



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Fakultät
Mathematik, Informatik und Naturwissenschaften

Bachelor-Thesis

Berechnung von Teilmengen-Relationen chemischer Muster

Calculation of Subset Relationships between chemical Patterns

Hamburg, den XY. Oktober 2015

Farina Ohm

Adresse: Friedastraße 12, 22043 Hamburg

Studiengang Computing in Science SP Biochemie
Matr.-Nr. 6314051

Erstgutachter Universität Hamburg:

Zweitgutachter Universität Hamburg:

Prof. Dr. Rarey

Prof. Dr. Heitmann

Zusammenfassung

Chemische Muster beschreiben strukturelle Eigenschaften oder funktionelle Gruppen, die in Molekülen enthalten sein können. Anhand dieser Darstellung von bestimmten chemischen Merkmalen werden ganze Molekülmengen beschrieben. Chemische Muster können zum Beispiel mit der SMARTS-Sprache ausgedrückt werden.

Vorhersagen zur molekularen Interaktion durch den Vergleich mit bereits bekannten Verbindungen zu treffen, ist eine häufig genutzte Methode in der modernen Wirkstoffentwicklung. So können neue Verbindungen gefunden werden, die sich in ihrer biologischen Aktivität ähneln. Der Vergleich erfolgt dabei meist durch eine Analyse auf molekulare Ähnlichkeit. Um Molekülmengen, beschrieben durch chemische Muster, zu vergleichen, können sie auf Teilmengen-Relationen geprüft werden. Solch ein Vergleich ermöglicht es Rückschlüsse auf eine vorliegende Hierarchie der chemischen Muster zu ziehen oder Redundanzen ausschließen zu können. Ein Vergleich von chemischen Mustern ist, durch die hohe Komplexität dieser Beschreibungen, bislang nicht realisiert worden.

Es wurde ein Verfahren entwickelt, welches SMARTS-Ausdrücke miteinander vergleicht und auf Teilmengen-Relationen überprüft. Realisiert wurde dieses Verfahren durch die Generierung von sogenannten Fingerprints, die es ermöglichen die komplexen Atom- und Bindungs-Repräsentationen der SMARTS-Sprache zu vergleichen. Abschließend wurde das Verfahren evaluiert und Experimente mit realen Daten durchgeführt, um ein möglichen Anwendungsbereich zu präsentieren.

Abstract

A chemical pattern describes characteristical structures or functional groups that occur in molecules. A set of molecules can be represented as a chemical pattern written as a SMARTS-pattern.

Predictions of molecular interactions of chemical compounds are a popular approach in modern drug-discovery. By that the biological activity will presumably be similar to known compounds. An often used method for comparison is the analysis of the molecular similarity. To compare chemical patterns it's possible to calculate subset relationships. This could clarify a hierarchy or point out redundancies. Until now the high complexity of SMARTS-pattern prevent a comparison between chemical patterns.

An approach is to compare chemical patterns and draw conclusions about the existing subset relationships. This approach was implemented by using the concept of fingerprints, which were calculated for each atom and bound in a chemical pattern respectively, to allow the comparison of those complex SMARTS-pattern. In conclusion the method was evaluated and applied on real data to present a possible scope of applications.

Inhaltsverzeichnis

1 Einleitung	4
1.1 Motivation	4
1.2 Grundlagen	6
2 Lösungsstrategie	12
2.1 Konzepte	12
2.2 Struktur	16
3 Realisierung	18
3.1 Genutze Software	18
3.2 Aufarbeitung und Auswahl	18
3.3 Fingerprintgenerierung	19
3.4 Vergleich und Aufbau Relations-Matrix	22
3.5 Subgraphisomorphie Analyse	24
3.6 Kennzeichnung	24
4 Evaluation	26
4.1 Validierung	26
4.2 Laufzeitanalyse	28
5 Experimente und Auswertung	30
5.1 Diskussion der Experimente	32
6 Fazit	35
6.1 Ungelöste Probleme	35
7 Weiterführende Arbeit	37
A Software User Guide	I
A.1 Command Line Syntax	I
A.2 Command Line Options	I
B Softwarearchitektur	IV
C Wichtige Module und Klassen	V
D Tests	VII
E APIs	VIII
F Genutzte Datensätze	IX
G Ergebnisse der Experimente	X
H Implementation	X

I Literatur

XI

1 Einleitung

1.1 Motivation

Die Vorhersage von molekularen Interaktionen ist in der modernen Arzneimittelentwicklung die Grundlage zur Gewinnung neuer Wirkstoffe. [1]

Es gibt viele Methoden um das Verhalten von Verbindungen einschätzen zu können. Ein Verfahren, welches von Beginn an in der Chemie genutzt wird, ist das induktive Lernen [1], bei dem es sich um einen Kreislauf handelt (siehe Abb. 1). Anhand von Experimenten werden Modelle entwickelt mit denen Prognosen aufgestellt werden, die im Anschluss mit neuen Experimenten validiert werden. Anschließend wiederholt sich das Vorgehen. Durch die gewonnenen Modelle können, je nach Art des Modells, Einschätzungen über die biologische Aktivität von Molekülen getroffen werden. Eine weitere Methode ist die

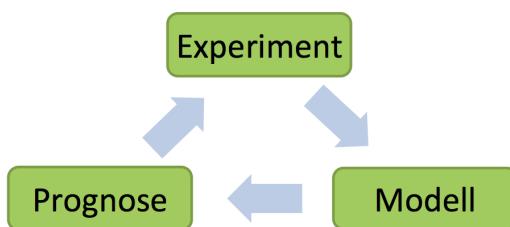


Abbildung 1: Kreislauf des induktiven Lernens

Quantitative Struktur-Aktivitätsanalyse (QSAR) [2], bei der die Wirkung eines Moleküls durch die molekularen Eigenschaften beschrieben werden kann. Durch QSAR kann eine Prognose, wie ein Molekül sich bei einer Interaktion verhält, aufgestellt werden.

Der Begriff der pharmakologisch motivierten molekularen Ähnlichkeit [3] ermöglicht die Vorhersage anhand bereits bekannter Verbindungen. Zwei Moleküle werden als ähnlich bezeichnet, wenn sie sich in biologischen Systemen ähnlich verhalten.

Besonders bei den letzten beiden vorgestellten Verfahren eignet sich der Einsatz von computergestützten Verfahren. Computergestützte Verfahren oder Werkzeuge finden in der modernen Chemie immer mehr Anwendung. Sie eignen sich besonders, da mit ihrer Hilfe große Mengen an Informationen analysiert werden können (z.B. durch QSAR). Die sogenannte Chemieinformatik ist ein Gebiet der Chemie, die mit Hilfe von Methoden der Informatik versucht chemische Probleme zu lösen. [1][3][4]

Die molekulare Ähnlichkeit von zwei Molekülen kann durch Einsatz von computergestützten Methoden auf verschiedene Arten untersucht werden. Sie kann durch topologische Indizes (z.B. Randić-Index [5]), durch Fingerprint Generierung [3][4], bei der angegeben wird welche strukturellen Eigenschaften oder funktionellen Gruppen ein Molekül enthält, oder auch mit Ähnlichkeits- bzw. Distanzmaßen (z.B. Tanimoto-Koeffizient [6]) bestimmt werden. [6]

Durch die Bestimmung der molekularen Ähnlichkeit lassen sich nicht nur für einzelne Moleküle Vorhersagen zu den Interaktionen treffen, sondern auch für Molekülmengen, die sich mit chemischen Mustern [7][8] beschreiben lassen. Die Interaktion einer Verbindung

mit anderen erfolgt meist aufgrund von bestimmten Strukturen oder funktionellen Gruppen innerhalb der Verbindung. Diese zu untersuchenden Eigenschaften lassen sich mit chemischen Mustern ausdrücken (z.B. ein Muster, welches alle Moleküle beschreibt, die eine Nitrogruppe enthalten). Diese Muster stellen eine Beschreibung einer Molekülmenge, bestehend aus Molekülen mit den relevanten Eigenschaften, dar. Sie können dazu genutzt werden Moleküle zu filtern. So ist eine Aussortierung von beispielsweise Toxikophoren oder auch das Auswählen von Verbindungen mit gewünschten Eigenschaften möglich.

Um es in der Chemieinformatik zu ermöglichen Interaktionen vorherzusagen oder zu analysieren, können chemische Verbindungen in eine Graphenstruktur überführt werden, wodurch eine mathematische Analyse möglich wird.

Es handelt sich bei solchen Molekülgraphen [4] um ungerichtete, endliche und markierte Graphen. Dabei werden die Atome als Knoten und die chemischen Bindungen als Kanten betrachtet. Die so entstehenden Graphen können anschließend im Rahmen der Graphentheorie auf gemeinsame Eigenschaften oder Teilstrukturen untersucht werden, um die molekulare Ähnlichkeit zu bewerten. Analog erfolgt die Übertragung von chemischen Mustern in eine Graphenstruktur, welche dann für den Vergleich aus der graphentheoretischen Sicht genutzt werden kann. Dies ist nötig, da auch chemische Muster und somit die beschriebenen Molekülmengen Ähnlichkeiten zueinander aufweisen können. Ein Vergleich würde es ermöglichen festzustellen, ob es eine Hierarchie der chemischen Muster gibt, ob es in einer Menge von chemischen Mustern redundante Muster gibt, aber auch um festzustellen, ob sich bestimmte Teilstrukturen oder funktionelle Gruppen in bereits genutzten Mustern wiederfinden. Bislang gibt es in der Chemieinformatik keine Methode, um chemische Muster auf ihre Ähnlichkeit zu überprüfen. In der Informatik und den zu chemischen Mustern äquivalenten, regulären Ausdrücken¹ gibt es Ansätze zur Ähnlichkeitsanalyse. Meist basiert die Analyse auf der Generierung und dem Vergleich von endlichen Automaten²[10]. Die Komplexität der chemischen Muster verhindert eine Übertragung dieses Ansatzes auf das hier vorgestellte Problem, weshalb es nötig ist, ein neues Verfahren zu entwickeln. Ein naiver Ansatz wäre es die Moleküle, die von chemischen Mustern beschrieben werden zu vergleichen. Es kann passieren, dass verschiedene Muster durch Zufall die gleichen Moleküle beschreiben, indem z.B. zwei funktionelle Gruppen, jeweils als chemisches Muster beschrieben, gemeinsam und dennoch unabhängig in einem Molekül auftreten. Bei diesem Ansatz muss es einen Abgleich geben, der überprüft welche Atome in dem Molekül von den Mustern beschrieben werden. Zudem ermöglicht dieser Vergleich von chemischen Mustern es nicht eine allgemeine Aussage über die Ähnlichkeit zu treffen. Grund dafür ist die Abhängigkeit von den chemischen Mustern zu der genutzten Molekülmenge, die unter Umständen keine Moleküle beinhaltet, die von chemischen Mustern beschrieben werden.

Eine Lösung zum Vergleich zweier chemischer Muster wird in dieser Arbeit vorgestellt und beschrieben. Der Lösungsansatz wurde im Rahmen einer Masterarbeit von A. Mashychev [11] entwickelt und in der hier vorgestellten Arbeit weiterentwickelt und implementiert.

¹Zeichenkette, die eine Menge von Zeichenketten beschreibt

²Modell zur Beschreibung eines Verhaltens nach Eingabe einer Anfrage, bestehend aus einer endlichen Anzahl an Zuständen, Zustandsübergängen und Aktionen [9]

1.2 Grundlagen

Für die in den folgenden Kapiteln beschriebene Arbeit ist es nötig einige Begriffe und Grundlagen zu erklären.

1.2.1 SMILES

Unter einem SMILES (*Simplified Molecular Input Line Entry Specification*) versteht man eine String-Repräsentation von Molekülen. Entwickelt wurde diese Beschreibung mit Hilfe von ASCII-Zeichen und Bindungssymbolen. Der Vorteil von SMILES gegenüber anderen Molekülrepräsentationen ist die Eigenschaft, dass die Molekülbeschreibung in einer Zeile kompakt und effizient gespeichert werden kann. Ein effizientes Arbeiten mit SMILES-Strings ist dadurch möglich, dass ein SMILES-String leicht in einen Molekülgraphen überführt werden kann. Die Grammatik eines SMILES-Strings gliedert sich in fünf grundlegende Einheiten:

- Atome (mit impliziten und expliziten Eigenschaften)
- Bindungen
- Verzweigungen
- Unterbrechungen
- Ringe

Kovalent gebundene Atome stehen benachbart in einer Zeichenkette. Bindungsprimitive, die Beschreibungen von chemischen Bindungen, werden zwischen den jeweiligen Atomen angegeben. [8][12]

Atome

Die Atom-Beschreibung innerhalb eines SMILES-String erfolgt über die Angabe des Element-Symbols. Eine Besonderheit ist dabei die Angabe eines Atoms durch „*“. Das „*“ gilt dabei als nicht spezifiziertes oder unbekanntes Atom, einer sogenannten *Wildcard*. Des Weiteren werden Elemente, die im *organic subset*¹ enthalten sind, von den restlichen Elementen des Periodensystems unterschieden.

Eine weitere Besonderheit betrifft die Atome C, N, P, O, S, As, Se und *. Diese Atome können, durch Angabe der Element-Symbole als Kleinbuchstaben, als aromatische Atome interpretiert werden.

Neben dem Element-Symbol werden Atome mit Hilfe weiterer Eigenschaften, den sogenannten Atomprimitiven, beschrieben. Grundsätzlich gilt dabei folgende Syntax für die Angabe einer Atombeschreibung:

```
[<mass>symbol<chiral><hcount><sign<charge>>]
```

¹Teilmenge der chemischen Elemente bestehend aus B, C, N, O, P, S, F, Cl, Br, I und *. Diese Elemente können ohne [] angegeben werden. Sie besitzen in dieser Form die implizite Anzahl an Bindungen zu Wasserstoffen aufgrund der kleinst möglichen Valenz, keine Ladung, keine Chiralität und keine isotopischen Eigenschaften. [13]

Tabelle 1 beschreibt die einzelnen Eigenschaften und wie sie in einem SMILES repräsentiert werden. Beispiel 1.2.1 zeigt drei unterschiedliche SMILES und ihre zugehörigen Moleküle, die sie beschreiben. [8][13]

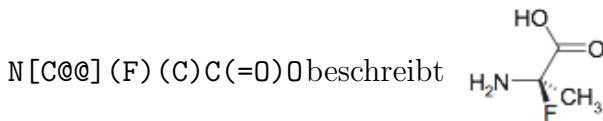
Tabelle 1: Beschreibung der möglichen Atomprimitive. **Symbol** beschreibt, wie in Kapitel 1.2.1 erläutert, das Element-Symbol. **@** gibt an, dass die folgenden Atome gegen den Uhrzeigersinn gelistet sind, bei **@@** sind sie im Uhrzeigersinn gelistet. **n** ist bei den Atomprimitiven eine natürliche Zahl. [8]

Atomprimitiv	SMILES-Syntax	Beschreibung
<mass>	<n>	Masse des Atoms
symbol	Symbol	Element-Symbol nach Periodensystem
<chiral>	@@ oder @	Chiralität an tetraedrischen Kohlenstoffen
<hcount>	H<n>	Anzahl gebundener Wasserstoff-Atome
<sign<charge>>	-<n> oder +<n>	Ladung des Atoms

Beispiel 1.2.1.

0 beschreibt H2O

[Fe+2] beschreibt Fe^{+2}



Bindungen

Innerhalb eines SMILES-Strings wird zwischen den in Tabelle 2 beschriebenen Bindungen unterschieden. Beispiel 1.2.2 zeigt drei Beispiele von SMILES und die Moleküle, die sie repräsentieren. [12]

Tabelle 2: Beschreibung der möglichen Bindungsprimitiven. **default** beschreibt den Fall, dass zwischen zwei Atombeschreibungen keine Bindung angegeben wird. \ und / beschreiben eine Konfigurationsisomerie, die *cis-trans-Isomerie*, die im Beispiel 1.2.2 dargestellt ist. [13]

SMILES-Syntax	Beschreibung
default	Einfachbindung
-	Einfachbindung
=	Doppelbindung
#	Dreifachbindung
:	aromatische Bindung (1.5 fach)
\ und /	nach oben und nach unten Bindung als Stereospezifikation von Doppelbindungen (angegeben in Paaren)

Beispiel 1.2.2.



Verzweigungen

Eine Verzweigung innerhalb des Molekülgraphs wird durch () angegeben. Die Ausdrücke können gereiht und geschachtelt werden. Beispiel 1.2.3 zeigt eine solche Verzweigung eines Moleküls. In diesem Molekül sind mindestens zwei Verzweigungen zur Beschreibung nötig. [8][12]

Beispiel 1.2.3.



Unterbrechungen

Um eine Unterbrechung zu kennzeichnen, kann in der SMILES Sprache ein „..“ genutzt werden, der zwischen zwei Atombeschreibungen steht. Dadurch werden Atome, die nicht miteinander verbunden sind, kenntlich gemacht, wie Beispiel 1.2.4 zeigt. [8][13]

Beispiel 1.2.4.



Ringe

Eine Ringbindung kann durch die Angabe eines Zahlenpaars dargestellt werden. Dabei wird eine Bindung in einem Molekül, die einen Ringschluss hervorrufen würde, entfernt und an die beiden beteiligten Atome ein passendes Zahlenpaar angefügt, wie das Beispiel 1.2.5 zeigt. [13]

Beispiel 1.2.5.



1.2.2 SMARTS

Die Abkürzung SMARTS steht für *SMiles ARbitrary Target Specification* und bezeichnet eine Sprache, die chemischer Muster und deren Eigenschaften mit Hilfe von kurzen Stringbeschreibungen repräsentiert. Die SMARTS Sprache ist eine Spezifizierung von SMILES. Daher gilt, jeder SMILES ist auch ein valider SMARTS. SMARTS-Ausdrücke besitzen zusätzliche Atom- bzw. Bindungsprimitive, die im Folgenden erklärt werden. [7][8]

Atome

Die SMARTS Sprache besitzt eine mächtigere Beschreibung im Vergleich zur SMILES-Sprache. Dazu wurden die Atomprimitive, beschrieben in Kapitel 1.2.1, um die in Tabelle 3 dargestellten Atomprimitive erweitert. [7][8]

Tabelle 3: Beschreibung der, für SMARTS hinzukommenden, Atomprimitive. n und c sind hierbei natürliche Zahlen. Die Zeile „default-Wert“ gibt den Wert an, den ein Atomprimitiv einnimmt, wenn n und c nicht angegeben sind. [7]

Syntax	Beschreibung	default-Wert
a	aromatisches Atom	-
A	aliphatisches Atom	-
D<n>	<n>Bindungen zu nicht-H Atomen	$n = 1$
X<n>	Atom mit insgesamt <n>Bindungen	$n = 1$
x<n>	Atom mit <n>Ringbindungen	$n \geq 1$
v<n>	Atom mit Bindungsordnung <n>	$n = 1$
H<n>	Atom mit <n>Bindungen zu expliziten Wasserstoffen	$n = 1$
h<n>	Atom mit <n>Bindungen zu impliziten Wasserstoffen	$n \geq 1$
R<n>	in <n>Ringen eines SSSR	jedes Ringatom
r<n>	in dem kleinsten Ring einer SSSR mit <n>Atomen	jedes Ringatom
#<n>	Atom mit Ordnungszahl <n>	-
@<c><n>	Chiralitätsklasse <c>, Chiralität <n>	-
@<c><n>?	Chiralitätsklasse <c>, Chiralität <n>oder nicht spezifiziert	-

Bindungen

Die Bindungsprimitive der SMARTS Sprache werden um die in Tabelle 4 dargestellten Eigenschaften erweitert.

Tabelle 4: Beschreibung der für SMARTS hinzukommenden Bindungsprimitive. **default** beschreibt den Fall, dass zwischen zwei Atombeschreibungen keine Bindung angegeben wird. [7][8]

SMARTS-Syntax	Beschreibung
default	Einfach- oder aromatische Bindung
?\\und ?/	nach oben oder nicht spezifiziert und nach unten oder nicht spezifiziert
~	beliebige Bindung (Wildcard)
©	beliebige Bindung in einem Ring

Logische Operatoren

Atom- und Bindungsprimitive können in der SMARTS Sprache mit Hilfe von logischen Operatoren verknüpft werden. In der Tabelle 5 sind die Operatoren aufgelistet.

Tabelle 5: Beschreibung der logischen Operatoren in der SMATRTS Sprache, wobei es sich bei SMARTSExp um ein Atom- oder Bindungsprimitiv handelt. [8]

SMARTS-Syntax	Beschreibung
$!<\text{SMARTSExp}>$	nicht $<\text{SMARTSExp}>$
$<\text{SMARTSExp1}> \& <\text{SMARTSExp2}>$	$<\text{SMARTSExp1}>$ und $<\text{SMARTSExp2}>$ (starke Bindung)
$<\text{SMARTSExp1}>, <\text{SMARTSExp2}>$	$<\text{SMARTSExp1}>$ oder $<\text{SMARTSExp2}>$
$<\text{SMARTSExp1}>; <\text{SMARTSExp2}>$	$<\text{SMARTSExp1}>$ und $<\text{SMARTSExp2}>$ (schwache Bindung)

Unterbrechungen

Auch in der SMARTS Sprache sind Atom-Unterbrechungen, also nicht miteinander verbundene Atome, mit Hilfe eines „.“ möglich. Bei der SMARTS Sprache ist es zusätzlich noch möglich, durch Setzung von Klammern, ein unterschiedliches Betrachten von SMARTS-Ausdrücken zu ermöglichen (siehe Tabelle 6).

Tabelle 6: Beschreibung für mögliche Unterbrechungen innerhalb der SMATRTS Sprache. Bei SMARTSExp handelt es sich um ein Atomprimitiv. [8]

SMARTS-Syntax	Beschreibung
$<\text{SMARTSExp1}>. <\text{SMARTSExp2}>$	$<\text{SMARTSExp1}>$ und $<\text{SMARTSExp2}>$ können, müssen aber nicht verbunden sein
$(<\text{SMARTSExp1}>). (<\text{SMARTSExp2}>)$	$<\text{SMARTSExp1}>$ und $<\text{SMARTSExp2}>$ treffen in verschiedenen (nicht kovalent verbundenen) Fragmenten
$(<\text{SMARTSExp1}>). <\text{SMARTSExp2}>$	$<\text{SMARTSExp1}>$ und $<\text{SMARTSExp2}>$ treffen unabhängig voneinander

Rekursive Ausdrücke

Ein weiterer Zusatz der SMARTS Sprache ist die Erweiterung der Beschreibung von Atomeigenschaften durch rekursive Ausdrücke. Unter rekursiven Ausdrücken versteht man die Beschreibung der Atomumgebung durch einen weiteren SMARTS-Ausdruck in der Form $\$(\text{SMARTSExp})$. [7][8]

Beispiel 1.2.6.

$[\text{C}[\$(\text{CO})]]$ beschreibt ein Kohlenstoffatom, welches als Umgebung kovalent an ein Sauerstoffatom gebunden ist

1.2.3 Subgraph-Isomorphie

Ein weiterer Begriff, der vorab erläutert werden muss, ist der der Subgraph-Isomorphie. Die Subgraph-Isomorphie ist ein \mathcal{NP} -vollständiges Graphenproblem der theoretischen Informatik. [4][8][14]

Gegeben sei ein Graph $G = (V, E)$. V beschreibt die Menge an Knoten, die im Graphen G enthalten sind und E die Menge an Kanten des Graphens, so dass gilt:

$$E \subseteq \{(v_i, v_j) | v_i, v_j \in V, v_i \neq v_j\} \quad (1)$$

Gegeben sei ein weiterer Graph $G' = (V', E')$. Besitzen zwei Graphen G und G' die gleiche Menge an Knoten $V = V'$ und die gleiche Menge an Kanten $E = E'$, so gilt auch $G = G'$. Sie sind also gleich oder automorph. Gilt für $G'(V', E')$

$$V' \subseteq V, E' \subseteq E \cap (V' \times V') \quad (2)$$

so ist G' ein Subgraph von G . Gilt weiter

$$E' = \{(v_i, v_j) \in E | v_i, v_j \in V'\}, \quad (3)$$

so ist G' ein induzierter Subgraph von G .

Gegeben seien nun zwei Graphen G und F , die eine unterschiedliche Knotenmenge haben und somit nicht gleich sein können. Sie lassen sich dennoch durch den Isomorphiebegriff miteinander vergleichen. Die Isomorphie ist ein Fachbegriff der Graphentheorie und wird geschrieben als $G \simeq F$. Ein Graph G ist zu einem anderen Graph F genau dann isomorph, wenn eine bijektive Abbildung $m : V_F \rightarrow V_G$ existiert, mit

$$(v_i, v_j) \in E_F \Leftrightarrow (m(v_i), m(v_j)) \in E_G. \quad (4)$$

Die Abbildung m wird dann als Isomorphismus von F nach G bezeichnet. Unter der Graphenisomorphie bezeichnet man einen Isomorphismus zweier Graphen gleicher Größe. Besitzt Graph F eine geringere Anzahl an Knoten und Kanten als G , spricht man von Subgraph-Isomorphie, falls F zu einem Teilgraphen von G isomorph ist. Also gilt

$$V_F \rightarrow V_{G'}, V_{G'} \subseteq V_G \quad (5)$$

mit

$$(v_i, v_j) \in E_F \Leftrightarrow (m(v_i), m(v_j)) \in E_{G'} \quad (6)$$

für einen induzierten Subgraph, bzw.

$$(u, v) \in E_F \Rightarrow (m(u), m(v)) \in E_{G'} \quad (7)$$

2 Lösungsstrategie

In dieser Arbeit wird ein Verfahren vorgestellt, das es ermöglicht zwei chemische Muster auf eine Teilmengen-Relation zu untersuchen. Dabei wird zwischen Anfrage- und Zielmuster unterschieden. Wird eine Teilemengen-Relation festgestellt, so erfolgt die Ausgabe der Lösung durch ein eindeutiges *Mapping* des Anfrage- und Zielmusters. Dieses *Mapping* ist eine bijektive Abbildung die den Atomen des Anfragemusters das dazugehörigen Atom des Zielmusters zuordnet.

Mit dieser Lösung kann herausgefunden werden, ob unterschiedlich definierte Muster dennoch Molekülmengen beschreiben, die einer Teilmengen-Relation entsprechen. Somit können, wie im Kapitel 1.1 beschrieben, zum Beispiel bei einer Beschreibung der gleichen Molekülmenge, redundante Muster identifiziert werden.

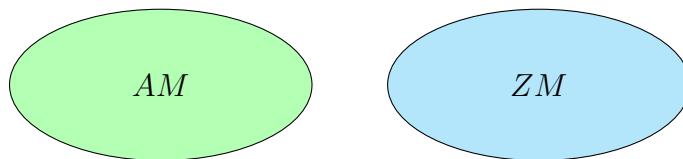
Im folgenden Kapitel soll ein Überblick über die entwickelten Konzepte [11] und die Struktur der Lösung, des in Kapitel 1.1 vorgestellten Problems, gegeben werden.

2.1 Konzepte

Zum besseren Verständnis werden im Folgenden die entwickelten Konzepte des Verfahrens erläutert, die eine Untersuchung auf Teilmengen-Relationen ermöglichen.

2.1.1 Relationen

Die vorliegende Arbeit behandelt den Vergleich von chemischen Mustern anhand von Teilmengen-Relationen. Das entwickelte Verfahren kann chemische Muster auf vier Teilmengen-Relationen überprüfen. Gleichheit ($=$), Untermenge (\subseteq), Obermenge (\supseteq) und Schnitt (\cap). Die genaue Definition dieser Teilmengen-Relationen soll in diesem Abschnitt beschrieben werden, wobei zwischen Anfrage- (A) und Zielmuster (Z) unterschieden wird. Die Mengen an Molekülen, die vom Anfrage- bzw. Zielmuster beschrieben wird, werden als AM und ZM bezeichnet und in diesem Kapitel zur Veranschaulichung als Venn-Diagramme präsentiert.



Gleichheit

Wenn zwei Beschreibungen von chemische Muster in allen atomaren und strukturellen Eigenschaften übereinstimmen, verhalten sich die beschriebenen Molekülmengen wie im folgenden Venn-Diagramm,

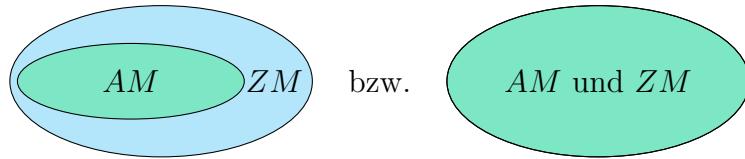


dann gilt:

$$A = Z \quad (8)$$

Untermenge

Wenn das Anfragemuster in seinen atomaren und strukturellen Eigenschaften im Vergleich zum Zielmuster spezifizierter oder gleich weit spezifiziert ist, verhalten sich die Molekülmengen der beiden chemischen Muster wie folgt,

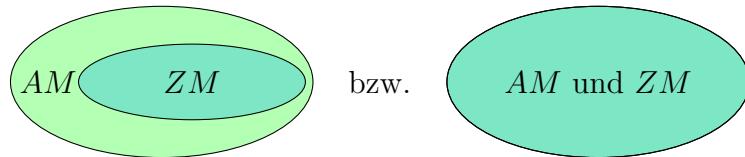


Es gilt:

$$A \subseteq Z \quad (9)$$

Obermenge

Wenn das Anfragemuster in seinen atomaren und strukturellen Eigenschaften im Vergleich zum Zielmuster allgemeiner oder gleich weit spezifiziert ist, verhalten sich die Molekülmengen der beiden chemischen Muster wie folgt,



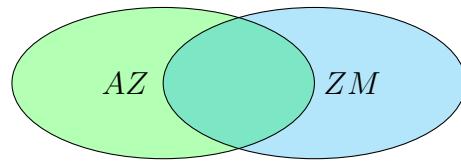
Es gilt:

$$A \supseteq Z \quad (10)$$

Schnitt

Zwei chemische Muster haben in zwei Fällen einen gemeinsamen, nicht leeren Schnitt. Im ersten Fall besteht eine Überlappung zwischen der Mengen der logischen Alternativen

des Anfrage- und Zielmusters. Im zweiten Fall besitzt das Anfragemuster Teilstrukturen, die eine Untermenge von Teilstrukturen des Zielmusters sind, sowie welche, die eine Obermenge von Teilstrukturen des Zielmusters sind.



dann gilt:

$$A \cap Z \quad (11)$$

2.1.2 Valenzzustände

Um zwei chemische Muster auf, die in Kapitel 2.1.1 vorgestellten, Teilmengen-Relationen zu überprüfen, muss es ermöglicht werden die Menge an Eigenschaften, die Atome und Bindungen beschreiben, miteinander vergleichen zu können. Für die komplexen Atombeschreibungen muss dafür der Begriff des Valenzzustandes eingeführt werden. Allgemein versteht man unter einem Valenzzustand einen relevanten Zustand, den ein zu untersuchendes Element einnehmen kann. Durch den anschließenden Vergleich der chemisch möglichen Valenzzustände von zwei zu untersuchenden Atomenbeschreibungen ist dann eine Untersuchung auf Teilmengen-Relationen möglich. In der hier vorgestellten Lösung besteht die Menge der Eigenschaften, die ein Valenzzustand eines Atoms beschreiben kann, aus:

- Atomname
- Atomnummer
- Atomvalenz
- Anzahl der Einfachbindungen
- Anzahl der Doppelbindungen
- Anzahl der Dreifachbindungen
- Formalladung
- Anzahl der Bindungen zu Wasserstoffatomen
- ob das Atom aromatisch ist
- in wie vielen Ringen eines SSSR¹ sich das Atom
- Größe des kleinsten SSSR-Rings in dem sich das Atom befindet
- Anzahl der Bindungen in Ringen

Anhand dieser Eigenschaften ist es möglich die komplexe Struktur der Atombeschreibungen eines chemischen Musters zu kategorisieren und zu analysieren.

2.1.3 Fingerprint

Anhand der Menge von chemisch möglichen Valenzzuständen eines, in chemischem Mustern vorkommenden, Atoms kann ein Fingerprint generiert werden. Ein Fingerprint ist

¹smallest set of smallest rings

eine Repräsentation aller, durch die oben beschriebenen Eigenschaften, möglichen Valenzzustände. Dieser Fingerprint in Form eines Bit-Arrays (siehe Abb. 2) gibt an, ob eine Atombeschreibung ein Valenzzustand beschreibt (Bit auf „1“ gesetzt) oder nicht (Bit auf „0“ gesetzt).

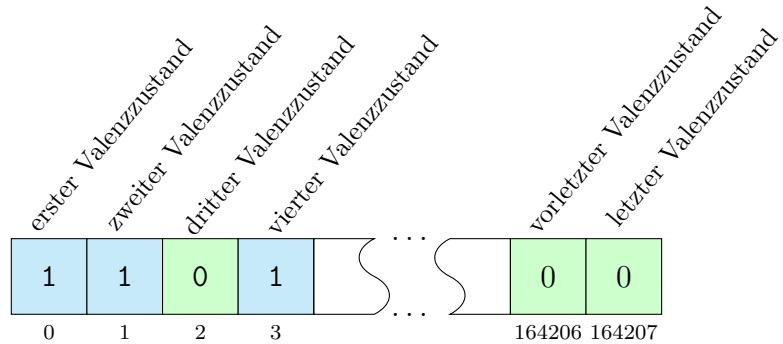


Abbildung 2: Schematische Darstellung eines Fingerprints für eine Atombeschreibung

Der Fingerprint für den Vergleich zwischen zwei Bindungen kann ohne den Begriff der Valenzzustände erstellt werden. Benötigt wird ein Bit-Array bestehend aus fünf Bits, die die folgenden Bindungen beschreiben:

1. Bit: Einfachbindung
2. Bit: Doppelbindung
3. Bit: Dreifachbindung
4. Bit: Ringbindung
5. Bit: Aromatische Bindung

Abbildung 3 zeigt schematisch wie dieses Bit-Array für den Fingerprint einer Bindung aufgebaut ist. Hier wird beispielsweise eine Bindung, die sowohl eine Einfach-, Doppel- oder Ringbindung beschreibt, dargestellt.

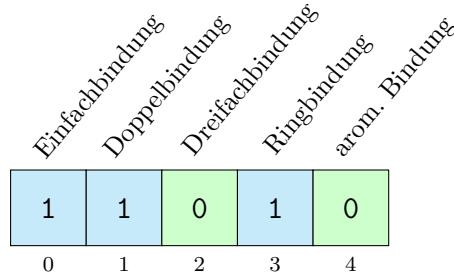


Abbildung 3: Schematische Darstellung eines Fingerprints für eine Bindungsbeschreibung

2.2 Struktur

Anhand der in Kapitel 2.1 erläuterten Konzepte ist es möglich das Verfahren zur Untersuchung auf Teilmengen-Relationen zu realisieren. Es gliedert sich in sechs grundlegende Schritte. Ein grober Überblick über die Struktur wird im Folgenden anhand der Abbildung 4 gegeben. Die Schritte sind in Kapitel 3 detailliert beschrieben.

Die Eingaben des Verfahrens beinhalten ein Anfragemuster mit Atomen (durch „VA“ mit eindeutiger Identifikationsnummer in der Abbildung 4 beschrieben) und Bindungen (durch „EA“ und eindeutiger Nummer beschrieben). Das Anfragemuster hat eine beliebige Größe (Ausgegrauer Bereich ab Atom „VA2“). Das Zielmuster hat die Atom- („VZ“) und Bindungsbezeichnung („EZ“) und hat ebenfalls eine beliebige Größe. Zu den beiden Mustern wird eine der vier möglichen Teilmengen-Relation angegeben.

Im ersten Schritt werden die Muster aufbereitet und gefiltert. Die Aufbereitung erfolgt durch die eindeutige Kennzeichnung jedes Atoms (Knoten) der Muster durch ein *Label*. Anschließend wird je nach Teilmengen-Relation überprüft, ob ein Vergleich der gegebenen Muster möglich ist (siehe Abbildung 4 Schritt 1).

Der zweite Schritt wird ausgeführt, falls die Überprüfung der Muster positiv war. Ansonsten erfolgt keine Ausgabe, da keine bijektive Abbildung gefunden wurde. In ihm werden für alle Atome und Bindungen der beiden Muster Fingerprints erstellt.

Im dritten Schritt werden anhand der gegebenen Teilmengen-Relation die Fingerprints verglichen. Die Fingerprints der Knoten des Anfragemusters werden mit allen Fingerprints der Knoten des Zielmusters paarweise verglichen (in Abbildung 4 entsprechen die grünen Pfeile einer positiven Überprüfung einer Teilmengen-Relation, rote Pfeile einer negativen Überprüfung).

Entsprechen zwei Fingerprints der gegebenen Teilmengen-Relation, so wird im vierten Schritt in der Relations-Matrix der entsprechende Eintrag auf „1“ gesetzt.

Im fünften Schritt werden die gewonnenen Matrizen an einen Algorithmus zur Untersuchung auf Subgraph-Isomorphie übergeben.

Im letzten Schritt wird das Ergebnis des gefundenen Isomorphismus in Form eines eindeutigen *Mappings* gebracht und ausgegeben.

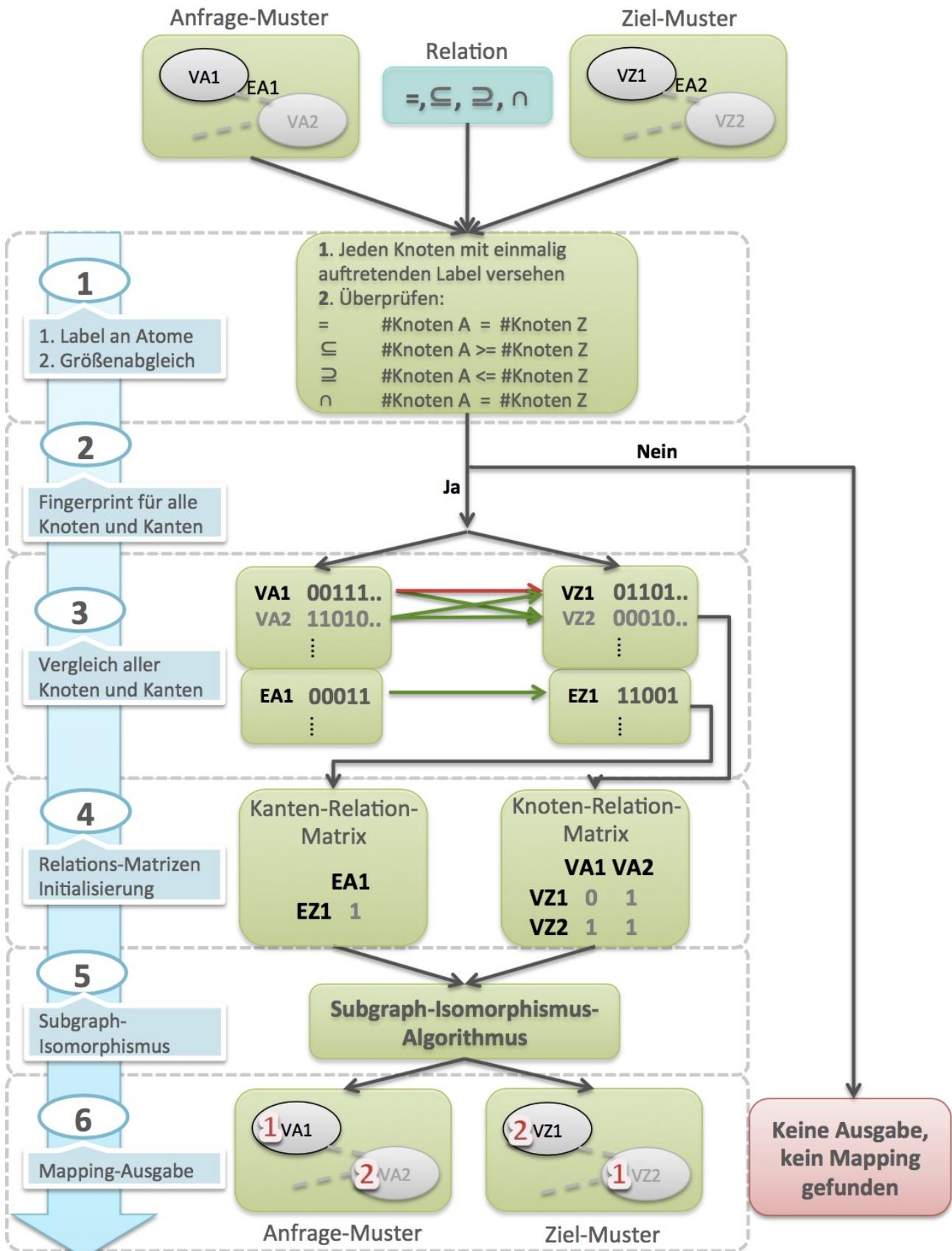


Abbildung 4: Schematischer Ablauf des entwickelten Verfahrens.

3 Realisierung

Das Programm *SmartsSmarts* wurde, wie in Kapitel 2.2 beschrieben, in sechs Schritten realisiert. Im folgenden Kapitel sollen diese Schritte erläutert werden.

Grundlage des Programms zum Vergleich von chemischen Mustern sind die in Kapitel 1.2.2 vorgestellten SMARTS-Ausdrücke. Dem Programm werden neben dem Anfrage-SMARTS und Ziel-SMARTS zusätzlich die zu untersuchende Teilmengen-Relation übergeben.

3.1 Genutzte Software

Das in dieser Arbeit vorgestellte Verfahren wurde im Rahmen der NAOMI-Softwarebibliothek [15] des Zentrums für Bioinformatik (ZBH) der Universität Hamburg implementiert. Aus den folgenden Modulen wurden Funktionen und Klassen genutzt:

1. Chemistry
2. SmartsParser
3. Molecule
4. SmartsMatcher

An welchen Stellen das *SmartsSmarts*-Programm die Module nutzt, wird in Abbildung 13 (Kapitel B) gezeigt.

3.2 Aufarbeitung und Auswahl

Der Anfrage- und Ziel-SMARTS wird zunächst in eine Form gebracht, die es im letzten Schritt des Verfahrens ermöglicht das *Mapping* zu erzeugen. Die dazu genutzten *Label* sind eindeutige Identifikationsnummern, die als :ID an die Knoten eines SMARTS-Ausdrucks angefügt werden können. Dabei wird jedes Label einmalig an das Vergleichspaar aus Anfrage- und Ziel-SMARTS vergeben. Bestehende *Label* werden übernommen.

Anschließend wird für Anfrage- und Ziel-SMARTS überprüft, ob ein Vergleich grundsätzlich anhand der Knoten-Anzahl und der gegebenen Relation möglich ist (siehe Tabelle 7). Für die Schnitt-Relation (\cap) ist zu beachten, dass Anfrage- und Ziel-SMARTS die gleiche Anzahl an Knoten haben müssen, damit ein nicht leerer Schnitt vorliegt. Der Grund dafür ist, dass die Schnitt-Menge zweier SMARTS-Muster ein SMARTS-Ausdruck beschreiben, welcher eine Untergruppe von Anfrage- und Ziel-SMARTS sein soll. Wären die beiden Muster unterschiedlich groß, wäre der nicht leere Schnitt maximal so groß wie der kleinere SMARTS-Ausdruck. Für das größere Muster kann also keine Untermengen-Relation vorliegen.

Tabelle 7: Anzahlüberprüfung der Knoten des Anfrage-SMARTS (*Query*) und Ziel-SMARTS (*Target*)

Relation	Bedingung
$=$ and \cap	$QueryNodeCount = TargetNodeCount$
\subseteq	$QueryNodeCount \geq TargetNodeCount$
\supseteq	$QueryNodeCount \leq TargetNodeCount$

3.3 Fingerprintgenerierung

Die Basis für die Untersuchung auf eine bestimmte Teilmengen-Relation wird in diesem Schritt gewonnen. Anhand der Fingerprints von Knoten und Kanten können zwei chemische Muster miteinander verglichen werden. In Kapitel 2.1.2 und 2.1.3 wurde bereits das Konzept des Fingerprints beschrieben. Beide SMARTS-Ausdrücke werden in eine digitale Graphenstruktur übertragen. Über diesen Graphen wird anschließend iteriert, um für jeden Knoten und jede Kante den Fingerprint zu erstellen.

Für Knoten werden, wie Tabelle 8 zeigt, Eigenschaften ausgelesen und anhand dieser der Fingerprint generiert.

Tabelle 8: Eigenschaften der Valenzzustände für Atome und ihre zugehörigen SMARTS-Atomprimitive

Eigenschaft	abgeleitet aus SMARTS-Atomprimitiv
Element	* , a, A und alle Elementnamen
Atomnummer	#n
Atomvalenz	v<n>
Anzahl der Einfachbindungen	D<n>, H<n>
Anzahl der Doppelbindungen	D<n>
Anzahl der Dreifachbindungen	D<n>
Formalladung	+<n>, -<n>
Anzahl der gebundenen Wasserstoffatome	D<n>, H<n>
ob das Atom aromatisch ist	* , a, A und alle Atomnamen
Anzahl der Ringe eines SSSR in dem sich das Atom befindet	R<n>
Größe des kleinsten SSSR-Ring in dem sich das Atom befindet	r<n>
Anzahl der Bindungen in Ringen	x<n>

Die Generierung erfolgt durch eine sequentielle Abfrage und Abgleich aller Eigenschaften und dementsprechender Bit-Setzung. Ausgegangen wird von einem Bit-Array bei dem jedes Bit auf „0“ gesetzt ist. Im ersten Schritt werden, wie im Algorithmus 1 beschrieben, die Bits der Valenzzustände, die nicht durch die Atomnamen-Eigenschaften des Knotens beschrieben sind, auf „1“ gesetzt. Zudem wird in diesem Schritt bereits die Aromazität untersucht, die im späteren Verlauf genutzt wird.

Die Eigenschaft der Atommasse ($<n>$) kann anhand des Fingerprints nicht verglichen werden, da die Menge an möglichen Atommassen nicht im Umfang der Valenzzustände enthalten ist. Diese Eigenschaft zwischen zwei zu vergleichenden Knoten wird mit Einschränkungen (siehe Kapitel 6.1) vor der Fingerprintgenerierung abgeglichen. Unterbrechungen durch „..“ innerhalb von SMARTS-Ausdrücken werden in dem hier entwickelten Konzept nicht berücksichtigt.

Algorithm 1 Fingerprintgenerierung Ausschnitt 1 [11]

```

1: procedure TAKEPROPERTYFINGERPRINT(node)
2:   node_is_aliphatic = true
3:   node_is_aromatic = true
4:   for all (i ∈ [1, nof_valenceStates]) do
5:     sec_temp_bit_array[i] = true
6:   for all (node_Symbol = GETNODESSYMBOL(node)) do ▷ Abbildung Atomname
7:     for all (i ∈ [1, nof_valenceStates]) do
8:       temp_bit_array[i] = false
9:     if (node_symbol ≠ SMARTS_ANY_ATOM)
10:      for all (i ∈ [1, nof_valenceStates]) do
11:        if (node_symbol = GETSYMBOL(valencestates[i]))
12:          temp_bit_array[i] = true
13:        if (NODESYMBOLNEGATIONFLAG(node) == true)
14:          for all (i ∈ [1, nof_valenceStates]) do
15:            if (temp_bit_array[i] == true)
16:              temp_bit_array[i] = false
17:            else
18:              temp_bit_array[i] = true
19:        else
20:          for all (i ∈ [1, nof_valenceStates]) do
21:            temp_bit_array[i] = true
22:          if (NODESYMBOLNEGATIONFLAG(node) == true)
23:            if (NODEISALIPHATIC(node) == NODEISAROMATIC(node))
24:              for all (i ∈ [1, nof_valenceStates]) do
25:                temp_bit_array[i] = false
26:              if (NODEISALIPHATIC(node)! = true)
27:                node_is_aliphatic = false
28:              if (NODEISAROMATIC(node)! = true)
29:                node_is_aliphatic = false
30:            sec_temp_bit_array = temp_bit_array ∧ sec_temp_bit_array
31:          bit_array = bit_array ∨ sec_temp_bit_array

```

Die meisten Eigenschaften können anschließend durch einen Abgleich von Graphen-Eigenschaften und Valenzzuständen im Bit Array weiter geprüft werden und gegebenenfalls weitere Bits auf "0" gesetzt werden. Im Algorithmus 2 und Algorithmus 3 sind die Verarbeitung der Eigenschaften der Bindungen ($D<n>$) und der Aromazität beschrieben, da diese nicht direkt aus der Graphenstruktur abzulesen sind.

Algorithm 2 Fingerprintgenerierung Ausschnitt 2 [11]

```

1: for all ( $i \in [1, \text{nof\_valenceStates}]$ ) do
2:    $\text{temp\_bit\_array}[i] = \text{false}$ 
3: for all ( $\text{node\_degree} = \text{GETNODESDEGREE}(\text{node})$ ) do            $\triangleright$  Abbildung D<n>
4:   for all ( $i \in [1, \text{nof\_valenceStates}]$ ) do
5:     if  $\left( \begin{array}{l} (\text{node\_degree} \neq (\text{GETNOFBONDS}(\text{valencestates}[i]) \\ - \text{GETNOFHYDROGENS}(\text{valencestates}[i]))) \text{ and} \\ (\text{NODEDEGREENEGATIONFLAG}(\text{node}) == \text{false}) \end{array} \right)$ 
6:     or  $\left( \begin{array}{l} (\text{node\_degree} = (\text{GETNOFBONDS}(\text{valencestates}[i]) \\ - \text{GETNOFHYDROGENS}(\text{valencestates}[i]))) \text{ and} \\ (\text{NODEDEGREENEGATIONFLAG}(\text{node}) == \text{true}) \end{array} \right)$ 
7:     and  $\text{bit\_array}[i] == \text{true}$ 
8:      $\text{bit\_array}[i] = \text{false}$ 
9:    $\text{temp\_bit\_array} = \text{temp\_bit\_array} \vee \text{bit\_array}$ 
10:   $\text{bit\_array} = \text{bit\_array} \wedge \text{temp\_bit\_array}$ 

```

Algorithm 3 Fingerprintgenerierung Ausschnitt 3 [11]

```

1: for all ( $i \in [1, \text{nof\_valenceStates}]$ ) do            $\triangleright$  Abbildung Aromazität
2:   if  $\left( \begin{array}{l} (\text{node\_is\_aromatic} == \text{true} \text{ and } \text{node\_is\_aliphatic} == \text{false}) \\ \text{and } \text{IsAROMATIC}(\text{valencestates}[i]) == \text{false} \end{array} \right)$ 
3:   or  $\left( \begin{array}{l} (\text{node\_is\_aromatic} == \text{false} \text{ and } \text{node\_is\_aliphatic} == \text{true}) \\ \text{and } \text{IsAROMATIC}(\text{valencestates}[i]) == \text{true} \end{array} \right)$ 
4:   and  $\text{bit\_array}[i] == \text{true}$ 
5:    $\text{bit\_array}[i] = \text{false}$ 

```

Für die Kanten können die Eigenschaften zur Fingerprintgenerierung direkt aus der Graphenstruktur abgelesen werden. Tabelle 9 stellt dabei die Eigenschaften, die der Fingerprint enthält, in Zusammenhang mit den darstellenden SMARTS-Bindungsprimitiven.

Tabelle 9: Für die Fingerprintgenerierung genutzte Eigenschaften einer Bindung und dazugehörige Bindungsprimitive

Eigenschaft	SMARTS-Bindungsprimitive
Einfachbindung	$-$, $/$, \backslash , \sim
Doppelbindung	$=$, \sim
Dreifachbindung	$\#$, \sim
Ringbindung	$@$, \sim
aromatische Bindung	$:$, \sim

Nicht alle Eigenschaften eines SMARTS-Ausdruck können in dem hier vorgestellten Verfahren verarbeitet werden. Zum Beispiel können die in Kapitel 1.2.2 beschriebenen Re-

kursionen, die für SMARTS-Ausdrücke genutzt werden, um Umgebungen von SMARTS-Primitiven zu definieren, nicht beachtet werden. Für weitere nicht unterstützte SMARTS-Primitve siehe Kapitel 6.1. Lösungsansätze werden in Kapitel 7 beschrieben.

3.4 Vergleich und Aufbau Relations-Matrix

Für die Untersuchung auf einen Subgraphisomorphismus von zwei chemischen Mustern benötigt man Initial-Matrizen für Knoten und Kanten, die die möglichen bijektiven Abbildungen der einzelnen Elemente repräsentieren. Anschließend wird die Initial-Matrix der Knoten und die der Kanten zusammen mit den Adjazenzmatrizen¹ der beiden chemischen Mustern an einen Algorithmus zur Berechnung des Isomorphismus übergeben. In dieser Arbeit wurde dazu der Algorithmus nach Ullmann [16] verwendet.

Anhand der Fingerprints, die für alle Bestandteile der beiden zu untersuchenden SMARTS-Ausdrücke erstellt wurden, können Rückschlüsse auf die Teilmengen-Relationen gezogen werden.

Für den Vergleich werden Bit-Operationen genutzt. Die Tabelle 10 zeigt welche Bit-Operationen dazu ausgeführt werden müssen, um die Teilmengen-Relationen zu überprüfen. Anschließend können mit den Ergebnissen der Vergleiche die Initial-Matrizen initialisiert werden. In diesem Kontext werden sie als Relations-Matrizen bezeichnet, da sie Rückschlüsse auf Teilmengen-Relationen geben.

Tabelle 10: Ergibt die durchgeführte Bit-Operation aus der linken Spalte ein Bit-Array, welches keine auf „1“ gesetzten Bits enthält, so gilt die in der rechten Spalte angegebene Teilmengen-Relation [11]

Bit-Operation	Teilmengen-Relation
$\text{QueryArray} \oplus \text{TargetArray}$	$\text{Query} = \text{Target}$
$(\text{QueryArray} \wedge \text{TargetArray}) \oplus \text{QueryArray}$	$\text{Query} \subseteq \text{Target}$
$(\text{TargetArray} \wedge \text{QueryArray}) \oplus \text{TargetArray}$	$\text{Query} \supseteq \text{Target}$

Wird für zwei Knoten oder zwei Kanten festgestellt, dass die angegebene Teilmengen-Relation besteht so wird der Matrix Eintrag auf „1“ gesetzt. Abbildung 5 zeigt ein Beispiel einer initialisierten Relations-Matrix von zwei SMARTS-Ausdrücken.

$$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Abbildung 5: Die Matrix ist ein Beispiel einer Relations-Matrix der Teilmengen-Relation \subseteq , die mit dem Anfrage-SMARTS $\text{P}(=\text{S})(\text{S})\text{S}$ und dem Ziel-SMARTS $\text{S}=\text{P}\sim[\ast,\#1]$ initialisiert wurde. Die Spalten entsprechen den Knoten des Anfrage-SMARTS und die Zeilen den Knoten des Ziel-SMARTS

¹Matrix zugehörig zu einem Graphen. Matrixeintrag „1“ bedeutet, dass eine Kante zwischen zwei Knoten besteht. Matrixeintrag „0“, dass es keine Kante zwischen zwei Knoten gibt. Dabei besitzt sie für jeden Knoten eine Zeile (i) und eine Spalte (j). Ein Element (i, j) gibt die Adjazenz vom i -ten und j -ten Knoten an. [14]

Zur Überprüfung der Teilmengen-Relation $Query \cap Target$ kann keine einfache Bit-Operation durchgeführt werden. Durch Zuhilfenahme der in Tabelle 10 beschriebenen Bit-Operationen können zwei Elemente auf $Query \cap Target$ geprüft werden. Zwei chemische Muster haben einen wie in Kapitel 2.1.1 beschriebenen nicht leeren Schnitt in zwei Fällen:

Sei $G^Q(V^Q, E^Q)$ die Graphenrepräsentation des Anfrage-SMARTS und $G^T(V^T, E^T)$ die Graphenrepräsentation des Ziel-SMARTS. Weiter sei ein Knoten $v \in V$ bestehend aus der Menge an logischen Alternativen LA_v , sowie eine Kante $e \in E$ bestehend aus der Menge an logischen Alternativen LA_e . Für ein mögliches Mapping M_{QT} , bestehend aus Knoten-Paaren (v_Q, v_T) und den impliziten Kanten-Paaren (e_Q, e_T) , muss gelten:

$$\begin{aligned} & \left[\forall(v_Q, v_T) \in M_{QT} : \left(LA_{v_Q} =, \cap, \subset, \supset LA_{v_T} \right) \right. \\ & \quad \wedge \forall(e_Q, e_T) \in M_{QT} : \left(LA_{e_Q} =, \cap, \subset, \supset LA_{e_T} \right) \Big] \\ & \wedge \left[\exists(v'_Q, v'_T) \in M_{QT} : \left(LA_{v'_Q} \neq LA_{v'_T} \wedge LA_{v'_Q} \not\subset LA_{v'_T} \wedge LA_{v'_Q} \not\supset LA_{v'_T} \right) \right. \\ & \quad \vee \exists(e'_Q, e'_T) \in M_{QT} : \left(LA_{e'_Q} \neq LA_{e'_T} \wedge LA_{e'_Q} \not\subset LA_{e'_T} \wedge LA_{e'_Q} \not\supset LA_{e'_T} \right) \Big] \end{aligned} \quad (1)$$

oder

$$\begin{aligned} & \left(\forall(v_Q, v_T) \in M_{QT} : \left(LA_{v_Q} =, \cap, \subset, \supset LA_{v_T} \right) \right. \\ & \quad \wedge \forall(e_Q, e_T) \in M_{QT} : \left(LA_{e_Q} =, \cap, \subset, \supset LA_{e_T} \right) \Big) \\ & \wedge \left[\left(\exists(v'_Q, v'_T), (v''_Q, v''_T) \in M_{QT} \setminus \{(v_Q, v_T)\} : \right. \right. \\ & \quad \left((v'_Q, v'_T) \neq (v''_Q, v''_T) \wedge LA_{v'_Q} \subset LA_{v'_T} \wedge LA_{v''_Q} \supset LA_{v''_T} \right) \Big) \\ & \quad \vee \left(\exists(e'_Q, e'_T), (e''_Q, e''_T) \in M_{QT} \setminus \{(e_Q, e_T)\} : \right. \\ & \quad \left. \left. \left((e'_Q, e'_T) \neq (e''_Q, e''_T) \wedge LA_{e'_Q} \subset LA_{e'_T} \wedge LA_{e''_Q} \supset LA_{e''_T} \right) \right) \right] \end{aligned} \quad (2)$$

Beispiel 3.4.1 zeigt zwei SMARTS-Ausdrücke für die Formel (1) gilt. Ein Knoten erfüllt die Eigenschaft des zweiten Teils der Formel. Die restlichen Knoten erfüllen den ersten Teil.

Beispiel 3.4.1.

Anfrage-SMARTS	$[C1, I]$	\sim	0
Teilmengen-Relation	\cap	$=$	$=$
Ziel-SMARTS	$[C1, F]$	\sim	0

Beispiel 3.4.2 zeigt zwei SMARTS-Ausdrücke bei denen Formel (2) gilt. Für mindestens einen Knoten gilt \supset . In diesem Beispiel ist es jeweils der letzte Knoten. Des Weiteren gilt für mindestens einen Knoten- oder Bindungspaar \subset . In diesem Beispiel sind dies die jeweils ersten beiden Knoten und die Bindung.

Beispiel 3.4.2.

Anfrage-SMARTS	$[C; !R]$	\sim	0
Teilmengen-Relation	\subset	$=$	\supset
Ziel-SMARTS	$[#6]$	\sim	$[OH]$

3.5 Subgraphisomorphie Analyse

Anhand der Relations-Matrizen von Knoten und Kanten ist es nun möglich zwei Muster auf einen Subgraphisomorphismus zu untersuchen.

Mit der für die Knoten erzeugten Relations-Matrix, den jeweiligen Adjazenz-Matrizen und der Information zur Vereinbarkeit der Kanten durch die entsprechende Relations-Matrix wird durch den Algorithmus nach Ullmann ein Isomorphismus gesucht. Die gefundenen bijektiven Abbildungen werden in Form von Permutationsmatrizen ausgegeben.

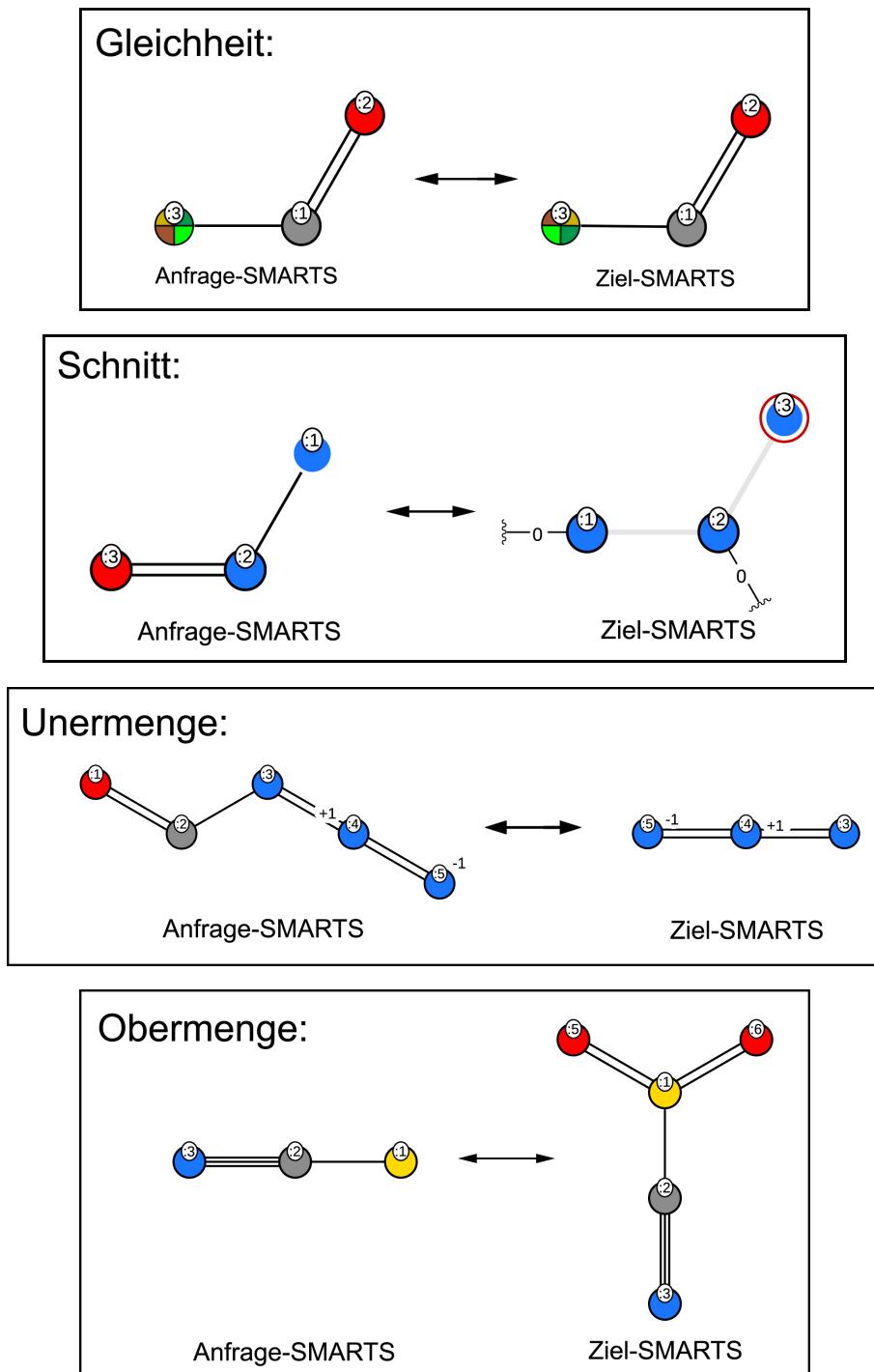
Bei der Überprüfung auf eine Schnitt-Relation werden die Permutationsmatrizen in einem weiteren Schritt erneut auf diese Teilmengen-Relation geprüft. In diesem Schritt wird für die resultierende bijektive Abbildung geprüft, ob Formel (1) oder Formel (2) gilt.

3.6 Kennzeichnung

Zur eindeutigen Zuordnung der gefundenen Isomorphismen wurde anschließend eine Änderung der *Label* des Ziel-SMARTS erzeugt, die als Abbildung des Anfrage-SMARTS dient. Durch diese Kennzeichnung der Knoten eines Anfrage-SMARTS in einem Ziel-SMARTS wird bei der Ausgabe deutlich, welche Knoten eine Abbildung darstellen. Beispiel 3.6.1 zeigt für die vier möglichen Teilmengen-Relationen die Ausgabe der gefundenen *Mappings*. Abbildung 6 zeigt eine Visualisierung (erzeugt mit SMARTSviewer [25][17]) der Ergebnisse.

Beispiel 3.6.1.

Gleichheit:	Anfrage-SMARTS	$[C:1] (= [0:2]) [Cl, Br, I, F:3]$
	Ziel-SMARTS	$[0:2] = [C:1] - [F, Cl, Br, I:3]$
Schnitt:	Anfrage-SMARTS	$[#7:1] - [N:2] = [0:3]$
	Ziel-SMARTS	$[!#7:3] \sim [NX2:2] \sim [NX1:1]$
Untermenge:	Anfrage-SMARTS	$[0:1] = [C:2] [N:3] = [N+:4] = [N-:5]$
	Ziel-SMARTS	$[N:3] = [N+:4] = [N-:5]$
Obermenge:	Anfrage-SMARTS	$[S:1] [C:2] \# [N:3]$
	Ziel-SMARTS	$[S:1] (= [0:5]) (= [0:6]) [C:2] \# [N:3]$

Abbildung 6: Visualisierung der gefundenen *Mappings* aus Beispiel 3.6.1.

4 Evaluation

4.1 Validierung

Nachdem die Korrektheit der einzelnen Funktionen getestet wurde (vgl. Anhang D), erfolgte eine anschließende Validierung des implementierten Verfahrens.

Die Validierung erfolgte durch den Vergleich von Molekülmengen. Es wurden 6 000 Moleküle aus der ZINC-Datenbank [18] genutzt (siehe Anhang F).

Die genutzten SMARTS-Ausdrücke [19] wurden vom Zentrum für Bioinformatik Hamburg veröffentlicht. Der Datensatz beinhaltet 1 469 SMARTS-Ausdrücke (siehe Anhang F). Da das implementierte Verfahren nicht alle SMARTS-Primitive verarbeiten kann (siehe Kapitel 6.1), konnten 1 038 SMARTS-Ausdrücke für die Validierung genutzt werden.

4.1.1 Durchführung

Das implementierte Verfahren wurde validiert, indem die Molekülmengen der SMARTS-Ausdrücken verglichen wurden. Das Vorgehen gliedert sich dabei in zwei Schritte.

Im ersten Schritt wird für zwei SMARTS-Ausdrücke und einer Teilmengen-Relation überprüft, ob ein *Mapping* gefunden wurde.

Schritt zwei überprüft das Verhalten von Anfrage- und Ziel-SMARTS im Bezug auf das Treffen (*Matching*) von Molekülen. Wenn ein *Mapping* gefunden wird, wird über die Molekülliste iteriert und geprüft, ob Anfrage- und Ziel-SMARTS ihrer Teilmengen-Relation entsprechend das gleiche Molekül treffen. Tabelle 11 zeigt welche Bedingungen für welche Teilmengen-Relation erfüllt sein muss.

Tabelle 11: Übersicht der Bedingungen, die die SMARTS-Ausdrücke eines *Mappings* erfüllen müssen, um für eine Teilmengen-Relation als valides *Mapping* zu gelten. Betrachtet werden dabei die Teilmengen-Relationen Gleichheit, Untermenge und Obermenge.

Teilmengen-Relation	Bedingung des <i>Matchings</i>
Gleichheit (=)	Anfrage-SMARTS und Ziel-SMARTS müssen gleiches Molekül treffen
Untermenge (\subseteq)	Wenn Anfrage-SMARTS Molekül trifft, muss auch Ziel-SMARTS treffen
Obermenge (\supseteq)	Wenn Ziel-SMARTS Molekül trifft, muss auch Anfrage-SMARTS treffen

Falls kein *Mapping* gefunden wird, wird ebenfalls über die Molekülliste iteriert und für jedes Molekül geprüft, ob beide SMARTS-Ausdrücke das zu überprüfende Molekül treffen. Ist dies der Fall, werden die beiden SMARTS-Ausdrücke zusammen mit der Teilmengen-Relation ausgegeben, um manuell ausschließen zu können, dass ein *Mapping* nicht gefunden wurde. Solch ein Fall tritt auf, wenn innerhalb eines Moleküls unterschiedliche Atome durch den SMARTS-Ausdruck beschrieben werden.

Die Schnitt-Relation kann mit dieser Methode nicht validiert werden, da nicht gewährleistet ist, dass der Teil des Schnittes des gefundenen *Mappings* ein Molekül trifft. Bei Beispiel

4.1.1 und Abbildung 7 tritt der Fall ein, dass anhand der Molekülmenge eine Validierung nicht möglich ist. Beispiel 4.1.1 zeigt ein *Mapping* der Schnitt-Relation. Wird nun das *Matching*-Verhalten der beiden SMARTS-Ausdrücke in Bezug auf das Molekül aus Abbildung 7 verglichen, wird festgestellt, dass trotz eines gefundenen *Mappings* die Molekülmenge nicht vergleichbar ist.

Beispiel 4.1.1.

Schnitt-<i>Mapping</i>:	Anfrage-SMARTS	[C1,I:1]~[c:2]
Ziel-SMARTS	[C1,F:1]~[c:2]	

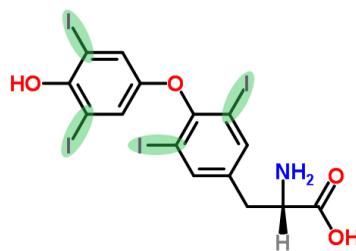


Abbildung 7: Levothyroxin mit grün gekennzeichneten Teilstrukturen, bei denen der Anfrage-SMARTS das Molekül trifft [20]

4.1.2 Resultate

Nachdem durch die Validierung eventuelle Fehlverhalten aufgedeckt und behoben wurden, war es möglich die Validierung erfolgreich durchzuführen. Bei einem Beispiel des Validierungsverfahrens wurde als Übergabeparameter der Anfrage-SMARTS $\text{N} \sim [\#6]$ zusammen mit der Angabe, dass alle vier Teilmengen-Relationen überprüft werden sollen, an das Programm *SmartsSmarts* übergeben. Als Ziel-SMARTS wurden die SMARTS-Ausdrücke als Liste übergeben. Es wurden vier Untermengen-*Mappings* und 109 Obermengen-*Mappings* gefunden, welche alle durch die Validierung bestätigt werden konnten. Die Fälle, bei denen ohne *Mapping* dennoch beide SMARTS-Ausdrücke ein Molekül getroffen haben, wurden wie beschrieben manuell überprüft. Es konnte kein fehlendes *Mapping* festgestellt werden.

Neben diesem Beispiel wurde die Validierung mehrfach mit unterschiedlichen Anfrage-SMARTS erfolgreich durchgeführt.

4.2 Laufzeitanalyse

Für die Laufzeitanalyse wurde die benötigte Zeit des Verfahrens in Zusammenhang mit der Größe der genutzten SMARTS-Ausdrücke gebracht. Durchgeführt wurde die Laufzeitanalyse auf einem Rechner mit Intel® Core™ 7-3770S CPU 3.10GHz. Dabei ist zu beachten, dass nebenläufige Prozesse nicht ausgeschlossen werden können.

Abbildung 8 zeigt, wie sich die Laufzeit (in Sekunden) bei Anstieg der Knotenanzahl von Anfrage- und Ziel-SMARTS verhält, wenn ein *Mapping* gefunden wird. Diese Form der Laufzeitanalyse ist für alle vier Teilmengen-Relationen möglich. Schwankungen der Laufzeit sind auf die Komplexität der SMARTS-Ausdrücke zurückzuführen. Je verzweigter die Graphenstruktur ist, desto länger dauert der Aufbau der Relations-Matrix. Deutlich ist zu erkennen, dass die Schnitt-Relation etwa die doppelte Zeit benötigt. Das liegt daran, dass nach der Untersuchung auf einen Isomorphismus durch den Algorithmus nach Ullmann die Schnitt-Relation erneut überprüft wird.

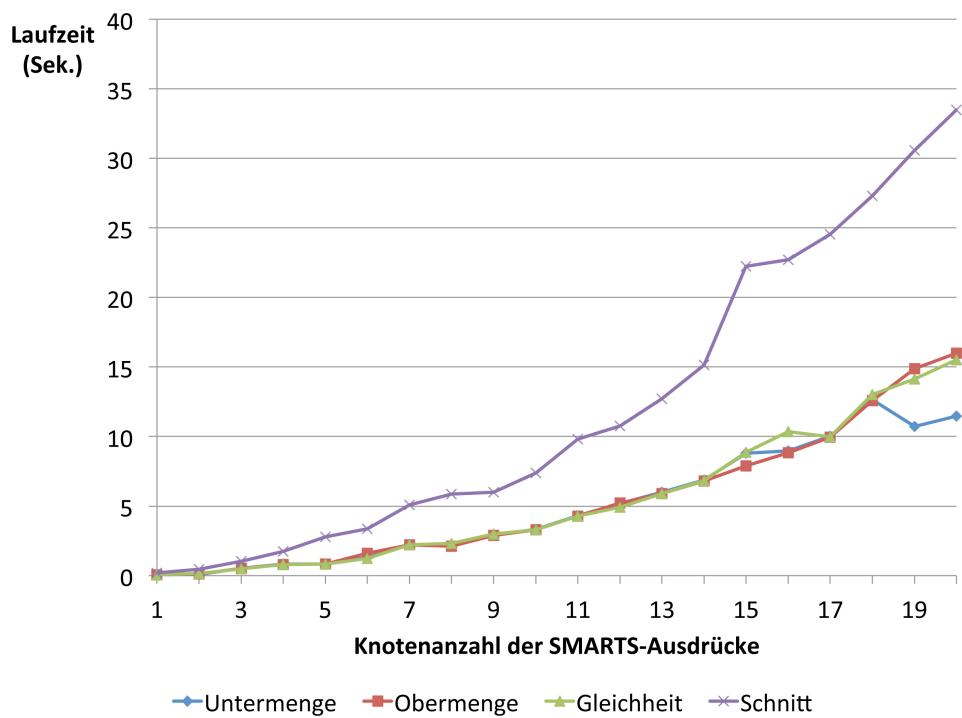


Abbildung 8: Auftragung der Laufzeit in Sekunden, gegen Anzahl der Knoten eines SMARTS-Ausdrucks, wobei Anfrage- und Ziel-SMARTS die gleiche Größe besitzen. Jede Kurve repräsentiert eine Teilmengen-Relation.

Für die Unter- und Obermenge lässt sich der Fall untersuchen, bei dem die Knoten-Anzahl der SMARTS-Ausdrücke unabhängig voneinander variiert. Abbildung 9 verdeutlicht die Zunahme der Laufzeit (in Sekunden) bei einer ansteigenden Größe der Anfrage-SMARTS zusammen mit einer gleichbleibenden Größe der Ziel-SMARTS. Die Größe der gleichbleibenden SMARTS-Ausdrücke liegt bei einer Überprüfung auf eine Untermengen-Relation konstant bei einem Knoten. Wird auf eine Obermengen-Relation geprüft, so sind die SMARTS-Ausdrücke konstant bei einer Länge von 21 Knoten.

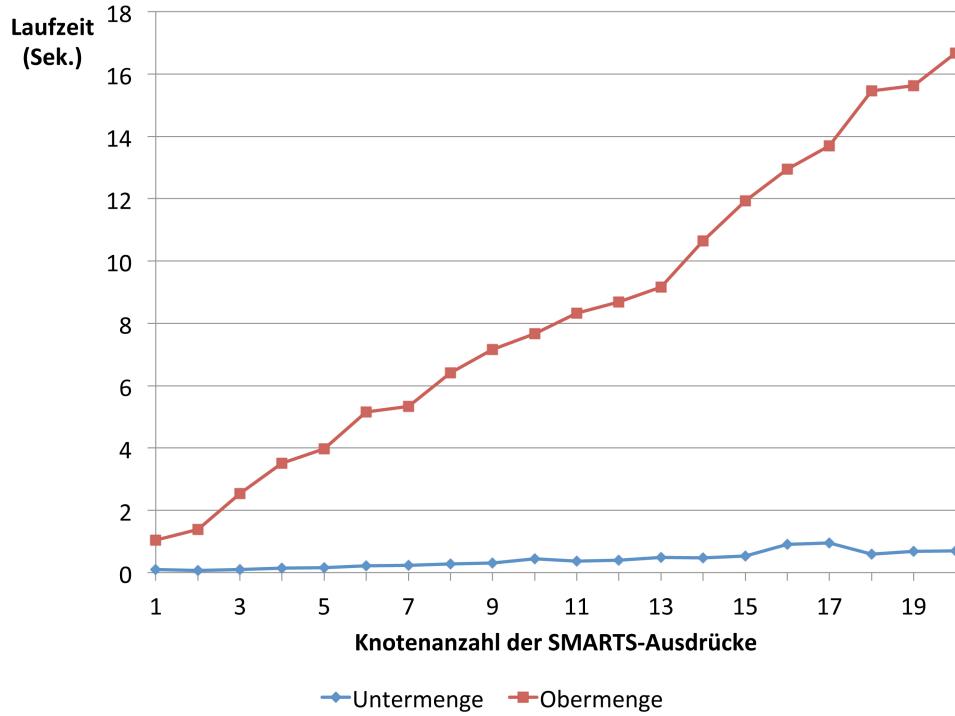


Abbildung 9: Auftragung der Laufzeit in Sekunden gegen Anzahl der Knoten des Ziel-SMARTS. Der Anfrage-SMARTS besteht konstant auf einem (Untermenge) bzw. konstant auf 21 (Obermenge) Knoten.

Diese Laufzeitanalyse ergibt eine annähernd lineare Regression. Der genutzte Algorithmus nach Ullmann benötigt im schlimmsten Fall, bei einem Anfrage-SMARTS mit einer Knotenanzahl n und einem Ziel-SMARTS mit der Knotenanzahl m , eine Laufzeit, die in $\mathcal{O}(m^n n^2)$ liegt. Bei dem Aufbau der Relations-Matrix werden nm Vergleiche und bei dem Aufbau der Kanten-Verträglichkeit $(n - 1)(m - 1)$ Vergleiche benötigt. Insgesamt lässt sich die asymptotische *Worst-case* Laufzeit des entwickelten Verfahrens auf $\mathcal{O}(m^n n^2)$ abschätzen.

5 Experimente und Auswertung

Nach Abschluss der Validierung wurde das vorgestellte und implementierte Verfahren mit realen Daten getestet, die einen möglichen Anwendungsbereich des Verfahrens darstellen. Bei den durchgeführten Experimenten handelt es sich um eine Untersuchung bei der chemische Filter-Sets, bestehend aus SMARTS-Ausdrücken, auf Überlappungen überprüft wurden. Es wurden acht verschiedene SMARTS-Filter miteinander verglichen. Diese Filter wurden im Rahmen der einer Datenbank-Veröffentlichung durch ChEMBL [21][22][23] aus den folgenden Quellen zusammengetragen:

1. Bristol-Myers Squibb HTS Deck Filters (**BMS**)
2. University of Dundee NTD Screening Library Filters (**Dundee**)
3. Glaxo Wellcome Hard Filters (**Glaxo**)
4. Inpharmatica Unwanted Fragments (**Inpharmatica**)
5. Pfizer LINT filters (**LINT**)
6. NIH MLSMR Excluded Functionality Filters (**MLSMR**)
7. Pan Assay Interference Compounds Filters (**PAINS**)
8. SureChEMBL Data (**SureChEMBL**)

Diese Filter-Sets bestehend aus SMARTS-Ausdrücken werden als sogenannte *structural alerts* eingesetzt. *Structural alerts* werden genutzt um unter Umständen problematische Moleküle bei der Wirkstoffentwicklung herauszufiltern. So werden Moleküle mit toxikologisch Substrukturen oder funktionellen Gruppen herausgefiltert. Ebenfalls werden bekannte instabile Moleküle, die bei Tests möglicherweise ein Hindernis sind oder solche, die beim HTS (*High throughput Screening*) [3] häufige Treffer darstellen herausgefiltert.[22] Die Filter-Sets wurden von Christian Laggner auf falsche SMARTS-Ausdrücke untersucht und diese gegebenenfalls korrigiert (siehe Anhang F).

Tabelle 12: Anzahl der SMARTS-Ausdrücke pro *structural alert*-Filter vor und nach der Übergabe an das Programm *SmartsSmarts* (siehe Kapitel 3.2).

Filter-Set	Anzahl SMARTS-Ausdrücke	nach Übergabe an <i>SmartsSmarts</i>
BMS	180	87
Dundee	105	100
Glaxo	55	54
Inpharmatica	92	83
LINT	58	48
MLSMR	116	105
PAINS	481	423
SureChEMBL	166	148

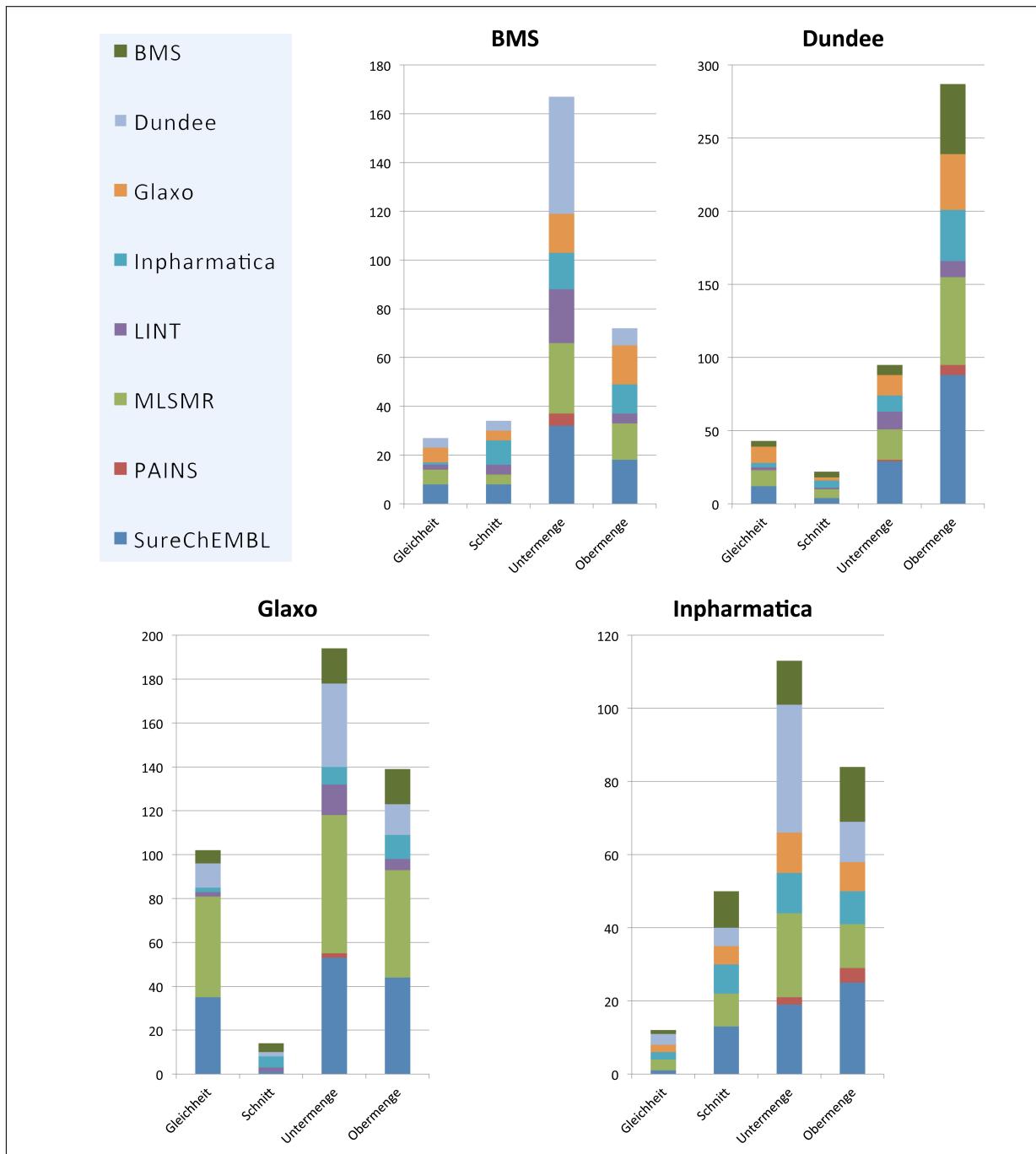


Abbildung 10: Erster Teil der Experimente: Dargestellt ist für wie viele Ziel-SMARTS ein *Mapping* zu den Anfrage-SMARTS der einzelnen SMARTS-Filter (BMS, Dundee, Glaxo und Inpharmatica) gefunden wurde. Die *Mappings* sind farbig markiert, je nachdem aus welchen anderen Filtern sie stammen. Zu beachten sind die unterschiedlichen Skalierungen (vgl. Tabelle 12).

5.1 Diskussion der Experimente

Anhand der Ergebnisse, die für die Vergleiche der SMARTS-Filter gewonnen wurden, können Rückschlüsse auf die Art der *structural alters* gezogen werden. Im Folgenden wird auf die Ergebnisse der einzelnen Filter eingegangen. Anschließend wird ein chemisches Muster gezeigt, welches in jedem Filter (ohne PAINS) in einer allgemeinen oder spezifizierten Form vorlag.

BMS (siehe Abbildung 10)

Nach Durchführung der Vergleiche von dem Filter BMS mit allen anderen Filtern wird deutlich, dass bei den Vergleichen vor allem eine Untermengen-Relation besteht. Es lässt darauf schließen, dass BMS im Vergleich zu den anderen (insbesondere zu Dundee) Filter nutzt, die spezifizierter eingesetzt werden.

Dundee (siehe Abbildung 10)

Durch die vorgehende Prüfung auf Redundanz wurde bei dem Filter Dundee ein redundantes Muster festgestellt. Bei Dundee lässt sich das Gegenteil zu BMS feststellen. So sind die Filter aus diesem Set allgemeiner gehalten. In Anbetracht der Anzahl der SMARTS-Ausdrücke wurde eine große Menge SMARTS-Ausdrücken gefunden, für die eine Obermengen-Relation festgestellt werden konnte.

Glaxo (siehe Abbildung 10)

Bei den Vergleichen mit dem Set von Glaxo wurden über 100 *Mappings* gefunden, die eine Gleichheit darstellen. Da der Glaxo-Filter mit 54 brauchbaren SMARTS-Ausdrücken eine relativ geringe Größe hat, deutet die große Anzahl der *Mappings* darauf hin, dass dieser Filter SMARTS-Ausdrücke benutzt, die in ähnlicher Form in den meisten anderen Filtern vorkommt.

Inpharmatica (siehe Abbildung 10)

Für Inpharmatica wurden beinahe ausgewogen Ober- und Untermengen-*Mappings gefunden*. Besonders auffällig ist die hohe Anzahl von Ziel-SMARTS, für die ein *Mapping* der Schnitt-Relation gefunden wurde. Zurückzuführen lässt sich dieses Verhalten darauf, dass die SMARTS-Ausdrücke, die der Filter Inpharmatica nutzt, häufig Teilstrukturen besitzen die allgemeiner sind, sowie welche die spezifischer sind.

LINT (siehe Abbildung 11)

Anhand der Ergebnisse der Vergleiche mit dem SMARTS-Filter LINT kann davon ausgegangen werden, dass die genutzten SMARTS-Ausdrücke größtenteils allgemeiner gehalten sind. Da der SMARTS-Filter nur 48 SMARTS-Ausdrücke nutzt und somit der kleinste Filter ist, konnte mit diesem Ergebnis gerechnet werden.

MLSMR (siehe Abbildung 11)

Ähnlich zu dem SMARTS-Filter Inpharmatica wurde auch bei dem Filter MLSMR eine Ausgewogenheit von gefundenen *Mappings* zwischen Unter- und Obermenge festgestellt. Zudem wurde zu über 100 Ziel-SMARTS eine Gleichheit-Relation festgestellt. MLSMR scheint damit einen SMARTS-Filter zu nutzen, der einen guten Repräsentanten der anderen Filter darstellen könnte.

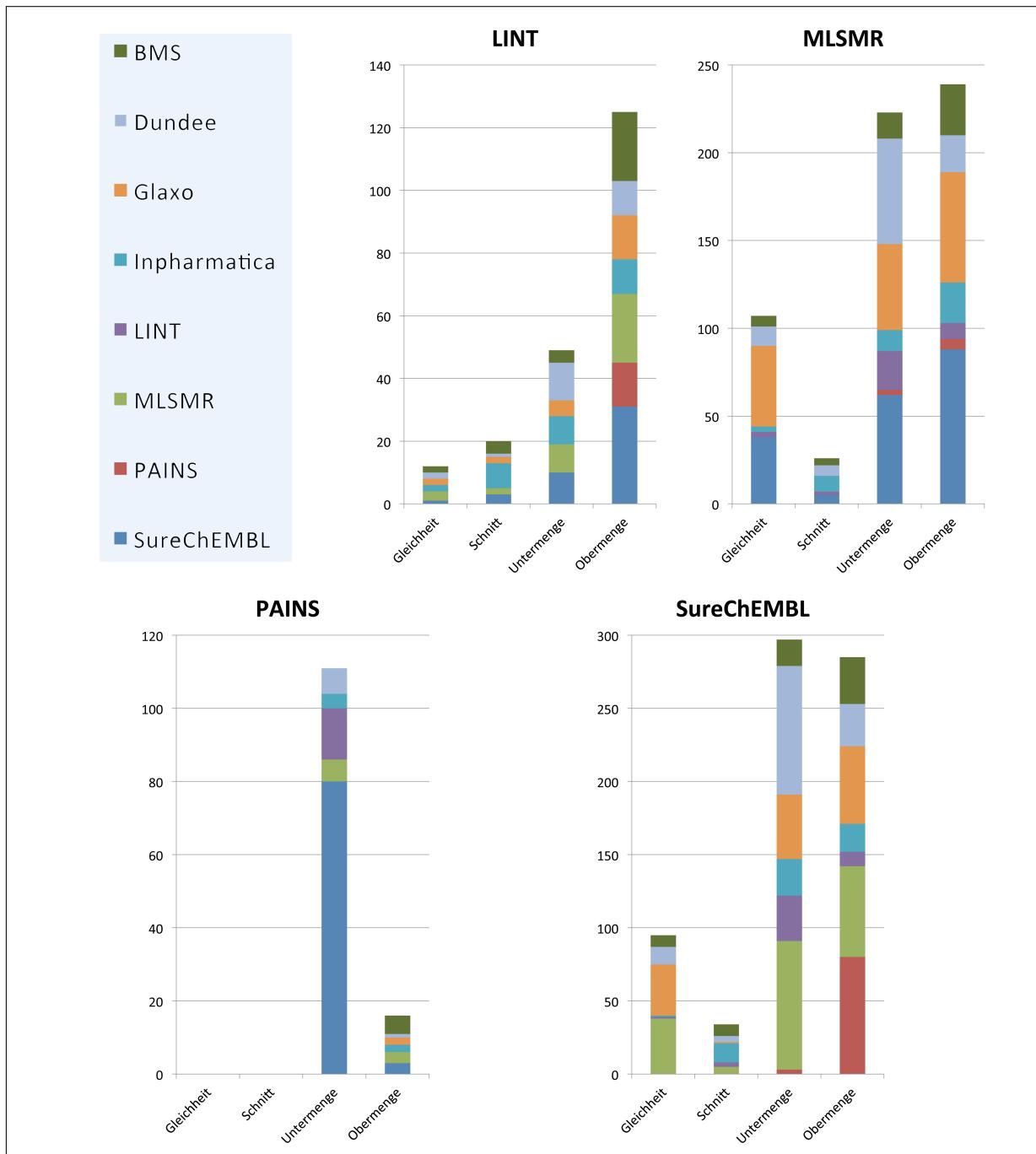


Abbildung 11: Zweiter Teil der Experimente: Dargestellt ist für wie viele Ziel-SMARTS ein *Mapping* zu den Anfrage-SMARTS der einzelnen SMARTS-Filter (LINT, MLSMR, PAINS, SureChEMBL) gefunden wurde. Die *Mappings* sind farbig markiert, je nachdem aus welchen anderen Filtern sie stammen. Zu beachten sind die unterschiedlichen Skalierungen (vgl. Tabelle 12).

PAINS (siehe Abbildung 11)

Die auffälligsten Ergebnisse ergaben die Vergleiche mit dem SMARTS-Filter PAINS. Es wurden weder Gleichheits-*Mappings* noch Schnitt-*Mappings* festgestellt, obwohl der Filter mehr als 400 SMARTS-Ausdrücke nutzt. Auch die geringe Anzahl der Ziel-SMARTS für die ein *Mapping* festgestellt wurde weist darauf hin, dass der SMARTS-Filter PAINS nur

im geringen Maße mit den anderen vergleichbar ist. Es wurden hauptsächlich Unter-*mengen-Mappings* gefunden, was darauf schließen lässt, dass die SMARTS-Ausdrücke eher spezifisch sind. Diese Spezifizierung ist durch eine Konvertierung von SLN-Beschreibungen (*SYBYL line notation* [24]) in SMARTS-Ausdrücke entstanden. Dabei wurden die Wasserstoffatome, die in SMARTS-Ausdrücken, als Eigenschaft behandelt werden als Atome überführt. [26]

SureChEMBL (siehe Abbildung 11)

Es wurde bei dem vorgehenden Vergleich auf redundante ein gleicher SMARTS-Ausdruck in der Liste festgestellt. SureChEMBL nutzt einen Filter, der ausgewogen Unter- und Obermengen-*Mappings* aufweist. Dabei ist dieser SMARTS-Filter derjenige der dem Filter PAINS am ähnlichsten ist. SureChEMBL nutzt mit 148 SMARTS-Ausdrücken einen eher großen Filter, der dennoch in ungefähr gleichen Teilen eine Untermenge sowie eine Obermenge der anderen Filter darstellt. Dadurch lässt sich sagen, dass SureChEMBL einen SMARTS-Filter nutzt, der ein breites Spektrum von SMARTS-Ausdrücken nutzt.

Zusammenfassung

Abschließend lässt sich sagen, dass mit Hilfe des entwickelten Verfahrens Rückschlüsse auf die Art der genutzten Filter gezogen werden können. So weisen die Filter Glaxo, MLSMR und SureChEMBL eine große Ähnlichkeit zueinander auf. Bei anderen Sets wie LINT und Dundee wurde vor allem die Untermenge als häufigste Relation festgestellt. Dies weist darauf hin, dass diese Sets Moleküle im Vergleich mit anderen grob herausfiltern. BMS und Inpharmatica hingegen für die vor allem *Mappings* gefunden wurden, die zu einer Obermenge gehören, weisen darauf hin, dass Moleküleigenschaften im Vergleich zu anderen Sets spezifizierter beschrieben sind und somit weniger Moleküle herausgefiltert werden. PAINS lässt sich kaum mit den anderen Filtern vergleichen, was auf die Konvertierung von SLNs zur SMARTS zurückzuführen ist. Filter, die ausgewogen Unter- und Obermengen-*Mappings* zu anderen Filtern besitzen, scheinen gute Repräsentanten der Filter zu sein. Festgestellt wurde, dass Filter mit einer großen Menge an SMARTS-Ausdrücken häufig auch spezifiziertere SMARTS-Ausdrücke nutzen. Mit Ausnahme des Filters PAINS weisen alle Filter eine hohe Ähnlichkeit zueinander auf. Abbildung 12 zeigt für alle Filter (außer PAINS) ein Muster, welches in einer ähnlichen Form in sämtlichen Filtern zu finden ist und verdeutlicht dennoch die Unterschiede der Filter.

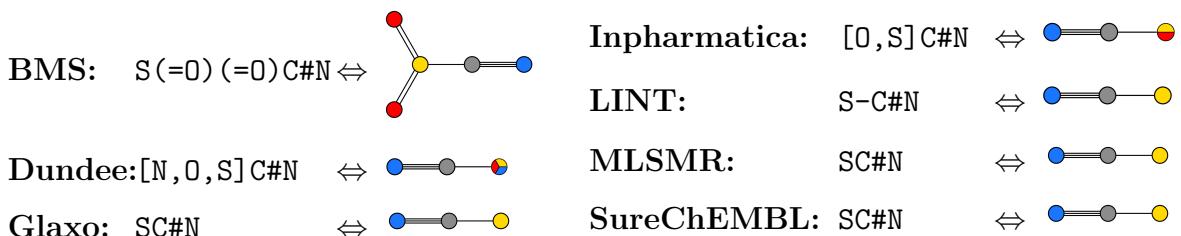


Abbildung 12: Beispiel eines Musters, welches in allgemeiner oder spezifizierten Form in den Filtern BMS, Dundee, Glaxo, Inpharmatica, LINT, MLSMR und SureChEMBL vorkommt. Die Graphiken (generiert mit SMARTSviewer) stellen die beschriebenen Moleküle der SMARTS-Ausdrücke schematisch dar. Rote Kugeln stehen dabei für Sauerstoffatome, Gelbe für Schwefel, Blaue für Stickstoff und Graue für Kohlenstoff. Eine mehrfarbige Kugel steht für die logischen Alternativen, die dieses Atom besitzt.

6 Fazit

Das in dieser Arbeit vorgestellte und implementierte Verfahren ermöglicht eine Lösung des in Kapitel 1.1 beschriebene Problems. Mit dem entwickelten Verfahren ist es möglich chemische Muster zu vergleichen.

Die bereits bestehenden Konzepte [11] konnten im Rahmen dieser Arbeit erweitert und verbessert werden. So wurde die Untersuchung zweier Muster auf eine Schnitt-Relation neu entwickelt und implementiert. Das SMARTS-Atomprimitiv $h< n>$ wurde mit der expliziten Beschreibung ($H< n>$) zusammengeführt, sodass kein zusätzlicher Aufwand im Verfahren nötig ist und das Atomprimitiv dennoch verarbeitet werden kann.

Die Experimente (vgl. Kapitel 5), die mit Hilfe des entwickelten Verfahrens durchgeführt wurden, zeigen eine mögliche Anwendung der implementierten Methode. Die Ergebnisse ermöglichen es Rückschlüsse auf die Art und Spezifizierung der SMARTS-Ausdrücke zu ziehen.

6.1 Ungelöste Probleme

Einige SMARTS-Atomprimitive können von dem vorgestellten Verfahren nicht behandelt werden. Dies verhindert, dass alle SMARTS-Ausdrücke auf Teilmengen-Relationen überprüft werden können.

Momentan werden bei der Übergabe an das implementierte Verfahren die SMARTS-Ausdrücke überprüft und gegebenenfalls aussortiert. Aussortiert werden SMARTS-Ausdrücke, die eine der folgenden Eigenschaften beinhalten:

@@ oder @ : Chiralität an tetraedrischen Kohlenstoffen

@<c><n> : Chiralitätsklasse $< c >$, Chiralität $< n >$

@<c><n>? : Chiralitätsklasse $< c >$, Chiralität $< n >$ oder nicht spezifiziert

$\$(\text{SMARTSExp})$: Rekursive Ausdrücke

Die drei Eigenschaften betreffend der Chiralität sind deshalb problematisch, da sie abhängig von den impliziten Eigenschaften sind, die aus der Molekülgraphenstruktur resultieren.

Ein besonderer Problemfall betrifft die rekursiven Ausdrücke (siehe Kapitel 1.2.2) der SMARTS-Sprache. Die Einbeziehung einer Umgebung für ein Atom oder eine Bindung ist für das hier vorgestellte Konzept der Fingerprints als Grundlage der Vergleiche nicht möglich. Die Eigenschaft ist für SMARTS-Ausdrücke extrem relevant, da sie es ermöglicht mit einer Definition der Umgebung ein chemisches Muster genauer und effizient zu beschreiben.

Es besteht das Problem, dass ein Abgleich der Atommasse durch die Fingerprintgenerierung nicht möglich ist. Die Atommasse-Eigenschaft, die in den Valenzzuständen beinhaltet ist, beschreibt lediglich die Masse der Elemente in ihrer natürlichen Form. Gewichtsangaben von Isotopen liegen in dem genutzten Konzept der Valenzzustände nicht vor.

Ein Problem, welches unabhängig von der SMARTS-Sprache ist, betrifft die Schnitt-Relation. Wie in Formel (2) beschrieben gilt entweder für Knoten oder Kanten, dass es eine Unter- und eine Obermengenrelation in einem möglichen *Mapping* gibt. Formel (3) beschreibt den Fall bei dem kein *Mapping* erkannt wird, da die Schnitt-Relation Knoten und Kanten der Molekülgraphen unabhängig voneinander beschreibt. Beispiel 6.1.1 zeigt den in (3) beschriebenen Fall.

$$\begin{aligned}
 & \left(\forall(v_Q, v_T) \in M_{QT} : (LA_{v_Q} =, \cap, \subset, \supset LA_{v_T}) \right. \\
 & \quad \wedge \forall(e_Q, e_T) \in M_{QT} : (LA_{e_Q} =, \cap, \subset, \supset LA_{e_T}) \Big) \\
 & \wedge \left[\left((\exists(v'_Q, v'_T) \in M_{QT} \setminus \{(v_Q, v_T)\} : (LA_{v'_Q} \subset LA_{v'_T})) \right. \right. \\
 & \quad \vee (\exists(e'_Q, e'_T) \in M_{QT} \setminus \{(e_Q, e_T)\} : (LA_{e'_Q} \subset LA_{e'_T})) \Big) \\
 & \quad \wedge \left((\exists(v''_Q, v''_T) \in M_{QT} \setminus \{(e_Q, e_T)\} : ((v''_Q, v''_T) \neq (v'_Q, v'_T) \wedge LA_{v''_Q} \supset LA_{v''_T})) \right. \\
 & \quad \left. \left. \vee (\exists(e''_Q, e''_T) \in M_{QT} \setminus \{(e_Q, e_T)\} : ((e''_Q, e''_T) \neq (e'_Q, e'_T) \wedge LA_{e''_Q} \supset LA_{e''_T})) \right) \right] \tag{3}
 \end{aligned}$$

Beispiel 6.1.1.

Anfrage-SMARTS	$[C1, I]$	=	0
Teilmengen-Relation	\supset	\subset	=
Ziel-SMARTS	$[C1]$	\sim	0

Ein interessantes Problem, welches durch die Auswertung der Experimente (vgl. Kapitel 5) aufgedeckt wurde, betrifft Ringschlüsse. Untersucht wurden die SMARTS-Ausdrücke auf Gleichheit und Schnitt-Relation. Beispiel 6.1.2 zeigt das Ergebnis des Programms *SmartsSmarts*. Offensichtlich wurde eine Schnitt-Relation festgestellt, obwohl Gleichheit festgestellt werden sollte.

Beispiel 6.1.2.

Schnitt:	Anfrage-SMARTS	$[C:1]1[C:2][N,S,0:3]1$
	Ziel-SMARTS	$[C:1]1[0,S,N:3][C:2]1$
	Anfrage-SMARTS	$[C:1]1[C:2][N,S,0:3]1$
	Ziel-SMARTS	$[C:2]1[0,S,N:3][C:1]1$

7 Weiterführende Arbeit

Das Problem dessen Lösung am interessantesten ist, ist die der Einbeziehung der rekursiven Ausdrücke. Ein Ansatz wäre es, falls eine Beschreibung eines Anfrage-SMARTS einen rekursiven Ausdruck besitzt, diese als Eigenschaft zu vergleichen. Eine Überprüfung, ob es im Ziel-SMARTS eine, der Teilmengen-Relation entsprechende, rekursive Beschreibung gibt, würde diesen Vergleich realisieren. Im ersten Schritt würde ohne Einbezug der rekursiven Ausdrücke ein *Mapping* gesucht werden. Falls es rekursive Beschreibungen von Atomen oder Bindungen gibt, würde im nächsten Schritt überprüft werden, ob das entsprechende Atom oder die entsprechende Bindung des Ziel-SMARTS einen rekursiven Ausdruck besitzt, der der Teilmengen-Relation entspricht. Dabei wird für Gleichheit und Schnitt-Relation das bestehende Verfahren genutzt. Die zu vergleichenden rekursiven Ausdrücke werden an das Verfahren übergeben, und geprüft, ob es ein *Mapping* gibt. Bei Unter- und Obermenge wird, falls es für beide Partner des *Mappings* einen rekursiven Ausdruck gibt, analog vorgegangen. Falls es nur für einen *Mapping*-Partner eine rekursive Beschreibung gibt, gilt:

Untermenge: Der *Mapping*-Partner des Anfrage-SMARTS hat eine rekursive Beschreibung.

Obermenge: Der *Mapping*-Partner des Ziel-SMARTS hat eine rekursive Beschreibung.

Für die Probleme bezüglich der nicht beachteten SMARTS-Primitive müsste ein Konzept entwickelt werden, welches die zu beachtenden Eigenschaften behandeln kann. Ein Ansatz wäre für die Chiralität die Information als kurze String-Repräsentation zu speichern und diese zu vergleichen. Ähnlich zum Ansatz für die rekursiven Ausdrücke, kann so gegebenenfalls die Eigenschaft abgeglichen werden.

Das Fehlverhalten welches für Beispiel 6.1.2 festgestellt wurde, konnte bereits eingegrenzt werden. Es wurde keine Gleichheit festgestellt, da für die Knoten $[N, S, 0]$ und $[0, S, N]$ keine Gleichheit festgestellt werden konnte. Übergibt man nur diese beiden Knoten an das Programm wird eine Gleichheit festgestellt. Auch wenn die Ringbindung (gekennzeichnet durch das Zahlenpaar „1“) entfernt wird, wird eine Gleichheit der beiden SMARTS-Ausdrücke festgestellt. Das Problem liegt also an den beiden Knoten $[N, S, 0]$ und $[0, S, N]$ in Zusammenhang mit einem Ringschluss. Es wurde weiter festgestellt, dass solange die erste logische Alternative in beiden Knoten gleich ist, das Problem nicht auftritt.

Das Problem tritt also auf wenn es in beiden SMARTS-Ausdrücken einen Knoten gibt, mit den gleichen logischen Alternativen. Diese Knoten müssen in einem Ring enthalten sein. Zudem muss an der ersten Position der Knoten unterschiedliche logische Alternativen stehen. Bis zu diesem Zeitpunkt konnte noch kein Lösungsansatz entwickelt werden.

A Software User Guide

A.1 Command Line Syntax

Command Line Syntax beginnt immer mit dem Command Line Programmnamen. Dem Programmnamen können dann Optionen und Dateinamen folgen. Die Optionen können in beliebiger Reihenfolge, angeführt von einem Minus und getrennt mit einem Leerzeichen angegeben werden.

Das folgende Beispiel zeigt die Command Line Syntax um *SmartsSmarts* mit einer SMARTS Datei auszuführen

- – Richtig: `SmartsSmarts -t /home/targetSMARTS.txt`
- Falsch: `SmartsSmarts -t/home/targetSMARTS.txt`
- Es müssen immer korrekte Dateiendungen mit angegeben werden (z.B. `.txt`)

A.2 Command Line Options

Die folgenden Optionen sind in dem *SmartsSmarts*-Programm beinhaltet:

- `-h` (Hilfe)
- `-s` (Anfrage-SMARTS Input (String))
- `-l` (Anfrage-SMARTS Input (Datei))
- `-t` (Ziel-SMARTS Input)
- `-r` (Teilmengen-Relation)
- `-v` (Validierung (Entwickleroption))

-h (Hilfe)

Wenn der Programmname mit dieser Option aufgerufen wird, wird eine Meldung angezeigt, die alle Optionen, exklusive der Entwickleroptionen, zusammen mit ihren benötigten Parametern, ihren Datentypen sowie ihren Beschreibungen für das Programm auflistet.

Syntax:

```
-h  
--help
```

-s (Anfrage-SMARTS Input (String))

Wenn der Programmname mit dieser Option eingeben wird, ist als weiterer Parameter ein String erforderlich, welcher in Form eines SMARTS vorliegt.

Syntax:

```
-s "querySMARTS"
```

```
--querystring "querySMARTS"
```

-I (Anfrage-SMARTS Input (Datei))

Wenn der Programmname mit dieser Option eingeben wird, ist als weiterer Parameter ein Name einer Datei erforderlich, welche SMARTS-Strings beinhaltet.

Syntax:

```
-l querySMARTS  
--querylist querySMARTS
```

Hinweis:

Die SMARTS Input-Datei darf pro Zeile nur einen SMARTS-String beinhalten.

-t (Ziel-SMARTS Input)

Wenn der Programmname mit dieser Option eingeben wird, ist als weiterer Parameter ein Name einer Datei erforderlich, welche SMARTS-Strings beinhaltet.

Syntax:

```
-t targetSMARTS  
--target targetSMARTS
```

Hinweis:

Die SMARTS Input-Datei darf pro Zeile nur einen SMARTS-String beinhalten.

-r (Teilmengen-Relation)

Wenn der Programmname mit dieser Option eingeben wird, ist als weiterer Parameter ein String erforderlich, welcher in Form eines vier-elementigen Bitstrings für Schnitt-Relation vorliegt.

Syntax:

```
-r "Bitstring"  
--relationship "Bitstring"
```

Hinweis:

Der Bitstring muss folgenden Syntax einhalten:

- Der Bitstring besteht aus 4 Zeichen
- Erlaubte Zeichen sind „0“ und „1“
- Mindestens ein Zeichen muss eine „1“ sein
- Dabei sind die Teilmengen-Relationen folgenden Bits zugeordnet:
 - 1. Bit: Gleichheit (1000)
 - 2. Bit: Schnitt (0100)
 - 3. Bit: Untermenge (0010)
 - 4. Bit: Obermenge (0001)

-v (Validierung (Entwickleroption))

Wenn der Programmname mit dieser Option eingeben wird, ist als weiterer Parameter ein Dateiname erforderlich, welche Molekülbeschreibungen beinhaltet. Mögliche Dateiformate sind: .mol2, .pdb, .sdf, .smi, .smiles

Die Optionen **-s** (oder **-l**), **-t** und **-r** sind bei dieser Option erforderlich. Es handelt sich bei dieser Option um eine Entwickleroption mit der die Validierung des Programms ausgeführt wird.

Syntax:

```
-v molecules
--validation molecules
```

Beim Aufruf des Programmes müssen die Optionen **-s** (oder **-l**), **-t** und **-r** immer mit angegeben werden.

B Softwarearchitektur

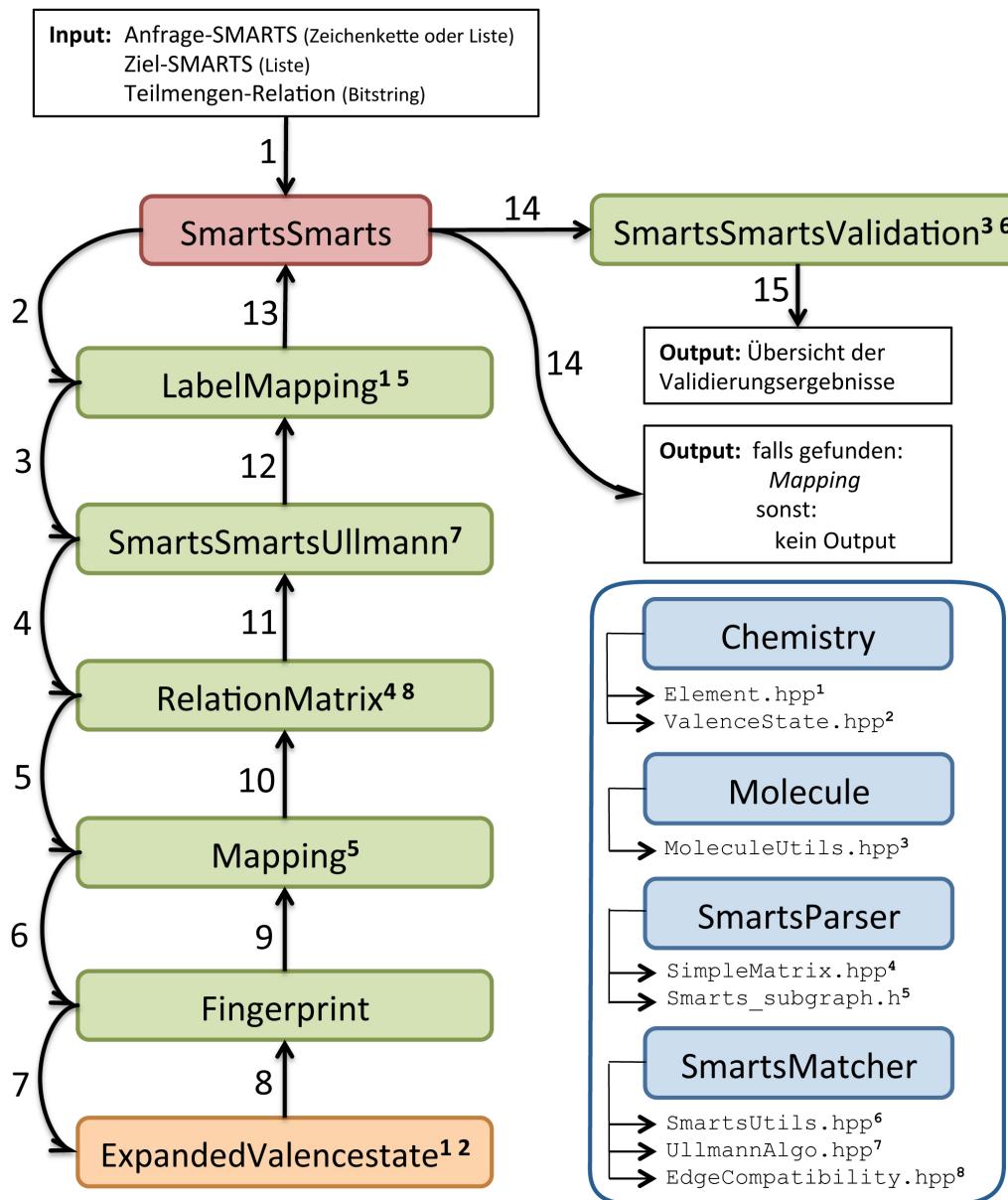


Abbildung 13: Schematische Darstellung der Softwarestruktur des Programms *SmartsSmarts*

In Abbildung 13 ist die Softwarestruktur des Programms *SmartsSmarts* (rot) schematisch dargestellt. Ausgehend von den Eingabedateien (Input) wird durch die nummerierten Pfeile der Ablauf des Programms dargestellt. Es wird gezeigt, welche Module (grün) und Klassen (orange) des Programms sich gegenseitig nutzen. Welche Module aus der NAOMI-Softwarebibliothek (blau) an welchen Stellen des *SmartsSmarts* Programms genutzt werden, ist durch Hochzahlen gekennzeichnet. Im folgenden Kapitel werden die wichtigsten Module und Klassen erläutert.

C Wichtige Module und Klassen

LabelMapping.hpp

```
std::pair<std::string, std::string> addLabels(const std::string &smarts_q,
                                              const std::string &smarts_t,
                                              const std::vector<std::vector<int>>
                                              existingLabels)
```

Diese Funktion versieht anhand von Anfrage-SMARTS, Ziel-SMARTS und einer Liste über die bestehenden *Label*, beide SMARTS-Ausdrücke mit *Labels* zur eindeutigen Kennzeichnung und gibt diese zurück.

```
std::pair<std::string, std::string> recalculateLabels(const std::string
                                                       &smarts_q, const std::string &smarts_t,
                                                       const std::vector<std::vector<int>>&existingLabels,
                                                       const Base::SimpleMatrix<int>&solutions)
```

Diese Funktion bekommt Anfrage-SMARTS, Ziel-SMARTS, eine Liste über die bestehenden *Label* und eine Permutationsmatrix, die einen Isomorphismus repräsentiert übergeben. Sie setzt die *Label* anhand der gefundenen Lösung neu, und gibt Anfrage- und Ziel-SMARTS zurück.

SmartsSmartsUllmann.hpp

```
std::vector<Base::SimpleMatrix<int>> getUllmannSolutions
                                         (const std::string &smarts_q,
                                          const std::string &smarts_t,
                                          const std::string &relation)
```

Diese Funktion berechnet mit einem Anfrage-SMARTS, Ziel-SMARTS und einer Teilmengen-Relation in Form eines Bitstrings ein eindeutiges *Mapping* und gibt dieses als Permutationsmatrix zurück.

RelationMatrix.hpp

```
Base::SimpleMatrix<int> initialiseNodeMatrix(const std::string &smarts_q,
                                                const std::string &smarts_t,
                                                const std::string &relation)
```

Anhand von Anfrage-SMARTS, Ziel-SMARTS und Teilmengen-Relation in Form eines Bitstrings initialisiert diese Funktion eine Relations-Matrix. Ausgegeben wird die Relations-

Matrix, die die Verträglichkeit der Knoten der beiden SMARTS-Ausdrücke repräsentiert.

```
SmartsMatcher::EdgeCompatibility initialiseEdgeCompatibility
    (const std::string &smarts_q,
     const std::string &smarts_t,
     const std::string &relation)
```

Anhand von Anfrage-SMARTS, Ziel-SMARTS und Teilmengen-Relation in Form eines Bitstrings initialisiert diese Funktion ein `SmartsMatcher::EdgeCompatibility` Ausgegeben wird das `EdgeCompatibility`-Objekt, welches die Verträglichkeit der Kanten der beiden SMARTS-Ausdrücke repräsentiert.

Mapping.hpp

```
std::string isNodeMappable(const sg_node *node_q, const sg_node *node_t,
                           const std::string &relation)
```

Anfrage-Knoten, Ziel-Knoten und Teilmengen-Relation in Form eines Bitstrings werden an diese Funktion übergeben, die anschließend prüft welche Teilmengen-Relation bei zwei Knoten vorliegt und gibt diese als Bitstring zurück.

```
std::string isEdgeMappable(const sg_edge *edge_q, const sg_edge *edge_t,
                           const std::string &relation)
```

Anfrage-Kante, Ziel-Kante und Teilmengen-Relation in Form eines Bitstrings werden dieser Funktion übergeben, die prüft welche Teilmengen-Relation zwischen zwei Kanten vorliegt und gibt diese als Bitstring zurück

Fingerprint.hpp

```
boost::dynamic_bitset<> takePropertyFingerprint(const sg_node *node)
```

Diese Funktion generiert einen Fingerprint für einen gegebenen Knoten und gibt diesen in Form eines Bits aus.

```
boost::dynamic_bitset<> takePropertyFingerprint(const sg_edge *edge)
```

Diese Funktion generiert einen Fingerprint für eine gegebene Kante und gibt diesen in Form eines Bits aus.

ExpendedValencestate.hpp

```
std::vector<ExpandedValenceState> createAllValenceStates()
```

Erstellt eine Liste über alle möglichen Valenzzustände von Atom-SMARTS-Ausdrücken und gibt diese aus.

SmartsSmartsValidation.hpp

```
valPair validateSolution(valPair validationOutput,
                        MolLib::MutableMolPtrVector valMolecules,
                        std::vector<Base::SimpleMatrix<int>> solutions,
                        std::string relation, std::string smarts_q,
                        std::string smarts_t)
```

Fügt einem gegebenen ValidierungsOutput anhand einer Liste von Molekülen, einer Liste über gefundene *Mappings*, einem Anfrage- und einem Ziel-SMARTS, neue Ergebnisse der Validierung zu.

D Tests

Zur Überprüfung der Korrektheit des implementierten Verfahrens *SmartsSmarts* wurden im ersten Schritt der Evaluation die Hauptfunktionen, die das Verfahren nutzt, auf korrekte Ergebnisse überprüft. Welche Testfunktionen genutzt wurden, wird in diesem Kapitel beschrieben.

fingerprint:

Die Testfunktion prüft, ob bei der Übergabe von einem Knoten eines SMARTS-Ausdrucks ein Fingerprint generiert wurde, der die richtigen Bits gesetzt hat.

mapping:

Überprüft die Korrektheit der Funktion bei der anhand von zwei Fingerprints und einer Teilmengen-Relation ermittelt wird, ob die gewünschte Teilmengen-Relation gefunden wurde.

nodeRelationMatrix:

Initialisiert eine Relations-Matrix und überprüft, ob diese korrekt erstellt wurde.

smartsSmartsUllmann:

Testfunktion die ermittelt, ob bei der Übergabe von zwei SMARTS-Ausdrücken und einer Teilmengen-Relation das erwartete *Mapping* in Form einer Permutationsmatrix ausgegeben wird.

Durch die Testfunktionen konnten Fehler in der Fingerprintgenerierung und der Isomorphismusanalyse durch den Algorithmus nach Ullmann identifiziert und behoben werden.

E APIs

Um das *SmartsSmarts*-Programm einzubinden müssen folgende Funktionen aufgerufen werden:

Prototyp:

```
std :: vector<int> extractExistingLabels( const std :: string &smarts );
```

Header:

„LabelMapping.hpp“

Beschreibung:

Gewinnt aus einem gegebenen SMARTS-Ausdruck bestehende *Label* und speichert diese in einem `std::vector`. Dieser Aufruf muss für Anfrage- und Ziel-SMARTS ausgeführt werden und in einem `std::vector<std::vector<int>>` gespeichert werden.

Prototyp:

```
std :: pair<std :: string , std :: string > addLabels( const std :: string &
    querySmarts ,
    const std :: string &targetSmarts ,
    const std :: vector<std :: vector<int>>
    &existingLabels );
```

Header:

„LabelMapping.hpp“

Beschreibung:

Fügt an alle Atome eines Anfrage- und eines Ziel-SMARTS, falls noch nicht vorhanden, *Label* hinzu, so dass jedes Atom eine eindeutige Kennzeichnung hat.

Prototyp:

```
std :: vector<Base :: SimpleMatrix<int>> getUllmannSolutions( const std :: string &querySmarts ,
    const std :: string &targetSmarts ,
    const std :: string &relation );
```

Header:

„SmartsSmartsUllmann.hpp“

Beschreibung:

Prüft für einen Anfrage-, Ziel-SMARTS und eine Teilmengen-Relation, ob es einen oder mehrere Isomorphismen gibt und gibt diesen in Form eines `std::vector<Base::SimpleMatrix<int>>` zurück.

Prototyp:

```
std :: pair<std :: string , std :: string > recalculateLabels( const std :: string &
    querySmarts ,
    const std :: string &targetSmarts ,
```

```
const std::vector<std::vector<int>>
&existingLabels ,
const Base::SimpleMatrix<int>
&solutions );
```

Header:

„LabelMapping.hpp“

Beschreibung:

Kenntzeichnet einen gefundenen Isomorphismus durch Neusetzung der *Label* und gibt Anfrage- und Ziel-SMARTS als std::pair<std::string, std::string> zurück.

F Genutzte Datensätze

Alle genutzten Datensätze finden sich auf dem Datenträger (Seite X)

ZINC-6k-Molekülliste

siehe zinc_rand_6k.smi

SMARTS Datensatz (ZBH)

siehe all_smarts_zbh.txt

SMARTS Filter (Experimente)

Bristol-Myers Squibb HTS Deck Filters

siehe ChEMBL_BMS.txt

University of Dundee NTD Screening Library Filters

siehe ChEMBL_Dundee.txt

Glaxo Wellcome Hard Filters

siehe ChEMBL_Glaxo.txt

Inpharmatica Unwanted Fragments

siehe ChEMBL_Inpharmatica.txt

Pfizer LINT filters

siehe ChEMBL_LINT.txt

NIH MLSMR Excluded Functionality Filters

siehe ChEMBL_MLSMR.txt

Pan Assay Interference Compounds Filters

siehe ChEMBL_PAINS.txt

SureChEMBL Data

siehe ChEMBL_SureChEMBL.txt

G Ergebnisse der Experimente

Ergebnisse, die durch das Experiment (vgl. Kapitel 5) der acht Filter entstanden sind. 28 Dateien mit Ergebnissen nach Ausführung des *SmartsSmarts*-Programms. Siehe Ordner **Ergebnisse_Experimente** auf Datenträger (Seite X).

H Implementation

Siehe Ordner **Implementation** auf Datenträger (Seite X)

I Literatur

- [1] J. Gasteiger and T. Engel, editors. *Chemoinformatics: A Textbook*. Wiley-VCH, 2003.
- [2] Roger Perkins, Hong Fang, Weida Tong, and William J. Welsh. Quantitative structure-activity relationship methods: Perspectives on drug discovery and toxicology. *Environmental Toxicology and Chemistry*, 22(8):1666–1679, 2003.
- [3] B. A. Bunin, B. Siesel, G. A. Morales, and J. Bajorath, editors. *Chemoinformatics: theory, practice, and products*. Springer, 2007.
- [4] Andrew R. Leach and Valerie J. Gillet. *An Introduction to Chemoinformatics*. Springer Publishing Company, Incorporated, 2007.
- [5] Milan Randic. Characterization of molecular branching. *Journal of the American Chemical Society*, 97(23):6609–6615, 1975.
- [6] Shihyen Chen, Bin Ma, and Kaizhong Zhang. On the similarity metric and the distance metric. *Theor. Comput. Sci.*, 410(24-25):2365–2376, 2009.
- [7] Inc. Daylight Chemical Information Systems. Daylight smarts. http://daylight.com/dayhtml_tutorials/languages/smarts/. Zugriff: 20.08.2015.
- [8] M. Rarey. Grundlagen der chemieinformatik. Vorlesung, Wintersemester 2014.
- [9] H.J. Böckenhauer and J. Hromkovic. *Formale Sprachen: Endliche Automaten, Grammatiken, lexikalische und syntaktische Analyse*. Springer, 2013.
- [10] Adrian Horia Dediu, Enrico Formenti, Carlos Martín-Vide, and Bianca Truthe, editors. *Language and Automata Theory and Applications - 9th International Conference, LATA 2015, Nice, France, March 2-6, 2015, Proceedings*, volume 8977 of *Lecture Notes in Computer Science*. Springer, 2015.
- [11] Andriy Mashychev. Eine computergestützte Methode zum Vergleich chemischer Muster am Beispiel von SMARTS. Masterarbeit, Universität Hamburg, 2011.
- [12] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988.
- [13] Inc. Daylight Chemical Information Systems. Daylight smiles. http://daylight.com/dayhtml_tutorials/languages/smiles/. Zugriff: 20.08.2015.
- [14] Dieter Jungnickel. *Graphs, Networks and Algorithms*. Springer, 3rd edition, 2007.
- [15] S. Urbaczek, A. Kolodzik, J.R. Fischer, T. Lippert, S. Heuser, I. Groth, T. Schulz-Gasch, and M. Rarey. Naomi: On the almost trivial task of reading molecules from different file formats. *J.Chem.Inf.Model*, 51:3199–3027, 2011.
- [16] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976.

- [17] ZBH Center for Bioinformatics Hamburg. SMARTSviewer. <http://smartsviewer.de/>. Zugriff: 20.10.2015.
- [18] San Francisco University of California. Zinc. <http://zinc.docking.org/>. Zugriff: 19.08.2015.
- [19] ZBH Center for Bioinformatics Hamburg. Smarts dataset. <http://www.zbh.uni-hamburg.de/forschung/arbeitsgruppe-algorithmisches-molekulares-design/datasets.html>. Zugriff: 01.08.2015.
- [20] ChemSpider. Levothyroxine. <http://www.chemspider.com/Chemical-Structure.5614.html>. Zugriff: 14.10.2015.
- [21] EMBL European Molecular Biology Laboratory. ChEMBL. <https://www.ebi.ac.uk/chembl/>. Zugriff: 14.10.2015.
- [22] EMBL European Molecular Biology Laboratory. ChEMBL 20 Release. <http://chembl.blogspot.de/2015/02/chembl-20-released.html>. Zugriff: 14.10.2015.
- [23] EMBL European Molecular Biology Laboratory and C. Laggner. Structural Alerts - ChEMBL 20. persönliche Korrespondenz mit C. Laggner, September 2015.
- [24] R. Webster Homer, Jon Swanson, Robert J. Jilek, Tad Hurst, and Robert D. Clark. SYBYL line notation (SLN): A single notation to represent chemical structures, queries, reactions, and virtual libraries. *Journal of Chemical Information and Modeling*, 48(12):2294–2307, 2008.
- [25] Karen Schomburg, Hans-Christian Ehrlich, Katrin Stierand, and Matthias Rarey. From structure diagrams to visual chemical patterns. *Journal of Chemical Information and Modeling*, 50(9):1529–1535, 2010. PMID: 20795706.
- [26] Simon Saubern, Rajarshi Guha, and Jonathan B. Baell. KNIME Workflow to Assess PAINS Filters in SMARTS Format. Comparison of RDKit and Indigo Cheminformatics Libraries. *Molecular Informatics*, 30(10):847–850, 2011.

Erklärung

Ich versichere, dass ich die Bachelorarbeit im Studiengang Computing in Science SP Biochemie selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Hamburg, den _____ Unterschrift: _____

Ich bin mit einer Einstellung der Bachelorarbeit in den Bestand der Bibliothek des Departments Informatik einverstanden.

Hamburg, den _____ Unterschrift: _____