse noch präzisieren:

- Während der Benutzer spricht, d.h. solange die Eingabe noch nicht vollständig vorliegt, wird jede Komponente versuchen, Aufträge soweit rechts wie möglich zu bearbeiten. Dies entspricht einer möglichst weitgehenden Zeitsynchronität. Nur wenn aufgrund des Erkennungsprozesses Leerlaufzeiten entstehen, kann eine Komponente zusätzliche Ergebnisse produzieren.

- Nach Ende der Äußerung[35] ändert sich das Verhalten. Für eine kurze Zeit (nämlich bis alle Komponenten bis zum Ende der Eingabe vorgedrungen sind) muß die nach rechts gerichtete Art der Analyse fortschreiten. Danach ist eine Rohübersetzung angefertigt, folglich ist die

[35] Selbst die Erkennung des Endes einer Äußerung stellt ein nichttriviales Problem dar. Es stellt sich die Frage, wie lang die Pause sein darf, bevor angenommen werden kann, der Benutzer habe seinen Redebeitrag beendet (Eine in der ersten Phase von Verbmobil angewendete Regelung war, daß der Benutzer aktiv einen Knopf während seines Sprechens gedrückt halten muß).

Erhöhung der Qualität das primäre Ziel. Eine sinnvolle Strategie ist hier, zunächst Segmente der Eingabe zu identifizieren, deren Übersetzung qualitativ relativ schlecht ist und Prozessorzeit für deren intensivere Bearbeitung zu verwenden. Die Bearbeitung dieser Segmente hat möglicherweise den größten Einfluß auf die Gesamtqualität der Ausgabe, folglich ist es lohnend, diese priorisiert zu verbessern.

4.12 Umfang des Systems

MILC und die unterstützenden Dienstprogramme (Grammatikübersetzer etc.) sind vollständig in C++ implementiert. Für die Visualisierung wurde auf der Skriptsprache Tcl/Tk (Ousterhout, 1994) aufgesetzt, die um Prozeduren in C zur graphischen Darstellung von Merkmalstrukturen erweitert wurde. Tabelle 4.4 gibt eine Übersicht über den Umfang des Systems in Zeilen Quellcode.

4.13 Zusammenfassung

In diesem Kapitel wurde die Implementation von MILC erläutert, deren architektonische Grundlagen diskutiert und das Verhalten jeder einzelnen Komponente sowie das Zusammenwirken in einem verteilten System dargestellt. Die dominante Motivation, ein System zum inkrementellen Übersetzen spontan gesprochener Sprache zu entwerfen und vollständig zu implementieren, dabei aber gleichzeitig einen Architekturrahmen zu konstruieren, der für zukünftige Experimente und Forschungen Raum läßt, führte dabei zunächst zur Entwicklung von Mehr-Ebenen-Charts, einer Datenstruktur, mit deren Hilfe verteilte, hochkomplexe Softwarepakete zur natürlichsprachlichen Verarbeitung entwickelt werden können. Die Integriertheit, welche die Mehr-Ebenen-Chart einführt, wird noch unterstützt durch die Verwendung eines einheitlichen Formalismus. Sie besitzt Eigenschaften, die sie Datenstrukturen mit zentraler Kontrolle (*Blackboards* oder *Whiteboards*) deutlich überlegen sein lassen.

Die Eignung und Validität der so implementierten Datenstruktur wird durch die vollständige Entwicklung des Übersetzungssystems MILC bestätigt. Eine umfassende Implementation stellt gleichzeitig sicher, daß eine theoretische Überlegenheit sich auch in der Praxis zeigt; eine wie auch immer geartete partielle Realisierung hätte dies nicht leisten können. Die Komponenten (Worterkennung bzw. Hypergraphenkonstruktion, Idiomerkennung, Partielle Strukturanalyse, Äußerungsintegration, Transfer, Generierung) sind durchgängig inkrementell, so daß diese Art der Verarbeitung bis in die Generierung von Oberflächenhypothesen beibehalten werden kann. MILC arbeitet dabei auf der Ebene von Einzeläußerungen.

Die hauptsächlich wünschenswerten Erweiterungsgebiete in konventionellem Rahmen sind die Nutzung prosodischer Information zur Disambiguierung und die Erweiterung auf die Verarbeitung von vollständigen Dialogen. Unter architektonischen Gesichtspunkten ist die Erweiterung um Rückkopplungen, insbesondere an der *Speech-Language*-Schnittstelle, gefragt.

Tabelle 4.4: Umfang des Systems (in Lines of Code)

Komponente	Zeilen
Kommunikation und Konfiguration	
Kommunikationssystem ICE	10715
Konfiguration	1158
Zubehör	
Bäume und Listen	1794
Systemstatistik, Objekterzeugung	465
Graphenanalyse	
DAGs	4825
Analysesystem	360
Merkmalstrukturen	
Kern	6178
Funktionen	103
Chart	
Chartverwaltung	1428
Agendenverwaltung	244
Wissensquellen	
Grammatikverwaltung	454
Lexikonverwaltung	317
Compiler für Grammatiken und Lexika	240
Typenverband	464
Grammatiken	2494
Lexika	4322
Komponenten	
Erkenner	362
Idiomprozessor	697
Parser	825
Integrator	765
Transfer	779
Generierung	439
Visualisierung	1118
Gesamt	40546

Kapitel 5

Experimente und Ergebnisse

In diesem Kapitel beschreiben wir die Experimente, die mit dem System MILC durchgeführt wurden. Den Kern bilden Untersuchungen zur Übersetzung einiger Dialoge aus der Verbmobil-Domäne. Begleitende Experimente widmen sich der Eignung von Hypergraphen für die zugrundeliegende Aufgabe sowie der Abschätzung, wie effizient inkrementelle Verfahren im Vergleich zu nicht-inkrementellen sein können.

In den vorangegangenen Kapiteln sind die theoretischen Grundlagen sowie die Architektur und Implementation des Systems MILC dargestellt worden. Wir haben argumentiert, daß die prädominante Motivation für die Realisierung der Anwendung war, eine Grundlage für die experimentelle Untersuchung architektonischer Varianten zur Verfügung zu stellen, die innerhalb eines inkrementellen Paradigmas der Sprachverarbeitung liegen. Die Eignung inkrementeller Verfahren zu zeigen, hat dabei zunächst zur Folge gehabt, das System in Gänze zu implementieren, da eine lediglich teilweise Realisierung eine vollständige Validierung nicht zuläßt. Um weiter zu demonstrieren, daß die Bearbeitung realistischer Eingaben im Bereich des Möglichen liegt, werden in diesem Kapitel Dialoge untersucht, die im Rahmen von Verbmobil aufgenommen wurden.

Wir werden zunächst die Auswirkungen zeigen, die der Einsatz von Hypergraphen auf die Performanz einer strukturellen Analyse gesprochener Sprache hat. Es zeigt sich, daß der Übergang von konventionellen Wortgraphen zu Graphen, deren Kanten Mengen von Start- und Endknoten besitzen, die Analyse großer, inkrementeller Graphen wesentlich erleichtert und in Regionen der Verarbeitungszeit bringt, die als akzeptabel gelten können. Daran schließt sich die Untersuchung ganzer Dialoge zur Terminvereinbarung an, deren Übersetzungen von englischen Muttersprachlern evaluiert wurden. Der letzte experimentelle Abschnitt dieses Kapitels widmet sich der Abschätzung, inwieweit der Einsatz eines Mehrprozessorsystems dazu führt, daß einerseits die Bearbeitung von inkrementellen Äußerungen beschleunigt wird und andererseits, wie das Laufzeitverhalten paralleler inkrementeller Systeme im Vergleich zu nichtinkrementellen Verfahren auf Einprozessormaschinen charakterisiert werden kann.

Alle hier beschriebenen Experimente (mit Ausnahme der zur Hypergraphenkonversion) wurden auf einer Sun UltraSparc 4 mit zwei Prozessoren und 1 GB Hauptspeicher unter Solaris 2.6 bei ansonsten geringer Last (Lastwert 0.03 ohne Verwendung von MILC) durchgeführt.

5.1 Hypergraphen

Die wesentliche Modifikation, die Hypergraphen zur Verarbeitung natürlicher Sprache von konventionellen Wortgraphen unterscheidet, ist deren Fähigkeit, Mengen von Knoten als Start- und Endpunkt einer Kante zuzulassen. Es ergibt sich, daß die Anzahl der Knoten eines Graphen durch die Hypergraphenkonversion nicht verändert wird, die Anzahl der Kanten jedoch drastisch sinken wird. Dies sollte gleichzeitig bedeutende Konsequenzen für die Laufzeit von Modulen zur Verarbeitung von gesprochener Sprache haben.

Zur Untersuchung beider Phänomene wurde ein Dialog in der Domäne der Terminvereinbarung untersucht. Es handelt sich hierbei um den Dialog n002k, dessen 41 Turns im Schnitt jeweils 4,65 Sekunden lang sind. Die vom Hamburger Erkenner für diese Äußerungen produzierten Wortgraphen enthalten im Durchschnitt 1828 Kanten.

Abbildung 5.1 zeigt die Reduktion in der Anzahl der Kanten durch die Transformation in Hypergraphen. Es verblieben lediglich 157 Kanten im Durchschnitt, eine Reduktion um etwa 91%.

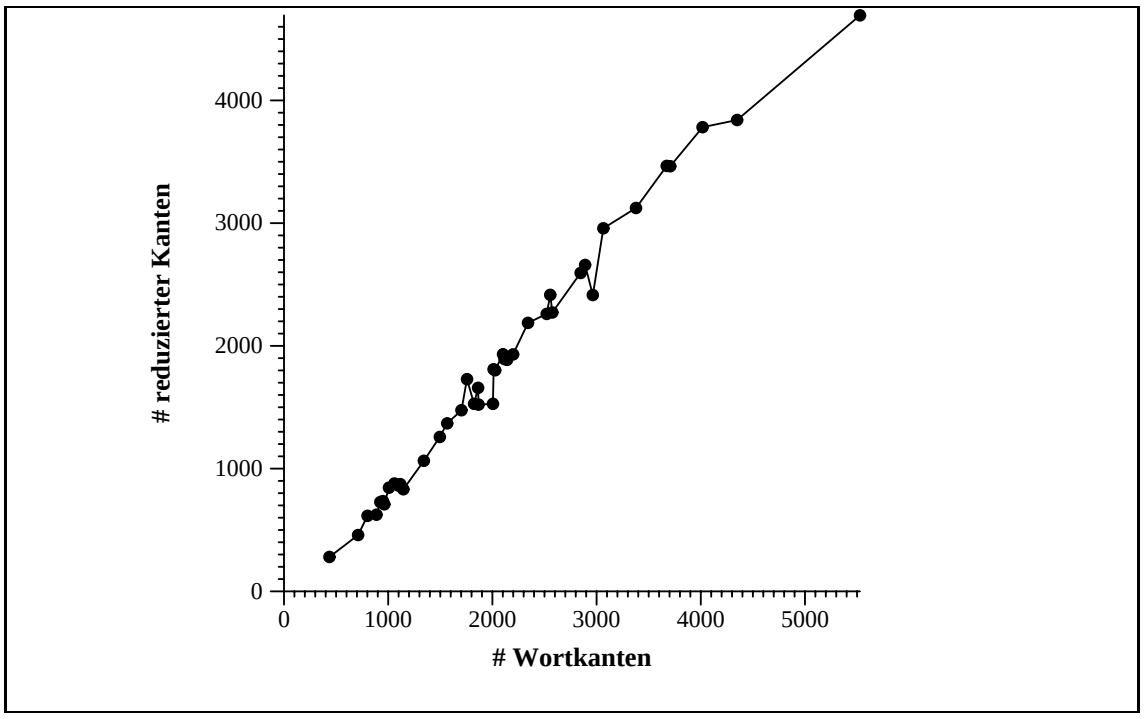


Abbildung 5.1: Reduktion von Kanten durch Hypergraphenkonversion

Um die Konsequenzen zu ermitteln, die eine derartige Behandlung von Graphen auf die Performanz der linguistischen Verarbeitung hat, sind die Wortgraphen und Hypergraphen einer partiellen Strukturanalyse unterworfen worden. Hierzu kam die bereits in Abschnitt 4.6 verwendete Fassung der Wissensquellen zum Einsatz, also eine Typenhierarchie von 345 Typen, ein Lexikon mit 413 Lesarten und eine Grammatik mit 80 Regeln.

Zur Charakterisierung der Performanzsteigerung durch den Einsatz von Hypergraphen verwenden wir zwei Parameter: Die Anzahl von Kanten der Chart, die im Laufe der Analyse durch den Parser erzeugt wurden und die zur Bearbeitung selbst verwendete Zeit. Die Ersparnis, die sich für die Analyse ergibt, liegt in einem ähnlichen Bereich wie die Reduzierung der Kantenanzahl; sie liegt nicht exponentiell höher, wie man vielleicht durch Berücksichtigung aller möglichen Pfade erwartet hätte. Dies liegt im wesentlichen am Redundanztest, der innerhalb des Parsers ausgeführt wird. Die Strukturanalyse der Ursprungsgraphen erzeugte im Schnitt 15547 Kanten in der Chart, während für Hypergraphen 3316 Kanten benötigt wurde, eine Ersparnis von 79%. Die detaillierte Analyse ist in Abb. 5.2 dargestellt. Die zeitliche Reduktion liegt konsequenterweise in gleichen Größenordnungen, wie Abb. 5.3 zeigt. Das Parsing von Wortgraphen benötigte etwa 87,7s, das von Hypergraphen 6,4s, eine Ersparnis von 93%. Die Anzahl der durch die Hypergraphenkonversion neu eingeführten Pfade, die eine Analyse erhalten, ist extrem gering (vgl. Abschnitt 2.6.3). Die Analyse des ursprünglichen Dialogs generiert 5586 syntaktische Konstituenten, die der zugehörigen Hypergraphen lediglich 12 Hypothesen mehr, eine Übergenerierungsrate von 0,2%. Die zusätzlich auftretenden Hypothesen sind darüber hinaus sehr kurz (zwei bzw. drei Wörter).

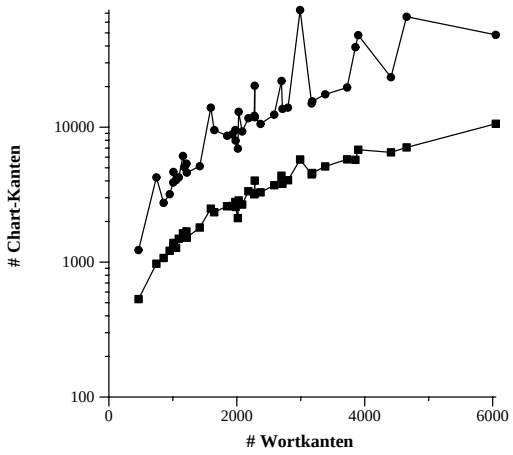


Abbildung 5.2: Reduktion von Chart-Kanten durch Hypergraphenkonversion

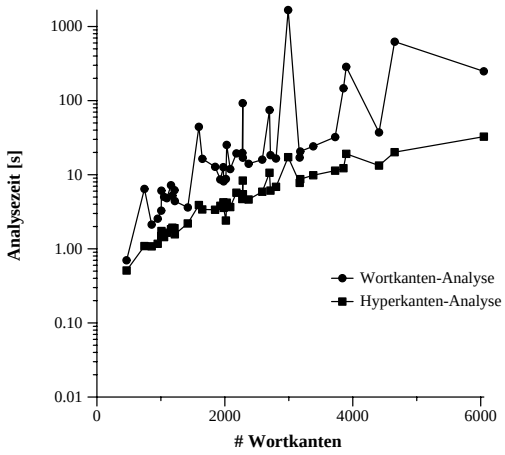


Abbildung 5.3: Reduktion der Analyse-Zeit durch Hypergraphenkonversion

Allerdings gab es extreme Beispiele, für welche die Analyse unter Verwendung von Hypergraphen 94 mal so schnell war die die Originalversion. Ein Turn mußte darüberhinaus aus dem Testset ausgeschlossen werden, da er als Wortgraph wegen Speichermangels nicht erfolgreich bearbeitet werden konnte. Die hier durchgeführten Experimente wurden auf einer Sun Ultra SparcStation 1 mit 128MB Hauptspeicher gerechnet.

5.2 Übersetzung

5.2.1 Datenmaterial

Die Aufnahmen, die der hier angestellten Untersuchung zugrunde liegt, bestehen aus fünf Dialogen der Verbmobil-Domäne. Es handelt sich um Gespräche, deren Ziel die Vereinbarung eines Termins ist. Einige charakteristische Angaben zu den Dialogen sind in Tab. 5.1 dargestellt. Wir verwenden Daten, die zur akustischen Evaluation herangezogen wurden, die mithin nicht zum Training des verwendeten Worterkenners (Huebener, Jost, und Heine, 1996) beigetragen haben. Die Stellung der Aktanten ist vom Geschlecht her gemischt. Die Erkennungsrate liegt im Bereich um 70%. Sie wurde erzielt, indem jeweils maximal drei aktive Marken pro Zustand des Netzes zugelassen wurden, welches das Erkennerlexikon beschreibt. Der Schwellwert für die Bewertung von Zwischenergebnissen innerhalb von Wörtern und am Wortende lag bei 100.

Die hier beschriebene Wortakkuratheit liegt um einige Prozentpunkte unter dem im Moment für die vorliegenden Signaldaten maximal erreichbaren Wert (Lehning, 1996). Allerdings werden die Erkennungsergebnisse für eine Evaluation normalerweise nichtinkrementell erhoben und gravierenden Modifikationen unterworfen, die für eine inkrementelle Arbeitsweise nicht unbedingt geeignet sind (Amtrup, Heine, und Jost, 1997; Jost, 1997). Außerdem liegt die Untersuchung von Graphen in diesem Bereich von Erkennungsraten prinzipiell in unserem Interesse, entsprechen sie doch den augenblicklich besten Ergebnissen für die Erkennung wesentlich unrestringierterer Sprache als sie für Verbmobil benutzt wird, nämlich ungefähr zwischen denen für den *Switchboard*-Korpus und den *CallHome*-Korpus (Finke et al., 1997).

Tabelle 5.1: Eigenschaften der für die Experimente genutzten Dialoge

Eigenschaft	Dialog j512	Dialog j533	Dialog j534	Dialog m111	Dialog m123
Anz. Äußer. [#]	17	6	24	24	11
Länge [s] (min)	0.76	0.65	0.83	0.67	0.96
(max)	36.09	18.89	17.29	10.19	10.7
(∅)	8.50	7.84	7.44	3.16	5.25
Anz. [Wörter] (∅)	23.8	24.3	22.2	10.5	14.9
Aktantenstellung	m/m	m/f	m/m	f/m	f/m
Erkennungsrate (%)	64.8	68.1	66.2	70.3	76.8
Erkennungsrate bester Pfad (%)	43.6	37.0	42.6	40.2	58.3
Anzahl Knoten (∅)	784	699	687	302	470
Anzahl Kanten (∅)	7340	7904	6869	3828	4342
Anzahl Pfade (∅)	$7.5 \cdot 10^{213}$	$7.4 \cdot 10^{150}$	$8.8 \cdot 10^{106}$	$8.3 \cdot 10^{81}$	$3.7 \cdot 10^{63}$

Tab. 5.2 gibt einen Eindruck von der Art der verwendeten Dialoge. Sie zeigt die Transliterationen

von Äußerungen, die im Dialog m123 enthalten sind. Die verwendeten außersprachlichen Hypothesen sind ⟨NIB⟩ (nichtinterpretierbarer Bereich) und ⟨SPELL⟩ (Buchstabiereinheiten, hier die Filiale O.B.K.I.).

Tabelle 5.2: Die Äußerungen des Dialogs m123

Turn	Äußerung
m123n000	guten Tag Herr ⟨NIB⟩ Klitscher hier ist wieder Fringes ich möchte gerne diesmal einen Termin ⟨NIB⟩ für das Arbeitstreffen in der Filiale ⟨SPELL⟩ in Potsdam mit Ihnen vereinbaren ⟨NIB⟩
m123n001	⟨NIB⟩ guten Tag Frau Fringes ja wie sieht es denn bei Ihnen aus in der Woche vom sechsten bis zehnten Mai
m123n002	⟨NIB⟩ sechster bis zehnter Mai ach da muß ich unbedingt zu einem Seminar nach Rothenburg ⟨NIB⟩
m123n003	⟨NIB⟩ dann schlagen Sie doch einen Termin vor der Ihnen passen würde
m123n004	ja also wie wäre es denn im April vom zweiundzwanzigsten bis zum sechsundzwanzigsten
m123n005	oh das geht ⟨NIB⟩ leider nicht da habe ich ein Seminar in Koblenz ⟨NIB⟩ haben Sie noch einen anderen Termin
m123n006	ja lassen Sie mich mal sehen ⟨NIB⟩ ja da hätte ich noch im Juni vom zehnten bis zum vierzehnten einen Termin frei ⟨NIB⟩
m123n007	das würde mir auch passen ja sollen wir uns dann in Potsdam treffen in der ⟨NIB⟩ Filiale ⟨SPELL⟩
m123n008	ja machen wir das
m123n009	okay auf Wiedersehen Frau Fringes ⟨NIB⟩
m123n010	auf Wiederhören

Die fünf Dialoge wurden unterteilt, um eine qualitative Abschätzung über die Abdeckung der im System integrierten Wissensquellen zu erlauben. Die Gespräche m111 und m123 wurden zur Entwicklung der Grammatik und des Lexikons herangezogen, während die Dialoge j512, j533 und j534 ohne Modifikation der Grammatik analysiert wurden. Die verschiedenen Lexika des Systems wurden jedoch um neu hinzugekommene Wörter erweitert.

5.2.2 Linguistische Wissensquellen

Die linguistischen Module verwenden einen einheitlichen Typenverband, um Merkmalstrukturen innerhalb des Systems vergleichbar zu machen. Diese Hierarchie besteht aus 633 Typen. Ihre Struktur wurde bereits in Abschnitt 3.3.1 erläutert. Unter Verwendung dieses Verbandes wurden die relevanten Wissensquellen erzeugt. Darunter fallen vier Lexika (für Idiome, Analyse, Transfer und Generierung) sowie vier Grammatiken (für partielle Strukturanalyse, Äußerungsintegration, Transfer und

Generierung):

- Das Lexikon für die Idiomerkennung besteht aus 21 Einträgen für kompositionell nicht zu behandelnde Floskeln wie “guten Tag” oder Interjektionen wie “einen Moment”. Zusätzlich wurde die Idiomverarbeitung dazu verwendet, mehrteilige Präpositionen (z.B. “bis auf”) zu modellieren. Auf diese Weise kann die Beschreibung von Präpositionalphrasen in der Analysegrammatik von einzelnen Wörtern als Präposition ausgehen, die Konstruktion aus mehreren Bestandteilen bleibt der Idiomverarbeitung vorbehalten. Naturgemäß verwendet die Idiomerkennung keine Grammatik.

- Die partielle Strukturanalyse verwendet zwei Quellen linguistischen Wissens: Eine Phrasenstrukturgrammatik und ein Lexikon. Die Grammatik besteht aus 35 Regeln zur Konstruktion “kleiner” Phrasen. Die Anbindung von Komplementen und Adjunkten wird folglich nicht modelliert. Die Menge von Regeln teilt sich auf in die Beschreibung von Datumsausdrücken, Präpositionalphrasen, Nominalphrasen sowie die Konstruktion von modifizierenden Adverbialphrasen. Adjektivphrasen werden bereits hier in die dominierenden Nominalphrasen integriert.

  Das zugehörige Lexikon besteht aus 772 Einträgen. Wir implementieren ein Vollformenlexikon, dessen Mehrdeutigkeit auf lexikalischer Ebene 1,15 ist, d.h. für eine Oberflächenrepräsentation eines Wortes existieren im Durchschnitt 1,15 Merkmalstrukturen, die unterschiedliche Facetten eines Wortes beschreiben. Allein auf Verben beschränkt, beträgt die lexikalische Ambiguität 1,38. Das Lexikon enthält Einträge für Verben, Nomen, Adjektive, Adverbien, Artikel, Datumsbestandteile, Konjunktionen und Interjektionen.

- Die Äußerungsintegration benötigt kein Lexikon. Alle wesentlichen Strukturen werden bereits durch die partielle Strukturanalyse erzeugt, insbesondere wird auch für Verben ein lexikalischer Zugriff durchgeführt und die Ergebnisse als präterminale Kanten an die Integration gesandt. Die Grammatik der Integration modelliert im wesentlichen die Anbindung von Komplementen innerhalb von Verbalphrasen, sowie die Ermittlung der Zugehörigkeit von Präpositional- und Adverbialphrasen. Sie besteht aus 18 allgemeinen Regeln.

- Für den Transfer sind sowohl eine Grammatik wie auch ein Lexikon notwendig. Die Grammatik beschreibt die Umsetzung deutscher semantischer Beschreibungen in deren englische Äquivalente. Sie besteht aus lediglich 10 Regeln, deren wesentlicher Teil sich mit der Transformation von verbalen und nominalen Ausdrücken beschäftigt. Ein Teil der Regeln beschreibt den Transfer von Modifikatoren. Datumsausdrücke hingegen, die während der Analyse relativ viele Regeln beanspruchen, werden ohne Modifikation in die Zielsprache übernommen. Das in diesem Teil des Systems verwendete Lexikon behandelt die Umsetzung semantischer Konzepte und ist ebenfalls hauptsächlich nach Wortarten unterteilt. Es enthält in der augenblicklichen Fassung 207 Beschreibungen.

- Die Generierung produziert englische Oberflächenrepräsentationen aufgrund englischer semantischer Beschreibungen. Das hierfür verwendete Lexikon enthält 311 Einträge, wesentlich weniger als das analog in der Analyse verwendete. Dies liegt neben der einfacheren

englischen Morphologie daran, daß in der Generierung weniger Gewicht auf eine stilistisch anspruchsvolle Wortwahl gelegt wurde als vielmehr auf die prinzipiell adäquate Realisierung des Inhalts von Äußerungen. Die Grammatik mit 40 Regeln beschreibt die wesentlichen notwendigen Konzepte wie Verbalphrasen, Nominalphrasen sowie präpositionale, adjektivische und adverbiale Modifikatoren.

5.2.3 Durchführung und Systemparameter

Die vorher beschriebenen Dialoge wurden mit Hilfe des Systems MILC übersetzt. Dabei wurden Laufzeiten und die Ausgaben der Generierung festgehalten, sowie einige potentiell interessante Systemparameter. Tab. 5.3 zeigt die wesentlichen Ergebnisse. Alle Berechnungen wurden mit einem Sprachmodellgewicht von 5,0 und einer Strahlensuche durchgeführt, die 200 Aufträge pro Knoten zuließ und alle anderen abschneidet. Alle Angaben stellen Durchschnittswerte für die Verarbeitung einer Äußerung eines Dialogs dar. Als Laufzeit ist die verstrichene Uhrzeit eingetragen, gemessen vom Laden der ersten Komponente bis zur Terminierung des Systems.

Tabelle 5.3: Ergebnisse der Analyse von fünf Dialogen

Parameter	Dialog j512	Dialog j533	Dialog j534	Dialog m111	Dialog m123
Laufzeit [ms]	63977	48974	54725	17361	19352
Anzahl Übersetzungen [#]	51	45	44	24	28
Anzahl Kanten [#]	9876	11065	8849	4952	5740
Anzahl Aufträge [#]	92579	190212	93961	44013	37793
davon abgeschnitten [#]	48137	137019	47121	21764	13600
Hauptspeicher [MB]	81	102	92	48	55
Unifikationen [#/s]	2068	3048	4520	3018	3622

Um einen Eindruck von der Art der Ausgaben zu geben, sind die Sequenzen von Generatorausgaben für die Äußerung j534a005 (“⟨NIB⟩ ⟨NIB⟩ oh das ist schade da bin ich zu einem geschäftlichen ⟨NIB⟩ Termin in Freiburg aber wie ist es denn vom vierzehnten bis zum neunzehnten”) in Tab. 5.4 dargestellt. Man sieht deutlich, wie etwa Erkennungsfehler zu Fehlübersetzungen führen (“Termin in Freiburg” wird zu “appointment on friday”); desgleichen wird die inkrementelle Natur der Ausgaben von MILC deutlich, indem sukzessive längere Ketten von englischen Äußerungsfragmenten produziert werden. Der Generator sucht jeweils den besten Pfad durch den bisher konstruierten Graphen und gibt diesen aus. Für den hier gezeigten Redebeitrag wurden in der Generierung 130 englische Teiläußerungen erzeugt, die den Teilgraphen ausmachen, der für die Oberflächenrepräsentationen zur Verfügung steht.

Tabelle 5.4: Die Generatorausgaben für die Äußerung j534a005

Yes
Yes |let's say
my |let's say
me |Yes |let's say
me |Yes |it |let's say
me |Yes |it |for me |let's say
me |Yes |it |for me |let's say |my
me |Yes |it |I would suggest
me |Yes |it |I would suggest |you
me |Yes |it |I would suggest
me |Yes |it |I would suggest |an appointment
me |Yes |it |I would suggest |an appointment |us
me |Yes |it |I would suggest |an appointment |we
me |Yes |it |I would suggest |an appointment |friday
me |Yes |it |I would suggest |an appointment |friday |we
me |Yes |it |I would suggest |an appointment |friday |we |us
me |Yes |it |I would suggest |an appointment |on friday |we
me |Yes |it |I would suggest |an appointment |on friday |we |us
me |Yes |it |I would suggest |an appointment |on friday |we
me |Yes |it |I would suggest |an appointment |we
me |Yes |it |I would suggest |an appointment |we |us
me |Yes |it |I would suggest |an appointment |we
me |Yes |it |I would suggest |an appointment |we |you
me |Yes |it |I would suggest |an appointment |we |something
me |Yes |it |I would suggest |an appointment |we |the fifth
me |Yes |it |I would suggest |an appointment on friday |we |the fifth
me |Yes |it |I would suggest |an appointment |we |the fifth
me |Yes |it |I would suggest |an appointment on friday |we |the fifth
me |Yes |it |I would suggest |an appointment on friday |we |something |the fourteenth
me |Yes |it |I would suggest |an appointment on friday |we |from the fourteenth
me |Yes |it |I would suggest |an appointment on friday |we |suggest maybe |you |
me |Yes |it |I would suggest |an appointment on friday |we |suggest maybe |you |the fourteenth
me |Yes |it |I would suggest |an appointment on friday |we |suggest maybe |you |the fourteenth |up to the nineteenth

5.2.4 Evaluation

Zur Evaluation der angefertigten Übersetzungen wurden die Ergebnisse der Systemläufe englischen Muttersprachlern vorgelegt. Nach Einführung in die Thematik und in die spezielle Art der Ergebnispräsentation durch MILC (nämlich durch eine Reihe von englischen Fragmenten für jede Äußerung) haben die Probanden jeweils drei Angaben zur jeder übersetzten Äußerung gemacht. Sie sollten den wesentlichen pragmatischen Gehalt der Übersetzung angeben (aus einer Auswahl dialogaktähnlicher Klassen, nämlich Begrüßung; Verabschiedung; Aufforderung, einen Vorschlag zu machen; Vorschlag; Ablehnung; Begründung; Akzeptanz).[1] Mehrfachnennungen waren möglich. Außerdem sollte, falls möglich, ein Datum angegeben werden, das nach Ansicht der Probanden zentral für die jeweilige Äußerung ist. Schließlich wurde nach der subjektiven Qualität der Übersetzungen gefragt, die auf einer Skala mit vier Werten (Gut, Rauh, Gerade noch verständlich, Nicht verständlich) gewertet werden konnte.

Tab. 5.5 zeigt die Ergebnisse der Evaluation. Die Ergebnisse der Probanden wurden mit Angaben verglichen, die aufgrund der deutschen Eingaben erstellt wurden. Es sind jeweils die Anteile der korrekten Antworten für den pragmatischen Gehalt und ein korrektes Datum angegeben. Außerdem wurden die Bewertungen der Übersetzungsgüte in numerische Werte umgesetzt, wobei 1 Gut meint und 4 Nicht verständlich.

Tabelle 5.5: Evaluation der Übersetzungen

Parameter	Dialog j512	Dialog j533	Dialog j534	Dialog m111	Dialog m123
Pragmatik korrekt [%]	74	65	58	63	81
Datum korrekt [%]	61	45	46	66	45
Qualität [1–4]	2.3	2.1	2.4	2.3	1,9

Diese Art der Evaluation ist recht grob. Sie kann weder den Beitrag einer einzelnen Komponente quantifizieren, noch ist die Datenbasis groß genug, um sichere Trends festzustellen. Für eine derartige Evaluation müßten schärfere Kriterien an die Methodik angelegt werden, größere Testkorpora bearbeitet werden und schließlich auch eine größere Zahl an Evaluatoren gewonnen werden. Ein solchermaßen umfangreicher Auswertungsapparat kann jedoch im vorliegenden Rahmen nicht gehandhabt werden, die ist größeren Projekten vorbehalten (Jost, 1997; Carter et al., 1997). Die Art der hier verwendeten Evaluation erlaubt jedoch eine qualitative Einordnung der Leistung des Systems MILC.

[1] Diese Liste stellt eine Auswahl von in Verbmobil gebräuchlichen Dialogakten dar, vgl. Jekat et al. (1995).

5.2.5 Erweiterungen

Die Ergebnisse der Evaluation zeigen, daß ein großer Teil der Eingaben für MILC in befriedigender Weise verarbeitet werden kann. Die Erfolgsquote kann mit 60,4% approximativ korrekten Übersetzungen angegeben werden (Wahlster, 1997). Die Systemlaufzeiten liegen im Bereich sechsfacher Echtzeit. Nimmt man an, daß die Bearbeitung im selben Moment anfängt, in dem der Sprecher seine Äußerung beginnt[2], so muß durchschnittlich noch 34 Sekunden nach Beendigung des Redebeitrages auf die komplette Übersetzung gewartet werden. Liegt dieser Wert auch recht hoch, mit Sicherheit jedoch jenseits der Toleranzschwelle naiver Benutzer, so ist die Performanz des Systems doch gut genug, um umfangreiche weitere Experimente zu erlauben.

Die Dialoge j512, j533 und j534 waren für die Entwicklung der Grammatiken von MILC nicht verwendet worden. Eine nachträgliche Analyse zeigt, daß die Analyse von Nebensätzen (z.B. j533a000: "... nachdem der letzte Termin im Juni gewesen ist würde es mir im Juli nicht so ⟨UNK⟩ gut passen ...") und Aussagesätzen mit spontansprachlicher Wortstellung (z.B. j533a001: "bin ich sehr mit einverstanden") noch nicht vollständig inkorporiert wurde. Eine weitergehende Analyse struktureller Varianten und deren Einbeziehung in die Grammatik ist folglich erforderlich. Desweiteren ergibt sich, daß eine elaboriertere Modellierung von Tempus und die konjunktiver Formulierungen, die überwiegend aus Gründen der Höflichkeit benutzt werden, notwendig ist.

Abschließend muß angemerkt werden, daß eine Untersuchung wie die vorliegende aus Aufwandsgründen eine Abdeckung ähnlich der größerer Forschungsprojekte (in diesem Bereich hauptsächlich Verbmobil (Wahlster, 1997), aber auch Janus (Lavie et al., 1997)) nicht erreichen kann. U.E. stellt aber die Modellierung eines kleinen Ausschnitts einer Domäne wie der Terminvereinbarung einen gut geeigneten Testfall für die Evaluierung der prinzipiellen Leistungsfähigkeit neuartiger Architekturschemata dar. Der Aufwand für eine Analyse (und damit die Laufzeit des Systems) steigt zwar stärker als linear mit der Anzahl der zur Verfügung stehenden Lexikoneinträge, insbesondere, wenn durch eine Mehrfachkategorisierung die Ambiguität erhöht wird, und auch die Verfeinerung von Grammatiken wird zur Steigerung der Verarbeitungslast beitragen. Gleichzeitig wird allerdings durch diese Verfeinerung eine selektivere Anwendung von Regeln ermöglicht. Unsere Vermutung geht dahin, daß sich beide Effekte nahezu ausgleichen. Unbedingt in Betracht gezogen werden sollte jedoch die bereits in Abschnitt 4.11 angesprochene Erweiterung von Grammatiken um statistische Anteile, was zu einer Verringerung der Suchzeiten führen sollte.

5.3 Vergleich mit nichtinkrementellen Verfahren

Ein interessanter Aspekt des Umgangs mit inkrementellen Verfahren ist die Frage, inwieweit der unvermeidliche Mehraufwand, der zunächst mit der Einführung von Inkrementalität einhergeht, durch den Einsatz von Parallelität intermodularer Natur wett gemacht werden kann. Um diese Frage ansatzweise zu beantworten, haben wir eine Reihe von Experimenten durchgeführt und Laufzeiten

[2] Wir vernachlässigen hierbei die Zeit, die zur Aufnahme und Worterkennung notwendig ist.

für verschiedene Konfigurationen gemessen. Die Ergebnisse sind in Tab. 5.6 aufgeführt.

Tabelle 5.6: Laufzeiten im Vergleich inkrementeller und nichtinkrementeller Verfahren

Verfahren			Dialog j512	Dialog j533	Dialog j534	Dialog m111	Dialog m123	rel. Leistung
I	I	-	63977	48974	54725	17361	19352	1.17
I	N	-	113245	99752	86179	29541	33457	2.08
N	I	-	22012	26692	40101	13915	14070	0.67
N	N	-	41322	39490	53600	19380	20634	1.00
N	N	B	56017	57486	73310	27377	28489	1.39

Die Dialoge des Testkorpus wurden wiederholt analysiert und die durchschnittliche Gesamtlaufzeit notiert. Die erste Spalte gibt dabei den Versuchsaufbau für das jeweilige Experiment an. Der erste Buchstabe besagt, ob inkrementell erzeugte, linksverbundene Graphen als Eingabe verwendet wurden (I) oder ob diese vorher in konventionelle Wortgraphen umgewandelt wurden (N). Der zweite Buchstabe charakterisiert die Verarbeitungsart von MILC. Ein Eintrag von I kennzeichnet die normale Art der Verarbeitung mit lediglich durch Datenabhängigkeiten auftretenden Wartezeiten. Falls hier ein N notiert ist, so wurde die Verarbeitung künstlich serialisiert. Dazu wurde ein Protokoll eingeführt, das eine Komponente erst dann mit der Berechnung beginnen läßt, wenn die vorher arbeitenden Komponenten alle Aufträge verarbeitet haben. Da innerhalb von MILC momentan kein Gebrauch von Rückkopplung gemacht wird, ist dies problemlos möglich. Allerdings muß angemerkt werden, daß die Serialisierung nicht vollständig ist, da zum einen der Programmstart jeder Komponente auf dem verwendeten Multiprozessor parallel abläuft und zum anderen die unterliegende Kommunikationsschicht ebenfalls nebenläufig arbeitet. Deswegen wurde eine weitere Konfiguration eingeführt, die durch den dritten Buchstaben gekennzeichnet wird. Steht hier ein B, so wurden neben dem System MILC ein weiterer Prozess gestartet, der für eine hohe Auslastung eines Prozessors sorgte, indem in einer Endlosschleife einfache arithmetische Berechnungen angestellt werden. Das führt dazu, daß in diesem Fall MILC mit genügender Genauigkeit als auf einen Prozessor beschränkt angesehen werden kann, während alle übrigen Parameter (Art des Prozessors, Hauptspeicherausbau etc.) konstant gehalten werden können.

Das Ergebnis der Versuche zeigt, daß inkrementelle Ansätze durchaus mit konventionellen Verfahren konkurrieren können. Wie zu erwarten war, ist die Bearbeitung nichtinkrementeller Graphen, also vollständig verbundener Eingaben, die keine “toten Enden” enthalten, durchgängig effizienter als die Analyse ihrer linksverbundenen Entsprechungen. Auch die Tatsache, daß die inkrementelle Übersetzung stets schneller erfolgte als die vergleichbare nichtinkrementelle, überrascht durch die Ausnutzung von Parallelität nicht. Die wesentliche Aussage dieser Experimente ist jedoch, daß im vorliegenden Fall die inkrementelle Bearbeitung inkrementeller Graphen lediglich um einen Faktor 1,17 langsamer verlief als die nichtinkrementelle Bearbeitung konventioneller Graphen. Bisher war trotz der Effekte der Parallelität die Behandlung linksverbundener Graphen immer mit einem erheblichen Mehraufwand verbunden, der einen herkömmlichen Ansatz deutlich favorisierte. Hier allerdings zeigt sich, daß sich durch den Einsatz der in dieser Arbeit beschriebenen Methoden auch

auf inkrementellem Wege eine ähnlich gute Effizienz erzielen läßt, insbesondere, als die für die Experimente verwendete Maschine lediglich zwei Prozessoren hat, innerhalb von MILC aber mindestens vier Hochlastprozesse (Strukturanalyse, Integration, Transfer und Generierung) gleichzeitig ablaufen können.

Allerdings muß angemerkt werden, daß eben getroffene Aussage lediglich für den hier skizzierten Fall gültig ist. Die nichtinkrementelle Variante der verwendeten Bearbeitungsmechanismen entstand durch die künstliche Serialisierung inhärent inkrementell angelegter Prozesse. Es steht außer Frage, daß eine durchgängig konventionelle Analyse in weiten Teilbereichen andersartige Mechanismen verwendet hätte. So sind die Analyseprozesse weiterhin an einem Verarbeitungsschema orientiert, das von links nach rechts in der Zeit operiert. Sämtliche Optimierungsmöglichkeiten, die sich aus der Zugänglichkeit kompletter Eingaben ergeben, sind nicht berücksichtigt. Ein korrekter evaluativer Vergleich beider Paradigmen kann erst dann erfolgen, wenn Optimierungen sowohl für das inkrementelle als auch für das konventionelle System vorgenommen werden. Wir schätzen, daß die nichtinkrementelle Verarbeitung so um einen Faktor zwei bis drei beschleunigt werden kann.

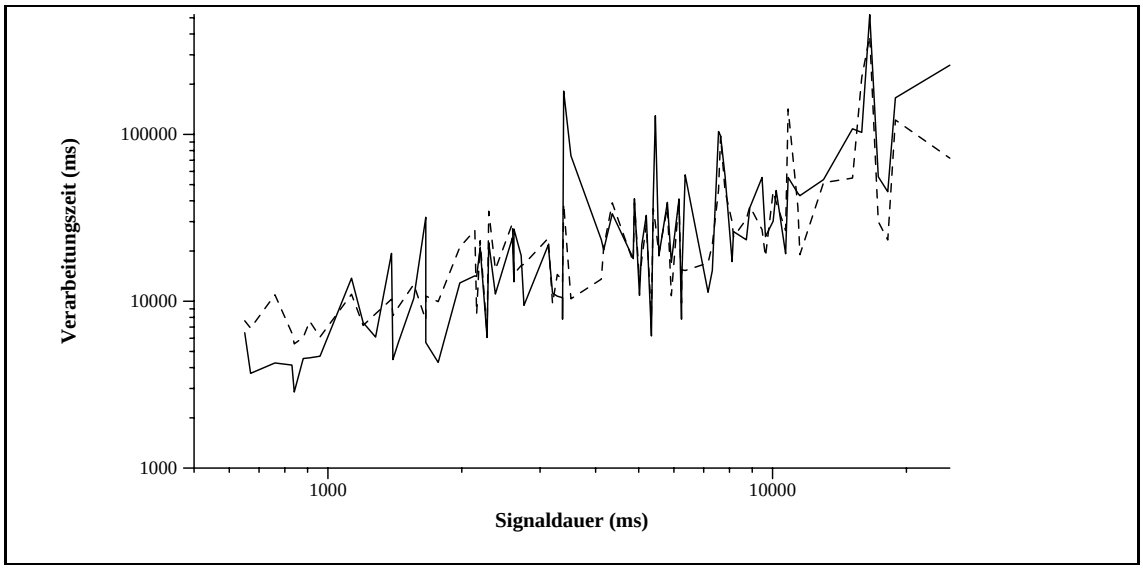


Abbildung 5.4: Vergleich inkrementeller (—) und nichtinkrementeller (- - -) Verarbeitung

Einen genauen Überblick über die unterschiedlichen Verarbeitungszeiten für einzelne Äußerungen gibt Abb. 5.4. Sie zeigt die Verarbeitungszeit für inkrementelle (durchgezogene Linien) und nichtinkrementelle (gestrichelte Linien) Systemläufe in Abhängigkeit von der Länge der einzelnen Äußerung. Als Trend läßt sich beobachten, daß der Einsatz inkrementeller Verfahren für kurze Äußerungen einen stärker positiv wirkenden Effekt zu haben scheint als für vergleichsweise lange Eingaben. Dies liegt wahrscheinlich daran, daß bei langen Eingabesequenzen der Effekt der Auffaltung von linksverbundenen Graphen überproportional zur Geltung kommt. Ein möglicher Ansatzpunkt zur Verbesserung des Verhaltens in diesem Fall besteht etwa darin, eine verzögerte Auswertung einzuführen, die eine Kante erst dann bearbeitet, wenn sichergestellt ist, daß von den Endknoten aus weitere Kanten folgen können. Dadurch ist es möglich, triviale Sackgassen der Länge eins zu ver-

meiden. Prinzipiell läßt sich dieses Verfahren ebenfalls auf eine weitergehende Vorausschau ausdehnen, allerdings ist dann die Einhaltung der Links-Rechts-Inkrementalität immer weniger gewahrt.

5.4 Zusammenfassung

In diesem Kapitel haben wir die zur Validierung der Mehr-Ebenen-Chart und Evaluation des Systems MILC durchgeführten Experimente dargelegt. Die Zielsetzung dieser Untersuchung bestand in drei Aspekten.

Zunächst sollte untersucht werden, inwieweit Hypergraphen eine sinnvolle Grundlage für die Verarbeitung gesprochener Sprache zur Verfügung stellen. Die Ergebnisse der Untersuchung eines Dialogs mit 41 Äußerungen ergaben eine Reduzierung der Wortkantenzahl um 91% sowie eine Verringerung der Analysezeit um 93% zur Anfertigung partieller Strukturanalysen. Damit erweisen sich Hypergraphen als ein mächtiges Instrument, dessen Anwendung die Bearbeitung großer, insbesondere inkrementeller, Graphen überhaupt erst möglich erscheinen läßt.

Das zweite Untersuchungsfeld betrifft das Verhalten von MILC bei der Übersetzung authentischer Dialoge aus der Domäne der Terminvereinbarung. Dazu wurden fünf Dialoge des Verbmobilkorpus untersucht. Zwei dieser Dialoge dienten als Grundlage zur Erstellung der im System verwendeten Grammatiken, während die drei übrigen lediglich als Quelle lexikalischen Materials dienten. Insgesamt ergibt sich, daß die Übersetzung von spontan gesprochenen Äußerungen in etwa sechsfacher Echtzeit angefertigt werden kann. Die englischen Ausgaben sind zu 60,4% approximativ korrekt, indem sie den wesentlichen pragmatischen Gehalt und zentrale propositionale Anteile korrekt wiedergeben. Die stilistische Qualität der Übersetzungen kann dabei allerdings lediglich als sehr rauh beschrieben werden.

Schließlich wurde untersucht, inwieweit inkrementelle Verfahren prinzipiell unterlegen sind, indem sie zunächst mit einer Eingabe umgehen müssen, die etwa eine Größenordnung umfangreicher ist als sie entsprechende nichtinkrementelle Methoden erwarten dürfen. Hier zeigt sich, daß — wenigstens im vorliegenden Fall — eine inkrementelle Verarbeitung nicht wesentlich weniger effizient ist als die nichtinkrementelle Version. Der Geschwindigkeitsunterschied liegt bei etwa 17 % zugunsten der konventionellen Analyse. Auch wenn wir anmerken müssen, daß lediglich eine simplifizierende Modellierung des herkömmlichen Ansatzes durchgeführt wurde, die jegliche Optimierung außer acht läßt, die durch das Vorliegen der gesamten Eingabe angebracht werden könnte, so kann gefolgert werden, daß inkrementelle Verfahren in Zukunft wesentlich stärkere Beachtung finden sollten, da sie bedeutende methodologische Vorteile bei geringfügigen Abstrichen an der Leistungsfähigkeit bieten.

Kapitel 6

Zusammenfassung und Ausblick

Das menschliche Sprachverstehen arbeitet inkrementell.

Diese Beobachtung bildete den Ausgangspunkt der vorliegenden Arbeit. Wir haben versucht darzulegen, daß die Anwendung analoger Prinzipien in der maschinellen Sprachverarbeitung ebenso wünschenswert wie sinnvoll ist. Sie ist wünschenswert, da erst durch die durchgängige Bearbeitung von mehr oder weniger kleinen Teilen der Eingabe fortgeschrittene Systeme zur Mensch-Maschine-Kommunikation oder zur Unterstützung der Kommunikation zwischen Menschen möglich sind. Als Beispiele hierfür mögen erneut Dialogsysteme dienen, die — genau wie Menschen dies untereinander zu tun pflegen — den Partner in wichtigen Fällen unterbrechen können sollten, oder aber Programmsysteme, die eine Simultanübersetzung gesprochener Sprache anstreben.

Sowohl die Leistungsfähigkeit von Computern allgemein, als auch die von Spracherkennungssystemen dringen seit einiger Zeit in Bereiche vor, die eine durchgängig inkrementelle Arbeitsweise möglich machen. Es ist heutzutage durchführbar, eine Menge von Workstations zusammenzuschalten, um mit Hilfe ihrer kombinierten Rechenleistung den Mehraufwand, den die Einführung inkrementeller Verfahren zunächst mit sich bringt, auf ein erträgliches Maß zu reduzieren. Unsere Experimente zeigen, daß eine inkrementelle Vorgehensweise um etwa eine Größenordnung schwieriger ist als vergleichbare nichtinkrementelle Verfahren. Trotzdem liegt die Verarbeitungszeit um lediglich einen geschätzten Faktor drei über der Analysezeit hoch optimierter konventioneller Ansätze.

Die Genauigkeit spracherkennender Systeme auf der anderen Seite ist in den letzten Jahren so gut geworden, daß immer mehr spontan gesprochene Sprache als Eingabe benutzt wird. Die im Rahmen des Projektes Verbmobil aufgenommenen Dialoge stellen dabei jedoch lediglich einen Meilenstein dar, da sie auf einen engen Themenbereich, die Terminvereinbarung, eingeschränkt sind. Die Erkennungsqualität, die den Eingaben in unser System zugrunde liegt, ist mit ca. 70% Wortakkuratheit deutlich geringer als die momentan besten Werte für Offline-Erkenner im Verbmobilkontext. Sie entspricht in etwa den Ergebnissen, die man für weitgehend unrestringierte Sprache, z.B. beim Telefonieren, erwarten könnte. Eine derart hohe Wortfehlerrate wird durch die Benutzung von Wortgraphen relativiert, durch die eine kompakte Darstellung großer Anzahlen von Alternativen möglich

ist. Unter diesen kann dann im Rahmen einer linguistischen Verarbeitung eine gezielte Suche nach plausiblen Interpretationen durchgeführt werden.

Die inkrementelle Art des menschlichen Sprachverstehens zeigt sich auf nahezu allen Ebenen. Wir haben versucht, dies zumindest für die Analyse in den Bereichen Worterkennung sowie der syntaktischen und semantischen Präferenzmodelllierung anhand psycholinguistischer Forschung zu demonstrieren. Ebenso "profitiert" ein Konferenzdolmetscher eminent von der Inkrementalität, indem sie ihm hilft, quellsprachlichen Äußerungen zuzuhören und doch gleichzeitig dazu in der Lage zu sein, schon früh zielsprachliche Sätze zu formulieren. Inkrementalität ist offensichtlich in mindestens dreifacher Hinsicht unabdingbar für die menschliche Sprachperformanz. Zuerst ermöglicht sie die gleichzeitige Verarbeitung nahezu identischer Äußerungsausschnitte auf verschiedenen Ebenen der Analyse. Darüberhinaus können Ergebnisse auf höheren linguistischen Ebenen zur Blockierung unwahrscheinlicher Alternativen in anderen Bereichen der Verstehensprozesses dienen. Und drittens schließlich findet eine Rückkopplung mittels Prädiktionen statt, die erwartungsgesteuert auf einige Bereiche des Sprachverstehensapparates wirkt.

Die Untersuchung der Möglichkeit einer durchgängig inkrementellen Analyse im Rahmen des Dolmetschens spontan gesprochener Sprache bildet folglich die Hauptmotivation zur Anfertigung dieser Arbeit.

Dabei sollte allerdings vermieden werden, ein allzu statisches System zu erstellen, das in der Zukunft nur mit hohem Aufwand modifizierbar ist. Vielmehr war das Ziel, neben einer qualitativen und groben quantitativen Untersuchung inkrementeller Algorithmen innerhalb einer vollständigen Anwendung gleichzeitig auch ein offenes, dynamisches System zu konzipieren, das als Grundlage für weitergehende architektonische Studien im Rahmen inkrementeller Sprachverarbeitung dienen kann. Insbesondere im Bereich der Rückkopplungen zwischen unterschiedlichen Modulen weiß man bisher viel zu wenig, als daß sich bereits eine allgemein akzeptierte architektonische Ausrichtung hätte ausprägen können. Wir haben folglich mit der Implementierung die Maxime verfolgt, sowenig Einschränkungen für das Design alternativer Module wie möglich aufzuprägen.

Die wesentlichen Beiträge dieser Arbeit sehen wir vor allem in:

- Der Charakterisierung wesentlicher Eigenschaften von Wortgraphen sowie der Einführung von Algorithmen, die eine Bearbeitung auch umfangreicher Graphen innerhalb der automatischen Sprachverarbeitung ermöglichen. Neben einer veränderten Sicht auf die Beurteilung der Größe von Graphen, die nicht ausschließlich erkennungszentriert ist, sondern sich an der Verwendung der Graphen in einem größeren Zusammenhang orientiert, sind hier außerdem Methoden zur Reduktion der Komplexität von Wortgraphen zu nennen. Für die Verarbeitung inkrementeller Graphen schließlich ist die theoretische Aufarbeitung und praktische Anwendung von Hypergraphen-Mechanismen ausgesprochen nützlich, hilft sie doch, bislang außerhalb der Reichweite aktueller Verarbeitungsmethoden liegende Eingaben auf ein Größenmaß zu reduzieren, das in vielen Fällen eine erfolgreiche Benutzung erlaubt.

- Die Konzeption eines Architekturschemas, das umfangreich und zugleich offen genug ist, um in der Zukunft Forschungsarbeiten im Gebiet der inkrementellen, interaktiven Sprachverar-

beitung zu ermöglichen. Dieses Architekturschema besteht im wesentlichen aus drei Komponenten, welche die wichtigen Bereiche Datenstrukturen, Repräsentationen und Systemintegration abdecken. Auf der Ebene der Datenstrukturen haben wir die Mehr-Ebenen-Chart eingeführt, eine weitgehende Verallgemeinerung bislang üblicher Charts. Mehr-Ebenen-Charts sind zur verteilten, synchronisationsarmen Speicherung mannigfaltiger Aspekte einer Eingabeäußerung in der Lage. Sie sind unabhängig von einer zentralen Speicherung oder Kontrolle und sorgen durch die integrierte Darstellung aller Hypothesen innerhalb eines Systems für eine unkomplizierte Interaktion von Komponenten. Jedweder Austausch von Information findet über Hyperkanten der Chart statt, so daß keine Schnittstellenproblematik auftritt. Zugleich stellt die Vereinigung aller Kanten stets ein kohärentes Bild des aktuellen Bearbeitungsstandes eines Systems dar. Mehr-Ebenen-Charts dehnen die bisher übliche Form der graphenartigen Datenhaltung für Worthypothesen auf alle Ebenen eines Systems aus, sie sorgen mithin für die Etablierung eines integrierten Interpretationsgraphen für eine Anwendung.

Auf der Ebene der Repräsentationen ist durch die Konzeption und Realisierung eines verschiebungsinvarianten getypten Merkmalformalismus mit Zulässigkeitsfunktion eine hocheffiziente, deklarative Sprache zur Beschreibung linguistischer Objekte entstanden. Diese Sprache erlaubt — wie einige ihrer Vorgänger — eine graphenartige Spezifikation getypter Merkmalstrukturen. Der Ausgangspunkt für die Entwicklung des in der vorliegenden Arbeit benutzten Formalismus war die Ansicht, daß für hochgradig verteilte, parallele Systeme — womit wir sowohl die intermodulare wie auch die intramodulare Parallelität meinen — eine stark verzeigerte Speicheraufteilung herkömmlicher Ansätze ungeeignet ist. Stattdessen wurde hier eine Implementierung verfolgt, die eine blockorientierte Allokation vorsieht. Sie ist extrem kompakt und erlaubt den Transport von Merkmalstrukturen in andere Adreßräume ohne aufwendige Linearisierung und Rekonstruktion. Der Formalismus ist allen Komponenten direkt zugänglich und realisiert eine uniforme Darstellung unterschiedlichster Arten linguistischen Wissens.

Auf der Ebene der Systemintegration schließlich ist ein Architekturschema zur Erstellung heterogener Systeme von Komponenten entstanden, das eine einheitliche Sicht auf die Kommunikation innerhalb eines Systems zuläßt. Der kanalorientierte Entwurf des Infrastruktursystems ICE garantiert einen hochgradig effizienten und konzeptionell geradlinigen Informationsaustausch. Die hier entwickelte Klasse von Anwendungen besteht aus weitgehend unabhängigen, heterogenen Modulen, die ohne Rückgriff auf eine zentrale Instanz miteinander zusammenwirken können. Eine flexible Konfiguration erlaubt die weitgehende Kontrolle über die tatsächliche Topologie eines Systems, ohne auf die Arbeitssweise einzelner Module übermäßigen Einfluß zu nehmen. Die Übermittlung getypter Nachrichten befreit den Entwickler von der Notwendigkeit, ein hohes Maß an Aufmerksamkeit auf infrastrukturelle Belange zu verwenden. Der Ablauf einer individuellen Komponente erfolgt hierbei datengetrieben; die verteilte Überwachung der Terminierung wird automatisch vom architektonischen Rahmen geleistet.

- Der vollständigen Implementierung eines Dolmetschsystems zur Übersetzung spontan gesprochener deutscher Äußerungen in englische Oberflächenrepräsentationen in der Domäne der Terminvereinbarung. Dieses System (MILC, *M*achine *I*nterpreting with *L*ayered *C*harts)

ist innerhalb des vorher beschriebenen Architekturrahmens verwurzelt. Es erbringt zum einen den Nachweis, daß durchgängig inkrementelles automatisches Dolmetschen unter Beibehaltung des Chart-Paradigmas möglich ist. Jedwede Form der Hypothese wird dabei als potentielle Interpretation eines Ausschnittes der Eingabeäußerung aufgefaßt. Dies gilt für Analyse und Generierung genauso wie für den Transfer, der in dieser Form bisher nicht mit der Unterstützung chartbasierter Verfahren durchgeführt wurde. Die Inkrementalität wird bis in die Konstruktion von englischen Oberflächenhypothesen beibehalten, was sich als sich zeitlich ändernder Strom von größer werdenden Fragmenten äußert. Ebenso wichtig wie die Bereitstellung eines funktionsfähigen, vollständigen Systems ist allerdings die Möglichkeit, an diesem Änderungen vorzunehmen. Exemplarisch wird dies in der vorliegenden Arbeit durch die Integration von Idiomen in den Übersetzungsprozeß gezeigt. Sie ist orthogonal zur Hauptverarbeitungsrichtung angelegt und dient in erster Linie dazu, aufzuzeigen, welche Form und Anwendungsweise zukünftige Komponenten besitzen können. Erkannte Idiome werden im restlichen System bevorzugt behandelt und modifizieren zugleich Suchräume, deren Beschreiten andernfalls unnötigen Aufwand bedeuten würde.

Innerhalb dieses Experimentierrahmens bieten sich zahlreiche Erweiterungsmöglichkeiten an, welche die Effizienz und Bandbreite der abgedeckten Phänomene betreffen. Unmittelbar durchführbar ist dies für die Ausnutzung weiterer im Sprachsignal vorhandener Information, die von der Worterkennung nicht unmittelbar ausgewertet wird. Insbesondere die Integration prosodischer Evidenz zur Bestimmung von Phrasengrenzen und Akzentuierungen wird einen relevanten Einfluß auf die Qualität der Übersetzungen und die Systemperformanz haben. Modusinformation kann zur Disambiguierung dienen. Darüberhinaus erscheint die Etablierung einer Dialogkomponente der nächste logische Schritt, um eine Bearbeitung vollständiger Gespräche durchführen zu können; dies führt nicht nur dazu, daß eine weitere Quelle von Wissen bezüglich der Auflösung von Mehrdeutigkeiten zur Verfügung steht, sondern auch zur Erhöhung der Ausgabequalität durch die Anwendung dialogischer Phänomene. Eine zusätzliche, ebenso einfache wie potentiell lohnende Erweiterung ergibt sich aus der Inkorporation beispielorientierter Übersetzungen. Diese können weitgehend analog zu Idiomen behandelt werden und sorgen im Erfolgsfall für eine extrem schnelle und zuverlässige Verarbeitung zumindest von Teilen der Eingabe.

Aus prinzipiell architektonischen Erwägungen heraus ist eine Kopplung des Worterkenners mit Modulen zur linguistischen Analyse lohnend, um den Einfluß linguistischer Prozesse auf die Worterkennung zu erforschen. Insbesondere gilt dies für die Anwendung von Restriktionen aus der Semantik und der Dialogverarbeitung, die bisher wenig untersucht wurden.

Schließlich sollte versucht werden, eine dynamische Ausweitung von Suchräumen nur dann vorzunehmen, wenn mit der bislang zur Verfügung stehenden Information eine erfolgreiche Analyse nicht möglich ist. Der Ansatz, den wir in dieser Richtung verfolgen, operiert inkrementell stets auf dem besten Pfad durch den bisher besuchten Graphen. Die Bewertungen von Kanten sind so gestaltet, daß für den Fall einer unwahrscheinlichen Analyse auf bisher zurückgehaltene akustische Hypothesen rekurriert werden kann.

Aus einem größeren Zusammenhang betrachtet, ist die Anwendungsbandbreite von Systemen, die

auf Mehr-Ebenen-Charts beruhen bzw. die auf dem in dieser Arbeit entwickelten System MILC aufbauen, sehr groß. Das automatische Dolmetschen stellt nur einen Ausschnitt der Möglichkeiten dar, wenngleich die Übersetzung gesprochener Sprache einen sehr hohen Anspruch an Systeme stellt. Desgleichen sind etwa Zugangsmechanismen zu Informationssystemen (etwa Datenbanken) oder Anwendungen in der natürlichsprachlichen Steuerung (z.B. Robotik) denkbar.

Neben dem dezidierten Einsatz für gesprochene Sprache ist mit dem vorliegenden Modell jedoch auch die Behandlung geschriebener Sprache leicht möglich. Durch die graphenartige Strukturierung aller Hypothesenräume ist die Darstellung von Ambiguität ohne konzeptionellen Mehraufwand erreichbar. Gerade im Bereich von hochgradig modularen Systemen, welche die Behandlung multilingualer Eingaben zum Ziel haben, bietet sich ein einheitliches Architekturschema an, wenn ein Anwendungssystem in kurzer Zeit auf unterschiedliche Quellsprachen adaptiert werden soll.

Eine Architektur, wie sie in dieser Arbeit dargelegt wurde, macht es zugleich möglich, unterschiedliche Aspekte von Information zu speichern und in einem Systemzusammenhang konsistent und effizient nutzbar zu machen. Auf den Bereich der Sprachverarbeitung beschränkt, ist angedeutet worden, wie dialogische Information in einer Anwendung, die äußerungsübergreifende Phänomene modelliert, durch eine einfache Erweiterung genutzt werden kann. In ähnlicher Weise kann prosodisches Wissen als Hypothesen über zeitliche Ausschnitte einer Äußerung repräsentiert werden.

Ein weiterer Blickwinkel macht deutlich, daß der Effekt einer graphenorientierten Darstellungsform nicht nur auf eigentlich sprachinhärente Merkmale restringiert ist. Die Erweiterung auf multimodale Ein- und Ausgaben sowie die Repräsentation von Ereignissen und Objekten sind ebenfalls denkbar. Unmittelbar einsichtig ist dies für alle Arten von Eingabeinformation, die zeitlich und thematisch mit der Spracheingabe im engeren Sinn korreliert sind. Zeigegesten eines Benutzers, seien sie durch Zeigeinstrumente wie der Maus, durch Augen- oder Handbewegungen realisiert, sind in einfacher Weise in den Systemverlauf integrierbar. Sie lassen sich etwa — analog zur Behandlung von Idiomen im vorliegenden System — in einer spezialisierten Komponente erkennen, die als Ausgabe eine Handlungsbeschreibung in Form einer Merkmalstruktur zusammen mit ihrer zeitlichen Ausdehnung liefert. Selbstverständlich müssen die restlichen Komponenten durch eine geeignete Modifikation ihrer Grammatiken dazu in der Lage sein, diese Information auch auszunutzen.

Eine weitere Form eines zusätzlichen Eingabekanals bildet die Auswertung von Lippenbewegungen während des Sprechens. Hier ist die zeitliche und konzeptuelle Assoziation mit der Originaleingabe in Form eines Wortgraphens unmittelbar einleuchtend. Die Information über die sichtbare Stellung der Lippen kann zur Disambiguierung dienen, indem ein Artikulationsmodell für die sich aus dem Wortgraphen ergebenden Wörter aufgestellt und mit der visuellen Eingabe verglichen wird. Auch hier stellen die zusätzlichen Informationen Hypothesen über zeitliche Ausschnitte des Eingabesignals dar.

Komplizierter, aber dennoch prinzipiell möglich, ist die Integration arbiträrer Objekte und ihrer sich zeitlich ändernden Zustände. Ausgangspunkt für ein derartiges Verfahren muß stets eine Ausdehnung der Abdeckung der Mehr-Ebenen-Chart über isolierte Äußerungen hinaus in Richtung auf eine kontinuierliche Repräsentation der Zeitachse sein. Um die Datenmenge dadurch nicht in unhandhabbare Dimensionen wachsen zu lassen, ist zunächst erforderlich, daß ein Fokuskonzept ausgearbeitet

wird. Der Wechsel des Fokus innerhalb des Graphen hat dann zur Folge, daß irrelevant gewordene Kanten entfernt werden und lediglich zentrale Analysen in der Chart verbleiben.

Das zweite Problem betrifft die Einführung einer Methode zur Kombination linguistischen und extralinguistischen Wissens. Dies kann in der vorliegenden Arbeit nicht geleistet werden. Allerdings ist die Art der hier verwendeten Graphenstrukturen außerordentlich gut geeignet, zeitliche Abhängigkeiten durch topologische Eigenschaften von Graphen abzubilden. Im einfachsten Fall lassen sie sich als Schnittoperationen auf Knotenmengen bzw. als Überprüfung der Erreichbarkeitsrelation modellieren.

Die Repräsentation nichtsprachlichen Wissens muß außerdem in den zugrundeliegenden formalen Beschreibungen, also dem verwendeten Typenverband, reflektiert werden. Es ist zu verhindern, daß nichtsprachliche Objekte und Zustandsdarstellungen unkontrolliert mit linguistischen Operationen interagieren. Vielmehr muß die Verarbeitung solchermaßen heterogenen Wissens stets explizit gemacht werden. Das schließt jedoch nicht aus, daß dieselben Mechanismen, wie sie in dieser Arbeit entwickelt wurden, nicht auch auf Kanten anderer Herkunft angewendet werden können, zumal durch die multiplen Ebenen einer Mehr-Ebenen-Chart unterschiedliche Aspekte von Information gut gegeneinander abgeschirmt werden können, ohne dadurch ihre integrierte Verarbeitung zu stören. Um nur ein Beispiel zu nennen, können Objekt- und Handlungsreferenzen in einem Wartungsszenario, in dem ein Benutzer von einem System instruiert wird (wie etwa dem Nachfüllen von Motoröl durch technisch nicht versierte Autofahrer), dadurch stark erleichtert werden. Deiktische Ausdrücke (“Nicht den!”, wenn der Benutzer statt des Öldeckels den Deckel des Kühlers öffnen will) lassen sich so einfacher generieren und disambiguieren. In diesem hypothetischen Beispiel war bisher von einem Öldeckel die Rede, die Objektmodellierung kennt jedoch auch andere Deckel innerhalb des Motorraumes. Durch die zeitliche Abhängigkeit der Aufforderung, den Öldeckel zu öffnen und der Handlung des Benutzers kann eine Eingrenzung der Referenz erfolgen und z.B. korrekt auf das eigentlich gemeinte Objekt referiert werden. In diesem Anwendungsfeld zeigt sich außerdem erneut die Notwendigkeit einer inkrementellen, im Idealfall zeitsynchronen, Verarbeitung. Das System sollte beispielsweise dazu in der Lage sein, den Benutzer unmittelbar zu warnen, um Fehler bei der Bedienung zu verhindern.

Alle hier aufgeführten hypothetischen Einsatzbereiche zeigen einige gemeinsame Eigenschaften: Die wesentliche Dimension der Eingabe ist die Zeit. Komplexe Objekte (linguistische Beschreibungen, Repräsentationen von Handlungen) dienen zur Darstellung von Wissen. Der Einsatz inkrementeller Techniken ist notwendig oder zumindest wünschenswert. Eine an Graphen orientierte Datenstruktur wie die Mehr-Ebenen-Chart bildet zusammen mit einem architektonischen Systemumfeld wie dem in dieser Arbeit beschriebenen eine gute Ausgangsbasis zur Untersuchung solcher Phänomene.

It is a curious fact, and one to which no one knows quite how much importance to attach, that something like 85% of all known worlds in the Galaxy, be they primitive or highly advanced, have invented a drink called jynnan tonnyx, or gee-N'N-T'N-ix, or jinond-o-nicks, or any one of a thousand or more variations on the same phonetic theme. The drinks themselves are not the same, and vary between the Sivolvian 'chinanto/mnigs' which is ordinary water served at slightly above room temperature, and the Gagrakackan 'tzjin-anthony-ks' which kills cows at a hundred paces; and in fact the one common factor between all of them, beyond the fact that the names sound the same, is that they were all invented and named *before* the worlds concerned made contact with any other worlds.

What can be made of this fact? It exists in total isolation. As far as any theory of structural linguistics is concerned it is right off the graph, and yet it persists. Old structural linguists get very angry when young structural linguists go on about it. Young structural linguists get deeply excited about it and stay up late at night convinced that they are very close to something of profound importance, and end up becoming old structural linguists before their time, getting very angry with the young ones. Structural linguistics is a bitterly divided and unhappy discipline, and a large number of its practicioners spend too many nights drowning their problems in Ouisghian Zodahs. (Adams, 1980, p. 138)

Literaturverzeichnis

Abney, Steven. 1991. Parsing By Chunks. In Robert Berwick, Steven Abney, und Carol Tenny, editors, *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht.

Abney, Steven. 1996. Partial Parsing via Finite-State Cascades. In *Proceedings of the ESSLLI '96 Robust Parsing Workshop*.

Adams, Douglas. 1980. *The restaurant at the end of the universe*. London: Pan Books.

Agnäs, Marie-Susanne, Hiyan Alshawi, Ivan Bretan, und David Carter et. al. 1994. Spoken Language Translator: First-Year Report. Research Report R94:03, SICS, Stockholm, Januar.

Aigner, Martin. 1984. *Graphentheorie: Eine Entwicklung aus dem 4-Farben Problem*. Teubner Studienbücher: Mathematik. Stuttgart: Teubner.

Aït-Kaci, Hassan, Robert Boyer, Patrick Lincoln, und Roger Nasr. 1989. Efficient Implementation of Lattice Operations. *ACM Transactions on Programming Languages and Systems*, 11(1):115–146, Januar.

Alexandersson, Jan, Elisabeth Maier, und Norbert Reithinger. 1995. A Robust and Efficient Three-Layered Dialog Component for a Speech-to-Speech Translation System. In *Proc. of the 7 th EACL*, S. 188–193, Dublin, Ireland.

Alexandersson, Jan, Norbert Reithinger, und Elisabeth Maier. 1997. Insights into the Dialogue Processing of Verbmobil. In *Proc. of the 5th Conference on Applied Natural Language Processing*, Washington, D.C.

Allen, James F. 1987. *Natural Language Understanding*. The Benjamin/Cummings Series in Computer Science. Menlo Park, CA.: Benjamin/Cummings.

Allen, James F., Bradford W. Miller, Eric K. Ringger, und Teresa Sikorski. 1996. A Robust System for Natural Spoken Dialogue. In *Proc. of the 34 nd ACL*, S. 62–70, Santa Cruz, CA, Juni.

Allen, James F., Lenhart K. Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel G. Martin, Bradford W. Miller, Massimo Poesio, und David R. Traum. 1994. The TRAINS Project: A case study in building a conversational planning agent. TRAINS Technical Note 94-3, University of Rochester, Rochester, NY, September.

Amtrup, Jan W. 1992. Parallele Strukturanalyse Natürlicher Sprache mit Transputern. ASL-TR 44-92/UHH, Univ. Hamburg.

Amtrup, Jan W. 1994a. ICE–Intarc Communication Environment: Design und Spezifikation. Verbmobil Memo 48, Univ. Hamburg, September.

Amtrup, Jan W. 1994b. Transfer and Architecture: Views from Chart Parsing. Verbmobil Report 8, Univ. Hamburg, Maerz.

Amtrup, Jan W. 1995a. Chart-based Incremental Transfer in Machine Translation. In *Proceedings of the sixth International Conference on Theoretical and Methodological Issues in Machine Translation, TMI '95*, S. 188–195, Leuven, Belgien, Juli.

Amtrup, Jan W. 1995b. ICE–Intarc Communication Environment: User's Guide and Reference Manual. Version 1.4. Verbmobil Technical Document 14, Univ. Hamburg, Dezember.

Amtrup, Jan W. 1995c. Parallel Parsing: Different Distribution Schemata for Charts. In *Proceedings of the 4^{th} International Workshop on Parsing Technologies (IWPT95)*, S. 12–13, Prague, September. Charles University.

Amtrup, Jan W. 1997a. ICE: A Communication Environment for Natural Language Processing. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA97)*, Las Vegas, NV, Juli.

Amtrup, Jan W. 1997b. Layered Charts for Speech Translation. In *Proceedings of the Seventh International Conference on Theoretical and Methodological Issues in Machine Translation, TMI '97*, Santa Fe, NM, Juli.

Amtrup, Jan W. 1997c. Perspectives for Incremental MT with Charts. In Christa Hauenschild und Susanne Heizmann, editors, *Machine Translation and Translation Theory. Perspectives of Cooperation*, Text, Translation, Computational Processing (TTCP), number 1. Mouton de Gruyter.

Amtrup, Jan W. 1998. Incremental Speech Translation: A Layered Chart Approach. In *28. Jahrestagung der gesellschaft für Informatik*, Magdeburg, September.

Amtrup, Jan W. und Jörg Benra. 1996. Communication in large distributed AI Systems for Natural Language Processing. In *Proceedings of the 16th international Conference on Computational Linguistics*, S. 35–40, Copenhagen, Denmark, August. Center for Sprogteknologi.

Amtrup, Jan W., Henrik Heine, und Uwe Jost. 1996. What's in a Word Graph — Evaluation and Enhancement of Word Lattices. Verbmobil Report 186, Univ. Hamburg, Hamburg, Dezember.

Amtrup, Jan W., Henrik Heine, und Uwe Jost. 1997. What's in a Word Graph — Evaluation and Enhancement of Word Lattices. In *Proc. of Eurospeech 1997*, Rhodes, Greece, September.

Amtrup, Jan W. und Susanne J. Jekat. 1995. Segmentation of Spoken Language for NLP. In *KI95-Activities: Workshops, Posters, Demos*, S. 298–299, Bielefeld, September.

Amtrup, Jan W. und Volker Weber. 1998. Time Mapping with Hypergraphs. In *Proc. of the 17 th COLING*, Montreal, Canada.

Anderson, Linda. 1994. Simultaneous Interpretation: Contextual and Translation Aspects. In Sylvie Lambert und Barbara Moser-Mercer, editors, *Bridging the Gap: Empirical Research in Simultaneous Interpretation*. John Benjamins Publishing Co., S. 101–120.

Aubert, Xavier und Hermann Ney. 1995. Large Vocabulary Continuous Speech Recognition Using Word Graphs. In *ICASSP 95*.

Ausiello, Giorgio, Giuseppe F. Italiano, Alberto Marchetti Marchetti-Spaccamela, und Umberto Nanni. 1991. Incremental Algorithms for Minimal Length Paths. *Journal of Algorithms*, 12:615–638.

Backofen, Rolf, Lutz Euler, und Günther Görz. 1991. Distributed Disjunctions in LIFER. In *Proc. International Workshop on Processing Declarative Knowledge*, S. 161–170, Berlin, Heidelberg, New York. Springer Verlag.

Batliner, Anton, Susanne Burger, und Andreas Kiessling. 1994. Außergrammatische Phänomene in der Spontansprache. VM Technisches Dokument 2, Univ. München.

Beaven, J.L. 1992. *Lexicalist Unification Based Machine Translation.* Ph.D. thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, UK.

Beskow, Björn. 1993. Unification Based Transfer: Multilingual Support for Translation and Writing. Draft, Uppsala University, Uppsala, Sweden, Februar.

Billot, Sylvie und Bernard Lang. 1989. The Structure of Shared Forests in Ambiguous Parsing. In *Proc. of the 27 th ACL*, S. 143–151, Vancouver, Juni.

Block, H. U. 1997. The Language Components in Verbmobil. In *Proc. ICASSP '97*, S. 79–82, Munich, Germany, April.

Boddy, Mark und Thomas L. Dean. 1994. Deliberation Scheduling for Problem Solving in Time-Constrained Environments. *Artificial Intelligence*, 67(2):245–285.

Boitet, Christian und Mark Seligman. 1994. The "Whiteboard" Architecture: A Way to Integrate Heterogeneous Components of NLP systems. In *COLING-94: The 15th International Conference on Computational Linguistics*, Kyoto, Japan.

Brandstädt, Andreas, 1994. "Kürzeste Wege". In *Graphen und Algorithmen*, Kapitel 5, S. 106–123. Stuttgart: Teubner.

Bresnan, Joan, Ed. 1982. *The Mental Representation of Grammatical Relation*. Cambridge, MA.: MIT Press.

Brew, Chris. 1992. Letting the Cat out of the Bag: Generation for Shake-and-Bake MT. In *COLING-92: The 15th International Conference on Computational Linguistics*, S. 610–616, Nantes, France.

Brietzmann, Astrid. 1992. “Reif für die Insel”. Syntaktische Analyse natürlich gesprochener Sprache durch bidirektionals Chart-Parsing. In Helmut Mangold, editor, *Sprachliche Mensch-Maschine-Kommunikation*. R. Oldenbourg Verlag, S. 103–116.

Briscoe, Edward J. 1987. *Modelling Human Speech Comprehension: A Computational Approach*. Wiley.

Brown, P., J. Cocke, S.A. Della Pietra, Felinek, J.D. F. Lafferty, R.L. Mercer, und P.S. Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16:79–85.

Brown, Ralf und Robert Frederking. 1995. Applying Statistical English Language Modelling to Symbolic Machine Translation. In *TMI95P*, TMI95L.

Bub, Thomas, Wolfgang Wahlster, und Alex Waibel. 1997. Verbmobil: The Combination of Deep and Shallow Processing for Spontaneous Speech Translation. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, S. 1/71–1/74, München.

Burns, Alan. 1988. *Programming in OCCAM 2*. Reading, Ma.: Addison-Wesley.

Buschbeck-Wolf, Bianka et al. 1995. Transfer in the Verbmobil Demonstrator. Technical report, IAI, Saarbrücken.

Carlson, Lauri und Maria Vilkuna. 1990. Independent Transfer Using Graph Unification. In *Proc. of the 13 th COLING*, S. 3/60–3/63, Helsinki, Finnland.

Carpenter, Bob. 1992. *The Logic of Typed Feature Structures*. Tracts in Theoretical Computer Science. Cambridge: Cambridge University Press.

Carpenter, Bob und Gerald Penn. 1998. ALE – The Attribute Logic Engine User’s Guide. Version 3.0 Beta. Technical report, Bell Labs/Univ. Tübingen, Maerz.

Carpenter, Bob und Yan Qu. 1995. An Abstract Machine for Attribute-Value Logics. In *Proceedings of the 4th International Workshop on Parsing Technologies (IWPT95)*, S. 59–70, Prague. Charles University.

Carter et al., David. 1997. Translation Methodology in the Spoken Language Translator: An Evaluation. In *ACL Workshop on Spoken Language Translation.*

Carver, N. und V. Lesser. 1994. The Evolution of Blackboard Control Architectures. *Expert Systems with Applications*, 7(1):1–30.

Caspari, Rudolf und Ludwig Schmid. 1994. Parsing und Generierung in TrUG. Verbmobil-Report 40, Siemens AG.

Chen, Wai-Kai. 1971. *Applied Graph Theory*. Applied Mathematics and Mechanics, Vol. 13. Amsterdam: North Holland.

Chernov, G. V. 1994. Message Redundancy and Message Anticipation in Simultaneous Interpretation. In Lambert und Moser-Mercer, editors, *Bridging the Gap: Empirical Research in Simultaneous Interpretation*. John Benjamins, S. 139–153.

Cheston, Grant A. 1976. *Incremental Algorithms in Graph Theory*. Ph.D. thesis, Univ. of Toronto, Maerz.

Chomsky, Noam. 1959. On certain formal properties of grammars. *Information and Control*, 2(2):137–167.

Chomsky, Noam. 1995. *The Minimalist Program*. Current Studies in Linguistics, Nr.. 28. Cambridge, MA: The MIT Press.

Copestake, Ann, Dan Flickinger, Rob Malouf, Susanne Riehemann, und Ivan Sag. 1995. Translation using Minimal Recursion Semantics. In *Proceedings of the sixth International Conference on Theoretical and Methodological Issues in Machine Translation, TMI '95*, Leuven, Belgien.

Corbin, John R. 1990. *The Art of Distributed Applications*. Sun Technical Reference Library. New York: Springer-Verlag.

Cormen, Thomas H, Charles E. Leiserson, und Ronald L. Rivest. 1990. *Introduction to Algorithms*. Cambridge, MA: MIT Press.

Denecke, Matthias. 1997. A Programmable Multi-Blackboard Architecture for Dialogue Processing Systems. In *ACL97P*, ACL97L.

Diagne, Abdel Kader, Walter Kasper, und Hans-Ulrich Krieger. 1995. Distributed Parsing with HPSG Grammars. In *Proceedings of the 4th International Workshop on Parsing Technologies (IWPT95)*, S. 79–86, Prague, September. Charles University.

Dorna, Michael. 1992. Erweiterung der Constraint-Logiksprache CUF um ein Typensystem. Master's thesis, University of Stuttgart.

Dorna, Michael und Martin C. Emele. 1996. Efficient Implementation of a Semantic-based Transfer Approach. In *Proc. of the 12 th ECAI*, Budapest, Hungary, August.

Dorr, Bonnie Jean. 1993. *Machine Translation: A View from the Lexicon*. Cambridge, MA.: MIT Press.

Doyle, Jon. 1979. A truth Maintenance System. *Artificial Intelligence*, 12:231–272.

Earley, Jay. 1970. An Efficient Context-Free Parsing Algorithm. *Communications of the ACM*, 13:94–102.

Eberle, Kurt, Walter Kasper, und Christian Rohrer. 1992. Contextual Constraints for MT. In *Proc. of the 4 th Int. Conf. on Theoretical and Methodological Issues in Machine Translation*, S. 213–224, Montreal, Canada, Juni.

Eisele, Andreas und Jochen Dörre. 1988. Unification of Disjunctive Feature Structures. In *Proc. of the 26 th ACL*, Buffalo, NY, Juni.

Elsner, Anja und Alexandra Klein. 1996. Erkennung des prosodischen Fokus und die Anwendung im dialogaktbasierten Transfer. Verbmobil Memo 107, Universität Bonn, Universität Hamburg.

Emele, Martin, Ulrich Heid, Stefan Momma, und Rémi Zajac. 1991. Interactions between Linguistic Constraints: Procedural vs. Declarative Approaches. *Machine Translation*, S. 61–98.

Emele, Martin E. und Rémi Zajac. 1990. Typed Unification Grammars. In *Proc. of the 13 th COLING*, S. 293–298, Helsinki, Finnland.

Engelmore, Robert und Tony Morgan. 1988. *Blackboard Systems*. Reading, MA: Addison-Wesley Publishing Company.

Erbach, G. und B. Krenn. 1994. Idioms and Support-Verb Constructions. In J. Nerbonne, K. Netter, und C. Pollard, editors, *German Grammar in HPSG*. CSLI, Stanford, Ca.

Erman, Lee D., Frederick Hayes-Roth, Victor R. Lesser, und Raj Reddy. 1980. The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys*, 12(2):213–253.

Fink, Gernot A., Franz Kummert, und Gerhard Sagerer. 1994. A Close High-Level Interaction Scheme for Recognition and Interpretation of Speech. In *Proc. ICSLP-94*, S. 2183–2186, Yokohama.

Finke, Michael, Jürgen Fritsch, Petra Geutner, Klaus Ries, und Torsten Zepenfeld. 1997. The Janus RTk Switchboard/Callhome 1997 Evaluation System. In *Proceedings of LVCSR Hub5-e Workshop*, Baltimore, MA.

Finkler, Wolfgang. 1996. *Automatische Selbstkorrektur bei der inkrementellen Generierung gesprochener Sprache unter Realzeitbedingungen — Ein empirisch-simulativer Ansatz unter Verwendung eines Begründungsverwaltungssystems*. Ph.D. thesis, Universität des Saarlandes, Saarbrücken.

Finkler, Wolfgang und Anne Schauder. 1992. Effects of Incremental Output on Incremental Natural Language Generation. In *Proc. of the 10 th ECAI*, S. 505–507, Vienna, Austria, August.

Florey, Friedemann. 1998. Automatisches Layout Inkrementeller Graphen. Unveröff. Studienarbeit, Universität Hamburg.

Fodor, J.A. 1983. *The Modularity of Mind*. Cambridge, MA: MIT Press.

Friederici, Angela. 1992. Natürliche Sprachverarbeitung: Funktionen und Dysfunktionen. *Künstliche Intelligenz*, 6(2):13–19.

Fung, Pascale und Kenneth Church. 1994. K-vec: a new approach for aligning parallel texts. In *COLING-94: The 15th International Conference on Computational Linguistics*, Kyoto, Japan.

Furuse, Osamu. 1994. Transfer-Driven Machine Translation. In *Proceedings of AMTA94*, S. 225–226.

Furuse, Osamu und Hitoshi Iida. 1992. Cooperaton between Transfer and Analysis in Example-Based Framework. In *COLING-92: The 15th International Conference on Computational Linguistics*, S. 645–651, Nantes, France.

Furuse, Osamu und Hitoshi Iida. 1996. Incremental Translation Utilizing Constituent Boundary Patterns. In *Proc. of the 16 th COLING*, S. 412–417, Copenhagen, Denmark.

Gale, Willliam und Kenneth Church. 1991. A program for aligning sentences in bilingual corpora. In *Proc. of the 29 th ACL*.

Gaskell, M. G. und W. D. Marslen-Wilson. 1995. Modeling the Perception of Spoken Words. In J.D. Moore und J.F. Lehman, editors, *Proceedings of the 17th Annual Conference of the Cognitive Science Society*, S. 19–24, Mahwah, NJ. Erlbaum.

Gazdar, Gerald. 1983. NLs, CFLs and CF-PSGs. In Karen Sparck Jones und Yorick Wilks, editors, *Automatic Natural Language Parsing*. Ellis Horwood Ltd., Chichester, S. 81–93.

Geist, Al, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek, und Vaidy Sunderam. 1994. PVM3 User's Guide and Reference Manual. Technical Report ORNL/TM-12187, Oak Ridge National Laboratory, Oak Ridge, Te., Mai.

Gerver, D. 1997. Empirical Studies of Simultaneous Interpretation: A Review and a Model. In R.W. Brislin, editor, *Translation: Applicatons and Research*. Gardner Press, New York, S. 165–207.

Goldman-Eisler, F. 1972. Segmentation of Input in Simultaneous Interpretation. *Journal of Psycholinguistuc Research*, 1:127–140.

Gondran, Michel und Michel Minoux. 1984. *Graphs and algorithms*. Wiley-Interscience Series in Discrete Mathematics. Chichester: John Wiley & Sons.

Görz, Günther. 1988. *Strukturanalyse natürlicher Sprache*. Bonn: Addison Wesley.

Görz, Günther. 1993. Kognitiv orientierte Architekturen für die Sprachverarbeitung. Technical Report ASL-TR-39-92, Universität Erlangen-Nürnberg, Februar.

Görz, Günther und Marcus Kesseler. 1994. Anytime Algorithms for Speech Parsing? In *COLING-94: The 15th International Conference on Computational Linguistics*, Kyoto, Japan.

Görz, Günther, Marcus Kesseler, Jörg Spilker, und Hans Weber. 1996. Research on Architectures for Integrated Speech/Language Systems in Verbmobil. In *Proc. of the 16 th COLING*, S. 484–489, Copenhagen, Denmark, August.

Grabski, Michael. 1990. Transfer Statements as Conditional Constraints. WP 18/90. Eurotra-D Working Papers, IAI, Saarbrücken.

Graham, Ian und Tim King. 1990. *The Transputer Handbook*. New York, London et al.: Prentice Hall.

Guard, J.R. 1964. Automated Logic for Semi-Automated Mathematics. Scientific Report 1, Air Force Cambridge Research Laboratory, Bedford, Ma.

Hager, Jochen und Martin Moser. 1989. An Approach to Parallel Unification Using Transputers. In *German Workshop on Artificial Intelligence*, S. 83–91, Eringerfeld.

Hahn, Walther v. 1992. Von der Verknüpfung zur Integration: Kontrollstrategie oder kognitive Architektur. In *Procceedings of KONVENS92*, S. 1–10, Berlin. Springer Verlag.

Hahn, Walther v. und Jan W. Amtrup. 1996. Speech-to-Speech Translation: The Project Verbmobil. In *Proceedings of SPECOM 96*, S. 51–56, St. Petersburg, Oktober.

Hanrieder, Gerhard. 1996. *Inkrementelles Parsing gesprochener Sprache mit einer linksassoziativen Unifikationsgrammatik*. DisKi, Nr.. 140. St. Augustin: Infix.

Harary, Frank. 1974. *Graphentheorie*. München, Wien: R. Oldenbourg Verlag.

Harbusch, Karin. 1990. Constraining Tree Adjoining Grammars by Unification. In *Proc. of the 13 th COLING*, S. 167–172, Helsinki, Finnland.

Haruno, Masahiko, Yasuharu Den, Yuji Mastumoto, und Makato Nagao. 1993. Bidirectional Chart Generation of Natural Language Texts. In *Proc. of AAAI-93*, S. 350–356.

Hauenschild, Christa. 1985. KIT/NASEV oder die Problematik des Transfers bei der Maschinellen Übersetzung. KIT Report 29, Technische Universität Berlin, Berlin, November.

Hauenschild, Christa und Birte Prahl. 1994. Konzept Translationsprobleme - Translationsstrategien. Technical report, Univ. Hildesheim.

Hauenstein, Andreas. 1996. *Aussprachewörterbücher zur automatischen Spracherkennung.* DISKI Dissertationen zur Künstlichen Intelligenz, Nr.. 133. St. Augustin: infix.

Hauenstein, Andreas und Hans Weber. 1994. An Investigation of Tightly Coupled Speech Language Interfaces Using an Unification Grammar. In *Proceedings of the Workshop on Integration of Natural Language and Speech Processing at AAAI '94*, S. 42–50, Seattle, WA.

Hayes-Roth, Barbara. 1985. A Blackboard Architecture for Control. *Artificial Intelligence*, 26:251–321.

Hillis, W.D. 1985. *The Connection Machine*. Cambridge, Mass: MIT Press.

Hoare, Charles A. Richard. 1978. Communicating Sequential Processes. *Communications of the ACM*, 21(8):666–677, August.

Hopcroft, J. und J. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley.

Horacek, Helmut. 1993. Sprachgenerierung: Planungsverfahren und Architekturmodelle. *Künstliche Intelligenz*, 7(2):8–13.

Huebener, Kai, Uwe Jost, und Henrik Heine. 1996. Speech Recognition for Spontaneously Spoken German Dialogs. In *ICSLP96*, Philadelphia.

Hutchins, John. 1994. Research Methods and System Designs in Machine Translation. In *Machine Translation: Ten Years On*, S. 5–1 – 5–16, Cranfield, UK, November.

Hutchins, W. John. 1986. *Machine Translation. Past, Present and Future*. New York: Horwood.

Hutchins, W. John und Harold L. Somers. 1992. *An Introduction to Machine Translation*. London: Academic Press.

Jagannathan, V., R. Dodhiawala, und L. Baum (eds.). 1989. *Blackboard Architectures and Applications*. Boston, Ma.: Academic Press.

Jekat, Susanne, Alexandra Klein, Elisabeth Maier, Ilona Maleck, Marion Mast, und Joachim Quantz. 1995. Dialogue Acts in Verbmobil. Verbmobil Report 65, Universität Hamburg, DFKI GmbH, Universität Erlangen, TU Berlin.

Jekat, Susanne J. 1997. Automatic Interpretation of Dialogue Acts. In Christa Hauenschild und Susanne Heizmann, editors, *Machine Translation and Translation Theory. Perspectives of Cooperation*, Text, Translation, Computational Processing (TTCP), number 1. Mouton de Gruyter.

Jekat, Susanne J., Alexandra Klein, Elisabeth Maier, Ilona Maleck, Marion Mast, und J. Joachim Quantz. 1995. Dialogakte in Verbmobil. Verbmobil Technisches Dokument 26, Universität Hamburg.

Joshi, Aravind K. 1985. How much Context-Sensitivity is Necessary for Characterizing Structural Descriptions—Tree Adjoining Grammars. In D. Dowty, L. Karttunen, und A. Zwicky, editors, *Natural Language Processing — Theoretical, Computational and Psychological Perspectives*. Cambridge University Press, New York.

Jost, Uwe. 1997. System- und Modulevaluation. Verbmobil-Memo 125, Universität Hamburg.

Juola, Patrick. 1994. Self-Organizing Machine Translation: Example-Driven Induction of Transfer Functions. Technical Report CU-CS722-94, Univ. of Colorado, Boulder, Co., Mai.

Kaplan, R., K. Netter, J. Wedekind, und A. Zaenen. 1989. Translation by Structural Correspondence. In *Proc. of the 4 th EACL*, Manchester, UK.

Kaplan, Ronald M. 1973. A General Syntactic Processor. In *Natural Language Processing*. Algorithmic Press, Inc., New York, S. 193–241.

Kaplan, Ronald M. und Joan Bresnan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, Ma., S. 173–281.

Kaplan, Ronald M. und John T. Maxwell. 1989. An Overview of Disjunctive Constraint Satisfaction. In *Proc. International Parsing Workshop*, S. 18–27, Pittsburgh, PA. Carnegie Mellon University.

Karlgren, Jussi. 1994. Mumbling — User-Driven Cooperative Interaction. Technical report, SICS, Stockholm, Januar.

Kasper, W., H.-U. Krieger, J. Spilker, und H. Weber. 1996. From Word Hypotheses to Logical Form: An Efficient Interleaved Approach. In *Proc. of KONVENS 96*.

Kasper, Walter und Hans-Ulrich Krieger. 1996. Integration of Prosodic and Grammatical Information in the Analysis of Dialogs. In *KI-96: Advances in Artificial Intelligence. 20th Annual German Conference on Artificial Intelligence*, S. 163–174, Berlin, September. Springer.

Katoh, Naoto und Teruaki Aizawa. 1994. Machine Translation of Sentences with Fixed Expressions. In *Proc. of the 4 th Conference on Applied Natural Language Processing*, S. 28–33, Stuttgart, Germany, Oktober.

Kay, M., J.M. Gawron, und P. Norvig. 1991. *Verbmobil: A Translation System for Face-to-Face Dialog*. CSLI.

Kay, Martin. 1973. The MIND System. In R. Rustin, editor, *Natural Language Processing*. Algorithmic Press, New York, S. 155–188.

Kay, Martin. 1979. Functional Grammar. In C. Chiarelloet et al., editor, *Proc. 5th Annual Meeting of the Berekeley Linguistic Society*, S. 142–158, Berkeley, Ca.

Kay, Martin. 1980. Algorithmic Schemata and Data Structures in Syntactic Processing. Technical Report CSL-80-12, Xerox Palo Alto Research Center, Palo Alto.

Kay, Martin. 1984. Functional Unification Grammar: A Formalism for Machine Translation. In *Proc. of the 10 th COLING*, S. 75–78, Stanford, CA.

Kay, Martin. 1996. Chart Generation. In *Proc. of the 34 nd ACL*, S. 200–204, Santa Cruz, CA, Juni.

Kesseler, Marcus. 1994. Distributed Control in Verbmobil. Verbmobil Report 24, Univ. of Erlangen-Nürnberg, August.

Kesseler, Marcus P. 1990. TransScheme: Entwurf und Implementierung eines verteilten Scheme-Lisp Systems für Transputernetzwerke. Master's thesis, Universität Erlangen-Nürnberg, Oktober.

Kiefer, Bernd und Thomas Fettig. 1993. FEGRAMED: An Interactive Graphics Editor for Feature Structures. DFKI-Report, April.

Kikui, Gen-ichiro. 1992. Feature Structure Based Semantic Head Driven Generation. In *COLING-92: The 15th International Conference on Computational Linguistics*, S. 32–38, Nantes, France.

Kilbury, James. 1985. Chart Parsing and the Earley Algorithm. KIT-Report 24, Projektgruppe Künstliche Intellligenz und Textverstehen.

Kilger, Anne. 1994. Using UTAGS for Incremental and Parallel Generation. *Computational Intelligence*, 10(4):591–603, November.

Kinoshita, Satoshi, John Phillips, und Jun-ichi Tsujii. 1992. Interaction between Structural Changes in Machine Translation. In *COLING-92: The 15th International Conference on Computational Linguistics*, S. 679–685, Nantes, France.

Kitano, H. 1990. ΦDMDIALOG: A Speech-to-Speech Dialouge Translation System. *Machine Translation*, 5.

Kitano, Hiroaki. 1994. *Speech-to-Speech Translation: A Massively Parallel Memory-Based Approach*. Boston: Kluwer Academic Publishers.

Klein, Alexandra, Susanne J. Jekat, und Jan W. Amtrup. 1996. Inkrementelle und erwartungsgesteuerte Verarbeitung beim Maschinellen Dolmetschen. In *Proceedings der zweiten Fachtagung der Gesellschaft für Kognitionswissenschaft*, S. 68–70, Hamburg, Maerz.

Knight, K. 1989. Unification: A Multi-Disciplinary Survey. *ACM Computer Surveys*, 21:98–124.

Knight, Kevin, I. Chander, M. Haines, V Hatzivassiloglou, E. H. Hovy, M. Iida, S. K. Luk, A. Okumura, R. A. Whitney, und K. Yamada. 1994. Integrating Knowledge Sources and Statistics in MT. In *Proceedings of the 1st AMTA Conference*, Columbia, MD.

Konieczny, Lars. 1996. *Human Sentence Processing: A Semantics-Oriented Parsing Approach*. Ph.D. thesis, Albert-Ludwigs-Universität, Freiburg.

König, Esther. 1994. Syntactic-Head-Driven Generation. In *COLING-94: The 15th International Conference on Computational Linguistics*, Kyoto, Japan.

Krieger, Hans-Ulrich. 1995. *TDL—A Type Description Language for Constraint-Based Grammars. Foundations, Implementation, and Applications.* Ph.D. thesis, Universität des Saarlandes, September.

Künzli, Alexander und Barbara Moser-Mercer. 1995. Human Strategies for Translation and Interpretation. In *KI95-Activities: Workshops, Posters, Demos*, S. 304–306, Bielefeld.

Lavie, Lavie, Alex Waibel, Lori Levin, Michael Finke, Donna Gates, Marsal Gavalda, Torsten Zeppenfeld, und Puming Zhan. 1997. JANUS III: Speech-to-Speech Translation in Multiple Languages. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, München.

Lehning, Michael. 1996. Evaluierung von signalnahen Spracherkennungssystemen fuer deutsche Spontansprache. Verbmobil Report 161, TU Braunschweig, www.dfki.uni-sb.de/verbmobil.

Levelt, Willem J.M. 1989. *Speaking: From Intention to Articulation*. Cambridge, MA: MIT Press.

Light, Marc. 1996. CHUMP: Partial Parsing and Underspecified Representations. In *Proceedings of the ECAI-96 Workshop: Corpus-Oriented Semantic Analysis*.

Luckhardt, Heinz-Dirk. 1987. *Der Transfer in der maschinellen Sprachübersetzung*. Sprache und Information. Tübingen: Niemeyer.

Marchetti-Spaccamela, Alberto, Umberto Nanni, und Hans Rohnert. 1992. On-line Graph Algorithms for Incremental Compilation. Technical Report TR-92-056, ICSI.

Marcus, M. 1980. *A Theory of Syntactic Regognition for Natural Language*. Cambridge, MA: MIT Press.

Marslen-Wilson, W.D. 1987. Functional Parallelism in Spoken Word Recognition. *Cognition*, 25:71–102.

Marslen-Wilson, W.D und A. Welsh. 1978. Processing Interactions during Word Recognition in Continuous Speech. *Cognitive Psychology*, 10:29–63.

Marslen-Wilson, William D. und Lorraine K. Tyler. 1980. The Temporal Structure of Spoken Language Understanding. *Cognition*, 8:1–71.

Matsubara, Shiegki und Yasuyoshi Inagaki. 1997a. Incremental Transfer in English-Japanese Machine Translation. *IEICE Transactions on Information and Systems*, 80(11):1122–1129, November.

Matsubara, Shiegki und Yasuyoshi Inagaki. 1997b. Utilizing Extra-Grammatical Phenomena in Incremental English-Japanese Machine Translation. In *Proceedings of the Seventh International Conference on Theoretical and Methodological Issues in Machine Translation, TMI '97*, S. 31–38, Santa Fe, NM, Juli.

Mattern, Friedemann. 1987. Algorithms for Distributed Termination Detection. Technical Report 20/87, SFB 124, Kaiserslautern.

Mayer, Otto. 1986. *Syntaxanalyse*. Reihe Informatik, Nr.. 27. Mannheim: Bibliographisches Institut.

McClelland, J.L. und J.L. Elman. 1986. The TRACE model of speech perception. *Cognitive Psychology*, 18:1–86.

McHugh, James A. 1990. *Algorithmic Graph Theory*. Englewood Cliffs, NJ: Prentice Hall.

Melamed, I. Dan. 1998. Word-to-Word Models of Translational Equivalence. IRCS Technical Report 98-08, UPenn.

Menzel, Wolfgang. 1994. Parsing of Spoken Language under Time Constraints. In T. Cohn, editor, *Proc. of the 11 th ECAI*, S. 560–564.

Menzel, Wolfgang. 1998. Constraint Satisfaction for Robust Parsing of Spoken Language. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(1):77–89.

Milward, David. 1995. Incremental Interpretation of Categorial Grammar. In *Proc. of the 7 th EACL*, Dublin, Ireland.

Mima, Hideki, Hitoshi Iida, und Osamu Furuse. 1998. Simultaneous Interpretation Utilizing Example-Based Incremental Transfer. In *COLING98P*, COLING98L.

Morimoto, T., M. Suzuki, T. Takazewa, F. Yato, S. Sagayama, T. Tashiro, und M. Nagata. 1993. ATR's Speech translation System: ASURA. In *Proc. of Eurospeech 1993*.

Morimoto, Tsuyoshi, Masami Suzuki, Tosiyuki Takezawa, Genichiro Kikui, Masaaki Nagata, und Mutsuko Tomokiyo. 1992. A Spoken Language Translation System: SL-TRANS2. In *COLING-92: The 15th International Conference on Computational Linguistics*, S. 1048–1052, Nantes, France.

Mouaddib, Abdel-illah und Shlomo Zilberstein. 1995. Knowledge-Based Anytime Computation. In *Proc. of the 14 th IJCAI*, S. 775–781, Montreal, Canada.

Nagao, M. und J. Tsujii. 1986. The Transfer Phase of the Mu Machine Translation System. In *Proc. of the 11 th COLING*, S. 97–103, Bonn, FRG.

Nederhof, Mark-Jan und Giorgio Satta. 1994. An Extended Theory of Head-Driven Parsing. In *Proc. of the 32 nd ACL*, Las Cruces, NM.

Niemann, Heinrich, Elmar Nöth, Andreas Kiessling, Ralf Kompe, und Anton Batliner. 1997. Prosodic Processing and its Use in Verbmobil. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*.

Niessen, S., S. Vogel, H. Ney, und C Tillmann. 1998. A DP based Search Algorithm for Statistical Machine Translation. In *COLING98P*, COLING98L.

Nirenburg, Sergei. 1987. Knowledge and Choices in Machine Translation. In Sergei Nirenburg, editor, *Machine Translation: Theoretical and Methodological Issues*. Cambridge University Press, S. 1–21.

Nirenburg, Sergei. 1992. *Machine Translation. A Knowledge-based Approach*. San Mateo, Ca.: Kaufmann.

Nirenburg, Sergei (ed.). 1993. *Progress in Machine Translation*. Amsterdam: IOS Press.

Niv, Michael. 1993. *A Computational Model of Syntactic Processing: Ambiguity Resolution from Interpretation*. Ph.D. thesis, Univ. of Pennsylvania.

Noeth, Elmar, Anton Batliner, Andreas Kiessling, Ralf Kompe, und Heinrich Niemann. 1997. Prosodische Information: Begriffsbestimmung und Nutzen für das Sprachverstehen. In Paulus und Wahl, editors, *Mustererkennung 1997*, Informatik Aktuell, Heidelberg. Springer Verlag.

Noord, Gertjan van. 1990. Reversible Unification Based Machine Translation. In *Proc. of the 13 th COLING*, S. 299–304, Helsinki, Finnland.

Oerder, Martin und Hermann Ney. 1993. Word Graphs: An Efficient Interface Between Continuous-Speech Recognition and Language Understanding. In *Proceedings of the 1993 IEEE International Conference on Acoustics, Speech & Signal Processing, ICASSP*, S. II/119–II/122, Minneapolis, MN.

Oi, Kozo, Eiichiro Sumita, Osamu Furuse, Hitoshi Iida, und Tetsuya Higuchi. 1994. Real-Time Spoken Language Translation Using Associative Processors. In *Proc. of the 4 th Conference on Applied Natural Language Processing*, S. 101–106, Stuttgart, Germany.

Ousterhout, John K. 1994. *Tcl and the Tk Toolkit*. Addison Wesley.

Palm, Christine. 1995. *Phraseologie: Eine Einführung*. Tübingen: Gunter Narr Verlag.

Paterson, M.S. und M.N. Wegman. 1978. Linear Unification. *Journal of Computer and System Sciences*, 16:158–167.

Pereira, Fernando C. N und Stuart M. Shieber. 1984. The Semantics of Grammar Formalisms Seen as Computer Languages. In *Proc. of the 10 th COLING*, Stanford, CA.

Peres, L. und B. Rozoy. 1991. On the Evaluation of Distributed Termination Protocols. Rapport de Recherche 656, LRI, Paris.

Pollard, Carl und Ivan A. Sag. 1987. *Information-based Syntax and Semantics. Vol 1: Fundamentals*. Stanford, Ca.: CSLI Lecture Notes 13.

Pollard, Carl und Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago, London: University of Chicago Press.

Poller, Peter. 1994. Incremental parsing with LD/TLP-TAGS. *Computational Intelligence*, 10(4):549–562, November.

Poznański, V., J.L. Beaven, und P. Whitelock. 1995. An Efficient Generation Algorithm for Lexicalist MT. In *Proc. of the 33 nd ACL*, Cambridge, MA, Juni.

Prahl, Birte, Susanne Petzold, Susanne Heizmann, und Christa Hauenschild. 1995. Variable Analysetiefe und Bewertungskriterien in Verbmobil: Translationswissenschaftliche Grundlagen. Verbmobil-Report 54, University of Hildesheim, Hildesheim, Januar.

Pyka, Claudius. 1992a. Management of Hypotheses in an Integrated Speech-Language Architecture. In *Proc. of the 10 th ECAI*, S. 558–560, Vienna, Austria.

Pyka, Claudius. 1992b. Schnittstellendefinition mit ASL-DDL. ASL-TR 42-92/UHH, Univ. Hamburg, Maerz.

Pyka, Claudius. 1992c. Spezifikation einer Komponente. Technical Report ASL-Memo-57-92/UHH, Univ. Hamburg, Hamburg, September.

Ramalingam, G. und T. Reps. 1992. An Incremental Algorithm for a generalization of the Shortest-Path Problem. Technical report, Univ, of Wisconsin – Madison.

Rayner, Manny und David Carter. 1996. Fast Parsing Using Pruning and Grammar Specialization. In *Proc. of the 34 nd ACL*, S. 223–230, Santa Cruz, CA, Juni.

Rayner, Manny und David Carter. 1997. Hybrid Language Processing in the Spoken Language Translator. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, München.

Reinecke, Joerg. 1996. Evaluierung der signalnahen Spracherkennung. Verbmobil Memo 113, TU Braunschweig, Nov.

Reithinger, Norbert. 1992. *Eine parallele Architektur zur Inkrementellen Generierung Multimodaler Dialogbeiträge*. Sankt Augustin: infix.

Robinson, J. A. 1965. A Machine-Oriented Logic based on the Resolution Principle. *J. ACM*, 12:23–41.

Russel, Stuart J. und Shlomo Zilberstein. 1991. Composing Real-Time Systems. In *Proc. of the 12 th IJCAI*, S. 212–217, Sidney, Australia, August.

Sadler, Louisa und Henry S. Thompson. 1991. Structural Non-Correspondence in Translation. In *Proc. of the 5 th EACL*, S. 293–298, Berlin, Germany.

Sampson, G. R. 1983. Context-Free Parsing and the Adequacy of Context-Free Languages. In M. King, editor, *Parsing Natural Language*. Academic Press, London, S. 151–170.

Samuel, Arthur G. 1990. Using Perceptual-Restoration Effects to Explore the Architecture of Perception. In G. Altmann, editor, *Cognitive Models of Speech Processing*. MIT Press, Cambridge, MA, Kapitel 14, S. 295–314.

Sato, S. und M. Nagao. 1990. Towards memory based translation. In *Proc. of the 13 th COLING*, S. 3/247–3/252, Helsinki, Finnland.

Satta, G. und Oliviero Stock. 1989. Formal Properties and Implementation of Bidirectional Charts. In *Proc. International Joint Conference on Artificial Intelligence*, S. 1480–1485, Detroit, Mich.

Satta, Giorgio und Oliviero Stock. 1994. Bidirectional context-free grammar parsing for natural language processing. *Artificial Intelligence*, 69:123–164.

Schöllhammer, Thomas. 1997. Übersetzung von Idiomen in einer Speechumgebung. Unveröff. Studienarbeit, Universität Hamburg.

Schröder, Martin. 1994. *Erwartungsgestützte Analyse medizinischer Befundungstexte. Ein wissensbasiertes Modell zur Sprachverarbeitung*. DISKI, Nr.. 54. St. Augustin: infix.

Schubert, Klaus. 1992. Esperanto as an intermediate language for Machine Translation. In John Newton, editor, *Computers in Translation: A Practical Appraisal*. Routledge, London, S. 78–95.

Seligman, Mark, Christian Boitet, und Boubaker Meddeb Hamrouni. 1998. Transforming Lattices into Non-deterministic Automata with Optional Null Arcs. In *COLING98P*, COLING98L.

Sheil, B. A. 1976. Observations on Context-Free Parsing. *Statistical Methods in Linguistics*, 6:71–109.

Shieber, Stuart M. 1984. The Design of a Computer Language for Linguistic Information. In *Proc. of the 10 th COLING*, S. 362–366, Stanford, CA, Juli.

Shieber, Stuart M. 1985. Evidence against the Context-Freeness of Natural Languages. *Linguistics and Philosophy*, 8:362–366.

Shieber, Stuart M., Gertjan van Noord, Robert C. Moore, und Fernando C.N. Pereira. 1989. A Semantic-Head-Driven Generation Algorithm for Unification-Based Formalisms. In *Proc. of the 27 th ACL*, S. 7–17, Vancouver.

Shieber, Stuart M., Gertjan van Noord, Robert C. Moore, und Fernando C.N. Pereira. 1990. Semantic-Head-Driven Generation. *Computational Linguistics*, 16(1):30–42.

Shillcock, Richard. 1990. Lexical Hypotheses in Continuous Speech. In G. Altmann, editor, *Cognitive Models of Speech Processing*. MIT Press, Cambridge, MA, Kapitel 2, S. 24–49.

Shillcock, Richard und Ellen Gurman Bard. 1993. Modularity and the Processing of Closed-class Words. In Gerry T.M. Altmann und Richard Shillcock, editors, *Cognitive Models of Speech Processing: The Second Sperlonga Meeting*. Lawrence Erlbaum, Hove, UK, Kapitel 9, S. 163–185.

Sobashima, Yasuhiro, Osamu Furuse, Susumu Akamine, Jun Kawai, und hitoshi Iida. 1994. A Bidirectional, Transfer-Driven Machine Translation System for Spoken Dialogues. In *COLING-94: The 15th International Conference on Computational Linguistics*, S. 64–68, Kyoto, Japan.

Somers, Harold L. 1993. Current Research in Machine Translation. *Machine Translation*, 7:231–246.

Sommerville, Ian. 1996. *Software Engineering*. Addison-WEsley.

Spilker, Jörg. 1995. Parallelisierung eines inkrementellen aktiven Chart-Parsers. Verbmobil-Report 105, Universität Erlangen-Nürnberg.

Steel, Sam und Anne de Roeck. 1987. Bidirectional Chart Parsing. In *Proc. of the 1987 AISB Conference*, S. 223–235, Edinburgh.

Steele, G.L. und W.D. Hillis. 1986. Connection Machine Lisp: Fine-Grained Symbolic Processing. In *Proceedings of 1986 Symposium on Lisp and Functional Programming*, S. 279–297.

Steinbiß, V., B.H. Tran, und H. Ney. 1994. Improvements in Beam Search. In *Proc. ICSLP-94*, S. 2143–2146, Yokohama.

Stock, Oliviero. 1989. Parsing with Flexibility, Dynamic Strategies, and Idioms in Mind. *Computational Linguistics*, 15(1):1–18, Maerz.

Stock, Oliviero, Rino Falcone, und Patrizia Insinnamo. 1988. Island Parsing and Bidirectional Charts. In *Proc. of the 12 th COLING*, S. 636–641, Budapest, Hungary, August.

Strom, Volker und G. Widera. 1996. What's in the "pure" Prosody? In *Proc. ICSLP 1996*, Philadelphia.

Tanenhaus, M.K., J.M. Leiman, und M.S. Seidenberg. 1979. Evidence for multiple stages in the processing of ambiguous words in syntactic contexts. *Journal of Verbal Learning and Verbal Behavior*, 18:427–440.

Thompson, Henry S. und Graeme Ritchie. 1984. Implementing Natural Language Parsers. In T. O'Shea und M. Einsenstadt, editors, *Artificial Intelligence — Tools, Techniques, and Application*. Harper and Row, London, S. 245–300.

Tran, B.H., F. Seide, und V. Steinbiss. 1996. A word graph based n-best search in continuous speech recognition. In *ICSLP*.

Tropf, Herbert S. 1994. Spontansprachliche syntaktische Phänomene: Analyse eines Korpus aus der Domäne „Terminabsprache". Technical report, Siemens AG, München, Januar.

Vitter, J. S. und R. A. Simons. 1986. New Classes for Parallel Complexity: A Study of Unification and Other Complete Problems for P. *IEEE Trans. Comp.*, S. C–35.

Vogel, Carl, Ulrike Hahn, und Holly Branigan. 1996. Cross-Serial Dependencies Are Not Hard to Process. In *Proc. of the 16 th COLING*, S. 157–162, Copenhagen, Denmark, August.

Wachsmuth, I. und Y. Cao. 1995. Interactive Graphics Design with situated Agents. In W. Strasser und F. Wahl, editors, *Graphics and Robotics*. Springer, S. 73–85.

Wahlster, Wolfgang. 1993. Translation of Face-to-Face-Dialogs. In *Proc. MT Summit IV*, S. 127–135, Kobe, Japan.

Wahlster, Wolfgang. 1997. Verbmobil: Erkennung, Analyse, Transfer, generierung und Synthese von Spontansprache. Verbmobil-Report 198, DFKI, Saarbrücken.

Waibel, Alex. 1996. Interactive Translation of Conversational Speech. *Computer*, 29(7), Juli.

Waibel, Alex, A.M. Jain, A.E. McNair, H. Saito, A.G. Hauptmann, und J. Tebelskis. 1991. JANUS: A Speech-to-Speech Translation System Using Connectionist and Symbolic Processing Strategies. In *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 1991*.

Wang, Ye-Yi und Alex Waibel. 1995. Connectionist Transfer in Machine Translation. In *Proc. International Conference on Recent Advantages in Natural Language Processing*, S. 37–44, Tzigov Chark, Bulgaria, September.

Warren, David. 1983. An Abstract Prolog Instruction Set. Technical Note 309, SRI International, menlo Park, CA.

Warren, R.M. 1970. Perceptual restoration of missing speech sounds. *Science*, 167:392–393.

Weber, Hans. 1992. Chartparsing in ASL-Nord: Berichte zu den Arbeitspaketen P1 bis P9. Technical Report ASL-TR-28-92/UER, Universität Erlangen-Nürnberg, Erlangen, Dezember.

Weber, Hans. 1995. *LR-inkrementelles, Probabilistisches Chartparsing von Worthypothesengraphen mit Unifikationsgrammatiken: Eine Enge Kopplung von Suche und Analyse*. Ph.D. thesis, Universität Hamburg.

Weber, Hans H., Jan W. Amtrup, und Jörg Spilker. 1997. Innovative Systemarchitekturen zur Inkrementellen Interaktiven Verarbeitung. *Künstliche Intelligenz*, 11(4):26–30, Dezember.

Weber, Volker. forthcoming. *Funktionales Konnektionistisches Unifikationsbasiertes Parsing*. Ph.D. thesis, Univ. Hamburg.

Weisweber, Wilhelm. 1992. Term-Rewriting as a Basis for a Uniform Architecture in Machine Translation. In *COLING-92: The 15th International Conference on Computational Linguistics*, S. 777–783, Nantes, France.

Weisweber, Wilhelm. 1994. The experimental MT system of the project KIT FAST. In *Machine Translation: Ten Years On*, S. 12–1 – 12–19, Cranfield, UK, November.

Weisweber, Wilhelm und Christa Hauenschild. 1990. A Model of Multi-Level Transfer for Machine Translation and Its Partial Realization. KIT-Report 77, Technical University of Berlin.

Whitelock, P. 1992. Shake-And-Bake Translation. In *COLING-92: The 15th International Conference on Computational Linguistics*, S. 784–791, Nantes, France.

Winograd, Terry. 1983. *Language as a Cognitive Process. Volume I: Syntax*. Reading, MA: Addison Wesley.

Wintner, Shuly. 1997. *An Abstract Machine for Unification Grammars*. Ph.D. thesis, Technion - Israel Institute of Technology, Haifa, Israel, Januar.

Wintner, Shuly und Nissim Francez. 1995a. Abstract Machine for Typed Feature Structures. In *Proceedings of the 5th Workshop on Natural Language Understanding and Logic Programming*, Lisbon, Spain.

Wintner, Shuly und Nissim Francez. 1995b. Parsing with Typed Feature Structures. In *Proceedings of the 4^{th} International Workshop on Parsing Technologies (IWPT95)*, S. 273–287, Prague, September. Charles University.

Wirén, Mats. 1988. On Control Strategies and Incrementality in Unification-Based Parsing. Linköping Studies in Science and Technology, Thesis No. 140. Master's thesis, Linköping University.

Wirén, Mats. 1992. *Studies in Incremental Natural-Language Analysis*. Ph.D. thesis, Linköping University, Linköping, Sweden.

Woodland, P.C., C.J. Leggetter, J.J. Odell, V. Valtchev, und S.J. Young. 1995. The 1994 HTK large vocabulary speech recognition system. In *ICASSP95*.

Woods, W. A. 1973. An Experimental Parsing System for Transition Network Grammars. In Randall Rustin, editor, *Natural Language Processing*. Algorithmic Press, New York.

Worm, Karsten und C.J. Rupp. 1998. Towards Robust Understanding of Speech by Combination of Partial Analyses. In *Proc. of the 13 th ECAI*, Brighton, UK.

Worm, Karsten L. 1998. A Model for Robust Processing of Spontaneous Speech by Integrating Viable Fragments. In *COLING98P*, COLING98L.

Wu, Dekai. 1995a. Grammarless Extraction of Phrasal Translation Examples From Parallel Texts. In *TMI95P*, TMI95L.

Wu, Dekai. 1995b. Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora. In *Proc. of the 14 th IJCAI*, S. 1328–1335, Montreal, Canada, August.

Ying, H.G. 1996. Multiple Constraints on Processing Ambiguous Sentences: Evidence from Adult L2 Learners. *Language Learning*, 46(4):681–711, Dezember.

Zajac, Rémi. 1990. A Relational Approach to Translation. In *Proc. of the 3 rd Int. Conf. on Theoretical and Methodological Issues of Machine Translation*, Austin, TX.

Zajac, Rémi. 1991. Notes on the Typed Feature System. Technical report, IMS-CL, Stuttgart.

Zajac, Rémi. 1992. Inheritance and Constraint-Based Grammar Formalisms. *Computational Linguistics*, 18(2):159–182.

Zajac, Remi. 1998. Feature Structures, Unification and Finite-State Transducers. In *FSMNLP'98, International Workshop on Finite State Methods in Natural Language Processing*, Ankara, Turkey, Juni.

Zajac, Rémi, Marc Casper, und Nigel Sharples. 1997. An open Distributed Architecture for Reuse and Integration of Heterogeneous NLP Components. In *Proc. of the 5 th Conference on Applied Natural Language Processing*, Washington, D.C.

Zechner, Klaus und Alex Waibel. 1998. Using Chunk Based Partial Parsing of Spontaneous Speech in Unrestricted Domains for Reducing Word Error Rate in Speech Recognition. In *COLING98P*, COLING98L.

Zwitserlood, P. 1989. The Locus of Effects of Sentential-Semantic Context in Spoken-Word Processing. *Cognition*, 32:25–64.

Index