

# Eigenschaftsbasierte Auswahl von Molekülen aus chemischen Fragmenträumen

**Dissertation**

zur Erlangung des akademischen Grades

Dr. rer. nat.

an der Fakultät für Mathematik, Informatik und Naturwissenschaften der  
Universität Hamburg

eingereicht beim Fachbereich für Informatik von

**Juri Pärn**

aus Pskov (Russland)

Januar 2012

Erstgutachter: Prof. Dr. Matthias Rarey  
Zweitgutachter: Prof. Dr. Wolfgang Menzel

Tag der mündlichen Prüfung: 06.07.2012

Meinen Eltern und Leo



Diese Dissertation fasst die Untersuchungen zusammen, die unter Leitung von Herrn Prof. Dr. Matthias Rarey am Zentrum für Bioinformatik, Universität Hamburg in der Zeit von Juli 2004 bis Dezember 2008 durchgeführt wurden.

Die Arbeit ist Teil des BMBF Projekts NovoBench, Fördernummer 313324A, und wurde größtenteils durch dieses finanziert.



## Kurzfassung

Die Anzahl synthetisch zugänglicher und potentiell pharmazeutisch relevanter Moleküle wird auf  $10^{20}$  bis  $10^{24}$  geschätzt. Es ist offensichtlich, dass bei diesen Größenordnungen nicht alle Moleküle auf ihre Wirkung getestet werden können. Untersuchungen oral verfügbarer Arzneimittel haben gezeigt, dass dies in den meisten Fällen auch gar nicht nötig ist. Werden Arzneimittel in einem chemischen Raum abgetragen, der durch einfache physikochemische Eigenschaften aufgespannt wird, häufen sich diese in bestimmten Regionen. Ein vielversprechender Ansatz, bei der Suche nach neuen Wirkstoffen, scheint daher die Suche in diesen Regionen zu sein.

Zur Modellierung von chemischen Räumen werden in dieser Arbeit Fragmenträume verwendet. Chemische Fragmenträume modellieren Moleküle über das Produkt von Molekül-Fragmenten und Regeln zur Verknüpfung dieser. Dadurch ist es möglich, pharmazeutisch sinnvoll und sehr effizient auch sehr große chemische Räume zu verwalten. Da Fragmenträume die Grundlage der in dieser Arbeit vorgestellten Programme sind, war ein Aspekt dieser Arbeit, Fragmenträume zunächst mathematisch formal einzuführen. Darauf aufbauend werden zwei Programme für die Navigation von Fragmenträumen vorgestellt. FRAGVIEW wurde für die interaktive, visuelle Exploration von Fragmenträumen entwickelt. Es stellt Fragmente als zweidimensionale Strukturdiagramme und Regeln in einer Kompatibilitätsmatrix dar. Fragmente können über logische Ausdrücke physikochemischer Eigenschaften ausgewählt werden. Ferner ermöglichen diverse Funktionen, Fragmenträume zu modifizieren und somit bestimmten Anforderungen anzupassen. FRAGENUM wurde für die effiziente Enumerierung aller Moleküle eines Fragmentraums entwickelt, die ein vorgegebenes physikochemisches Profil aufweisen. Da, je nach vorgegebenem Profil, sehr viele Moleküle generiert werden können, musste auf eine effiziente Implementierung geachtet werden. Zwei Punkte bedurften dabei besonderer Beachtung: Benutzung des vorgegebenen Profils für die Auswahl von Fragmenten und die Vermeidung von redundanten Molekülen. Letzteres führte zu der Entwicklung eines Redundanzfilters, welcher auf Baumtopologien operiert.

FRAGENUM konnte in einem Anwendungsszenario 25 von 33 Fragmenträume enumerieren, die aus bekannten Target-Klassen gewonnen wurden. Die Ergebnisse lieferten interessante Einblicke in die Räume und werden in dieser Arbeit vorgestellt.

## Abstract

The number of chemically accessible compounds with potential drug properties is estimated to be in the range of  $10^{20}$  to  $10^{24}$ . Clearly, an exhaustive exploration of these compounds is not possible. Studies have shown that not all of these molecules need to be tested, since known oral drugs share common attributes. The studies demonstrated that drugs cluster in certain regions, if they are placed in a chemical space that is spanned by basic physicochemical properties. Exploring these regions therefore seems to be a promising strategy in the search for new drugs.

To model chemical spaces, this work utilises fragment spaces, which represent molecules as products of molecular fragments, and rules that define how to combine them. This provides an efficient and pharmaceutically sensible way of modelling even very large chemical spaces. This work provides a mathematical foundation for fragment spaces and presents two tools which were developed for the navigation of them. FRAGVIEW facilitates visual and interactive exploration of fragment spaces. It represents fragments as two-dimensional structure diagrams, and rules by means of a compatibility matrix. Fragments can be selected via logical expressions over physicochemical properties and several functions support the customisation of fragments and rules. FRAGENUM enumerates all molecules in a fragment space that obey a physicochemical profile. Efficiency was one of the key objectives since, depending on the profile, a vast number of molecules need to be enumerated. Two aspects were especially crucial: using the physicochemical profile to guide the selection of fragments, and preventing the occurrence of redundant molecules. The latter drove the development of an efficient tree topology redundancy filter.

During application testing, FRAGENUM was able to enumerate 25 of 33 fragment spaces derived from known target classes. The results provided some interesting insights and are presented in this work.







# Vorwort

Die in dieser Arbeit vorgestellten Ergebnisse sind während meiner Zeit als wissenschaftlicher Mitarbeiter am Zentrum für Bioinformatik der Universität Hamburg, im Rahmen des *NovoBench* Projekts entstanden. NovoBench war ein vom BMBF<sup>1</sup> gefördertes Verbundprojekt, unter der Beteiligung von Industrie- und Universitätspartnern<sup>2</sup>. Die Mitarbeit in diesem Projekt ermöglichte mir nicht nur mit akademischen Partnern zusammenzuarbeiten, sondern gewährte mir auch wertvolle Einblicke in die industrielle Sichtweise von Projekt- und Arbeitsabläufen.

Danksagungen waren für mich immer etwas suspekt. Jeder dankt seinem Doktorvater, seinen Kollegen, Freunden und Partner. Meistens in genau dieser Reihenfolge. Wenn es so offensichtlich ist, warum tut es jeder? Nun ja, jetzt weiß ich es. Es gehört soviel mehr dazu eine solche Arbeit zu erstellen, als der Inhalt dieses Schriftstücks es auszudrücken vermag. Daher sollen hier, wenigstens ansatzweise, denen gedankt werden, die diese Arbeit möglich gemacht haben, in welcher Weise auch immer.

Ich möchte mich ausdrücklich bei meinem Doktorvater Matthias Rarey bedanken. Zum einen, weil er mir dies ermöglicht hat und zum anderen, weil er immer mit seinem Wissen und Ratschlägen zur Verfügung stand. Ich weiß von Freunden an anderen Universitäten, dass dies leider keine Selbstverständlichkeit ist. Des Weiteren gilt mein Dank auch allen NovoBench Projektpartnern, insbesondere Dr. Andrea Zaliani. Andrea habe ich nicht nur als Quelle unermesslichen Wissens, sondern auch menschlich sehr zu schätzen gelernt.

Ich möchte mich auch bei meinen Kollegen am Zentrum für Bioinformatik, der Universität Hamburg bedanken. Besonders möchte ich hier erwähnen: Axel Griewel, Gordon Gremme, Ingo Reulecke, Ingo Schellhammer, Jochen Schlosser, Jörg Degen, Patrick Maass und Tobias Lippert. Auch möchte ich mich bei meinen langjährigen Freunden Birsen, Margaritha und Per für alle *extrakurrikulären* Aktivitäten bedanken. Klemens Pilcher sei, unter anderem, für das Korrekturlesen gedankt.

Zum Schluß möchte ich mich bei meiner Freundin Franziska bedanken. Mit ihr ist das Leben einfach schöner und frustrierende Schreibtage sind gar nicht mehr so schlimm, wenn ich mit ihr zusammen bin.

---

<sup>1</sup>Fördernummer 313324A.

<sup>2</sup>4SC AG, ALTANA Pharma, BioSolveIT, Computer-Chemie-Centrum der Universität Erlangen-Nürnberg, Molecular Networks GmbH, Lilly Research Laboratories und das Zentrum für Bioinformatik, Universität Hamburg.



# Inhaltsverzeichnis

Glossar	xv
Akronyme	xvii
Target-Klassen Akronyme	xix
Abbildungsverzeichnis	xxi
Tabellenverzeichnis	xxiii
<b>1 Einleitung</b>	<b>1</b>
<b>2 Wirkstoffentwurf</b>	<b>5</b>
2.1 Wirkstoffraum . . . . .	7
<b>3 Bestehende Ansätze</b>	<b>11</b>
3.1 Navigieren von Fragmenträumen . . . . .	11
3.2 Enumerierung von Molekülen . . . . .	16
3.3 Enumerierung von Bäumen . . . . .	19
<b>4 Modellierung und theoretische Grundlagen</b>	<b>21</b>
4.1 Organische Moleküle . . . . .	21
4.2 Chemische Fragmenträume . . . . .	25
4.2.1 Fragment . . . . .	25
4.2.2 Regeln . . . . .	25
4.2.3 Fragmentraum Generierung . . . . .	27
4.3 Atome, Moleküle und Fragmente als formale Strukturen . . . . .	31
4.4 Fragmentraum . . . . .	32
4.4.1 Fragmentbaum . . . . .	34
4.4.2 Initialisierung von Fragmenten und Molekülen . . . . .	38
<b>5 Navigieren von Fragmenträumen</b>	<b>41</b>
5.1 Visuelle Navigation . . . . .	41
5.2 Enumeration von Molekülen . . . . .	49
5.2.1 Vermeidung von redundanten Fragmentbäumen . . . . .	52
5.2.2 Eigenschaftsbasierte Enumeration . . . . .	57

<b>6</b>	<b>Anwendungsszenario</b>	<b>65</b>
6.1	Target-Klassen und Generierung zugehöriger Fragmenträume . . . . .	65
6.2	Ergebnisse der Enumeration . . . . .	69
6.2.1	Quantitative Analyse . . . . .	69
6.2.2	Ähnlichkeitsvergleiche . . . . .	72
<b>7</b>	<b>Zusammenfassung</b>	<b>77</b>
7.1	Ausblick . . . . .	78
7.1.1	Steigerung der Effizienz von FRAGENUM . . . . .	79
7.1.2	Neue Funktionen für FRAGENUM . . . . .	80
<b>A</b>	<b>Fragmentraum Regeln</b>	<b>81</b>
<b>B</b>	<b>Implementierungsdetails</b>	<b>83</b>
B.1	FragView . . . . .	83
B.2	FragEnum . . . . .	86
	<b>Literaturverzeichnis</b>	<b>87</b>

# Glossar

$\mathcal{A}$	Menge der Atome
$\mathcal{B}$	Menge der chemischen Bindungen
$\mathcal{E}$	Eigenschaftsmenge
$FR$	Fragmentraum
$FB$	Fragmentbaum
$\mathcal{F}$	Fragmentmenge
$\mathcal{FB}$	Fragmentbaum Menge
$\mathcal{I}$	Indexmenge
$\mathcal{C}$	Kompatibilitätsklasse
$\mathcal{K}$	Kompatibilitätstupel
$\vec{K}$	Kompatibilitätsmenge
$\mathcal{L}$	Menge der Link-Atome
$\mathcal{A}_{\mathcal{L}}$	Menge der Atome und Linkatome: $\mathcal{A} \cup \mathcal{L}$
$\mathcal{M}$	Molekülmenge
$\mathcal{R}$	Menge der Verknüpfungsregeln für Fragemente
$\mathcal{R}_{FB}$	Tupel der Verknüpfungsregeln für Fragmentebäume





# Akronyme

2D	Zwei Dimensionen, zweidimensional
3D	Drei Dimensionen, dreidimensional
ADMET	<i>Absorption, distribution, metabolism, excretion and toxicity.</i> Zusammenfassung wichtiger Eigenschaften, die beim Wirkstoffentwurf, neben der Protein-Ligand-Wechselwirkung, berücksichtigt werden müssen.
CATS	<i>Chemical advanced template search</i> Moleküldeskriptor, basierend auf Atom-Typen und deren topologischen Entfernungen.
COLIBREE	<i>Combinatorial library breeding</i> Programm zum Verwalten von virtuellen kombinatorischen Bibliotheken.
CPU	<i>Central Processing Unit</i>
Da	Dalton
EA	Evolutionärer Algorithmus
FTREE	<i>Feature Tree</i> Programm zur paarweisen Berechnung von Molekülähnlichkeiten.
FTREE-FS	<i>Feature Trees Fragment Space Search</i>
HTS	<i>High-throughput screening</i>
PC	<i>Personal Computer</i>
PSO	<i>Particle swarm optimization</i>
RDB	Referenzdatenbank
RECAP	<i>Retrosynthetic combinatorial analysis procedure</i> Regelmenge und Vorgehensweise zur Erzeugung von virtuellen Molekülfragmenten.

SMARTS	Spracherweiterung von SMILES zur Beschreibung von Molekülmustern.
SMILES	<i>Simplified molecular-input line-entry specification</i> Lineare, relativ leicht interpretierbare Sprache zur Repräsentation von Molekülen. Beispiel: <chem>CC(=O)O</chem> (Essigsäure).
TOPAS	<i>Topology assigning system</i> Ein Programm, das mithilfe von evolutionären Algorithmen eine Molekül-Ähnlichkeitssuche in Fragmenträumen durchführt.
VKB	Virtuelle kombinatorische Bibliothek
WDI	<i>World drug index</i> Umfangreicher Index von Wirkstoffen.

# Target-Klassen Akronyme

ACE	Angiotensin-konvertierendes Enzym
AChE	Acetylcholinesterase
ADA	Adenosin Desaminase
ALR2	Aldose Reduktase
AmpC	AmpC $\beta$ -Lactamase
AR	Androgenrezeptor
CDK2	Cyclin-abhängige Kinase 2
COMT	Catechol <i>O</i> -Methyltransferase
COX-1	Cyclooxygenase-1
COX-2	Cyclooxygenase-2
DHFR	Dihydrofolat Reduktase
EGFr	Epidermal growth factor receptor
ER	Estrogen Rezeptor
FGFr	Fibroblast growth factor receptor
FXa	Faktor Xa
GART	Glycinamid-ribonucleotid Transformylase
GPB	Glycogen-Phosphorylase $\beta$
GR	Glukokortikoid-Rezeptor
HMGa	High mobility group Proteine
HIVPR	Human immunodeficiency virus protease
HIVRT	Human immunodeficiency virus reverse Transkriptionase
HSP90	Heat shock protein 90
InhA	Enoyl-acyl carrier protein reductase
MR	Mineralokortikoidrezeptor
NA	Neuraminidase
P38 MAP	p38 Mitogen-aktivierte Proteinkinase
PARP	Poly adenosine diphosphate ribose polymerase
PDE5	Phosphodiesterase-5
PDGFr $\beta$	Platelet derived growth factor receptor <i>beta</i>
PNP	Purin-Nukleosid-Phosphorylase
PPAR $\gamma$	Peroxisom Proliferator-aktivierter Rezeptor $\gamma$
PR	Progesteron Rezeptor
RXR $\alpha$	Retinoid X Rezeptor $\alpha$
SAHH	S-Adenosylhomocystein Hydrolase
SRC	Tyrosinkinase SRC

TK	Thymidinkinase
VEGFr2	Vascular endothelial growth factor receptor 2

# Abbildungsverzeichnis

2.1	Typische erste Schritte bei der Suche nach einem neuen Wirkstoff. . .	6
2.2	Chemischer Raum mit <i>Absorption, distribution, metabolism, excretion and toxicity</i> (ADMET) und Arzneimittel Unterräumen. . . . .	8
2.3	$A \log P$ und molekulare Masse 1.791 bekannter oral verfügbarer Wirkstoffe. . . . .	9
2.4	ADMET-Score Verteilung oraler Wirkstoffe und einer Referenz Molekülmenge. . . . .	10
4.1	Freiheitsgrade in einem Molekül. . . . .	24
4.2	Modellierung chemischer Reaktionen durch Fragmente und Regeln. .	26
4.3	Generierung von Fragmenten. . . . .	29
4.4	<i>Feature Trees Fragment Space Search</i> (FTREE-FS) Fragmentraum . .	30
4.5	Fragmentbaum und korrespondierendes Molekül. . . . .	37
5.1	Kriterien für darzustellende Fragmente. . . . .	42
5.2	Fragmentraum Visualisierung. . . . .	44
5.3	Regeln eines Fragmentraums. . . . .	45
5.4	Ausrichten von Fragmenten. . . . .	46
5.5	Selektieren, Verknüpfen und Terminieren von Fragmenten. . . . .	48
5.6	Redundante Enumeration von Fragmentbäumen. . . . .	51
5.7	Räumliche Darstellung der Hyperebenen eines k-d Baums. . . . .	59
5.8	k-d Baum zur Abbildung 5.7. . . . .	59
5.9	Beispieleingabe für FRAGENUM . . . . .	61
6.1	Ausgewählte Beispiele für Inhibitor und ein enumeriertes Molekül. . .	76
B.1	Kompositum Klassendiagramm. . . . .	83
B.2	Klassenstruktur von FRAGVIEW. . . . .	84
B.3	Beispiel einer Besucher-Klasse in FRAGVIEW . . . . .	85
B.4	Beispiel einer Implementierung eines Besuchers in FRAGVIEW . . . .	85



# Tabellenverzeichnis

4.1	Initialisierungsstufen mit verbundenen Berechnungen. . . . .	40
5.1	Physikochemische Eigenschaften verwendbar in FRAGVIEW. . . . .	43
5.2	FRAGVIEW Funktionen für Fragmente. . . . .	45
5.3	FRAGVIEW Funktionen für Tabellen. . . . .	47
5.4	Globale FRAGVIEW Funktionen. . . . .	47
5.5	Beispiel für die Effektivität des Baumtopologie Redundanztests. . . .	56
6.1	Quantitative Ergebnisse und Eigenschaften generierter Fragmenträume.	67
6.2	Target-Klassen und zugehörige physikochemische Intervalle . . . . .	68
6.3	Quantitative Werte der Enumerierung. . . . .	70
6.4	Feature Tree Ähnlichkeitsverteilung für verschiedene Target-Klassen .	73
6.5	Repräsentanten drei unterschiedlicher Gruppen. . . . .	74





# Liste der Algorithmen

5.1	Funktion BasisEnumeration(FragmentRaum: $FR$ ). Algorithmus zum Enumerieren aller Fragmentbäume eines Fragmentraums. . . . .	49
5.2	Funktion BasisEnumeriereRekursive(FragmentBaum: $FB$ , KompatibilitätsTupel: $\vec{K}$ , FragmentRaum: $FR = (\mathcal{F}, \mathcal{R})$ ). Rekursive Funktion zur Enumeration aller Fragmentbäume eines Fragmentraums. . . . .	50
5.3	Algorithmus zur Überführung eines Fragmentbaums $FB$ in einen Graph-Isomorphen Fragmentbaum, der Bedingungen aus Satz 5.2.2 erfüllt. . .	53
5.4	Funktion IstReduziert(Fragmentbaum: $FB$ ). Funktion zum Testen, ob ein Fragmentbaum reduziert ist, unter Verwendung des Satzes 5.2.2. . . . .	55
5.5	Funktion GeneriereKDBaeume(Fragmentbaum: $FR$ , Liste: $L$ ). Funktion zum Generieren von link spezifischen k-d Bäumen über einer Menge von Eigenschaften. . . . .	60
5.6	Funktion Enumeriere(Fragmentraum: $FR$ , Liste: $L$ ). Funktion zum eigenschaftsbasierten Enumerieren aller Fragmentbäume eines Fragmentraums. . . . .	62
5.7	Funktion EnumeriereRekursive . . . . .	63



# 1 Einleitung

*Chemische Fragmenträume* stellen eine attraktive und effiziente Methode dar, chemische Räume kombinatorisch zu modellieren[1, 2]. Fragmenträume bestehen aus einer Menge von Molekül-Fragmenten und Regeln, die festlegen wie die Fragmente verknüpft werden können. Moleküle müssen durch diese Modellierung nicht explizit gespeichert werden, sie sind vielmehr indirekt durch das Produkt aus Fragmenten und Regeln gegeben. Ein Fragmentraum enthält somit alle Moleküle, die durch die Kombination von Fragmenten, gemäß der Regeln, generiert werden können. Das erlaubt es, eine sehr große, potentiell unendliche, Menge von Molekülen sehr effizient zu verwalten. Ein weiterer Vorteil von Fragmenträumen ist ihre kombinatorische Struktur, die es ermöglicht, diese Räume systematisch zu verarbeiten. Auch pharmazeutisch und chemisch sind Fragmenträume sehr attraktiv. Fragmente können so modelliert werden, dass sie pharmazeutisch relevante Motive beinhalten. Die Regeln können bekannte chemische Reaktion nachbilden und dadurch ein Problem vieler computergestützter Verfahren im Wirkstoffentwurf mindern: die synthetische Zugänglichkeit von computergenerierten Molekülen.

Wird die geschätzte Zahl organischer Moleküle betrachtet, die potentiell als Wirkstoffe in Frage kommen,  $10^{60}$  bis  $10^{100}$ [3, 4, 5], wird ersichtlich, warum eine effiziente Verwaltung von Molekülen notwendig ist. Diese Zahlen entsprechen aber lediglich theoretischen Betrachtungen. Werden heute verfügbare Techniken zur Synthese von Molekülen zugrunde gelegt, wird die Anzahl von *Drug-like* Molekülen auf  $10^{20}$  bis  $10^{24}$  geschätzt[5]. Diese Zahl ist immer noch sehr beeindruckend und für alle praktischen Belange kann sie faktisch als unendlich angesehen werden<sup>1</sup>[6]. Diese sehr große Zahl ist Segen und Fluch zugleich. Zum einen verspricht sie, immer ein Molekül zu einer Proteinbindetasche zu finden, um dadurch eine pharmazeutische Wirkung herbeizuführen (siehe Kapitel 2), zum anderen ist es aber unmöglich, erschöpfend alle Moleküle zu testen. Untersuchungen haben aber gezeigt, dass es nicht nötig ist, alle diese Moleküle zu betrachten[7].

Eine Möglichkeit Moleküle systematisch zu untersuchen, besteht darin diese in einem *chemischen Raum* anzuordnen. Ein chemischer Raum wird von einem Koordinatensystem aufgespannt, dessen Achsen beliebige Molekül-Deskriptoren bilden können[8]. Werden sehr grundlegende physikochemische Eigenschaften, wie zum Beispiel molekulares Gewicht und der Lipophilie-Verteilungskoeffizient (siehe Kapitel 4.1) als Dimensionen gewählt und bekannte, oral verfügbare Arzneimittel gemäß dieser Dimensionen abgetragen, kann beobachtet werden, dass sich diese in bestimmten Re-

---

<sup>1</sup>Würden jede Sekunde eine Million Moleküle synthetisiert werden, würden für  $10^{20}$  Moleküle 3.170.979 Jahre benötigt werden!

gionen des Raums anhäufen (siehe Abbildung 2.2). Es scheint daher vielversprechend in diesen Regionen des chemischen Raums nach neuen Wirkstoffen zu suchen. Dies spiegelt sich auch in der Aussage von Sir James Black, einem erfolgreichen Entdecker von Arzneimitteln, wider: “The most fruitful basis of the discovery of a new drug is to start with an old drug“[9].

### Zielsetzung der Arbeit

Die vorliegende Arbeit beschäftigt sich mit chemischen Fragmenträumen und Programmen, um in diesen zu navigieren. Aufgrund ihrer Attraktivität waren chemische Fragmenträume schon Grundlage vieler Programme im Bereich des Wirkstoffentwurfs[10, 11, 1]. Diese Programme fokussierten sich auf die Anwendung von Fragmenträumen und nicht so sehr auf ihre formalen Grundlagen. Ein Punkt dieser Arbeit wird daher sein, chemische Fragmenträume zunächst formal einzuführen. Die Einführung sollte sich auf mathematische Strukturen fokussieren, da die notwendigen chemischen Grundlagen schon gelegt wurden[12].

Des Weiteren sollten Programme entwickelt werden, die es Benutzern ermöglichen, in Fragmenträumen zu navigieren. Als erstes sollte ein Programm implementiert werden, welches es erlaubt, Fragmenträume zu visualisieren. Fragmente sollten als Strukturdiagramme und Regeln in einer Kompatibilitätsmatrix dargestellt werden. Dieses Werkzeug sollte auch die Möglichkeit bieten, Fragmenträume Bedürfnissen anzupassen. Es sollte also möglich sein, Fragmente über logische Ausdrücke zu selektieren, zu löschen, zu modifizieren und neue Fragmente in einen bestehenden Raum aufzunehmen. Neben den Fragmenten sollten auch Regeln eines Raums modifiziert werden können. Wie oben dargelegt, scheinen sich Wirkstoffe im physikochemischen Raum in bestimmten Bereichen zu häufen. Aus diesem Grund sollte ein Programm entwickelt werden, welches alle Moleküle eines Fragmentraums generiert, die ein bestimmtes physikochemisches Profil besitzen. Da, abhängig vom Profil, sehr viele Moleküle generiert werden können, musste diese Aufzählung sehr effizient und performant realisiert werden.

### Aufbau der Dissertation

Zunächst werden im Kapitel 2 wichtige Prinzipien des rationalen Wirkstoffentwurfs präsentiert und der Wirkstoffraum vorgestellt. Kapitel 3, *Bestehende Ansätze*, widmet sich dann Ansätzen, die zu den Methoden und Programmen ähnlich sind, die in dieser Arbeit vorgestellt werden. Die Aufzählung erhebt nicht den Anspruch auf Vollständigkeit. Es wird vielmehr versucht, repräsentative Ansätze darzustellen, die ähnlich zu denen in dieser Arbeit sind. Das nächste Kapitel, *Modellierung und theoretische Grundlagen*, führt zunächst in die biochemischen Grundlagen ein, die zum Verstehen der Arbeit notwendig sind. Auf diesen Grundlagen werden Fragmenträume, ein zentrales Konstrukt dieser Arbeit, vorgestellt. Von diesen werden dann formale Strukturen abstrahiert, die im nächsten Kapitel verwendet werden. Das 5. Kapitel, *Navigieren von Fragmenträumen*, stellt zwei Programme vor, die entwickelt

---

wurden, um in chemischen Fragmenträumen zu navigieren. Im Genaueren sind das FRAGVIEW, ein Programm zur visuellen Navigation von Fragmenträumen, und FRAGENUM, ein Programm zum eigenschaftsbasierten Enumerieren von Fragmenträumen. FRAGENUM wurde verwendet, um Moleküle verschiedener Target-Klassen zu enumerieren und die Ergebnisse wurden mit bekannten Wirkstoffen verglichen. Die Ergebnisse dieser Untersuchung sind in Kapitel 6, *Anwendungsszenario*, wiedergegeben. Kapitel 7 fasst die Ergebnisse dieser Arbeit zusammen und gibt einen Ausblick auf Erweiterungsmöglichkeiten der vorgestellten Programme. Im Anhang A finden sich die verwendeten Regeln für die Untersuchungen im 6. Kapitel und im Anhang B Implementierungsdetails der vorgestellten Programme.



## 2 Wirkstoffentwurf

Praktisch alle Funktionen eines Organismus werden durch Proteine gesteuert[13, 14, 15]. Wichtige Proteinklassen sind Enzyme, Rezeptoren, Ionenkanäle und Transporter. *Enzyme* katalysieren wichtige Reaktionen und sind somit für den Stoffwechsel und die Regulation von physiologischen Prozessen unabdingbar. *Rezeptoren* vermitteln unter anderem den Informationsaustausch zwischen Zellen und regulieren Genabschnitte. *Ionenkanäle* sind schnelle Schalter und steuern, neben vielen anderen Funktionen, Kontraktionen von Muskeln, wie zum Beispiel des Herzens. *Transporter* können Stoffe aktiv in eine Zelle befördern und sorgen somit zum Beispiel dafür, dass eine Zelle mit Aminosäuren versorgt wird. Es sollte daher nicht weiter verwundern, dass durch eine Beeinflussung dieser Proteine eine Wirkung auf einen Organismus ausgeübt werden kann. Genau bei dieser Beeinflussung setzen Arzneimittel an. Praktisch alle Arzneimittel sind kleine Moleküle, die sich an Proteine binden können und dadurch eine Wirkung herbeirufen. Wirkstoffe wirken meist als *Inhibitoren* von Enzymen oder als *Agonisten* beziehungsweise *Antagonisten* von Rezeptoren. Je nachdem an welche Proteinklasse ein Ligand bindet, verhindert er die Bindung des natürlichen Liganden oder es wird eine strukturelle Änderung des Proteins induziert, so dass das Zielprotein nicht mehr seine Funktion ausüben kann oder nicht mehr mit anderen Proteinen interagieren kann.

Ende des 19. und Anfang des 20. Jahrhunderts wurden zwei wichtige Prinzipien des Wirkstoffentwurfs entdeckt. Emil Fischer formulierte 1894 das Schlüssel und Schloss Prinzip[16]. Es besagt, dass die dreidimensionale (3D) Struktur eines Liganden und ein Teil eines Proteins komplementär sein müssen, damit der Ligand binden kann. 1909 erkannte Paul Ehrlich: *corpora non agunt nisi fixata*, Körper wirken nicht, wenn sie nicht gebunden sind. Diese beiden Erkenntnisse zusammengefasst besagen, dass eine Wirkung nur erreicht wird, wenn sich ein Wirkstoff an ein Protein binden kann und, dass eine Bindung nur stattfindet, wenn die 3D Struktur des Wirkstoffs komplementär zu einem Teil der Proteinoberfläche ist. Durch diese Erkenntnisse war es zum ersten Mal möglich *rational* Wirkstoffe zu entwerfen. Bevor es aber möglich war die 3D Struktur von Proteinen aufzuklären, konnte dieses Wissen zunächst nur benutzt werden, um bekannte oder durch Zufall entdeckte Wirkstoffe, wie zum Beispiel Penicillin[17], zu verstehen und zu verbessern. Mit der Entwicklung der Kryo-Elektronenmikroskopie[18], hochauflösender NMR-Spektroskopie[19] und hauptsächlich der Röntgenstrukturanalyse[20] konnte ab Mitte des 20. Jahrhunderts die 3D Struktur von vielen Proteinen aufgelöst werden. Anfangs konnten nur einfache und symmetrische Strukturen aufgelöst werden, aber mit technischen Fortschritten ist es nun möglich, für fast beliebige Proteine die 3D Positionen ihrer Atome mit einer sehr hohen Auflösung zu bestimmen. Die *Protein Data Bank* (PDB)[21], mit ihrer

immer rasanter wachsenden Zahl von Proteinen, zeigt eindrucksvoll die Wichtigkeit und immer besser werdenden Verfahren zur Strukturauflösung von Proteinen.

Eine weitere wichtige Entwicklung im Rahmen des Wirkstoffentwurfs war die Einführung von automatischen Testsystemen, welche sehr viele Moleküle relativ schnell auf ihre Wirkung testen konnten (engl. *High-Throughput Screening* (HTS)). Die Hoffnung war, dass durch das automatische Testen von sehr vielen Molekülen, nicht selten Hunderttausend und mehr[6], sich einige Treffer (englisch *Hits*) ergeben würden, die als Ausgangsbasis für einen neuen Wirkstoff dienen können. Die gleichzeitige Entwicklung der kombinatorischen Chemie[22], die auf einfache, automatische und systematische Weise eine sehr große Menge von Molekülen produzieren kann, weckte die Hoffnung, Wirkstoffe fast automatisch entdecken zu können[15]. Ein entgültiges Urteil über HTS steht noch aus, da diese Methode noch nicht lang genug im Einsatz ist[6]. Aufgrund hoher Kosten und geringer Trefferraten lässt sich aber jetzt schon sagen, dass HTS nicht das Ende des rationalen Wirkstoffentwurfs eingeläutet hat[6].

Die ersten Schritte in einem typischen Prozess zum Finden eines neuen Wirkstoffs sind in Abbildung 2.1 dargestellt.

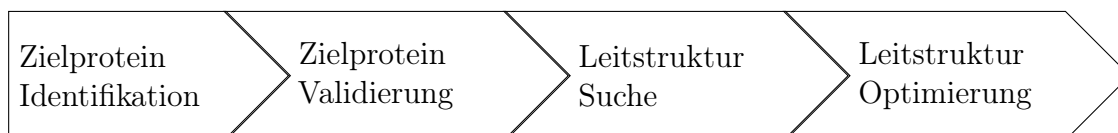


Abbildung 2.1: Typische erste Schritte bei der Suche nach einem neuen Wirkstoff.

In der ersten Phase wird nach Zielproteinen gesucht, mithilfe derer sich das Krankheitsbild beeinflussen lässt. In der zweiten Stufe wird die Rolle des gefundenen Proteins genauer untersucht und getestet, ob dessen Beeinflussung auch zu einer gewünschten Änderung führt. In der dritten Phase wird, häufig mithilfe von HTS, nach Treffern gesucht, von denen die aussichtsreichsten durch genaueres Testen validiert und als *Leitstruktur* weiter verwendet werden. Diese Leitstrukturen werden dann in einem nächsten Schritt optimiert. Dabei wird neben der Erhöhung der Bindungsaffinität, insbesondere auf ADMET (siehe Kapitel 2.1) Eigenschaften geachtet.

Für die zwei letzten beiden Schritte, Leitstruktur Suche und Optimierung, wurden auch computergestützte Verfahren entwickelt. Die Leitstruktursuche wird von *Docking* Programmen unterstützt. Diese Programme versuchen kleine Moleküle in eine vorgegebene Proteinbindetasche einzupassen und die Bindungsenthalpie abzuschätzen[23, 24, 25, 26].

Unabhängig davon, ob *in vitro* oder *in silico* Verfahren zum Screenen und Docken verwendet werden, stellt sich immer die Frage, welche Moleküle getestet werden sollen. Der nächste Abschnitt widmet sich dem chemischen Raum und untersucht genauer, ob es Bereiche gibt, die vielversprechender für Wirkstoffe sind als andere.



## 2.1 Wirkstoffraum

Eine erfolgreiche Methode einen neuen Wirkstoff zu suchen, scheint darin zu bestehen, bei einem bekannten Wirkstoff zu starten. Wird die Arbeit von Paul Jansen, dem wahrscheinlich erfolgreichsten Wirkstoffentdecker, betrachtet, so fällt auf, dass er immer mit sehr ähnlichen Molekülen gearbeitet hat[6].

Systematische Untersuchungen oral verfügbarer Arzneimittel haben ergeben, dass diese viele gemeinsame Eigenschaften besitzen[27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43]. All diese Untersuchungen beschränken sich auf relativ einfache Eigenschaften, wie molekulares Gewicht oder  $\log P$  Werte und nicht auf Ähnlichkeit im Sinne der 3D Struktur. Dies scheint auf den ersten Blick etwas verwunderlich, da nach dem Schlüssel und Schloss Prinzip nur die komplementäre, dreidimensionale Struktur über die Wirkung entscheidet. Es muss aber bedacht werden, dass ein Wirkstoff erstmal zu einem Protein gelangen muss, bevor er seine Wirkung entfalten kann. Auch sollte ein Wirkstoff möglichst wenig Nebenwirkungen, insbesondere keine toxischen, aufweisen. Des Weiteren sollte ein eingenommener Wirkstoff nach einer Zeit wieder abgebaut und ausgeschieden werden, um keine dauerhafte Wirkung auszuüben. All diese Eigenschaften sind unter dem Begriff ADMET zusammengefasst und die Ergebnisse der angegebenen Publikationen besagen im Wesentlichen, dass sich ADMET Eigenschaften anhand von einfachen physikochemischen Parametern, im gewissen Rahmen, relativ gut abschätzen lassen. Die wohl bekannteste Untersuchung von Arzneimitteln ist von Lipinski et al., die den Begriff *rule of five* einführte[7]. Die besagt, dass falls ein Arzneimittel mehr als eine der folgenden Eigenschaften verletzt, es höchstwahrscheinlich eine *schlechte* Absorption oder Permeation aufweisen wird<sup>1</sup>:

- Mehr als 5 Wasserstoffdonoren (ausgedrückt als Summe von OH und NH).
- Das molekulare Gewicht ist über 500 Dalton (Da).
- $\log P$  ist größer als 5.
- Mehr als 10 Wasserstoffbrücken (ausgedrückt als Summe von N und O).

Anzumerken sei noch, dass natürlich nicht jedes Molekül, das die obigen Eigenschaften erfüllt, automatisch ein Arzneimittel ist. Vielmehr erfüllen fast alle Arzneimittel mindestens drei der vier Bedingungen.

Eine andere Interpretation der obigen Ergebnisse ist, dass sich praktisch alle oralen Arzneimittel in einer Region häufen, wenn sie in einem Raum plaziert werden, der durch die vier betrachteten Dimensionen der *rule of five* aufgespannt wird.

Abbildung 2.2 zeigt schematisch, wie sich Arzneimittel in einem chemischen Raum verteilen. Wirkstoffe für verschiedene Target-Klassen häufen sich in bestimmten Regionen des chemischen Raums. Aber nur Moleküle, die sich im grünen ADMET Unterraum befinden, kommen als Arzneimittel in Frage.

---

<sup>1</sup>Ausgenommen sind Moleküle, die Substrate für Transporter sind.

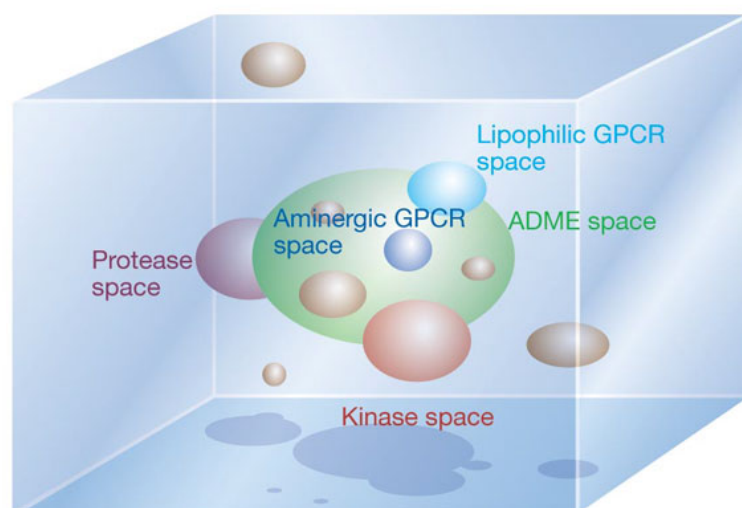


Abbildung 2.2: Chemischer Raum mit ADMET und Arzneimittel Unterräumen.

Der hellblaue Bereich repräsentiert den chemischen Raum, der Subräume wichtiger Target-Klassen enthält. Der grüne Bereich stellt den Raum dar, der wünschenswerte ADMET Eigenschaften besitzt.

Grafik aus [6], mit freundlicher Genehmigung der Nature Publishing Group, Lizenznummer 2827231403210.

Eine Veröffentlichung von Gleeson et al.[44] betrachtete das molekulare Gewicht und den  $\log P$  Wert von 1.791 zugelassenen, oralen Arzneimitteln genauer<sup>2</sup>. In der Veröffentlichung wurde ein ADMET-Score entwickelt, der sich aus einer leicht modifizierten z-Transformation<sup>3</sup> des molekularen Gewichts und des  $A \log P$  Wertes bezüglich oraler Arzneimittel ergibt (siehe Abbildung 2.3). Je kleiner der Score, desto mehr entspricht ein Molekül zugelassenen Arzneimitteln, bezüglich des molekularen Gewichts und  $A \log P$  Wertes. Die Verteilung der untersuchten Moleküle ist in Abbildung 2.3 zu sehen. 56% haben einen Score  $\leq 1$  und fast 80% der Wirkstoffmoleküle haben eine Score von  $\leq 1,5$ , das heißt vereinfacht gesagt, dass sich fast 80% der Wirkstoffe in einem 1,5 Standardabweichungs-Radius vom kombinierten Mittelwert aus molekularem Gewicht und  $A \log P$  befinden.

Dieses Ergebnis zeigt, dass orale Arzneimittel, bezüglich ihres molekularen Gewichts und  $A \log P$  Wertes, relativ homogen sind. Die Frage ist, ob sie sich von anderen Molekülen unterscheiden. Dazu wurde eine Referenzmenge von 201.355 Molekülen aus der ChEMBL Datenbank[45] erstellt. Diese Menge wurde aus Veröffentlichungen extrahiert, die im Rahmen des Wirkstoffentwurfs entstanden sind. Diese Menge

<sup>2</sup>Der  $\log P$  Wert wurde mithilfe von Programmen berechnet und wird in der Veröffentlichung  $A \log P$  genannt.

<sup>3</sup>z-Transformation:  $\frac{X - \text{Mittelwert}}{\text{Standardabweichung}}$

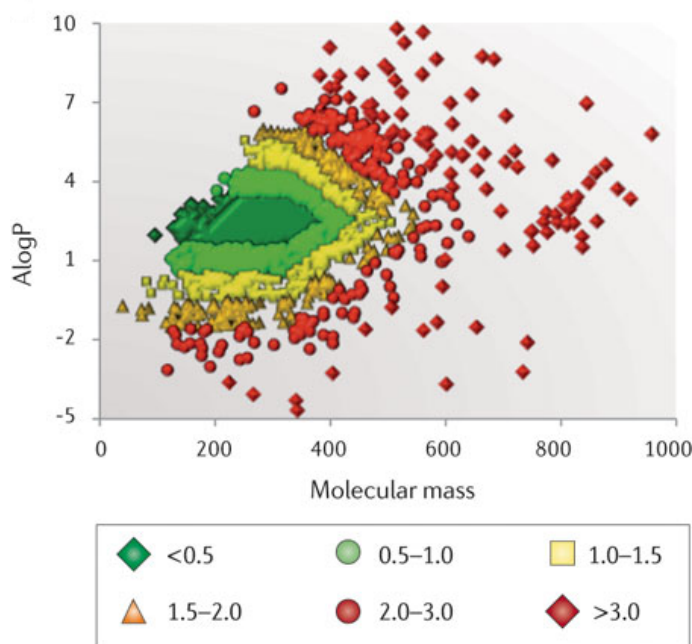


Abbildung 2.3:  $A \log P$  und molekulare Masse 1.791 bekannter oral verfügbarer Wirkstoffe.

Bekannte Wirkstoffe sind bezüglich ihrer molekularen Masse und ihres  $A \log P$  Wertes abgetragen. Die Farbe repräsentiert den *ADMET-Score* eines Wirkstoffs und gibt an, wie weit beide Werte kombiniert vom Mittelwert, dividiert durch die Standardabweichungen abweicht:  $\frac{2.5 - A \log P}{2.0} + \frac{330 - \text{molekulare Masse}}{120}$ . Kleiner ADMET-Score bedeutet eine hohe Konformität zum mittleren molekularem Gewicht und  $A \log P$ . Wie aus dem Graphen abgelesen werden kann, besitzen sehr viele orale Wirkstoffe einen geringen Score.

Grafik aus [44], mit freundlicher Genehmigung der Nature Publishing Group, Lizenznummer 2805820790792.

beschreibt somit Moleküle, die im Rahmen des Wirkstoffentwurfs von Interesse sind, und dürfte den Wirkstoffraum relativ gut repräsentieren. Wird zunächst die Verteilung der ADMET-Scores der Referenzmenge betrachtet (schwarze Balken in Abbildung 2.4), fällt auf, dass diese relativ gleichmäßig alle Score-Bereiche abdecken. Daraus folgt, dass die Referenzmoleküle nicht homogen bezüglich ihres molekularen Gewichts und  $A \log P$  Wertes sind. Wird die Referenzverteilung mit der Verteilung der oralen Arzneimittel verglichen, kann festgestellt werden, dass diese sich eindeutig unterscheiden. Die Verteilung der Arzneimittel ist eindeutig in Richtung geringeren Scores verschoben.

Aus diesen Ergebnissen und oben genannten Publikationen folgt, dass orale Arzneimittel bezüglich ihrer physikochemischen Eigenschaften eine gewisse Homogenität aufweisen. Mit der *rule of five* existieren sogar harte Grenzen für diese. Mit diesen

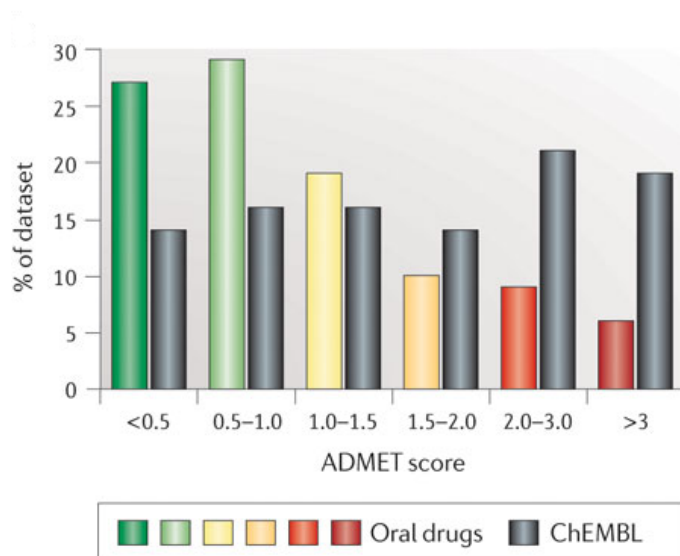


Abbildung 2.4: ADMET-Score Verteilung oraler Wirkstoffe und einer Referenz Molekülmenge.

Die Abbildung zeigt die Verteilung des ADMET-Scores oraler Arzneimittel und einer Menge von Referenzmolekülen. Es zeigt sich auch hier, dass orale Wirkstoffe tendenziell homogener sind als Wirkstoffe im weitesten Sinne.

Grafik aus [44], mit freundlicher Genehmigung der Nature Publishing Group, Lizenznummer 2805820790792.

Grenzen kann ein genau definierter Raum angegeben werden, in dem gesucht werden sollte. Der Raum verkleinert sich sogar noch etwas, da für Leitstrukturen sogar engere Grenzen gefunden wurden und *rule of three*[46] benannt wurden. Die betrachteten physikochemischen Eigenschaften sind dieselben, nur die Werte verkleinern sich auf 3 beziehungsweise 300 bei molekularem Gewicht.

Aus all diesen Betrachtungen folgt, dass es sehr fruchtbar sein könnte, sich viele oder sogar alle Moleküle mit *rule of five*, *rule of three* oder ADMET-Score  $\leq 1$  Eigenschaften anzusehen. Um dies computergestützt zu ermöglichen, muss zunächst ein Weg gefunden werden, chemisch sinnvolle Moleküle zu generieren. Aufbauend darauf müssten dann Programme entwickelt werden, die es erlauben, physikochemisch diese Moleküle zu erkunden. Genau dies ist Gegenstand dieser Arbeit und wird in den nächsten Kapiteln behandelt.

## 3 Bestehende Ansätze

Diese Arbeit stellt Methoden für das Navigieren in Fragmenträumen vor, im Genaue-  
ren zur visuellen Exploration und Enumerierung von Molekülen mit gewünschten  
physikochemischen Eigenschaften aus Fragmenträumen. Da es meines Wissens nach  
keine vergleichbaren Ansätze gibt, sollen hier Methoden vorgestellt werden, die ent-  
weder in Fragmenträumen navigieren oder Moleküle enumerieren. Als Hilfsstruktur  
zur Enumerierung von Fragmenträumen werden in dieser Arbeit Bäume verwendet.  
Es werden daher auch Methoden zur Enumerierung von Bäumen vorgestellt. Die  
Methoden sind in den einzelnen Abschnitten chronologisch sortiert, so dass auch eine  
Evolution von einzelnen Methoden beobachtet werden kann, wie zum Beispiel im  
Fall von TOPAS und COLIBREE.

### 3.1 Navigieren von Fragmenträumen

In diesem Unterkapitel werden bestehende Ansätze zur Navigation von Fragment-  
räumen vorgestellt. Unter Navigation werden hier alle Methoden verstanden, die  
es erlauben, von einem Fragment/Molekül zu einem anderen Fragment/Molekül zu  
gelangen, wobei ein oder mehrere Kriterien die Navigation leiten. Alle Methoden  
verwenden entweder identische oder semantisch ähnliche Fragmenträume, so dass ein  
Vergleich noch sinnvoll ist.

#### Topas

*Topology assigning system* (TOPAS)[10] führt eine Ähnlichkeitssuche mithilfe von  
Evolutionären Algorithmen (EAs)[47] in Fragmenträumen durch. Zur Repräsentation  
von Molekülen wurden zwei verschiedene *Deskriptoren* verwendet. Der erste, ein bit-  
basierter Deskriptor, eignet sich, um eine generelle Ähnlichkeitssuche durchzuführen.  
Als zweiter Fingerprint wurde ein pharmakophor-basierter[48] Deskriptor verwendet.

In der Veröffentlichung[10] wurde gezeigt, dass TOPAS mit beiden verwendeten  
Deskriptoren relativ schnell konvergiert. Da beide Deskriptoren verschiedene Eigen-  
schaften auf Ähnlichkeit überprüfen, wurde auch gezeigt, dass TOPAS ein Programm  
ist, das in verschiedenen Szenarien eingesetzt werden kann. Verwunderlich ist, dass  
nur lineare, ja sogar einfach additive Deskriptoren verwendet wurden. Es können sehr  
viel effektivere Algorithmen verwendet werden, um Moleküle gemäß der in der Veröf-  
fentlichung verwendeten Deskriptoren zu generieren. Tatsächlich wurde der in dieser  
Arbeit vorgestellte Enumerator verwendet, um Moleküle gemäß eines vorgegebenen  
Deskriptors zu enumerieren, der sehr dem Pharmakophor-Deskriptor von TOPAS

ähnelt. Der hier vorgestellte Algorithmus konnte nach wenigen Schritten ein Molekül generieren, dessen Deskriptor sehr ähnlich dem vorgegeben Molekül-Deskriptor war. TOPAS benötigt mindestens 10 Iterationen, was bedeutet, dass mindestens 1.000 intermediäre Moleküle generiert werden. Es wäre daher sehr interessant gewesen zu sehen, wie sich TOPAS bei der Verwendung von nicht linearen Deskriptoren oder kompliziert linearen Deskriptoren, wie zum Beispiel Docking-Ergebnissen, verhält.

Der in dieser Arbeit vorgestellte Enumerator sucht nicht primär nach ähnlichen Molekülen wie TOPAS, sondern er generiert alle Moleküle, die, unabhängig ihrer Struktur, ein bestimmtes physikochemisches Profil erfüllen. Ein weiterer Unterschied ist, dass die Verwendung eines generischen EA Optimierer es erlaubt, verschiedene Deskriptoren zu verwenden. Es soll aber angemerkt werden, dass der Enumerator, ähnlich zu TOPAS, so entworfen wurde, dass er verschiedene Deskriptoren verwenden kann.

### **FTrees-FS**

FTREE-FS[11] sind eine Weiterentwicklung von *Feature Tree* (FTREE)[49] auf Fragmenträume. FTREE selbst ist ein baumbasiertes Ähnlichkeitsmaß für Moleküle. Zur Berechnung der Ähnlichkeit werden bestimmte Bindungen im Molekül ausgewählt, die die Kanten des Baums darstellen. Die funktionalen Gruppen zwischen diesen Bindungen modellieren die Knoten. Für jeden Knoten wird dann ein physikochemisches Profil berechnet. Die Ähnlichkeitsberechnung bestimmt dann ein Baummatching unter Berücksichtigung des physikochemischen Profils in jedem Knoten.

FTREE-FS erweitert den Feature Trees Ansatz dahingehend, dass als Knoten Fragmente eines Fragmentraums verwendet werden und Kanten nur zwischen Fragmenten mit kompatiblen Link-Typen zugelassen werden. In Kombination mit dynamischer Programmierung ist es möglich sehr effizient nach ähnlichen Molekülen, gemäß dem Feature Tree Ähnlichkeitsmaß, in Fragmenträumen zu suchen.

Vom Ansatz ist FTREE-FS sehr ähnlich zu der in dieser Arbeit vorgestellten Methode. In beiden Fällen werden Bäume für die Repräsentation von Molekülen verwendet. Ebenfalls wird in beiden Fällen ein physikochemisches Profil der Fragmente berechnet anhand dessen die Selektion der nächsten Fragmente gesteuert wird. Die Ziele und daraus folgend die zugrundeliegenden Vorgehensweisen sind aber sehr verschieden. FTREE-FS versucht Moleküle mit einer vorgegebenen Ähnlichkeit zu generieren. Die Vorgehensweise ist dabei, Fragmente mit ähnlichen physikochemischen Profilen an ähnlichen Stellen im Fragmentbaum zu platzieren. In FRAGENUM hingegen gibt es keine strukturellen Präferenzen. Sobald ein Fragment zu einem Molekül mit gewünschten physikochemischen Eigenschaften führen kann, wird es iterativ als Knoten an alle Fragment-Knoten mit kompatiblen Link-Atomen gehängt. Daraus resultiert, dass strukturell sehr verschiedene Moleküle erzeugt werden können, deren einzige Gemeinsamkeit ist, dass ihre physikochemischen Eigenschaften in vorgegebenen Intervallen zu liegen kommen.

## FlexNovo

FLEXNOVO[1] ist ein Programm zur strukturbasierten Suche in Fragmenträumen. Das heißt, FLEXNOVO versucht diejenigen Moleküle in einem Fragmentraum zu finden, die die größte Bindungsenthalpie zu einer gegebenen Proteinbindetasche besitzen. Für alle *Docking* relevanten Funktionen greift FLEXNOVO auf FLEXX[23] zurück. Im Gegensatz zu FLEXX, welches gegebene Moleküle *dockt*, kann FLEXNOVO neue Moleküle aus Fragmenträumen generieren und bewerten. FLEXNOVO erlaubt es auch, unter Zuhilfenahme der FLEXX Erweiterung FLEXX-PHARM[50], *Pharmakophore* zu definieren. Für das Aufbauen von Molekülen in einer gegebenen Bindetasche verwendet FLEXNOVO ein zyklisches Vorgehen. In jedem Zyklus wird jedes betrachtete Fragment erschöpfend um ein atomares Fragment erweitert. Das heißt, jede mögliche Erweiterung des bestehenden Fragments mit einem atomaren Fragment wird betrachtet. Um nicht alle Fragmente/Moleküle eines Fragmentraums zu testen, aber um dennoch gute Lösungen zu finden, verfolgt FLEXNOVO eine *k-Greedy* Strategie. In jedem Anbauzyklus  $z$  werden die besten  $k$  bewerteten Fragmente mit  $c$  verschiedenen Konformationen des vorhergehenden Zyklus  $z - 1$  verwendet. Um zu verhindern, dass einige wenige Fragmente die Ergebnisse dominieren und, um eine größere Variation in den Ergebnissen zu ermöglichen, unterstützt FLEXNOVO verschiedene Diversitätsfilter.

In einer Fallstudie wurde gezeigt, dass FLEXNOVO bekannte Inhibitoren bis zu einem gewissen Maß reproduzieren konnte. Eine vollständige Reproduktion war nicht zu erwarten, da dazu die Inhibitoren prinzipiell aus Fragmenten eines Fragmentraums assemblierbar sein müssen. Dies kann aber nicht kategorisch angenommen werden, insbesondere wenn verschiedene Filter zur Generierung von Fragmenträumen verwendet werden. Die Reproduktion dient auch primär der Validierung. Viel interessanter sind neue, unbekannte Strukturen. In diesem Bereich konnte FLEXNOVO einige interessante Ergebnisse produzieren. Alle Berechnungen konnten auf einem Standard PC in fünf bis zehn Stunden durchgeführt werden, wobei weniger als ein Gigabyte Speicher benötigt wurde. Benötigte Rechenzeit und Speicher sind direkt an die Größe des verwendeten Fragmentraums gekoppelt. Diese Ergebnisse zeigen aber, dass sinnvolle "Real World" Szenarien auf normalen *Personal Computer* (PC) durchgeführt werden können.

Im Gegensatz zu dem in dieser Arbeit vorgestellten Ansatz, sucht FLEXNOVO strukturbasiert in einem Fragmentraum. Das heißt, die Moleküle müssen nicht nur ein bestimmtes physikochemisches Profil aufweisen, sie müssen auch in bestimmten räumlichen Orten zu liegen kommen. Darüber hinaus werden bei FLEXNOVO pro Fragment mehrere Konformationen betrachtet, was die betrachteten Fragmenträume nochmals vergrößert. Um dennoch eine Laufzeit von wenigen Stunden zu ermöglichen, verwendet FLEXNOVO eine *k-Greedy* Strategie. Dies ist möglich, da FLEXNOVO letztendlich auf den *Score* optimiert. Der hier vorgestellte Enumerator besitzt keinen Wert, den er optimieren kann. Vielmehr versucht er, unter Zuhilfenahme von additiven Eigenschaften, so effizient wie möglich alle Moleküle in einem Intervall von physikochemischen Eigenschaften zu enumerieren.

## COLIBREE

*Combinatorial library breeding* (COLIBREE)[51] wurde wie FTREE-FS und TOPAS für eine ähnlichkeitsbasierte Suche in Fragmenträumen entwickelt. Wie bei TOPAS kommt eine stochastische Optimierung zum Suchen in Fragmenträumen zum Einsatz. Als Optimierungsmethode wird *Particle swarm optimization* (PSO)[52, 53] verwendet. PSO ist inspiriert vom biologischen Schwarmverhalten und versucht dies in *in silico* nachzuahmen. Als Deskriptor verwendet COLIBREE einen Vektor, der topologische Distanzen zwischen Atom-Typen speichert und *Chemical advanced template search* (CATS)[54, 55] benannt wurde. Die Ähnlichkeit oder *fitness* zu einem Referenz CATS Vektor kann dann einfach über die euklidische Distanz berechnet werden. Neben der CATS Repräsentation verwaltet ein Partikel noch weitere *Qualitäts-Vektoren*, die im Wesentlichen angeben wie oft ein Fragment oder Link-Fragment in bisherigen Lösungen vorkam und einen Einfluss bei der Auswahl von (Link-)Fragmenten im nächsten Optimierungsschritt besitzen. Qualitäts-Vektoren *ziehen* den Schwarm somit in eine Richtung im hochdimensionalen Suchraum. COLIBREE Fragmenträume unterscheiden sich etwas von den in dieser Arbeit verwendeten Fragmenträumen, da sie zwei Arten von Fragmenten besitzen: Fragmente und *Link-Fragmente*, wobei zwei Fragmente nur über ein Link-Fragment verknüpft werden können. COLIBREE verwaltet Moleküle, wie FRAGENUM, in einem Fragmentbaum, in dem Fragmente die Knoten und die Bindungen zwischen diesen die Kanten darstellen. In jedem Optimierungsschritt traversiert ein Partikel seinen Baum und ersetzt mit einer gewissen Wahrscheinlichkeit ein (Link-) Fragment.

COLIBREE wurde erfolgreich in einem Anwendungsszenario dazu verwendet, ähnliche Moleküle zu Rosiglitazon[56] in einem Fragmentraum zu suchen.

Grundsätzlich können hier alle Punkte angeführt werden, die auch schon bei TOPAS erwähnt wurden. Es stellt sich auch hier die Frage, wieso für einen einfachen additiven Deskriptor wie CATS ein stochastisches Optimierungsverfahren verwendet wird. Es könnte höchstwahrscheinlich mit dynamische Programmierung, ähnlich wie bei FTREE-FS, ein deterministisches Verfahren angewendet werden, um mithilfe von CATS in Fragmenträumen zu suchen. Auch könnte leicht eine Enumeration implementiert werden, die alle Moleküle mit identischen CATS Deskriptoren aufzählt. Dazu müsste nur jedes Fragment einen *Intra* und jedes Link-Atom mit einem *Inter-Fragment* CATS Vektor ausgestattet werden. Ein Inter-Fragment Vektor würde alle Atom-Typen speichern, die vom betrachteten Link-Atom erreichbar wären. Eine Verknüpfung zweier Fragmente würde dann nur eine einfache Addition der Intra-Fragment Vektoren und eine angepasste Addition der Inter-Fragment Vektoren bedeuten. Bei der Inter-Fragment Addition könnten die Werte für die Atom-Typen einfach summiert werden. Ihre topologische Distanz und somit ihre Position im resultierenden Vektor, ergäbe sich aber aus der Summe der betrachteten Atome zu ihrem Link-Atom. Abhängig von der maximalen Länge der betrachteten topologischen Distanz und der Größe des entstehenden neuen Fragments, müssten dann noch neue Inter-Fragment Vektoren für die Link-Atome der verbundenen Fragmente generiert



werden. Dies würde sich aber mit einer einfachen Tiefen- oder Breitensuche schnell realisieren lassen.

## ReCore

Das Modifizieren bekannter aktiver Strukturen ist eine häufig angewendete Methode in der pharmazeutischen Chemie, um neue Wirkstoffe zu generieren. Eine Möglichkeit neue Moleküle aus bekannten Molekülen zu generieren besteht darin, Teile eines Moleküls zu ersetzen, ein *Scaffold Hopping*[57] durchzuführen. Genau hier setzt RECORE[58, 59] an. RECORE wurde entwickelt, um zentrale Teile eines Moleküls, unter Beibehaltung der räumlichen Ausrichtung der nicht zu ersetzenden Teile, zu substituieren. In einer Vorverarbeitungsphase wird zunächst eine Datenbank von Fragmenten generiert, aus welcher dann nach Substituenten für den zentralen Teil eines Moleküls gesucht werden kann. Dazu werden als Eingabe Moleküle mit 3D-Informationen, Schneide- und Filterregeln benötigt. Schnitte werden bei RECORE an den Molekülen beziehungsweise deren Bindungen nur annotiert und nicht wirklich durchgeführt. Dies hat den Vorteil, dass alle räumlichen Informationen, wie zum Beispiel die relative Anordnung und der Torsionswinkel (siehe Abbildung 4.1), noch verfügbar sind. Filterregeln können verwendet werden, um festzulegen welche Fragmente verworfen werden sollen. Sogenannte *Exit-Vektoren* markieren Bindungen an zentralen, zu ersetzenden Fragmenten. Da immer ein zentrales Fragment eines Moleküls ersetzt wird, existieren immer mindestens zwei Exit-Vektoren. Diese Exit-Vektoren werden in einer rotations- und translationsinvarianten Datenbank gespeichert, um eine effiziente Suche zu ermöglichen. Neben Exit-Vektoren können noch Pharmakophor-Eigenschaften als Vektoren verwendet werden. Um ein effizientes Suchen nach Vektoren in der Datenbank zu ermöglichen, werden die Vektoren geometrisch sortiert und in *optimalen R-Bäumen*[60] gespeichert. Dazu wird der  $n$ -dimensionale Raum, der durch die Eigenschaften aufgespannt wird, gemäß VAM-SPLIT[61] rekursiv aufgeteilt.

Nachdem eine Datenbank von Fragmenten in Form von R-Bäumen erstellt wurde, kann nach alternativen Kernfragmenten gesucht werden. Dazu müssen an einem Molekül mindestens zwei rotierbare Bindungen markiert werden. Optional können noch Pharmakophore festgelegt werden. RECORE verwaltet Ergebnisse in einer Prioritätswarteschlange, wobei die Entfernung der Vektoren die Priorität bestimmen. Dieses Vorgehen ermöglicht es, Ergebnisse mit absteigender Ähnlichkeit effizient zu generieren. Werden neben mindestens zwei erforderlichen Exit-Vektoren noch andere Eigenschaften abgefragt, müssen mehrere R-Bäume mithilfe von Prioritätswarteschlangen traversiert werden.

RECORE exploriert chemische Räume durch das Ersetzen von Kernfragmenten in Molekülen. Obwohl RECORE nicht direkt auf chemischen Fragmenträumen im Sinne dieser Arbeit operiert, sind die Schneideregeln und die Fragmentierung von Molekülen von diesen abgeleitet. Durch das Annotieren von Schnitten, im Gegensatz zu dem Generieren von Fragmenten, werden benötigte 3D-Informationen beibehalten. Damit sind auch schon die zwei größten Unterschiede zu dieser Arbeit genannt:

Ersetzen des zentralen Fragments und extensiver Gebrauch von 3D-Informationen. Aufgrund der sehr spezifischen Vorgaben können effektive Datenstrukturen und Algorithmen verwendet werden, um chemische Räume zu erkunden. RECORE ist ein sehr schönes Beispiel dafür, wie Anforderungen, unter Verwendung von geeigneten Datenstrukturen und Algorithmen, elegant und performant in Software umgesetzt werden können.

### CoLibri

Zu COLIBRI gibt es leider nur wenig Literatur. Das was an Literatur vorhanden ist, beschreibt leider auch nur wie COLIBRI angewendet werden kann, aber nicht wie es funktioniert. COLIBRI wird als ein Verwaltungsprogramm für mehrere virtuelle kombinatorische Bibliotheken (VKBs) und/oder Fragmenträume[62] verwendet. Mithilfe von COLIBRI kann eine einheitliche Sicht auf mehrere, heterogene Räume generiert werden. Dies wird zum Beispiel dadurch ermöglicht, indem Link-Typen aus den verschiedenen Räumen unter einem Link-Typ zusammengefasst werden können. Unterstützt wird dies, indem die zugrundeliegenden Räume analysiert und in gewissem Maße modifiziert werden können. Dadurch, dass COLIBRI neben Fragmenträumen auch VKBs verarbeiten und als Fragmenträume zur Verfügung stellen kann, ermöglicht es fragmentraum basierenden Programmen VKBs als Eingabe zu verwenden. Dies wird in der angegebenen Zitierung unter anderem dazu verwendet, um eine Ähnlichkeitssuche über mehrere VKBs mit FTREE-FS, einem Fragmentraum basierenden Programm, durchzuführen.

COLIBRI besitzt eine gewisse Ähnlichkeit zu FRAGVIEW, da es beide Programme erlauben Fragmenträume zu verwalten und zu editieren. COLIBRI's Einsatzgebiet liegt aber hauptsächlich im Verwalten von mehreren und diversen Räumen. FRAGVIEW wurde hingegen konzipiert, um einen oder wenige Räume visuell zu explorieren. FRAGVIEW würde in einer Abfolge von Programmen für Fragmenträume vor COLIBRI verwendet werden. Mit FRAGVIEW würden einzelne Fragmenträume optimiert werden und mit COLIBRI könnten diese in einem späteren Schritt verschmolzen werden.

## 3.2 Enumerierung von Molekülen

Die folgende Aufzählung ist nicht erschöpfend. Sie beschränkt sich vielmehr auf bekannte oder, zu dem in dieser Arbeit vorgestellten Enumerationsprozess, ähnliche Ansätze.

### GDB- $n$

GDB- $n$ [63, 64, 65] ist eine Datenbank, die den Anspruch erhebt, alle chemisch *sinnvollen* Moleküle mit bis zu  $n$  Atomen enumeriert zu haben. Um einer kombinatorischen Explosion vorzubeugen, wurde die Menge der konstituierende chemischen Elemente bei GDB-11 auf C, N, O und F beschränkt. Der Enumerierungsprozess unterteilt sich in mehrere Phasen: In einem ersten Schritt wurden mithilfe des Programms

GENG[66] alle möglichen Graphen mit bis zu  $n$  Knoten und jeweils mit maximal vier Kanten erstellt. Im zweiten Schritt wurden verschiedene Filter verwendet, um chemisch instabile oder unerwünschte Graphen auszuschließen, was eine Reduzierung der Graphen um 98,14% zur Folge hatte. Für die restlichen Graphen wurden dann die Automorphie-Klassen[67] bestimmt, wodurch das Testen auf isomorphe Graphen sehr effizient wird. Im dritten Schritt wurden alle Knoten des Graphen zunächst durch Kohlenstoffe ersetzt und ungesättigte Valenzen mit Wasserstoffen abgesättigt. Dann wurden alle Kohlenstoffe systematisch durch die anderen Elemente ersetzt. Alle möglichen Kombinationen erzeugen  $1,7 \cdot 10^{12}$  unikale Moleküle. Nach Anwendung verschiedener Filter befanden sich 110.979.507 Moleküle in der Datenbank. Die gesamte Enumeration hat 1.600 *Central Processing Unit* (CPU) Stunden benötigt. Das Optimieren von einzelnen Schritten beziehungsweise das Verwenden von Heuristiken erlaubt es die Datenbank GDB-13 zu enumerieren. Fluor wurde durch Chlor ersetzt und es wurde mit Schwefel ein weiteres Element hinzugefügt. Obwohl durch die eingeführten Modifikationen weniger Moleküle pro  $n$  erzeugt wurden, liegen die Zahlen in derselben Größenordnung. Die Enumerierung ergab 910.111.673 Moleküle und dauerte 16.000 CPU Stunden.

Werden zunächst die Zahlen betrachtet, kann festgestellt werden, dass die Anzahl der Moleküle exponentiell mit der Anzahl der Atome steigt, was genau den Erwartungen entspricht. Das beobachtete Wachstum kann aber auch verwendet werden, um die Anzahl von Molekülen mit einer bestimmten Anzahl von Atomen zu schätzen. Das durchschnittliche Gewicht der Moleküle in GDB-11 beträgt  $153 \pm 7$  Da, was dem Gewicht typischer Fragmente im Wirkstoffentwurfs-Prozess entspricht[15]. Wird auf 25 Atome extrapoliert, der typischen Größe eines Wirkstoffs, entspricht das  $10^{27}$  Moleküle und liegt damit relativ nah an anderen Schätzungen[5].

Die enumerierten Moleküle wurden mit 63.857 Molekülen mit bis zu 11 Atomen aus bekannten Datenbanken verglichen. Der Vergleich von 63.857 zu 110.979.507 zeigt schon eindrucksvoll, dass der potentielle chemische Raum mit bekannten Strukturen bei weitem noch nicht ausgereizt wurde. Von den Molekülen aus der Referenzdatenbank (RDB) befanden sich 58,6% in der GDB, die Restlichen enthielten zum Beispiel Elemente oder Graphen, die für GDB ausgeschlossen wurden. Daraus folgt, dass diese Methode höchstwahrscheinlich noch zu konservative Kriterien für die Generierung von Molekülen enthält und den pharmazeutischen chemischen Raum noch nicht vollständig beschreibt. Alle Moleküle in GDB-11 erfüllen die *rule of 5*[7] und 50% die *rule of 3*[46]. Dies zeigt noch einmal eindrucksvoll, dass der Raum für potentielle *Leads* und Wirkstoffe bei weitem noch nicht ausgeschöpft wurde.

Der offensichtlichste Unterschied zu der in dieser Arbeit vorgestellten Methode ist die Enumeration von Atomen, im Gegensatz zu Fragmenten. Dennoch kann in der Vermeidung von Redundanzen ein gemeinsames Problem gefunden werden. Aufgrund der Verschiedenartigkeit von Atomen und Fragmenten konnte die Lösung von GDB- $n$  nicht für FRAGENUM verwendet werden. Grundsätzlich bauen aber beide Redundanzfilter auf demselben Prinzip auf, die Verwendung von Invarianten. Vergleiche zwischen GDB- $n$  und Datenbanken bekannter Strukturen zeigen eindrucksvoll, dass bisher nur ein sehr kleiner Bruchteil des pharmazeutisch interessanten chemischen

Raums betrachtet wurde. Ein Ergebnis, das sich mit den gefundenen Ergebnissen des in dieser Arbeit vorgestellten Enumerators deckt (siehe Kapitel 6).

### SmiLib

SMILIB[68, 69] ist ein Programm zur Enumerierung von VKBs[70, 71]. VKBs unterscheiden sich von Fragmenträumen im Wesentlichen dadurch, dass es meistens ein zentrales Fragment gibt (Kern), an welches alle anderen kompatiblen Reagenzien<sup>1</sup>, angebaut werden und, dass meistens nur eine sehr geringe Anzahl von chemischen Reaktionen modelliert werden, was sich in sehr wenigen Link-Typen niederschlägt. Fragmenträume besitzen diese Einschränkungen nicht und können daher als eine Verallgemeinerung von VKBs angesehen werden. SMILIB verwendet zur Repräsentation von Fragmenten und Molekülen die *Simplified molecular-input line-entry specification* (SMILES)[72] Notation. Die Enumerierung erfolgt durch das Ersetzen von Zeichenketten: Link-Atome werden einfach durch die SMILES Zeichenkette der anzufügenden Fragmente ersetzt, was eine Enumerierung von bis zu 3.500.000 Molekülen pro Minute erlaubt.

Es wird leider nur kurz in der ersten Veröffentlichung[68] erwähnt, dass oft eine komplette Enumerierung von virtuellen kombinatorischen Bibliotheken nicht gewünscht ist und dass der Ansatz daher auch mit Filtern kombiniert werden kann. Es wird aber nicht weiter erläutert, wann und wie die Filter zum Einsatz kommen, insbesondere nicht, ob die Filter zur Auswahl von anzuhängenden Fragmenten verwendet werden. Ich vermute, dass nachgeschaltete Filter gemeint sind, das heißt, ein Molekül wird enumeriert und dann Subjekt eines Filters. In SMILIB V2.0 besteht die Möglichkeit die Enumerierung einzuschränken, indem durch Regeln festgelegt werden kann, welche Fragmente an welche Kern-Fragmente angehängt werden können. Es wird leider nicht erwähnt, wie Duplikate erkannt beziehungsweise vermieden werden.

Die größten Unterschiede, zu dem in dieser Arbeit vorgestellten Ansatz, scheinen die nicht Verwendung von Filtern während des Enumerierungsprozesses und das Fehlen eines Mechanismus zum Erkennen oder Entfernen von Duplikaten zu sein. Die Einschränkung des Algorithmus auf kombinatorische Bibliotheken ist nicht wirklich eine Einschränkung, da der Ansatz wohl ohne viele Änderungen direkt für Fragmenträume übernommen werden kann. Es besteht mit COLIBRI auch ein Programm, welches eine Fragmentraum-Sicht auf VKBs anbietet. Die Enumerierung von Zeichenketten, die Moleküle repräsentieren, scheint sehr attraktiv, da sie relativ einfach und sehr effizient ist. Tatsächlich wurde in einer frühen Entwicklungsphase, des in dieser Arbeit vorgestellten Enumerators, darüber diskutiert, eine sehr ähnliche Zeichenketten basierte Enumerierung mithilfe von SMILES durchzuführen. Diese Idee wurde zugunsten von Bäumen verworfen, da für diese eine einfache und effektive Möglichkeit zur Vermeidung von strukturellen Duplikaten entwickelt werden konnte.

---

<sup>1</sup>Reagenzien können als das Pendant zu Fragmenten in VKBs angesehen werden.

### 3.3 Enumerierung von Bäumen

Es lässt sich relativ viel Literatur zum Enumerieren von Bäumen finden[73, 74, 75, 76], auch chemisch Motivierte[77]. Da es sich aber um mathematische Ansätze handelt, wird unter Enumeration das Zählen von verschiedenen Bäumen verstanden und nicht ihre algorithmische Generierung. Werden die Ansätze betrachtet, fällt auf, dass es sehr viele Arten von Bäumen gibt. Das erste Problem besteht also darin, genau zu definieren, welche Bäume enumeriert werden sollen. Die Grundcharakteristika eines Fragmentbaums sind, dass es eine eindeutige Wurzel hat, gerichtet ist, seine Knoten unterscheidbar sind und die Ordnung der Kinder eines Knotens nicht relevant ist. Dies entspricht einem *gewurzelten, gerichteten und annotierten* Baum, der *nicht geordnet* ist<sup>2</sup>. Diese Eigenschaften setzen implizit voraus, dass jeder Knoten mit jedem anderen verbunden werden und dass jeder Knoten beliebig viele Kanten haben kann. Fragmentbäume weichen von diesen Eigenschaften etwas ab, da bei diesen nur Fragmente miteinander verbunden werden können, die freie und kompatible Link-Atome besitzen. Weiterhin ist die Anzahl der Verknüpfungen für jedes Fragment durch die Anzahl der Link-Atome beschränkt. In typischen Fragmenträumen haben die meisten Fragmente nur ein Link-Atom, das heißt, sie können nur eine Verbindung ausbilden.

Nachdem prinzipiell feststand, welche Art von Bäumen enumeriert werden soll, wurde versucht ein Algorithmus zur Erzeugung dieser zu finden. Wie eingangs erwähnt, findet sich nur Literatur zum Zählen aber nicht zum Generieren von Bäumen. Dennoch bestand die Hoffnung, über die meist rekursive Definition der Zählmethoden, einen Algorithmus für das Generieren von Bäumen ableiten zu können. Leider erwiesen sich die Formeln als nicht geeignet, um daraus Algorithmen abzuleiten, insbesondere, da die Autoren bemüht waren eine geschlossene Form einer Formel zu präsentieren. So ist zum Beispiel die Anzahl der oben beschriebenen Bäume (gewurzelt, gerichtet, annotiert und ungeordnet) durch  $n^{n-2}$  gegeben, für Bäume mit  $n$  Knoten (siehe Satz 5.2.1).

Da gewisse chemische Strukturen sich durch Bäume gut repräsentieren lassen, sind neben Mathematikern und Informatikern auch Chemiker sehr daran interessiert Bäume aufzuzählen. Tatsächlich ist die Situation für chemische Strukturen sehr ähnlich zu Fragmentbäumen. Elemente haben verschiedene Wertigkeiten und nicht immer kann jedes Element mit jedem anderen eine Verbindung eingehen. Da die Aufgabe sehr ähnlich ist, müssen auch sehr ähnliche Probleme gelöst werden, wie zum Beispiel das Vermeiden von Redundanzen[78, 67]. Die Strategie zur Lösung besteht immer darin, eine Invariante zu finden und diese auszunutzen. Genau dieser Ansatz wird auch in dieser Arbeit, durch die Verwendung von IDs für Fragmente, verfolgt.

---

<sup>2</sup>Wird jedoch Definition 4.4.4 betrachtet, wird man feststellen, dass die in dieser Arbeit verwendeten Bäume sehr wohl eine Ordnung voraussetzen. Dies Voraussetzung ist aber nur notwendig, um einen einfachen Topologietest auf Bäumen zu ermöglichen, sie ist aber nicht für die eigentliche Enumeration notwendig.

Der größte Unterschied zu allen in der Literatur gefundenen Bäumen und zu den in dieser Arbeit vorgestellten Fragmentbäumen besteht darin, dass bei letzteren Kompatibilitäten und eine beschränkte Anzahl von Link-Atomen berücksichtigt werden müssen. Die größte Ähnlichkeit besteht zu gewissen Bäumen, die im Zusammenhang von chemischen Zählproblemen eingeführt wurden. Diese Ansätze konnten nicht direkt übernommen werden aber es wird, wie bei diesen, letztendlich eine Invariante ausgenutzt, um ein effizientes Enumerieren zu ermöglichen.

# 4 Modellierung und theoretische Grundlagen

In diesem Kapitel werden die für diese Arbeit relevanten Konzepte und Modelle vorgestellt. Zunächst werden Atome, chemische Elemente, Bindungen und Moleküle eingeführt. Aufbauend auf diesen Grundlagen werden chemische Fragmente und Fragmenträume erläutert und pharmazeutisch motiviert. Abgeleitet von den biochemischen Modellen werden die entsprechenden formalen Strukturen definiert, welche dann verwendet werden, um Fragmenträume als Algebra zu beschreiben.

## 4.1 Organische Moleküle

Dieser Abschnitt gibt eine Einführung in die notwendigen biochemischen Grundlagen. Im Einzelnen sind dies Moleküle und deren Konstituenten: Chemische Elemente beziehungsweise Atome und chemische Bindungen. Während die Beschreibung von Atomen und chemischen Elementen der allgemeinen Chemie entnommen ist, beschränkt sich die Beschreibung von Bindungen und Molekülen auf die organische Chemie. Alle Informationen in diesem Kapitel stammen aus [79, 80, 13, 14] und [15]. Anzumerken sei noch, dass dieses Kapitel nicht den Anspruch erhebt, eine vollständige und wissenschaftlich aktuelle Einführung zu geben. Vielmehr werden chemische Konzepte bis zu einem Grad eingeführt, welcher für diese Arbeit notwendig ist<sup>1</sup>. Dadurch bedingt werden einige Einzelheiten vernachlässigt oder vereinfacht. Oder, um es mit den Worten von Sir Karl Popper auszudrücken: *Science may be described as the art of systematic over-simplification - the art of discerning what we may with advantage omit*[81].

Die wichtigsten biochemischen Strukturen dieser Arbeit sind Moleküle. Ein Molekül besteht aus mindestens zwei kovalent verbundenen Atomen und stellt für einen Organismus eine stabile Struktur da.

In dieser Arbeit werden nur die Elemente der organischen Chemie betrachtet. Die organische Chemie wird gelegentlich auch als Kohlenstoffchemie bezeichnet, wodurch das wichtigste Element schon genannt wäre: Kohlenstoff (C). Obwohl grundsätzlich alle Elemente in organischen Verbindungen vorkommen können, bestehen diese neben Kohlenstoff hauptsächlich aus Wasserstoff (H), Sauerstoff (O), Stickstoff (N), Schwefel (S), Halogenen<sup>2</sup> und Phosphor (P). Auf den ersten Blick mag dies wie eine Einschränkung wirken, wenn berücksichtigt wird, dass zur Zeit 118 Elemente bekannt

<sup>1</sup>Hauptsächlich für die physikochemischen Eigenschaften von Molekülen.

<sup>2</sup>Elemente: Fluor (F), Chlor (Cl), Brom (Br), Iod (I) und Astat (At).

sind[82]. Aber aufgrund der chemischen Eigenschaften dieser Elemente, insbesondere von Kohlenstoff, sind viel mehr organische Verbindungen bekannt als anorganische. X, R und L sind keine gültigen Elementbezeichnungen und werden meistens als Platzhalter benutzt. Das Symbol X steht für ein beliebiges Atom. R bezeichnet einen Rest und kann im Gegensatz zu X für ein ganzes Submolekül stehen. Das Symbol L, meistens mit hochgestellter Zahl, wird in dieser Arbeit für Link-Atome verwendet, die in Kapitel 4.2.1 eingeführt werden.

Chemische Bindungen beruhen grundsätzlich auf Vorgängen, die in der Elektronenhülle der Atome ablaufen. Elektronen auf dem äußersten Orbital werden *Valenzelektronen* genannt und nur diese können eine Paarung mit einem Valenzelektron eines anderen Atoms eingehen. Eine einfache chemische Bindung ist eine Paarung von jeweils einem Valenzelektron eines Atoms mit einem Valenzelektron eines anderen Atoms. Diese Paarung wird *kovalente Bindung*, *Atombindung* oder auch *Elektronenpaarbindung* genannt. Ist mehr als ein Elektron pro Atom beteiligt, wird von *Doppel-*, *Dreifach-* oder *Vierfachbindung* gesprochen. Bindungen werden graphisch durch Striche zwischen den Elementen dargestellt, Mehrfachbindungen durch die entsprechende Anzahl von Strichen:  $\text{C} - \text{C}$ ,  $\text{C} = \text{C}$  oder  $\text{C} \equiv \text{C}$ .

Prinzipiell kann jedes ungepaarte Valenzelektron eines Atoms eine Bindung mit einem ungepaarten Valenzelektron eines anderen Atoms eingehen. Eine Bindung etabliert sich aber nur beziehungsweise bleibt stabil, wenn die neu entstandene Verbindung energetisch günstig ist.

### Schwache Wechselwirkungen

Schwache, nicht permanente Wechselwirkungen erlauben es biologischen Prozessen, Moleküle kurz und reversibel miteinander interagieren zu lassen. Für diese Bindungen macht sich die Natur verschiedene schwächere Wechselwirkungen zunutze. Die für diese Arbeit relevanten nicht kovalenten Wechselwirkungen sind<sup>3</sup>:

- Ionische Wechselwirkungen, Salzbrücken.
- Wasserstoffbrücken.
- Hydrophobe Wechselwirkungen.

Die ersten beiden Wechselwirkungen lassen sich unter elektrostatischer Wechselwirkung subsumieren und beruhen auf der unterschiedlichen Neigung von Elementen Elektronen zu binden. Dieser Unterschied findet in der *Elektronegativität* seinen Ausdruck. Sind zwei Atome kovalent gebunden, werden die Elektronen in Richtung des Atoms mit der höheren Elektronegativität gezogen. Diese Verschiebung von Elektronen induziert Partialladungen<sup>4</sup>, welche wiederum ein Dipol generieren. An den Polen

---

<sup>3</sup>Weitere häufig auftretende, aber hier nicht besprochene Wechselwirkungen sind: Van-der-Waals Wechselwirkung, Metallkomplexierung und Kation- $\pi$ -Wechselwirkung.

<sup>4</sup>Diese Wechselwirkung ist nicht mit der Ionischen-Bindung zu verwechseln, bei der ein Atom komplett ein Elektron abgibt. Es gibt keine genaue Festlegung, wann eine ionische Wechselwirkung



dieses Dipols können jeweils entgegengesetzt geladene Gruppen durch Dipol-Dipol-Wechselwirkung anlagern. Eine Wasserstoffbrücke basiert ebenfalls auf der Anziehung entgegengesetzter Ladungen, ist aber spezifischer in dem Sinne, dass immer Wasserstoff involviert ist und dass diese Wechselwirkung geometrisch restriktiv ist. Das an den Wasserstoff gebundene Atom muss eine höhere Elektronegativität als Wasserstoff besitzen und wird als *Wasserstoffdonor* bezeichnet. Die entgegengesetzte, partiell negativ geladene Gruppe, wird *Wasserstoffakzeptor* genannt:  $\delta^- \text{D} - \text{H}^{\delta+} \cdots \delta^- \text{A}$ . D steht für einen Wasserstoffdonor mit hoher Elektronegativität ( $\delta^-$ ), an welchen ein Wasserstoff-Atom kovalent gebunden ist. A ist ein Wasserstoffakzeptor, ebenfalls mit einer hohen Elektronegativität. Durch die geometrische Restriktivität eignen sich Wasserstoffbrücken sehr gut für Selektivität, da geringe Abweichungen von der idealen räumlichen Ausrichtung von Akzeptor und Donor schon einen starken Einfluss auf die Stärke der Wechselwirkung einer Wasserstoffbrücke haben.

Der Begriff Wechselwirkung bei hydrophoben oder *unpolaren* Atomen/Gruppen ist etwas irreführend, da im eigentlichen Sinn keine Wechselwirkung stattfindet. Es sollte vielmehr von einem hydrophoben Effekt in einem polaren Medium, wie zum Beispiel Wasser, gesprochen werden. In Wasser können unpolare Gruppen keine Wasserstoffbrücken eingehen, was zur Folge hat, dass Wassermoleküle in direkter Nachbarschaft zu hydrophoben Gruppen weniger Wasserstoffbrücken ausbilden können. Des Weiteren werden die Wassermoleküle an der unpolaren Oberfläche in eine höhere Ordnung gezwungen, da sie nur Wasserstoffbrücken ausbilden können, die von der hydrophoben Oberfläche wegzeigen, was eine Reduzierung der Entropie zur Folge hat. Es ist daher energetisch und entropisch günstiger, wenn sich hydrophobe Oberflächen direkt aneinander lagern und somit ihre Gesamtoberfläche verkleinern. Aus der Natur des hydrophoben Effekts ist leicht ersichtlich, dass dieser mit der Größe der Kontaktflächen der beteiligten hydrophoben Gruppen wächst und dass dieser Effekt typischerweise ungerichtet ist. Eine Ausnahme bilden hier aromatische Systeme. Bei diesen Systemen gibt es eine relative Orientierung zwischen beteiligten Aromaten, die den Effekt maximiert.

Wie eingangs erwähnt, besteht ein Molekül aus mindestens zwei kovalent verbundenen Atomen und stellt für einen Organismus eine stabile Struktur da. Die Stabilität bezieht sich aber nur auf die Bindungen, nicht auf die relative räumliche Verteilung der Atome eines Moleküls. Dies ist darauf zurückzuführen, dass die verschiedenen Bindungstypen und intramolekularen Kräfte den möglichen Konformationsraum, also die räumliche relative Anordnung der Atome eines Moleküls, einschränken, ihn aber nicht auf nur eine mögliche Konformation restriktivieren. Von den möglichen Freiheitsgraden werden in dieser Arbeit nur Rotationen um eine Bindung betrachtet. Die beiden anderen, *Bindungslänge* und *Bindungswinkel*, werden nicht näher betrachtet.

Die Rotationseigenschaft von Bindungen ist auf Einfachbindungen beschränkt. Jedoch gilt nicht der Umkehrschluss, dass alle Einfachbindungen rotierbar sind. Die Rotation um eine Bindung wird durch den Winkel zweier gedachter Parallelogramme

---

zu einer Ionischen-Bindung wird. In der Literatur ist häufig ein Wert von  $\Delta E$  1,8 zu finden. Dieser ist aber eher als ein Richtwert zu betrachten und nicht als eine harte Schwelle.

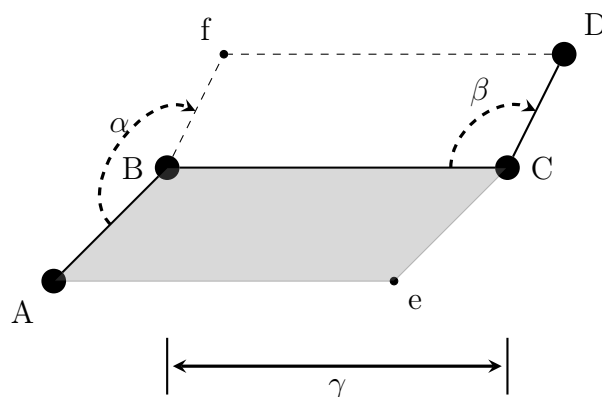


Abbildung 4.1: Freiheitsgrade in einem Molekül.

Fiktives Molekül mit vier Atomen A, B, C und D und drei Bindungen zwischen diesen.  $\alpha$  beschreibt den Torsionswinkel, der zwischen den beiden gedachten Parallelogrammen entsteht, die durch A, B, C, e und B, C, D, f aufgespannt werden. Der Winkel  $\beta$  bezeichnet den Bindungswinkel zwischen B, C und C, D.  $\gamma$  ist die Bindungslänge zwischen B und C.

beschrieben, die entstehen, wenn die jeweils direkt benachbarten Bindungen mit betrachtet werden (siehe Abbildung 4.1). Bedingt durch die räumliche Anordnung der Atome und die dadurch entstehenden Abstoßungseffekte ist jeder Torsionswinkel mit einer Energie verbunden. Daher werden meistens nicht beliebige Torsionswinkel eingenommen, vielmehr bewegt sich der Torsionswinkel immer von Minimum zu Minimum.

Zum Schluss soll noch auf den Lipophilie-Verteilungskoeffizienten eingegangen werden.

## log P

Der Lipophilie-Verteilungskoeffizient<sup>5</sup>, oder kurz log P<sup>6</sup> genannt, gibt die Konzentrationsverteilung eines Moleküls zwischen einem unpolaren und polaren Medium wieder. Als Medien werden meistens Oktanol und Wasser verwendet.

$$P = \frac{\text{Molekülkonzentration in unpolarem Medium (z. B. Oktanol)}}{\text{Molekülkonzentration in polarem Medium (z. B. Wasser)}}$$

Dieser Wert gibt im Wesentlichen an, wie (un)polar ein Molekül ist. Ein polares Molekül lässt sich häufiger in Wasser wiederfinden, da es Interaktionen mit dem polaren Wasser eingehen kann, aber keinen polaren Gegenpart im lipophilen Medium findet. Ist ein Molekül hingegen unpolar wird es aufgrund des hydrophoben Effekts in

<sup>5</sup>Lipophilie beschreibt die "Fettliebbarkeit" eines Moleküls oder wie hydrophob es ist.

<sup>6</sup>P steht für: *Partition Coefficient*.

das unpolare Oktanol gedrängt. Somit ist  $\log P > 0$ , wenn es sich um ein lipophiles und  $< 0$ , wenn es sich um ein hydrophiles Molekül handelt. Zur Berechnung des  $\log P$  Wertes haben sich verschiedene Methoden etabliert[83]. Alle Methoden basieren im Wesentlichen darauf, den  $\log P$  Wert eines Moleküls mithilfe der gemessenen  $\log P$  Werte der konstituierenden Atome oder Fragmente anhand einer Regression zu berechnen. In der Literatur steht  $c \log P$  entweder für *calculated*  $\log P$  oder für ein Programm aus dem Daylight[84] Programmpaket. Für den Rest dieser Arbeit steht  $c \log P$  für irgendeine Methode, die  $\log P$  berechnet.

## 4.2 Chemische Fragmenträume

Die dieser Arbeit zugrundeliegenden chemischen Fragmenträume bestehen aus einer Menge von Fragmenten und einer Menge von Regeln und enthalten alle Fragmente und Moleküle, die sich aus den initialen Fragmenten in Kombination mit den Regeln theoretisch generieren lassen. Trotz ihrer Kürze liefert diese Zusammenfassung schon eine recht gute und anschauliche Beschreibung chemischer Fragmenträume. Der Rest dieses Abschnitts widmet sich den Details von Fragmenten, Verknüpfungs-, und Terminierungsregeln und den darauf aufbauenden chemischen Fragmenträumen.

### 4.2.1 Fragment

Fragmente sind zusammenhängende, meistens kleine, chemische Teilmoleküle, welche nur vollständige Ringsysteme und mindestens ein spezielles Verknüpfungsatom enthalten. Diese Verknüpfungsatome besitzen außer einem *Typ* keine weiteren Eigenschaften, insbesondere keine physikochemischen. Der üblichen Terminologie folgend[11], werden diese *Link* oder *Link-Atome* beziehungsweise *Link-Typ* genannt, wenn nur der Typ von Interesse ist. Im Folgenden werden Link-Atome immer durch  $L^i$  dargestellt, wobei das  $i$  für den Link-Typ steht. Link-Typen werden in dieser Arbeit immer durch natürliche Zahlen dargestellt, wobei eine Zahl eindeutig einen Link-Typ in einer Regelmenge kodiert. Fragmente, die nicht weiter fragmentiert werden können, werden *atomare Fragmente* genannt. Unglücklicherweise ist der Begriff des chemischen Fragments in der Chemie und insbesondere in der pharmazeutischen Chemie schon mehrfach und semantisch ähnlich belegt. Wenn in dieser Arbeit über chemische Fragmente gesprochen wird und nicht obige Beschreibung gemeint ist, wird explizit darauf hingewiesen.

### 4.2.2 Regeln

Neben Fragmenten werden noch Regeln für das Verknüpfen und Terminieren von Link-Typen für einen Fragmentraum benötigt. Die Verknüpfungsregeln legen fest, welche Link-Typen kompatibel sind und somit dazu verwendet werden können, neue Fragmente zu generieren. Die virtuelle Synthese zweier Fragmente erfolgt dabei unter Substituierung der beiden beteiligten Link-Atome und ihrer Bindung durch eine

neue Bindung zwischen den beiden, zu den Link-Atomen adjazenten Atomen (siehe Abbildung 4.2). Verknüpfungsregeln sind binär, das heißt, falls eine Kompatibilität zwischen zwei Link-Typen besteht, können die entsprechenden Fragmente immer über diese verbunden werden. Ferner gibt es in Fragmenträumen für jeden Link-Typ ein spezielles *Link-Terminal*, welches ein Link-Atom mit einer sinnvollen chemischen Gruppe substituiert. Diese Terminierung dient dazu, ein Fragment in ein Molekül zu transformieren oder den Anbau weiterer Fragmente in einem bestimmten Bereich eines Fragments zu unterbinden. Es ist offensichtlich, dass Link-Terminals nur ein Link-Atom besitzen dürfen und, dass der Typ mit dem zu terminierenden Link-Typ kompatibel sein muss. Um zu verhindern, dass unerwünschte physikochemische Eigenschaften einem Fragment durch ein Link-Terminal hinzugefügt werden, sind Link-Terminals meistens klein und bezüglich ihrer Umgebung relativ neutral. Typische terminale Gruppen sind  $L-H$  oder  $L-CH_3$ . Jede Regel kann außer der

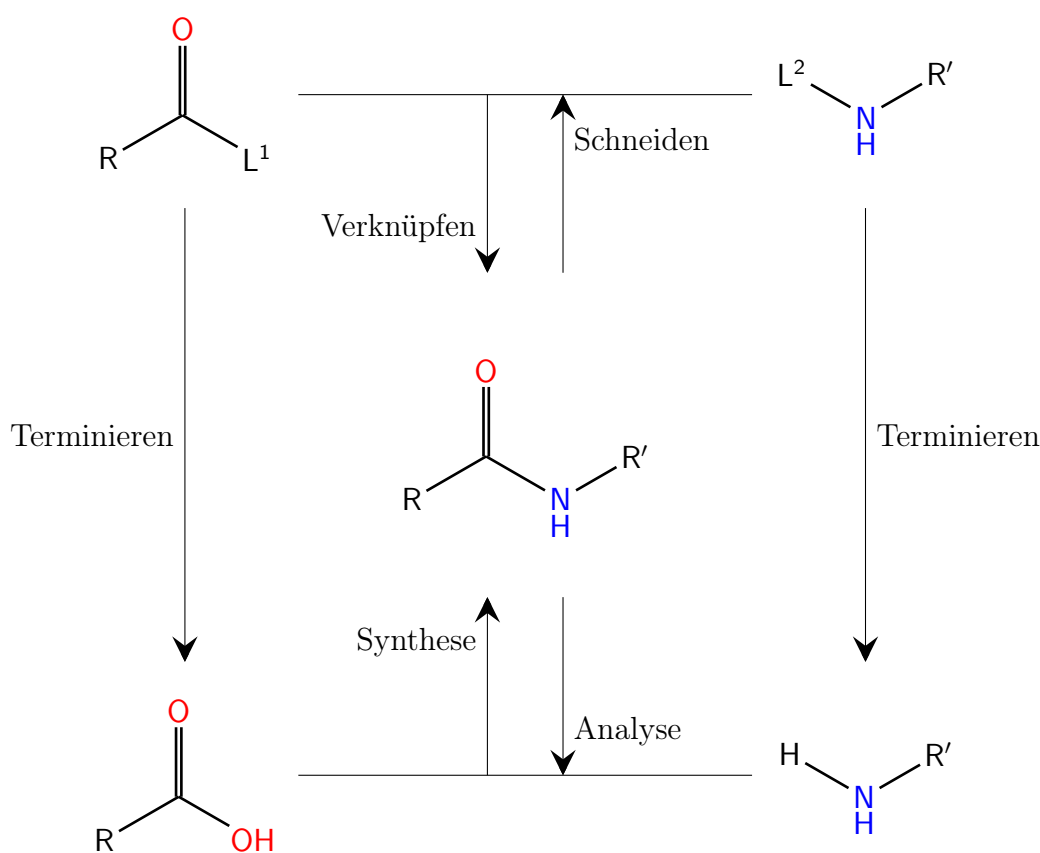


Abbildung 4.2: Modellierung chemischer Reaktionen durch Fragmente und Regeln. Oben in der Abbildung sind zwei hypothetische Fragmente mit kompatiblen Link-Typen  $L^1$  und  $L^2$  zu sehen. In der Mitte die resultierende Verknüpfung und unten die terminierten Fragmente. Die terminierten Fragmente sind die Edukte der modellierten chemischen Reaktion, die zu demselben Produkt führt. R und R' stehen für Reste der Fragmente beziehungsweise Moleküle. Vorlage der Grafik aus [1].

Kompatibilität noch nötige Anpassungen beschreiben, die bei einer Verknüpfung vorgenommen werden müssen. Dies kann den Typ, Länge und Torsionswinkel der entstehenden Bindung beinhalten sowie gegebenenfalls nötige Änderungen der Hybridisierung der an der neuen Bindung beteiligten Atome. Mithilfe dieser Informationen können Regeln als eine einfache Modellierung chemischer Reaktionen betrachtet werden. Die Betonung liegt hier aber auf einfach, da viele Aspekte einer chemischen Reaktion, die zum Teil einen erheblichen Einfluss auf die Reaktion haben, durch die Anpassungen nicht berücksichtigt werden. All die Informationen, die nicht modelliert werden, müssen somit indirekt beim Design der Regeln berücksichtigt werden.

### 4.2.3 Fragmentraum Generierung

Obwohl nicht Gegenstand dieser Arbeit, soll hier ein kurzer Exkurs in die Generierung von Fragmenten und Fragmenträumen aus Molekülen gegeben werden. Die in dieser Arbeit verwendete Definition von Fragmenträumen geht auf eine Veröffentlichung von Lewell et al. [85] zurück. In dieser wurde die Methode *Retrosynthetic combinatorial analysis procedure* (RECAP) vorgestellt, die auf systematische Weise Fragmente aus Molekülen generiert. Die Vorgehensweise dabei ist, Schneideregeln zu definieren, die eine zu schneidende Bindung anhand ihrer chemischen Umgebung festlegt. Die Schneideregeln können darüber hinaus Bedingungen enthalten, wann ein Schnitt nicht durchzuführen ist, um dadurch zum Beispiel zu kleine Fragmente oder Fragmente mit unerwünschten chemischen Gruppen zu vermeiden. Ferner werden zu jeder Schnittregel zwei Link-Typen definiert, mit welchen die entstehenden Enden nach einem Schnitt annotiert werden. Fragmenträume unterscheiden sich etwas von RECAP, da nicht die beiden Atome der geschnittenen Bindung annotiert werden, vielmehr wird die Schnittbindung durch eine neue ersetzt, an deren Ende ein Link-Atom mit dem entsprechenden Typ sitzt. Nachdem alle zu schneidenden Bindungen an einem Molekül identifiziert wurden, werden alle Schnitte simultan durchgeführt. Das bedeutet, es entsteht kein Fragment, welches nach den gegebenen Regeln weiter fragmentiert werden könnte. Diese Prozedur wird an allen gewünschten Molekülen durchgeführt und anschließend werden Fragment-Duplikate entfernt.

Dieses Vorgehen stellt einen sehr attraktiven Ansatz dar, da es Anwendern die Möglichkeit gibt, chemisches und pharmazeutisches Wissen in Regeln zu transformieren. Chemisches Wissen kann in die Festlegung bekannter Synthesen als Schnittregeln einfließen. Dadurch wird, so die Idee, die Wahrscheinlichkeit erhöht, dass Moleküle, die aus Fragmenten zusammengesetzt werden, zu denen eine bekannte Synthese besteht, auch synthetisch zugänglicher sind. Pharmazeutisches Wissen kann verwendet werden, um Schnittregeln so zu definieren, dass bekannte Wirkstoffgruppen in Fragmenten konserviert werden und somit die Wahrscheinlichkeit erhöht wird, dass aus Fragmenten zusammengesetzte Moleküle pharmazeutische Wirkung zeigen.

Die Generierung von Molekülen ist die inverse Operation zum Schneiden: Zwei kompatible Fragmente werden verbunden, indem die beiden beteiligten Link-Atome und ihre Bindung durch eine neue Bindung ersetzt werden. Hat nach mehreren Anbausritten ein Fragment die gewünschten Eigenschaften, aber noch offene Link-

Atome, können diese durch die terminalen Fragmente substituiert werden. Obwohl nicht empfehlenswert, spricht grundsätzlich nichts dagegen verschiedene Regelmengen für das Schneiden und das Verknüpfen von Fragmenten zu verwenden, solange dieselben Link-Typ Bezeichnungen in beiden Regelmengen verwendet werden.

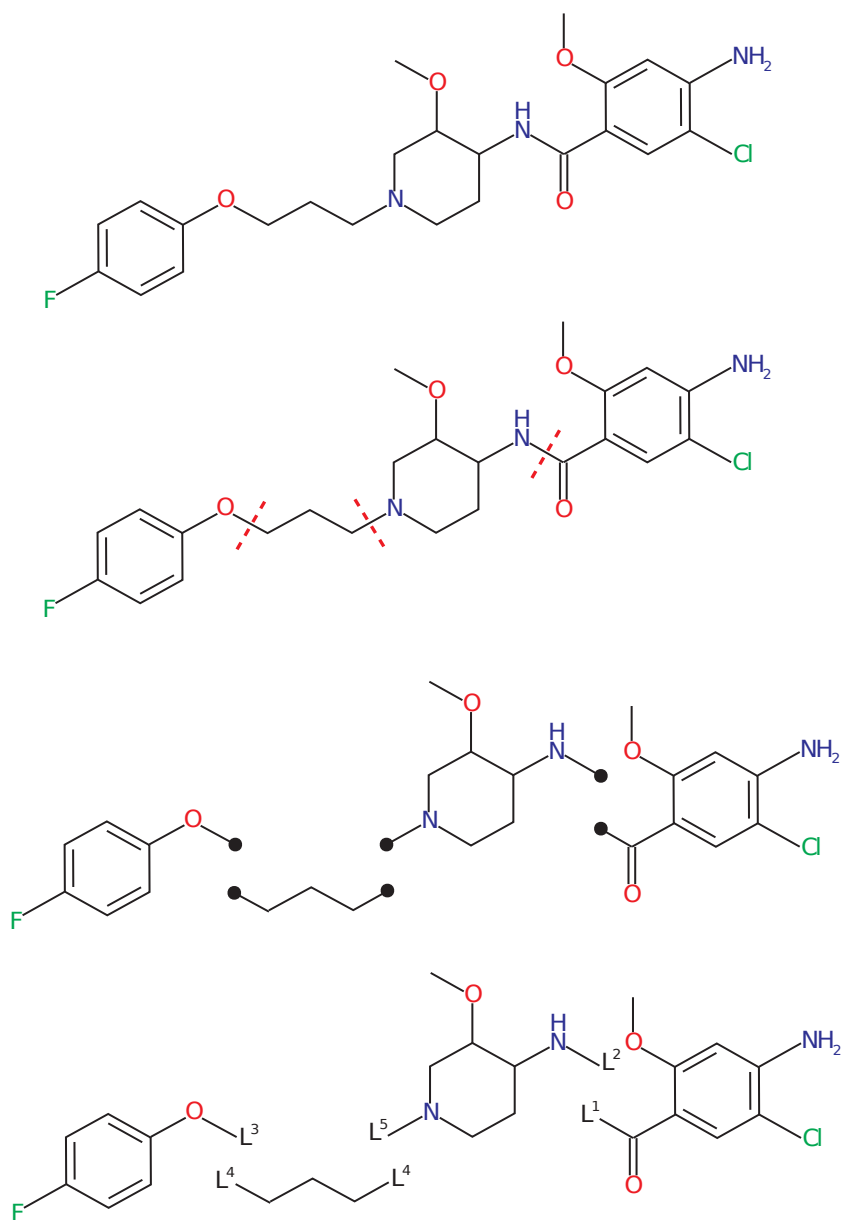


Abbildung 4.3: Generierung von Fragmenten[85].

Ganz oben ist das Wirkstoff-Molekül Cisapride zu sehen. In einem ersten Schritt werden alle Bindungen markiert, die gemäß der verwendeten Regeln geschnitten werden. Im nächsten Schritt werden die Bindungen geschnitten und zum Schluss mit den Regeln entsprechenden Link-Typen annotiert.

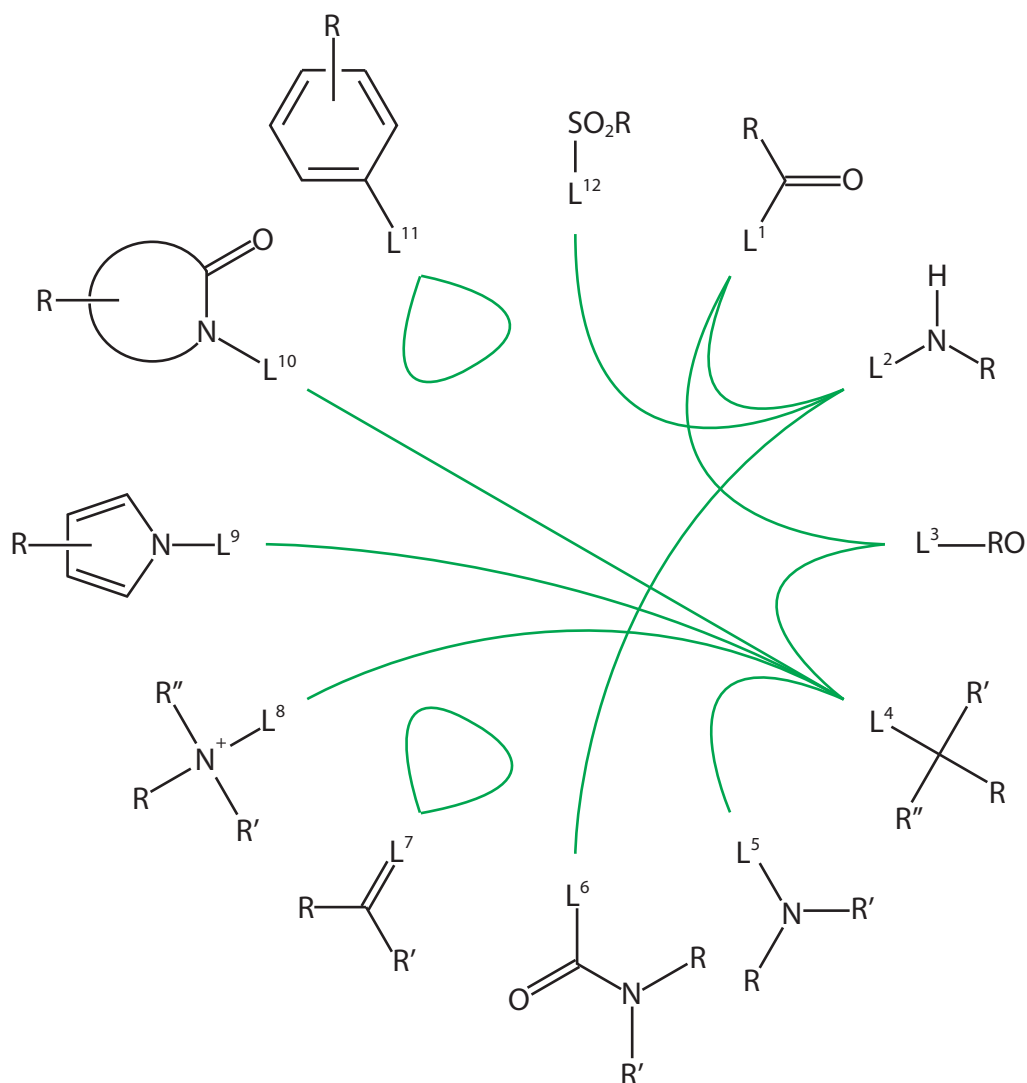


Abbildung 4.4: FTREE-FS Fragmentraum[11].

Zu sehen sind 12 chemische Umgebungen und der zugehörige Link-Typ. R, R' und R'' stehen für Fragmentreste, welche weitere Links enthalten können. Die grünen Linien symbolisieren die Kompatibilitäten der Link-Typen. Zum Beispiel ist der Link-Typ L<sup>1</sup> mit dem Link-Typ L<sup>2</sup> kompatibel. Diese zwei Link-Typen können somit verwendet werden, um die dazugehörigen Fragmente zu einem neuen Fragment zu verschmelzen. Terminale Fragmente sind nicht abgebildet.



## 4.3 Atome, Moleküle und Fragmente als formale Strukturen

Die chemische Beschreibung von Molekülen im letzten Kapitel legt nahe, diese als ungerichtete Multi-Graphen zu modellieren, wobei Atome die Knoten und (Mehrfach-) Bindungen die Kanten darstellen. Es stellt sich aber schon nach kurzer Zeit heraus, dass Multi-Graphen nicht geeignet sind, da sich zum Beispiel aromatische Bindungen damit nicht sinnvoll modellieren lassen. Moleküle werden vielmehr als einfache, ungerichtete Graphen definiert.

### Definition 4.3.1 (Molekül):

*Ein Molekül  $M \in \mathcal{M}$  ist ein zusammenhängender Graph, dessen Knotenmenge  $\mathcal{A}$  die Menge der Atome und dessen Kantenmenge  $\mathcal{B}$  die Menge von Bindungen ist.*

$$M = (\mathcal{A}, \mathcal{B})$$

*Die Menge aller Moleküle wird mit  $\mathcal{M}$  bezeichnet.*

Zur Beschreibung der hier vorgestellten Konzepte und Algorithmen sind nur die physikochemischen Eigenschaften von Atomen, Bindungen und Molekülen notwendig. Es wird daher vorausgesetzt, dass jede verwendete Eigenschaft einen eindeutigen Namen besitzt und eine Funktion  $f_p$ , die sie berechnen oder annähern kann. Die Eigenschaftsmenge wird mit  $\mathcal{E}$  bezeichnet und enthält folgende Submengen:

- Atom-Eigenschaften:  $\mathcal{E}_A \subseteq \mathcal{E}$ .
- Bindungs-Eigenschaften:  $\mathcal{E}_B \subseteq \mathcal{E}$ .
- Molekül-Eigenschaften:  $\mathcal{E}_M \subseteq \mathcal{E}$ .

Physikochemische Eigenschaften lassen sich weiterhin in folgende Mengen unterteilen:

- Additiv:  $\mathcal{E}_+ \subseteq \mathcal{E}$ .
- Nicht additiv:  $\mathcal{E}_N \subseteq \mathcal{E}$ .

Es ist offensichtlich, dass  $\mathcal{E}_+ \cap \mathcal{E}_N = \emptyset$ ,  $\mathcal{E}_+ \cup \mathcal{E}_N = \mathcal{E}$ . Additive Attribute besitzen den Vorteil, dass sie nicht aufwendig neu berechnet werden müssen, wenn zwei oder mehrere Atome oder Fragmente zu einem Molekül verbunden werden. Die Berechnung beschränkt sich dann auf das Aufsummieren der Eigenschaften der Edukte. Bei nicht additiven Attributen kann keine Vorhersage gemacht werden wie sich dieses Attribut entwickelt, wenn zwei oder mehrere Elemente verbunden werden.

Um Fragmente einführen zu können, fehlt noch das Konstrukt eines Link Atoms. Wie in Abschnitt 4.2.1 beschrieben, sind Link-Atome artifizielle Platzhalteratome, die außer einem Typ keine weiteren Eigenschaften besitzen.

**Definition 4.3.2 (Link-Atom und Link-Typ):**

Ein Link-Atom  $l \in \mathcal{L}$  repräsentiert ein Pseudo-Atom und besitzt genau einen Link-Typ  $t \in \mathcal{I}$ .

Die Bindungswertigkeit eines Link-Atoms wird durch den zugehörigen Link-Typ festgelegt und ist die einzige physikochemische Eigenschaft, die ein Link-Atom besitzt.

Die Funktionen:

$$\begin{aligned} \text{linkTyp} : \mathcal{L} &\rightarrow \mathcal{I} \\ \text{linkTypen} : 2^{\mathcal{L}} &\rightarrow 2^{\mathcal{I}} \end{aligned} \tag{4.1}$$

extrahieren den Link-Typ eines Link-Atoms beziehungsweise die Menge der Link-Typen aus einer Menge von Link-Atomen.

Manchmal kann es nützlich sein, Link-Atomen gewisse Eigenschaften zuzuweisen. So kann es zum Beispiel sinnvoll sein, Link-Atomen ein Volumen zuzuweisen, falls das Volumen eines Fragments berechnet beziehungsweise abgeschätzt werden soll. Dies sind aber immer an ein gegebenes Problem angepasste Modifikationen.

Aufbauend auf Molekülen werden Fragmente ebenfalls als Graphen modelliert, wobei die Menge der Atome um die Menge der *Link-Atome*  $\mathcal{L}$  erweitert wird.

**Definition 4.3.3 (Fragment):**

Ein Fragment  $F \in \mathcal{F}$  ist ein Molekül, dessen Knotenmenge um die Menge der Link-Atome erweitert wurde.

$$F = (\mathcal{A}_{\mathcal{L}}, \mathcal{B})$$

Wobei  $\mathcal{A}_{\mathcal{L}} = \mathcal{A} \cup \mathcal{L}$  und  $\mathcal{B}$  wieder die Menge der Bindungen beschreibt.

Die Funktion:

$$\text{linkAtome} : F \rightarrow 2^{\mathcal{L}} \tag{4.2}$$

extrahiert die Link-Atome eines Fragments.

Es sei noch angemerkt, dass aus der Definition folgt:  $M \subseteq F$ . Dies bedeutet, dass in dieser Arbeit ein Molekül ein Fragment ohne Link-Atome ist.

## 4.4 Fragmentraum

Mit Fragmenten können nun Fragmenträume definiert werden. Ein Fragmentraum besteht aus einer Menge von Fragmenten und einer Menge von Funktionen, die definieren, wie zwei Link-Atom-Typen verknüpft werden können. Die Menge der Funktionen beinhaltet auch Terminierungsfunktionen mit entsprechenden Terminierungsfragmenten. Terminierungsfunktionen unterscheiden sich praktisch nicht von normalen Verknüpfungsfunktionen. Der einzige Unterschied besteht darin, dass bei Terminierungsfunktionen das terminale Fragment konstant ist. Terminale Fragmente unterscheiden sich, außer in ihrer Auszeichnung als solche, nicht von Fragmenten in der Fragmentmenge,

sie können sogar Bestandteil dieser sein. Somit unterscheiden sich Terminierungsfunktionen faktisch nicht von Verknüpfungsfunktionen. Sie werden daher im Folgenden unter den Verknüpfungsfunktionen subsumiert und nur falls notwendig, durch eine explizite Unterscheidung kenntlich gemacht. Verknüpfungsfunktionen sind Link-Typ selektiv, das heißt, eine Verknüpfung kann nicht auf beliebige Link-Typen angewendet werden und nicht alle Link-Typen sind kompatibel zueinander. Genau genommen existiert für je zwei kompatible Link-Typen eine separate Funktion. Diese Struktur, eine Menge von Elementen und eine Menge von Funktionen auf diesen, legt nahe, Fragmenträume als Algebra zu definieren. Die Selektivität der Funktionen macht es erforderlich diese als partielle Algebra zu definieren.

**Definition 4.4.1 (Fragmentraum):**

*Ein Fragmentraum ist eine partielle Algebra  $FR = (\mathcal{F}, \mathcal{R})$ , wobei  $\mathcal{F}$  eine nicht leere Menge von Fragmenten und  $\mathcal{R}$  eine Menge von partiellen Funktionen auf  $\mathcal{F}$  ist. Jedes Fragment  $F \in \mathcal{F}$  und jedes Link-Atom  $l \in \mathcal{L}$  eines Fragmentraums hat eine Nummer  $id(F)$ ,  $id(l)$ , die bezüglich des zugrundeliegenden Fragmentraums  $FR$  eindeutig ist.*

Die Elemente der Initialmenge  $\mathcal{F}$  werden als *atomare Fragmente* bezeichnet. Die Menge der partiellen Funktionen wird im Folgenden als Verknüpfungs- oder Regelmengemenge bezeichnet, ein Element der Menge entsprechend Verknüpfung oder *Regel*. Eine Regel verbindet immer zwei Fragmente zu einem neuen Fragment, wobei die Ausgangsfragmente nicht verändert werden. Anders ausgedrückt, ein nicht atomares Fragment besteht nur aus Kopien von atomaren Fragmenten. Falls die Operationen auf Fragmenten nicht diese Eigenschaft hätten, würde eine Verknüpfung zwischen zwei atomaren Fragmenten die beiden beteiligten Link-Atome für alle zukünftigen Verknüpfungen blockieren. Da jedes Link-Atom eine eindeutige Nummer besitzt, würde es genügen, zwei Link-Atome für die Verknüpfungsfunktion anzugeben. Aus Gründen der Lesbarkeit werden noch die beiden beteiligten Fragmente angegeben. Die Syntax ist wie folgt:

$$\begin{aligned} r : \mathcal{F} \times \mathcal{L} \times \mathcal{F} \times \mathcal{L} &\rightarrow \mathcal{F} \\ r_{o,p}(F_k, l_i^o, F_m, l_j^p) &= F_n \quad F_k, F_m \in \mathcal{F}, \quad l_i^o, l_j^p \in \mathcal{L} \end{aligned} \quad (4.3)$$

$r_{op} \in \mathcal{R}$  ist eine Regel, die die beiden Link-Typen  $o$  und  $p$  verknüpft.  $F_k$  und  $F_m$  sind die beiden zu verknüpfenden Fragmente.  $l_i^o$  und  $l_j^p$  sind die beteiligten Link-Atome mit  $o, p$  als Link-Typen und  $i, j$  als Indizes der Link-Atome.

Wie bereits erwähnt, werden in dieser Arbeit Moleküle als Fragmente ohne Link-Atome betrachtet. Daraus folgt, dass ein Fragmentraum abgeschlossen ist, das heißt, alle erzeugbaren Fragmente, beziehungsweise Moleküle, sind Teil des zugrundeliegenden Fragmentraums.

Folgende abkürzende Schreibweisen werden verwendet, um anzuzeigen, dass zwei Link-Typen  $o, p$  der Regelmengemenge  $\mathcal{R}$  kompatibel sind:

$$\begin{aligned} r_{o,p} &\in \mathcal{R} \\ \text{SindKompatibel}(\mathcal{R}, o, p) &= true \end{aligned} \quad (4.4)$$

Falls eindeutig ist, welche Regelmenge  $\mathcal{R}$  zugrunde liegt, kann bei der zweiten, funktionalen, Schreibweise die Regelmenge weggelassen werden.

### Eigenschaften von Fragmenträumen

Zusätzliche Informationen beziehungsweise Anpassungen, wie im Kapitel 4.2 beschrieben, obwohl chemisch relevant, werden hier nicht explizit betrachtet, da es hier primär um die strukturelle Beschreibung von chemischen Fragmenträumen geht. Es wird vielmehr davon ausgegangen, dass alle nötigen Anpassungen implizit vorgenommen werden.

Link-Typ Kompatibilitäten sind in der Regel weder reflexiv noch transitiv. Das heißt, nicht immer ist  $r_{o,o} \notin \mathcal{R}$  und falls  $r_{o,p} \in \mathcal{R} \wedge r_{p,q} \in \mathcal{R}$  folgt daraus nicht zwangsläufig  $r_{o,q} \in \mathcal{R}$ . Regeln sind aber symmetrisch:  $r_{o,p} \in \mathcal{R} \Leftrightarrow r_{p,o} \in \mathcal{R}$ . Abhängig von den Regeln, können Fragmenträume unendlich sein, das heißt, sie können eine unendliche Anzahl von Fragmenten enthalten. Ferner können Fragmenträume zusammenhängend sein, oder in Komponenten zerfallen. Besteht ein Fragmentraum aus  $n$  Komponenten, gibt es mindestens  $n$  Fragmente, die gemäß den Regeln nicht in einem zusammengesetzten Fragment vorkommen können.

### Kompatibilitätsklassen und Kompatibilitätsmengen

Als letzte Eigenschaften von Fragmenträumen werden noch die Kompatibilitätsklasse und Kompatibilitätsmenge eines Link-Typs eingeführt.

#### Definition 4.4.2 (Kompatibilitätsklasse):

Sei  $\mathcal{R}$  eine Regelmenge eines Fragmentraums und  $t \in \mathcal{I}_{\mathcal{R}}$  ein Link-Typ aus  $\mathcal{R}$ . Die Kompatibilitätsklasse  $\mathcal{C}$  des Link-Typs  $t$  ist die Menge aller kompatiblen Link-Typen:

$$\mathcal{C}_t = \{i \mid r_{i,t} \in \mathcal{R}\} \subseteq \mathcal{I}_{\mathcal{R}}$$

Da Regeln meistens weder transitiv noch reflexiv sind, dürfen Kompatibilitätsklassen nicht mit Äquivalenzklassen verwechselt werden. Das bedeutet:  $i \in \mathcal{C}_t \not\Rightarrow \mathcal{C}_i = \mathcal{C}_t$ .

#### Definition 4.4.3 (Kompatibilitätsmenge):

Die Kompatibilitätsmenge  $\mathcal{K}_t$  eines Link-Typs  $t \in \mathcal{I}_{\mathcal{R}}$  in einem Fragmentraum  $FR = (\mathcal{F}, \mathcal{R})$  ist die Menge aller Fragmente  $F$  des Fragmentraums  $FR$ , die mindestens einen kompatiblen Link-Typ zu  $t$  haben:

$$\mathcal{K}_t = \{F \mid i \in \text{linkTypen}(\text{linkAtome}(F)) \wedge r_{i,t} \in \mathcal{R}\}$$

### 4.4.1 Fragmentbaum

Um das systematische Zusammensetzen von Fragmenten und Molekülen aus Fragmenträumen zu ermöglichen, wurde die Datenstruktur eines Fragmentbaums entwickelt.

Ein Fragmentbaum ist wie ein normaler Baum ein gerichteter, zusammenhängender und azyklischer Graph. Jeder Knoten eines Fragmentbaums besteht aus einem atomaren Fragment. Die Kanten speichern Informationen, über welche Link-Atome zwei Fragmentknoten verbunden sind. Würden Kanten nicht die einzelnen Link-Atome, sondern nur ihre Kompatibilitäten berücksichtigen, könnte nicht immer eindeutig ein Link-Atom zum Eltern- und Kinderknoten zugeordnet werden, da jeweils mehr als eine Kompatibilität existieren kann. Wenn ein Fragment, bildlich gesprochen, durch die Kanten durchrotieren könnte, hätte das zwar keinen Einfluss auf die Topologie des Fragmentbaums, sehr wohl aber auf das physikochemische Profil des korrespondierenden Moleküls. Aus Gründen, die später ersichtlich werden, werden die Kanten in einer geordneten Menge verwaltet, deren Ordnungsrelation den Einfügezeitpunkt eines Knotens widerspiegelt. Die geordnete Menge der Kanten spiegelt somit die Chronologie des Baums wieder.

Fragmentbäume werden analog zur obigen Definition von Fragmenträumen definiert, das heißt, es wird bei den Kanten das Link-Atom und das zugehörige Fragment mit angegeben. Die Regeln berücksichtigen die Eltern-Kind-Beziehung zwischen zwei Knoten. Das erste (Fragment, Link-Atom) Paar bezieht sich immer auf das Eltern-, das Zweite auf das Kind-Fragment.

**Definition 4.4.4 (Fragmentbaum):**

Für ein Fragmentbaum  $FB_{FR} = (\mathcal{F}_{FB}, \mathcal{R}_{FB})$  eines Fragmentraums  $FR = (\mathcal{F}, \mathcal{R})$  mit  $\mathcal{F}_{FB} \subseteq \mathcal{F}$ ,  $|\mathcal{F}_{FB}| = m$ ,  $\mathcal{R}_{FB} \subseteq \mathcal{R}$ ,  $|\mathcal{R}_{FB}| = n$ , gilt:

- Zwischen je zwei Knoten von  $FB$  gibt es maximal eine Kante.
- $FB$  ist zusammenhängend und azyklisch.
- Es gilt  $m = n - 1$ .
- Das erste  $(\mathcal{F}_{FB}, \mathcal{L})$  Paar einer Regel  $R \in \mathcal{R}_{FB}$  repräsentiert den Eltern-Knoten, das Zweite das Kind-Fragment.

Die Menge aller Fragmentbäume zu einem Fragmentraum  $FR$  wird als  $\mathcal{FB}_{FR}$  bezeichnet:  $\mathcal{FB}_{FR} = \bigcup FB_{FR}$ .

Der Index  $FB$  bei  $\mathcal{F}_{FB}$  und  $\mathcal{R}_{FB}$  zum Identifizieren der Fragmentbaum-Submenge von Fragmenten und Regel-Tupel werden im Folgenden implizit vorausgesetzt und nicht mehr angegeben.

Fragmentbäume ermöglichen auf einfache und systematische Weise alle Moleküle eines Fragmentraums zu enumerieren. Das nächste Kapitel wird im Detail beschreiben, wie eine Enumerierung mithilfe eines rekursiven Algorithmus realisiert werden kann. Ein weiterer Vorteil von Fragmentbäumen ist eine indirekte Darstellung eines Moleküls. Für einige Anwendungen, wie zum Beispiel die Berechnung von bestimmten Eigenschaften, ist es nicht erforderlich ein voll initialisiertes Molekül zu besitzen. Wie noch später dargelegt wird, ist die Initialisierung eines Moleküls, das heißt, die Berechnung von verschiedenen Eigenschaften der Atome, Bindungen und des

gesamten Moleküls, sehr Rechenzeit aufwendig. Verbesserungen bei Algorithmen und Hardware haben diese Zeit zwar drastisch reduziert, so dass sie bei der Berechnung eines Moleküls nicht mehr ins Gewicht fällt, sollen aber mehrere Millionen Moleküle verarbeitet werden, macht sich jede nicht notwendige Berechnung sehr bemerkbar. Als Beispiel seien hier zweidimensionale (2D)-Deskriptoren genannt, für die es nicht notwendig ist, die 3D-Koordinaten der Atome zu kennen.

Es lässt sich aber jetzt schon relativ einfach ein Problem im Zusammenhang einer Enumerierung von Fragmentbäumen erkennen. Dem gleichen Molekül können verschiedene Fragmentbäume zugewiesen werden (siehe Abbildung 5.6). Eine einfach zu erkennende Redundanz kann erzeugt werden, indem ein Baum in einen Graphen überführt wird. Im Wesentlichen bedeutet dies, die Unterscheidung zwischen Eltern- und Kinderkanten aufzuheben und keinem Knoten mehr die Eigenschaft Wurzel zuzuweisen. Wird nun dieser Graph wieder in einen Baum überführt, kann jeder Knoten als Wurzel ausgewählt werden. Egal welcher Knoten als Wurzel ausgewählt wird, das repräsentierte Molekül bleibt immer dasselbe, da die Graphtopologie nicht verändert wurde, es wurde nur einem anderen Knoten die Wurzeleigenschaft zugewiesen. Damit ergeben sich schon pro Molekül  $|F| - 1$  redundante Bäume. Bei genauerer Betrachtung können leicht noch andere Redundanzen gefunden werden. Es stellt sich somit prinzipiell die Frage, ob sich diese Redundanzen vermeiden lassen oder ob es nicht sinnvoller ist, nach anderen Strukturen für die Enumerierung zu suchen. Wie das folgende Kapitel darstellen wird, lässt sich eine einfache, elegante und doch sehr effektive Methode angeben, um die meisten strukturellen Redundanzen zu vermeiden.

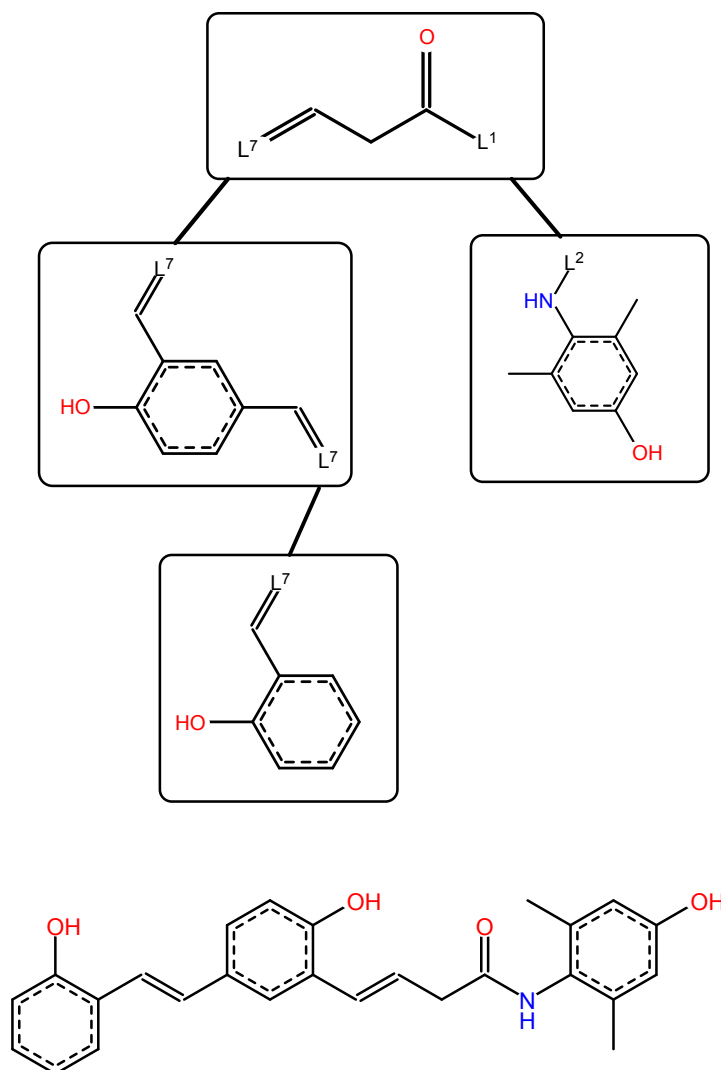


Abbildung 4.5: Fragmentbaum und korrespondierendes Molekül.

Jeder Knoten des Fragmentbaums repräsentiert ein atomares Fragment. Die Kanten speichern nicht nur die beteiligten Knoten, sondern auch über welche Link-Atome zwei Fragmente verbunden sind. Würde diese Information nicht in den Kanten gespeichert werden, wäre das resultierende Molekül nicht immer eindeutig. Am obigen Beispiel kann das am linken Kind der Wurzel veranschaulicht werden. Das Fragment hat zwei Link-Atome mit demselben Typ. Das Fragment könnte somit im Knoten "rotieren" ohne die Kompatibilitäten zu verletzen. Die Topologie des Baums würde sich nicht ändern, sehr wohl aber das resultierende Molekül.

#### 4.4.2 Initialisierung von Fragmenten und Molekülen

Nachdem ein Fragmentraum eingelesen wurde, stehen, abhängig vom Dateiformat, nur wenige Informationen über die Fragmente des Fragmentraums zur Verfügung. Im Wesentlichen sind nur die Atom-Typen und ihre Bindungen, also der Molekülgraph, bekannt. Verschiedene Dateiformate unterstützten zwar unterschiedliche Mengen von Annotationen, die über den Molekülgraph hinausgehen, aber im Wesentlichen müssen noch verschiedene physikochemische Eigenschaften berechnet werden, um sinnvoll mit Fragmenten arbeiten zu können. In einem ersten Schritt muss daher jedes Fragment zunächst initialisiert werden, das heißt, es müssen alle später benötigten Informationen zu jedem Fragment berechnet werden. Das Berechnen dieser kann je nach Eigenschaft sehr aufwendig sein. Für einzelne Fragmente mag das nicht ins Gewicht fallen, aber bei der Initialisierung mehrerer tausender Fragmente kann dies schon einige Zeit beanspruchen. Die Situation verschärft sich noch während der Enumeration von Fragmenträumen, da dann jedes erzeugte Molekül und potentiell alle Zwischenergebnisse initialisiert werden müssen. Es wäre daher sinnvoll nur die Informationen zu berechnen, die während den verschiedenen Schritten der Enumeration benötigt werden. Da während der Enumeration viele Fragmente ausgeschlossen werden können, da sie nicht gewünschten Kriterien entsprechen, ist es auch sinnvoll eine *Lazy-Initialisierung* einzuführen, also aufwendige Berechnungen erst dann durchzuführen, wenn sie wirklich gebraucht werden.

In dieser Arbeit wird auf das Initialisierungskonzept von FLEXNOVO zurückgegriffen[12]. FLEXNOVO bedient sich der Initialisierungsfunktionen von FLEXX, unterteilt sie aber in drei disjunkte, aufeinander bauende Phasen (siehe Tabelle 4.1). Bevor Berechnungen an einem Fragment durchgeführt werden, wird es zunächst auf Konsistenz überprüft, das heißt, es wird überprüft, ob das Fragment zusammenhängend ist und, ob die 3D Informationen der Atome plausibel sind. In der ersten Phase werden relativ einfach zu bestimmende, aber wichtige physikochemische Eigenschaften berechnet und annotiert, wie zum Beispiel rotierbare Bindungen oder Ringsysteme. Die zweite Phase berechnet die meisten physikochemischen Eigenschaften eines Fragments. Die dritte Phase stellt die volle Initialisierung dar und berechnet im Wesentlichen alle Eigenschaften die FLEXX zur Verfügung stellt. Informationen, berechnet in der dritten Phase, sind hauptsächlich für das Docken von Fragmenten nötig und sind für diese Arbeit nicht relevant. Wie eingangs erwähnt, bauen die Phasen aufeinander auf. Um Phase  $n$  zu berechnen, muss das Fragment sich in Initialisierungsphase  $n - 1$  befinden.

Link-Atome stellen künstliche Platzhalter dar und müssen somit bei der Initialisierung gesondert betrachtet werden. Obwohl es nur Pseudo-Atome sind, wird ihnen das Volumen von Wasserstoff-Atomen zugewiesen. Weiterhin müssen noch Eigenschaften berücksichtigt werden, die für das Docken von Fragmenten relevant sind. Da diese Eigenschaften nicht für diese Arbeit relevant sind, wird auch auf diese hier nicht weiter eingegangen. Wichtiger für diese Arbeit ist die Zuweisung von Eigenschaften für die Bindung zum Link-Atom. Eine wichtige Eigenschaft von Molekülen, die zur Enumerierung von Fragmenträumen herangezogen werden kann, ist die Anzahl der



rotierbaren Bindungen. Aus diesem Grund ist eine korrekte Zuweisung dieser Eigenschaft erstrebenswert, falls die Regeln des Fragmentraums dies zulassen. Im Prinzip ist die Eigenschaft von rotierbaren Bindungen strikt additiv. In Fragmenträumen verkompliziert sich die Lage etwas, da Regeln Bindungseigenschaften zwischen zwei Fragmenten ändern können, wenn diese verschmolzen werden. Aus diesem Grund muss für jeden Fragmentraum festgelegt werden, ob rotierbare Bindungen additiv sind oder nicht.

Die Aufteilung der Rechenzeit auf die einzelnen Phasen entspricht etwa: 10%, 40% und 50%. Im Vergleich zu einer normalen FLEXX Initialisierung können somit, abhängig von den Enumerierungseigenschaften, 50% bis 90% der Initialisierungszeit eingespart werden. Da die Initialisierung die rechenzeitaufwendigste Funktion darstellt, hat diese Phasen getriebene Initialisierung offensichtlich einen sehr positiven Einfluss auf die Reduzierung der Laufzeit.

Phase	Überprüfungen/Initialisierungen
0. Phase	Konsistenz des Fragmentgraphens und der 3D-Struktur
1. Phase	Einfache Atom- und Bindungseigenschaften Identifizierung von Ringen und Ringsystemen Überprüfung und Annotation der SYBYL-Atom- und Bindungstypen Überprüfung und Annotation der Hybridisierungszustände Identifizierung aromatischer Systeme Bestimmung der Formalladungen Delokalisierung aromatischer Systeme Identifikation rotierbarer Bindungen
2. Phase	Grundlegende Fragment-Informationen Bestimmung der Kraftfeld-Parameter (optional) Berechnung der unifizierten Atomradien Aufstellung der lokalen Koordinatensysteme Berechnung der molekularen Hydrophobizität Identifizierung der PMF-Atom-Typen (optional) Berechnung der Stereodeskriptoren Identifizierung der Symmetrie-Klassen Zuordnung der Wechselwirkungstypen Zuweisung der Atominkremente für die log P Berechnung Generierung der diskreten Wechselwirkungspunkte (optional)
3. Phase	Vollständige Molekül-Informationen Berechnung von Ringkonformationen Annotierung der Torsionswinkel-Daten Aufstellen des Komponentenbaums Abschätzung der Wechselwirkungsenergien der einzelnen Komponenten

Tabelle 4.1: Initialisierungsstufen mit verbundenen Berechnungen.

Die unterschiedlichen Phasen sind bezüglich ihrer Berechnungen disjunkt, bauen aber aufeinander auf. Zur Berechnung von Phase  $n$  wird Phase  $n - 1$  benötigt.  
Tabelle aus[12].

## 5 Navigieren von Fragmenträumen

In diesem Kapitel werden zwei Methoden und zugehörige Programme zum Navigieren von Fragmenträumen vorgestellt. Der erste Ansatz ermöglicht es, mithilfe des Programms FRAGVIEW, Fragmenträume visuell zu explorieren. Als zweites wird ein Algorithmus vorgestellt, der es ermöglicht Fragmenträume eigenschaftsbasiert zu enumerieren, das heißt, alle Moleküle eines Fragmentraums aufzuzählen, die ein bestimmtes, vom Benutzer vorgegebenes physikochemisches Profil erfüllen.

### 5.1 Visuelle Navigation

Wie in Kapitel 4.2.3 beschrieben, können Fragmenträume durch das Definieren von Schneideregeln und deren Anwendung auf Moleküle generiert werden. Weiterhin können Filter definiert werden, um unerwünschte Fragmente schon beim Schneiden von Molekülen zu verwerfen. Es ist aber dennoch wünschenswert, die generierten Fragmente einer genaueren Inspektion zu unterwerfen. Bei einer visuellen Inspektion wäre es weiterhin vorteilhaft, wenn einige einzelne unerwünschte Fragmente entfernt werden könnten. Bei einer Inspektion könnten auch Fehler in den Schneideregeln entdeckt werden und, falls ein Zugriff auf Regeln und Ausgangsmoleküle vorhanden ist, diese verbessert werden. Weiterhin wäre es wünschenswert, die Regeln eines Fragmentraums genauer inspizieren zu können und diese gegebenenfalls eigenen Wünschen anzupassen. Auch wäre es vorteilhaft, manuell Fragmente zu neuen Fragmenten oder zu Molekülen zu verknüpfen. Für all diese beschriebenen Abläufe wurde das Programm FRAGVIEW entwickelt.

Das Programm besitzt drei Abstraktionsebenen: Fragmentraum, Tabelle und Fragment, wobei die Strukturen aufeinander aufbauen. Ein Fragmentraum kann aus mehreren Tabellen bestehen und eine Tabelle wiederum verwaltet ein oder mehrere Fragmente. Jede Tabelle repräsentiert eine Selektion von Fragmenten. Die Selektion kann beim Laden stattfinden oder während der Programmbenutzung. Jede Tabelle besitzt eine Kopie der Regeln des zugrundeliegenden Fragmentraums. Somit ist es möglich, die Regeln für jede Tabelle einzeln zu ändern. Entsprechend der drei Strukturen sind auch die Funktionen gegliedert. Funktionen können auf einem Fragmentraum, einer Tabelle oder einem Fragment durchgeführt werden. Soll eine Funktion auf einem Fragment durchgeführt werden, muss das Kontextmenü für dieses Fragment aufgerufen werden. Das Fragment muss dazu nicht vom Benutzer durch anklicken aktiviert worden sein. Wird auf einer Tabelle eine Funktion durchgeführt,

sind nur alle aktivierten, das heißt, vom Benutzer angeklickten Fragmente, betroffen. Gleiches gilt für Funktionen auf einem Fragmentraum. FRAGVIEW kann mehrere Tabellen gleichzeitig verwalten, wobei diese nicht vom selben Fragmentraum stammen müssen. Um dennoch Funktionen zu ermöglichen, die auf allen Tabellen eines Fragmentraums ausgeführt werden können, betrachtet FRAGVIEW alle Tabellen mit denselben Regeln als Teil desselben Fragmentraums. Das bedeutet aber auch, dass falls zufällig oder beabsichtigt zwei verschiedene Fragmenträume denselben Regelsatz besitzen, FRAGVIEW sie als Teil desselben Fragmentraums betrachtet. Zwei Regelsätze sind für FRAGVIEW gleich falls sie:

- die gleiche Anzahl von Regeln besitzen.
- die Mengen der Link-Namen gleich sind,
- die gleichen Link-Typen kompatibel sind,
- die Anpassungen für eine Verknüpfung zwischen jeweils zwei kompatiblen Link-Typen gleich sind.

Nach dem Starten des Programms und der Auswahl eines Fragmentraums kann mittels logischer Ausdrücke über physikochemische Eigenschaften festgelegt werden, welche Fragmente dargestellt werden sollen (siehe Abbildung 5.1). Die einzelnen physikochemischen sind in Tabelle 5.1 aufgelistet.

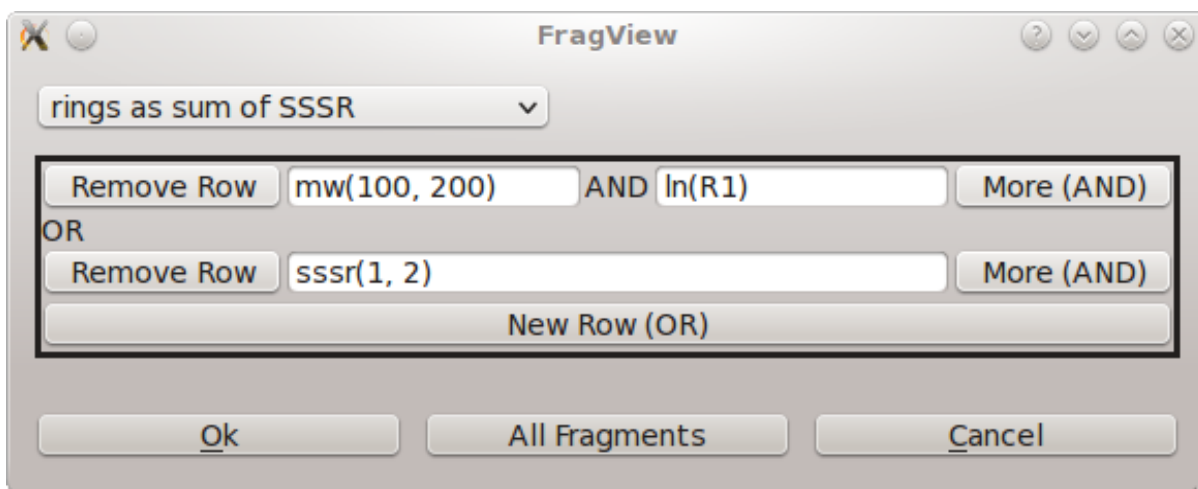


Abbildung 5.1: Kriterien für darzustellende Fragmente.

Eigenschaften in einer Zeile werden logisch UND verbunden, Zeilen werden logisch ODER verbunden. Im obigen Beispiel werden alle Fragmente selektiert, die ein Molekulargewicht von 100 bis 200 UND den Link-Typ R1 besitzen ODER ein bis zwei Ringe. Links oben können Eigenschaften für die Eingabe ausgewählt werden. Es können beliebig viele Eigenschaften logisch verbunden werden. Über **All Fragments** werden alle Fragmente selektiert.

Eigenschaften die zur Selektion verwendet werden können	
Anz. Link Atome	Anz. Link-Typen
Molekulares Gewicht	Anz. nicht Kohlenstoff-Atome
Anz. nicht Kohlenstoff-Bindungen	Anz. Ringe
Anz. rotierbarer Bindungen	Anz. Wasserstoff Donoren
Anz. Wasserstoff Akzeptoren	Anz. hydrophobe Interaktionen
Anz. Wasserstoffbrücken + Salzbrücken	EZ Stereo Zentren
Anz. RS Stereo Zentren	Star Atoms
$\log p$	Polare Oberfläche
Inclusion SMARTS	Exclusion SMARTS

Weitere, nur anzeigbare Eigenschaften	
Name	Unique SMILES

Tabelle 5.1: Physikochemische Eigenschaften verwendbar in FRAGVIEW.

Die Eingabe erfolgt dabei über eine an Funktionen angelehnte Syntax: Der Eigenschaftsname gefolgt von Parametern in einer Klammer. Akzeptiert die Eigenschaft ein Intervall, werden zwei Parameter erwartet, das Minimum und das Maximum. Andere Eigenschaften, wie zum Beispiel Link-Typ und Inclusion- beziehungsweise Exclusion SMARTS[86], akzeptieren beliebig viele, durch Komma getrennte Parameter. Für die Berechnung der einzelnen physikochemischen Eigenschaften wird auf die FLEXNOVO[12] Programm-Bibliothek zurückgegriffen, welches wiederum auf FLEXX[23] zurückgreift.

Nachdem Kriterien festgelegt wurden, werden die selektierten Fragmente in einer an Tabellenkalkulationen angelehnten Form angezeigt. In der Standarddarstellung werden nur Strukturdiagramme von Fragmenten untereinander angezeigt. Zur Visualisierung von Fragmenten wird die SDG Bibliothek[87] verwendet. Über ein Kontextmenü können alle in Tabelle 5.1 aufgelisteten Eigenschaften eingeblendet werden.

Um einen schnellen Zugriff auf die Fragmente eines Fragmentraums zu gewährleisten, wurde zunächst eine Version entwickelt, welche alle selektierten Fragmente in den Hauptspeicher lädt. Aus diesem Grund ist die Anzahl der darstellbaren Fragmente pro Fragmentraum auf 25.000 beschränkt. FRAGVIEW wurde aber so entwickelt, dass es sehr einfach ist, Fragmente zum Beispiel in einer Datenbank zu verwalten und somit Probleme im Zusammenhang mit zu wenig Hauptspeicher zu umgehen. Es darf aber angezweifelt werden, ob jemand wirklich mehr als 25.000 Fragmente sehen will beziehungsweise kann.

Fragmente stellen nur einen Aspekt von Fragmenträumen dar. Ein anderer wichtiger Bestandteil sind die Kompatibilitätsregeln und Terminierungsfragmente. Diese werden in FRAGVIEW in einem Regelfenster visualisiert (siehe Abbildung 5.3). Dieses Fenster ist zweigeteilt, auf der linken Seite ist eine Kompatibilitätsmatrix der Regeln dargestellt, auf der rechten Seite können terminale Fragmente festgelegt werden. Von

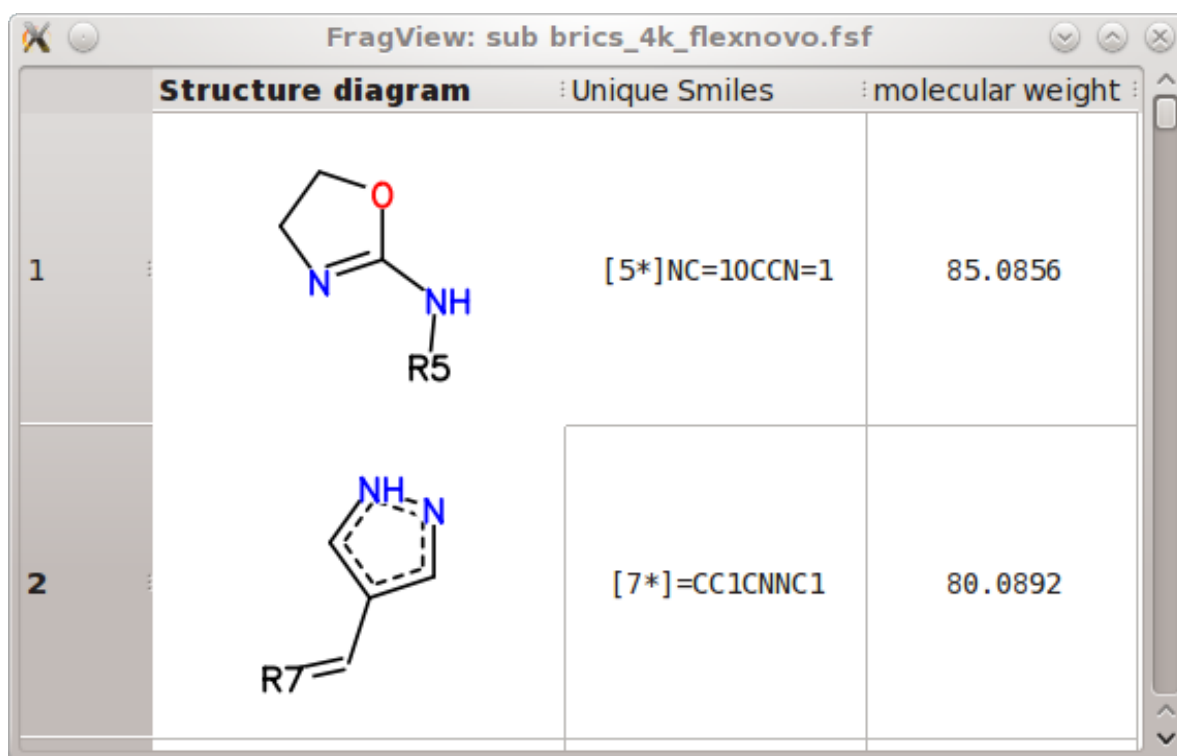


Abbildung 5.2: Fragmentraum Visualisierung.

Selektierte Fragmente werden in einer Tabellenkalkulation ähnlichen Form dargestellt. Direkt nach dem Laden werden nur die Strukturdiagramme der Fragmente dargestellt. Über ein Kontextmenü können physikochemische Eigenschaften beliebig hinzugefügt und entfernt werden. Im hier dargestellten Fall wurden Unique Smiles und molecular weight hinzugefügt. Die Fragmente lassen sich durch klicken auf physikochemische Eigenschaft gemäß dieser sortieren.

der Kompatibilitätsmatrix ist aufgrund der Symmetrie der Regeln nur die obere Dreiecksform dargestellt. Sind zwei Link-Typen kompatibel, ist dies durch ein Haken dargestellt. Durch klicken in der entsprechenden Zeile und Spalte können Kompatibilitäten de- beziehungsweise aktiviert werden. Terminale Fragmente können durch das Klicken auf den Knopf ganz links einer Regel festgelegt werden. FRAGVIEW kann terminale Fragmente aus einer Datei oder als SMILES einlesen.

Ein weiteres nennenswertes Feature von FRAGVIEW ist, dass es unter Zuhilfenahme der SDG-Bibliothek, Struktur Diagramme von Fragmenten nach Link-Typen ausrichten kann. Dies kann zum Beispiel dazu verwendet werden, um zwei Selektionen eines Fragmentraums zu laden, wobei die erste Auswahl auf einen Link-Typ reduziert wird und die Zweite auf einen Kompatiblen. Dann können beide Tabellen nebeneinander angeordnet werden und die Link-Atome in die Richtung der jeweils anderen Tabelle ausgerichtet werden. So kann visuell eine Inspektion von potentiellen Verknüpfungen vorgenommen werden (siehe Abbildung 5.4). Neben dem Ausrichten von Fragmenten,

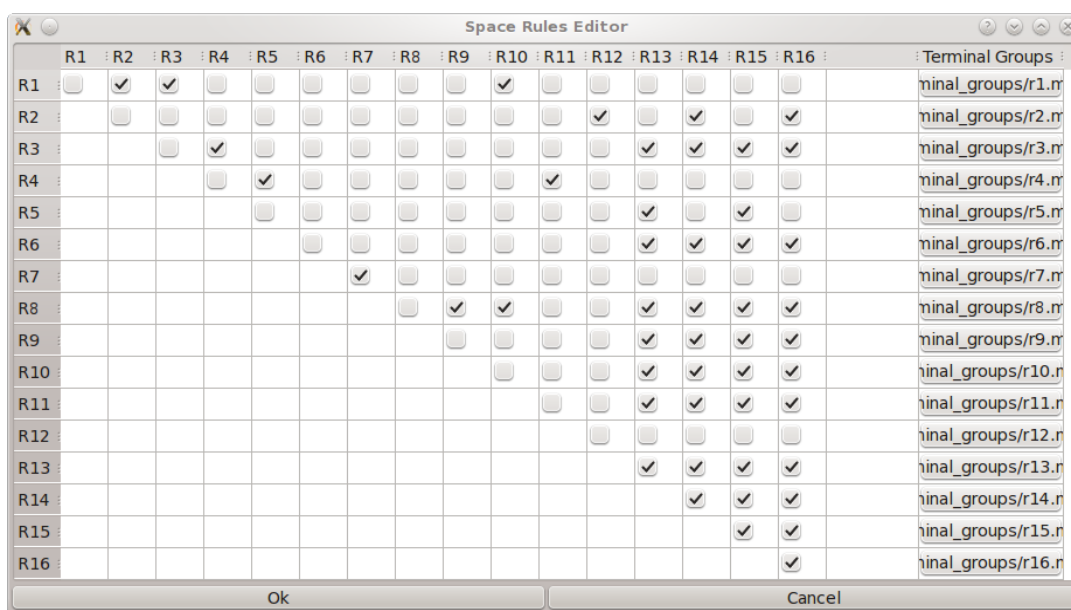


Abbildung 5.3: Regeln eines Fragmentraums.

Dieser Fragmentraum hat 16 Link-Typen, R1 bis R16. Im linken Teil ist die Kompatibilitätsmatrix der Link-Typen dargestellt. Sind zwei Link-Typen kompatibel, ist dies durch einen Haken in der entsprechenden Zeile und Spalte markiert. Durch klicken auf eine Regel kann diese de- oder aktiviert werden. Rechts können die terminalen Fragmente für jeden Link-Typ festgelegt werden. Es kann entweder eine Datei mit einem Fragment oder ein SMILES angegeben werden.

gemäß ihrer Link-Atome, können noch verschiedene andere Operationen auf einzelnen Fragmenten oder ganzen Fragmenträumen ausgeführt werden. Durch drücken der rechten Maustaste über einem Fragment erhält der Benutzer ein Kontextmenü, welches drei Untermenüs enthält: *Global*, *Table* und *Fragment*. Tabelle 5.2 listet alle Fragmentfunktionen des Fragment-Untermenüs auf.

Funktion	Beschreibung
Delete	Molekül aus der umgebenden Tabelle löschen.
Orientate	Molekül anhand eines Link-Atoms ausrichten.
Terminate	Ein oder alle Link-Atome durch terminale Gruppe substituieren.
Edit Fragment as mol2	Fragment in einem Editor im Mol2 Format öffnen.
Edit Fragment as SMILES	Fragment in einem Editor als SMILES öffnen.
Save Molecule	Fragment/Molekül im aktuellen Zustand speichern.

Tabelle 5.2: FRAGVIEW Funktionen für Fragmente.

Wird eine dieser Funktionen ausgewählt, wird die entsprechende Aktion auf dem Fragment ausgeführt, unabhängig davon, ob es aktiviert ist.

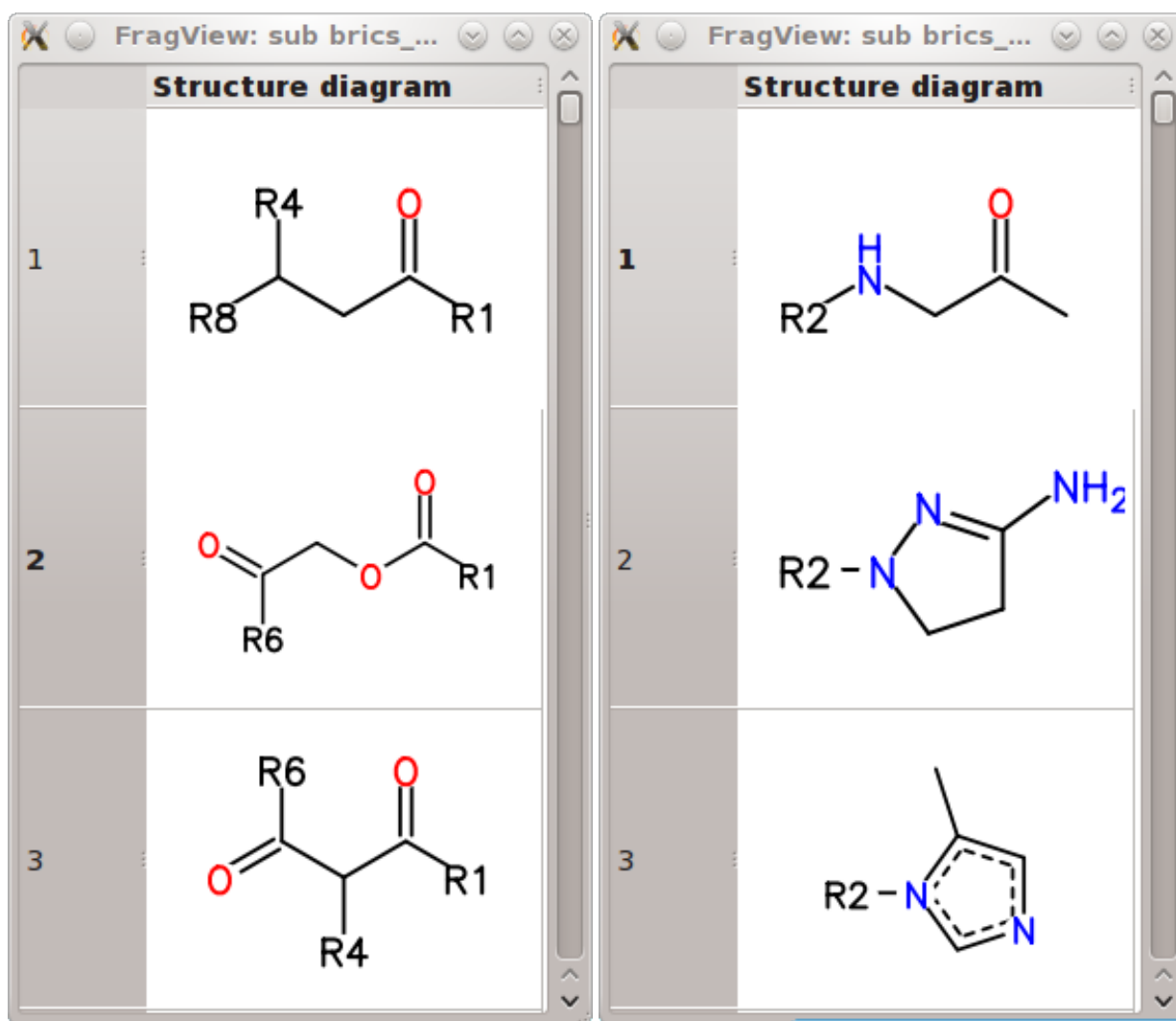


Abbildung 5.4: Ausrichten von Fragmenten.

Für die linke Tabelle wurden nur Fragmente mit dem Link-Typ R1 und für die rechte mit Link-Typ R2 ausgewählt. Dann wurden die Fragmente so ausgerichtet, dass alle Link-Atome mit dem Typ R1 der linken Tabelle nach rechts zeigen und alle Link-Atome mit dem Typ R2 der rechten Tabelle nach links zeigen. Es kann nun durch die Tabellen gescrollt werden und überprüft werden, ob sinnvolle Paarungen vorhanden sind.

Das Untermenü *Table* enthält Funktionen für eine Tabelle. Im Gegensatz zu den Funktionen für ein einzelnes Fragment, müssen Fragmente, auf welchen eine Tabellenfunktion ausgeführt werden soll, vorher aktiviert werden. Das Aktivieren erfolgt über einen Linksklick auf dem Fragment. Fragmente, die angeklickt wurden, erhalten eine blaue Umrandung. Wird ein aktiviertes Fragment nochmals angeklickt, wird es



deaktiviert und verliert die blaue Umrahmung. Die einzelnen Funktionen auf einer Tabelle sind in Tabelle 5.3 aufgelistet.

Funktion	Beschreibung
Delete	Aktivierte Fragmente aus dieser Tabelle löschen.
Join	Aktivierte Fragmente in allen möglichen Kombinationen verbinden und Ergebnisse in einer neuen Tabelle anzeigen.
Orientate	Aktivierte Fragmente anhand eines Link-Typs ausrichten.

Tabelle 5.3: FRAGVIEW Funktionen für Tabellen.

Neben Funktionen für Fragmente und Tabellen gibt es noch globale Funktionen, die auf allen Tabellen der entsprechenden FRAGVIEW Instanz ausgeführt werden (siehe Tabelle 5.4).

Funktion	Beschreibung
Join	Aktivierte Fragmente in allen kompatiblen Tabellen in allen möglichen Kombinationen verbinden und Ergebnisse in einer neuen Tabelle anzeigen.
Delete	Aktivierte Fragmente in allen Tabellen löschen.
Orientate	Aktivierte Fragmente in allen Tabellen gemäß eines Link-Typs ausrichten.
Terminate	Spezifizierten Link-Typ an allen aktivierten Fragmenten aller Tabellen terminieren.
Load new Space	Neuen Fragmentraum laden.
Quit	FRAGVIEW beenden

Tabelle 5.4: Globale FRAGVIEW Funktionen.

Alle Funktionen sind bezüglich des zugrundeliegenden Fragmentraums abgeschlossen, das heißt, alle modifizierten oder neu generierten Fragmente sind Bestandteil des Fragmentraums und können somit für nachfolgende Operationen verwendet werden.

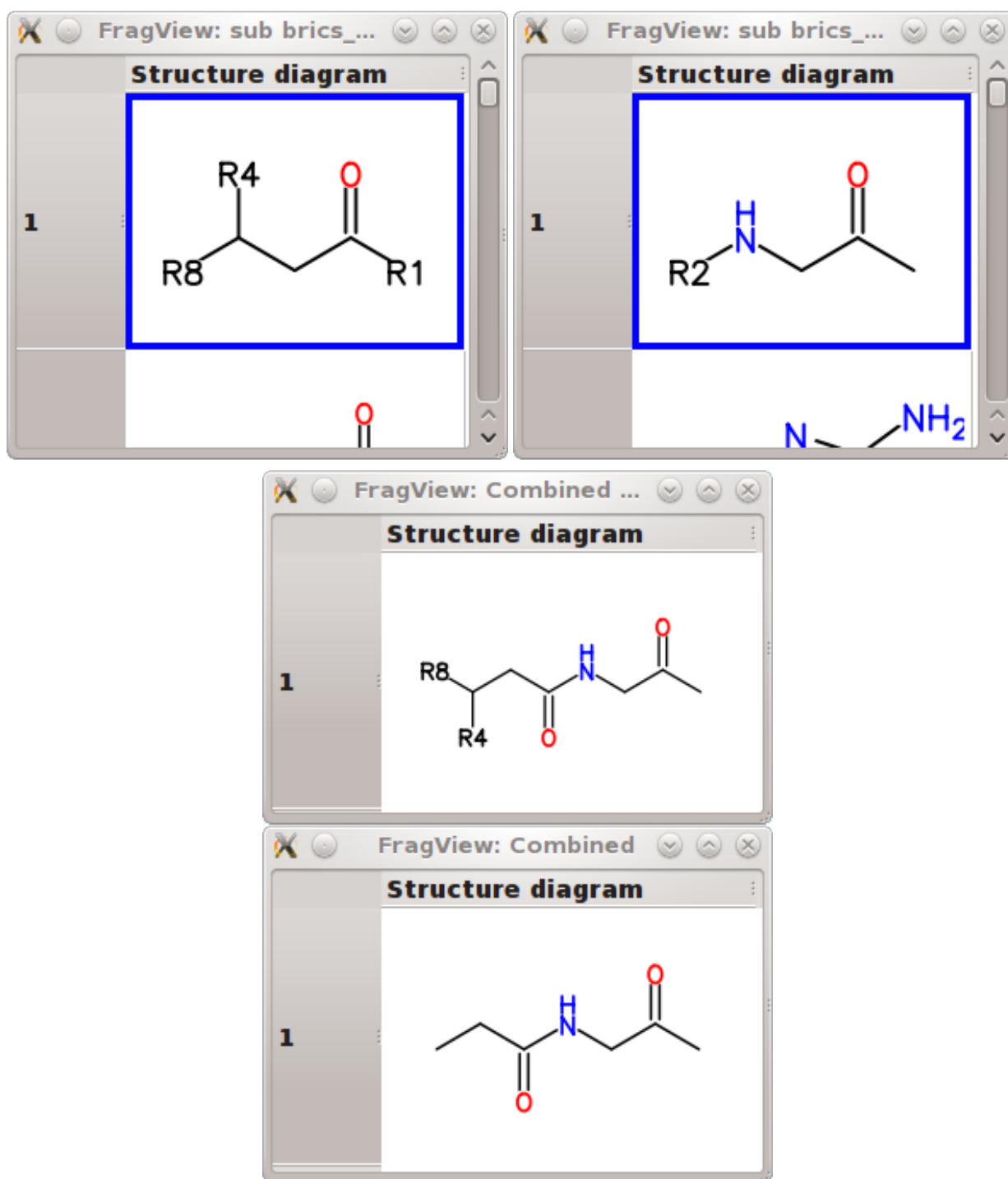


Abbildung 5.5: Selektieren, Verknüpfen und Terminieren von Fragmenten.

In den beiden ersten Tabellen wurde jeweils ein Fragment ausgewählt, was durch eine blaue Umrahmung sichtbar gemacht wird. Diese beiden Fragmente wurden dann zu einem neuen Fragment kombiniert, welches direkt darunter dargestellt ist. Alle Link-Atome des kombinierten Fragments wurden dann terminiert. Das Ergebnis, ein Molekül, ist ganz unten zu sehen.

## 5.2 Enumeration von Molekülen

FRAGENUM wurde entwickelt, um Moleküle mit vorgegebenen physikochemischen Eigenschaften effizient aus einem Fragmentraum zu enumerieren. Die Basis Enumerations-Funktion wurde bewußt sehr einfach ausgelegt. Es wird eine rekursive Funktion verwendet, die Fragmentbäume solange um Fragmente anreichert bis ein Fragmentbaum keine offenen Links mehr hat (siehe Funktion 5.1).

---

**Funktion 5.1 :** BasisEnumeration(FragmentRaum:  $FR$ ).

Algorithmus zum Enumerieren aller Fragmentbäume eines Fragmentraums.

---

**Eingabe** : Fragmentraum  $FR = (\mathcal{F}, \mathcal{R})$ .

**Ausgabe** : Durch aufrufen der Funktion 5.2 alle Moleküle eines Fragmentraums.

```

1  $\vec{K} \leftarrow \text{GeneriereVektor}();$ 
2 foreach linkTyp  $\in \mathcal{R}$  do
3    $\mathcal{K} \leftarrow \text{GeneriereKompatibilitätsMenge}(\text{linkTyp});$       // Def. 4.4.2
4    $\text{FügeHinzu}(\vec{K}, \mathcal{K});$ 
5 foreach  $F \in \mathcal{F}$  do
6    $FB \leftarrow \text{GeneriereFragmentBaum}(F);$       // FB mit F als Wurzel
7    $\text{BasisEnumeriereRekursiv}(FB, \vec{K}, FR);$       // Funktion 5.2
8    $\text{Lösche}(FB);$ 
```

---

Funktion 5.1 erstellt die nötigen Kompatibilitätsmengen für jeden Link-Typ und startet die Enumeration, indem es iterativ jedes Fragment des verwendeten Fragmentraums zur Wurzel eines Fragmentbaums macht. Mit diesem Fragmentbaum, den Kompatibilitätsmengen und dem verwendeten Fragmentraum wird dann die Funktion 5.2 aufgerufen, die die eigentliche Enumeration leistet. Zunächst wird in Funktion 5.2 zu jedem übergebenen Fragmentbaum das entsprechende Molekül generiert und abgespeichert (Zeile 1 und 2). Danach wird über alle offenen Link-Atome des Fragmentbaums iteriert und für jedes das zugehörige Fragment, der Link-Typ und die Kompatibilitätsmenge des Link-Typs bestimmt (Zeile 4 bis 7). In den beiden inneren Schleifen wird jedes Link-Atom jedes Fragments (Zeile 8 bis 10) der Kompatibilitätsmenge auf Kompatibilität mit dem aktuell betrachteten Link-Atom des Fragmentbaums getestet (Zeile 12). Ist eine Kompatibilität gegeben, wird das Fragment aus der Kompatibilitätsmenge über die beiden betrachteten Link-Atome an den Fragmentbaum gebunden und die Funktion 5.2 wird rekursiv mit dem neuen Baum aufgerufen (Zeile 12 - 14). Nachdem der Aufruf aus der Rekursion zurück kommt, wird die zuvor geknüpfte Verbindung getrennt und die innerste Schleife weiter iteriert.

Wie bereits erwähnt, verwaltet ein Fragmentbaum nur die Informationen wie Fragmente miteinander verbunden sind, ohne sie wirklich zu verbinden. Daher verbindet die Funktion `Verbinde` in Zeile 13 der Funktion 5.2 nicht wirklich zwei

---

**Funktion 5.2 :** BasisEnumeriereRekursive(FragmentBaum:  $FB$ , KompatibilitätsTupel:  $\vec{K}$ , FragmentRaum:  $FR = (\mathcal{F}, \mathcal{R})$ ).  
 Rekursive Funktion zur Enumeration aller Fragmentbäume eines Fragmentraums.

---

**Eingabe** : Fragmentbaum  $FB$ , Tupel von Kompatibilitätsmengen  $\vec{K}$  und Fragmentraum  $FR = (\mathcal{F}, \mathcal{R})$

**Ausgabe** : Alle Moleküle eines Fragmentraums.

```

1 mol ← ÜberführeFragmentBaumInMolekül( $FB$ );
2 SpeichereMolekül( $mol$ );
3 linksBaum ← OffeneLinks( $FB$ ) ;           // Alle offenen Links von  $FB$ 
4 foreach  $l_B \in \text{links}_{\text{Baum}}$  do
5    $F_B \leftarrow \text{FragmentZuLink}(l_B)$ ;
6    $t_B \leftarrow \text{LinkTyp}(l_B)$ ;
7    $\mathcal{K}_{t_B} \leftarrow \text{KompatibilitätsMenge}(\vec{K}, t_B)$ ;
8   foreach Frag  $\in \mathcal{K}_{t_B}$  do
9     links = ListeAllerOffenenLinks(Frag);
10    foreach  $l_{\text{Frag}} \in \text{links}$  do
11       $t_{l_{\text{Frag}}} \leftarrow \text{LinkTyp}(l_{\text{Frag}})$ ;
12      if SindKompatibel( $\mathcal{R}, r_{t_B, t_{l_{\text{Frag}}}}$ ) then
13        Verbinde(Frag,  $l_{\text{Frag}}$ ,  $F_B$ ,  $l_B$ );
14        BasisEnumeriereRekursiv( $FB, \vec{K}, FR$ );
15        Trenne(Frag,  $l_{\text{Frag}}$ ,  $F_B$ ,  $l_B$ );
```

---

Fragmente zu einem Neuen, sie annotiert vielmehr an den entsprechenden Fragmenten, welche Link-Atome bei dieser Verbindung verwendet wurden. Entsprechend entfernt die Funktion **Trenne** in Zeile 15 diese Informationen wieder. Einzig die Funktion **ÜberführeFragmentBaumInMolekül** in Zeile 1 generiert ein Molekül durch Verknüpfung aller Fragmente und Terminierung aller offenen Link-Atome.

Die Funktion 5.1 und 5.2 stellen nur das Gerüst der Enumeration dar. Viele wichtige Aspekte fehlen noch, wie zum Beispiel die Beschränkung der Fragmente pro Fragmentbaum. Auch werden Eigenschaften noch nicht in die Enumeration einbezogen. Eine einfache Implementierung, bei der ein Molekül nach dem anderen generiert würde und einer Überprüfung unterzogen würde, ob es die gewünschten Eigenschaften erfüllt, scheidet aufgrund der potentiellen Größe von Fragmenträumen aus. So enthält zum Beispiel der Fragmentraum, der aus Molekülen des *World drug index* (WDI)[88], mithilfe des FTREE-FS Regelsatzes generiert wurde, ca.  $10^{18}$  Moleküle[11]. Auch wenn eine Million Moleküle pro Sekunde generiert und getestet werden könnten, würde dieses einfache Vorgehen immer noch mehr als 31.000 Jahre dauern. Ein weiteres Problem ergibt sich bei der Enumeration selbst: Wie können Duplikate während der Enumeration vermieden werden? Eine Enumeration mit

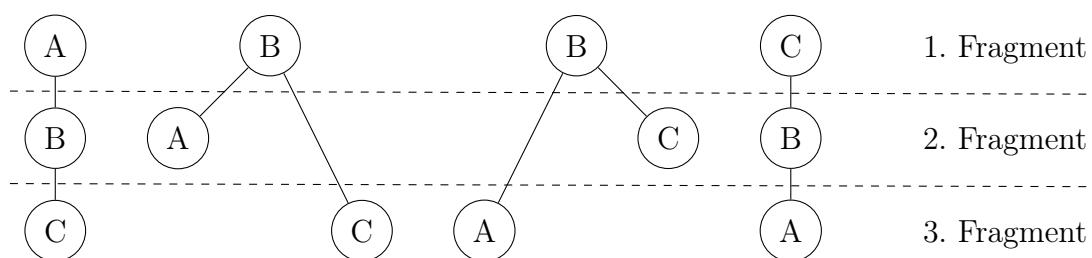


Abbildung 5.6: Redundante Enumeration von Fragmentbäumen.

Fragment B hat zwei Link-Atome, A und C einen. Jeweils ein Link-Atom von B ist mit dem Link-Atom von A und C kompatibel.

Funktion 5.1 produziert sehr viele redundante Kombinationen. Folgendes kleines Beispiel soll das Problem illustrieren: Angenommen ein Fragmentraum bestünde aus drei Fragmenten A, B und C und angenommen A und C hätten ein und B zwei Link-Atom(e). Weiterhin seien die Link-Typen von A und C mit jeweils einem Link-Typen von B kompatibel. Dann würden alle Kombinationen aus Abbildung 5.6 generiert werden. Bei diesem einfachen Beispiel würden schon für das Molekül A-B-C vier verschiedene Kombinationen generiert werden. Wären A und C zu jedem der beiden Link-Typen von B kompatibel, würde sich die Anzahl sogar auf acht verdoppeln. Eine Abschätzung, wieviele Permutationen es für einen Fragmentbaum gibt, liefert der Satz von Cayley[73, 89]:

**Satz 5.2.1 (Satz von Cayley):**

*Es gibt  $n^{n-2}$  verschiedene bezeichnete Bäume mit  $n$  Knoten.*

Wie in der Einleitung schon dargestellt wurde, hat ein typisches *Drug-Like* Molekül bis zu fünf Fragmente. Gemäß dem Satz von Cayley ergibt dies  $5^{5-2} = 125$  verschiedene Bäume für ein Molekül mit fünf Fragmenten. Der Satz lässt sich aber nicht direkt auf Fragmentbäume anwenden, da bei dem für den Satz zugrundeliegenden Bäumen die Reihenfolge der Knoten keine Rolle spielt, jeder Knoten beliebig viele Kinder haben kann und es keine Typen für Kanten gibt. Dennoch liefert der Satz einen Anhaltspunkt dafür, dass höchstwahrscheinlich zwei Größenordnungen zuviel Fragmentbäume, bei einer einfachen Enumeration generiert werden. Diese Redundanz würde die Situation bei der Enumeration von Fragmenträumen offensichtlich zusätzlich verschärfen. Die potentielle kombinatorische Explosion würde durch eine *Enumerations-Explosion* weiter potenziert werden. Es stellt sich daher die Frage, ob Fragmentbäume irgendwie effizient auf Redundanz überprüft werden können oder ob es nicht sinnvoller wäre gleich nach einer anderen Hilfsstruktur zu suchen. Wie der folgende Abschnitt zeigen wird, lässt sich ein sehr einfacher und eleganter Test angeben, mit dem ein Fragmentbaum auf Redundanz überprüft werden kann und somit eine Enumerations-Explosion verhindert werden kann.

### 5.2.1 Vermeidung von redundanten Fragmentbäumen

Bevor auf die Vermeidung von redundanten Fragmentbäumen eingegangen wird, wird eine Äquivalenzrelation auf Fragmentbäumen definiert.

**Definition 5.2.1 (Graph-Isomorphismus von Fragmentbäumen):**

Zwei Fragmentbäume  $FB_1$  und  $FB_2$  sind Graph-Isomorph, in Zeichen  $FB_1 =_{\Gamma} FB_2$  oder  $FB_1 = FB_2 \pmod{\Gamma}$ , wenn die jeweiligen zugrundeliegenden Graphen, unter Miteinbeziehung der zur Verknüpfung verwendeten Link-Atome, isomorph sind. Die Menge  $[FB_i]_{\Gamma} := \{FB_j | FB_i =_{\Gamma} FB_j\}$  wird als Graph-Isomorphieklasse von  $FB_i$  bezüglich  $\Gamma$  bezeichnet.

Eine Möglichkeit zur Vermeidung von Redundanzen besteht darin, einen Fragmentbaum einer Graph-Isomorphieklasse als Repräsentanten dieser zu definieren und nur diesen zu generieren. Diese Forderung macht mathematisch Sinn, lässt sich aber praktisch nicht verwirklichen, da dazu alle Fragmentbäume generiert werden und auf Graph-Isomorphismen überprüft werden müssten. Ein anderer, praktikabler Weg besteht darin, Invarianten eines Fragmentbaums zu definieren, die in nur einen oder sehr wenigen Fragmentbäumen einer Graph-Isomorphieklasse resultieren. Als Invariante wird in dieser Arbeit die ID eines Fragments verwendet. Des Weiteren wird ausgenutzt, dass bei der Definition von Fragmentbäumen (siehe Definition 4.4.4) verlangt wurde, dass die Regeln beziehungsweise Kanten in einer geordneten Menge gespeichert werden, wobei die Ordnung der Regeln die Chronologie des Baums widerspiegelt. Folgender Satz definiert eine Struktur für Fragmentbäume und besagt, dass jeder Fragmentbaum in diese Struktur *reduziert* werden kann. Für den Satz werden noch Funktionen auf Fragmentbäumen beziehungsweise den Regeln benötigt: Die Funktionen *vater* und *kind* liefern zu einer Regel  $R \in \mathcal{R}_{FB}$  ein Paar bestehend aus einem Fragment und Link-Atom, *vater* das erste und *kind* das zweite (Fragment, Link-Atom) Paar. Das Ergebnis der Funktionen *frag* und *link* ist das Fragment beziehungsweise das Link-Atom eines (Fragment, Link-Atom) Tupels:

$$vater, kind : \mathcal{F} \times \mathcal{L} \times \mathcal{F} \times \mathcal{L} \rightarrow \mathcal{F} \times \mathcal{L} \quad (5.1)$$

$$frag : \mathcal{F} \times \mathcal{L} \rightarrow \mathcal{F} \quad (5.2)$$

$$link : \mathcal{F} \times \mathcal{L} \rightarrow \mathcal{L} \quad (5.3)$$

Es ist noch eine Methode zur Umwandlung eines Fragmentbaums in ein Molekül notwendig.

$$FBzuMol : FB \rightarrow \mathcal{M} \quad (5.4)$$

Es werden bewusst keine Anforderung an die Moleküle  $M \in \mathcal{M}$  zu diesem Zeitpunkt gemacht.

Ausgestattet mit diesen Funktionen kann nun der Satz zur isomorphen Umformung von Fragmentbäumen angegeben werden.

**Satz 5.2.2 (Graph-Isomorphieumformung eines Fragmentbaums):**

Es gibt zu jedem Fragmentbaum  $FB_{FR} = (\mathcal{F}_{FB}, \mathcal{R}_{FB})$  eines Fragmentraums  $FR$  einen Graph-Isomorphen Fragmentbaum  $FB_{FR}^\Gamma = (\mathcal{F}_{FB}, (\mathcal{R}_{FB}, \prec))$ ,  $FB_{FR} =_\Gamma FB_{FR}^\Gamma$  mit folgenden Eigenschaften:

1. Die Menge  $\mathcal{R}_{FB}$  ist bezüglich der Relation  $\prec$  geordnet:  
 $R_i, R_j \in \mathcal{R}_{FB}, (R_i, R_j) \in \prec \Leftrightarrow R_i$  wurde vor  $R_j$  in den Baum eingefügt
2. Die ID der Wurzel ist kleiner gleich aller IDs des Fragmentbaums:  
 $id(vater(frag(R_1))) \leq id(kind(frag(R_i))) \quad R_1, R_i \in \mathcal{R}_{FB}$
3. Die IDs aller Kinder eines Knotens sind monoton steigend bezüglich ihrer Chronologie. Seien  $R_i, R_j \in \mathcal{R}_{FB}$  und  $F_v \in \mathcal{F}_{FB}$ :  
 $F_v = frag(vater(R_i)) = frag(vater(R_j)) \wedge (R_i, R_j) \in \prec$   
 $\Rightarrow id(frag(kind(R_i))) \leq id(frag(kind(R_j)))$
4.  $FBzuMol(FB_{FR}) = FBzuMol(FB_{FR}^\Gamma)$

Es ist nun zu zeigen, dass jeder Fragmentbaum in einen mit den geforderten Eigenschaften umgeformt werden kann und, dass der transformierte Fragmentbaum dasselbe Molekül repräsentiert. Der erste Teil wird durch ein Programm gezeigt. Das Programm hat gemäß Satz 5.2.2 zwei Teile. Im ersten Teil wird das Fragment mit der kleinsten ID zur Wurzel gemacht und im zweiten Teil werden die Kinder jedes Fragments gemäß ihrer ID sortiert. Mit dem Satz wurde gezeigt, dass Fragmentbäume

---

**Algorithmus 5.3:** Algorithmus zur Überführung eines Fragmentbaums  $FB$  in einen Graph-Isomorphen Fragmentbaum, der Bedingungen aus Satz 5.2.2 erfüllt.

---

**Eingabe** : Fragmentbaum  $FB$ .

**Ausgabe** : Graph-Isomorpher Baum  $FB^\Gamma$ , der die Bedingungen des Satzes 5.2.2 erfüllt.

- 1 Überführe Baum  $FB$  in einen Graphen  $G_{FB}$ ;
  - 2 Traversiere Graphen  $G_{FB}$  und speichere Knoten  $F_i$  mit kleinster ID;
  - 3 Generiere einen neuen Baum  $FB^\Gamma$  mit  $F_i$  als Wurzel;
  - 4 Füge  $F_i$  in Liste  $L$  ein;
  - 5 **foreach**  $F \in L$  **do**
  - 6     Finde Knoten  $F$  und dessen Vater  $F_v$  in  $FB^\Gamma$ ;
  - 7     Generiere Liste  $L_n$  mit allen Nachbarn von  $F$ ;
  - 8     Entferne Element  $F_v$  aus Liste  $L_n$ ;
  - 9     Sortiere  $L_n$  gemäß der IDs;
  - 10    **foreach**  $F_n \in L_n$  **do** //  $L_n$  ist sortiert
  - 11     Füge  $F_n$  über die selben Link-Atome wie in  $FB$  als Kind an  $F$  an;
  - 12    Füge Liste  $L_n$  an  $L$  an;
-

sich auf einen Repräsentanten einer Graph-Isomorphieklasse reduzieren lassen. Eine Implikation des Satzes ist, dass ein Fragmentraum vollständig aufgezählt werden kann, indem nur die reduzierten Bäume enumeriert werden.

Jetzt ist noch zu zeigen, dass die beiden Moleküle  $FB_{FR}$ ,  $FB_{FR}^\Gamma$  identisch sind. Da keine Verknüpfung zwischen den Fragmenten geändert wurde, das heißt, alle beteiligten Fragmente immer noch über dieselben Link-Atome verbunden sind, müssen beide Moleküle identisch sein. Das einzige, das geändert wurde sind Baumeigenschaften, wie die Eigenschaft Wurzel oder die Ordnung der Kinder eines Knotens.

Leider lassen sich nicht alle graph-isomorphen Fragmentbäume auf einen Fragmentbaum reduzieren. Der Grund dafür liegt darin, dass die IDs zwischen den Knoten auf kleiner-gleich oder größer-gleich verglichen werden, in beiden Fällen also gleiche IDs zugelassen werden. Dies hat zur Folge, dass zum Beispiel ein Baum, dessen Knoten alle aus demselben Fragment bestehen, also alle Fragmente dieselbe ID haben, nicht reduziert werden kann. Obwohl oben vorgestellte Reduktion diesen Defekt hat, liefert sie in der Praxis sehr gute Ergebnisse (siehe Tabelle 5.5).

Es ist wichtig anzumerken, dass die obige Reduktion und damit verbundene Vermeidung von Redundanzen sich nur auf Topologien von Fragmentbäumen bezieht und nicht zwangsläufig auf die Moleküle, die ein Fragmentbaum repräsentiert. *Je nachdem wie ein Fragmentraum definiert ist, lässt sich ein Molekül durch verschiedene, nicht graph-isomorphe Fragmentbäume repräsentieren.* Ein einfacher Fall ist gegeben, wenn ein atomares Fragment identisch zu einem zusammengesetzten Fragment ist. In diesem Fall entspricht die topologische Eindeutigkeit nicht der Molekül-Eindeutigkeit. Obwohl die Unterscheidung zwischen der Fragmentbaum-Topologie Eindeutigkeit und der Molekül-Eindeutigkeit wichtig ist und stets berücksichtigt werden sollte, ist sie bei vernünftig entworfenen Fragmenträumen zu vernachlässigen.

Die letzten beiden Argumente, dass sich ein Fragmentbaum nicht immer reduzieren lässt und, dass potentiell zwei nicht graph-isomorphe Fragmentbäume dasselbe Molekül repräsentieren, implizieren, dass eine weitere Methode nötig ist, um auf eindeutige Moleküle zu testen. Dies ist per se kein Problem, da die hier vorgestellte Reduzierung dazu dienen soll möglichst effizient, mit einer falsch positiv Rate von 0, so viel wie möglich redundante Moleküle auszusondern. Beide Voraussetzungen erfüllt der nächste Test, der eine Abwandlung von Algorithmus 5.3 ist.

Die Effektivität des Tests soll an einem kleinen Beispiel vorgeführt werden. Es wurden zwei Fragmenträume erstellt, einer mit 47 (`wdi_sub_47`), der andere mit 48 Fragmenten (`wdi_sub_48`). Die Fragmenträume sind bis auf ein Fragment identisch. Die Regeln und die Fragmente stammen aus der Fragmentierung des WDI[88, 11]. Um eine möglichst realistische Testmenge zu generieren, wurde bei den Subräumen die Verteilung der Link-Typen und die Link-Anzahl pro Fragment beibehalten. Das zusätzliche Molekül in `wdi_sub_48` ist neben der Tabelle dargestellt. Dieses Fragment wurde mit Bedacht ausgewählt. Zum einen hat es sechs Link-Atome, was zu einer deutlichen Vergrößerung des Fragmentraums führen sollte und zum anderen ist es symmetrisch, was wiederum eine Herausforderung für einen Redundanztest darstellt.

Die Ergebnisse in Tabelle 5.5 zeigen, dass der hier vorgestellte Redundanztest eindrucksvoll die Anzahl der enumerierten Fragmente reduzieren kann. In beiden



---

**Funktion 5.4 :** IstReduziert(Fragmentbaum:  $FB$ ).

 Funktion zum Testen, ob ein Fragmentbaum reduziert ist, unter Verwendung des Satzes 5.2.2.
 

---

**Eingabe** : Fragmentbaum  $FR = (\mathcal{F}, \mathcal{R})$ .

**Ausgabe** : *true* falls gegebener Fragmentbaum reduziert ist, ansonsten *false*.

```

1 IDWurzel ← ID(Fragment(Wurzel( $FB$ )));
2 foreach Knoten  $K \in FB$  do
3    $F \leftarrow$  Fragment( $K$ );
4   if ID( $F$ ) < IDWurzel then
5     return false;
6   Kinder $K$  ← Kinder( $K$ ) ;           // Kinder sind bzgl < geordnet
7   IDprev ← -1;                       // IDs sind positiv
8   foreach Knoten  $B \in$  Kinder $K$  do
9     IDcurr ← ID(Fragment( $B$ ));
10    if IDcurr < IDprev then
11      return false;
12    IDprev ← IDcurr;
13 return true;
```

---

Räumen wurde die Anzahl um eine Größenordnung reduziert, im `wdi_sub_47` um ziemlich genau den Faktor 10, im `wdi_sub_48` Fall sogar um den Faktor 17. Die Tabelle zeigt aber auch, dass ein Redundanztest auf Topologie-Basis nicht ausreicht, um nur eindeutige Moleküle zu enumerieren. In beiden Beispiel-Fragmenträumen besteht eine Diskrepanz zwischen der Anzahl der Moleküle mit aktivierten Topologie-Redundanztest und eindeutigen Molekülen. Wie bereits angeführt, wurde dieses Verhalten erwartet und daher wird von jedem Molekül, bevor es gespeichert wird, die Unique-SMILES[90] Repräsentation berechnet und mit allen bisher gespeicherten Unique-SMILES verglichen. Die Diskrepanz in dem Faktor zwischen der Anzahl der enumerierten Moleküle mit aktivierten Topologie-Filter und eindeutigen Molekülen zwischen den beiden Räumen, 1,32 zu 11,14, zeigt die inherente Schwachstelle des hier vorgestellten Redundanztests: Der Test basiert auf Baum-Topologien und nicht auf den zugrundeliegenden Fragmenten, er "sieht" nicht die Symmetrie des zusätzlichen Fragments. Das hat zur Folge, dass zum Beispiel alle Bäume, die das zusätzliche Fragment als Wurzel haben und den Bedingungen aus Satz 5.2.2 genügen, sechsmal generiert und nicht als redundant erkannt werden. Das Fragment rotiert als Wurzel durch alle Link-Atome durch und erfüllt für jede Rotation die Bedingungen aus Satz 5.2.2. Die Reduktion der enumerierten Moleküle spiegelt sich natürlich auch in der benötigten Zeit wieder. Die komplette Enumeration aller Moleküle mit bis zu fünf Fragmenten benötigt im `wdi_sub_47` Fall 9.412 Sekunden. Mit aktiviertem Redundanzfilter nur noch 686 Sekunden. Natürlich sind die absoluten

Fragmentraum	Enum. Mol.	Eind. Mol.	Faktor	%	Sek.	%
wdi_sub_47	1048401	73147	14,33	6,98%	9412	100%
wdi_sub_47	96549	73147	1,32	75,76%	686	7,28%
wdi_sub_48	17399458	95713	181,79	0,55%	114697	100%
wdi_sub_48	1066648	95713	11,14	8,97%	7311	6,37%

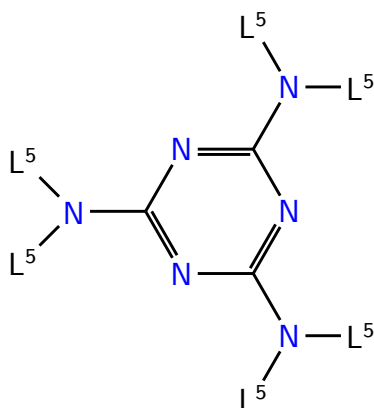


Tabelle 5.5: Beispiel für die Effektivität des Baumtopologie Redundanztests.

Abgebildet sind die Anzahl generierter Moleküle (*Enum. Mol.*), Anzahl eindeutiger Moleküle (*Eind. Mol.*), Faktor zwischen enumerierter Moleküle und eindeutigen Molekülen (*Faktor* und *%*) und in der letzten Spalte Zeiten für die Enumeration (*Sek.* und *%*) von zwei Fragmenträumen: **wdi\_sub\_47** und **wdi\_sub\_48**. Die beiden Räume unterscheiden sich nur durch das unten abgebildete Fragment, welches in **wdi\_sub\_48** zusätzlich enthalten ist. In der ersten Zeile jedes Raums sind die Werte für eine vollständige Enumeration angegeben, in der zweiten Zeile die Werte mit aktivierten Redundanztest. Wie man sehen kann, reduziert der Baumtopologiefilter die Anzahl der enumerierten Moleküle als auch die benötigte Zeit um mehr als eine Größenordnung.

Der Faktor in der vierten Spalte berechnet sich aus  $\frac{\text{Enum. Mol.}}{\text{Eind. Mol.}}$ . Die Prozentzahl ist der Kehrwert von Faktor multipliziert mit 100. Die Prozentzahl in der letzten Spalte gibt das Verhältnis zwischen der Zeit ohne und mit aktiviertem Redundanzfilter an, wobei die Enumeration ohne Filter 100% entspricht.

Zeiten hardwareabhängig, so dass das Verhältnis mehr von Interesse ist. In beiden Fällen benötigt die Enumeration mit aktivierten Topologiefilter nur etwa 7% der Zeit einer ungefilterten Enumeration. Es zeigt sich somit auch bei der benötigten Enumerationszeit eine Reduzierung von etwas mehr als einer Größenordnung.

Mit Tabelle 5.5 lässt sich zusammenfassend sagen, dass der vorgestellte Redundanztest auf Baumtopologien die Enumerations-Explosion grundsätzlich eindämmen kann. Es muss aber darauf hingewiesen werden, dass der Test eine strukturelle Schwäche hat.

Um alle möglichen Fragmentkombinationen zu erlauben, belässt der Test gleiche IDs ungefiltert. Auch wenn diese Schwäche vorhanden ist, kann sie nur bei Fragmenten auftreten, die mindestens zwei Link-Atome haben, deren Typen kompatibel sind. Eine grundsätzliche Vermeidung dieser Schwäche ist aufgrund des Verfahrens nicht möglich. Eine Minderung scheint auch nicht möglich, da dazu im Wesentlichen alle Bäume mit gleichen Fragmenten, die entweder in Vater, Kind oder Geschwisterbeziehung stehen abgespeichert werden müssten, um später zu testen, ob diese schon generiert wurden. Für einen Spezialfall, der als nächstes behandelt wird, könnte aber dennoch eine Minderung erreicht werden. Eine andere Schwäche des Redundanztests ist, dass es nicht Symmetrien in Fragmenten berücksichtigt. Dies kann dem Verfahren aber nicht grundsätzlich zugeschrieben werden, da es für Baumtopologien entwickelt wurde und, außer den IDs der Knoten, keine Information über die Knoten besitzt. Dennoch könnte man hier für einige Spezialfälle den Redundanztest erweitern. In einem Vorverarbeitungsschritt könnten alle Fragmente, die Link-Atom Symmetrien aufweisen, markiert werden. Diese Information könnte dann verwendet werden, um ein Rotieren der Fragmente bezüglich der symmetrischen Link-Atome in der Wurzel oder allen Blattknoten zu vermeiden. Weitere Tests wären für Fragmente möglich, die nur Link-Atome eines Link-Typs haben und bezüglich dieser symmetrisch sind. Für diese Fragmente könnten auch Tests für innere Knoten im Baum entwickelt werden.

### 5.2.2 Eigenschaftsbasierte Enumeration

Das Ziel dieser Arbeit ist es, Fragmenträume gemäß vorgegebener physikochemischer Eigenschaften zu enumerieren. Es wurde schon erläutert, dass schon Fragmenträume mit nur wenigen Fragmenten, aufgrund der kombinatorischen Explosion, sehr viele Moleküle enthalten können. Im Bereich des Wirkstoffentwurfs sind auch nicht alle Moleküle eines Fragmentraums von Interesse, sondern nur diejenigen, die ein bestimmtes physikochemisches Profil aufweisen (siehe Kapitel 2.1)[7, 91, 29, 92, 93, 40, 44]. Die kombinatorische Explosion macht es aber in vielen Fällen unmöglich, zuerst die Moleküle zu enumerieren und dann auf gewünschte Eigenschaften zu testen. Vielmehr ist es nötig, schon während der Auswahl von Fragmenten für den nächsten Anbauschnitt nur diejenigen zu selektieren, die zu den gewünschten Eigenschaften führen. Eine einfache Möglichkeit dies zu realisieren besteht darin, *additive Eigenschaften* von Fragmenten und Molekülen zu betrachten. Dies mag auf den ersten Blick wie eine Einschränkung wirken, aber bei genauerer Betrachtung stellen sich viele physikochemische Eigenschaften als additiv oder linear heraus. Die offensichtlichste ist die Anzahl der Atome und somit das Gewicht von Fragmenten. Andere Eigenschaften, wie zum Beispiel  $c \log P$  (siehe Abschnitt 4.1) können mithilfe von linearer Regression sehr gut modelliert werden. Es gibt aber auch Eigenschaften, die weder additiv noch linear sind, wie zum Beispiel SMARTS[86]. Bei nicht additiven Eigenschaften kann keine Vorhersage über das Verhalten gemacht werden. Vielmehr müssen diese Eigenschaften explizit neu berechnet werden, wenn zwei Fragmente kombiniert werden. FRAGENUM unterteilt Eigenschaften daher in einer Hierarchie: additiv und nicht

additiv (siehe Kapitel 4.3). Additive Eigenschaften werden immer zuerst verarbeitet. Wenn alle gewünschten additiven Eigenschaften mit ihren unteren und oberen Schranken die vorgegebenen Intervalle erfüllen und nicht additive Eigenschaften abgefragt werden, muss das Fragment zusammengesetzt und die nicht additive Eigenschaft berechnet werden. Dadurch, dass die nicht additiven Eigenschaften als letztes berechnet werden, wurden schon alle Fragmente aussortiert, die die additiven Eigenschaften nicht erfüllen. Falls nur nicht additive Eigenschaften angefragt werden, müssen alle enumerierten Fragmente zusammengesetzt und die Eigenschaft berechnet werden. Dies führt zu einer sehr ineffektiven Enumeration. Es zeigt sich aber, dass sehr fundamentale Eigenschaften, wie zum Beispiel das molekulare Gewicht, ausschlaggebend für pharmazeutisch wirksame Moleküle sind[7, 92, 40, 44]. Daher sollte es nicht schwer fallen, einige elementare, additive Eigenschaften für jede Enumeration als Eigenschaften anzugeben.

### Effiziente Verwaltung von Fragmenteigenschaften

Auch wenn additive Eigenschaften eine Evaluierung mehrerer Fragmente erlauben, ohne diese explizit zusammenzusetzen, stellt sich das Problem des effizienten Zugriffs auf Fragmente mit Eigenschaften in bestimmten Bereichen. Eine eigenschaftsbasierte Enumeration eines Fragmentraums sollte mehrere Eigenschaften gleichzeitig unterstützen. Dies bedeutet, dass in einem mehrdimensionalen Eigenschaftsraum nach Fragmenten gesucht werden muss. Diese Art von Anfragen wird effizient von binären *k dimensionalen Suchbäumen* (k-d Bäume)[94, 95] unterstützt. Ein k-d Baum verwaltet mehrdimensionale Daten, indem er pro Knoten den mehrdimensionalen Raum durch eine Hyperebene unterteilt. K-d Bäume verarbeiten die Daten genau wie binäre Suchbäume, mit dem Unterschied, dass pro Ebene eine andere Dimension unterteilt wird. Die Söhne eines Knotens entsprechen dann allen Vektoren, die bezüglich der betrachteten Dimension kleiner oder größer sind. Der Einfachheit wird bei k-d Bäumen jeder Dimension eine Zahl zugeordnet und die Dimension für eine Baumebene ergibt sich aus der Anzahl der Dimensionen modulo der Baumtiefe eines Knotens.

Ein Vorteil dieser Datenstruktur ist, insbesondere im Rahmen dieser Arbeit, dass sich mehrdimensionale Bereichsanfragen effizient realisieren lassen. Dazu wird an jedem Knoten der Wert des Knotens mit dem Intervall der entsprechenden Dimension verglichen. Ist der Wert des Knotens kleiner als das Minimum des Intervalls, wird die Suche rekursiv auf dem rechten, ist er größer auf dem linken Subbaum weitergeführt. Sobald der Wert eines Knotens im Intervall der entsprechenden Dimension liegt, werden die anderen Dimensionen auf Inklusion geprüft und bei Erfolg in eine Ergebnisliste eingefügt. Die Suche wird dann rekursiv in beiden Kindern fortgesetzt.

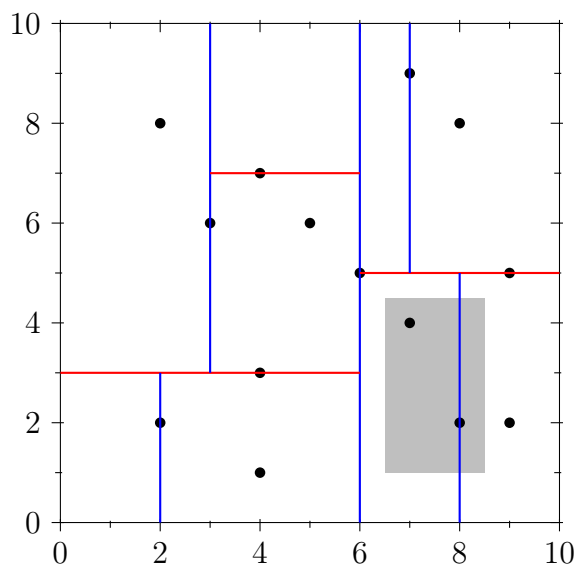


Abbildung 5.7: Räumliche Darstellung der Hyperebenen eines k-d Baums.

In diesem Beispiel ist eine mögliche Aufteilung in Hyperebenen für einen k-d Baum dargestellt. Die blauen Linien trennen die  $x$ , die roten die  $y$ -Achse. Das graue Rechteck repräsentiert ein Beispiel für eine Bereichsanfrage.

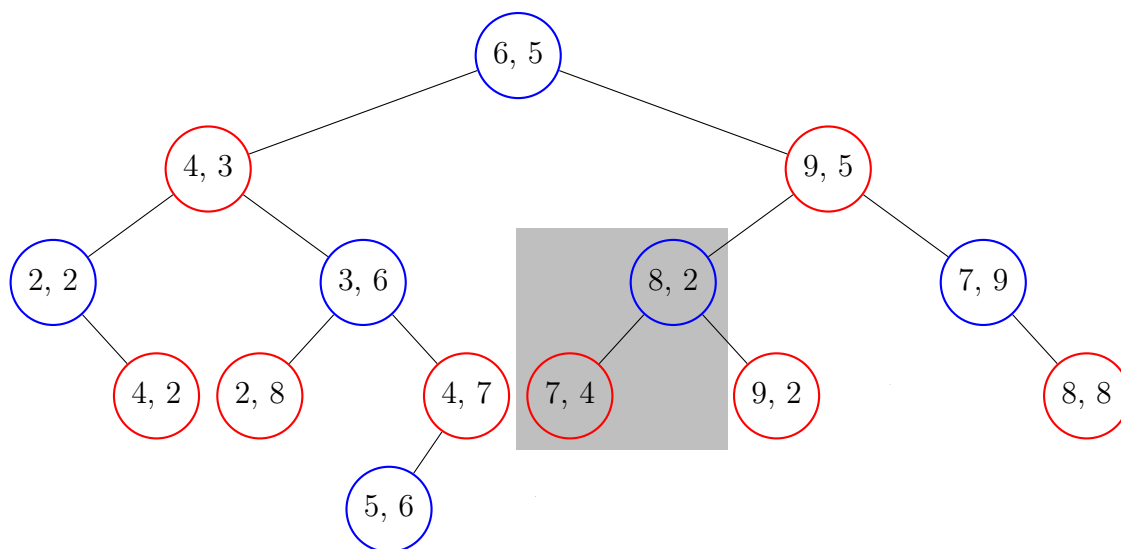


Abbildung 5.8: k-d Baum zur Abbildung 5.7.

Es lässt sich leicht erkennen, dass pro Baumebene alle Knoten dieselbe Farbe haben, also dieselbe Dimension repräsentieren. Die Knoten für die Bereichsanfrage aus Abbildung 5.7 sind grau hinterlegt.

Diese Eigenschaft von k-d Bäumen wird ausgenutzt, um bei additiven Eigenschaften gezielt Fragmente abzufragen, die in einem Bereich liegen, die die vorgegebenen Eigenschaften erfüllen können. Wird zum Beispiel vom Benutzer ein Intervall für das molekulare Gewicht von 300 bis 500 Da vorgegeben und der aktuell betrachtete Fragmentbaum repräsentiert ein Fragment 200 Da, dann können nur Fragmente mit einem Gewicht, welches zwischen 100 und 300 liegt, die vom Benutzer vorgegebene Anfrage erfüllen. Eine weitere Beschleunigung wird erzielt, indem für jeden Link-Typ ein k-d Baum aus der Kompatibilitätsmenge des Link-Typs generiert wird. Dadurch enthält jeder k-d Baum nur Fragmente, die mindestens ein kompatibles Link-Atom zu einen Link-Typen besitzen. Als weitere Optimierung werden k-d Bäume nur über

---

**Funktion 5.5 :** GeneriereKDBaeume(Fragmentbaum:  $FR$ , Liste:  $L$ ).

Funktion zum Generieren von link spezifischen k-d Bäumen über einer Menge von Eigenschaften.

---

**Eingabe** : Fragmentraum  $FR = (\mathcal{FR})$ .

Liste  $L$  von IDs für physikochemischen Eigenschaften, für welche k-d Bäume generiert werden sollen.

**Ausgabe** : Ein Vektor  $T_{kd}$  von k-d Bäumen.

Jedes  $t_{lt} \in T_{kd}$  wird aus der Kompatibilitätsmenge des Link-Typs  $lt$  generiert und wird an der Stelle  $lt$  abgespeichert:  $T_{kd}[lt] = t_{lt}$ .

```

1 foreach  $f \in \mathcal{F}$  do
    | // Initialisiere Fragment  $f$  mit allen Eigenschaften aus  $EL$ 
2 | InitialisiereFragment( $f$ ,  $EL$ );
3 foreach Link-Typ  $lt \in \mathcal{R}$  do
4 |   Generiere Kompatibilitätsmenge  $\mathcal{K}_{lt}$  ;           // Definition 4.4.3
5 |   Erstelle k-d Baum  $t_{lt}$  aus  $\mathcal{K}_{lt}$ ;
6 |    $T_{kd}[lt] = t_{lt}$ ;
7 return  $T_{kd}$ 
```

---

die Eigenschaften gebildet, welche der Benutzer angefragt hat. Obwohl dadurch für jeden Enumerationsprozess jeweils neue k-d Bäume generiert werden müssen, sollte das durch die Verminderung von Dimensionen in den generierten k-d Bäumen wieder mehr als ausgeglichen werden. Die wesentlichen Schritte zur Initialisierung sind in Funktion 5.5 dargestellt.

Abbildung 5.9 zeigt das Menü von FRAGENUM zur Eingabe von Min-Max Werten für unterstützte physikochemische Eigenschaften.

Mit Benutzer vorgegebenen Intervallen und entsprechend generierten k-d Bäumen, können nun die Funktionen 5.6 und 5.7 modifiziert werden.

Ein kritischer Funktionsaufruf in Funktion 5.7 ist **BerechneIntervalle** in Zeile 4. Diese Funktion berechnet für jede abgefragte additive Eigenschaft einen Status und ein Intervall. Der Status gibt an, ob eine Eigenschaft unter, im oder über dem geforderten minimal-maximal Wert für diese Eigenschaft liegt. Im Status sind

#	Property	[min	max]
1	number of atoms	[ not	set ]
2	molecular weight	[ 176.20,	516.30 ]
3	number of non-hydrogen atoms	[ not	set ]
4	number of non-hydrogen bonds	[ not	set ]
5	number of rings (sssr)	[ 0,	3 ]
6	number of h-bond acceptors	[ 4,	9 ]
7	number of h-bond donors	[ 0,	4 ]
8	number of specific hydrophobic contacts	[ not	set ]
9	number of h-bonds + salt bridges	[ not	set ]
10	number of rotatable bonds	[ 3,	12 ]
11	max path of contiguous rotatable bonds	[ not	set ]
12	topological polar surface area	[ not	set ]
18	calculated log P value	[ -0.20,	4.20 ]
19	molar refractivity	[ not	set ]
20	polar surface area	[ not	set ]
23	inclusion smarts	[ not	set ]
24	exclusion smarts	[ not	set ]

Abbildung 5.9: Beispieleingabe für FRAGENUM.

Die dargestellten Werte für die verschiedenen physikochemischen Eigenschaften sind aus dem ersten Beispiel aus Tabelle 6.2.

alle noch offenen Link-Atome mit ihren terminalen Fragmenten berücksichtigt. Das berechnete Intervall beschreibt, um wieviel der Fragmentbaum minimal und maximal in dieser Eigenschaft wachsen darf, damit er noch die geforderten minimal-maximal Werte erfüllt. Ist der Status einer Eigenschaft **GRÖßER**, kehrt die Funktion 5.7 sofort zurück (Zeile 8), da keine Möglichkeit mehr existiert, die vorgegebenen Eigenschaften zu erfüllen. Hat mindestens eine Eigenschaft den Wert **KLEINER**, wird kein Molekül generiert, der Fragmentbaum wird aber weiterhin verwendet, da er potentiell noch die Vorgaben erfüllen kann. Liegen alle Werte in den entsprechenden vorgegebenen Intervallen, wird das korrespondierende Molekül zum Fragmentbaum generiert. Dieses wird dann darauf überprüft, ob es auch alle nicht additiven Eigenschaften erfüllt (Zeile 13). Diese Funktion nimmt alle nötigen Initialisierungen vor, um die benötigten Eigenschaften zu berechnen (siehe Tabelle 4.1). Bedingt durch die Initialisierungsberechnungen ist diese Funktion die rechenzeitaufwendigste. Wenn das Molekül auch alle nicht additiven Eigenschaften erfüllt, wird der *Unique* SMILES zum Molekül berechnet und überprüft, ob schon ein Molekül mit demselben *Unique* SMILES

---

**Funktion 5.6** : Enumeriere(Fragmentraum:  $FR$ , Liste:  $L$ ).

Funktion zum eigenschaftsbasierten Enumerieren aller Fragmentbäume eines Fragmentraums.

---

**Eingabe** : Fragmentraum:  $FR = (\mathcal{F}, \mathcal{R})$ Liste von Eigenschaftsintervallen:  $L$ .**Ausgabe** : Alle Fragmente aus  $FR$ , die die Eigenschaftensintervalle aus  $L$  erfüllen.

---

```
1 Tkd = GeneriereKDBaeume(FR, EL) ;                               // Funktion 5.5
2 uniqueMols = GeneriereAVLBaum();
3 foreach F ∈  $\mathcal{F}$  do
4   | FB ← GeneriereFragmentBaum(F) ;                               // FB mit F als Wurzel
   | // Funktion 5.7
5   | EnumeriereRekursiv(FB, Tkd, EL, FR, uniqueMols);
6   | Lösche(FB);
```

---

generiert wurde. Da *Unique* SMILES Zeichenketten Repräsentationen eines Molekül sind, lassen sie sich sehr effizient in einem AVL-Baum[96, 97, 98] verwalten. Ab Zeile 18 werden alle nötigen Informationen für die nächste Iteration gesammelt. In Zeile 23 werden die zuvor berechneten Intervalle benutzt, um nur Fragmente aus einem k-d Baum zu extrahieren, die in Kombination mit dem aktuellen Fragmentbaum die Benutzervorgaben erfüllen können. In Zeile 29 wird ein neuer Baum mit einem zuvor extrahierten Fragment generiert, welcher dann in Zeile 30 für den rekursive Aufruf genutzt wird.



**Funktion 5.7 :** EnumeriereRekursiv.

Funktion zur Enumeration aller Fragmentbäume eines Fragmentraums unter Berücksichtigung von Eigenschaftsintervallen.

---

**Eingabe** : Fragmentbaum:  $FB$ , Vektor von k-d Bäumen:  $T_{kd}$ , Liste von Eigenschaftsintervallen:  $EL$ , AVL\_Baum:  $UniqueMols$ , Fragmentraum:  $FR = (\mathcal{F}, \mathcal{R})$ .

**Ausgabe** : Alle Moleküle eines Fragmentraums, die Bedingungen aus  $EL$  erfüllen.

---

```

1 maxAnzFragmente  $\leftarrow$  MaximaleAnzahlFragmenteProBaum( $EL$ );
2 if AnzahlKnoten( $FB$ ) > maxAnzFragmente or not IstReduziert( $FB$ )
  then
3   return;
4 intervale  $\leftarrow$  BerechneIntervalle( $FR$ ,  $FB$ ,  $EL$ );
5 erfülltBedingungen  $\leftarrow$  true;
6 foreach  $i \in$  intervale do
7   if  $i.status = GRÖßER$  then
8     return;
9   else if  $i.status = KLEINER$  then
10    erfülltBedingungen  $\leftarrow$  false;
11 if erfülltBedingungen then
12   molekül  $\leftarrow$  ÜberführeFragmentBaumInMolekül( $FB$ );
13   if ErfülltNichtAdditiveEigenschaften(molekül,  $EL$ ) then
14     us  $\leftarrow$  UniqueSMILES(molekül);
15     if not Enthält( $UniqueMols$ , us) then
16       SpeichereMolekül(molekül);
17       FügeEin( $UniqueMols$ , us);
18 linksFB  $\leftarrow$  ListeAllerOffenenLinks( $FB$ );
19 foreach  $l_{FB} \in$  linksFB do
20   fFB  $\leftarrow$  FragmentZuLink( $l_{FB}$ );
21   tFB  $\leftarrow$  LinkTyp( $l_{FB}$ );
22   kdBaum  $\leftarrow$  KDBaum( $T_{kd}$ , tFB) ;           // k-d Baum zu Link-Typ tFB
23   fragskd  $\leftarrow$  ExtrahiereFragmente(kdBaum, intervale);
24   foreach fkd  $\in$  fragskd do
25     linkskd  $\leftarrow$  ListeAllerOffenenLinks(fkd);
26     foreach  $l_{kd} \in$  linkskd do
27       tkd  $\leftarrow$  LinkTyp( $l_{kd}$ );
28       if SindKompatibel( $\mathcal{R}$ , tkd, tFB) then
29         Verbinde(fFB,  $l_{FB}$ , fkd,  $l_{kd}$ );
30         EnumeriereRekursiv( $FB$ ,  $T_{kd}$ ,  $EL$ ,  $UniqueMols$ ,  $FR$ );
31         Trenne(fFB,  $l_{FB}$ , fkd,  $l_{kd}$ );

```

---



## 6 Anwendungsszenario

Um die Fähigkeiten, aber auch die Grenzen des Programms FRAGENUM zu demonstrieren, wurde es in einem Anwendungsszenario verwendet, welches eine realistische Anwendung des Programms widerspiegelt[2].

### 6.1 Target-Klassen und Generierung zugehöriger Fragmenträume

Für das Szenario wurden zu den folgenden pharmazeutisch relevanten Target-Klassen Inhibitoren aus der ZINC Datenbank[99, 100] ausgewählt: Angiotensin-konvertierendes Enzym (ACE), Acetylcholinesterase (AChE), Adenosin Desaminase (ADA), Aldose Reduktase (ALR2), AmpC  $\beta$ -Lactamase (AmpC), Androgenrezeptor (AR), Cyclin-abhängige Kinase 2 (CDK2), Catechol *O*-Methyltransferase (COMT), Cyclooxygenase-1 (COX-1), Cyclooxygenase-2 (COX-2), Dihydrofolat Reduktase (DHFR), Epidermal growth factor receptor (EGFr), Estrogen Rezeptor (ER), Fibroblast growth factor receptor (FGFr), Faktor Xa (FXa), Glycinamid-ribonucleotid Transformylase (GART), Glycogen-Phosphorylase  $\beta$  (GPB), Glukokortikoid-Rezeptor (GR), Human immunodeficiency virus protease (HIVPR), Human immunodeficiency virus reverse Transkriptase (HIVRT), High mobility group Proteine (HMGA), Heat shock protein 90 (HSP90), Enoyl-acyl carrier protein reductase (InhA), Mineralokortikoidrezeptor (MR), Neuraminidase (NA), p38 Mitogen-aktivierte Proteinkinase (P38 MAP), Poly adenosine diphosphate ribose polymerase (PARP), Phosphodiesterase-5 (PDE5), Platelet derived growth factor receptor *beta* (PDGFr $\beta$ ), Purin-Nukleosid-Phosphorylase (PNP), Peroxisom Proliferator-aktivierter Rezeptor  $\gamma$  (PParg), Progesteron Rezeptor (PR), Retinoid X Rezeptor  $\alpha$  (RXRa), S-Adenosylhomocystein Hydrolase (SAHH), Tyrosinkinase SRC (SRC), Thrombin, Thymidinkinase (TK), Trypsin und Vascular endothelial growth factor receptor 2 (VEGFr2)<sup>1</sup>. Für jede Target-Klasse wurde aus den korrespondierenden Inhibitoren ein Fragmentraum mithilfe des Programms RECORE[58] generiert. Als Schneideregeln wurde ein modifizierter RECAP Regelsatz verwendet (siehe Abbildung 4.4). Es ist wichtig, darauf hinzuweisen, dass aufgrund der Schneideregeln nicht alle Moleküle fragmentiert werden konnten. So gab es zum Beispiel kleine Moleküle, auf die, aufgrund der Größe,

---

<sup>1</sup>Wenn immer möglich und sinnvoll, wurde der deutsche Name einer Target-Klasse verwendet. Die Kommunikation über Target-Klassen findet sehr häufig über ihre Abkürzungen statt, wie zum Beispiel HSP90. Auch wenn es sicherlich ein deutsches Gegenstück zu *Human heat shock protein 90* gibt, würde diese nicht auf die Abkürzung passen. Aus diesem Grund habe ich bei einigen Target-Klassen die englischen Namen belassen.

keine Schneideregeln angewendet werden konnte. In anderen Fällen hätte eine Fragmentierung ein sehr kleines Fragment mit nur einem Link-Atom generiert, wie zum Beispiel ein einzelnes Wasserstoff- oder Halogenatom. Die Regeln untersagen auch das Schneiden von Bindungen, wenn eines der resultierenden Fragmente eine kleine Nitro- oder Hydroxygruppe ist. Dies bedeutet im Umkehrschluss, dass während der Enumeration nicht alle Ausgangsmoleküle generiert werden konnten. Das Ergebnis der Schneideprozedur waren Fragmente, von denen 75% ein, 22% zwei, 2,5% drei und 0,5% vier Link-Atome hatten. Es wurde kein Fragment mit mehr als vier Link-Atomen generiert. Tabelle 6.1 listet die Ergebnisse und einige Eigenschaften der generierten Fragmenträume auf. Wie aus der Tabelle hervorgeht, besteht ein großer Unterschied zwischen den verschiedenen Target-Klassen. Die Anzahl der Inhibitoren reicht von 11 (COMT) bis 475 (EGFr), die daraus generierten Fragmente von 4 (RXRa) bis 271 (COX-2). Fast alle generierten Fragmenträume sind grundsätzlich unendlich in ihrer Größe. Das heißt, würden diese Fragmenträume ohne Einschränkungen enumeriert werden, würde die Enumeration unendlich viele Moleküle produzieren. Das ist nicht wirklich verwunderlich, da schon ein Fragment mit mindestens zwei kompatiblen Link-Typen reicht, um einen unendlichen Fragmentraum zu generieren. Es zeigt aber auch, dass für Anwendungen in der Praxis Beschränkungen für die Enumeration notwendig sind. Wird zum Beispiel gefordert, dass ein Molekül aus maximal fünf atomaren Fragmenten zusammengesetzt sein darf, was einer plausiblen Regel für Wirkstoffe entspricht, ergeben sich die Anzahlen pro Fragmentraum, die in Tabelle 6.3, in der Spalte *Theoretische Anzahl von Molekülen* abgebildet sind. Die letzte Spalte in Tabelle 6.1 gibt die Anzahl der Komponenten pro generiertem Fragmentraum wieder. Eine Komponente eines Fragmentraums ist analog zu einer Zusammenhangskomponente[101] in einem Graphen. Hat ein Fragmentraum eine Komponente, so können aufgrund der Regeln alle atomaren Fragmente in einem zusammengesetzten Fragment vorhanden sein. Hat ein Fragmentraum hingegen zwei Komponenten, so gibt es mindestens zwei Fragmente, die aufgrund der Regeln nicht im selben zusammengesetzten Fragment vorkommen können. Insbesondere kleinere Inhibitoren Mengen zerfallen in viele Komponenten. Per Definition sind dies zwar noch Fragmenträume, aber für praktikable Zwecke nicht geeignet. Daher wurden kleine Fragmenträume mit vielen Komponenten nicht weiter betrachtet, insbesondere, da die geschätzte Anzahl der Moleküle mit bis zu fünf Fragmenten sehr klein war: ADA, AmpC, COMT, MR, PNP, PR und RXRa.

Target-Klasse	Inhib.	Frag.	Größe	Komp.
ACE	49	45	$\infty$	2
AChE	107	100	$\infty$	2
ADA	39	25	endl.	12
ALR2	26	17	$\infty$	4
AmpC	21	26	endl.	12
AR	79	15	$\infty$	4
CDK2	72	62	$\infty$	4
COMT	11	7	endl.	12
COX-1	25	25	$\infty$	5
COX-2	426	271	$\infty$	1
DHFR	410	147	$\infty$	3
EGFr	475	125	$\infty$	2
ER <sub>Agonist</sub>	67	29	$\infty$	4
ER <sub>Antagonist</sub>	39	51	$\infty$	3
FGFr	120	106	$\infty$	2
FXa	146	189	$\infty$	1
GART	40	23	$\infty$	3
GPB	52	24	$\infty$	2
GR	78	57	$\infty$	5
HIVPR	62	70	$\infty$	2
HIVRT	43	40	$\infty$	2
HMGA	35	31	$\infty$	1
HSP90	37	26	$\infty$	4
InhA	86	92	$\infty$	1
MR	15	3	endl.	12
NA	49	26	$\infty$	5
P38 MAP	454	194	$\infty$	2
PARP	35	20	$\infty$	5
PDE5	88	93	$\infty$	2
PDGFr <sub>b</sub>	170	141	$\infty$	2
PNP	50	27	endl.	5
PPAR <sub>g</sub>	85	90	$\infty$	4
PR	27	26	endl.	9
RXR <sub>a</sub>	20	4	$\infty$	5
SAHH	33	27	$\infty$	5
SRC	159	122	$\infty$	2
Thrombin	72	87	$\infty$	2
TK	22	23	$\infty$	5
Trypsin	49	58	$\infty$	2
VEGFr <sub>2</sub>	88	118	$\infty$	1
Alle Klassen	3961	1942	$\infty$	1

Tabelle 6.1: Quantitative Ergebnisse und Eigenschaften generierter Fragmenträume. *Inhib.* steht für die Anzahl der verwendeten Inhibitoren pro Target-Klasse. *Frag.* für die Anzahl der aus den Inhibitoren generierten Fragmente. *Größe* spiegelt die potentielle Anzahl der Moleküle im generierten Fragmentraum wieder und *Komp.* steht für die Anzahl der Komponenten, in die ein Fragmentraum zerfällt. Gibt es nur eine Komponente, so erlauben es die Regeln und Fragmente eines Fragmentraums, dass in einem zusammengesetzten Fragment jedes Paar von atomaren Fragmenten vorkommen kann.

In jeder Inhibitorenklasse wurde das Minimum und Maximum folgender physikochemischer Eigenschaften<sup>2</sup> bestimmt: molekulares Gewicht, rotierbare Bindungen, Anzahl der Ringe, Wasserstoffbrücken-Akzeptoren, Wasserstoffbrücken-Donoren und  $c \log P$ .

Target-Klasse	Mol. Gewicht	Rot. Bind.	Ringe	H Akz.	H Don.	$c \log P$
ACE	176,2 - 516,3	3 - 12	0 - 3	4 - 9	0 - 4	-0,2 - 4,2
AChE	166,2 - 482,6	0 - 21	0 - 5	2 - 8	0 - 6	1,1 - 6,0
ALR2	225,2 - 449,2	0 - 8	1 - 4	2 - 7	0 - 4	-1,2 - 6,2
AR	256,4 - 556,5	1 - 9	1 - 5	1 - 8	0 - 3	2,0 - 10,0
CDK2	235,2 - 483,6	0 - 14	2 - 8	4 - 10	1 - 7	-1,1 - 6,2
COX-1	137,1 - 371,4	1 - 11	1 - 3	2 - 6	0 - 4	1,4 - 5,6
COX-2	243,3 - 571,1	0 - 12	2 - 6	0 - 8	0 - 7	1,7 - 9,2
DHFR	206,3 - 521,4	0 - 11	2 - 5	4 - 13	4 - 8	-0,4 - 5,7
EGFr	221,3 - 597,5	0 - 12	2 - 5	2 - 10	0 - 8	-2,6 - 7,3
ER <sub>Agonist</sub>	212,2 - 356,4	1 - 11	2 - 5	1 - 6	0 - 4	2,0 - 7,2
ER <sub>Antagonist</sub>	347,5 - 508,7	5 - 12	3 - 6	2 - 5	1 - 3	5,1 - 8,8
FGFr	281,3 - 597,5	0 - 14	3 - 5	4 - 10	1 - 5	0,7 - 7,5
FXa	293,4 - 574,4	1 - 13	2 - 7	4 - 12	1 - 12	-1,9 - 6,7
GART	429,4 - 479,5	7 - 11	2 - 3	11 - 14	4 - 7	-2,7 - 3,6
GPB	180,2 - 424,9	4 - 11	1 - 4	4 - 11	0 - 8	-6,2 - 4,3
GR	271,4 - 460,6	3 - 11	2 - 5	2 - 6	0 - 3	1,1 - 8,8
HIVPR	362,5 - 631,8	5 - 16	2 - 7	3 - 10	0 - 7	2,0 - 8,2
HIVRT	244,3 - 584,3	1 - 13	1 - 5	2 - 9	0 - 4	1,3 - 8,6
HMGA	256,3 - 557,6	3 - 20	2 - 4	3 - 9	0 - 3	2,2 - 6,7
HSP90	296,3 - 433,9	3 - 13	3 - 4	5 - 8	2 - 7	1,7 - 5,0
InhA	197,2 - 646,3	0 - 12	1 - 7	2 - 9	0 - 4	0,8 - 6,9
NA	193,2 - 408,5	1 - 14	1 - 3	5 - 11	1 - 9	-7,2 - 2,4
P38 MAP	239,2 - 509,6	0 - 10	2 - 5	2 - 8	0 - 5	0,3 - 7,8
PARP	136,2 - 366,5	0 - 4	1 - 5	3 - 5	1 - 4	0,3 - 6,2
PDE5	271,3 - 620,7	1 - 14	2 - 7	5 - 12	0 - 4	1,4 - 7,4
PDGFr <sub>b</sub>	210,2 - 597,5	0 - 12	3 - 5	1 - 10	0 - 5	0,2 - 7,5
PPAR <sub>g</sub>	264,3 - 640,7	2 - 15	2 - 6	3 - 10	0 - 3	1,0 - 10,4
SAHH	233,2 - 321,4	3 - 7	2 - 3	6 - 9	4 - 6	-3,8 - -0,9
SRC	226,2 - 597,5	0 - 13	3 - 5	4 - 10	0 - 5	1,6 - 7,5
Thrombin	250,3 - 647,8	2 - 17	2 - 6	3 - 12	1 - 9	-0,4 - 5,0
TK	184,2 - 405,2	3 - 7	1 - 3	5 - 9	2 - 5	-2,2 - 2,0
Trypsin	124,2 - 620,9	3 - 17	1 - 6	2 - 11	5 - 15	-0,3 - 4,7
VEGFr <sub>2</sub>	238,3 - 692,7	0 - 14	2 - 7	2 - 14	0 - 6	1,6 - 8,0
Alle Klassen	122,1 - 692,7	0 - 21	0 - 8	0 - 14	0 - 15	-7,2 - 10,4

Tabelle 6.2: Target-Klassen und zugehörige physikochemische Intervalle, die für die Enumerierung verwendet werden.

<sup>2</sup>Diese Eigenschaften wurden von den Inhibitoren, nicht den generierten Fragmenten, abgeleitet.

Die Berechnung von Wasserstoffdonoren und Akzeptoren wurde gemäß der Beschreibung von Lipinski et al. durchgeführt[7]. Die Anzahl der Ringe wurde über die Anzahl der Ringschlüsse bestimmt: Anzahl Bindungen - Anzahl Atome + 1.  $c \log P$  wurde über den Deskriptor von Wildman und Crippen[102] berechnet. Die Werte für die einzelnen Klassen sind in Tabelle 6.2 abgebildet.

## 6.2 Ergebnisse der Enumeration

Die Ergebnisse der verschiedenen Target-Klassen zeigen ein sehr inhomogenes Bild. Im Folgenden werden die Ergebnisse zunächst rein quantitativ analysiert. Darauf folgend werden die Ergebnisse mit bekannten Inhibitoren verglichen und zum Schluss werden einige ausgewählte Ergebnisse der Enumeration dargestellt.

### 6.2.1 Quantitative Analyse

Als erstes fällt beim Blick in Tabelle 6.3 auf, dass bei einigen Klassen ein *NB* in der Zeile *Enumerierte Moleküle* steht. NB steht für "Nicht Beendet" und bedeutet, dass die Enumeration gestoppt wurde, bevor alle Moleküle mit den gewünschten Eigenschaften enumeriert werden konnten. Enumerationen wurden gestoppt, sobald eine Million Fragmente generiert wurden. Dies ist einzig dem Umstand endlichen Hauptspeichers geschuldet. Um einen effizienten Zugriff auf schon generierte Moleküle zu erlauben, wurde die unique SMILES Repräsentation im Hauptspeicher gehalten. Das entspricht bei einem Rechner mit 4GB ungefähr einer Million Moleküle. Das Programm FRAGENUM hat grundsätzlich keine Beschränkung bezüglich der generierten Moleküle, muss aber, wie jedes andere Programm, mit endlichen Ressourcen auskommen.

Die *Theoretische Anzahl von Molekülen* in Tabelle 6.3 ist eine Abschätzung der Menge von Molekülen mit bis zu fünf Fragmenten eines Fragmentraums. Diese Zahl ist interessant, da Moleküle mit bis zu fünf Fragmenten im Bereich des molekularen Gewichts für oral verfügbare Wirkstoffe liegen[7] und somit eine plausible Obergrenze für pharmazeutische interessante Moleküle darstellt. Zunächst fällt auf, dass diese Zahl sehr gut mit der Anzahl der generierten Moleküle in Tabelle 6.1 korreliert. Dies ist nicht weiter verwunderlich, da es intuitiv erscheint, dass mehr Fragmente zu mehr Molekülen führen. Interessanter ist der Vergleich zwischen *Enumerierte Moleküle* und *Theoretische Anzahl von Molekülen* (der Prozentwert dieser beiden ist in Spalte *Anteil %* zu finden). Es fällt auf, dass diese beiden Werte praktisch gar nicht miteinander korrelieren. Eine Konsequenz daraus ist, dass einige relativ kleine Räume gar nicht enumeriert werden konnten, wie zum Beispiel ER<sub>Antagonist</sub> oder HIVPR mit 51 beziehungsweise 70 Fragmenten, wohingegen Räume mit mehr Fragmenten enumeriert werden konnten, beispielsweise PDGFr<sub>b</sub> mit 141 Fragmenten. In diesem Zusammenhang ist auch interessant, wieviel Prozent der theoretischen Moleküle enumeriert wurden. Grundsätzlich zeigt sich hier, dass für fast alle Räume

weniger als 10% der theoretischen Moleküle enumeriert wurden und für viele Räume der Anteil sogar im unteren einstelligen Prozentbereich liegt.

Target-Klasse	Enumerierte Moleküle	Theoretische Anzahl von Molekülen	Anteil [%]	Laufzeit [Sek]	[Mols/Sek]
ALR2	21	1.06e+04	0,2	1	21
AR	37	9.40e+01	39,4	1	37
NA	56	9.71e+03	0,6	2	28
PARP	102	1.06e+04	1,0	3	34
SAHH	128	4.54e+03	2,8	2	64
GART	188	3.70e+03	5,1	4	47
COX-1	298	9.60e+03	3,1	9	33
TK	416	2.93e+04	1,4	18	23
GR	752	1.74e+04	4,3	23	33
GPB	940	5.96e+03	15,8	27	35
ER <sub>Agonist</sub>	1387	1.61e+06	0,1	31	45
HIVRT	2666	4.68e+04	5,7	238	11
ACE	5604	2.38e+06	0,2	159	35
HSP90	5839	2.28e+06	0,3	560	10
CDK2	7610	1.60e+05	4,8	259	29
Trypsin	23716	3.07e+05	7,7	4816	5
HMGA	27264	1.14e+06	2,4	5102	5
DHFR	36691	9.82e+05	3,7	3017	12
FGFr	84608	2.66e+06	3,2	25079	3
SRC	105529	3.47e+06	3,0	35553	3
AChE	136025	8.98e+06	1,5	26551	5
PDGFr <sub>b</sub>	205119	7.46e+06	2,7	33175	6
EGFr	396085	1.29e+07	3,1	30505	13
InhA	544705	3.45e+06	15,8	227866	2
Thrombin	779213	5.23e+06	14,9	212086	4
COX-2	NB	8.99e+11	-	-	-
ER <sub>Antagonist</sub>	NB	3.44e+07	-	-	-
FXa	NB	1.61e+09	-	-	-
HIVPR	NB	1.07e+09	-	-	-
P38 MAP	NB	5.78e+10	-	-	-
PDE5	NB	9.15e+07	-	-	-
PPAR <sub>g</sub>	NB	1.86e+08	-	-	-
VEGFr <sub>2</sub>	NB	9.30e+07	-	-	-
Alle Klassen	NB	5.38e+14	-	-	-

Tabelle 6.3: Quantitative Werte der Enumerierung.

Die Tabelle ist nach Anzahl der enumerierten Moleküle sortiert.



Dies bedeutet eine Reduzierung um eine beziehungsweise zwei Größenordnungen und zeigt das Potential des Enumerators. Es soll nochmal darauf hingewiesen werden, dass diese Reduzierung schon während der Enumeration erreicht wird und somit sehr viel Rechenzeit eingespart werden konnte.

Die Laufzeiten der Enumerationen liegen zwischen wenigen Sekunden und zweieinhalb Tagen. Wie erwartet korreliert die Laufzeit mit der Anzahl der enumerierten Moleküle. Interessanter dürfte die Anzahl der eindeutig enumerierten Moleküle pro Sekunde sein, sozusagen die Effektivität des Enumerators. Die Werte schwanken zwischen  $45 \frac{\text{Moleküle}}{\text{s}}$  und  $2 \frac{\text{Moleküle}}{\text{s}}$ . Grundsätzlich ist aufgrund des Enumerations-Algorithmus zu erwarten, dass die Effektivität der Enumeration mit der Anzahl der enumerierten Moleküle sinkt. Dies ist darauf zurückzuführen, dass der Topologie-Redundanzfilter nicht alle Redundanzen erkennen kann und, dass jedes neu generierte Molekül auf Gleichheit mit allen zuvor generierten Molekülen getestet werden muss.

Die Ergebnisse lassen sich grob in drei Gruppen einteilen. Enumerationen, die bis zu einige hundert Moleküle produzierten und deren Laufzeiten im Bereich von Sekunden bis wenigen Minuten liegen: ALR2 bis CDK2 in Tabelle 6.3. Die zweite Gruppe bilden Enumerationen, die mehrere tausend Moleküle produzierten und innerhalb einer Stunde abgeschlossen waren: Trypsin, HMGA und DHFR. Die dritte und letzte Gruppe stellen Räume dar, deren Enumeration mehrere zehntausend bis hunderttausend Moleküle generiert und deren Laufzeiten sich im Bereich von mehreren Stunden bis Tagen bewegten oder aufgrund der Größe gar nicht abgeschlossen werden konnten. Das Potential des hier vorgestellten Enumerators zeigt sich besonders in den ersten beiden Gruppen, die die Hälfte der Fälle repräsentieren. Die unterliegenden chemischen Räume dieser Gruppen konnten in unter einer Stunde eigenschaftsbasiert enumeriert werden. Durch die Verwendung von Minimal-Maximal Intervallen für physikochemische Eigenschaften, musste auch nur ein Bruchteil der theoretisch interessanten Moleküle enumeriert werden. Dies erlaubt eine Fokussierung auf einige wenige tausend Moleküle, deren physikochemische Eigenschaften ein gewünschtes Profil aufweisen. Wird die Anzahl der enumerierten Moleküle pro Sekunde betrachtet, so zeigt sich, dass der Enumerator auch hier in den beiden ersten Gruppen seine höchste Effizienz erreicht.

Um einen transparenten Vergleich zwischen den Ergebnissen zu ermöglichen, wurden alle physikochemischen Eigenschaften für die Enumerationen automatisch, das heißt, ohne Zuhilfenahme weiteren Expertenwissens der Target-Klasse, aus den korrespondierenden Inhibitoren extrahiert. In einer realistischen Anwendung würde höchstwahrscheinlich weiteres Wissen in die Festlegung der physikochemischen Intervalle fließen. Auch könnten die hier nicht enumerierbaren Räume sicherlich zugänglich gemacht werden, indem Intervalle restriktiver gewählt werden, die maximale Anzahl der Fragmente pro Molekül reduziert oder Fragmente aus den zugrundeliegenden Räumen entfernt werden.

Diese erste Analyse zeigt, dass es möglich ist, mithilfe von nur automatisch erzeugten physikochemischen Intervallen, erste wertvolle quantitative Informationen über die Fragmenträume einzelner Target-Klassen zu erhalten. Es wurden aber noch keine Aussagen über die einzelnen Moleküle gemacht. Der nächste Abschnitt

vergleicht die generierten Moleküle der einzelnen Target-Klassen mit einer Menge von pharmazeutischen Referenzmolekülen, um dadurch einen besseren Einblick in die pharmazeutische Relevanz der enumerierten Moleküle zu ermöglichen.

### 6.2.2 Ähnlichkeitsvergleiche

Um eine Aussage über die topologische Qualität der generierten Moleküle zu ermöglichen, wurde eine Analyse mithilfe des FTREE Ähnlichkeitsmaß[49] durchgeführt. FTREE bewertet Ähnlichkeit gemäß der relativen Anordnung von funktionalen Gruppen. Gemeinsame Substrukturen spielen eine untergeordnetere Rolle. Dies macht dieses Ähnlichkeitsmaß besonders interessant für Moleküle, die aus Fragmenträumen generiert wurden. Für die Berechnung der Ähnlichkeit wurde für jedes Molekül die Ähnlichkeit zu jedem Molekül der entsprechenden Target-Klassen berechnet und der höchste Wert als Ähnlichkeit verwendet. Diese wurden dann als Verteilung in einem Graphen aufgetragen. Es soll hier nochmal darauf hingewiesen werden, dass aufgrund der Schneide- und Filterregeln nicht zu erwarten ist, dass alle Inhibitoren reproduziert werden können. Dies hatte zur Folge, dass nicht für alle Inhibitoren ein Molekül mit dem Ähnlichkeitswert 1 gefunden werden konnte.

Um einen Vergleich zu den generierten Verteilungen zu haben, wurde eine Referenz-Ähnlichkeitsverteilung generiert. Dazu wurde ein Datensatz aus 7.528 zufällig ausgewählten, wirkstoffartigen Molekülen aus dem WDI und 452 bekannte Inhibitoren verwendet[103]. Aus proprietären Gründen konnten nur 398 der 452 Inhibitoren verwendet werden. Für jedes Molekül der Referenzmenge wurde dann die Paar-Ähnlichkeit zu allen Inhibitoren aller Enumerations-Target-Klassen berechnet und wieder die höchste Ähnlichkeit verwendet. Für die grafische Repräsentation wird die Enumerationsverteilung durch einen schwarzen, geglätteten Graphen und die Referenzverteilung durch einen grau hinterlegten, geglätteten Graphen dargestellt.

Werden die Verteilungen der Enumerationen betrachtet, fällt zunächst auf, dass alle eine relativ hohe FTREE-Ähnlichkeit aufweisen. Praktisch alle Enumerationsverteilungen haben ihr Maximum bei mehr als 0,85, was einer relativ hohen topologischen Ähnlichkeit entspricht. Bei einer genaueren Betrachtung fällt weiter auf, dass sich die Verteilungen wieder in drei Gruppen gliedern lassen. Die erste Gruppe repräsentiert die Verteilungen der Enumerationen, die weniger als zehntausend Moleküle produzierten: ALR2 bis CDK2 in Tabelle 6.3. In dieser Gruppe schwanken die Verteilungen relativ stark, was zum Teil auf die relativ kleine Menge von Molekülen zurückgeführt werden kann. Es zeigt sich aber, dass in dieser Gruppe die Verteilung im Allgemeinen eine höhere Ähnlichkeit aufweist als die Referenzverteilung. Die zweite Gruppe besteht aus Enumerationen mit wenigen zehntausend Molekülen und beinhaltet die Target-Klassen Trypsin bis SRC in Tabelle 6.3. In dieser Gruppe schwanken die Verteilungen nicht mehr ganz so stark wie in der ersten Gruppe und sind relativ ähnlich zu der Referenzverteilung. Die letzte Gruppe beinhaltet die Enumerationen mit mehreren zehntausend bis hunderttausend Molekülen: HMGA bis Thrombin. In dieser Gruppe sind die Enumerationsverteilungen viel glatter und haben im Wesentlichen die Form der Referenzverteilung, sind jedoch in Richtung geringerer Ähnlichkeit verschoben.

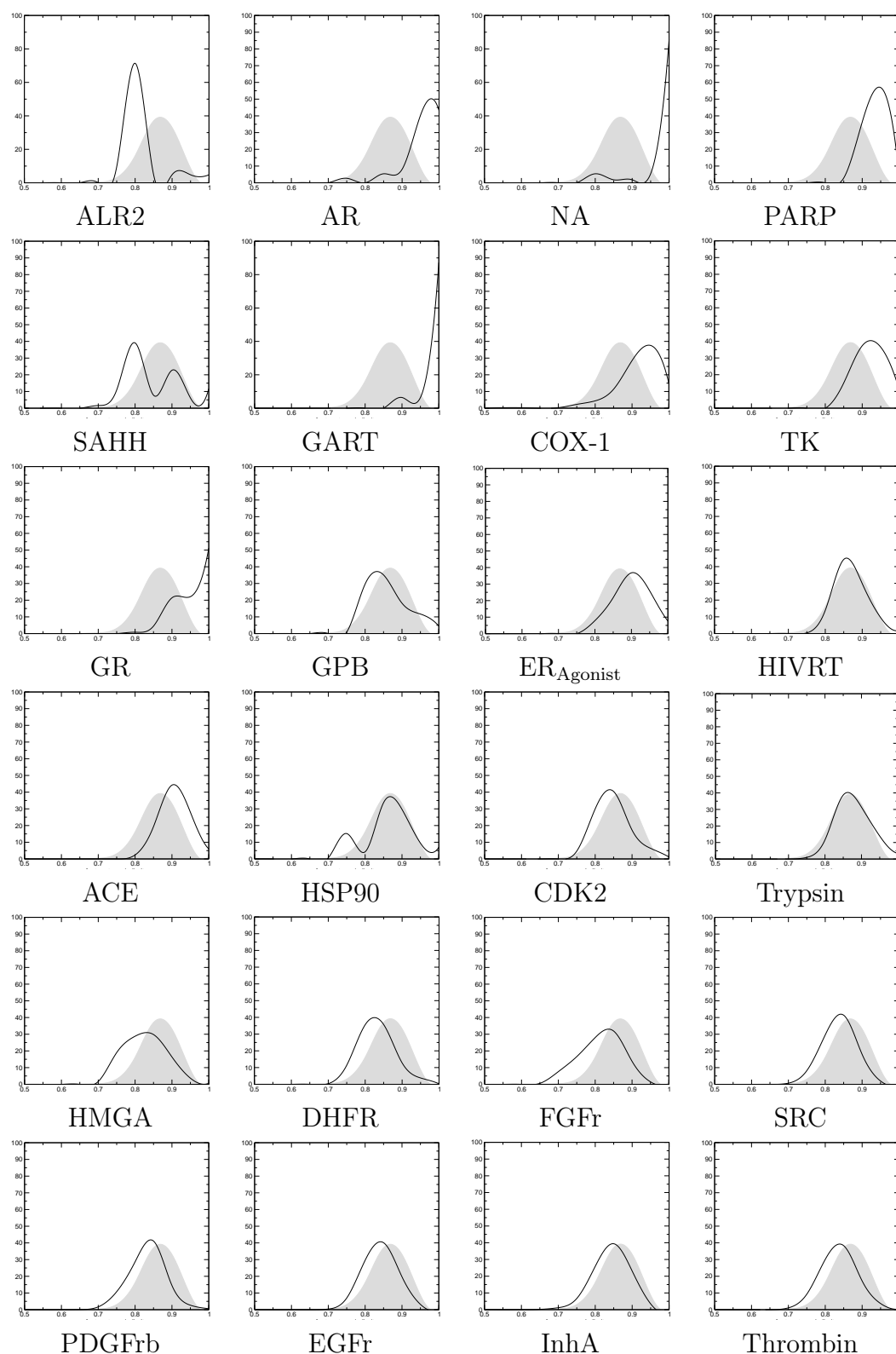


Tabelle 6.4: FTREE Ähnlichkeitsverteilung für die verschiedenen Target-Klassen.  
 $x$ -Achse: FTREE Ähnlichkeit,  $y$ -Achse: Anzahl Moleküle in Prozent.  
 Schwarze Linie: Enumerierte Verteilung, grauer Bereich: Referenzverteilung.

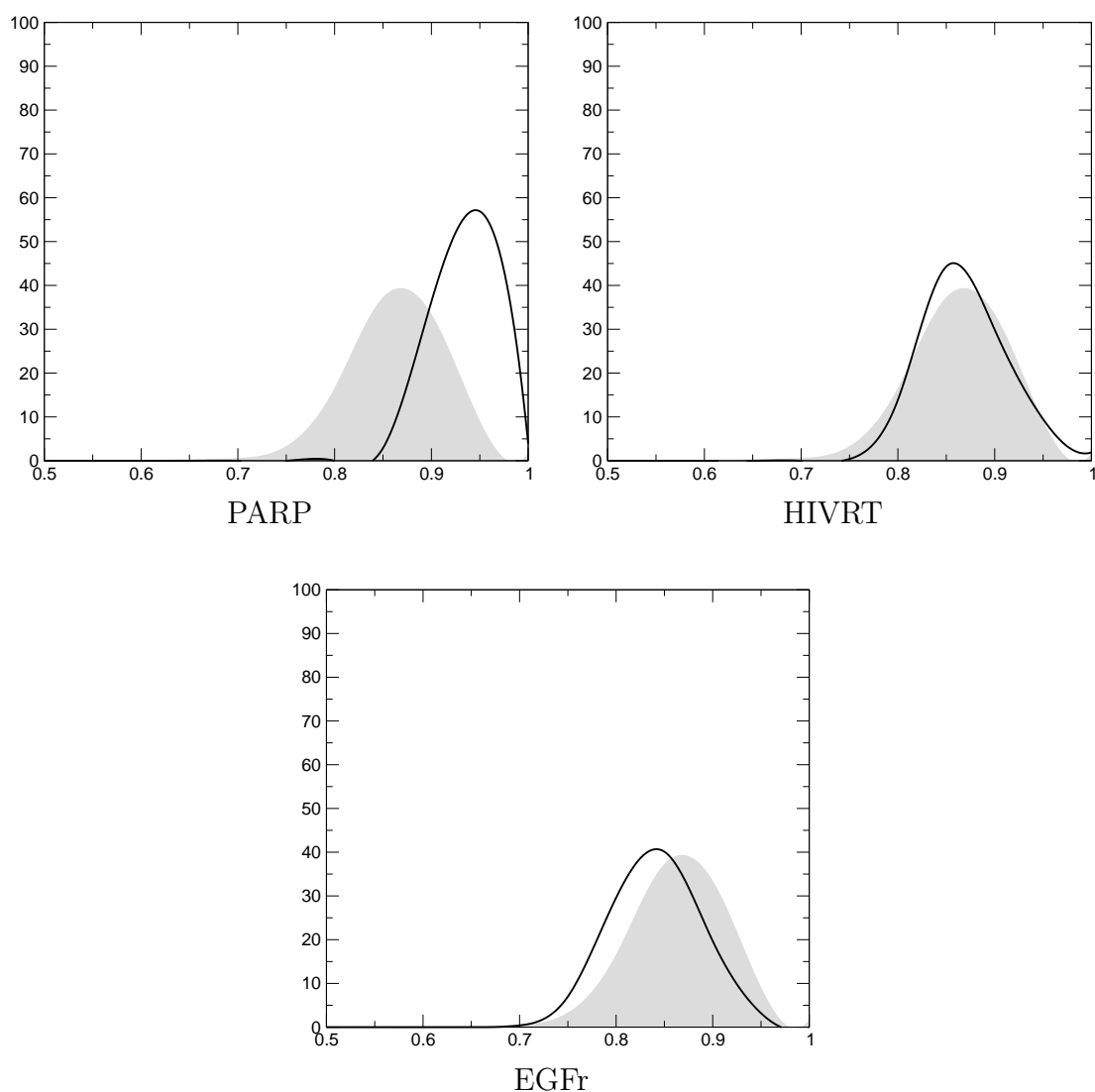


Tabelle 6.5: Repräsentanten drei unterschiedlicher Gruppen.

$x$ -Achse FTREE Ähnlichkeit,  $y$ -Achse: Anzahl Moleküle in Prozent.

Schwarze Linie: Enumerierte Verteilung, grauer Bereich: Referenzverteilung.

Die Verteilungen der Enumerationen lassen sich in drei Gruppen gliedern. Die erste Gruppe (PARP) enthält Räume mit wenigen hundert Molekülen und die Verteilung zeigt eine hohe Ähnlichkeit. Die zweite Gruppe (HIVRT) repräsentiert Enumerationen mit wenigen tausend Fragmenten und entspricht ungefähr der Referenzverteilung. Die dritte Gruppe (EGFr) besteht aus Enumerationen mit bis zu mehreren hunderttausend Molekülen und zeigt eine Verschiebung in den unteren Ähnlichkeitsbereich gegenüber der Referenzverteilung.

Das Interessante ist, dass diese drei Gruppen sich ungefähr mit den drei Gruppen aus der quantitativen Betrachtung decken. Somit eignen sich die ersten beiden Gruppen, um relativ schnell möglichst viele ähnliche Moleküle zu einer Gruppe von Molekülen zu finden. Die Varianz in den Verteilungen lässt wohl auf viele Ursachen zurückführen. Zwei Faktoren spielen aber sicherlich eine entscheidende Rolle: Zum einen hat die Anzahl der Moleküle in einer Verteilung einen großen Einfluss auf diese. Bei einer kleinen Menge können wenige Ausreißer eine ganze Verteilung verschieben. Zum anderen sind die Unterschiede sicherlich auch auf die einzelnen Target-Klassen und die resultierenden Fragmenträume zurückzuführen. Alle Fragmenträume wurden zwar mit denselben Regeln erstellt, dies bedeutet aber nicht, dass die resultierenden Fragmentmengen immer gleich sind. Einige Target-Klassen besitzen sehr spezifische Struktur motive. Werden diese in den Fragmenten erhalten, können Moleküle mit hoher Ähnlichkeit enumeriert werden. Wird diese Spezifität nicht in den Schneideregeln berücksichtigt, können gegebenenfalls keine ähnlichen Moleküle rekonstruiert werden. Besitzt eine Target-Klasse keine spezifischen Struktur motive und enthalten die Fragmenträume alle relevanten Motive, ist davon auszugehen, dass eine Enumeration ein sehr breites Spektrum von Molekülen generieren wird. Dies ist ja genau das Ziel der Enumeration. Dies hat aber zur Folge, dass die Ähnlichkeit zu den Inhibitoren sinkt.

Dieser, oberflächlich betrachteter, Nachteil ist der eigentliche Vorteil des Enumerators. Er wird immer strukturell ähnliche Moleküle zu allen Inhibitoren generieren, wenn es Fragmente, Regeln und Profil zulassen. Diese Ähnlichkeit wird aber nicht erreicht, weil explizit nach ihr gesucht wird, sie wird vielmehr erreicht, weil alle Moleküle generiert werden. Tabelle 6.1 zeigt einige Beispiele von Inhibitoren und enumerierten Molekülen mit einer FTREE-Ähnlichkeit größer als 0,9. Jedoch sind andere Programme im Bereich der Ähnlichkeitssuche sehr viel effektiver. Als Beispiel sei hier FTREE-FS genannt. Das eigentliche Einsatzgebiet des Enumerators ist mehr im vollständigen Explorieren von chemischen Unterräumen zu sehen. Wie in Kapitel 1 dargestellt und Ergebnisse des Programms GDB-*n*[65] zeigen, wurde bisher in der pharmazeutischen Chemie nur ein sehr kleiner Teil des höchstwahrscheinlich pharmazeutisch relevanten Raums betrachtet. Mit FRAGENUM besteht nun die Möglichkeit, die Räume komplett zu betrachten. Da die Räume sehr groß sind, kann FRAGENUM als Molekülproduzent für andere Programme eingesetzt werden, wie zum Beispiel Programme für das virtuelle *High-Throughput Screening*[104, 105, 106]. Das Programm kann auch als *Ideengenerierer* eingesetzt werden. Da es alle pharmazeutisch relevanten Moleküle eines Fragmentraums generieren kann, können sich Pharmazeuten oder Medizinal-Chemiker stichprobenartig die Ergebnisse ansehen und eventuell eine neue Idee für Wirkstoffe bekommen. Dies trifft insbesondere für Enumerationen zu, die, wie die ersten beiden Gruppen, nicht ganz so viele Moleküle generieren und relativ schnell durchgeführt werden können. Die Beispiele in Tabelle 6.1 zeigen auch dafür das Potential von FRAGENUM. Die Moleküle weisen zwar alle ein ähnliches Gerüst auf, daher die relativ hohe Ähnlichkeit von mehr als 0,9, aber funktionale Gruppen sind verschoben, entfernt oder hinzugefügt worden.

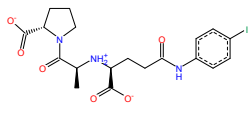
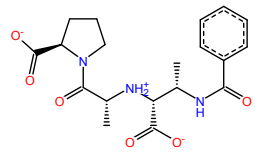
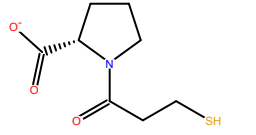
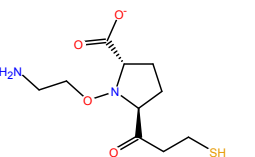
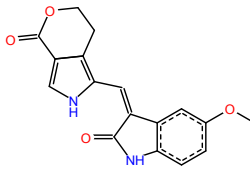
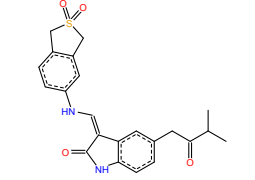
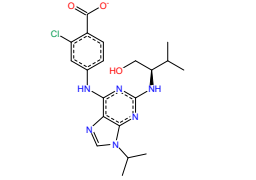
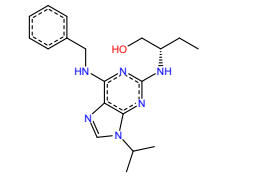
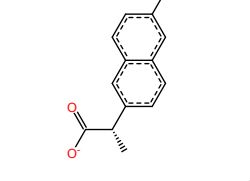
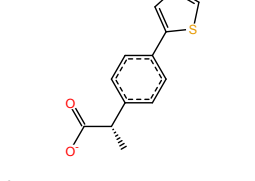
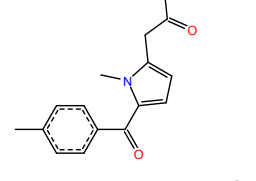
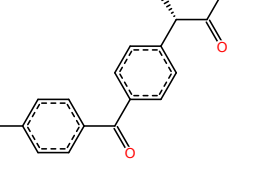
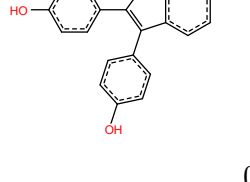
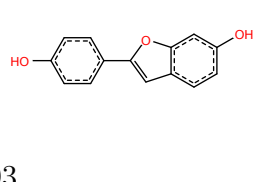
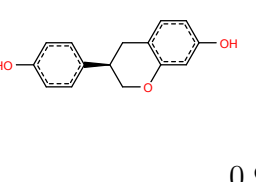
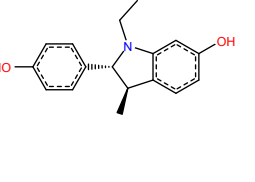
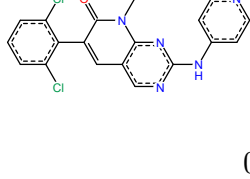
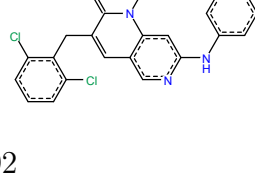
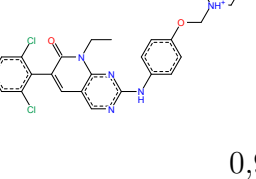
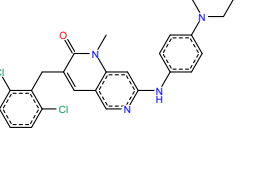
Inhibitor	Molekül	Inhibitor	Molekül
			
0,95		0,93	
			
0,90		0,95	
			
0,92		0,93	
			
0,93		0,95	
			
0,92		0,92	

Abbildung 6.1: Ausgewählte Beispiele für Inhibitor und ein enumeriertes Molekül. Abgebildete Target-Klassen sind, von oben nach unten, ACE, CDK2, COX-1, ER<sub>Agonist</sub> und SRC. Die Zahl unter jedem Paar steht für die FTREE-Ähnlichkeit.

## 7 Zusammenfassung

Diese Arbeit befasste sich mit Fragmenträumen und Methoden, um in diesen zu navigieren. Dies ist unter anderem dadurch motiviert, dass sich oral verfügbare Arzneimittel in bestimmten Regionen des chemischen Raums häufen.

Um den chemischen Raum virtuell modellieren zu können, wurden in dieser Arbeit chemische Fragmenträume verwendet. Fragmenträume bestehen aus einer Menge von Molekül-Fragmenten und Regeln. Jedes Fragment hat mindestens ein sogenanntes Link-Atom mit einem zugehörigen Link-Typ. Die Regeln beschreiben, welche Link-Typen wie verbunden werden können. Diese Modellierung macht sie für Pharmazeuten, Chemiker und Informatiker zugleich interessant. Fragmente können pharmazeutisch relevante Motive beinhalten, Regeln können chemische Reaktionen modellieren und die kombinatorische Struktur lässt sich gut in computergestützten Verfahren verwenden.

Obwohl Fragmenträume schon häufig die Grundlage von Programmen im Bereich des Wirkstoffentwurfs waren, fehlte ihnen bisher eine formale Grundlage. Aus diesem Grund widmete sich der erste Schwerpunkt dieser Arbeit der mathematischen Modellierung von Fragmenträumen. Aufbauend auf der mathematischen Modellierung wurden Programme entwickelt, die es ermöglichen, Fragmenträume visuell und eigenschaftsbasiert zu explorieren.

Als erstes wurde das Programm FRAGVIEW entwickelt. Es dient als Werkzeug, um Fragmenträume visuell zu untersuchen und zu modifizieren. Fragmente, welche mithilfe von logischen Ausdrücken über physikochemische Eigenschaften ausgewählt werden können, werden als zweidimensionale Strukturdiagramme in einer tabellenkalkulationsähnlichen Oberfläche dargestellt. Die Darstellung kann nach physikochemischen Eigenschaften sortiert werden, welche sich beliebig ein- und ausblenden lassen. Zur Anpassung von Fragmenträumen lassen sich Fragmente modifizieren, entfernen oder aus einem anderen Raum hinzufügen. Neben den Fragmenten können auch die Regeln eines Fragmentraums dargestellt und modifiziert werden. Kompatibilitäten zwischen den verschiedenen Link-Typen werden in einer Dreiecksmatrix dargestellt. Eine Kompatibilität zwischen zwei Typen kann durch ein einfaches klicken in der entsprechenden Zeile und Spalte aktiviert beziehungsweise deaktiviert werden.

Das zweite Programm, FRAGENUM, wurde entwickelt, um bestimmte Regionen des chemischen Raums zu enumerieren. FRAGENUM erwartet die Beschreibung der gewünschten Region durch Minimal- und Maximalwerte für verschiedene physikochemische Eigenschaften und generiert dann alle Moleküle eines Fragmentraums, die dieses Profil besitzen. Aufgrund der potentiell unendlichen Anzahl von Molekülen in einem Fragmentraum, können nicht alle Moleküle generiert und dann darauf überprüft werden, ob sie das vorgegebene Profil erfüllen. Um eine effiziente eigenschaftsbasierte

Enumeration zu ermöglichen, muss schon während der Enumeration das vorgegebene Profil verwendet werden, um nur Fragmente auszuwählen, die potentiell in der Lage sind, das Profil zu erfüllen. Eine weitere Herausforderung bestand in der Vermeidung von redundanten Molekülen. Moleküle werden in FRAGENUM über Fragmentbäume modelliert, wobei Fragmente die Knoten und Verbindungen zwischen diesen die Kanten darstellen. Um Redundanzen während der Enumerierung zu vermeiden, wurde ein Filter entwickelt, der auf Baumtopologien operiert. Es konnte gezeigt werden, dass dieser fast alle redundanten Bäume vermeiden kann und somit erheblich zur Effizienz beiträgt.

Um die Möglichkeiten von FRAGENUM zu testen, wurde es in einem Anwendungsszenario eingesetzt. Dazu wurden Inhibitoren von 33 bekannten Target-Klassen aus einer Datenbank extrahiert und zu jeder Target-Klasse ein Fragmentraum generiert. Als Profil für die Enumeration jeder Target-Klasse wurden die Minimal- und Maximalwerte verschiedener physikochemischer Eigenschaften der jeweiligen Inhibitoren extrahiert. Es konnte gezeigt werden, dass FRAGENUM in der Lage war 25 Räume vollständig mit den jeweiligen Profilen zu enumerieren. Die generierten Moleküle jeder Target-Klasse wurden dann, mithilfe von FTREE, mit den entsprechenden Inhibitoren auf Ähnlichkeit verglichen. Es zeigte sich, dass die generierten Moleküle tendenziell hohe Ähnlichkeit zu ihren Inhibitoren besitzen, aber die Ähnlichkeitsverteilungen in den verschiedenen Klassen sehr unterschiedlich sind. Räume mit wenig Fragmenten zeigen tendenziell höhere Ähnlichkeiten als Räume mit vielen. Dies scheint intuitiv, da es bei mehr Fragmenten mehr Raum zu enumerieren gibt. Eine weitere Erklärung ist, dass einige Klassen sehr spezifische Strukturen besitzen. Werden Moleküle generiert, die dieselben physikochemischen Eigenschaften aufweisen, aber nicht über die spezifischen Strukturen verfügen, dann besitzen sie eine geringe FTREE Ähnlichkeit.

## 7.1 Ausblick

Eine natürliche Erweiterung für FRAGVIEW wäre die Anbindung an andere Fragmentraum-Programme, wie zum Beispiel FLEXNOVO oder FRAGENUM. Im Fall von FLEXNOVO könnte eventuell sogar ein Schritt weiter gegangen werden und Ergebnisse beziehungsweise Zwischenergebnisse mithilfe von POSEVIEW[107, 108, 109] visualisiert werden. POSEVIEW versucht die dreidimensionalen Protein-Ligand Interaktionen mithilfe von zweidimensionalen Struktur-Diagrammen darzustellen. POSEVIEW und FRAGVIEW bauen auf derselben Grafikbibliothek auf, so dass einer Integration grundsätzlich nichts im Wege stünde.

Erweiterungen für FRAGENUM lassen sich in zwei Klassen einteilen: Steigerung der Effizienz und neue Funktionen:



### 7.1.1 Steigerung der Effizienz von FragEnum

Die größte Einschränkung von FRAGENUM ist die Beschränkung auf den Hauptspeicher und auf einen Prozess. Lösungen für diese Einschränkungen bestehen in der Verwendung von Datenbanken und in der Parallelisierung des Enumerationsprozesses.

- **Datenbanken:**  
Um Moleküle auf Gleichheit prüfen zu können, verwaltet FRAGENUM alle Moleküle im Hauptspeicher. Dies ermöglicht zwar einen effizienten Vergleich, schränkt die Menge der generierbaren Moleküle aber drastisch ein. Es wäre daher interessant, FRAGENUM in Kombination mit Datenbanken zu verwenden. Denkbar wäre ein Mehrstufenmodell, in dem eine Menge von Molekülen im Hauptspeicher und der Rest in einer Datenbank verwaltet wird. Soll ein neues Molekül auf Gleichheit zu den bisher generierten getestet werden, würden erst die Moleküle im Hauptspeicher und, falls nötig, dann die in der Datenbank überprüft werden.
- **Threads:**  
Threads eines Programms haben den Vorteil, dass sie alle auf demselben Adressraum operieren. Diese würde es erlauben, dass mehrere Enumerations-Threads für die finale Eindeutigkeitsüberprüfung auf denselben AVL-Baum zugreifen (siehe 31) könnten. Es muss aber bedacht werden, dass bei Einfügeoperationen der Baum für andere Threads gesperrt werden muss. Es müsste daher empirisch bestimmt werden, ab wievielen Threads die Threads mehr Zeit mit Warten auf die Freigabe des Baums verbringen als mit der Enumeration. Eine Möglichkeit dies abzumildern würde darin bestehen, mehrere AVL-Bäume zu verwenden, die sich zum Beispiel bezüglich des  $n$ -ten Zeichens des generierten Unique-SMILES unterscheiden.
- **MapReduce:**  
*MapReduce*[110] ist ein von Google entwickeltes System zur Parallelisierung von Anwendungen. Um das System nutzen zu können, müssen zwei Funktionen implementiert werden, *Map* und *Reduce*. Beide Funktionen erwarten ein Schlüssel-Wert Tupel als Eingabe und die Ausgabe muss wieder ein Schlüssel-Wert Tupel sein. Ergebnisse der *Map* Prozesse dienen als Eingabe für die *Reduce* Prozesse. Im Zusammenhang mit dem Enumerator könnte *Map* dazu genutzt werden, um Bäume zu enumerieren und Unique-SMILES zu generieren und *Reduce* um die generierten Unique-SMILES auf Einzigartigkeit zu testen. Dabei würde ausgenutzt werden, dass alle Schlüssel-Wert Tupel mit demselben Schlüssel zur selben *Reduce* Funktion gesendet werden. Jede *Reduce* Funktion könnte somit ihren eigenen AVL-Baum mit Unique-SMILES verwalten und somit eine Parallelisierung des finalen Unique-Test ermöglichen<sup>1</sup>.

<sup>1</sup>Die *Reduce*-Funktionen werden erst aufgerufen, nachdem die ganze Eingabe verarbeitet wurde, also ein ganzer Raum vollständig enumeriert wurde. Um Speicherproblemen vorzubeugen, können mehrere Stufen von *MapReduce*-Abfolgen verwendet werden. *HadHoop*[111], eine *open-*

### 7.1.2 Neue Funktionen für FragEnum

- Neue Deskriptoren:  
In dieser Arbeit wurden nur einfache physikochemische Deskriptoren verwendet. FRAGENUM kann aber alle Deskriptoren, die additiv sind, effizient enumerieren. Zum Beispiel könnte ein Deskriptor verwendet werden, der topologische Distanzen zwischen bestimmten Atom-Typ Paaren verwaltet und somit auch strukturelle Informationen trägt. Beim Kombinieren könnten die Intra-Fragment Werte einfach aufaddiert werden. Damit Inter-Fragment Distanzen effizient berechnet werden können, müsste jedes Link-Atom die topologische Entfernung zu jedem Atom-Typ, ausgehend von sich selbst, abspeichern. Beim Kombinieren kann dann diese Information genutzt werden, um effizient Inter-Fragment Atom-Paar Entfernungen zu berechnen.
- Sampeln:  
Um schnell einen Einblick in Fragmenträume zu bekommen, wäre es vorteilhaft eine *repräsentative* Menge von Molekülen zu generieren, die möglichst alle Eigenschaften des Raums abdecken. Eine einfache Methode würde darin bestehen, Fragmente zufällig auszuwählen und diese dann zu Molekülen zu verschmelzen. Im Zusammenhang mit dem Enumerator könnte dies realisiert werden, in dem bei jeden Anbauschnitt zufällig  $n$  Fragmente aus der Kompatibilitätsmenge des aktuell betrachteten Link-Atoms ausgewählt werden.
- Negative und alternierende Eigenschaften:  
Um strikt negative Eigenschaften zu berücksichtigen, muss der Enumerator nur minimal angepasst werden. Bei dem Test in Zeile 7 des Algorithmus 5.7 müsste sich der Test auf **größer** auf den Absolutwert beziehen. Additive alternierende Eigenschaften, Eigenschaften die je nach Fragment negativ oder positiv sein können, lassen sich nicht effizient in die Enumeration einbeziehen. Der Test, ob eine Eigenschaft größer als die obere Grenze des vorgegebenen Intervalls ist, Algorithmus 5.7, Zeile 7, kann für alternierende Eigenschaften in dieser Form nicht angewendet werden. Eine Erweiterung des Tests wäre möglich, wenn die maximale Anzahl von Fragmenten für ein Molekül berücksichtigt wird. In einem Vorverarbeitungsschritt müsste zunächst für alternierende Eigenschaften der maximale negative als auch positive Wert über den zu enumerierenden Fragmentraum bestimmt werden. Wird der Einfachheit zunächst nur das Überschreiten der oberen Schranke betrachtet, muss berechnet werden, ob mit der verbleibenden Anzahl von Fragmenten, multipliziert mit dem maximalen negativen Wert, wieder ein Erreichen des vorgegebenen Intervalls möglich ist.

---

*source* Implementierung von *MapReduce* unterstützt zum Beispiel *Combine*-Funktionen, die im Wesentlichen lokale *Reduce*-Funktionen sind. Sie verarbeiten Ausgaben einer *Map*-Funktion zunächst lokal und senden ihr Ergebnis an die globalen *Reduce*-Funktionen.

# A Fragmentraum Regeln

Fragmentraum Regeln verwendet für das Anwendungsszenario in Kapitel 6.

```
# Fragment space file for FlexNovo/FragView
# -----
#
# syntax description:
# @link_types <nof links>
#   <link name 1> <link name 2> ...
#   <link name k> <link name k+1> ... <link name <nof links>>
# @fragment_files
#   < frag file 1>
#   < frag file 2>
#   :
# @link_terminals
#   <link name> <group> [<atom type> <bond type> <bond length> <torsion angle>]
#   :
# @link_connects
#   <name of> <name of> [<aatom type> <aatom type>] <bond> <bond> [<torsion>]
#   link 1    link 2    for link 1  for link 2  type  length  angle
#   :
# -----
```

```
@link_types 11
R1 R2 R3 R4 R5 R6 R7 R9 R10 R11 R12
```

```
@link_terminals
# link  group  aatom  bond  blen  Tangle
R1  [O-]  *      1     1.280  *
R2  [H]   *      1     1.009  *
R3  [H]   *      1     0.967  *
R4  [H]   *      1     1.090  *
R5  [H]   *      1     1.009  *
R6  [CH3] *      1     1.510  *
R7  [CH2] *      2     1.340  180.0
R9  [H]   *      1     0.970  *
R10 [H]   *      1     0.970  *
R11 [H]   *      1     1.080  *
R12 [CH3] *      1     1.780  *
```

```
@link_connects
# link1 link2 aatom1 aatom2 bond blen Tangle Comment
R1     R2    C.2    *      am   1.355 180.0
R1     R3    C.2    0.3    1     1.362 *
R2     R4    *      C.3    1     1.465 *
R2     R6    *      *      am   1.355 180.0
```

R2	R12	*	*	1	1.656	*
R3	R4	0.2	C.3	1	1.362	*
R4	R5	C.3	*	1	1.469	*
R4	R9	C.3	*	1	1.469	*
R4	R10	C.3	*	1	1.469	*
R7	R7	C.2	C.2	2	1.316	180.0
R11	R11	*	*	1	1.473	180.0

# B Implementierungsdetails

## B.1 FragView

Die Implementierung von FRAGVIEW erfolgte in C++[112]. Als Bibliothek für die grafische Oberfläche wurde Qt[113] verwendet. FRAGVIEW baut im Wesentlichen auf vier Entwurfsmuster[114] auf: Kompositum, Besucher, Beobachter und der Model-View-Controller Architektur<sup>1</sup>[115]. Das Kompositum Entwurfsmuster wird für die Verwaltung von Fragmenten und Fragmenträumen verwendet. Das Besucher Entwurfsmuster erlaubt es, Funktionen von den Strukturklassen zu trennen. Die Besucherklassen entsprechen somit dem *Controller* der Model-View-Controller Architektur. Benachrichtigungen über Veränderungen von Fragmenten oder Fragmenträumen werden über das Beobachter Entwurfsmuster an die Anzeige kommuniziert.

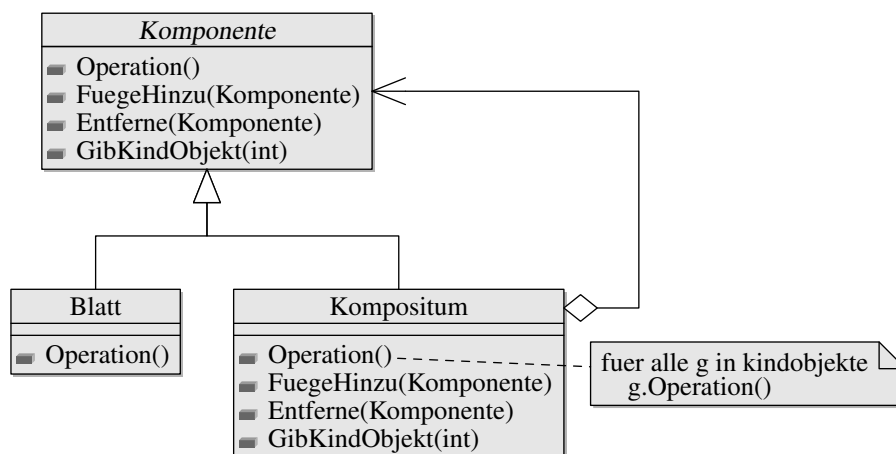


Abbildung B.1: Kompositum Klassendiagramm[114].

Das Kompositum Entwurfsmuster erlaubt es Komponenten mit gleicher Struktur rekursiv in einer Baumstruktur zu verwalten.

<sup>1</sup>Model-View-Controller wird eher als Architekturmuster angesehen, da es ein übergeordnetes Prinzip beschreibt ohne eine genauere Vorgabe für eine Implementierung.

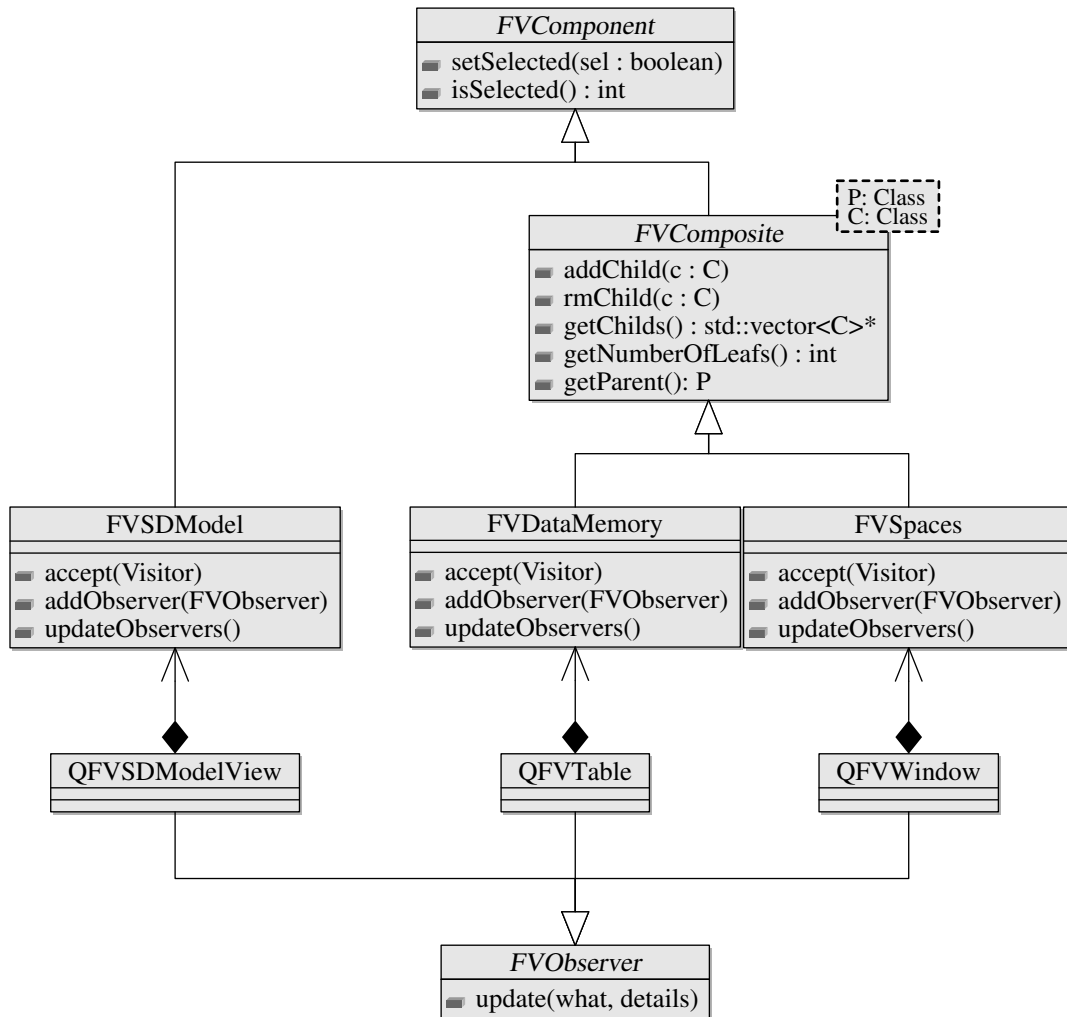


Abbildung B.2: Klassenstruktur von FRAGVIEW.

Das zugrundeliegende Entwurfsmuster für FRAGVIEW ist das Kompositum. Um Funktionen außerhalb der Klassenstruktur zu ermöglichen, wird das Besucher Entwurfsmuster verwendet. Die Daten und ihre Repräsentation wurden gemäß der Model-View-Controller Architektur getrennt. Die Klassennamen, die mit einem Q anfangen, repräsentieren die grafische Oberfläche (View). Die Kommunikation zwischen Model und View erfolgt über das Beobachter Entwurfsmuster.

Aufgrund von Implementierungsdetails wurde in einigen Fällen von der Architektur in Abbildung B.1 abgewichen, aber die Modellierung spiegelt sich eindeutig in der Implementierung wieder. Das Besucher Entwurfsmuster wurde mithilfe von C++ Templates auf der Basis der Loki Bibliothek[116] implementiert. Einer der Vorteile der gewählten Template Implementierung ist, dass keine *casts* durchgeführt werden müssen. Ein konkreter Besucher gibt alle Typen als Template Parameter in der Klassen-Definition an, die er besuchen will, stellt eine Implementierung einer `Visit`

Methode für jeden Typen zur Verfügung und der Compiler und die dynamische Bindung kümmern sich um den Rest (siehe Beispiel B.3). In der Implementierung

```
namespace fv
{
    class FVVReverseSelected : public Loki::BaseVisitor,
                               public Loki::Visitor<FVSpaces>,
                               public Loki::Visitor<FVDataMemory>,
                               public Loki::Visitor<FVSDModel>
    {
        public:
            void Visit(FVSpaces &spaces);
            void Visit(FVDataMemory &data);
            void Visit(FVSDModel &model);
    }; // class FVVSelection
} // namespace fv
```

Abbildung B.3: Beispiel eines Besuchers in FRAGVIEW.

Die Klasse gibt alle Typen an, die es besuchen wird und stellt für jeden Typ eine Implementierung bereit.

```
void FVVReverseSelected::Visit(FVDataMemory &data)
{
    std::vector<FVSDModel*>::iterator iter;
    for(iter = data.begin(); iter != data.end(); ++iter) {
        Visit(*iter);
    }
}

void FVVReverseSelected::Visit(FVSDModel &model)
{
    model.setSelected(!model.isSelected());
}
```

Abbildung B.4: Beispiel einer Implementierung eines Besuchers in FRAGVIEW.

Dieses Beispiel illustriert einen typischen Besucher. Innere Knoten iterieren meistens nur über ihre Kinder und meistens steckt die eigentliche Funktionalität nur in den Blättern.

iterieren alle Komponenten über ihre Kinder und rufen für jedes die `Visit` Methode auf (siehe Abbildung B.1). Da es für jeden Kind-Typen eine `Visit` Methode gibt, wird mittels dynamischer Bindung jeweils die entsprechende Methode aufgerufen. In den meisten Fällen iterieren innere Knoten nur über ihre Kinder und die Funktionalität

steckt nur in den Blatt-Konten des Kompositums Entwurfsmusters. Mithilfe dieser Strukturen ist es sehr einfach FRAGVIEW um weitere Funktionalitäten zu erweitern. In den meisten Fällen reicht es eine Besucher-Klasse zu implementieren, die die Kompositum Struktur traversiert und auf den Blatt-Knoten konkrete Operationen durchführt.

## B.2 FragEnum

Die Implementierung von FRAGENUM erfolgte in ANSI-C[117] und baut auf dem FLEXNOVO[1] Programmpaket auf, welches wiederum auf FLEXX[118] zurückgreift.

Die FRAGENUM Menustruktur, die in das bestehende FLEXX Programmpaket eingefügt wurde:

```
FRAGSPACE
-> READ      : Einlesen eines Fragmentraums.
FRAGENUM
-> PRINTPROP : Auflisten der enumerierbaren Eigenschaften und
                gesetzter min max Werte fuer diese.
-> SETPROP   : Setzen von min max Werten fuer Eigenschaften
-> OUTFILE   : Datei, in der die (Unique) SMILES der enumerierten
                Molekuele gespeichert werden.
-> ENUM      : Startet die Enumeration.
                Kann nur aufgerufen werden, nachdem ein Fragmentraum
                geladen wurde und eine Ausgabedatei angegeben wurde.
-> ESTI      : Schaetzt die Anzahl der zu enumerierenden Molekuele
                mit den gegebenen min max Werten.
-----
-> SETDBG    : In diesem Menue koennen verschiedene Debug Optionen
                gesetzt werden.
                -> time file: Datei, in die geschrieben wird, wie
                    lange jeweils die Enumerierung von "time mod"
                    Molekuelen gedauert hat.
                -> time mod: Alle "time mod" Molekuele wird die
                    vergangene Zeit in "time file" geschrieben.
                -> use NOT topology filter:
                    Deaktivierung des Topologie Filters. Damit
                    kann ueberprueft werden, ob der Topologie
                    Filter nicht zu viel filtert.
                -> write red mols to file:
                    In diese Datei werden alle redundanten
                    Molekuele geschrieben.
                -> use NOT kd-trees:
                    Deaktivieren der KD-Baeume.
                -> use NOT order:
                    Damit wird ein Teil des Topo Filters
                    deaktiviert.
                    Es wird der Teil deaktiviert, der prueft, ob
                    die Soehne eines Knotens in aufsteigender
                    Reihenfolge, bezueglich ihrer ID, angeordnet
                    sind.
-> PRINTDBG  : Gibt einen Status ueber die Debug Einstellungen aus.
```



# Literaturverzeichnis

- [1] J. Degen and M. Rarey, "Flexnovo: structure-based searching in large fragment spaces," *ChemMedChem*, vol. 1, no. 8, pp. 854–868, 2006.
- [2] J. Paern, J. Degen, and M. Rarey, "Exploring fragment spaces under multiple physicochemical constraints," *Journal of Computer-Aided Molecular Design*, vol. 21, no. 6, pp. 327–340, 2007.
- [3] R. S. Bohacek, C. McMartin, and W. C. Guida, "The art and practice of structure-based drug design: A molecular modeling perspective," *Medicinal Research Reviews*, vol. 16, no. 1, pp. 3–50, 1996.
- [4] W. Walters, M. Stahl, and M. Murcko, "Virtual screening – an overview," *Drug Discovery Today*, vol. 3, no. 4, pp. 160–178, 1998.
- [5] P. Ertl, "Cheminformatics analysis of organic substituents: Identification of the most common substituents, calculation of substituent properties, and automatic identification of drug-like bioisosteric groups," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 2, pp. 374–380, 2003. PMID: 12653499.
- [6] C. Lipinski and A. Hopkins, "Navigating chemical space for biology and medicine," *Nature*, vol. 432, no. 7019, pp. 855–861, 2004.
- [7] C. Lipinski, F. Lombardo, B. Dominy, and P. Feeney, "Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings," *Adv Drug Deliv Rev*, vol. 23, pp. 3–25, jan 1997.
- [8] C. Dobson, "Chemical space and biology," *Nature*, vol. 432, no. 7019, pp. 824–828, 2004.
- [9] C. Wermuth, "Selective optimization of side activities: another way for drug discovery," *Journal of Medicinal Chemistry*, vol. 47, no. 6, pp. 1303–1314, 2004.
- [10] G. Schneider, M. Lee, M. Stahl, and P. Schneider, "De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks," *Journal of Computer-Aided Molecular Design*, vol. 14, no. 5, pp. 487–494, 2000.
- [11] M. Rarey and M. Stahl, "Similarity searching in large combinatorial chemistry spaces," *Journal of Computer-Aided Molecular Design*, vol. 15, no. 6, pp. 497–520, 2001.

- [12] J. Degen, *Entwicklung eines struktur-basierten de-novo-Design-Verfahrens und chemischer Fragmenträume für den rationalen Wirkstoffentwurf*. PhD thesis, Zentrum für Bioinformatik, Universität Hamburg, 2008.
- [13] J. L. T. Jeremy M Berg and L. Stryer, *Biochemistry*, 5th edition. W.H. Freeman & Co Ltd, 2002.
- [14] D. Voet and J. Voet, *Biochemistry*. John Wiley & Sons, 3 ed., 2004.
- [15] G. Klebe, *Wirkstoffdesign*. Heidelberg: Spektrum Akademischer Verlag, 2 ed., 2009.
- [16] E. Fischer, “Einfluss der configuration auf die wirkung der enzyme,” *Berichte der deutschen chemischen Gesellschaft*, vol. 27, no. 3, pp. 2985–2993, 1894.
- [17] A. Fleming, “On the antibacterial action of cultures of a penicillium, with special reference to their use in the isolation of b. influenzae,” *British Journal of Experimental Pathology*, vol. 10, pp. 226–236, 1929.
- [18] W. Kühlbrandt and K. A. Williams, “Analysis of macromolecular structure and dynamics by electron cryo-microscopy,” *Current Opinion in Chemical Biology*, vol. 3, no. 5, pp. 537 – 543, 1999.
- [19] J. Cavanagh, *Protein NMR spectroscopy: principles and practice*. Academic Press, 2 ed., 2007.
- [20] J. C. Kendrew, H. M. Bodo, G. Dintzis, R. G. Parrisch, and D. C. Wyckoff H. & Phillips, “A three-dimensional model of the myoglobin molecule obtained by x-ray analysis,” *Nature*, vol. 181, pp. 662 – 666, 1958.
- [21] H. M. Berman, “The protein data bank: a historical perspective,” *Acta Crystallographica Section A*, vol. 64, pp. 88–95, Jan 2008.
- [22] W. A. Warr, “Combinatorial chemistry and molecular diversity. an overview†,” *Journal of Chemical Information and Computer Sciences*, vol. 37, no. 1, pp. 134–140, 1997.
- [23] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe, “A fast flexible docking method using an incremental construction algorithm,” *Journal of Molecular Biology*, vol. 261, no. 3, pp. 470–489, 1996.
- [24] M. Verdonk, J. Cole, M. Hartshorn, C. Murray, and R. Taylor, “Improved protein-ligand docking using gold,” *Proteins*, vol. 52, no. 4, pp. 609–623, 2003.
- [25] R. Friesner, J. Banks, R. Murphy, T. Halgren, J. Klicic, D. Mainz, M. Repasky, E. Knoll, M. Shelley, J. Perry, D. Shaw, P. Francis, and P. Shenkin, “Glide: a new approach for rapid, accurate docking and scoring. 1. method and assessment of docking accuracy,” *Journal of Medicinal Chemistry*, vol. 47, no. 7, pp. 1739–1749, 2004.

- 
- [26] C. Sotriffer, R. Mannhold, H. Kubinyi, and G. Folkers, *Virtual Screening: Principles, Challenges, and Practical Guidelines*. Methods and Principles in Medicinal Chemistry, John Wiley & Sons, 2011.
- [27] S. Teague, A. Davis, P. Leeson, and T. Oprea, "The design of leadlike combinatorial libraries," *Angewandte Chemie (International ed. in English)*, vol. 38, no. 24, pp. 3743–3748, 1999.
- [28] M. M. Hann, A. R. Leach, and G. Harper, "Molecular complexity and its impact on the probability of finding leads for drug discovery," *Journal of Chemical Information and Computer Sciences*, vol. 41, no. 3, pp. 856 – 864, 2001.
- [29] M. C. Wenlock, R. P. Austin, P. Barton, M. Davis, Andrew, and P. D. Leeson, "A comparison of physiochemical property profiles of development and marketed oral drugs," *J. Med. Chem.*, vol. 46, no. 7, p. 1250–1256, 2003.
- [30] P. D. Leeson and A. M. Davis, "Drug-like properties: guiding principles for design – or chememical prejudice?," *Drug Discovery Today – Technologies*, vol. 1, no. 3, pp. 189–195, 2004.
- [31] M. S. Lajiness, M. Vieth, and J. Erickson, "Molecular properties that influence oral drug-like behaviour," *Curr. Opin. Drug Disc. Devel.*, vol. 7, pp. 470 – 477, 2004.
- [32] M. Hann and T. Oprea, "Pursuing the leadlikeness concept in pharmaceutical research," *Current Opinion in Chemical Biology*, vol. 8, no. 3, pp. 255–263, 2004.
- [33] A. M. Leeson, P. D. & Davis, "Time-related differences in the physical property profiles of oral drugs," *J. Med. Chem.*, vol. 47, pp. 6338 – 6348, 2004.
- [34] A. L. Hopkins, C. R. Groom, and A. Alex, "Ligand efficiency: a useful metric for lead selection," *Drug Discovery Today*, vol. 9, no. 10, pp. 430 – 431, 2004.
- [35] R. John and Proudfoot, "The evolution of synthetic oral drug properties," *Bioorganic & Medicinal Chemistry Letters*, vol. 15, no. 4, pp. 1087 – 1090, 2005.
- [36] E. H. Li, D. & Kerns, "Biological assay challenges from compound solubility: strategies for bioassay optimization," *Drug Discov. Today*, vol. 11, pp. 446 – 451, 2006.
- [37] T. Wunberg, M. Hendrix, A. Hillisch, M. Lobell, H. Meier, C. Schmeck, H. Wild, and B. Hinzen, "Improving the hit-to-lead process: data-driven assessment of drug-like and lead-like screening hits," *Drug Discovery Today*, vol. 11, no. 3–4, pp. 175 – 180, 2006.

- [38] R. S. De Witte, "Avoiding physicochemical artefacts in early adme-tox experiments," *Drug Discov. Today*, vol. 11, pp. 855 – 859, 2006.
- [39] T. Oprea, T. Allu, D. Fara, R. Rad, L. Ostopovici, and C. Bologa, "Lead-like, drug-like or "pub-like": how different are they?," *Journal of Computer-Aided Molecular Design*, vol. 21, no. 1-3, pp. 113–119, 2007.
- [40] P. D. Leeson and B. Springthorpe, "The influence of drug-like concepts on decision-making in medicinal chemistry," *Nature Reviews Drug Discovery*, vol. 6, pp. 881–890, November 2007.
- [41] M. Gleeson, "Generation of a set of simple, interpretable admet rules of thumb," *Journal of Medicinal Chemistry*, vol. 51, no. 4, pp. 817–834, 2008.
- [42] M. J. Waring, "Defining optimum lipophilicity and molecular weight ranges for drug candidates — molecular weight dependent lower logd limits based on permeability," *Bioorg. Med. Chem. Lett.*, vol. 19, pp. 2844 — 2851, 2009.
- [43] C. Tyrchana, N. Blomberg, O. Engkvista, and S. Kogej, T. & Muresan, "Physicochemical property profiles of marketed drugs, clinical candidates and bioactive compounds," *Bioorg. Med. Chem. Lett.*, vol. 19, pp. 6943 — 6947, 2009.
- [44] P. M. Gleeson, A. Hersey, D. Montanari, and J. Overington, "Probing the links between in vitro potency, admet and physicochemical parameters," *Nature Reviews Drug Discovery*, vol. 10, p. 197–208, March 2011.
- [45] P. de Matos, R. Alcántara, A. Dekker, M. Ennis, J. Hastings, K. Haug, I. Spiteri, S. Turner, and C. Steinbeck, "Chemical entities of biological interest: an update," *Nucleic Acids Research*, vol. 38, no. suppl 1, pp. D249–D254, 2010.
- [46] M. Congreve, R. Carr, C. Murray, and H. Jhoti, "A 'rule of three' for fragment-based lead discovery?," *Drug Discovery Today*, vol. 8, pp. 876–877, October 2003.
- [47] I. Rechenberg, *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart, GER: Frommann-Holzboog, 1973.
- [48] C. G. Wermuth, C. R. Ganellin, P. Lindberg, and L. A. Mitcher, "Glossary of terms used in medicinal chemistry (iupac recommendations 1998)," *Pure and Applied Chemistry*, vol. 70, no. 5, pp. 1129–1143, 1998.
- [49] M. Rarey and J. Dixon, "Feature trees: a new molecular similarity measure based on tree matching," *Journal of Computer-Aided Molecular Design*, vol. 12, no. 5, pp. 471–490, 1998.

- [50] S. Hindle, M. Rarey, C. Buning, and T. Lengaue, "Flexible docking under pharmacophore type constraints," *Journal of Computer-Aided Molecular Design*, vol. 16, no. 2, pp. 129–149, 2002.
- [51] M. Hartenfeller, E. Proschak, A. Schueller, and G. Schneider, "Concept of combinatorial de novo design of drug-like molecules by particle swarm optimization," *Chemical Biology & Drug Design*, vol. 72, no. 1, pp. 16–26, 2008.
- [52] "Particle swarm optimization," *Proceedings of the 1995 IEEE International Conference on Neural Networks*, vol. 4, pp. 1942 – 1948.
- [53] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, pp. 33 – 57, 2007.
- [54] G. Schneider, W. Neidhart, T. Giller, and G. Schmid, "'scaffold-hopping' by topological pharmacophore search: A contribution to virtual screening," *Angewandte Chemie International Edition*, vol. 38, no. 19, pp. 2894–2896, 1999.
- [55] G. Schneider, O. Clément-Chomienne, L. Hilfiger, P. Schneider, S. Kirsch, H.-J. Böhm, and W. Neidhart, "Virtual screening for bioactive molecules by evolutionary de novo design," *Angewandte Chemie International Edition*, vol. 39, no. 22, pp. 4130–4133, 2000.
- [56] J. Balfour and G. Plosker, "Rosiglitazone," *Drugs*, vol. 57, pp. 921 – 930, 1999.
- [57] H. Boehm, A. Flohr, and M. Stahl, "Scaffold hopping," *Drug Discov Today: Technol*, vol. 1, no. 3, pp. 217–224, 2004.
- [58] P. Maass, T. Schulz-Gasch, M. Stahl, and M. Rarey, "Recore: a fast and versatile method for scaffold hopping based on small molecule crystal structure conformations," *Journal of Chemical Information and Modeling*, vol. 47, no. 2, pp. 390–399, 2007.
- [59] P. Maass, *Neue rechnergestützte Methoden zum Scaffold-Hopping unter Verwendung von Kristallstrukturen kleiner Moleküle*. PhD thesis, Zentrum für Bioinformatik, Universität Hamburg, 2009.
- [60] A. Guttman, "R-trees: a dynamic index structure for spatial searching," *SIGMOD Rec.*, vol. 14, pp. 47–57, June 1984.
- [61] A. White, D. David, and R. Jain, "Similarity indexing: Algorithms and performance," *Proc. SPIE: Storage and Retrieval for Still Image and Video Databases IV*, vol. 2670, pp. 62–73, 1996.
- [62] M. Boehm, T.-Y. Wu, H. Claussen, and C. Lemmen, "Similarity searching and scaffold hopping in synthetically accessible combinatorial chemistry spaces," *Journal of Medicinal Chemistry*, vol. 51, no. 8, pp. 2468–2480, 2008.

- [63] J.-L. Reymond, T. Fink, and H. Heinz Bruggesser, "Virtual exploration of the small-molecule chemical universe below 160 daltons," *Angewandte Chemie International Edition*, vol. 44, no. 10, p. 1504–1508, 2005.
- [64] J.-L. Reymond and T. Fink, "Virtual exploration of the chemical universe up to 11 atoms of c, n, o, f: Assembly of 26.4 million structures (110.9 million stereoisomers) and analysis for new ring systems, stereochemistry, physicochemical properties, compound classes, and drug discovery," *J. Chem. Inf. Model.*, vol. 47, p. 342–353, 2007.
- [65] J.-L. Reymond and L. C. Blum, "970 million druglike small molecules for virtual screening in the chemical universe database gdb-13," *J. Am. Chem. Soc.*, vol. 131, no. 25, p. 8732–8733, 2009.
- [66] B. D. McKay, "Practical graph isomorphism," *Congr. Numerantium*, vol. 30, pp. 45–87, 1981.
- [67] S. Bohanec and M. Perdih, "Symmetry of chemical structures: A novel method of graph automorphism group determination," *J. Chem. Inf. Comput. Sci.*, vol. 33, p. 719–726, 1993.
- [68] A. Schüller, G. Schneider, and E. Byvatov, "Smilib: Rapid assembly of combinatorial libraries in smiles notation," *QSAR & Combinatorial Science*, vol. 22, no. 7, pp. 719–721, 2003.
- [69] A. Schüller, V. Hähnke, and G. Schneider, "Smilib v2.0: A java-based tool for rapid combinatorial library enumeration," *QSAR & Combinatorial Science*, vol. 26, no. 3, pp. 407–410, 2007.
- [70] J. M. Barnard and G. Downs, "Computer representation and manipulation of combinatorial libraries," *Perspectives in Drug Discovery and Design*, vol. 7-8, pp. 13–30, 1997.
- [71] J. M. Barnard, G. M. Downs, A. von Scholley-Pfab, and R. D. Brown, "Use of markush structure analysis techniques for descriptor generation and clustering of large combinatorial libraries," *Journal of Molecular Graphics and Modelling*, vol. 18, no. 4-5, pp. 452 – 463, 2000.
- [72] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of Chemical Information and Computer Sciences*, vol. 28, no. 1, pp. 31–36, 1988.
- [73] A. Cayley, "A theorem on trees.," *Quart. J. Math.*, vol. 23, p. 376–378, 1889.
- [74] F. Harary and G. Prins, "The number of homeomorphically irreducible trees, and other species," *Acta Mathematica*, vol. 101, pp. 141–162, 1959. 10.1007/BF02559543.

- [75] I. J. Good, "Mathematical proceedings of the cambridge philosophical society," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 61, pp. 499–517, 1965.
- [76] D. E. Knuth, *The art of computer programming. Vol. 1, Fundamentals algorithms*. Addison-Wesley Pub. Co., 3 ed., July 1997.
- [77] G. Pólya, "Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen," *Acta Mathematica*, vol. 68, pp. 145–254, 1937. 10.1007/BF02546665.
- [78] R. Aringhieri, P. Hansen, F. Malucelli, R. Aringhieri, P. Hansen, and F. Malucelli, "Chemical tree enumeration algorithms," *Technical Report, University of Pisa*, 1999.
- [79] W. Schröter, K.-H. D. h. Lautenschläger, and H. Bibrack, *Chemie Fakten und Gesetze*. VEB Fachbuchverlag Leipzig, 1985.
- [80] H. Beyer, W. Francke, and W. Walter, *Lehrbuch der Organischen Chemie*. S. Hirzel Verlag Stuttgart - Leibzig, 24 ed., 2004.
- [81] K. Popper, *The open universe: an argument for indeterminism*. Routledge, 1992.
- [82] Y. T. Oganessian, V. K. Utyonkov, Y. V. Lobanov, F. S. Abdullin, A. N. Polyakov, R. N. Sagaidak, I. V. Shirokovsky, Y. S. Tsyganov, A. A. Voinov, G. G. Gulbekian, S. L. Bogomolov, B. N. Gikal, A. N. Mezentsev, S. Iliev, V. G. Subbotin, A. M. Sukhov, K. Subotic, V. I. Zagrebaev, G. K. Vostokin, M. G. Itkis, K. J. Moody, J. B. Patin, D. A. Shaughnessy, M. A. Stoyer, N. J. Stoyer, P. A. Wilk, J. M. Kenneally, J. H. Landrum, J. F. Wild, and R. W. Loughheed, "Synthesis of the isotopes of elements 118 and 116 in the  $^{249}\text{Cf}$  and  $^{245}\text{Cm} + ^{48}\text{Ca}$  fusion reactions," *Phys. Rev. C*, vol. 74, p. 044602, Oct 2006.
- [83] A. K. Ghose, V. N. Viswanadhan, and J. J. Wendoloski, "Prediction of hydrophobic (lipophilic) properties of small organic molecules using fragmental methods: An analysis of alogp and clogp methods," *J. Phys. Chem. A*, vol. 102, no. 21, p. 3762–3772, 1998.
- [84] Daylight  
CLOGP Reference Manual  
<http://www.daylight.com/dayhtml/doc/clogp/index.html>  
accessed October 2011.
- [85] X. Lewell, D. Judd, S. Watson, and M. Hann, "Recap-retrosynthetic combinatorial analysis procedure: a powerful new technique for identifying privileged molecular fragments with useful applications in combinatorial chemistry," *Journal of Chemical Information and Computer Sciences*, vol. 38, no. 3, pp. 511–522, 1998.

- [86] SMARTS - A Language for Describing Molecular Patterns  
<http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html>  
 accessed October 2011.
- [87] P. Fricker, M. Gastreich, and M. Rarey, "Automated drawing of structural molecular formulas under constraints," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 3, pp. 1065–1078, 2004.
- [88] WDI - World Drug Index  
[http://thomsonreuters.com/products\\_services/science/science\\_products/a-z/world\\_drug\\_index/](http://thomsonreuters.com/products_services/science/science_products/a-z/world_drug_index/)  
 accessed October 2011.
- [89] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2nd ed., 1994.
- [90] D. Weininger, A. Weininger, and J. Weininger, "Smiles. 2. algorithm for generation of unique smiles notation," *Journal of Chemical Information and Computer Sciences*, vol. 29, no. 2, pp. 97–101, 1989.
- [91] M. Vieth, M. G. Siegel, R. E. Higgs, I. A. Watson, D. H. Robertson, K. A. Savin, G. L. Durst, and P. A. Hipskind, "Characteristic physical properties and structural fragments of marketed oral drugs," *J. Med. Chem.*, vol. 47, p. 224–232, November 2004.
- [92] R. A. Carr, M. Congreve, C. W. Murray, and D. C. Rees, "Fragment-based lead discovery: leads by design," *Drug Discovery Today*, vol. 10, no. 14, pp. 987 – 992, 2005.
- [93] J. R. Proudfoot, "The evolution of synthetic oral drug properties," *Bioorganic & Medicinal Chemistry Letters*, vol. 15, no. 4, pp. 1087–1090, 2005.
- [94] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, pp. 509–517, September 1975.
- [95] D. T. Lee and C. K. Wong, "Worst-case analysis for region and partial region searches in multidimensional binary search trees and balanced quad trees," *Acta Informatica*, vol. 9, pp. 23–29, 1977.
- [96] G. M. Adelson-Velskii and Y. M. Landis, "An algorithm for the organization of information," *Doklady Akademii Nauk USSR, Moscow*, vol. 16, no. 2, pp. 263–266, 1962.
- [97] G. M. Adelson-Velskii and Y. M. Landis, "An algorithm for the organization of information, (english translation)," *U.Soviet Math. Doklady*, vol. 3, p. 1259–1263, 1962.



- 
- [98] R. Sedgewick, *Algorithmen in C*. Addison-Wesley (Deutschland) GmbH, 1 ed., 1993.
- [99] J. Irwin and B. Shoichet, “Zinc – a free database of commercially available compounds for virtual screening,” *J Chem Inf Model*, vol. 45, no. 1, pp. 177–182, 2005.
- [100] N. Huang, B. Shoichet, and J. Irwin, “Benchmarking sets for molecular docking,” *Journal of Medicinal Chemistry*, vol. 49, no. 23, pp. 6789–6801, 2006.
- [101] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2001.
- [102] S. Wildman and G. Crippen, “Prediction of physicochemical parameters by atomic contributions,” *Journal of Chemical Information and Modeling*, vol. 39, no. 5, pp. 868–873, 1999.
- [103] M. Stahl and M. Rarey, “Detailed analysis of scoring functions for virtual screening,” *J Med Chem*, vol. 44, no. 7, pp. 1035–1042, 2001.
- [104] B. Shoichet, “Virtual screening of chemical libraries,” *Nature*, vol. 432, no. 7019, pp. 862–865, 2004.
- [105] J. Schlosser and M. Rarey, “Beyond the virtual screening paradigm: Structure-based searching for new lead compounds,” *Journal of Chemical Information and Modeling*, vol. 49, no. 4, pp. 800–809, 2009.
- [106] G. Schneider, “Virtual screening: an endless staircase?,” *Nature Reviews Drug Discovery*, vol. 9, pp. 273–276, April 2010.
- [107] K. Stierand, P. Maass, and M. Rarey, “Molecular complexes at a glance: automated generation of two-dimensional complex diagrams,” *Bioinformatics (Oxford, England)*, vol. 22, no. 14, pp. 1710–1716, 2006.
- [108] K. Stierand and M. Rarey, “From modeling to medicinal chemistry: automatic generation of two-dimensional complex diagrams,” *ChemMedChem*, vol. 2, no. 6, pp. 853–860, 2007.
- [109] K. Stierand and M. Rarey, “Flat and easy: 2d depiction of protein-ligand complexes,” *Molecular Informatics*, vol. 30, no. 1, pp. 12–19, 2011.
- [110] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” in *Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation - Volume 6*, (Berkeley, CA, USA), pp. 10–10, USENIX Association, 2004.
- [111] T. White, *Hadoop: The Definitive Guide*. Yahoo Press, 2010.

- [112] B. Stroustrup, *The C++ Programming Language*. Addison-Wesley Longman Publishing Co., Inc., 3rd ed., 2000.
- [113] Qt - Cross-platform application and UI framework  
<http://qt.nokia.com>  
accessed October 2011.
- [114] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1995.
- [115] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns*. Chichester, UK: Wiley, 1996.
- [116] A. Alexandrescu, *Modern C++ design: generic programming and design patterns applied*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2001.
- [117] B. W. Kernighan and D. M. Ritchie, *C Programming Language*. Prentice Hall, 2 ed., 1998.
- [118] M. Rarey, *Rechnergestützte Vorhersage von Rezeptor-Ligand-Wechselwirkungen*. Oldenbourg, München, 1996.

## Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich meine Dissertation *Eigenschaftsbasierte Auswahl von Molekülen aus chemischen Fragmenträumen* ohne unerlaubte Hilfe angefertigt und mich dabei keiner anderen als der von mir ausdrücklich bezeichneten Quellen bedient habe.

Hamburg, den 24. Januar 2012

(Juri Pärn)