SMILES - 3D

Das weiterentwickelte Konzept zur

Codierung dreidimensionaler Molekülstruktur

in einer linearen Notation

und die programmtechnische Umsetzung

in der Software WinSmiles – 3D

Dissertation

zur Erlangung des

Doktorgrades der Naturwissenschaften

der Fakultät für Chemie

an der Universität Bielefeld

vorgelegt von

Andreas Bruder

Bielefeld, im Dezember 2003

Die vorliegende Arbeit wurde in der Zeit von Januar 1999 bis Dezember 2003 an der Fakultät für Chemie an der Universität Bielefeld unter Anleitung von Herrn Prof. J. Hinze, PhD angefertigt.

Meinem akademischen Lehrer Herrn Prof. J. Hinze, PhD danke ich herzlich für die freundliche Aufnahme in seinen Arbeitskreis und die Überlassung des Themas, sowie sein Interesse und seine wohlwollende Unterstützung beim Fortgang der vorliegenden Arbeit.

Meinem akademischen Lehrer Herrn PD Dr. D. Andrae danke ich für sein Interesse und seine stete Bereitschaft zu anregenden und informativen wissenschaftlichen Gesprächen.

Herrn U. Welz danke ich für die Überlassung zahlreicher Quellcodedateien. Des weiteren danke ich Herrn Dipl. – Inform. T. Tönsing für seine Unterstützung bei der Gestaltung der grafischen Benutzeroberfläche der entwickelten Software.

Ferner danke ich allen Mitgliedern der Arbeitsgruppe theoretische Chemie, die meine Arbeit durch ihre stete Hilfsbereitschaft unterstützt und für ein gutes Arbeitsklima gesorgt haben.

Für meine liebe Frau Claudia
und für meinen kleinen Sohn Felix

Inhaltsverzeichnis

1. Einleitung

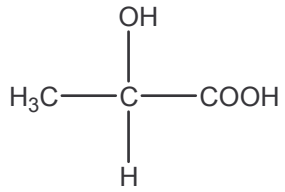
In den letzten Jahrzehnten hat sich der Einsatzbereich von Computern in der Chemie deutlich erweitert und die Entwicklung von Software zur Lösung von unterschiedlichen wissenschaftlichen Fragestellungen zu einem interessanten Forschungsbereich in der theoretischen Chemie entwickelt.

Die dreidimensionale Struktur von Molekülen hat einen entscheidenden Einfluss auf die chemischen, physikalischen und die biologischen Eigenschaften einer chemischen Verbindung. Durch technische Verfahren wie der Röntgenstrukturanalyse, der Mikrowellenspektroskopie oder der Kernresonanzspektroskopie kann die dreidimensionale Struktur von stabilen chemischen Verbindungen ermittelt werden. Allerdings ist die dreidimensionale Molekülstruktur nur für einen geringen Teil der bekannten chemischen Verbindungen experimentell bestimmt und darüber hinaus für einen Großteil der chemischen Verbindungen aus technischen Gründen nicht bestimmbar.[1] Die Bestimmung vieler Eigenschaften der Moleküle, sowie deren eindeutige Spezifizierung, verlangt die Kenntnis der dreidimensionalen Struktur. Es ist daher sinnvoll, die dreidimensionale Struktur von Molekülen zu simulieren und diese durch die Verwendung von geeigneter Software mit Hilfe von Computern zu berechnen.

Durch die Verwendung von quantenchemischen[2-4] oder molekularmechanischen[5] Methoden können nahezu exakte dreidimensionale Molekülstrukturen berechnet werden. Diese Berechnungsmethoden benötigen allerdings zum einen Anfangsinformationen über die genäherte (vermutete) geometrische Molekülstruktur, und zum anderen eine große Rechenkapazität der Hardware.

Zur Beschreibung der Struktur von chemischen Verbindungen ist bereits eine Vielzahl an Systematiken entwickelt worden. Eine weit verbreitete Methode zur Beschreibung von chemischen Verbindungen ist die Codierung der molekularen Informationen durch eine lineare Notation. Das lineare Notierungssystem SMILES (**S**implified **M**olekular **I**nput **L**ine **E**ntry **S**ystem) ist das derzeit bekannteste und weit verbreiteteste System zur Codierung der zweidimensionalen Molekülstruktur in einer

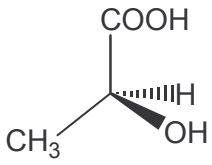
linearen Zeichenfolge.[6] Dieses Verfahren ist durch die Verwendung der Atomsymbole zur Darstellung eines Atoms und einfacher Regeln leicht anwendbar. Exemplarisch ist in Abbildung 1-1 die Codierung der zweidimensionalen Struktur der Milchsäure durch die lineare SMILES Notation gezeigt.



SMILES: CC(O)C=OO

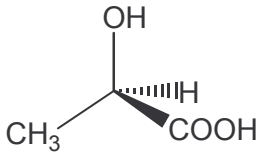
Abbildung 1-1: SMILES Notation von 2-Hydroxypropansäure (Milchsäure)

Das SMILES Notierungssystem ist durch die Spezifikation der Atomsymbole und der Symbole zur Darstellung von Bindungen von Hinze et al. weiterentwickelt worden.[7] Durch die Integration von strukturbildenden Zellen in das sogenannte Broad SMILES Konzept ist die Codierung der dreidimensionalen Molekülstruktur in einer linearen Zeichenfolge realisiert. Dieses Konzept ist im Rahmen dieser Arbeit modifiziert und zu der linearen SMILES - 3D Notation weiterentwickelt worden. Die Weiterentwicklung der linearen Notation erfolgte mit den Zielen, zum einen durch die SMILES - 3D Notation eine effiziente Codierung der genäherten dreidimensionalen Molekülstruktur in einer äußerst kompakten und leicht verständlichen, übersichtlichen linearen Zeichenfolge zu erreichen; und zum anderen durch ergänzende optionale Spezifikationen der linearen SMILES - 3D Zeichenfolge die Codierung von optimierten dreidimensionalen Molekülstrukturen in einer modifizierten Zeichenfolge zu ermöglichen. In Abbildung 1-2 ist die unterschiedliche Codierung der S-Form und der R-Form der Milchsäure in der linearen SMILES - 3D Notation dargestellt.



SMILES - 3D: CC(C=OO)O

S - Form



SMILES - 3D: CC(O)C=OO

R - Form

Abbildung 1-2: SMILES - 3D Notation der beiden chiralen Formen von 2-Hydroxypropansäure ˆ (Milchsäure)

Im Rahmen dieser Arbeit ist das SMILES - 3D Konzept in der hier weiter entwickelten Software WinSmiles - 3D umgesetzt worden. Das Programm WinSmiles - 3D erzeugt aus dem spezifizierten SMILES - 3D String eine dreidimensionale Molekülstruktur. Diese Struktur wird zum einen innerhalb der Software visualisiert und kann zum anderen als genäherte Anfangsstruktur in Programmen zur Berechnung von exakten Molekülstrukturen verwendet werden. Die Software läuft unter Windows Betriebssystemen auf gängigen Personalcomputern und zeichnet sich durch eine hohe Geschwindigkeit bei der Erzeugung der dreidimensionalen Molekülstruktur aus.

Die dreidimensionale Molekülstruktur kann bei der Visualisierung durch ein dreidimensional wirkendes Abbild des Moleküls dargestellt werden. Durch Rotation oder Zoomen kann das Molekül aus unterschiedlichen Positionen betrachtet werden und ermöglicht so dem Betrachter einen Einblick in die dreidimensionale Molekülstruktur.

Darüber hinaus ist innerhalb dieser Arbeit ein Algorithmus zur Erzeugung der modifizierten linearen SMILES - 3D Notierung aus einer definierten Molekülstruktur entwickelt und in die Software WinSmiles - 3D implementiert worden. Dies erlaubt den Test, ob die Abbildung der dreidimensionalen Molekülstruktur bijektiv zu der linearen SMILES - 3D Notation ist. Leider heißt dies noch nicht, dass die Beziehung zwischen linearer Notation und der Abbildung eineindeutig ist. Für letzteres wären noch weitere Konventionen notwendig, die hier nur angedeutet sind.

2. Codierung von molekularen Strukturen

2.1 Allgemeine Vorbemerkungen

Der Einfluss der dreidimensionalen Struktur eines Moleküls auf das chemische Verhalten, die physikalischen Eigenschaften und zum Teil auf die biologische Aktivität einer chemischen Substanz zeigen die Notwendigkeit, strukturrelevante Informationen mit in der Chemie üblichen Methoden zur Beschreibung von Molekülen zu verbinden. Es sind daher eine Vielzahl an Methoden zur Kennzeichnung einer chemischen Verbindung entwickelt worden, die von der Struktur eines Moleküls abgeleitet sind. Diese strukturbasierenden Methoden sind zum Beispiel in verschiedenen Schreibweisen einer stöchiometrischen Formel, in verschiedenen Trivialnamen, in der systematischen Nomenklatur oder in linearen Notationen dokumentiert.

Sowohl die Trivialnamen als auch die Handelsnamen einer chemischen Verbindung sind häufig die älteste Bezeichnung einer Verbindung und werden darüber hinaus häufig in chemischen Namen einer Substanz verwendet, die von der ursprünglichen Verbindung abgeleitet ist.[8] Diese Namen sind zur Bezeichnung einer chemischen Verbindung nicht besonders geeignet, da die Namen zum einen weder eindeutig und exakt, und zum anderen auf die jeweilige Sprache beschränkt sind. Deshalb wurden verschiedene Methoden zur systematischen Nomenklatur einer chemischen Verbindung entwickelt. Die bekanntesten und am weit verbreitetesten Nomenklaturen sind die IUPAC (**I**nternational **U**nion of **P**ure and **A**pplied **C**hemistry)[9-11] und die CAS (**C**hemical **A**bstract **S**ervice)[12,13] Nomenklatur, sowie das Beilsteinsystem zur Kennzeichnung von organischen Verbindungen.[14] All diese Verfahren zur systematischen Nomenklatur sind kontinuierlich weiter entwickelt worden und durch zahlreiche Erweiterungen der Regeln zur Nomenklatur an die neu entwickelten chemischen Verbindungen angepasst worden. Die Anwendung der Methoden zur systematischen Nomenklatur ist oftmals wegen der komplexen Regeln nur schwierig möglich und erfordert eine intensive Einarbeitung in die Regeln zur Nomenklatur; des weiteren ist der systematische Name einer chemischen Verbindung extrem schwer lesbar und generierbar. Die Verfahren erzeugen darüber

hinaus oftmals keine einheitlichen und eindeutigen Namen für eine chemische Verbindung.

Durch die Weiterentwicklung der Computertechnologie sind neue Möglichkeiten zur Datenverarbeitung und Datenspeicherung von chemischen Informationen entstanden; und hieraus resultierend ein neues Anforderungsprofil an die Verfahren zur Codierung von chemischen Informationen. Eine zur automatischen Datenverarbeitung geeignete systematische Beschreibung von chemischen Verbindungen ist von Morgan entwickelt worden.[15] Basierend auf dieser Systematik sind die Algorithmen der elektronischen Datenbanksuche CAS-ONLINE entwickelt worden.[16] Weitere Erfolge bei der Entwicklung von systematischen Beschreibungen von Molekülen, die zur elektronischen Datenverarbeitung geeignet sind, wurden durch die Anwendung von Graphentheorie auf die Moleküle erreicht.[17-21]

Im Verlauf der Entwicklung der Computertechnologie hat sich die Datenverarbeitung von linearen Zeichenfolgen als besonders effizient erwiesen. Daher sind verschiedene lineare Notationen entwickelt worden um, zum einen die Zusammensetzung des Moleküls, und zum anderen die räumliche Orientierung der Atome, in einer linearen Schreibweise darzustellen. Die Verfahren der linearen Notation sind beliebte und weit verbreitete Methoden, chemische Verbindungen darzustellen.[22] Die Anwendbarkeit von Systemen zur linearen Notation ist dabei weder auf die Beschreibung ganzer Moleküle noch auf die Beschreibung von einfachen Spezies beschränkt.

2.2 Historische Entwicklung von linearen Notationen

Historisch betrachtet ist bereits in dem Jahr 1954 die erste Publikation von Wiswesser[23] erschienen, die eine lineare Notation einer chemischen Verbindung beschreibt. In den folgenden Jahrzehnten sind zahlreiche Publikationen erschienen, die unterschiedliche Verfahren zur Codierung von Molekülen durch eine lineare alphanumerische Zeichenfolge beschreiben. Die bekanntesten Verfahren sind die HOSE-Methode,[24] das ROSDAL-Verfahren,[25] das LNSCS-Verfahren[26] und die zahlreichen weniger bekannten Verfahren zur Codierung von molekularen Teilstrukturen, sowie die Verfahren zur Codierung von nur acyclischen oder nur cyclischen Molekülstrukturen.[27-29] Viele dieser aufgeführten Verfahren sind heute praktisch bedeutungslos, da diese Verfahren eine intensive und umfangreiche Einarbeitung in die komplexen und abstrakten Notationsregeln erfordern und darüber hinaus die Anwendung der Notationsregeln oftmals nicht praktikabel ist. Lediglich die von Wiswesser entwickelte Notation[30, 31] ist in der Chemie etabliert und ist trotz der komplexen Notationsregeln von vielen Chemikern zur Codierung von chemischen Verbindungen verwendet worden. Die Wiswesser Line Notation (WLN) reduziert die zweidimensionale Darstellung einer chemischen Verbindung auf eine kompakte, eindeutige alphanumerische Zeichenfolge unter Berücksichtigung der Struktur. Basierend auf der linearen Wiswesser Notation ist 1980 das AWLN (**A**dvanced **W**iswesser **L**ine **N**otation) Konzept entwickelt worden.[32, 33] Dieses Konzept ist allerdings in der Komplexität der Notationsregeln, durch zahlreiche Ergänzungen, nicht verbessert worden und wird folglich nur selten angewendet.

Durch die rasante Weiterentwicklung der Computertechnologie wurden die technischen Voraussetzungen entwickelt, um zum einen sämtliche chemische Informationen auf den existierenden Hardwaresystemen zu speichern, und zum anderen die Verarbeitung von großen Datenmengen sehr schnell zu realisieren. Aus dieser technischen Entwicklung resultiert ein neues und modernes Anforderungsprofil an die Methoden zur linearen Notation. Es war daher erforderlich, ein weiterentwickeltes System zu konzipieren mit dem die Codierung der relevanten chemischen Informationen äußerst effizient zu erreichen war. Ein derartig weiterentwickeltes und weit verbreitetes System ist SMILES (**S**implified **M**olecular **I**nput **L**ine **E**ntry **S**ystem), das 1987 von Weininger veröffentlicht wurde.[6] Das

SMILES System benötigt nur eine äußerst geringe Anzahl an Codierungsregeln und ist durch die leicht verständliche und flexible Anwendbarkeit der Regeln auf die chemischen Strukturen bei den Benutzern äußerst beliebt. Eine weitere Vereinfachung der Regeln ist in dem SMILES Konzept durch den Verzicht auf die Generierung einer eindeutigen Zeichenfolge zur Darstellung der codierten Molekülstruktur realisiert worden.[6, 34] Die Entwicklung des SMILES Systems basierte auf den Konzepten der Graphentheorie, die auf die Molekülstruktur übertragen wurden und auf Algorithmen, die von der Maschinensprache der Computer abgeleitet worden sind.[6]

Das SMILES System ist gegenwärtig das am häufigsten genutzte Konzept zur linearen Notation bei der Codierung von chemischen Verbindungen. Das ursprüngliche SMILES Konzept ist von Weininger weiter entwickelt worden mit dem Ziel, ein Verfahren zur Erzeugung einer eindeutigen SMILES Zeichenfolge zu generieren. Die Erzeugung einer eindeutigen SMILES Zeichenfolge bei der Codierung der Molekülstruktur ist durch einen speziellen Algorithmus, dem CANGEN Algorithmus, realisiert worden.[35]

Im Jahr 1986 ist ein weiteres Verfahren zur linearen Notation, das im Gegensatz zum SMILES Konzept relativ unbekannt ist, von Wilcox und Levinson publiziert worden.[36] Bei diesem Verfahren wird die Molekülstruktur zunächst in lineare und eventuelle monocyclische Teilstrukturen zerlegt, und anschließend die einzelnen Teilstrukturen durch lineare Zeichenfolgen dargestellt und zu einer linearen Zeichenfolge zusammengefasst. Mit dem von Wilcox und Levinson entwickelten Verfahren können allerdings weder aromatische noch geladene Teilstrukturen codiert werden. Durch die Erweiterung der von Wilcox und Levinson vorgeschlagenen linearen Notation konnte diese Problematik durch zahlreiche Ergänzungen in der linearen Zeichenfolge gelöst werden.[37] Allerdings ist die grundsätzliche Problematik der von Wilcox und Levinson entwickelten Notation, das Nichtvorhandensein eindeutiger Regeln zur Fragmentierung einer Molekülstruktur und die konzeptionelle Beschränkung auf die organische Chemie. Des weiteren konnte die ursprüngliche Problematik der Codierung der dreidimensionalen Struktur nicht durch das erweiterte, aber auf die Codierung der zweidimensionalen Struktur beschränkte, SMILES Konzept gelöst werden.

Basierend auf dem von Weininger publizierten SMILES Konzept ist 1996 von Hinze et al. ein erweitertes und modifiziertes Konzept, das als Broad SMILES bezeichnet wird, entwickelt worden.[7] Das Konzept realisiert die Codierung der dreidimensionalen Struktur einer chemischen Verbindung durch eine kompakte lineare Zeichenfolge. Das ursprüngliche Konzept ist im Rahmen dieser Arbeit überarbeitet und zum SMILES - 3D Konzept weiterentwickelt worden und ist die konzeptionelle Grundlage der hier entwickelten Software WinSmiles - 3D. Eine detaillierte Darstellung der Grundlagen des SMILES - 3D Konzeptes, sowie die Umsetzung des Konzeptes in der Softwareentwicklung wird in den nachfolgenden Kapiteln beschrieben.

Im Jahr 1997 ist eine weitere lineare Notation, die SYBYL Line Notation (SLN), von Ash et al. von der TRIPOS inc. entwickelt und veröffentlicht worden.[38] Die lineare SYBYL Notation basiert ebenfalls auf dem SMILES Konzept und ist zur Codierung der Struktur von organischen Molekülen, von Makromolekülen und Polymeren geeignet. Des weiteren ist das SLN Konzept zur Speicherung von codierten Molekülstrukturen in Datenbanken oder einzelnen Dateien, sowie zur Suche von codierten Molekülstrukturen geeignet und unterstützt die zweidimensionale Suche von molekularen Teilstrukturen.[39-41] Die lineare SYBYL Notation ist im wesentlichen zur Darstellung von molekularen Strukturen bzw. Teilstrukturen, zur Darstellung von Suchmustern, zur zweidimensionalen Suche von Molekülfragmenten und zur Darstellung von kombinierten Teilstrukturen geeignet. Durch zahlreiche Erweiterungen und Modifikationen sind die Zeichenfolgen der SLN weit aus komplexer als die Zeichenfolgen des ursprünglichen SMILES oder des SMILES - 3D Konzeptes. Im Gegensatz zu dem SMILES - 3D Konzept wird bei der SYBYL Notation die dreidimensionale Molekülstruktur nicht direkt durch die lineare Zeichenfolge codiert, sondern kann lediglich durch die optionale Angabe von zuvor berechneten Atomkoordinaten oder Bindungswinkeln zusätzlich in der Notation angegeben werden. Die SLN war ein Versuch, die SMILES Notation zur Codierung von zweidimensionalen Strukturen durch die zusätzliche explizite Angabe von geometrischen Informationen auf die Codierung von dreidimensionalen Molekülstrukturen zu erweitern.

Ein weiteres Anwendungsgebiet von linearen Notationen ist die Codierung von chemischen Strukturformeln zur elektronischen Datenverarbeitung. Im Jahr 1999 ist die XyM Notation von Fujita und Tanaka zur Codierung einer zweidimensionalen chemischen Strukturformel in einer linearen Zeichenfolge publiziert worden.[42, 43] Die XyM Notation ist in eine LaTex2e Anwendung implementiert und kann lediglich zur Codierung einer Strukturformel, nicht aber zur Codierung einer dreidimensionalen Molekülstruktur verwendet werden.

Im Jahr 2001 ist von Gakh und Burnett die MCDL (**M**odular **C**hemical **D**escriptor **L**anguage) Sprache zur Codierung von Molekülstruktur publiziert worden.[44] Die MCDL Sprache codiert die Struktur eines Moleküls und weitere Informationen über die chemische Verbindung durch ein modularisiertes Verfahren in einer linearen Zeichenfolge. Die bei der Codierung durch die MCDL Notation erzeugten wesentlichen Hauptmodule sind zum einen das „Composition Modul“ zur Darstellung der einzelnen Fragmente des Moleküls und zum anderen das „Connection Modul“ zur Darstellung der Verknüpfung der einzelnen Fragmente. Darüber hinaus können weitere ergänzende Module zur Darstellung von weiteren Eigenschaften (z. Bsp. Atomkoordinaten, Bindungstypen oder physikalisch-chemische Informationen) der chemischen Verbindung erzeugt werden.

Die MCDL Sprache ist nicht besonders zur Codierung dreidimensionaler Molekülstruktur geeignet, da einerseits die räumliche Anordnung der einzelnen Atome explizit durch die Eingabe der Koordinaten der Atome in einem ergänzenden Modul und andererseits der Bindungstyp zwischen den Atomen in einem weiteren ergänzendem Modul codiert werden muss. Des weiteren sind die Algorithmen zur Erzeugung eines redundanten „Connection Moduls“ nur teilweise geeignet zur Erzeugung einer eindeutigen linearen Notation. Die Anwendung des MCDL Konzeptes ist auf die Codierung von organischen Verbindungen beschränkt.

2.3 Entwicklungsrelevante Konzepte zur linearen Notation

In den nachfolgenden Unterabschnitten werden die grundlegenden Prinzipien der Wiswesser Line Notation[23, 30, 31] und des SMILES Konzepts[34, 35] erläutert. Die Entwicklung dieser beiden Konzepte zur linearen Notation ist, unter Berücksichtigung des historischen Entwicklungsprozesses, als besonders bedeutend einzustufen. Diese Entwicklungsrelevanz ist darüber hinaus durch die Verwendung dieser beiden linearen Notationen in vielfältigen Anwendungsbereichen dokumentiert.

2.3.1 Das Konzept der Wiswesser Line Notation (WLN)

Bereits im Dezember 1949 hat Wiswesser begonnen, die ersten Symbole und Regeln einer neuen Notation zur Codierung von Molekülstruktur zu entwickeln.[32] Diese ursprünglichen Regeln und Zeichen zur Darstellung der chemischen Verbindungen sind bis zum Juni 1950 weiter zusammengefasst und vereinfacht worden. Basierend auf den weiter entwickelten Regeln und Zeichen zur Codierung der Molekülstruktur ist das Konzept der Wiswesser Line Notation erstmals im Jahr 1954 von Wiswesser veröffentlicht worden.[23] Die WLN Notation hat sich in den folgenden Jahrzehnten in der Chemie etabliert und die Anwendung dieser Notation war weit verbreitet.

Das Konzept der Wiswesser Line Notation verwendet zur Codierung nur die großen alphabetischen Buchstaben, die Zahlen 0 bis 9, die Symbole &, /, - und das Leerzeichen.

Der erste Schritt der Verschlüsselung einer Molekülstruktur durch die Wiswesser Line Notation ist die Aufspaltung der chemischen Strukturformel in einzelne Atome und häufig vorkommende vordefinierte Molekülfragmente. Im nächsten Schritt werden die Atome und Molekülfragmente, die durch die Aufspaltung generiert werden, durch definierte Buchstaben und Zahlen codiert. Diese Symbole und Zahlen werden entsprechend der Bindungen zwischen den einzelnen Atomen oder den Molekülfragmenten zu einer linearen Zeichenfolge zusammengefasst. Bei

der Zusammenfassung wird die längste Bindungskette des Moleküls ermittelt und der Anfang der linearen Zeichenfolge auf Basis einer hierarchischen Ordnung festgelegt.

Nachstehend erfolgt eine kurze, punktuelle Beschreibung der wesentlichen Regeln zur Codierung und der Definitionen der Codierungszeichen.

1) Aufspaltung der Molekülstruktur

Die Struktur eines Moleküls wird in nicht verzweigte, gesättigte Seitenketten (Alkylketten), in vordefinierte Molekülfragmente (z. B. Carbonylgruppe oder Phenylrest) und in einzelne Atome, die kein Bestandteil der zuvor definierten Fragmente sind, aufgespaltet.

2) Codierung der Atome

In der WLN werden die Atome durch einen großen Buchstaben codiert und unterscheiden sich somit von den Atomsymbolen der IUPAC Nomenklatur.

a) Wasserstoffatome

Wasserstoffatome werden durch den Buchstaben H dargestellt. Der Buchstabe H wird in der linearen Zeichenfolge immer direkt nach dem Symbol des Fragments, mit dem das Wasserstoffatom verbunden ist, dargestellt.

b) Kohlenstoffatome

Das Kohlenstoffatom wird in Abhängigkeit von der Konstitution des Fragments durch eine Zahl oder durch den Buchstaben C, Y oder X dargestellt.

Eine nicht verzweigte, gesättigte Alkylkette wird durch die Zahl dargestellt, deren Wert der Anzahl der Kohlenstoffatome der Alkylkette entspricht.

Ein sekundäres Kohlenstoffatom, das mit zwei anderen Atomen, die keine Wasserstoffatome sind, verbunden ist, wird durch den Buchstaben C dargestellt.

Ein tertiäres Kohlenstoffatom, das mit drei anderen Atomen, die keine Wasserstoffatome sind, verbunden ist, wird durch den Buchstaben Y dargestellt.

Ein quartäres Kohlenstoffatom, das mit vier anderen Atomen, die keine Wasserstoffatome sind, verbunden ist, wird durch den Buchstaben X dargestellt.

c) Sauerstoffatome

Ein Sauerstoffatom das mit zwei anderen Atomen, die keine Wasserstoffatome sind, verbunden ist, wird durch den Buchstaben O dargestellt. Sauerstoffatome, die Bestandteil eines definierten Molekülfragments sind, werden durch die Symbole des Fragments codiert.

d) Halogenatome

Die Atome Fluor und Jod werden durch die Buchstaben F und J dargestellt. Das Chloratom wird durch den Buchstaben G und das Bromatom durch den Buchstaben E dargestellt.

3) Codierung von Molekülfragmenten

In der WLN Notation werden verschiedene molekulare Strukturen durch Molekülfragmente codiert, die in der linearen Notation durch einzelne Buchstaben dargestellt werden.

Die nachfolgende Tabelle 2-1 zeigt die Aufstellung der Codierung der vordefinierten Molekülfragmente.

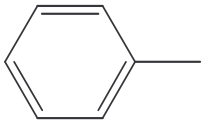
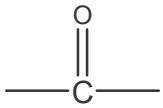
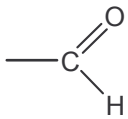
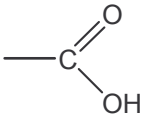
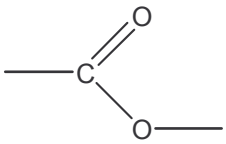
Funktionelle Gruppe		WLN – Symbol
Alkan	$H\text{-}(CH_2)_n\text{-}H$	*n*H, *n*=1, 2, 3,...
Alken	-CH=CH-	U
Alkin	-CC-	UU
Phenylrest		R
Halogenide	-F, -Cl, -Br, -I	F, G, B, I
Hydroxylgruppe	-OH	Q
Ether	-O-	O
Carbonylgruppe		V
Aldehydgruppe		VH
Carbonsäure		VQ
Ester		VO
primäres Amin	$-NH_2$	Z
sekundäres Amin	-NH-	M
tertiäres Amin	$-NR_2$	N
Nitrilgruppe	-CN	CN
Nitrogruppe	$-NO_2$	NW
Sulfidgruppe	-S-	S
Thiolgruppe	-SH	SH

Tabelle 2-1: WLN - Notation der vordefinierten Molekülfragmente

4) Codierung von Bindungen

Eine Einfachbindung zwischen zwei Atomen wird nicht explizit dargestellt, sondern ergibt sich indirekt aus der Reihenfolge der einzelnen Zeichen, die das Molekül codieren. Zur Darstellung einer Doppelbindung bzw. einer Dreifachbindung zwischen zwei Atomen wird der Buchstabe U bzw. die Buchstaben UU verwendet.

Durch das Voranstellen des Zeichens % wird ein Atom oder Molekülfragment gekennzeichnet, das an eine vor dem letzten Atom oder Fragment freie Bindungsstelle gebunden ist.

5) Codierung von cyclischen Fragmenten

Zur Codierung von cyclischen Fragmenten wird zur Kennzeichnung des ersten Atoms des cyclischen Fragmentes der Buchstabe L und zur Kennzeichnung des letzten Atoms des cyclischen Fragmentes der Buchstabe J verwendet. Zwischen den Buchstaben J und L werden die einzelnen Atome oder Fragmente des cyclischen Fragmentes codiert. Ein vollständig gesättigtes cyclisches Bindungssystem wird zusätzlich vor dem Buchstaben J mit dem Buchstaben T gekennzeichnet.

Eventuelle Substituenten des cyclischen Fragmentes werden durch ein Leerzeichen von der Ringnotation getrennt. Die Bindungspositionen im cyclischen Bindungssystem werden entsprechend der alphabetischen Reihenfolge mit Buchstaben gekennzeichnet. In der Notation wird der Buchstabe der Bindungsstelle der Notation des Substituenten vorangestellt.

6) Hierarchische Ordnung der Notation

Zur Erzeugung einer eindeutigen linearen Notation ist in der WLN der Anfang einer linearen Zeichenfolge durch eine hierarchische Ordnung festgelegt. Das Atom oder das Fragment mit der höchsten Priorität, entsprechend der in Abbildung 2-1 dargestellten Reihenfolge, ist der Anfang der linearen Notation. Die weitere Reihenfolge der einzelnen Atome oder der Molekülfragmente in der linearen Notation ergibt sich aus den Bindungsverhältnissen des Moleküls.

Aromatischer Ring	Symbole	Alkylkette	Funktionelle Gruppen
R	& - /	0 1 2 3 4 5 6 7 8 9	A B C D E ...
höchste Priorität	⟶		geringste Priorität

Abbildung 2-1: Hierarchische Reihenfolge der Molekülfragmente der WLN - Notation

Die Algorithmen der WLN sind in der Software CROSSBOW (**C**omputerized **R**etrieval of **O**rganic **S**tructure**s** **B**ased **o**n **W**iswesser)[45] zur Eingabe der Molekülstruktur umgesetzt.

Die Wiswesser Line Notation erzeugt in einem sehr kompakten Codierungsprozess auf Basis einer chemischen zweidimensionalen Strukturformel eine eindeutige lineare Zeichenfolge und ist daher zur Registrierung von chemischen Verbindungen angewendet worden. Das Verfahren zur Indexierung von Teilstrukturen, Chemical Substructure Index[46] des Institute of Scientific Information (ISI) und das Verfahren Index Chemicus Registry System (ICRS)[47] basieren auf den Prinzipien der Wiswesser Line Notation. Des weiteren ist die WLN zur Suche von Teilstrukturen und Molekülstrukturen in geeigneten Datenbanken durch die Permutation der linearen Notation verwendet worden.[48] Durch die Integration der Software CROSSBOW in ein interaktives online Informationssystem ist die Kombination einer strukturbasierenden Datenbanksuche mit der Datenbanksuche nach funktionellen Molekülfragmenten realisiert worden.[49]

Die gravierendsten Nachteile der Wiswesser Line Notation sind zum einen die Beschränkung der Regeln auf die Codierung von organischen Molekülen, und zum anderen die komplexen Regeln zur Codierung der Atome und der Fragmente. Insbesondere bei polycyclischen Verbindungen ist es schwierig, die längste Kette der miteinander verbundenen Atome zu ermitteln. Des weiteren ist die durch die WLN bei der Codierung einer chemischen Verbindung erzeugte lineare Zeichenfolge für einen Chemiker nur schwer lesbar und ohne grundlegende Kenntnis der Notationsregeln nicht intuitiv verständlich. Dies ist insbesondere auf die bei der Codierung verwendeten Zeichen, die oftmals von den Zeichen der Atomsymbole der IUPAC Nomenklatur abweichen, zurück zu führen.

Die Wiswesser Line Notation ist nicht geeignet zur Codierung von Bindungslängen, Bindungswinkeln oder Atomeigenschaften.

2.3.2 Das SMILES Konzept

Durch die rasante Weiterentwicklung der Computertechnologie und durch die starke Zunahme der Anzahl der Informationen über die chemischen Verbindungen hat sich das Anforderungsprofil an ein Verfahren zur linearen Notation entscheidend verändert. Das weiterentwickelte Anforderungsprofil an ein Verfahren zur linearen Notation ist, neben den existierenden Anforderungen zur Speicherung und erneuten Verfügbarkeit von chemischen Informationen, dahingehend erweitert worden, dass ein Verfahren zur linearen Notation die chemisch relevanten Informationen effektiv codieren sollte.

Im Jahr 1986 ist von Weininger das Verfahren SMILES (**S**implified **M**olecular **I**mput **L**ine **E**ntry **S**ystem) entwickelt worden. Das SMILES Konzept ist ein hoch effizientes Verfahren zur linearen Notation von chemischen Verbindungen, das zur Datenverarbeitung von chemischen Informationen gut geeignet ist. Das Konzept ist basierend auf den Prinzipien der molekularen Graphentheorie entwickelt worden.

Die Terminologie des SMILES Konzeptes definiert als Molekülstruktur die zweidimensionale Strukturformel eines Moleküls. Diese wird in der chemischen Literatur häufig zur Darstellung eines Moleküls verwendet und vereinfacht hierdurch grundlegend die dreidimensionale Molekülstruktur. Bei der Codierung dieser Molekülstruktur durch das SMILES Konzept wird die chemische Verbindung durch eine lineare Zeichenfolge, bestehend aus Buchstaben, Zahlen und Symbolen, dargestellt. Das Ende der linearen Zeichenfolge ist durch ein Leerzeichen gekennzeichnet. Die Wasserstoffatome der zu codierenden chemischen Verbindung müssen in der linearen SMILES Notation nicht explizit codiert werden, sie können aber optional ergänzt werden. Nachfolgend werden die grundlegenden Codierungsregeln der SMILES Notation kurz punktuell beschrieben.

1) Codierung der Atome

Die Atome werden in der linearen Zeichenfolge durch die IUPAC-Atomsymbole dargestellt. Die Atomsymbole aller Elemente werden durch eckige Klammern eingeschlossen, mit Ausnahme der in der organischen Chemie häufig vorkommenden Elemente Kohlenstoff, Stickstoff, Phosphor, Sauerstoff, Schwefel, Bor und der Halogene. Bei diesen Elementen können die eckigen Klammern weggelassen werden, wenn die Anzahl der Wasserstoffatome, die mit dem Element verbunden sind, gleich der Anzahl der freien Bindungsvalenzen des Atoms ist.

Die Spezifikation (Atomladung, Wasserstoffatome) eines Atoms erfolgt immer innerhalb der eckigen Klammern unabhängig von den Buchstaben des Atomsymbols.

2) Codierung von Wasserstoffatomen

Die Wasserstoffatome eines Moleküls werden in der Regel nicht explizit codiert, optional können die Wasserstoffatome zusätzlich spezifiziert werden. Die Spezifikation von Wasserstoffatomen erfolgt immer innerhalb der eckigen Klammern des Atoms, an das die Wasserstoffatome gebunden sind, durch die Ergänzung des Atomsymbols H. Die Darstellung von weiteren Wasserstoffatomen, die an dasselbe Atom gebunden sind, erfolgt durch eine dem Wasserstoffsymbol H nachfolgende optionale Zahl, deren Wert der Anzahl der Wasserstoffatome entspricht, oder durch weitere optionale Atomsymbole für das Wasserstoffatom.

3) Codierung von formalen Ladungen

Die formale Ladung eines Atoms wird durch das Symbol +, für eine positive Ladung, oder das Symbol -, für eine negative Ladung innerhalb der eckigen Klammern dargestellt. Durch eine optionale Zahl entweder vor oder nach dem Ladungssymbol ist eine entsprechende Multiplikation der Ladung möglich.

4) Codierung der Bindungen

Die Symbole -, =, # und : werden zur Darstellung einer Einfach-, Doppel-, Dreifachbindung und einer aromatischen Bindung zwischen zwei Atomen verwendet. Die Verwendung der Zeichen zur Codierung einer Einfachbindung bzw. einer aromatischen Bindung ist optional, da die aufeinander folgenden Atomsymbole in der linearen Zeichenfolge standardmäßig mit einer Einfachbindung bzw. bei kleingeschriebenen Atomsymbolen durch eine aromatische Bindung miteinander verbunden werden.

5) Codierung von Seitenketten und Verzweigungen

Die Verzweigung bzw. die Seitenkette in einer Molekülstruktur wird entsprechend der Regeln zur Codierung von Atomen und Bindungen codiert. Die lineare Notation der Seitenkette wird in runden Klammern eingeklammert und so innerhalb der linearen Notation des Moleküls als Seitenkette bzw. Verzweigung gekennzeichnet.

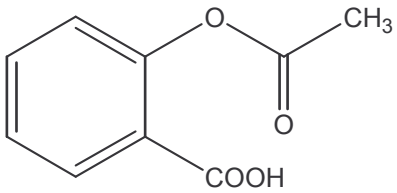
6) Codierung von cyclischen Strukturen

Zur Codierung von Molekülen mit cyclischen Strukturen wird jeder Ring des Moleküls durch die formale Spaltung einer Einfachbindung oder einer aromatischen Bindung in eine lineare Molekülstruktur überführt. Jede gespaltene Bindung wird mit einer Zahl nummeriert und die lineare Struktur entsprechend der Notationsregeln codiert. Dabei werden die Atome der gespaltenen Bindung in der linearen Notation direkt nach dem Atomsymbol mit der Nummer der gespaltenen Bindung gekennzeichnet. Innerhalb einer linearen Zeichenfolge werden die in Leserichtung von links nach rechts aufeinander folgenden Atomsymbole, die mit der gleichen Zahl gekennzeichnet sind, als die Atome der gespaltenen Bindung interpretiert. Die Nummer einer gespaltenen Bindung kann innerhalb einer linearen Notation mehrfach verwendet werden, wenn die Zuordnung der Atome zu den gespaltenen Bindungen eindeutig ist. Bei der Verwendung von Zahlen deren Wert größer 10 ist, wird die Zahl durch ein vorangestelltes Prozentzeichen zusätzlich gekennzeichnet.

7) Codierung von aromatischen Bindungsstrukturen
Aromatische cyclische Bindungsstrukturen werden durch die Verwendung von kleingeschriebenen Buchstaben für die entsprechenden Atomsymbole der Atome codiert.

8) Codierung von nicht kovalenten Bindungen
Die Kennzeichnung von nicht kovalenten Bindungen zwischen molekularen Teilstrukturen erfolgt durch das Punktzeichen.

Durch die zuvor beschriebenen grundsätzlichen Regeln der SMILES Notation ist es leicht möglich, die zweidimensionale Molekülstruktur von chemischen Verbindungen durch eine lineare SMILES Zeichenfolge darzustellen. Die Abbildung 2-2 zeigt die Anwendung der SMILES Notation auf die zweidimensionale Strukturformel von Acetylsalicylsäure.



SMILES: CC(=O)O%1ccccc%1cC(=O)O

Abbildung 2-2: SMILES Notation von Acetylsalicylsäure (Aspirin)

Eine weitere Vereinfachung des SMILES Konzeptes hat Weininger durch die Beschränkung der Notationsregeln auf einen Großteil der organischen Verbindungen erreicht und die sogenannten Basic SMILES als eine Unterklasse des SMILES Konzeptes definiert.[34] Die entscheidenden Vereinfachungen dieser Unterklasse sind zum einen die Beschränkung des Konzeptes auf die Elemente Wasserstoff, Kohlenstoff, Stickstoff, Phosphor, Sauerstoff, Schwefel, Fluor, Chlor Brom und Jod; und zum anderen der Verzicht auf die optionale Spezifikation von formalen Atomladungen, von zusätzlichen Wasserstoffatomen und von aromatischen Bindungseigenschaften.

Zur Erreichung einer einheitlichen Notation von unterschiedlichen chemischen Verbindungen durch das SMILES Konzept hat Weininger weitere ergänzende

Konventionen zur Codierung definiert.[34] Diese Konventionen zur SMILES Notation definieren detailliert die Spezifikationen von Bindungen und zusätzlichen Wasserstoffatomen, sowie die aromatischen Eigenschaften einer chemischen Verbindung.

Im Rahmen dieser kurzen Beschreibung des SMILES Konzeptes wird weder auf die Regeln der Basic SMILES, noch auf die zusätzlichen Konventionen der SMILES Notation im Detail eingegangen.

Die zweidimensionale Struktur einer chemischen Bindung kann durch eine Vielzahl von unterschiedlichen SMILES Notationen dargestellt werden. Der wesentliche Grund für die synonymen linearen Notationen ist, dass im SMILES Konzept keine hierarchischen Regeln zur Codierungsreihenfolge der Atome existieren. Dies hat zur Folge, dass zum einen der Beginn der linearen Notation in der zweidimensionalen Molekülstruktur nicht festgelegt, und zum anderen dass die Codierungsreihenfolge der Seitenketten an Verzweigungsstellen nicht definiert ist. Zur Generierung von eindeutigen SMILES Notationen haben Weininger et al. einen zweistufigen Algorithmus entwickelt.[35] Der CANGEN Algorithmus erzeugt im ersten Schritt eine kanonische Struktur aus der SMILES Zeichenfolge und generiert anschließend auf Basis dieser Struktur eine eindeutige SMILES Notation. Bei der Generierung der kanonischen Struktur wird das Molekül zunächst als mathematischer Graph, bestehend aus Ecken (Atome) und Kanten (Bindungen), betrachtet und im weiteren die Atome des Moleküls hierarchisch geordnet und gekennzeichnet.

Die SMILES Notation ist in zahlreiche Programme implementiert worden. Zunächst ist die SMILES Notation in Programmen zur effizienten Speicherung von Informationen von chemischen Verbindungen, sowie zur schnellen Verfügbarkeit der gespeicherten Informationen verwendet worden. Ein weiteres Anwendungsgebiet ist die Integration der SMILES Notation als Eingabe-Interface in Programmen zur Darstellung von Molekülstrukturen. Das Programm DEPICT ist von Weininger zur Darstellung eines Moleküls durch eine zweidimensionale Molekülstruktur aus der SMILES Notation entwickelt worden.[50] Des weiteren werden auf Basis der SMILES Notation verschiedene Moleküleigenschaften berechnet. Das Programm ESCHER

berechnet auf Basis der SMILES Notation die Symmetrie des codierten Moleküls.[51] Ein weiteres Programm ermittelt aus der SMILES Notation die funktionellen Gruppen der codierten chemischen Verbindung und berechnet Vorhersagewerte der thermophysikalischen Eigenschaften des Moleküls. [52]

Die SMILES Notation ist besonders gut zur Speicherung und zur Suche von molekularen Strukturen bzw. Teilstrukturen in Datenbanken geeignet. Das Programm CHUCKLES realisiert die Darstellung und die Suche von Peptiden bzw. Peptidsequenzen in Datenbanken durch die Konvertierung der Peptide in SMILES Notationen auf der atomaren Darstellungsebene.[53] Basierend auf dem CHUCKLES Algorithmus ist das Programm CHOCKLES zur Darstellung von Oligomeren entwickelt worden.[54] Der CHOCKLES Algorithmus realisiert die Darstellung und die Suche der Oligomere ebenfalls durch die Codierung der Monomereinheiten in die SMILES Notation.

Das SMILES Konzept ist geeignet, acyclische und cyclische chemische Verbindungen auf Basis der zweidimensionalen Molekülstruktur in einer linearen Zeichenfolge zu codieren. Die SMILES Notation ermöglicht die einfache Codierung von Seitenketten und verzweigten Molekülstrukturen, sowie die Spezifikation von Ladungen und molekularen Teilladungen. Die Zeichenfolge der SMILES Notation einer Molekülstruktur ist äußerst kompakt und gut lesbar, da die explizite Codierung von Wasserstoffatomen nicht erforderlich ist. In Folge der wenigen und leicht verständlichen Regeln zur Codierung der Molekülstruktur ist das SMILES Konzept weit verbreitet und in zahlreiche Programme, wie zuvor beschrieben, implementiert.

Das SMILES Konzept ist allerdings nicht zur Codierung einer dreidimensionalen Molekülstruktur geeignet, da die SMILES Notation lediglich die Atome und die Bindungsverhältnisse zwischen den Atomen in der linearen Zeichenfolge codiert. Zur Codierung der dreidimensionalen Molekülstruktur ist es aber erforderlich, die Eigenschaften der Atome und die Bindungseigenschaften bei der Codierung zu berücksichtigen.

Das ursprüngliche SMILES Konzept ist durch zahlreiche Erweiterungen der Zeichen zur Codierung und durch die Ergänzung der Codierungsregeln modifiziert worden.

Basierend auf dem SMILES Konzept ist von Weininger das SMART Konzept zur linearen Darstellung von molekularen Teilstrukturen entwickelt worden.[55] Zur Darstellung von chemischen Reaktionen ist die SMILES Notation von Bone et al. auf die Codierung von chemischen Reaktionen übertragen worden und die Codierungsregeln in dem SUPER-SMILES Konzept erweitert worden.[56]

3. Das weiterentwickelte SMILES - 3D Konzept

3.1 Vorbemerkungen zum SMILES - 3D Konzept

Die zuvor in Kapitel 2 näher beschriebenen Konzepte zur linearen Notation, die Wiswesser Line Notation und das SMILES Konzept generieren die lineare Zeichenfolge auf Basis der zweidimensionalen Strukturformel der zu codierenden chemischen Verbindung und sind auch nur zur Erzeugung der zweidimensionalen Struktur aus der linearen Notation geeignet. Zur Codierung der dreidimensionalen Struktur einer chemischen Verbindung durch eine lineare Zeichenfolge und zu der umgekehrten Generierung der dreidimensionalen Struktur einer chemischen Verbindung aus der linearen Notation ist es erforderlich, die Konzepte zur linearen Notation zu erweitern und die Codierungsregeln entsprechend zu modifizieren. Ein entsprechend modifiziertes Konzept ist das SMILES - 3D Konzept, das im Rahmen dieser Arbeit, ausgehend von dem von Hinze et al. publizierten Broad SMILES Konzept, entwickelt worden ist.[7] Das SMILES - 3D Konzept basiert, wie der Konzeptname implizit andeutet, auf dem SMILES Konzept, das sofern erforderlich ergänzt und erweitert worden ist, um eine dreidimensionale Molekülstruktur durch eine eindeutige, bijektive lineare Notation darzustellen. Die notwendigen Ergänzungen der ursprünglichen SMILES Notation sind dabei auf ein Minimum an äußerst kurzen und eindeutigen Modifikationen reduziert. In den folgenden Abschnitten werden die Regeln zur Codierung der dreidimensionalen Struktur einer chemischen Verbindung im Detail beschrieben.

3.2 Grundsätzliche Codierungsregeln

Die Codierung der dreidimensionalen Molekülstruktur erfolgt durch eine lineare Zeichenfolge die aus Buchstaben, Zahlen und Symbolen besteht. In der linearen Zeichenfolge wird ein Atom durch die entsprechenden Atomsymbole der IUPAC Nomenklatur und die Bindung zwischen zwei Atomen durch Bindungssymbole dargestellt. Jedes Atomsymbol und jedes Bindungssymbol kann zusätzlich spezifiziert werden. Darüber hinaus wird jedem Atom ein Zelltyp zur Codierung der dreidimensionalen Struktur zugewiesen. Der Zelltyp repräsentiert die

räumliche Anordnung der Valenzorbitale des Atoms und bestimmt die Bindungsstellen zwischen zwei benachbarten Atomen. Die dreidimensionale Struktur der Zelle kann durch die optionale Spezifikation der Atomsymbole und der Bindungssymbole modifiziert werden.

Zur eindeutigen Kennzeichnung der einzelnen Atome in der linearen SMILES - 3D Notation einer dreidimensionalen Molekülstruktur werden die Begriffe „From-Atom“, „To-Atom“ und „Add-Atom“ verwendet. Der Begriff „From-Atom“ bezeichnet das Atom, das in der linearen Notation durch ein bereits existierendes Atomsymbol dargestellt ist. Das "To-Atom" ist das Atom, das in der existierenden linearen Notation durch das nachfolgende Atomsymbol codiert und an das "From-Atom" gebunden ist. Das "Add-Atom" ist das neu darzustellende Atom, das in der linearen Zeichenfolge noch nicht vorhanden ist und in der linearen Notation durch das hinzuzufügende Atomsymbol codiert wird.

Die Verwendung der verschiedenen Begriffe zur Kennzeichnung der Beziehung zwischen den Atomen ist in Abbildung 3-1 schematisch dargestellt.

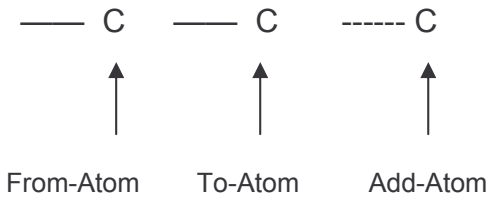


Abbildung 3-1: Schematische Darstellung von From-Atom, To-Atom und Add-Atom

3.3 Die dreidimensionale Molekülstruktur

Zur Darstellung der dreidimensionalen Struktur eines Moleküls wird jedem Atom eine Zelle zugewiesen. Die Zelle eines Atoms repräsentiert die räumliche Anordnung der Valenzorbitale des Atoms, die zur Bildung der Bindungen oder zur Darstellung freier Elektronenpaare verwendet werden, und bestimmt die Bindungsstellen zwischen zwei benachbarten Atomen. Das Atom ist immer im Zentrum der zugewiesenen Zelle angeordnet.

3.3.1 Die Zelltypen

In dem SMILES - 3D Konzept werden sechs unterschiedliche Zelltypen zur Realisierung der räumlichen Anordnung der „Elektronengruppen“ um ein Atom verwendet. Der Begriff „Elektronengruppe“ bezeichnet das elektronische Bindungsverhältnis des Atoms entsprechend der Regeln des VSEPR (Valence Shell Electron Pair Repulsion) Konzepts[57, 58] , sie kann ein einzelnes Elektron, eine Einfachbindung, ein freies Elektronenpaar, eine Doppelbindung oder eine Dreifachbindung bedeuten.

In das Konzept sind die folgenden Zelltypen implementiert:

1) Monohedrale Zelle (1 Elektronengruppe)
   Die monohedrale Zelle stellt eine Elektronengruppe dar und verfügt über eine Bindungsstelle. Die Zelle wird zur Darstellung eines terminalen Atoms, z. B. eines Wasserstoffatoms oder des Liganden L zur Spezifikation der freien Bindungsstelle eines Molekülfragmentes verwendet. Des weiteren wird der Zelltyp zur Darstellung eines einfach gebundenen Halogenatoms, zur Darstellung der mit einer Doppelbindung gebundenen Atome der Chalkogengruppe und zur Darstellung der mit einer Dreifachbindung gebundenen Atome der Stickstoffgruppe verwendet.

2) Dihedrale Zelle (2 Elektronengruppen)
   Die dihedrale Zelle stellt zwei Elektronengruppen dar und verfügt über zwei Bindungsstellen die linear angeordnet sind. Der Winkel zwischen den Bindungsstellen beträgt 180°. Die Zelle wird zur Darstellung der Struktur von normal gebundenen Erdalkaliatomen verwendet. Des weiteren wird der Zelltyp zur Darstellung der Struktur eines sp-hybridisierten Kohlenstoffatoms, das entweder eine Einfachbindung und eine Dreifachbindung oder zwei Doppelbindungen besitzt, verwendet.

3) Trigonale Zelle (3 Elektronengruppen)

Die trigonale Zelle stellt drei Elektronengruppen dar und besitzt drei Bindungsstellen die trigonal planar angeordnet sind. Der Bindungswinkel zwischen den Bindungen beträgt 120°. Die Zelle wird zur Darstellung der Struktur der Elemente der Borgruppe verwendet. Des weiteren wird der Zelltyp zur Darstellung eines sp^2-hybridisierten Kohlenstoffatoms, das eine Doppelbindung und zwei Einfachbindungen oder zwei aromatische Bindungen (die aromatische Bindung wird durch einen Doppelpunkt codiert) besitzt, verwendet.

4) Tetrahedrale Zelle (4 Elektronengruppen)

Die Zelle vom Typ Tetrahedral stellt vier Elektronengruppen dar und besitzt vier Bindungsstellen die tetraedrisch um das zentrale Atom angeordnet sind. Die Zelle wird zur Darstellung der Struktur eines sp^3-hybridisierten Kohlenstoffatoms, das vier Einfachbindungen besitzt, verwendet. Des weiteren wird die Zelle, zum Beispiel zur Darstellung der Struktur eines Stickstoffatoms das drei Einfachbindungen besitzt, oder zur Darstellung der Struktur eines Sauerstoffatoms das zwei Einfachbindungen besitzt, verwendet. In diesen Fällen werden die nicht mit Einfachbindungen besetzten Bindungsstellen mit den freien Elektronenpaaren des Atoms belegt.

5) Pentagonale Zelle (5 Elektronengruppen)

Die pentagonale Zelle stellt fünf Elektronengruppen dar und besitzt fünf Bindungsstellen, die trigonal-bipyramidal um das zentrale Atom angeordnet sind. Die Zelle wird zum Beispiel zur Darstellung der Struktur eines Phosphoratoms, das fünf Einfachbindungen besitzt, verwendet.

6) Oktahedrale Zelle (6 Elektronengruppen)

Die Zelle vom Typ Oktahedral stellt sechs Elektronengruppen dar und besitzt sechs Bindungsstellen, die oktaedrisch um das zentrale Atom angeordnet sind. Dieser Zelltyp wird zum Beispiel zur Darstellung eines Schwefelatoms, das sechs Einfachbindungen besitzt, verwendet.

3.3.2 Nummerierung der Bindungsstellen einer Zelle

Zur Erzeugung einer eindeutigen dreidimensionalen Molekülstruktur aus der linearen Notation durch die Zuordnung von unterschiedlichen Zelltypen zu den entsprechend codierten Atomen, ist es erforderlich, die Bindungsstellen einer Zelle eindeutig zu kennzeichnen und die Reihenfolge, in der die Bindungsstellen einer Zelle mit den Bindungsstellen der Zellen der Add-Atome verbunden werden, eindeutig festzulegen. Die Nummerierung der Bindungsstellen der verschiedenen Zelltypen ist durch die folgende Konvention definiert:

1) Monohedrale Zelle
   Die monohedrale Zelle verfügt nur über eine Bindungsstelle, daher ist eine spezielle Nummerierung nicht erforderlich.

2) Dihedrale Zelle
   Die dihedrale Zelle wird immer mit der ersten Bindungsstelle an das bereits existierende From-Atom gebunden und somit die verbleibende freie Position der Zelle als zweite Bindungsstelle festgelegt.

3) Trigonale Zelle
   Die trigonale Zelle wird immer mit der ersten Bindungsstelle an das bereits existierende From-Atom gebunden. Die zweite und dritte Bindungsstelle sind in einer Ebene und die räumliche Orientierung dieser Bindungsstellen erfolgt durch die Spezifizierung des Dihedralwinkels der Bindung mit dem From-Atom.

4) Tetrahedrale Zelle

Die tetraedrische Zelle wird mit der ersten Bindungsstelle an das bereits existierende From-Atom gebunden. Die Nummerierung der zweiten, dritten und vierten Bindungsstelle erfolgt mit Blickrichtung von der ersten Bindungsstelle zum Add-Atom gegen die Uhrzeigerrichtung. Die weitere räumliche Orientierung der zweiten bis vierten Bindungsstelle erfolgt durch die Spezifikation des Dihedralwinkels. In Abbildung 3-2 ist die Nummerierung der Bindungsstellen einer tetraedrischen Zelle schematisch dargestellt.

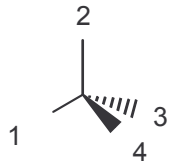


Abbildung 3-2: Nummerierung der tetraedrischen Zelle

5) Pentagonale Zelle

Die erste Bindungsstelle mit der die pentagonale Zelle standardmäßig an das bereits existierende From-Atom gebunden wird ist eine äquatoriale Position. Die zweite Bindungsstelle der Zelle ist immer eine axiale Position. Die dritte, vierte und fünfte Bindungsstelle wird mit Blickrichtung von der ersten Bindungsstelle der Zelle zum Atom gegen den Uhrzeigersinn angeordnet. Die Nummerierung der standardmäßigen pentagonalen Zelle ist in Abbildung 3-3 schematisch dargestellt.

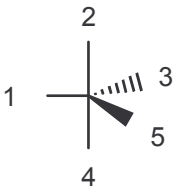


Abbildung 3-3: Nummerierung der pentagonalen Zelle

5 a) Axiale Pentagonale Zelle

Die erste Bindungsposition mit der die axiale pentagonale Zelle an das existierende From-Atom gebunden wird ist immer die axiale Position, die zweite Bindungsposition ist ebenfalls eine axiale Position, die entgegengesetzt zu der ersten Position orientiert ist.

Die übrigen drei Bindungsstellen werden gegen den Uhrzeigersinn mit Blickrichtung von der ersten Bindungsstelle zum Atom nummeriert.

In Abbildung 3-4 ist die Nummerierung der axialen pentagonalen Zelle dargestellt.

Abbildung 3-4: Nummerierung der axialen pentagonalen Zelle

6) Oktahedrale Zelle

Die erste Bindungsstelle der oktaedrischen Zelle wird an das bereits existierende From-Atom gebunden. Die zweite Bindungsstelle der Zelle ist immer die in der Zelle der ersten Bindungsstelle gegenüber liegende Bindungsstelle. Die übrigen vier Bindungsstellen werden mit Blickrichtung von der ersten Bindungsstelle zum Atom gegen den Uhrzeigersinn nummeriert. Die räumliche Orientierung der dritten bis sechsten Bindungsstelle erfolgt durch die Spezifikation des Dihedralwinkels. Die Anordnung der Bindungsstellen der oktraedrischen Zelle ist in Abbildung 3-5 schematisch dargestellt.

Abbildung 3-5: Nummerierung der oktraedrischen Zelle

Bei der Generierung der Molekülstruktur werden die freien Bindungsstellen eines Atoms, die nicht explizit mit einem Liganden verbunden sind, mit

Wasserstoffatomen besetzt, vorausgesetzt die freien Bindungsstellen verfügen über ein freies Elektron zur Bildung einer Bindung. Die Wasserstoffatome werden an die freien Bindungsstellen niedrigster Ordnung gebunden. In Abhängigkeit von der Anzahl der freien Valenzelektronen werden die freien Bindungsstellen höherer Ordnung mit Elektronenpaaren besetzt.

Besitzt die Zelle des ersten in der linearen SMILES - 3D Notation codierten Atoms mehr als eine Bindungstelle, so wird die erste Bindungsstelle der Zelle automatisch mit einem Wasserstoffatom verbunden. Die weiteren Bindungsstellen der Zelle werden mit den in der linearen Zeichenfolge nachfolgend codierten Atomen besetzt.

3.3.3 Der Dihedralwinkel zwischen verbundenen Zellen

Zur eindeutigen Generierung der dreidimensionalen Molekülstruktur aus der linearen Notation ist es sinnvoll, den dihedralen Winkel zwischen den verbundenen Strukturzellen der codierten Atome zu definieren. Für die Verknüpfung der Zellen durch eine Bindung sind für die verschiedenen Kombinationsmöglichkeiten der unterschiedlichen Zelltypen Standardwerte für den dihedralen Winkel festgelegt worden. Die Konvention basiert auf dem Prinzip, mit Ausnahme der pentagonalen Zellen, dass die niedrigst mögliche Bindungsstelle der strukturbildenden Zelle des Add-Atom in einer Ebene in trans Stellung zu der ersten Bindungsstelle der Zelle des From-Atoms angeordnet wird.

In Tabelle 3-1 ist eine Übersicht über die standardmäßig bei der Verknüpfung zweier Zellen verwendeten Dihedralwinkel dargestellt.

Zelltyp From-Atom	Zelltyp Add-Atom	Bindungsstelle From-Atom	Position Add-Atom	Orientierung	Position From-Atom
Trigonal	Trigonal	2	2	trans	1
		3	2	cis	1
	Tetrahedral	2	2	trans	1
		3	2	cis	1
	Pentagonal	2	3	trans	1
		3	3	cis	1
	Pentagonal (axial)	2	5	trans	1
		3	5	cis	1
	Octahedral	2	3	trans	1
		3	3	cis	1
Tetrahedral	Trigonal	2 - 4	2	trans	1
	Tetrahedral	2 - 4	2	trans	1
	Pentagonal	2 - 4	3	trans	1
	Pentagonal (axial)	2 - 4	5	trans	1
	Octahedral	2 - 4	3	trans	1
Pentagonal	Trigonal	2 - 5	2	trans	1
	Tetrahedral	2 - 5	2	trans	1
	Pentagonal	2 - 5	3	trans	1
	Pentagonal (axial)	2 - 5	5	trans	1
	Octahedal	2 - 5	3	trans	1
Pentagonal (axial)	Trigonal	2	2	trans	3
		3 - 5	2	trans	1
	Tetrahedral	2	2	trans	3
		3 - 5	2	trans	1
	Pentagonal	2	3	trans	3
		3 - 5	3	trans	1
	Pentagonal (axiall)	2	5	trans	3
		3 - 5	5	trans	1
	Octahedral	2	3	trans	3
		3 - 5	3	trans	1
Octahedral	Trigonal	2	2	trans	3
		3 - 6	2	trans	1
	Tetrahedral	2	2	trans	3
		3 – 6	2	trans	1
	Pentagonal	2	3	trans	3
		3 – 6	3	trans	1
	Pentagonal (axial)	2	5	trans	3
		3 – 6	5	trans	1
	Octahedral	2	3	trans	3
		3 – 6	3	trans	1

Tabelle 3-1: Standardwerte des Dihedralwinkels zwischen verbundenen Zellen

In der zuvor abgebildeten Tabelle 3-1 ist die Verknüpfung der existierenden Zelle des From-Atoms mit der Zelle des Add-Atoms dargestellt. Das Add-Atom wird mit der ersten Bindungsstelle der Zelle an die durch den Positionsindex spezifizierte Bindungsstelle des From-Atoms gebunden. Die Begriffe „Position Add-Atom“ und „Position From-Atom“ bezeichnen jeweils die Bindungsstelle der Zelle des Add-Atoms und die Bindungsstelle der Zelle des From-Atoms. Diese Bindungsstellen sind zusammen mit der Bindung zwischen den Zellen in einer Ebene angeordnet. Durch die Begriffe „cis“ und „trans“ wird die räumliche Stellung der Bindungsstellen der Zelle des Add-Atoms zu der Bindungsstelle der Zelle des From-Atoms, hinsichtlich der Bindung zwischen den Zellen, festgelegt.

3.4 Codierungsregeln der SMILES - 3D Notation

In den folgenden Unterkapiteln werden die weiterentwickelten und modifizierten Regeln zur Codierung der dreidimensionalen Molekülstruktur, basierend auf dem ursprünglichen SMILES Konzept, dargestellt und im Detail erläutert.

3.4.1 Codierung und Spezifikation von Atomen

Die grundlegenden Elemente bei der Codierung eines Moleküls sind die Atome und deren Anordnung innerhalb der Molekülstruktur. Darüber hinaus ist bei der Codierung von dreidimensionalen Strukturen die räumliche Anordnung der einzelnen Valenzorbitale der Atome und die Anzahl der Bindungen der Atome von entscheidender Bedeutung.

Die Atome eines Moleküls werden in der linearen Zeichenfolge durch die IUPAC Atomsymbole dargestellt. Atomsymbole die aus einem Buchstaben bestehen werden in der SMILES - 3D Notation durch einen großen Buchstaben, und Atomsymbole die aus zwei Buchstaben bestehen werden durch einen ersten großen Buchstaben und einen zweiten kleinen Buchstaben codiert.

Die räumliche Umgebung eines Atoms ist durch die Anzahl und die Orientierung der Valenzorbitale festgelegt. Jedem Atom wird daher ein Standardwert für die Anzahl der Valenzorbitale zugewiesen und in Abhängigkeit von diesem Wert jedem Atom ein geometrischer Zelltyp zugeordnet, der die räumliche Orientierung der Orbitale repräsentiert. Eine Übersicht über die vordefinierten Standardwerte der Anzahl der Valenzorbitale und der hieraus resultierenden Zelltypen der virtuellen Atome ist in Kapitel 8.6 im Anhang dargestellt. Des weiteren wird der kovalente Radius eines Atoms als der Standardwert zur Ermittlung der Bindungslänge zwischen zwei Atomen definiert.

In der linearen SMILES - 3D Notation werden die Wasserstoffatome in der Regel nicht explizit codiert. Alle freien Bindungsstellen eines Atoms werden

automatisch mit einfach gebundenen Wasserstoffatomen besetzt. Die Wasserstoffatome können optional in der linearen Zeichenfolge zur selektiven Zuweisung an eine Bindungsstelle eines Atoms angegeben werden.

Die Eigenschaften eines Atoms können in der Molekülstruktur von den vordefinierten Standardwerten eines Atoms abweichen. Es ist daher erforderlich, die Atome in der linearen Notation entsprechend zu spezifizieren. Die Spezifikation eines Atoms erfolgt selektiv und optional in einer vordefinierten standardisierten Form für jedes Atom. Hierdurch wird die einheitliche Darstellung der Atome in der linearen Notation erreicht und hieraus resultierend die Lesbarkeit verbessert, sowie die Datenverarbeitung der linearen Zeichenfolge optimiert. Die standardisierte Form der optionalen Spezifikationsmöglichkeiten eines Atoms ist, wie nachfolgend in Gl. 1 dargestellt, definiert.

[<*mass*> ***atsymb*** <*val*> <*fcharge*> <*oxnum*> <*a*>] (Gl. 1)

Die Spezifikation eines Atoms wird in der linearen Notation mit eckigen Klammern eingefasst. Die eckigen Klammern sind nur bei der optionalen Spezifikation eines Atoms erforderlich. Die Zeichen < und > sind keine Zeichen der SMILES - 3D und sind hier lediglich zur Kennzeichnung der einzelnen optionalen Spezifikationsmöglichkeiten eines Atoms ergänzt worden.

Zur Codierung eines Atoms ist lediglich das Atomsymbol erforderlich, jede weitere Spezifikation ist optional. Die einzelnen optionalen Spezifikationsmöglichkeiten eines Atoms werden in der linearen Notation durch mindestens ein Leerzeichen von einander getrennt. Im folgenden werden die Abkürzungen der Spezifikationsmöglichkeiten in Gl. 1 erläutert.

<*mass*> Atommasse (Numerisch, Dezimalzahl)
Die Atommasse bezeichnet die von dem normalen Atomgewicht abweichende Masse eines Atoms (z. B. Isotopenmasse) und wird direkt vor dem Atomsymbol spezifiziert. Dezimalwerte werden mit einem Dezimalpunkt eingegeben.

atsymb Atom- oder Variablensymbol (Alphanumerischer Wert)
Die Angabe eines Atomsymbols zur Codierung eines Atoms oder der Name einer Variablen zur Codierung eines Molekülfragmentes ist notwendig und beginnt immer mit einem großen Buchstaben.

<*val*> Anzahl der Valenzorbitale (Natürliche Zahl)
Die Spezifikation der Anzahl der Valenzorbitale erfolgt direkt nach dem Atomsymbol durch die explizite Angabe der Anzahl der Valenzorbitale als natürliche Zahl. Die Valenzelektronen des Atoms werden auf die spezifizierte Anzahl der Orbitale des Atoms entsprechend der Regeln des VSEPR Modells verteilt und die geometrische Anordnung der Bindungsstellen und der besetzten Orbitale ermittelt.

<*fcharge*> Formale Atomladung
Die formale Ladung eines Atoms bezeichnet die Ladung die ein Atom in der Lewis Struktur trägt. Die formale Ladung eines Atoms wird durch das Symbol + für eine positive Ladung, oder das Symbol - für eine negative Ladung dargestellt. Die mehrfache Ladung eines Atoms wird durch mehrere aufeinanderfolgende Ladungssymbole, oder durch die Multiplikation des Ladungssymbols durch das Zeichen * und eine nachfolgende Zahl, dargestellt. Unter Berücksichtigung der formalen Ladung eines Atoms wird die Anzahl der Valenzelektronen des spezifizierten Atoms festgelegt.

<*oxnum*> Oxidationszahl
Die Oxidationszahl eines Atoms wird durch die römischen Zahlen, bestehend aus den Buchstaben I, V und X, dargestellt.

<*a*> Axiale Bindungsstelle
Der Buchstabe a kennzeichnet, dass die erste Bindungsstelle des spezifizierten Atoms die axiale Position der pentagonalen

Zelle ist. Die axiale Bindungsstelle der pentagonalen Zelle des spezifizierten Atoms wird, unter Berücksichtigung der Bindungsverhältnisse in der linearen SMILES - 3D Notation, an die entsprechende freie Bindungsstelle des From-Atoms gebunden. Existiert in der linearen Notation vor dem spezifizierten Atom kein From-Atom, so wird das in der linearen Zeichenfolge nachfolgende Add-Atoman die axiale Bindungsposition des spezifizierten Atoms gebunden.

3.4.2 Codierung und Spezifikation von Bindungen

Zur Darstellung einer Einfach-, Doppel- oder Dreifachbindung zwischen zwei Atomen werden in der linearen Notation die Bindungssymbole -, = oder # verwendet. Die Verwendung des Bindungssymbols – ist optional, da die explizite Codierung einer Einfachbindung nicht erforderlich ist und zwei in der linearen Notation aufeinanderfolgende Atome, die freie Bindungsstellen besitzen, standardmäßig durch eine Einfachbindung miteinander verbunden werden. Des weiteren wird durch das Bindungssymbol : eine konjugierte Bindung und durch den Punkt eine nicht vorhandene Bindung gekennzeichnet.

Bei der Codierung von dreidimensionalen Strukturen ist es sinnvoll, gegebenenfalls die Eigenschaften einer Bindung (z. B. die Bindungslänge) zu modifizieren und die Bindung direkt zu spezifizieren. Die optionale Spezifikation einer Bindung erfolgt in der linearen Notation in der in Gl. 2 dargestellten standardisierten Form.

[<*dihang*> <***bosymb***> <*bondle*>] (Gl. 2)

Die Spezifikation einer Bindung wird in der linearen Notation mit eckigen Klammern eingefasst. Die Zeichen [und] sind nur bei der optionalen Spezifikation des Dihedralwinkels oder der Bindungslänge einer Bindung erforderlich. Die Zeichen < und > sind keine Zeichen der SMILES - 3D Notation und sind hier lediglich zur Kennzeichnung der einzelnen optionalen Spezifikationsmöglichkeiten einer Bindung ergänzt worden.

Im folgenden werden die Spezifikationsmöglichkeiten einer Bindung detailliert erläutert. Die Angabe der einzelnen Spezifikationen ist optional, lediglich die Angabe des Bindungssymbols ist notwendig, wenn der Dihedralwinkel oder die Bindungslänge der Bindung optional spezifiziert wird.

<*dihang*> Dihedralwinkel (Dezimalzahl)
Der dihedrale Winkel einer Bindung bezeichnet die Drehung der Zelle des Add-Atoms im Uhrzeigersinn um die Bindungsachse mit Blickrichtung zum From-Atom. Die Angabe des Dihedralwinkels erfolgt in Grad. Dezimalwerte werden unter Verwendung des Dezimalpunktes angegeben.

<***bosymb***> Bindungssymbol (-, =, #, :, .)
Die Angabe des Bindungssymbols ist bei der optionalen Spezifikation des Bindungstyps, mit Ausnahme von Einfachbindungen, notwendig und bezeichnet die zu spezifizierende Bindung.

<*bondle*> Bindungslänge (Dezimalzahl)
Die Angabe der Bindungslänge erfolgt nach dem Bindungssymbol in der Längeneinheit pm. Dezimalwerte werden unter Verwendung des Dezimalpunktes angegeben.

3.4.3 Codierung von verzweigten Strukturen und Seitenketten

Die Seitenketten und Verzweigungen in einer Molekülstruktur werden in der linearen Notation durch die Zeichen (und) gekennzeichnet. Die Atome und Bindungen der Seitenkette oder der verzweigten Molekülstruktur werden entsprechend der Regeln zur Codierung der Atome und der Bindungen codiert und durch die lineare Zeichenfolge eingefasst mit runden Klammern dargestellt. Die Bindung der Seitenkette kann optional vor der öffnenden runden Klammer spezifiziert werden.

3.4.4 Codierung von cyclischen Molekülstrukturen

Zur Codierung einer cyclischen Struktur durch eine lineare Zeichenfolge werden die Ringsysteme des Moleküls durch Ringöffnung in eine lineare Struktur überführt und umgekehrt durch Ringschluss aus einer linearen Notation eine cyclische Molekülstruktur generiert.

Bei der Codierung einer cyclischen Molekülstruktur wird zunächst formal eine Bindung je Ring gespalten und so die cyclische Struktur durch die Ringöffnung auf eine lineare Struktur reduziert. Die formal gespaltene Ringbindung wird mit einer Zahl eindeutig nummeriert und anschließend werden die Atome und Bindungen der formal linearen Molekülstruktur entsprechend der Notationsregeln codiert. Die Bindungsstellen der formal gespaltenen Ringbindung werden entsprechend der Position der Bindungsstelle an dem Atom in der SMILES - 3D Notation direkt an dem Atomsymbol durch das Zeichen % und dem numerischen Index der gespaltenen Bindung gekennzeichnet. In Abbildung 3-6 ist das Prinzip der Codierung von cyclischen Strukturen am Beispiel der schrittweisen Codierung von Benzol veranschaulicht.

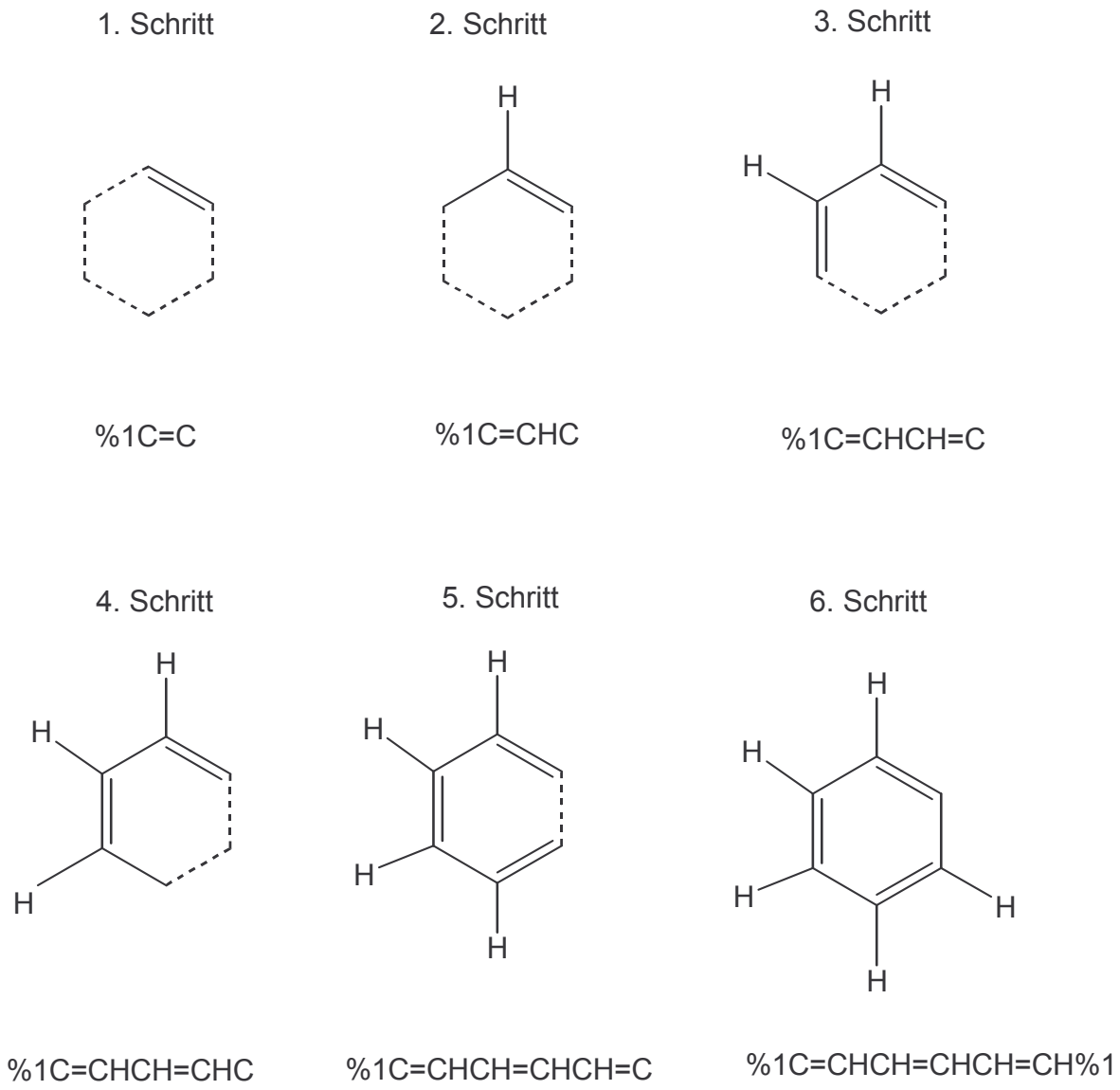
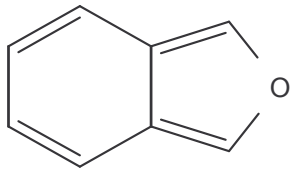


Abbildung 3-6: Schrittweise Codierung von Benzol

Zur Generierung der cyclischen Molekülstruktur aus der linearen Notation wird das Atom einer gespaltenen Ringbindung mit dem nächsten Ringatom in der linearen Notation, dessen Bindungsstelle durch den gleichen numerischen Index gekennzeichnet ist, verbunden.

Innerhalb der linearen Notation sind die Ringatome, die durch die gleiche Bindungsnummer gekennzeichnet sind und die in der linearen Zeichenfolge in Lesrichtung von links nach rechts als nächstes aufeinander folgen, die Atome die durch die gleiche Ringbindung miteinander verbunden sind. Diese Definition ermöglicht die Mehrfachverwendung von Zahlen zur Nummerierung von

Ringbindungen in der linearen Notation von multicyclischen Strukturen und ist nachfolgend am Beispiel der Codierung von Isobenzofuran veranschaulicht.



SMILES - 3D %1OC=C(C=CHCH=CH%2)C%2=CH%1

Abbildung 3-7: Codierung von Isobenzofuran

Der Typ der gespaltenen Bindung wird bei der Ringöffnung vor dem Prozentzeichen des in der linearen Notation zweiten Atoms der gespaltenen Ringbindung durch die Angabe des Bindungssymbols festgelegt und gegebenenfalls in eckigen Klammern optional spezifiziert. In multicyclischen Systemen, in denen ein Atom eventuell in mehrere Ringsysteme eingebunden ist, wird jede Ringbindung nach der zuvor beschriebenen Methode codiert.

3.4.5 Codierung von Molekülfragmenten

Bei der Codierung von Molekülen können vordefinierte und bereits codierte Molekülfragmente, sogenannte Templates, zur Codierung verwendet und in die lineare SMILES - 3D Notation eines Moleküls integriert werden. Das Molekülfragment ist eine beliebige dreidimensionale Struktur eines Moleküls oder einer molekularen Teilstruktur, die entsprechend der Notationsregeln durch eine lineare Zeichenfolge codiert wird. Der linearen Notation des Molekülfragments wird ein Name zur Kennzeichnung und zur weiteren Integration des Templates in andere Moleküle zugewiesen.

Die Definition eines Templates ist durch die Zeichenfolge := und den in Gl. 3 dargestellten Syntax in der Notation realisiert.

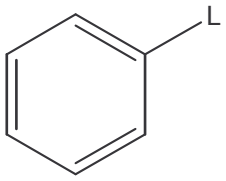
<*Templatename*> := <*SMILES String*> (Gl.3)

Die Zeichen < und > sind keine Zeichen der SMILES - 3D und sind hier lediglich zur Kennzeichnung des Syntax ergänzt worden. Nachfolgend werden die Abkürzungen in Gl. 3 erläutert.

<*Templatename*>
Der Name des Templates besteht aus einer Zeichenfolge die entweder mit einem großen Buchstaben oder dem Zeichen _ und einem nachfolgendem großen Buchstaben beginnt. Die Zeichenfolge darf maximal einen großen Buchstaben enthalten und der gesamte Templatename darf nicht mit den Zeichen eines Atomsymbols übereinstimmen.

<SMILES String>
Die lineare SMILES - 3D Notation des codierten Molekülfragments.

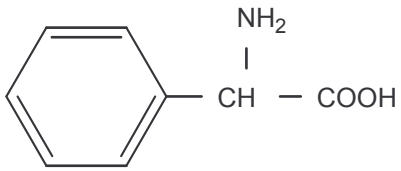
Die erste Bindungsstelle des ersten Atoms der linearen Zeichenfolge des codierten Templates wird standardmäßig durch einen terminalen Liganden zur weiteren Integration des Fragmentes freigehalten. Des weiteren können in der linearen Zeichenfolge die freien Bindungsstellen des Molekülfragments selektiv durch die Verwendung des Buchstaben L zur Kennzeichnung eines terminalen Liganden dargestellt werden. Bei mehreren freien Bindungsstellen eines Molekülfragments wird durch eine, dem Buchstaben L nachfolgende, optionale Zahl die Reihenfolge der freien Bindungsstellen explizit festgelegt. In Abbildung 3-8 ist die Definition eines Templates am Beispiel der Codierung der Phenylgruppe dargestellt.



Phe:=LC%1=CCH=CHCH=CHCH=%1

Abbildung 3-8: Definition der Phenylgruppe als Template

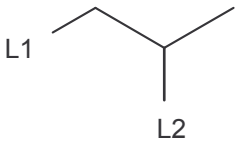
Zur Integration eines definierten Templates in die lineare Notation eines Moleküls wird der Name des Templates in die Zeichenfolge eingebunden und das Template mit der freien Bindungsstelle entweder an die freie Bindungsstelle des bereits vorhandenen From-Atoms gebunden, oder die freie Bindungsstelle des Templates mit der freien Bindungsstelle des in der Zeichenfolge nachfolgenden Add-Atoms besetzt. Die Integration des zuvor in Abbildung 3-8 definierten Templates in die lineare Notation der Molekülstruktur der Aminosäure Phenylalanin ist in Abbildung 3-9 gezeigt.



SMILES: PheC(N)C=OO

Abbildung 3-9: Integration des Templates in die lineare Notation

Die Definition eines Templates mit zwei freien Bindungsstellen zur weiteren Integration des Fragmentes in eine Molekülstruktur ist nachfolgend in Abbildung 3-10 veranschaulicht. In der dargestellten vordefinierten Molekülstruktur ist die erste Bindungsstelle des in der Notation ersten Kohlenstoffatoms und die zweite Bindungsstelle des in der Zeichenfolge dritten Kohlenstoffatoms jeweils mit einem Liganden L zur weiteren Integration des Templates freigehalten. Die lineare Notation weist dem Template den Namen "Prop" zu.



Prop:=L1CCL2C

Abbildung 3-10: Definition eines Templates mit zwei freien Bindungsstellen

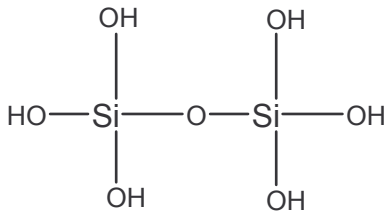
Die unterschiedlichen Integrationsmöglichkeiten des zuvor in Abbildung 3-10 definierten Templates mit zwei Liganden ist nachfolgend in Abbildung 3-11 gezeigt. Mehrere freie Bindungsstellen eines Templates werden in der Reihenfolge der Nummerierung der Bindungsstellen mit den in der linearen Zeichenfolge codierten Atomen der Molekülstruktur verbunden.

OPropC CPropO

Abbildung 3-11: Integration eines Templates mit zwei freien Bindungspositionen

3.4.6 Verwendung von Multiplikatoren

In der linearen SMILES - 3D Notation können Zeichenfolgen, zum Beispiel Atome oder Templates, die sich in der linearen Notation nacheinander wiederholen durch die Verwendung eines Multiplikators codiert werden. Als Multiplikator wird in der Notation das Zeichen * und eine nachfolgende Zahl, die Anzahl der wiederholenden Einheiten, verwendet. In der linearen Notation wird die Einheit bestehend aus einem Atom, aus einer molekularen Teilstruktur die durch runde Klammern eingefasst ist oder aus einem Template, die unmittelbar vor dem Multiplikatorzeichen steht entsprechend oft ersetzt. In Abbildung 3-12 ist die Codierung des Dimeren der Kieselsäure unter Verwendung des Multiplikatorzeichens dargestellt.



SMILES: O(Si(O)(O)(O))*2

Abbildung 3-12: Anwendung des Multiplikators bei der Codierung von Dikieselsäure

4. Die Software WinSmiles - 3D

4.1 Das Anforderungsprofil an die Software WinSmiles - 3D

Ein wesentlicher Bestandteil der vorliegenden Arbeit ist die Umsetzung des in Kapitel 3 vorgestellten entwickelten SMILES - 3D Konzepts in einer anwenderorientierten Software.

Die grundlegenden Anforderungen an die Software WinSmiles - 3D sind daher zum einen die Visualisierung der linearen SMILES - 3D Notation durch die grafische dreidimensionale Darstellung der codierten Molekülstruktur, sowie die Berechnung der Atomkoordinaten, und zum anderen die Generierung der Linearen SMILES - 3D Zeichenfolge aus der grafischen Darstellung der Molekülstruktur. Des weiteren sollen durch die Software Informationen, die direkt oder indirekt in der linearen Zeichenfolge codiert sind, dem Anwender durch die Software leicht zugänglich gemacht werden. Zu den direkten Informationen gehören zum Beispiel die explizit in der Zeichenfolge codierte Atommasse oder die explizit codierte Bindungslänge, darüber hinaus sind in der linearen SMILES - 3D Notation indirekte Informationen wie die räumliche Anordnung der einzelnen Atome in einem Molekül oder die Gesamtmasse eines codierten Moleküls enthalten.

Die wesentlichen Anforderungen an die weiterentwickelte Software sind in der nachfolgenden Aufzählung punktuell angeführt.

- Effektive Interpretation der linearen SMILES - 3D Notation
- Automatische Generierung der SMILES - 3D Notation auf Basis der Molekülstruktur
- Dreidimensionale Visualisierung der Molekülstruktur
- Selektive Modifikation der Atomeigenschaften
- Selektive Modifikation der Bindungseigenschaften
- Verwendung des Programmes auf handelsüblichen PC unter Windows Betriebssystemen
- Grafische Benutzeroberfläche (GUI)

Das SMILES - 3D Konzept ist, unter Berücksichtigung des zuvor dargestellten Anforderungsprofils, in der Softwareentwicklung umgesetzt und das Programm WinSmiles - 3D konzipiert worden. In den folgenden Kapiteln wird die Konzeption der Software und die programmtechnische Realisierung beschrieben.

4.2 Programmtechnische Entwicklungsvoraussetzungen

Die Software WinSmiles - 3D ist, basierend auf dem UNIX –Programm SMILES, im Rahmen dieser Arbeit modifiziert und weiterentwickelt worden. Die Software ist speziell für handelsübliche Rechner, die Windows Betriebssysteme verwenden, konzipiert und auf einem Rechner mit Pentium II Prozessor mit 256 MB RAM, sowie dem Betriebssystem Windows 98 second Edition entwickelt und getestet worden. Die entwickelte Software sollte auf neueren, leistungsstärkeren Rechnern noch effizienter laufen.

Das Programm WinSmiles - 3D wurde unter Verwendung der objektorientierten Programmiersprache C++ entwickelt und die interne Programmstruktur entsprechend der objektorientierten Anforderungen konzipiert. Der Quellcode der Software ist mit dem Programm Visual C++ 6.0 aus der Microsoft Entwicklungsumgebung Visual Studio Pro 6.0 editiert und strukturiert worden[59] Des weiteren wurden mit dem Programm die grafischen Benutzeroberflächen der Anwendung generiert und der gesamte Quellcode der Software kompiliert. Zur Erzeugung von dreidimensionalen Bildern ist die von Silicon Graphics [60] entwickelte und von Microsoft implementierte Software OpenGL in das Programm integriert worden.

Die Informationen und Dokumentationen zur technischen Programmentwicklung sind der Microsoft Developer Network Library 6.0 (MSDN) entnommen.[61]

4.3 Konzeption der Software WinSmiles - 3D

4.3.1 Die funktionellen Objekte der Software WinSmiles - 3D

In der objektorientierten Programmstruktur der weiterentwickelten Software WinSmiles - 3D wird ein Molekül durch eine virtuelle Molekülstruktur dargestellt. Die virtuelle Molekülstruktur ist ein Objekt vom Typ Mol das in der Headerdatei mol.h definiert und in der Quellcodedatei mol.cpp modifiziert ist. Das Objekt vom Typ Mol realisiert den Aufbau einer Molekülstruktur und bildet das Interface zwischen der linearen SMILES - 3D Notation und der virtuellen Molekülstruktur. Des weiteren sind in das Objekt vom Typ Mol die Methoden zur Serialisierung und zur Visualisierung des Moleküls implementiert.

In Analogie zum Aufbau eines chemischen Moleküls sind die Atome und Bindungen des codierten Moleküls in der virtuellen Molekülstruktur durch die virtuellen Atome und die virtuellen Bindungen repräsentiert. In der Programmstruktur sind die virtuellen Atome Objekte vom Typ Atom, sowie die virtuellen Bindungen Objekte vom Typ Bond. Die Objektstruktur der Atome und Bindungen ist in den Klassen Atom und Bond definiert. In die Headerdatei Smiles.h und die Quellcodedatei Smiles.cpp ist die Funktionalität der Objekte vom Typ Atom und Bond implementiert. Die Definition der Klasse Atom realisiert Methoden zur Erzeugung virtueller Atome und generiert Membervariablen zur Darstellung der Eigenschaften des Atoms. Des weiteren verfügt das Objekt vom Typ Atom über Methoden zur Serialisierung und zur Visualisierung des virtuellen Atoms.

Das Objekt vom Typ Bond generiert Methoden zur Erzeugung der virtuellen Bindung und erzeugt Membervariablen zur Steuerung der Bindungseigenschaften. Darüber hinaus verfügt das Objekt über Methoden zur Visualisierung der Bindung und zur Serialisierung.

Die Generierung der dreidimensionalen Struktur des Moleküls wird, entsprechend des SMILES - 3D Konzepts, durch die Verwendung von virtuellen Zellen realisiert. Die virtuellen Zellen sind in der Programmstruktur Objekte vom Typ Cell. Die Klasse Cell ist in der Headerdatei Smiles.h definiert und die Funktionalität in

der Quellcodedatei Smiles.cpp modifiziert. Die virtuelle Zelle verfügt über Methoden zur Erzeugung der unterschiedlichen Zelltypen und zur Serialisierung.

In den nachfolgenden Abschnitten wird die Konzeption und die Funktionalität der Objekte vom Typ Atom, Bond, Cell und Mol beschrieben. Darüber hinaus enthält der Anhang in Kapitel 8 eine detaillierte, systematische programmtechnische Erläuterung der Objekte.

4.3.2 Konzeption des virtuellen Atoms – Objekt vom Typ Atom

Das Objekt vom Typ Atom wird durch den Aufruf des Konstruktors der Klasse Atom erzeugt. Die Membervariablen und Methoden eines virtuellen Atoms sind so konzipiert, dass jede Membervariable eine Analogie zu einer Eigenschaft eines Atoms bildet und die Methoden des Objektes vom Typ Atom die Eigenschaften des virtuellen Atoms ermitteln und verändern.

Ein wesentliches Kennzeichen, um Atome bei der Darstellung zu unterscheiden, ist das Atomsymbol, welches entsprechend der IUPAC Regeln aus ein oder zwei Buchstaben besteht und in einem virtuellen Atom in der Membervariablen m_sym gespeichert wird. Des weiteren ist die Ordnungszahl eines Atoms ein eindeutiges Kennzeichen zur Unterscheidung von Atomen. Diese wird in der Membervariablen m_oz eines Objektes vom Typ Atom gespeichert. Zur Unterscheidung von Isotopen ist die Masse eines Atoms erforderlich, deshalb wird die Masse eines virtuellen Atoms in der Membervariablen m_mass gespeichert.

Das Vorhandensein von Elektronen in Atomen ist eine Voraussetzung zur Bildung von Bindungen zwischen Atomen und somit notwendig für die Existenz von Molekülen. In einem Objekt vom Typ Atom werden die Valenzelektronen eines Atoms durch die Membervariable m_ele dargestellt, die den Wert der Anzahl der Valenzelektronen enthält. Die Oxidationsstufe und Ladung eines Atoms sind weitere wichtige Eigenschaften, welche die Bindungsverhältnisse eines Atoms bestimmen. In einem virtuellen Atom ist der Wert der Oxidationsstufe in der Membervariablen m_oxistate und der Wert der formalen Ladung eines Atoms in der Membervariablen m_charge gespeichert. Neben der Anzahl der Elektronen, der formalen Ladung und

der Oxidationsstufe eines Atoms ist die Anzahl der Valenzen eine wichtige Eigenschaft, welche die räumliche Struktur eines Atoms bestimmt. Der Wert der Anzahl der Valenzen eines virtuellen Atoms ist in der Membervariablen m_bind gespeichert.

Zur räumlichen Beschreibung von Molekülen ist zum einen die Position eines Atoms und zum andern die räumliche Orientierung der Bindungsstellen eines Atoms notwendig. Die Werte der Position eines virtuellen Atoms sind in den Membervariablen m_x, m_y und m_z gespeichert. Die räumliche Orientierung der Bindungsstellen eines Atoms ist in der Membervariablen m_cell realisiert. Die Membervariable m_cell ist ein Objekt vom Typ Cell, das die Bindungsstellen der unterschiedlichen Zelltypen enthält.

In einem Molekül sind die einzelnen Atome durch Bindungen miteinander verbunden. Die Bindungen eines virtuellen Atoms sind durch die einzelnen Elemente der Membervariablen m_bond realisiert. Die Membervariable m_bond ist ein Objekt vom Typ Bond, das entsprechend des Wertes der globalen Variablen MAX_BOND Elemente bzw. virtuelle Bindungen hat. Des weiteren sind die einzelnen Atome eines virtuellen Moleküls durch eine Referenzverknüpfung miteinander verbunden. Jedes virtuelle Atom enthält sowohl eine Referenz auf das zuvor erzeugte virtuelle Atom, als auch eine Referenz auf das nachfolgend erzeugte Atom. Die Referenzverknüpfung der virtuellen Atome ist in den Membervariablen m_next und m_prev realisiert. Darüber hinaus werden die einzelnen virtuellen Atome entsprechend der Reihenfolge ihrer Erzeugung nummeriert. Die Nummerierung wird in der Membervariablen m_no des Objektes vom Typ Atom gespeichert.

Virtuelle Atome die aus einer vordefinierten Molekülstruktur, einem Template, erzeugt und in eine neue Molekülstruktur integriert sind, werden in der Membervariablen m_varname mit dem Namen des Templates gekennzeichnet, sowie in der Membervariablen m_varcount der Index gleicher Templates innerhalb eines Moleküls gespeichert.

Bei der Definition eines Templates werden die freien Bindungsstellen der vordefinierten Molekülstruktur durch einen terminalen Liganden erzeugt. Der

terminale Ligand ist ein virtuelles Atom, das durch das Atomsymbol „L“ und durch die Nummerierung der Liganden innerhalb einer Molekülstruktur gekennzeichnet ist. Der Index eines virtuellen Liganden wird in der Membervariablen m_noligand des Objektes vom Typ Atom gespeichert.

Nachdem die virtuelle Molekülstruktur aus dem eingegebenen SMILES String erzeugt ist, werden die freien Bindungsstellen der Atome mit virtuellen Wasserstoffatomen besetzt. Diese automatisch generierten Wasserstoffatome sind Objekte vom Typ Atom und werden in der boolschen Membervariablen m_isauto mit dem Wert true gekennzeichnet.

Die Erzeugung eines virtuellen Atoms erfolgt durch den Aufruf des Konstruktors der Klasse Atom, der als Parameter die Referenz auf eine Atomtabelle benötigt und zusätzlich optional die Referenz auf ein virtuelles Atom enthalten kann. Bei der Konstruktion werden den Membervariablen, die die Eigenschaften des Atoms repräsentieren, die entsprechenden Werte aus der Atomtabelle zugewiesen und gegebenenfalls die Referenzverknüpfung der Objekte vom Typ Atom zu dem optionalen Parameter realisiert. Die Atomtabelle ist ein Objekt vom Typ AtomTable, das die vordefinierten Standardwerte der Atomeigenschaften enthält. Die Referenz auf die Atomtabelle wird in der Membervariablen m_at des erzeugten virtuellen Atoms gespeichert.

Neben der Erzeugung von virtuellen Atomen ist die Speicherung und das Lesen von Daten eines Objektes vom Typ Atom eine wichtige Funktionalität, um Daten dauerhaft verfügbar zu machen. Das virtuelle Atom hat zur Speicherung der Daten die Methode Write und zum Lesen der zuvor gespeicherten Daten die Methode Read.

Um den Aufbau einer virtuellen Molekülstruktur aus den Objekten vom Typ Atom effizient zu realisieren, ist es erforderlich die Eigenschaften eines virtuellen Atoms ermitteln und die Werte der entsprechenden Membervariablen verändern zu können. Die Funktion SetAtom eines Objektes vom Typ Atom verändert den Wert der Membervariablen in Abhängigkeit von dem verwendeten Parameter und die Funktion SetCell ändert die Position der Bindungsstellen.

Des weiteren sind für die Erzeugung einer flexiblen Molekülstruktur die Bindungsmöglichkeiten eines virtuellen Atoms von entscheidender Bedeutung. Das virtuelle Atom hat deshalb Methoden, um die Bindungsverhältnisse eines Atoms ermitteln und verändern zu können. Die Anzahl der freien Bindungsstellen eines virtuellen Atoms wird durch den Aufruf der Methode FreeBondCount ermittelt, sowie durch den Aufruf der Methode IsBondOk überprüft, ob das Atom freie Bindungsstellen hat.

Innerhalb eines Moleküls sind die einzelnen virtuellen Atome durch die Referenzverknüpfung der Objekte vom Typ Atom miteinander verbunden. Diese Referenzverknüpfung ermöglicht das gezielte und schnelle Ermitteln eines Atoms aus der virtuellen Molekülstruktur. Durch den Aufruf der Methode First bzw. Last wird die Referenz ermittelt, die auf das erste bzw. letzte Atom eines Moleküls verweist.

Weiterhin ist die Nummerierung der einzelnen virtuellen Atome innerhalb eines Moleküls ein eindeutiges Kennzeichen zur gezielten Ermittlung eines Atoms. Dieser Index wird durch den Aufruf der Methode Count aktualisiert. Durch den Aufruf der Methode GetAtom mit dem entsprechenden Index als Parameter kann gezielt ein Atom ermittelt werden.

Bei der Visualisierung einer virtuellen Molekülstruktur werden die einzelnen Objekte, welche die Molekülstruktur aufbauen, visualisiert. Dies wird durch den Aufruf der Methode Draw des Objektes vom Typ Atom realisiert.

In der weiterentwickelten Software ist die automatische Generierung des SMILES String aus einer virtuellen Molekülstruktur realisiert. Diese Funktionalität ist in das Objekt vom Typ Atom implementiert.

Bei der Umsetzung der virtuellen Molekülstruktur in den linearen SMILES String ist es erforderlich, die automatisch generierten Wasserstoffatome aus der Molekülstruktur zu entfernen. Dies erfolgt durch den Aufruf der Methode StripH. Die Erzeugung des SMILES Strings aus der Molekülstruktur erfolgt durch den Aufruf der Methode Smiles des virtuellen Atoms.

Ein wesentliches Leistungsmerkmal der hier entwickelten Software ist die Integration von vordefinierten Molekülstrukturen, den Templates, in eine andere Molekülstruktur.

Die Integration eines Templates in eine virtuelle Molekülstruktur erfolgt durch die Erzeugung einer Kopie des Templates durch den Aufruf der Methode Copy und die Einbindung der Kopie in die vorhandene Molekülstruktur, sowie die anschließende Aktualisierung der Nummerierung der Atome innerhalb der virtuellen Molekülstruktur.

Ist die Existenz eines virtuellen Atoms nicht mehr erforderlich, so wird das Objekt vom Typ Atom durch den Destruktor zerstört. Bei der Destruktion wird die Methode delete des virtuellen Atoms aufgerufen und der allokierte Speicherplatz freigegeben.

Die Methoden und Membervariablen der Klasse Atom sind ergänzend im Anhang in Kapitel 8.1 im Detail beschrieben.

4.3.3 Konzeption der virtuellen Bindung – Objekt vom Typ Bond

Ein Objekt vom Typ Bond wird durch den Aufruf des Konstruktors der Klasse Bond erzeugt. Ein Objekt vom Typ Bond ist so konzipiert, dass die Werte der Membervariablen die Eigenschaften einer Bindung (z.B. Bindungslänge oder Bindungstyp) zwischen Atomen repräsentieren und die virtuelle Bindung Methoden zur Visualisierung und Serialisierung hat.

In einem Molekül können die einzelnen Atome durch unterschiedliche Bindungen miteinander verbunden sein. Die unterschiedlichen Bindungstypen sind durch den Wert der Membervariablen m_type der virtuellen Bindung charakterisiert. Mit dem Objekt vom Typ Bond kann eine Einfach-, Doppel- oder Dreifachbindung, sowie eine aromatische Bindung zwischen den virtuellen Atomen realisiert werden. Die aromatische Bindung wird durch eine alternierende Reihenfolge einer Einfachbindung und einer Doppelbindung zwischen den virtuellen Atomen der

aromatischen Struktur dargestellt. Die aromatische Bindung ist zusätzlich durch den Wert true in der Membervariablen m_aromatic gekennzeichnet.

Die Länge einer Bindung zwischen zwei Atomen ist durch die Eigenschaften der jeweiligen Atome bestimmt. In Analogie dazu wird die Länge einer virtuellen Bindung aus den entsprechenden Membervariablen der virtuellen Atome ermittelt und in der Membervariablen m_rad des Objektes vom Typ Bond gespeichert. Darüber hinaus ist es möglich, die Bindungslänge im SMILES String direkt zu spezifizieren, diese Bindungslänge wird ebenfalls in der Membervariablen m_rad gespeichert und zusätzlich mit dem Wert true in der Membervariablen m_radimput gekennzeichnet.

Des weiteren ist die räumliche Anordnung der Bindungsstellen zweier Atome zueinander, die durch eine Bindung miteinander verbunden sind, durch den dihedralen Winkel der Bindung festgelegt. Der Wert des Dihedralwinkels bestimmt die Rotation im Uhrzeigersinn um die Bindungsachse mit Blickrichtung zum From-Atom in Grad und ist in der Membervariablen m_rot der virtuellen Bindung gespeichert.

Neben Zuweisung des Bindungstyps und der Bindungslänge ist es wichtig, in dem Objekt vom Typ Bond die virtuellen Atome festzulegen zwischen denen die Bindung realisiert werden soll. Dies erfolgt durch Speicherung der Referenzen, die in den Membervariablen m_toatom bzw. m_fromatom der virtuellen Bindung auf das To-Atom bzw. das From-Atom verweisen. Darüber hinaus ist es erforderlich, die Bindungsstelle an dem virtuellen Atom eindeutig festzulegen. Deshalb werden die Bindungsstellen eines virtuellen Atoms nummeriert und der Wert der zu verwendenden Bindungsstelle des From-Atoms bzw. des To-Atoms in den Membervariablen m_tobond bzw. m_frombond der virtuellen Bindung gespeichert.

Im SMILES String sind die Atome, zwischen denen ein Ringschluss zur Erzeugung eines cyclischen Systems durch eine virtuelle Bindung erfolgt, durch das Zeichen „%“ und die nachfolgende Nummerierung der Bindung gekennzeichnet. Der Wert der Nummerierung wird in der Membervariablen m_prozent der virtuellen Bindung gespeichert, die den Ringschluss zwischen den Atomen realisiert.

Die gleiche Bindung zwischen zwei Atomen kann zum einen als Bindung vom From-Atom zum To-Atom, als Hinbindung, und zum andern als Bindung vom To-Atom zum From-Atom, als Rückbindung, beschrieben werden. In der Membervariablen m_backbond der virtuellen Bindung wird die Referenz auf die entsprechende Rückbindung gespeichert.

Der Bearbeitungszustand einer virtuellen Bindung wird durch ein Flag gekennzeichnet. Der Wert des Flags wird in der Membervariablen m_flag gespeichert.

Die Erzeugung einer virtuellen Bindung erfolgt durch den Aufruf des Konstruktors der Klasse Bond. Bei der Konstruktion des Objektes vom Typ Bond werden die Membervariablen initialisiert und die Methoden Write und Read zur Speicherung der Daten einer virtuellen Bindung, sowie die Methode Draw zur Visualisierung der Bindung erzeugt.

Ergänzend sind die Methoden und Membervariablen der Klasse Bond im Anhang in Kapitel 8.2 detailliert beschrieben.

4.3.4 Konzeption der virtuellen Zelle – Objekt vom Typ Cell

Das Objekt vom Typ Cell wird zur Generierung der dreidimensionalen Molekülstruktur verwendet. Die einzelnen Zelltypen sind so konzipiert, dass das Atom der Mittelpunkt der Zelle ist und die Geometrie der Zelle die räumliche Orientierung der Bindungsstellen des Atoms bestimmt. Die Erzeugung der dreidimensionalen Molekülstruktur erfolgt durch Verbinden der Bindungsstellen der einzelnen Zellen der entsprechenden Atome des Moleküls, unter Berücksichtigung der Reihenfolge der Bindungsstellen einer Zelle.

In der hier weiterentwickelten Software WinSmiles - 3D werden die Zelltypen MONOHEDRAL, DIHEDRAL, TRIGONAL, TETRAHEDRAL, PENTAGONAL und OCTAHEDRAL verwendet. Die räumliche Orientierung der Bindungsstellen ergibt sich aus der Anordnung der Eckpunkte des entsprechenden geometrischen Körpers und der Reihenfolge der Bindungsstellen der Zelle. Der Zelltyp eines Objektes vom

Typ Cell wird in der Membervariablen m_type und die Anzahl der Bindungsstellen der Zelle in der Membervariablen m_n gespeichert. Die Werte der Koordinaten der Bindungsstellen werden durch den Aufruf der Funktion SetCell des Objektes vom Typ Atom in Abhängigkeit vom Zelltyp initialisiert und in den Elementen der Membervariablen m_xyz der Zelle gespeichert. Die Berechnung der Koordinaten der Bindungsstellen in der Molekülstruktur erfolgt durch den Aufruf der Funktion SetCoord des Objektes vom Typ Mol.

Nachfolgend wird die Konzeption der verschiedenen Zelltypen detailliert beschrieben. Bei allen Zelltypen ist der Mittelpunkt der Zelle im Ursprung eines Koordinatensystems, die Position der Atome, platziert und alle Zellpunkte, die eine Bindungsstelle des Atoms darstellen, sind um den Wert 1 vom Ursprung des Koordinatensystems entfernt.

1) MONOHEDRAL

   Die monohedrale Zelle platziert das Atom im Koordinatenursprung und hat eine Bindungsstelle im Koordinatenursprung. Dieser Zelltyp wird zur Positionierung von terminalen Atomen, wie z.B. Wasserstoffatomen oder Liganden verwendet.

2) DIHEDRAL

   Die Zelle vom Typ DIHEDRAL hat zwei Bindungsstellen, die erste Bindungsstelle ist im positiven Bereich auf der x-Achse und die zweite Bindungsstelle im negativen Bereich auf der x-Achse. Dieser Zelltyp wird zur Positionierung der Bindungsstellen z. B. der Erdalkalimetalle oder der Positionierung der Bindungsstellen eines Kohlenstoffatoms mit zwei Doppelbindungen verwendet.

3) TRIGONAL

   Die trigonale Zelle hat drei Bindungsstellen die alle in der xy-Ebene angeordnet sind. Die erste Bindungsstelle ist auf der x-Achse und die zwei weiteren Bindungsstellen sind rotationsförmig um die z-Achse platziert. Der Winkel zwischen den Bindungsstellen beträgt 120°, die zweite Bindungsstelle hat einen positiven y-Wert, sowie die dritte

Bindungsstelle einen negativen y-Wert. Die trigonale Zelle wird z. B. zur Darstellung der Bindungsstellen der Atome der dritten Hauptgruppe oder zur Darstellung der Bindungsstellen eines Kohlenstoffatoms mit einer Doppelbindung verwendet.

4) TETRAHEDRAL

Die Zelle vom Typ TETRAHEDRAL hat vier Bindungsstellen, die tetraedrisch um den Koordinatenursprung angeordnet sind. Die erste Bindungsstelle ist auf dem positiven Teil der x-Achse und die drei weiteren Bindungsstellen sind rotationsförmig um den negativen Teil der x-Achse platziert. Die zweite Bindungsstelle hat einen negativen y-Wert, sowie einen positiven z-Wert. Die dritte Bindungsstelle ist in der xz-Ebene platziert. Die vierte Bindungsstelle hat positive y und z Koordinaten. Die tetraedrische Zelle wird z. B. zur Darstellung der Bindungsstellen eines Kohlenstoffatoms mit vier Einfachbindungen oder eines Stickstoffatoms mit drei Einfachbindungen verwendet.

5) PENTAGONAL

Die Zelle vom Typ PENTAGONAL hat fünf Bindungsstellen die trigonal-bipyramidal um den Koordinatenursprung angeordnet sind. Die erste Bindungsstelle einer pentagonalen Zelle kann sowohl axial, als auch äquatorial sein. In Abhängigkeit von der Position der ersten Bindungsstelle existieren zwei verschiedene pentagonale Zellen mit unterschiedlichen Reihenfolgen der Bindungsstellen.

Bei axialer Position der ersten Bindungsstelle werden die Bindungsstellen wie folgt positioniert. Die erste Bindungsstelle ist im positiven Teil und die zweite Bindungsstelle ist im negativen Teil auf der x-Achse platziert. Die drei weiteren Bindungsstellen sind rotationsförmig um die x-Achse in der yz-Ebene angeordnet. Die dritte Bindungsstelle hat negative y Koordinaten, sowie positive z Koordinaten. Die vierte Bindungsstelle hat negative y und z Werte. Die fünfte Bindungsstelle ist im positiven Teil auf der y-Achse platziert.

Bei äquatorialer Position der ersten Bindungsstelle werden die Bindungsstellen der pentagonalen Zelle wie folgt positioniert. Die erste Bindungsstelle ist im positiven Teil auf der x-Achse angeordnet. Die zweite Bindungsstelle ist im negativen Teil und die vierte Bindungsstelle im positiven Teil der z-Achse platziert. Die dritte und die fünfte Bindungsstelle sind, zusammen mit der ersten Bindungsstelle, in der xy-Ebene platziert. Die x Koordinate der dritten Bindungsstelle ist negativ und die y Koordinate positiv. Die fünfte Bindungsstelle hat negative x und y Werte.

Die pentagonale Zelle wird z. B. zur Darstellung der Bindungsstellen eines Phosphoratoms mit fünf Einfachbindungen verwendet.

6) OKTAHEDRAL

Die Zelle vom Typ OKTAHEDRAL hat sechs Bindungsstellen, die oktaedrisch um den Koordinatenursprung angeordnet sind. Die erste Bindungsstelle ist im positiven Teil und die zweite Bindungsstelle ist im negativen Teil auf der x-Achse angeordnet. Die vier weiteren Bindungsstellen sind in der yz-Ebene auf den Koordinatenachsen angeordnet. Die dritte Bindungsstelle ist im positiven Teil und die fünfte Bindungsstelle ist im negativen Teil auf der y-Achse platziert. Die vierte Bindungsstelle ist im positiven Teil und die sechste Bindungsstelle ist im negativen Teil auf der z-Achse platziert.

Die oktaedrische Zelle wird z. B. zur Darstellung der Bindungsstellen eines Schwefelatoms mit sechs Einfachbindungen verwendet.

Die Methoden und Membervariablen der Objekte vom Typ Cell sind im Anhang in Kapitel 8.3 im Detail beschrieben.

4.3.5 Konzeption der virtuellen Molekülstruktur – Objekt vom Typ Mol

Die virtuelle Molekülstruktur wird durch den Aufruf des Konstruktors der Klasse Mol erzeugt. In das Objekt vom Typ Mol ist die Funktionalität zur Analyse der linearen SMILES - 3D Zeichenfolge und die Funktionalität zur Generierung einer dreidimensionalen Struktur bestehend aus virtuellen Atomen, Bindungen und Zellen implementiert.

Zur Verarbeitung der linearen Notation des codierten Moleküls ist es sinnvoll, die Zeichenfolge zu speichern und gegebenenfalls zur weiteren Datenverarbeitung die Zeichenfolge mit einem Namen zu kennzeichnen. Das Objekt vom Typ Mol speichert den SMILES String in der Membervariablen m_smiles und einem optionalen Namen der Zeichenfolge durch den Aufruf der Methode SetName in der Membervariablen m_name.

Die Funktionalität zur Analyse der linearen Zeichenfolge ist in den Methoden DoParse und Lex realisiert. In Abhängigkeit von dem Analyseergebnis werden die virtuellen Atome, Bindungen und Zellen erzeugt, sowie die Eigenschaften der Atome und der Bindungen in Membervariablen des Objektes vom Typ Mol zwischengespeichert.

Um mit der Software WinSmiles - 3D einen flexiblen Anwendungsbereich zu realisieren, können die vordefinierten Standardwerte der Atome und der kovalenten Bindungsradien selektiv für die virtuelle Molekülstruktur bei der Objektkonstruktion bestimmt werden. Die Membervariablen m_atomtable und m_kovrad des Objektes vom Typ Mol verweisen auf die Objekte der zu verwendenden Standardwerte.

Bei der Erzeugung der virtuellen Molekülstruktur werden die einzelnen virtuellen Atome durch Referenzen und virtuelle Bindungen miteinander verbunden. Die Referenz auf die Atome ist in der Membervariablen m_atom des Objektes vom Typ Mol gespeichert. Die Bindung zwischen den Atomen wird durch die Funktion MakeBond der virtuellen Molekülstruktur generiert.

Neben der Analyse des SMILES String ist in das Objekt vom Typ Mol die Funktionalität zur Generierung der dreidimensionalen Struktur in die Methode SetUp implementiert. Die Koordinaten der virtuellen Atome werden durch die Methode SetCoord des Objektes vom Typ Mol berechnet

Zur Abbildung der Darstellung der virtuellen Molekülstruktur ist es erforderlich, die Größe des Moleküls zu ermitteln. Daher werden durch die Methode SetMinMaxCoord die minimalen und maximalen Koordinaten ermittelt und in den Membervariablen m_mincoord und m_maxcoord gespeichert. Die Methode Draw visualisiert die virtuelle Molekülstruktur und speichert die Referenz auf das erzeugte Bild in der Membervariablen m_bitmap der virtuellen Molekülstruktur.

Zur selektiven Ermittlung eines virtuellen Atoms, der verwendeten vordefinierten Standardwerte der Atome oder des Namens der virtuellen Molekülstruktur sind in die Klasse Mol die Methoden GetAtom, GetAtomtable und GetName implementiert.

Die Serialisierung des Objektes vom Typ Mol ist in den Methoden Read und Write realisiert.

Im Anhang in Kapitel 8.4 sind die Methoden und Membervariablen der Klasse Mol detailliert beschrieben.

4.4 Programmtechnische Realisation der Software WinSmiles - 3D

4.4.1 Die essentiellen Programmschritte der Software WinSmiles - 3D

In den nachfolgenden Abschnitten werden die wesentlichen Programmschritte der weiterentwickelten Software WinSmiles - 3D beschrieben, mit denen die Erzeugung eines Abbildes einer dreidimensionalen Moleküldarstellung aus dem SMILES String realisiert wird. Die Abfolge der essentiellen Programmschritte ist in Abbildung 4-1 schematisch dargestellt.

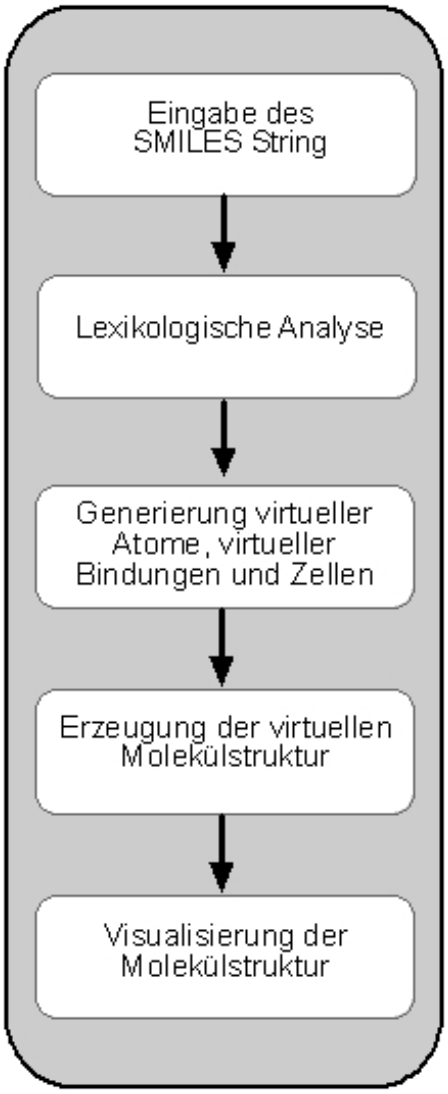


Abbildung 4-1: Schematische Darstellung der essentiellen Programmschritte

4.4.2 Die Eingabe des SMILES Strings

Die zu bearbeitende lineare SMILES - 3D Notation wird über die Tastatur in das entsprechende Eingabefeld der grafischen Benutzeroberfläche eingegeben und anschließend entweder durch das Betätigen der Returntaste oder durch das Aktivieren der Schaltfläche „New“ mittels der Nachrichtenbehandlungsroutinen in einer Variablen gespeichert. Diese Variable wird als Parameter im Konstruktor zur Erzeugung eines Objektes vom Typ Mol verwendet.

4.4.3 Die lexikologische Analyse des linearen SMILES Strings

In dem Programmschritt der lexikologischen Analyse werden die Zeichen der eingegebenen linearen SMILES - 3D Zeichenfolge analysiert, entsprechend der Stellung der Zeichen im SMILES String interpretiert und die hieraus resultierenden Objekte für die virtuellen Atome und Bindungen erzeugt.

Die Analyse der SMILES Notation erfolgt in drei Schritten: Zuerst werden aus dem gesamten SMILES String die Leerzeichen, die Tabulatorzeichen, die Zeilenumbrüche und die Enterzeichen entfernt, mit Ausnahme der Leerzeichen in dem SMILES String Teilen, die in eckigen Klammern stehen. Dieser Programmschritt erfolgt bereits bei der Erzeugung eines Objektes vom Typ Mol.

Die weiterführende Analyse der SMILES Zeichenfolge erfolgt in den Funktionen DoParse und lex der Klasse Mol.

Die Funktion DoParse hat die Aufgabe, den Rückgabewert der Funktion Lex zu analysieren und in Abhängigkeit von dem Rückgabewert virtuelle Atome und Bindungen zu erzeugen, die Atomeigenschaften und Bindungseigenschaften festzulegen, sowie die Definition und Integration von Templates zu realisieren.

Die in der Funktion DoParse aufgerufene Funktion Lex analysiert und interpretiert die Zeichen der eingegebenen SMILES Notation und liefert das Interpretationsergebnis an die aufrufende Funktion DoParse.

In den folgenden Kapiteln wird die programmtechnische Realisation der Funktionen DoParse und Lex im Detail erläutert.

4.4.3.1 Die Funktion DoParse

In den folgenden Abschnitten ist der Quellcode der Funktion DoParse der Klasse Mol auszugsweise abgebildet und die Funktionsweise der einzelnen Programmsequenzen erläutert.

Die Funktion DoParse der Klasse Mol interpretiert den eingegebenen SMILES String durch den Aufruf der Funktion Lex und analysiert den Rückgabewert der Funktion. In Abhängigkeit von dem Interpretationsergebnis erzeugt die Funktion DoParse virtuelle Atome und virtuelle Bindungen, legt die Atomeigenschaften und Bindungseigenschaften fest, generiert und integriert Templates in die virtuelle Molekülstruktur.

Die Funktion DoParse initialisiert zunächst die lokalen und öffentlichen Membervariablen und analysiert anschließend in einer while – Schleife den Rückgabewert der Funktion Lex, bis die Funktion Lex den Wert END zurück gibt. Der entsprechende Quellcodeteil der Funktion ist nachfolgend dargestellt.

```
Atom *Mol::DoParse(Atom *a, int bondexpected)
{
        // Standartmäßig wird a mit NULL und bondexspected mit 0 in Headerdatei initialisiert
        // Initialisierung der lokalen Variablen
        int i,j, token, bondtype, charge, cell, noligand, prozent, firstprozent=0;
        double mass, bondangle, bondlength; char axial; Mol *var;
        Atom *firstatom=NULL, *lastatom=NULL, *aa, *newatom;
        char *p,*pp;
        bool flag, varassign=false; const char *errstr="ERROR in Mol::DoParse ";
        // Initialisierung der Membervariablen
        m_parse_lastbond=-2; m_parse_charge=0, m_parse_celltype=-1; m_parse_multi=1;
        m_parse_axial=0; m_parse_mass=0.; m_parse_noligand=0; m_parse_prozent=0;
        m_parse_bondangle=m_parse_bondlength=0.;

        while((token=Lex(firstatom))!=END)
        {
                // Zuweisung der Membervariablen in die lokalen Variablen
                bondtype=m_parse_lastbond; charge=m_parse_charge; cell=m_parse_celltype;
                mass=m_parse_mass; axial=m_parse_axial; noligand=m_parse_noligand;
                prozent=m_parse_prozent;
                //Reinitialisierung der Membervariablen
                m_parse_charge=0; m_parse_celltype=-1; m_parse_mass=0.; m_parse_axial=0;
                m_parse_noligand=0; m_parse_prozent=0;
                ...
}
```

Im weiteren Funktionsablauf wird der Rückgabewert der Funktion Lex in einer switch-Anweisung unterschieden. Die switch-Anweisung unterscheidet die Rückgabewerte ATOM, PROZENT, VAR, VARASSIGN, ERROR und die runden Klammerzeichen. In Abhängigkeit von dem Rückgabewert erzeugt die Funktion virtuelle Atome, realisiert cyclische Bindungsstrukturen, generiert und integriert Templates in die Molekülstruktur. Der nachfolgende Quellcodeabschnitt zeigt die Programmsequenz zur Generierung und Initialisierung der virtuellen Atome der Molekülstruktur.

```
Atom *Mol::DoParse(Atom *a, int bondexpected)
{
...

switch(token)
{
        case ATOM:
                a=new Atom(m_atomtable->GetEntry(m_buf), lastatom?lastatom:firstatom);
                //Verwendeten Celltypen ermitteln, Werte kleiner 0 sind Standartcellen
                if (cell<0) cell= - *a->m_at->celltypes; else a->m_cell.m_typeset=cell;i=abs(cell);
                a->m_cell.m_type=cell;
                //Ermittlung der Ladung
                a->m_charge=charge;
                //Abweichung von Standardmasse?
                if (mass!=0.) a->m_mass=mass;
                a->m_cell.m_case=axial;
                a->m_noligand=noligand;
                //Überprüfung ob Celltyp erlaubt ist?
                for (p=pp=a->m_at->celltypes; p && *p; p++)
                {
                        if (*p==i) break;
                }
                if (*p==0 && cell>0 || *pp && cell==0 || *p && abs(i)!=*p)
                {
                        m_err << errstr << "Celltype not defined for " << a->m_sym;
                        Error(firstatom?firstatom:a);
                }
...
}
```

In dem zuvor abgebildeten Quellcodeabschnitt ist die Auswertung des Rückgabewertes ATOM der Funktion Lex dargestellt. Die Funktion erzeugt zunächst ein virtuelles Atom, ein Objekt der Klasse Atom, mit den Initialisierungswerten aus der Atomtabelle durch den Aufruf der Funktion GetEntry und ermittelt anschließend aus den Membervariablen den zu verwendenden Zelltyp und weist den Eigenschaften des Atoms, die von den Standardwerten abweichenden Werte für die Atomladung, die Atommasse und den Wert der eventuellen axialen Position des Atoms zu.

Im weiteren Programmablauf überprüft die Funktion, ob bereits ein Atom der zu erzeugenden virtuellen Molekülstruktur existiert und erzeugt gegebenenfalls durch den Aufruf der Funktion MakeBond eine virtuelle Bindung zwischen den Atomen. Die Funktion DoParse berücksichtigt dabei die Anzahl der freien Bindungen des bereits existierenden Atoms und eventuelle cyclische Bindungen des neu erzeugten virtuellen Atoms.

```
Atom *Mol::DoParse(Atom *a, int bondexpected)
{
...

if (firstatom==NULL)
{
        firstatom=a;
        if (firstprozent)
        {
                MakeBond(firstatom,NULL,bondtype,/*flag*/ false,/*onlyatom*/ false,firstprozent);
                firstprozent=0;
        }
}
else    //firstatom ist vorhanden
{
        if (lastatom==NULL && bondexpected && firstatom->FreeBondsCount()<abs(bondtype)/2
          +abs(bondexpected)/2)
        {
                m_err << errstr << "Too many bonds on '" << firstatom->m_sym << "'";
                Error(firstatom);
        }
        if (lastatom==NULL && firstatom->FreeBondsCount()==abs(bondtype)/2)
                // Mit der Bindung sind alle  Bindungen belegt
                flag=false;
        else
                // Nach der Bindung sind noch Bindungen frei
                flag=true;
        MakeBond(lastatom?lastatom:firstatom,a,bondtype,flag);
        // Weitere Bindungen mögliche
        if (a->FreeBondsCount()>0) lastatom=a;
}
...
}
```

Im weiteren Verlauf der Funktionsroutine wird der Multiplikator des virtuellen Atoms, in der linearen SMILES - 3D Notation durch das Zeichen * und eine nachfolgende Zahl gekennzeichnet, berücksichtigt. Die Programmsequenz erzeugt die dem Wert der Membervariablen m_parse_multi entsprechende Anzahl an gleichen virtuellen Atomen und integriert diese in die virtuelle Molekülstruktur.

```
Atom *Mol::DoParse(Atom *a, int bondexpected)
{
...
//Erzeugung gleicher Atome entsprechend des Multiplikators
for (j=1; j<m_parse_multi; j++)
{
        a=new Atom(m_atomtable->GetEntry(m_buf),lastatom?lastatom:firstatom);
        // Ermittlung des Celltypes, negative Werte für Standartcelltypen
        if (cell<0) cell= - *a->m_at->celltypes;
        i=abs(cell);
        a->m_cell.m_type=cell;
        //Ermittlung der Ladung
        a->m_charge=charge;
        // Ermittlung der Nichtstandardmasse
        if (mass!=0.) a->m_mass=mass;
        a->m_cell.m_case=axial;
        a->m_noligand=noligand;
        // Ist der Celltype erlaubt?
        for (p=pp=a->m_at->celltypes; p && *p; p++)
        {
                if (*p==i) break;
        }
        if (*p==0 && cell>0 || *pp && cell==0 || *p && abs(i)!=*p)
        {
                m_err << errstr << "Celltype not defind for '" << a->m_sym << "'";
                Error(firstatom);
        }
        MakeBond(lastatom?lastatom:firstatom,a,bondtype);
        lastatom=a;
}
m_parse_multi=1;
break;
...
}
```

Im nächsten Schritt der switch-Anweisung wird der Wert PROZENT unterschieden und durch den nachstehenden Quellcodeabschnitt behandelt.

```
Atom *Mol::DoParse(Atom *a, int bondexpected)
{
...
case PROZENT:
        a=lastatom?lastatom:firstatom;
        if (a==NULL) { firstprozent=prozent; break; }
        if (a->m_varname)
        {
                if (lastatom==NULL && firstatom->m_varname==NULL
                  && firstatom->FreeBondsCount()==abs(bondtype)/2)
                        //Bindung belegt alle Bindungsmöglichkeiten an firstatom
                        flag=false;
                else // freie Bindungen an firstatom
                        flag=true;
                MakeBond(a,NULL,bondtype,flag,false,prozent);
        }
        else // kein Varname
        {
                if (lastatom==NULL && firstatom->m_varname==NULL
                  && firstatom->FreeBondsCount()==abs(bondtype)/2)
```

```
                    flag=false;
                else
                    flag=true;
                MakeBond(a,NULL,bondtype,flag,false,prozent);
        }
        break;
...
}
```

Der zuvor dargestellte Quellcodeabschnitt zeigt die Routine zur Erzeugung einer cyclischen Bindungsstruktur. Durch den Aufruf der Funktion MakeBond wird eine virtuelle Bindung zu der im Parameter prozent spezifizierten Bindungsstelle erzeugt. Die Funktionsroutine berücksichtigt bei dem Aufbau der cyclischen Bindungsstruktur, ob die Bindung zu einem bereits existierenden virtuellen Atom oder die Bindung zu einem virtuellen Atom eines Templates generiert wird.

Der nächste Quellcodeabschnitt zeigt die Programmsequenz, die bei dem Rückgabewert VAR ausgeführt wird. Die Funktion Lex gibt den Wert Var zurück, wenn die Funktion die Zeichenfolge in der linearen Notation als den Namen einer vordefinierten Molekülstruktur, einem Template, interpretiert.

```
Atom *Mol::DoParse(Atom *a, int bondexpected)
{
...

case VAR:
        // Ermittlung der Variablen aus der Molliste
        if ((var=m_mollist->GetVar(m_buf))==NULL)
        {
                m_err << errstr << "Variable '" << m_buf << "' not defined";
                Error(firstatom);
        }
        // Erzeugung einer Kopie der Atome der Variablen
        m_on_err_delete_atom1=newatom=var->m_atom->Copy();
        // Entfernung der H - Atome
        newatom->StripH();
        // Ermittlung der Nummer der Variablen
        for (i=0, a=firstatom; a; a=a->m_next)
        {
                if (a->m_varname && strcmp(a->m_varname,var->m_varname)==0
                  && i<a->m_varcount)
                        i=a->m_varcount;
        }
        i++;
        // Zuweisung des Namens bzw. der Nummer der Variablen in die jeweiligen Atome
        for (a=newatom; a; a=a->m_next)
        {
                a->m_varname=uw_salloc(var->m_varname);
                a->m_varcount=i;
        }
        // Überprüfung ob erstes Atom schon vorhanden ist ?
```

```
        if (firstatom==NULL)
        {
                // Erzeugung eines ersten Atoms
                firstatom=newatom->Copy();
                if (firstprozent)
                {
                        MakeBond(firstatom,NULL,bondtype,false,false,firstprozent);
                        firstprozent=0;
                }
                j=1;
        }
        else j=0;
        {
                // Erzeugung der multiplen Variablen
                for (i=j; i<m_parse_multi; i++)
                {
                        a=m_on_err_delete_atom2=newatom->Copy();
                        if (lastatom==NULL && firstatom->m_varname==NULL
                          && firstatom->FreeBondsCount()==abs(bondtype)/2) flag=false;
                        else flag=true;
                        MakeBond(lastatom?lastatom:firstatom,a,bondtype,flag);
                        m_on_err_delete_atom2=NULL; lastatom=firstatom->Last();
                }
        }
        m_on_err_delete_atom1=NULL;
        // Zerstörung des lokalen Atoms
        newatom->First()->Delete();
        m_parse_multi=1;
        lastatom=firstatom->Last();
        break;
...
}
```

Der zuvor abgebildete Quellcodeabschnitt zeigt die Erzeugung eines Templates und dessen Integration in die aus dem SMILES String generierte Molekülstruktur.

Die Funktion ermittelt zunächst das Template aus der Molekülliste, erzeugt von dem ermittelten Template eine lokale Kopie der Molekülstruktur und entfernt durch den Aufruf der Funktion StripH die Wasserstoffatome der lokalen Kopie des Templates. Anschließend überprüft die Funktion, ob das gleiche Template bereits in dem neu erzeugten Molekül vorhanden ist und inkrementiert entsprechend den Index des Templates. Im nächsten Programmschritt wird die lokal erzeugte Molekülstruktur des Templates durch den Aufruf der Funktion MakeBond mit einer Bindung mit dem aus dem SMILES String erzeugten Molekül verbunden und in die aus dem SMILES String erzeugte Molekülstruktur kopiert. Falls noch keine aus dem eingegebenen SMILES String erzeugte Molekülstruktur existiert, interpretiert die Funktionsroutine die lokale Kopie des Templates als Anfang der aus dem SMILES String zu erzeugenden Molekülstruktur.

Im nachfolgenden Quellcodeabschnitt ist die Analyse des Rückgabewertes „(dargestellt. Bei der Analyse einer Seitenkette bzw. Verzweigung des SMILES Strings wird zunächst die Variable m_parcounter, zur Überprüfung der Anzahl der runden Klammern in der linearen Zeichenfolge, um den Wert 1 inkrementiert und anschließend die Funktion DoParse rekursiv aufgerufen. Im weiteren Ablauf der Funktionsroutine ermittelt die Funktion einen eventuellen Multiplikator der Seitenkette und erzeugt eine lokale Kopie der Molekülstruktur der Seitenkette. Anschließend wird die lokale Kopie entsprechend des Wertes des Multiplikators durch den Aufruf der Funktion MakeBond unter Berücksichtigung der Bindungsmöglichkeiten mit dem Molekül verbunden und in die aus dem SMILES String generierte Molekülstruktur kopiert.

```
Atom *Mol::DoParse(Atom *a, int bondexpected)
{
...
case '(':
        m_parse_parcounter++;
        // Zuweisung des Bindungswinkels bzw. der Bindungslänge der Seitenkette
        bondangle=m_parse_bondangle; bondlength=m_parse_bondlength;
        m_parse_bondangle=m_parse_bondlength=0;
        // Analyse der Seitenkette (siehe DoParse() )
        try { a=DoParse(NULL,bondtype); }
        catch (int)
        {
                if (firstatom) firstatom->Delete();
                throw 1;
        }
        m_parse_bondangle=bondangle; m_parse_bondlength=bondlength;
        if (a==NULL)
        {
                m_err << errstr << "'(...)' no atom returned";
                Error(firstatom);
        }
        else    // Seitenkette existiert
        {
                for (j=0; j<m_parse_multi; j++)
                {
                        // Erzeugung einer Kopie der Seitenkette
                        newatom=a->Copy();
                        if (firstatom==NULL) firstatom=newatom;
                        else if (newatom)
                        {
                                // Ermittlung des letzten Atoms
                                aa=firstatom->Last();
                                if (aa->m_varname==NULL && newatom->m_varname==NULL)
                                {
                                        aa->m_next=newatom; newatom->m_prev=aa;
                                }
                                if (lastatom==NULL
                                  && firstatom->FreeBondsCount()==abs(bondtype)/2) flag=false;
                                // bindung des letzten atoms an das newatom
                                MakeBond(lastatom?lastatom:firstatom,newatom,bondtype,flag);
```

```
                        }
                }
                // lastatom=lastatom?lastatom:firstatom;
                // Zerstörung des Hilfsatoms
                a->First()->Delete(); m_parse_multi=1;
        }
        break;
...
}
```

Im weiteren Programmablauf erfolgt die Auswertung des Rückgabewertes „)“ Die Funktionsroutine dekrementiert die Variable m_parcounter um den Wert 1 und liefert, wenn die Anzahl der öffnenden und schließenden runden Klammern gleich ist, als Rückgabewert die Referenz auf das erste Atom der zuvor erzeugten Seitenkette.

```
Atom *Mol::DoParse(Atom *a, int bondexpected)
{
...
case ')':
        if (m_parse_parcounter<0)
        {
                m_err << errstr << "no '(' for ')'";
                Error(firstatom);
        }
        m_parse_parcounter--; return firstatom;
...
}
```

4.4.3.2 Die Funktion Lex

In den nachfolgenden Abschnitten wird die Funktionsweise der einzelnen Programmsequenzen der Funktion Lex erläutert und der Quellcode der Funktion, zur besseren Übersicht, auszugsweise dargestellt.

Die Funktion Lex der Klasse Mol analysiert die einzelnen Zeichen der eingegebenen SMILES Notation durch Inkrementierung der auf die Zeichenfolge verweisenden Referenz. In Abhängigkeit von dem Interpretationsergebnis des jeweils referenzierten Zeichens, liefert diese Funktion die Rückgabewerte ATOM, PROZENT, VAR, VARASSIGN und die runden Klammern. Des weiteren erzeugt die Funktion eine öffentliche Referenz auf den noch nicht analysierten Teil der SMILES Zeichenfolge.

Der nachfolgende Quellcodeabschnitt zeigt die Initialisierungsroutine und verdeutlicht das grundlegende Prinzip der Zerlegung der linearen Notation in die

codierten Bestandteile. Die Funktion Lex initialisiert zuerst die lokalen Variablen und erzeugt eine lokale Referenz auf den Anfang der eingegebenen SMILES Zeichenfolge. In der anschließenden while-Schleife wird die Referenz inkrementiert und der referenzierte Wert analysiert. Nachdem alle Zeichen der linearen Notation interpretiert sind wird die while-Schleife beendet und die Funktion mit der Rückgabe des Wertes END beendet.

```
int Mol::Lex(Atom *delatom)
{
	char *p, *p1; int i; static bool onlysym;
	const char *errstr="ERROR in Mol::Lex ";
	*m_buf=0; p=m_smilesp;
	while(1)
	{
		if (*p==0)
			return END;

		…

		}
}
```

Im weiteren Ablauf der while-Schleife werden die referenzierten Zeichen der linearen Notation interpretiert. Die Analyseroutine fasst einen großen Buchstaben und nachfolgende kleine Buchstaben als ein Atom auf, wenn die Zeichenfolge einem Atomsymbol der IUPAC entspricht, die in der Atomtabelle m_atomtable aufgeführt sind. Die Routine erkennt den Buchstaben L als terminalen Liganden und bestimmt die Nummerierung des Liganden. Der Index des terminalen Liganden wird in der Membervariablen m_parse_noligand gespeichert. Weiterhin ermittelt die Funktionsroutine den eventuellen Multiplikator des virtuellen Atoms und speichert diesen in der Membervariablen m_parse_multi zwischen. Anschließend wird die while-Schleife beendet und die Funktion mit der Rückgabe des Wertes ATOM beendet.

```
…
	if (isupper(*p) || *p=='_') // Var oder Symbol
	{
		p1=(char *) m_buf; *p1++=*p++;
		while(*p && (islower(*p)  || *p=='_' || isdigit(*p) || (*m_buf=='_'
			&& isupper(*p) && (*(p+1)=='_' || islower(*(p+1))
			|| isdigit(*(p+1))))
			|| (onlysym==false && isdigit(*p))
			&& p1-(char *) m_buf<sizeof(m_buf)-2)) *p1++=*p++;
		*p1=0;
		// Ermittlung des Multiplikators m_parse_multi
		if (*p=='*' && isdigit(*(p+1)))
		{
			p++; m_parse_multi=atoi(p);
```

```
                while(*p && isdigit(*p)) p++;
        }

        if (*m_buf=='L')
        {
                // Ermittlung ob eine Zahl folgt
                p1=(char *) m_buf+1;
                while (*p1 && isdigit(*p1)) p1++;
                if (*p1==0)
                {
                        m_parse_noligand=atoi(m_buf+1);
                        *(m_buf+1)=0;
                }
        }
        if (m_atomtable->GetEntry(m_buf))
        { // Atom
                m_smilesp=p;
                return ATOM;
        }
…
```

Der nachfolgende else- Teil der Analyseroutine interpretiert die referenzierten Zeichen, die kein Atomsymbol darstellen als den Namen eines Templates. Die while-Schleife wird beendet und die Funktion gibt, in Abhängigkeit von den nachfolgenden Zeichen der linearen Notation, entweder den Wert VARASSIGN, zur Kennzeichnung der Definition eines neuen Templates, oder den Wert VAR zur Kennzeichnung eines bereits existierenden Templates zurück. Bei der Definition eines neuen Templates wird die lineare Notation in der Membervariablen m_varname gespeichert und der Referenz m_smilesp der Anfang der SMILES Zeichenfolge der neu definierten Molekülstruktur zugewiesen

```
…
        else
        { // VAR
                *p1=0; m_smilesp=p;
                if (*p && *p==':' && *(p+1)=='=') // varname definition
                {
                        if (m_varname)
                        {
                                m_err << errstr << "definition in smiles string not allowed";
                                Error(delatom);
                        }
                        m_varname=uw_salloc(m_buf);
                        m_smilesp=p+2;
                        return VARASSIGN;
                }
                return VAR;
        }
...
```

In einer weiteren else if-Anweisung überprüft die Analyseroutine das referenzierte Zeichen der linearen Zeichenfolge auf eine cyclische Bindung, die

durch das Zeichen % und eine nachfolgende Zahl gekennzeichnet ist. Die Funktion ermittelt die Nummerierung der cyclischen Bindung und speichert die in eine Zahl konvertierten Buchstaben in der Membervariablen m_parse_prozent zwischen. Anschließend wird die while-Schleife beendet und die Funktion mit der Rückgabe des Wertes PROZENT beendet.

```
...
// Prüfung auf Ringsystem
else if (*p=='%')
{
        p++; m_parse_prozent=atoi(p);
        while (isdigit(*p)) p++;
        m_smilesp=p;
        return PROZENT;
}
...
```

Im nächsten Programmsschritt der while-Schleife ermittelt die Funktion Lex eine Verzweigung bzw. eine Seitenkette in der Molekülstruktur. In einer weiteren else if-Anweisung überprüft die Funktion das referenzierte Zeichen auf die runden Klammern, ermittelt einen optionalen Multiplikator der Seitenkette und speichert den in eine Zahl konvertierten Wert in der Membervariablen m_parse_multi zwischen. Die inkrementierte Referenz auf die lineare Notation der Seitenkette wird der Membervariablen m_smilesp zugewiesen und die Zeichenfolge in der Membervariablen m_buf zwischengespeichert. Abschließend wird die while-Schleife beendet und die Funktion mit der Rückgabe der Referenz auf das kontrollierte runde Klammerzeichen beendet.

```
...

// Prüfung auf Seitenkette
else if (*p=='(' || *p==')')
{
        p1=p++;
        if (*p=='*' && isdigit(*(p+1)))
        {
                m_parse_multi=atoi(++p);
                while(*p && isdigit(*p)) p++;
        }
        m_smilesp=p; sprintf(m_buf,"%c",*p);
        return *p1;
}

...
```

Der nachfolgend dargestellte Programmabschnitt zeigt die Überprüfung des referenzierten Zeichens auf eine explizit codierte Bindung, die in der linearen SMILES Notation durch die Zeichen ., :, -, = und # dargestellt ist. Die Analyseroutine

kontrolliert in der while-Schleife durch eine else-Anweisung die Zeichenfolge, durch Vergleich des referenzierten Zeichens mit den Bindungszeichen in der globalen Bindungssymboltabelle, auf ein Bindungssymbol. Das ermittelte Bindungssymbol wird in der Funktion, entsprechend der Reihenfolge der Bindungssymbole in der globalen Bindungssymboltabelle, in die jeweilige Zahl konvertiert und der Wert des Bindungstyps in der Membervariablen m_parse_lastbond zwischengespeichert. Anschließend wird die while-Schleife verlassen und die Funktion mit der Rückgabe des Wertes PROZENT beendet.

```
...

else
{
        // Überprüfung auf Bindungszeichen
        for (i=0; g_bondstring[i]; i++)
                if (*p==*g_bondstring[i]) break;
        if (g_bondstring[i]) // Bindung gefunden
        {
                m_parse_lastbond=i; p++; continue;
        }

...
```

Abschließend überprüft die Funktion in der while-Schleife, ausgehend von dem referenzierten Zeichen, die lineare Zeichenfolge auf die optionale Spezifikation, die durch eckige Klammern gekennzeichnet ist, einer Bindung oder eines Atoms.

Die Analyseroutine kontrolliert in einer if-Anweisung das referenzierte Zeichen der Notation auf das Zeichen „[„ und interpretiert die dem Zeichen „[„ folgende Zahl zunächst als spezifizierte Atommasse und weist den Wert der entsprechenden Membervariablen m_parse_mass zu. Folgt der Zahl ein Bindungssymbol, so wird die zuvor ermittelte „Atommasse“ als der dihedrale Winkel der Bindung aufgefasst und der Wert in der Membervariablen m_parse_bondangle gespeichert. Eine dem Bindungssymbol folgende Zahl wird entsprechend als die spezifizierte Bindungslänge interpretiert und in der Membervariablen m_parse_bondlength zwischengespeichert.

Zur Prüfung der Zeichen in eckigen Klammern auf ein spezifiziertes Atom wird die Funktion Lex rekursiv aufgerufen. Die dem Atomsymbol folgende Zahl wird als Spezifizierung der Valenzen gewertet und der entsprechenden Variablen zugewiesen. Die weiterführende Analyse der Spezifikation des Atoms durch die

Funktion Lex erfolgt, wenn die rekursiv aufgerufene Funktion Lex den Rückgabewert ATOM hat. Bei der Analyse der weiterführenden Spezifikation des Atoms wird die Atomladung, gekennzeichnet durch die Zeichen „+“ oder „-„,entsprechend der Anzahl der Zeichen in eine integer Zahl und der Buchstabe „a“, zur Kennzeichnung einer axialen Position, in den Wert 1 konvertiert, sowie die Werte in den entsprechenden Variablen gespeichert. Abschließend überprüft die Funktion lex am Ende einer Spezifikation eines Atoms, gekennzeichnet durch das Zeichen „]“, auf eine nachfolgende Zahl und speichert diese als Multiplikator des spezifizierten Atoms in der entsprechenden Variablen.

```
...
if (*p=='[')
{
        p++;
        while (*p && *p==' ') p++;
        // Masse oder Bindungslänge spezifiziert
        if (isdigit(*p))
        {
                m_parse_mass=atof(p); // mass or bondangle
                while (*p && (isdigit(*p) || *p=='.')) p++;
                while (*p && *p==' ') p++;
        }
        m_smilesp=p;
        // Überprüfung auf Bindungssymbol
        for (i=0; g_bondstring[i]; i++)
        {
                if (*p==*g_bondstring[i]) break;
        }
        if (g_bondstring[i]) // Bindung gefunden
        {
                m_parse_bondangle=m_parse_mass; m_parse_mass=0;
                m_parse_lastbond=i; p++;
                while (*p && *p==' ') p++;
                // Bindungslänge spezifiziert
                if ((m_parse_bondlength=atof(p))!=0.)
                {
                        while (*p && (isdigit(*p) || *p=='.')) p++;
                        while (*p && *p==' ') p++;
                }
                if (*p !=']')
                {
                        m_err << errstr << "syntax error in bonddefinition";
                        Error(delatom);
                }
                p++; m_smilesp=p;
                continue;
        }
        // Atom
        onlysym=true;
        i=Lex(delatom);
        onlysym=false;
        p=m_smilesp;
        while (*p && *p==' ') p++;
        // Celltyp bzw. Valenzen ermitteln
```

```
if (isdigit(*p))
{
        m_parse_celltype=atoi(p);
        while(*p && isdigit(*p)) p++;
}
switch(i)
{
        case ATOM:
                while(*p)
                {
                        switch(*p)
                        {
                                case ']':
                                        p++;
                                        // Multiplikator ermitteln
                                        if (*p=='*' && isdigit(*(p+1)))
                                        {
                                                p++; m_parse_multi=atoi(p);
                                                while(*p && isdigit(*p)) p++;
                                        }
                                        m_smilesp=p; return ATOM;
                                case '+': case '-':
                                        // Ladung des Atoms ermitteln
                                        m_parse_charge=*p=='+'?1:-1;
                                        p++;
                                        if (isdigit(*p)) m_parse_charge*=atoi(p);
                                        while(isdigit(*p)) p++;
                                        break;
                                case 'a':
                                        m_parse_axial=1; p++;
                                        break;
                                case ' ':
                                        break;
                                default:
                                        m_err << errstr << "Lex (Error in parsing";
                                        m_smilesp=p+1; Error(delatom);
                        }
                        p++;
                }
                break;
...
```

4.4.4 Erzeugung der virtuellen Molekülstruktur

4.4.4.1 Funktioneller Aufbau der virtuellen Molekülstruktur

In Analogie zu einer chemischen Verbindung, die aus Atomen besteht die durch Bindungen miteinander verbunden sind, wird in der weiterentwickelten Software WinSmiles - 3D eine sogenannte virtuelle Molekülstruktur erzeugt. Diese virtuelle Molekülstruktur besteht aus virtuellen Atomen, Objekten vom Typ Atom, die mit virtuellen Bindungen, den Objekten vom Typ Bond, miteinander verbunden sind. Die dreidimensionale Struktur eines virtuellen Moleküls wird durch die Integration von vordefinierten strukturbildenden Zellen, den Objekten vom Typ Cell, realisiert.

Die in dem Objekt vom Typ Mol erzeugte virtuelle Molekülstruktur besteht aus Objekten vom Typ Atom, die durch eine Referenzverknüpfung miteinander verbunden sind. Diese Objekte vom Typ Atom enthalten wieder ein Objekt vom Typ Cell zur räumlichen Orientierung der Bindungsstellen des virtuellen Atoms und mehrere Objekte vom Typ Bond zur Realisierung der virtuellen Bindungen zwischen den Atomen. Jede Bindung besteht aus zwei Objekten vom Typ Bond, eine virtuelle Hinbindung und eine virtuelle Rückbindung, die durch eine Referenzverknüpfung miteinander verbunden sind. Der zuvor beschriebene programmtechnische Aufbau der virtuellen Molekülstruktur ist in Abbildung 4-2 schematisch dargestellt.

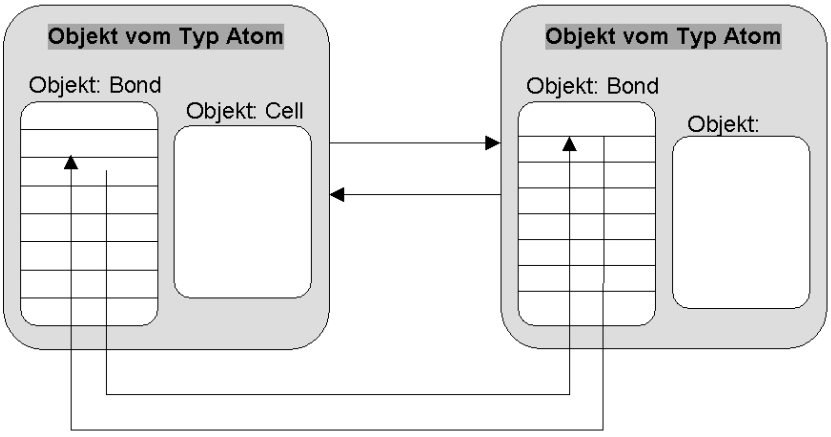


Abbildung 4-2: Programmtechnische Konzeption der virtuellen Molekülstruktur

Die Funktionalität zur Generierung einer virtuellen Molekülstruktur ist weitestgehend in dem Objekt vom Typ Mol realisiert, darüber hinaus kann das Objekt vom Typ Mol über die gesamte Funktionalität, die Methoden und Membervariablen, der zum Aufbau der Molekülstruktur verwendeten Objekte vom Typ Atom, Bond und Cell verfügen. Die Methoden und Membervariablen des Objektes vom Typ Mol sind in der Klasse Mol definiert und der Quellcode ist in die Dateien Mol.h und Mol.cpp implementiert.

Zur Erzeugung der virtuellen Molekülstruktur wird der Konstruktor der Klasse Mol aufgerufen. Als Parameter benötigt der Konstruktor die Referenz auf den eingegebenen SMILES String und die Referenz auf die zu verwendende Atomtabelle, welche die Standardwerte der Eigenschaften der Atome enthält, sowie die Referenz auf die Molekülliste, welche die Referenzen auf die bereits generierten Moleküle enthält, und ein Flag zur Steuerung des Ausgabeumfanges. Bei der

Konstruktion eines Objektes vom Typ Mol werden zunächst die Membervariablen initialisiert und anschließend der in Parameter referenzierte SMILES String durch die bereits im Abschnitt 4.4.3 „Die lexikologische Analyse des linearen SMILES Strings“ beschriebenen Methoden DoParse und Lex analysiert. Innerhalb der Methode DoParse werden, in Abhängigkeit von dem Rückgabewert der lexikologischen Analyse durch die Methode Lex, die virtuellen Atome erzeugt und der zu verwendende Zelltyp in dem Objekt vom Typ Cell des virtuellen Atoms festgelegt. Des weiteren werden durch den Aufruf der Methode MakeBond die Eigenschaften der virtuellen Bindung in den Objekten vom Typ Bond des virtuellen Atoms festgelegt und die Referenzverknüpfung der Objekte vom Typ Atom realisiert. Nach Beendigung der lexikologischen Analyse wird die Methode DoParse beendet und liefert als Rückgabewert die Referenz auf die erzeugte virtuelle Molekülstruktur. Anschließend werden durch den Aufruf der Methode SetUp in der zuvor generierten Molekülstruktur die aromatischen und cyclischen Bindungen erzeugt und die dreidimensionale Struktur des Moleküls berechnet, sowie die Referenz auf die neu erzeugte Molekülstruktur zu der Molekülliste hinzugefügt.

4.4.4.2 Berechnung der dreidimensionalen virtuellen Molekülstruktur

Die dreidimensionale Struktur des virtuellen Moleküls wird durch die Aneinanderreihung der unterschiedlichen Zelltypen der jeweiligen Atome und die anschließende Berechnung der Koordinaten der virtuellen Atome erreicht. Die Koordinaten des Atoms werden in den Membervariablen m_x, m_y und m_z des virtuellen Atoms, sowie die Koordinaten der Bindungsstellen der verwendeten Zellen in den Elementen der Membervariablen m_xyz des Objektes vom Typ Cell gespeichert.

Zur Beschreibung von geometrischen Transformationen von dreidimensionalen Objekten ist die Verwendung von homogenen Koordinaten geeignet, da durch diese eine einheitliche und effiziente Transformation der Koordinaten durch Matrixmultiplikation erzielt wird.[62] Dabei wird die Gesamttransformation, bestehend aus Rotation und Translation der Koordinaten, sowie die Skalierung der Koordinaten in einer (4x4) - Matrix zusammengefasst.

Wesentliche Vorteile der homogenen Koordinaten sind zum einen, dass mehrere nacheinander auszuführende Transformationen durch Matrixmultiplikation auf eine komplexe Gesamttransformation in einer Matrix reduziert werden können und zum andern, dass die Umkehrung der Transformation durch Inversion der Matrix realisierbar ist.

Zur Erzeugung der dreidimensionalen virtuellen Molekülstruktur werden die Zellen der durch eine Bindung miteinander verbundenen virtuellen Atome formal im Koordinatenursprung miteinander verbunden. Hierzu wird das To-Atom zunächst im Koordinatenursprung positioniert und die Bindungsstellen der entsprechenden Zelle um den Ursprung platziert. Anschließend wird die in Gl. 4 dargestellte homogene Matrix M_T zur Transformation der zu verwendenden Bindungsstelle, deren Koordinaten durch den Vektor $\vec{r}$ gegeben sind, erzeugt.

$$M_T = M_1 R_y\left(-\operatorname{acos}\left(\frac{(\mathrm{M}_1\,\vec{r})_z}{|\mathrm{M}_1\,\vec{r}|}\right)\right) R_z\left(\frac{\pi}{2}\right) \qquad \text{(Gl. 4)}$$

Die Matrix M_1 ist gegeben durch:

$$M_1 = R_z\left(-\operatorname{atan}\left(\frac{\vec{\mathrm{r}}_y}{\vec{r}_x}\right)\right) \qquad \text{(Gl. 5)}$$

Die Matrix M_T transformiert die entsprechende Bindungsstelle durch eine anfängliche Rotation um die y-Achse und eine nachfolgende Rotation um die z-Achse auf die x-Achse. Zur Rotation um die y-Achse wird die in Gl. 6 abgebildete homogene Rotationsmatrix R_y verwendet;

$$R_y(\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(Gl. 6)}$$

sowie zur Rotation um die z-Achse die in Gl. 7 dargestellte homogene Rotationsmatrix R_z .

$$R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(Gl. 7)}$$

Anschließend werden die übrigen Bindungsstellen der Zelle durch Rotation um die x-Achse unter Berücksichtigung des Zelltyps und des Dihedralwinkels β

ausgerichtet und die Bindungsstelle auf der x-Achse um die Bindungslänge *a* durch die Translationsmatrix T_x verschoben. Die homogene Translationsmatrix T_x ist in Gl. 8 dargestellt.

$$T_x(d) = \begin{pmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(Gl. 8)}$$

Die einzelnen Transformationsschritte der Bindungsstelle des To-Atoms werden, wie in Gl. 9 dargestellt, in der homogenen Matrix M_{To} zusammengefasst.

$$M_{To} = M_T T_x(d) R_x(\beta) \qquad \text{(Gl. 9)}$$

Zur Rotation um die x-Achse wird die in Gl. 10 dargestellte homogene Rotationsmatrix R_x verwendet.

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(Gl. 10)}$$

In einem weiteren Prozess wird die Position des bereits existierenden From-Atoms bestimmt und die in Gl. 11 dargestellt homogene Matrix M_{From} zur Transformation der Bindungsstelle des From-Atoms erzeugt.

$$M_{From} = M_T R_x(\gamma) \qquad \text{(Gl. 11)}$$

Die Matrix M_{From} transformiert die zu verwendende Bindungsstelle des From-Atoms auf die x-Achse und richtet die übrigen Bindungsstellen der Zelle des From-Atoms, unter Berücksichtigung des Dihedralwinkels γ durch Rotation um die x-Achse aus.

Die zuvor erzeugten homogenen Matrizen M_{To} und M_{From} werden, wie in Gl. 12 dargestellt, in der Gesamttransformationsmatrix *G* zusammengefasst.

$$G = M_{From}^{-1} M_{To} \qquad \text{(Gl. 12)}$$

Anschließend werden die Koordinaten der virtuellen Atome und die Koordinaten der Bindungsstellen der Zellen durch Multiplikation mit der Gesamttransformationsmatrix G neu berechnet. Die Teilschritte zum Aufbau der Matrizen sind in Abbildung 4-3 schematisch dargestellt.

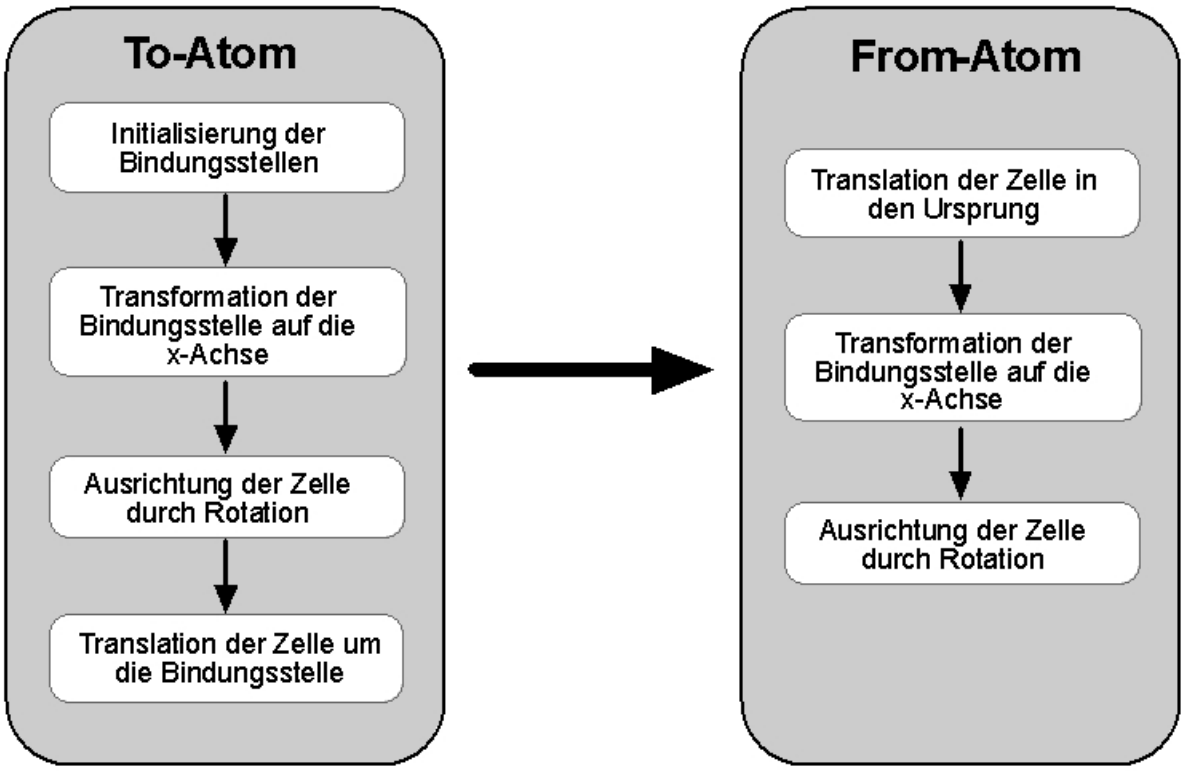


Abbildung 4-3: Transformationsschritte zum Bindungsaufbau

Zur programmtechnischen Erzeugung der virtuellen dreidimensionalen Molekülstruktur wird zunächst die Funktion SetCell des ersten virtuellen Atoms des Moleküls aufgerufen. Die Funktion SetCell erzeugt die Werte der Koordinaten der Bindungsstellen in Abhängigkeit von dem Zelltyp des virtuellen Atoms und weist diese Werte den Elementen der Variablen m_xyz des Objektes vom Typ Cell zu. Die Koordinaten der Bindungsstellen sind so festgelegt, dass der Mittelpunkt der Zelle im Koordinatenursprung ist und alle Bindungsstellen um den Wert 1 vom Koordinatenursprung entfernt und in Abhängigkeit vom Zelltyp räumlich orientiert sind. Nachdem die Position des ersten virtuellen Atoms des Moleküls im Koordinatenursprung platziert und die Position der Bindungsstellen festgelegt ist, wird die Methode SetCoord mit der Referenz auf die Bindung der ersten Bindungsstelle als Parameter aufgerufen. Innerhalb der Funktionsroutine SetCoord wird zunächst aus der virtuellen Rückbindung das To-Atom und das From-Atom ermittelt, die durch die Bindung miteinander verbunden sind.

Anschließend wird die Funktion SetCell des To-Atoms aufgerufen und dadurch die noch nicht existierenden Koordinaten des To-Atoms initialisiert. Bei der Erzeugung der Koordinaten wird das To-Atom ebenfalls im Koordinatenursprung und

die Bindungsstellen, in Abhängigkeit vom verwendeten Zelltyp, um den Wert 1 vom Ursprung entfernt platziert. Zur Transformation der Koordinaten der Bindungsstellen wird eine Matrix erzeugt, die die zu verwendende Bindungsstelle des To-Atoms auf die x-Achse transformiert und eine Rotation der Zelle des To-Atoms um 180° im Uhrzeigersinn um die y-Achse ausführt. Des weiteren wird durch die Matrix eine Drehung der Zelle um die x-Achse ausgeführt, die die zweite Bindungsstelle, bei Verwendung einer trigonalen oder einer tetraedrischen Zelle, bzw. die dritte Bindungsstelle, bei Verwendung einer oktaedrischen Zelle, im Uhrzeigersinn in die xz-Ebene dreht. Eine eventuelle Spezifikation des Dihedralwinkels der virtuellen Bindung wird durch eine weitere Drehung der Zelle des To-Atoms um die x-Achse in Uhrzeigersinn realisiert und in die Matrix implementiert. Abschließend wird in die Matrix noch die Translation integriert, die die Zelle des To-Atoms in x-Richtung um den Wert des Radius der Bindung des To-Atoms und den Wert des Radius der Bindung des From-Atoms verschiebt.

Zur Transformation der Koordinaten der Bindungsstellen des From-Atoms wird eine zweite Matrix erzeugt, die zunächst den Ursprung des Koordinatensystems in die Position des From-Atoms verschiebt und dann die Koordinaten der entsprechenden Bindungsstelle auf die x-Achse transformiert. Bei Verwendung einer trigonalen oder tetraedrischen Zelle werden zusätzlich die Koordinaten der ersten Bindungsstelle des From-Atoms durch die Drehung um die x-Achse in die xz-Ebene transformiert und die Zelle um weitere 180° um die x-Achse in Uhrzeigerrichtung gedreht. Durch diese Transformation wird erreicht, dass die übrigen Bindungsstellen der Zellen des To-Atoms bzw. des From-Atoms möglichst weit voneinander entfernt angeordnet werden. Die Bindungsstellen zweier benachbarter tetraedrischer Zellen werden im Bezug auf die Bindung zwischen den Zellen „staggered“ zueinander angeordnet.

Durch Matrixmultiplikation der ersten Matrix, die zur Transformation der Koordinaten des To-Atoms verwendet wird, mit dem Inversen der zweiten Matrix, die zur Transformation der Koordinaten des From-Atoms verwendet wird, wird eine Gesamttransformationsmatrix erzeugt. Anschließend werden die Koordinaten des To-Atoms und die Koordinaten aller Bindungsstellen des To-Atoms durch Multiplikation mit der Gesamttransformationsmatrix transformiert.

Durch diese Gesamttransformation wird formal die Bindungsstelle der Zelle des To-Atoms mit der Bindungsstelle der Zelle des From-Atoms im Ursprung des Koordinatensystems unter Berücksichtigung des Dihedralwinkels der Bindung miteinander verbunden.

Durch den rekursiven Aufruf der Methode SetCoord für alle Bindungen des To-Atoms wird die dreidimensionale virtuelle Molekülstruktur weiter aufgebaut, bis sämtliche Koordinaten der Atome und Bindungsstellen berechnet sind.

4.4.5 Visualisierung der virtuellen Molekülstruktur

4.4.5.1 Programmtechnische Voraussetzungen zur Visualisierung

Zur Visualisierung der virtuellen Molekülstruktur wird die standardisierte Software OpenGL in die Software WinSmiles - 3D implementiert. Die ursprünglich von Silicon Graphics [60] entwickelte plattform-unabhängige Software OpenGL ist von Microsoft in den Windows Betriebssystemen implementiert und wird von der hier weiterentwickelten Software WinSmiles - 3D zur Erzeugung eines Bildes der Moleküle verwendet. Das OpenGL Utility Toolkit (GLUT) ist ein leistungsstarkes programmierbares Interface zur Erzeugung von qualitativ hochwertigen dreidimensionalen Bildern.[63]

In dem nachfolgenden Abschnitt wird kurz die Funktionsweise des Softwareinterfaces OpenGL erläutert. Die grundlegende Aufgabe des Softwareinterfaces ist die Generierung einzelner Bilder zur Abbildung von zwei- oder drei-dimensionalen Objekten auf der zwei-dimensionalen Bildschirmoberfläche. Diese einzelnen Bilder werden aus der Beschreibung der geometrischen Struktur der Objekte erzeugt. Die Beschreibung der Struktur erfolgt entweder durch Sequenzen von Punkten, Linien oder Flächen oder durch ein Bild, das aus Pixelpunkten besteht.

In der folgenden Abbildung 4-4 sind die wesentlichen Programmschritte der Software OpenGL schematisch dargestellt.

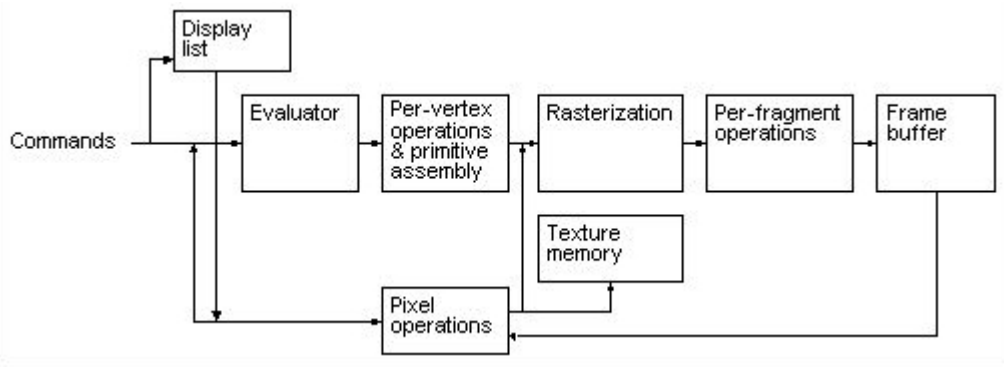


Abbildung 4-4: Wesentliche Programmschritte des OpenGL Interfaces [61]

Im ersten Programmschritt werden die dem Interface übergebenen Werte, welche die geometrischen Objekte beschreiben, zunächst ausgewertet und durch effektive polynomische Algorithmen an Linien und geometrische Flächen approximiert (Evaluator). Im anschließenden Programmschritt werden die evaluierten Werte der geometrischen Objekte auf Punkte, Linien und Flächensegmente reduziert, sowie durch Transformation und Beleuchtung auf den sichtbaren Ausgabebereich angepasst (per-vertex). Das so erzeugte drei-dimensionale Objekt wird in eine Serie von zwei-dimensionalen Einzelbildern in der xz-Ebene, die Punkte, Liniensegmente oder Polygone enthalten, in z-Richtung zerlegt. Im letzten Programmschritt werden die reduzierten zwei-dimensionalen Einzelbilder der geometrischen Struktur zu einem zwei-dimensionalen Gesamtbild des drei-dimensionalen Objektes weiterverarbeitet und das Gesamtbild als Pixelpunkte gespeichert. Bei der Erzeugung des Gesamtbildes des drei-dimensionalen Objektes werden die Einzelbilder unter Berücksichtigung der z-Werte der Einzelbilder und des Betrachtungspunktes des Gesamtbildes miteinander verglichen und die sichtbaren Punkte des Gesamtbildes als Pixel gespeichert, sowie die Farben der einzelnen Bildpunkte ermittelt.

Des weiteren verfügt das Softwareinterface OpenGL über die Möglichkeit bereits generierte Bildkomponenten, zum Beispiel das Bild einer Kugel in einer entsprechenden Variablen zu speichern (Displayliste) und erzielt durch die Integration der gespeicherten Komponenten in ein Gesamtbild eine hohe Effizienz bei der Erzeugung des Bildes.

4.4.5.2 Konzeption des Visualisierungsprozesses

Der Prozess der Visualisierung der virtuellen Molekülstruktur in der weiterentwickelten Software WinSmiles - 3D kann in mehrere Teilprozesse aufgeteilt werden, die in der nachfolgenden Abbildung 4-5 schematisch dargestellt sind.

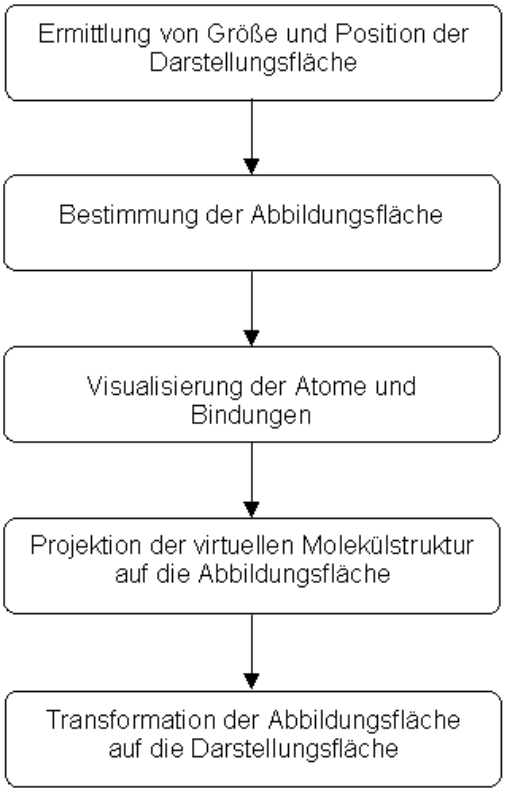


Abbildung 4-5: Visualisierungsprozess der virtuellen Molekülstruktur

Im ersten Teilprozess der Visualisierung wird zunächst die Position und Größe der Darstellungsfläche innerhalb der grafischen Benutzeroberfläche der Anwendung WinSmiles - 3D bestimmt. Die Darstellungsfläche in der Benutzeroberfläche ist der auf dem Monitor sichtbare Bereich, in dem das dreidimensionale Bild des visualisierten Moleküls dargestellt wird. Die Größe und Position der Darstellungsfläche ist von den Ansicht-Einstellungen der Benutzeroberfläche abhängig und individuell vom Anwender variabel.

Im zweiten Teilprozess wird die Größe des virtuellen Moleküls ermittelt und eine in der Größe optimierte Abbildungsfläche berechnet. Die Abbildungsfläche ist eine virtuelle Fläche, auf der das zu visualisierende Molekül unter Berücksichtigung

des Betrachtungspunktes vollständig abgebildet werden kann und die durch den letzten Teilprozess die „Transformation der Abbildungsfläche auf die Darstellungsfläche“ auf der Darstellungsfläche dargestellt wird. Die Größe der Abbildungsfläche ist von der Größe des virtuellen Moleküls abhängig.

Im dritten Teilprozess wird die virtuelle Molekülstruktur durch die Funktionsroutinen der virtuellen Atome und der virtuellen Bindungen visualisiert. Bei der Visualisierung der einzelnen Atome und Bindungen werden die Funktionsroutinen des OpenGL Interface aufgerufen und in Abhängigkeit von der Darstellungsform der Atome (z.B. Kugeldarstellung der Atome, Atomsymbole und Linien, etc.) dem Interface die entsprechenden Parameter (z.B. Darstellungsfarbe des Atoms, Atomradius, etc.) übergeben.

Im vierten Teilprozess wird die im vorherigen Teilprozess generierte virtuelle Molekülstruktur auf die im zweiten Teilschritt erzeugte Abbildungsfläche unter Berücksichtigung des Betrachtungspunktes projektiert.

Im fünften Teilprozess wird die Abbildung des Moleküls von der Abbildungsfläche auf die Darstellungsfläche in der grafischen Benutzeroberfläche transformiert und so das Bild der virtuellen Molekülstruktur auf dem Bildschirm erzeugt.

4.4.5.3 Programmtechnische Realisierung des Visualisierungsprozesses

Der zuvor beschriebene Prozess zur Visualisierung der virtuellen Molekülstruktur ist in der hier weiter entwickelten Software WinSmiles - 3D durch die Methoden zur Visualisierung der unterschiedlichen Objekte realisiert. Die nachfolgende Abbildung 4-6 zeigt eine schematische Darstellung der unterschiedlichen Programmschritte, die von den Objekten bei der Visualisierung der virtuellen Molekülstruktur aufgerufen werden.

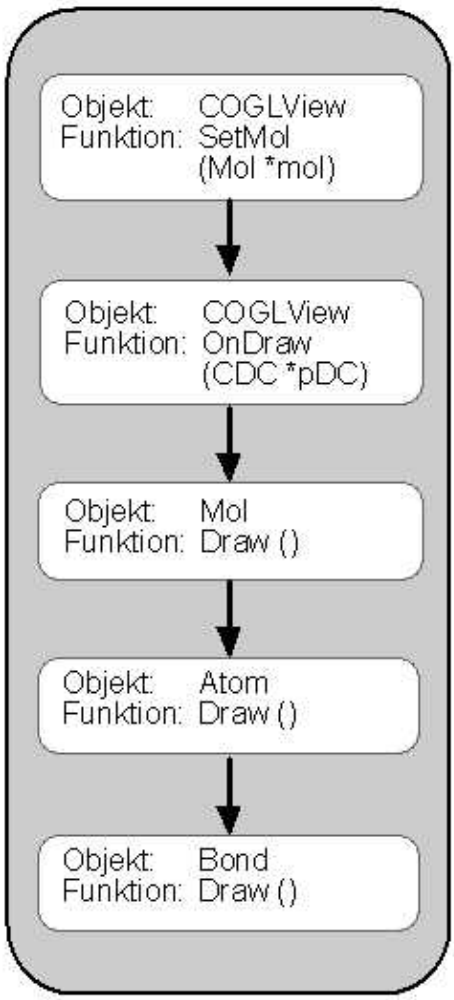


Abbildung 4-6: Visualisierungsfunktion der Objekte der virtuellen Molekülstruktur

Die Visualisierung der virtuellen Molekülstruktur wird, wie auch die Erzeugung der virtuellen Molekülstruktur aus dem eingegebenen SMILES String, durch das Betätigen der Schaltfläche „New“ in der grafischen Benutzeroberfläche der Software WinSmiles - 3D initiiert. Innerhalb der Funktion OnOgldlgnew wird, nachdem die virtuelle Molekülstruktur erzeugt ist, die Methode SetMol der Klasse COglView zur Visualisierung der im Parameter referenzierten Molekülstruktur aufgerufen.

Die Methode SetMol realisiert die Erzeugung einer größenoptimierten Abbildungsfläche und die Erzeugung eines auf die Benutzeroberfläche angepassten Darstellungsbereiches.

Die Methode SetMol ermittelt zunächst die minimalen, maximalen und durchschnittlichen Werte der x-, y- und z-Koordinaten der virtuellen Molekülstruktur.

Anschließend wird aus dem minimalen und dem maximalen Wert der x-Koordinate, sowie dem minimalen und dem maximalen Wert der y-Koordinate die Größe des Moleküls in der xy-Ebene bestimmt und die minimalen und maximalen Werte der x- und y-Koordinaten der Abbildungsebene berechnet. Die Größe der Abbildungsebene wird dabei so gewählt, dass die minimalen und maximalen Werte der Koordinaten auf der Koordinatenachse an die größere Differenz zwischen dem minimalen und dem maximalen Wert der Koordinaten auf der Koordinatenachse angepasst werden und die Werte der so berechneten Koordinaten die Eckpunkte der Abbildungsfläche bilden. Hieraus resultiert eine quadratische Abbildungsfläche, in der die visualisierte Molekülstruktur in Richtung der beiden Koordinatenachsen vollständig abgebildet werden kann und die in der Größe der Fläche optimiert ist. Abschließend ruft die Funktion SetMol die Methode OnDraw zur weiteren Visualisierung der Molekülstruktur auf und erzeugt ein zum Device Kontext, der die Zeichenbefehle des verwendeten OpenGL Rendering Kontextes enthält, kompatibles Objekt vom Typ Cbitmap, das auf die Größe des Darstellungsbereiches in der grafischen Benutzeroberfläche angepasst ist.

Die Methode OnDraw realisiert die Transformation der homogenen Koordinaten des virtuellen Moleküls von der Abbildungsfläche auf die Koordinaten der Darstellungsfläche in der grafischen Benutzeroberfläche und erzeugt eine Projektionsmatrix durch die das dreidimensionale virtuelle Molekül auf der zweidimensionalen Abbildungsfläche dargestellt wird.

Die Methode OnDraw, die innerhalb der Routine SetMol aufgerufen wird, generiert beim ersten Aufruf einen OpenGL kompatiblen alphanumerischen Zeichensatz zur Visualisierung der Atome der virtuellen Molekülstruktur durch die entsprechenden Atomsymbole. Anschließend wird die Größe der Darstellungsfläche innerhalb der grafischen Benutzeroberfläche ermittelt und die affine Transformation der Werte der x und y – Koordinaten von den Koordinaten der Abbildungsfläche auf die Koordinaten der Darstellungsfläche durch den Aufruf der OpenGL Funktion glViewport spezifiziert. Des weiteren wird eine Projektionsmatrix erzeugt, die eine parallele Projektion der dreidimensionalen virtuellen Molekülstruktur auf die xy-Ebene, die Abbildungsfläche, realisiert.

Die Größe der Abbildungsfläche ist durch die in der Funktion SetMol aus den minimalen und maximalen Werten der Koordinaten des Moleküls berechneten Werte festgelegt. Die Werte der Membervariablen m_minx, m_miny und m_minz ; sowie die Werte der Membervariablen m_maxx, m_maxy und m_maxz spezifizieren den unteren linken bzw. oberen rechten Eckpunkt des sichtbaren Bereiches auf der Abbildungsebene. Der Betrachtungspunkt des virtuellen Moleküls ist bei der Projektion im Koordinatenursprung lokalisiert.

Des weiteren wird durch Matrixmultiplikation der homogenen Koordinaten der Abbildungsmatrix die Rotation des virtuellen Moleküls um die x und y-Achse, sowie die Skalierung der Abbildungsgröße des Moleküls realisiert.

Zur Rotation und zur Skalierung des virtuellen Moleküls wird durch Koordinatentransformation zuerst eine Translation des Koordinatenursprungs in den Mittelpunkt der Abbildungsfläche bzw. in den Mittelpunkt des virtuellen Moleküls, der durch die Werte der Membervariablen m_midx, m_midy und m_midz definiert ist, ausgeführt und anschließend die Rotation um die x und y-Achse, sowie die Skalierung der Abbildungsgröße des Moleküls durch Koordinatentransformation ausgeführt. Anschließend wird durch eine weitere Transformation der Koordinaten eine Translation des Koordinatenursprungs vom Mittelpunkt des virtuellen Moleküls in den ursprünglichen Ursprung, der linken unteren Ecke der Abbildungsfläche ausgeführt.

Innerhalb der Funktionsroutine OnDraw wird abschließend die Methode Draw der Klasse Mol des virtuellen Moleküls zur weiteren Visualisierung aufgerufen.

Die Methode Draw der virtuellen Molekülstruktur, ein Objekt vom Typ Mol, kennzeichnet zunächst alle Bindungen eines Moleküls als noch nicht visualisiert und ruft anschließend die Methode Draw der Membervariablen m_atom, ein Objekt vom Typ Atom, zur Visualisierung der einzelnen Atome eines Moleküls auf.

Die Methode Draw des zu visualisierenden Atoms, ein Objekt vom Typ Atom, ermittelt zunächst den Radius des virtuellen Atoms aus der globalen Radientabelle g_KovRad. Anschließend wird bei der Darstellung der Atome durch Atomsymbole, in

Abhängigkeit vom ausgewählten Darstellungsmodus (z.B. nur Atomsymbol, Atomsymbol und Nummer des Atoms, etc.), die darzustellenden Buchstaben generiert und im Mittelpunkt des Atoms, der durch die Werte der Membervariablen m_x, m_y und m_z definiert ist, dargestellt.

Zur Darstellung der Atome durch dreidimensionale Kugeln wird eine Transformationsmatrix erzeugt, die eine Translation des Koordinatenursprungs in den Mittelpunkt des darzustellenden Atoms erzeugt und anschließend um den Koordinatenursprung eine dreidimensionale Kugel mit dem Radius m_rad des Atoms erzeugt. Die Erzeugung der Kugel zur Visualisierung des Atoms erfolgt durch den Aufruf der Funktion glutSolidSphere, eine Funktion die in dem OpenGL Interface implementiert ist. Die Farbe der Kugel ist durch die Darstellungsfarbe des zu visualisierenden Atoms in den Werten der Variablen m_r, m_g und m_b in der Atomtabelle m_at festgelegt .

Zur Darstellung der virtuellen Molekülstruktur durch Linien oder Kugeln, die durch Linien miteinander verbunden sind, ist es erforderlich die Bindungen der Atome zu visualisieren. Die Visualisierung der Bindungen erfolgt durch den Aufruf der Methode Draw jeder virtuellen Bindung des Atoms. Bei der Visualisierung einer Bindung wird zunächst die Bindungslänge aus dem Radius des From-Atoms und dem Radius des To-Atoms ermittelt. Anschließend wird eine Linie vom Koordinatenursprung, die Position des From-Atoms, zu der Position des To-Atoms erzeugt. Die Farbe der Linie ist entsprechend des Bindungsanteils des From-Atoms und des Bindungsanteils des To-Atoms durch die Darstellungsfarbe des From-Atoms und der Darstellungsfarbe des To-Atoms festgelegt.

Die von den Visualisierungsfunktion der virtuellen Atome, den Objekten vom Typ Atom, und von den Visualisierungsfunktionen der virtuellen Bindungen, den Objekten vom Typ Bond, in dem OpenGL Rendering context erzeugte Beschreibung der virtuellen dreidimensionalen Molekülstruktur wird durch die, in der Funktion OnDraw spezifizierte Projektion, auf der Abbildungsfläche unter Berücksichtigung des Betrachtungspunktes abgebildet. Abschließend wird das Bild der Abbildungsfläche auf die Darstellungsfläche transformiert und die dreidimensional

erscheinende Darstellung des Moleküls auf der grafischen Benutzeroberfläche erzeugt.

4.5 Automatische Generierung der SMILES - 3D Notation

In der weiterentwickelten Software WinSmiles - 3D ist die automatische Generierung der linearen SMILES - 3D Zeichenfolge basierend auf der internen virtuellen Molekülstruktur realisiert worden. Die virtuelle Molekülstruktur, bestehend aus virtuellen Atomen, die durch virtuelle Bindungen miteinander verbunden sind, wird durch diesen speziellen Algorithmus in die lineare SMILES - 3D Zeichenfolge konvertiert. Der Algorithmus realisiert die Erzeugung der Atom- und Bindungssymbole in der linearen Zeichenfolge unter Berücksichtigung der Regeln des SMILES - 3D Konzepts und stellt cyclische Bindungen und Verzweigungen der Molekülstruktur entsprechend dar. Des weiteren wird bei der Konvertierung der virtuellen Molekülstruktur in die lineare SMILES - 3D Zeichenfolge die optionale Modifikation von Atomen und Bindungen berücksichtigt und die entsprechenden Werte in der Notation erzeugt.

Die automatische Erzeugung der linearen SMILES - 3D Notation aus der virtuellen Molekülstruktur ermöglicht dem Anwender die einfache Codierung eines Moleküls in einer linearen Zeichenfolge ohne die Kenntnis der Regeln des SMILES - 3D Konzeptes und unabhängig von der Erzeugung der virtuellen Molekülstruktur.

4.5.1 Konzeption der Konvertierungsroutine

Im folgenden Abschnitt wird die grundlegende Konzeption des Konvertierungsalgorithmus erläutert und die wesentlichen Programmschritte der Generierung der linearen Zeichenfolge aus der virtuellen Molekülstruktur dargestellt. Der Konvertierungsablauf ist in Abbildung 4-7 schematisch abgebildet.

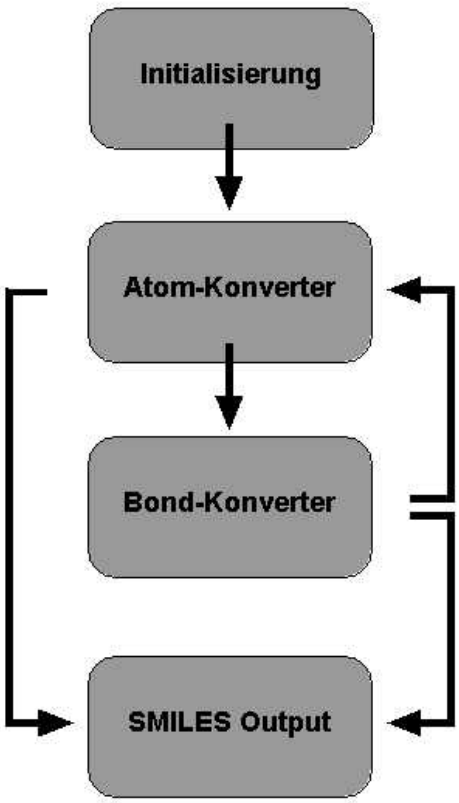


Abbildung 4-7: Programmschritte des Konvertierungsalgorithmus

Im ersten Schritt der Konvertierungsroutine, der Initialisierung, werden die lokalen Variablen initialisiert und der Bearbeitungszustand aller Bindungen der virtuellen Molekülstruktur als noch nicht konvertiert gekennzeichnet.

Bei der Konvertierung der internen virtuellen Molekülstruktur in die lineare SMILES Zeichenfolge wird zunächst das erste virtuelle Atom der Molekülstruktur analysiert. Die Analyse und die Konvertierung des virtuellen Atoms ist in der schematischen Darstellung durch die Programmsequenz „Atom-Konverter“ repräsentiert. In dieser Programmsequenz wird aus dem Objekt vom Typ Atom das IUPAC Atomsymbol erzeugt und aus den entsprechenden Membervariablen die optionalen Spezifikationen bestimmt, sowie eine eventuelle cyclische Bindung an der ersten Bindungsstelle des virtuellen Atoms berücksichtigt.

Im nächsten Schritt „Bond-Konverter“ wird jede virtuelle Bindung des Objektes vom Typ Atom analysiert und in Abhängigkeit vom Bindungstyp in das entsprechende Zeichen der linearen Notation konvertiert. Bei der Generierung des Bindungssymbols werden die optionalen Spezifikationen der Bindung aus den

entsprechenden Membervariablen der Objekte vom Typ Bond ermittelt und berücksichtigt.

Zur weiteren Generierung der linearen SMILES Notation aus der internen virtuellen Molekülstruktur wird für jede virtuelle Bindung, an die ein virtuelles Atom gebunden ist, die Konvertierungsroutine rekursiv aufgerufen und mit der Programmsequenz „Atom-Konverter“ fortgeführt. Der Programmschritt „Atom-Konverter“ konvertiert wiederum das virtuelle Atom in die entsprechende lineare Zeichenfolge und analysiert im weiteren Programmablauf in dem Schritt „Bond-Konverter“ alle Bindungen des Objektes vom Typ Atom. Der rekursive Konvertierungsprozess wird für alle Atome der virtuellen Molekülstruktur durchgeführt und abschließend mit der Ausgabe der generierten linearen Zeichenfolge in dem Schritt „SMILES Output“ beendet.

4.5.2 Programmtechnische Realisation der Konvertierungsroutine

Der Algorithmus zur automatischen Generierung der linearen Zeichenfolge ist in die Funktion Smiles() des Objektes vom Typ Atom, dem virtuellen Atom, implementiert. Im folgenden wird die Funktionsweise der Konvertierungsfunktion im Detail erläutert und der Quellcode der Funktion Smiles auszugsweise dargestellt. Zur automatischen Generierung der SMILES - 3D Notation wird die Funktion Smiles zuerst mit der Referenz auf das virtuelle Atom m-atom als Parameter aufgerufen. Die Methode Smiles unterscheidet zunächst die im Parameter referenzierten Objekte vom Typ Atom und Bond in einer if-else-Anweisung. Die if-Anweisung analysiert die referenzierte virtuelle Bindung und speichert das aus der Bindung ermittelte gebundene, virtuelle To-Atom in einer lokalen Variablen. Die else-Anweisung kennzeichnet zunächst alle Bindungen der virtuellen Molekülstruktur als nicht bearbeitet und initialisiert den lokalen Zwischenspeicher der linearen Zeichenfolge, sowie die lokalen Variablen.

```
char *Atom::Smiles(Atom *atom , Bond *bond)
{
	int i,j,n,m,mm; Bond *b; static char buffer[4090]; static bool bufferflag;
	bool extended=false, flag;

	if (bond)
	{
		if (bond->m_flag) return buffer;
		// Bearbeitungsstatus der Bindung kennzeichnen
		bond->m_flag=1;
		// Atom ermitteln und lokal speichern
		atom=bond->m_toatom;
	}
	else
	{
		// Bearbeitungsstatus zurücksetzen, Speicher initialisieren.
		SetBondFlag(0);
		*buffer=0; bufferflag=false;
	}
...
}
```

Zur Ermittlung der Seitenketten der virtuellen Molekülstruktur bestimmt die Routine bereits jetzt die Anzahl der nicht automatisch erzeugten To-Atome und die höchste besetzte Bindungsstelle des zu analysierenden virtuellen Atoms und speichert die Werte in lokalen Variablen. Die Werte dieser Variablen werden im weiteren Funktionsablauf der Konvertierungsroutine zur Bestimmung der Seitenketten der virtuellen Struktur verwendet.

```
// Ermittlung der Anzahl der To-Atome und die höchste Bindungsnummer mit einem To-Atom
for (i=m=1, mm=0; i<MAX_BOND; i++)
{
	b=&atom->m_bond[i];
	if (b->m_toatom && b->m_toatom->m_isautoatom==false) m++, mm=i;
}
```

Bei der Analyse des ersten Atoms der virtuellen Molekülstruktur überprüft die Routine das virtuelle Atom zunächst auf eine Ringbindung und speichert gegebenenfalls das Zeichen % und den nachfolgenden Ringbindungsindex zwischen. Bei einer cyclischen Mehrfachbindung ermittelt die Funktion zusätzlich das entsprechende Bindungssymbol aus der globalen Bindungssymboltabelle und speichert ebenfalls, unter Berücksichtigung einer soptionalen Spezifikation des dihedralen Bindungswinkels und der Bindungslänge, das Zeichen der virtuellen Bindung zwischen.

```
b=&atom->m_bond[0];
// Ermittlung ob erstes Atom im Ring
if (atom==atom->First() && b->m_prozent)
{
                sprintf(buffer,"%s%%%d",buffer,b->m_prozent);
                // Bindungstyp ermitteln
                if (b->m_type>-1)
                {
                        // Drehung der Bindung oder Bindungslänge definiert ?
                        if (b->m_rot || b->m_radinput) flag=true; else flag=false;
                        if (flag) strcat(buffer,"[");
                        if (b->m_rot) sprintf(buffer,"%s%.1f",buffer,b->m_rot);
                        strcat(buffer,g_bondstring[b->m_type]);
                        if (b->m_radinput) sprintf(buffer,"%s%.3f",buffer,2*b->m_rad);
                        if (flag) strcat(buffer,"]");
                }
}
```

Im weiteren Funktionsablauf überprüft die Methode das darzustellende virtuelle Atom auf eine von den Standardwerten abweichende Atommasse, einen abweichenden Zelltyp und eine abweichende Oxidationsstufe, sowie die optionale Spezifikation der axialen Bindungsposition einer pentagonalen Zelle und die Ladung des virtuellen Atoms. Die von den Standardwerten abweichenden Werte werden entsprechend der Konvention des SMILES - 3D Konzeptes in eckigen Klammern in der richtigen Reihenfolge zwischengespeichert. Das Atomsymbol wird aus der Membervariablen m_sym des darzustellenden Atoms ermittelt und durch den Aufruf der Funktion strcat lokal zwischengespeichert

```
// Ermittlung ob Abweichung von Standardwerten
if (atom->m_cell.m_case || atom->m_cell.m_typeset
        || atom->m_mass!=atom->m_at->mass || atom->m_charge
        || atom->m_oxistate!=*atom->m_at->oxi) extended=true;
if (extended) strcat(buffer,"[");
// Abweichung von Standardmasse ausgeben
if (atom->m_mass!=atom->m_at->mass) sprintf(buffer,"%s%.4f ",buffer,atom->m_mass);
// Atomsymbol oder Ligandennummer ausgeben
strcat(buffer,atom->m_sym);
if (atom->m_oz==0) sprintf(buffer,"%s%d",buffer,atom->m_noligand);
// Abweichenden Zelltypen ausgeben
if (atom->m_cell.m_typeset)
        sprintf(buffer,"%s %d",buffer,atom->m_cell.m_typeset);
// Ladung ausgeben
if (atom->m_charge)
{
        if (atom->m_charge > 0)
                sprintf(buffer,"%s %+d",buffer,atom->m_charge);
        else
                sprintf(buffer, "%s %-%d", buffer, atom->m_charge);
}
// Axiale Position ausgeben
if (atom->m_cell.m_case) strcat(buffer," a");
if (extended) strcat(buffer,"]");
```

Im weiteren Ablauf der Konvertierungsroutine werden in einer for-Schleife alle noch nicht analysierten virtuellen Bindungen des virtuellen Atoms, an die kein automatisch generiertes Wasserstoffatom gebunden ist, ermittelt. Aus den so bestimmten Objekten vom Typ Bond wird der Bindungstyp und eine optionale Spezifikation des dihedralen Bindungswinkels und der Bindungslänge bestimmt. Die Konvertierungsroutine ermittelt das Bindungssymbol aus der globalen Bindungssymboltabelle und speichert das Zeichen und die eventuell ermittelten Werte des Bindungswinkels bzw. der Bindungslänge, unter Berücksichtigung der Konvention des SMILES - 3D Konzeptes, in der richtigen Reihenfolge und gegebenenfalls in eckigen Klammern zwischen.

```
for (i=0; i<MAX_BOND; i++)
{
        b=&atom->m_bond[i];
        if (b->m_toatom==NULL && b->m_prozent==0) continue;
        if (b->m_flag || (b->m_toatom && b->m_toatom->m_isautoatom)) continue;
        // Aromatische Bindung ausgeben
        if (b->m_aromatic) strcat(buffer,":");
        else if (b->m_type>-1)
        {
                if (b->m_rot || b->m_radinput) flag=true; else flag=false;
                if (flag) strcat(buffer,"[");
                // Drehung der Bindung ausgeben
                if (b->m_rot) sprintf(buffer,"%s%.1f",buffer,b->m_rot);
                // Bindungssymbol ausgeben
                strcat(buffer,g_bondstring[b->m_type]);
                // Bindungsradius ausgeben
                if (b->m_radinput) sprintf(buffer,"%s%.3f",buffer,2*b->m_rad);
                if (flag) strcat(buffer,"]");
        }
        flag=false;
```

Bei der automatischen Generierung des SMILES Strings wird im weiteren Funktionsablauf in einer if-else-Anweisung die virtuelle Molekülstruktur auf cyclische Bindungen und Seitenkettenüberprüft Die in der if-Anweisung ermittelte cyclische Bindung wird durch das Zeichen % und den Index der Bindung in der linearen Zeichenfolge dargestellt und die Zeichen entsprechend zwischengespeichert. In der else-Anweisung wird die eventuell vorhandene Seitenkette der virtuellen Molekülstruktur ermittelt und die Konvertierungsroutine Smiles () mit der Referenz auf die virtuelle Bindung der Seitenkette als Parameter aufgerufen. Dieser Funktionsaufruf generiert die lineare SMILES Zeichenfolge der Seitenkette, die durch den Aufruf der Funktion sprintf in runden Klammern im lokalen Speicher zwischengespeichert wird .

Die weitere Erzeugung der linearen SMILES Notation wird durch den rekursiven Aufruf der Konvertierungsroutine Smiles () mit der Referenz auf die in der for-Schleife ermittelte virtuelle Bindung als Parameter realisiert.

```
                    if (b->m_prozent)
                    {
                            if (i>0 && atom!=atom->First()) sprintf(buffer,"%s%%%d",buffer,b-
>m_prozent);
                            continue;
                    }
                    else

                            // Flag der Rückbindung als bearbeitet markieren
                            b->m_backbond->m_flag=1;
                            // Ermittlung der Anzahl der  To-Atome des ToAtoms
                            for (j=n=0; j<MAX_BOND; j++) if (b->m_toatom->m_bond[j].m_toatom) n++;
                            // Ermittlung ob Seitenketten vorhanden
                            if (m>2 && n>1 && i<mm) flag=true;
                            if (flag) strcat(buffer,"(");
//                          sprintf(buffer, "%s%s", buffer, Smiles(NULL,b));
                            Smiles(NULL,b);
                            if (flag) strcat(buffer,")");
                    }
```

Abschließend wird die Konvertierungsroutine mit der Rückgabe der Referenz auf den lokalen statischen Speicher, der die aus der virtuellen Molekülstruktur generierte lineare SMILES Zeichenfolge enthält, beendet.

```
            // Erzeugten SMILES String zurückgeben
            return buffer;
    }

.
```

4.6 Die programminternen Parameter der Software WinSmiles - 3D

In diesem Kapitel werden die relevanten internen Parameter beschrieben die den virtuellen Atomen und Bindungen zum Aufbau der Molekülstruktur zugewiesen werden.

In der Software WinSmiles - 3D sind als Parameter der virtuellen Atome die allgemein verwendeten IUPAC-Atomsymbole, die Ordnungszahlen und die Anzahl der Bindungselektronen, sowie das Atomgewicht durch konstante Werte definiert. Des weiteren sind für jedes Atom die Werte der möglichen Oxidationsstufen festgelegt und die standardmäßige Anzahl an Bindungsorbitalen definiert. Zur Darstellung der dreidimensionalen Molekülstruktur werden in der Software WinSmiles - 3D verschiedene vordefinierte Zelltypen, zur räumlichen Orientierung der freien Elektronenpaare und der Bindungen, verwendet. Daher wird jedem Atom ein standardmäßig zu verwendender Zelltyp zugewiesen und darüber hinaus, in Abhängigkeit von dem Bindungsverhältnissen, die weiteren möglichen Zelltypen des virtuellen Atoms als interne Parameter festgelegt. Des weiteren ist es zur Visualisierung der Atome erforderlich, für jedes Atom einen Farbwert zur Darstellung festzulegen. Die zuvor beschriebenen internen Parameter, die für jedes Atom spezifisch sind, sind für jedes Atom in einem Objekt vom Typ AtomTable definiert und in einer statischen globalen Membervariablen, die aus den Objekten vom Typ AtomTable besteht, gespeichert.

Die weiterentwickelte Software WinSmiles - 3D ermöglicht dem Anwender zum einen, die standardmäßigen Werte der globalen Atomtabelle zu verwenden und zum anderen, die internen Parameter zu verändern. Die Werte der globalen Atomtabelle werden in dem Dialog „SMILES Periodic table“ editiert.

Zum Aufbau der molekularen Struktur ist es erforderlich, den Abstand zwischen benachbarten Atomen festzulegen. In der weiterentwickelten Software WinSmiles - 3D wird die Bindungslänge aus den kovalenten Radien der miteinander verbundenen Atome berechnet. Der kovalente Radius ist für jedes Atom in dem Objekt vom Typ KovRadTable definiert und in der statischen kovalenten Radientabelle in Abhängigkeit von der Oxidationsstufe des Atoms gespeichert.

4.6.1 Spezifische Parameter der virtuellen Atome

Zur Analyse der linearen SMILES Zeichenfolge und zum Aufbau der Molekülstruktur aus den virtuellen Atomen ist es notwendig, die atomspezifischen Parameter Atomsymbol und Ordnungszahl der Objekte vom Typ Atom zu definieren. Des weiteren ist es erforderlich, die Anzahl der Bindungselektronen des virtuellen Atoms festzulegen, da die Software WinSmiles - 3D die Anzahl der Bindungen des Atoms in der virtuellen Molekülstruktur entsprechend des SMILES - 3D Konzeptes auf der Basis der Regeln des VSEPR Konzeptes berechnet.

Die spezifischen Parameter des Objektes vom Typ Atom werden entsprechend der objektorientierten Programmkonzeption der Software in einem eigenständigen Objekt vom Typ CAtomTable abgekapselt. Das Objekt vom Typ CAtomTable enthält die Funktionalität zur Konstruktion und zur Destruktion des Objektes, sowie Methoden zur Serialisierung. In das Objekt vom Typ CAtomTable sind die objektartigen Strukturen vom Typ AtomTable eingebettet. In der objektartigen Struktur AtomTable sind unter anderem die Werte für das Atomsymbol, die Ordnungszahl, die Anzahl der Bindungselektronen und die Atommasse als spezifische Parameter für jedes Atom definiert.

Standardmäßig werden in der Software WinSmiles - 3D für die atomspezifischen Parameter die allgemein bekannten Werte des Periodensystems der IUPAC verwendet.

4.6.2 Die Oxidationsstufen der virtuellen Atome

Der erweiterte Syntax des SMILES - 3D Konzeptes ermöglicht dem Anwender, die Oxidationsstufe des Atoms in der linearen Zeichenfolge optional zu spezifizieren. Die Oxidationsstufe des Atoms wird in der linearen SMILES Zeichenfolge durch römische Ziffern dargestellt.

Zur Integration der Oxidationsstufen in die Analyse- und Interpretationsprozesse ist es erforderlich, die zulässigen Oxidationsstufen für jedes virtuelle Atom festzulegen.

Die Oxidationsstufen des virtuellen Atoms sind in der objektartigen Struktur vom Typ AtomTable für jedes Atom definiert. In der objektartigen Struktur vom Typ AtomTable werden die Werte der Oxidationsstufen durch eine Zeichenfolge von Buchstaben dargestellt. Die alphabetische Zeichenfolge wird durch entsprechende Methoden, die in das Objekt vom Typ CAtomTable implementiert sind, interpretiert und gegebenenfalls in die entsprechenden römischen Ziffern oder in natürliche ganze Zahlen konvertiert.

Im Anhang in Kapitel 8.5 sind in einem Periodensystem die zulässigen Oxidationsstufen der Atome übersichtlich dargestellt.

4.6.3 Die Zelltypen der virtuellen Atome

Die Generierung der dreidimensionalen Struktur der Moleküle wird durch die Integration von strukturbildenden Zellen erreicht. Die verschiedenen strukturbildenden Zelltypen realisieren die räumliche Orientierung der virtuellen Bindungen und der freien Elektronenpaare um das virtuelle Atom, das im Mittelpunkt der Zelle platziert ist.

In die Software WinSmiles - 3D sind sechs verschiedene Zelltypen zum Aufbau der dreidimensionalen Struktur implementiert. Bei dem Aufbau der virtuellen Molekülstruktur aus den virtuellen Atomen wird jedem Atom, in Abhängigkeit von der Anzahl der Valenzorbitale und der Anzahl der Valenzelektronen, der entsprechende Zelltyp zugewiesen. Um die möglichst realistische Darstellung der virtuellen Molekülstruktur zu erreichen, ist es sinnvoll, für jedes Atom die möglichen Zelltypen festzulegen. Daher sind für jedes Atom, entsprechend dem chemischen Reaktionsverhalten, ein standardmäßig zu verwendender Zelltyp festgelegt und darüber hinaus die weiteren zur Erzeugung der dreidimensionalen Struktur verwendbaren möglichen Zelltypen eines Atoms explizit definiert. Der standardmäßig einem virtuellen Atom zugewiesene Zelltyp entspricht der am häufigsten vorkommenden Koordination des Atoms.

Die möglichen Zelltypen eines Atoms sind in der objektartigen Struktur vom TypAtomTable definiert. In der objektartigen Struktur werden die möglichen Zelltypen

durch eine alphabetische Zeichenfolge dargestellt. Dabei werden die verschiedenen Zelltypen, entsprechend der Anzahl der Bindungsstellen einer Zelle, durch die Buchstaben von a bis f dargestellt. Die Funktionalität zur Konvertierung und Integration der Zeichenfolge in die virtuellen Atome ist in das Objekt vom Typ CAtomTable implementiert.

Die nachfolgende Tabelle 4-1 zeigt die Zelltypen, die den virtuellen Atomen standardmäßig zugewiesen werden. Der Standardzelltyp kann durch die Angabe der Oxidationsstufe modifiziert werden.

Zelltyp	Elemente
Monohedral	1. und 11. Gruppe
Dihedral	2., 10. und 12. Gruppe
Trigonal	3. 9. und 13. Gruppe, Lanthanide, Actinide
Tetrahedral	4., 8 und 14. – 17. Gruppe
Pentagonal	5. und 7. Gruppe
Octahedral	6. Gruppe

Tabelle 4-1: Standardzelltypen der virtuellen Atome

Die darüber hinaus möglichen Zelltypen eines Atoms sind im Anhang in Kapitel 8.6 in einem Periodensystem dargestellt.

4.6.4 Die Darstellungsfarben der virtuellen Atome

Zur einheitlichen Visualisierung der virtuellen Atome ist es erforderlich, die standardmäßig zur Darstellung der Atome und Bindungen verwendeten Farben zu definieren.

Die Farben der visualisierten Atome sind in Anlehnung an das 1951 von Corey und Pauling entwickelte[64], und später von Kultun[65] weiterentwickelte Kunststoffkugelmodell festgelegt. Die virtuellen Atome werden durch entsprechend

farbige Linien oder Kugeln dargestellt. In Abhängigkeit von dem Darstellungsmodus werden die ebenfalls visualisierten Bindungen zwischen zwei Atomen anteilig des kovalenten Radius der virtuellen Atome durch die gleichen Farben dargestellt.

Die Farbwerte der virtuellen Atome sind durch die RGB-Tripelwerte in der objektartigen Struktur vom Typ AtomTable für jedes Atom definiert. Eine kurze Übersicht über die Farbwerte der gebräuchlichsten Atome ist nachfolgend in Tabelle 4-2 gezeigt.

Element	Darstellungsfarbe	Rotanteil	Grünanteil	Blauanteil
Wasserstoff	Weiß	255	255	255
Kohlenstoff	Grau	200	200	200
Sauerstoff	Rot	240	0	0
Schwefel	Gelb	255	200	50
Stickstoff	Hell blau	143	143	255
Phosphor	Orange	255	165	0
Fluor	Goldrot	218	165	32
Chlor	Grün	0	255	0
Brom	Braun	165	42	42

Tabelle 4-2: Farbschema der virtuellen Atome

Die standardmäßig zugewiesenen Farbwerte aller Atome sind im Anhang in Kapitel 8.7 in einem Periodensystem gezeigt.

4.6.5 Die kovalenten Radien der virtuellen Atome

Zum Aufbau der dreidimensionalen virtuellen Molekülstruktur ist, neben der räumlichen Orientierung der Bindungsstellen, der Abstand zwischen den miteinander verbundenen virtuellen Atomen von entscheidender Bedeutung. Die Bindungslänge zwischen zwei verbundenen Atomen wird in der Software WinSmiles - 3D aus der Summe der kovalenten Atomradien der miteinander verbundenen virtuellen Atome berechnet. Der kovalente Radius eines Atoms ist abhängig von der Oxidationsstufe und der räumlichen Orientierung der Bindungsorbitale des Atoms; es ist daher sinnvoll die kovalenten Radien der virtuellen Atome in Abhängigkeit von der

Oxidationsstufe und der verwendeten Zelle, zur Darstellung der dreidimensionalen Struktur zu definieren.

In der objektorientierten Programmstruktur der Software WinSmiles - 3D ist die Funktionalität zur Ermittlung des kovalenten Atomradius in Abhängigkeit von der Oxidationsstufe und dem Zelltyp in die Klasse KovRad implementiert.

Die Werte des kovalenten Radius sind für die Oxidationsstufen des Atoms in ein strukturartiges Objekt vom Typ KovRadTable abgekapselt. Die Werte der definierten kovalenten Radien der Atome sind in einer statischen globalen Membervariablen, die aus den objektartigen Strukturen vom Typ KovRadTable besteht, gespeichert und werden durch die Funktionalität der Klasse KovRad in die virtuellen Atome integriert.

Die in die Software WinSmiles - 3D implementierten Werte der kovalenten AtomRadien sind im Anhang in Kapitel 8.8 in einer Tabelle aufgeführt. Den Atomen, deren kovalenter Radius nicht in der kovalenten Radientabelle definiert ist, wird der kovalente Radius von 76,6 pm zugewiesen.

5. Die Anwendung der Software WinSmiles - 3D

In den nachfolgenden Kapiteln wird die Anwendung der weiterentwickelten Software WinSmiles - 3D an unterschiedlichen Molekülen demonstriert und die programmtechnische Realisation des SMILES - 3D Konzeptes kontrolliert.

5.1 Codierung und Darstellung linearer Molekülstrukturen

In diesem Kapitel wird die Codierung von linearen, nicht verzweigten oder cyclischen Molekülstrukturen erläutert und die Umsetzung in dem Programm WinSmiles - 3D beschrieben. Die im folgenden beschriebenen Moleküle sind organische und anorganische Verbindungen, die exemplarisch ausgewählt sind und deren lineare Notation in das Programm eingegeben worden ist.

Die linearen Alkane können in der SMILES Eingabe durch zwei unterschiedliche Zeichenfolgen codiert werden. Die lineare Notation besteht entweder aus der entsprechenden Anzahl an Kohlenstoffsymbolen oder aus einem Kohlenstoffsymbol multipliziert mit der Zahl, die der Anzahl der Kohlenstoffatome entspricht. Die Verbindung Propan kann also durch die Zeichenfolge "CCC" oder "C*3" codiert werden. Unabhängig von der Form der Codierung generiert die Software aus beiden Zeichenfolgen die virtuelle Molekülstruktur der linearen Alkane. In der virtuellen Molekülstruktur wird den sp^3-hybridisierten Kohlenstoffatomen jeweils eine tetraedrische Zelle zugewiesen und die Zellen zum Aufbau der dreidimensionalen Struktur standardmäßig staggered zueinander an der CC-Bindung verbunden, so dass eine lange Kohlenstoffkette erzeugt wird. Die nicht explizit codierten Wasserstoffatome der Alkane werden automatisch in der Molekülstruktur ergänzt und den terminalen Wasserstoffatomen eine monohedrale Zelle zugewiesen. Bei der Berechnung der Atomkoordinaten wird die Länge der CC-Einfachbindung aus der Summe der kovalenten Atomradien der miteinander verbundenen Kohlenstoffatome bestimmt und die Bindung mit einer Bindungslänge von 152,6 pm simuliert. Entsprechend Analog wird die Länge der CH-Bindung aus der Summe des kovalenten Radius des Kohlenstoffatoms und des kovalenten Radius des Wasserstoffatoms berechnet und die Bindung mit einer Länge von 110,6 pm simuliert. Bei der Visualisierung wird das Alkan standardmäßig durch graue

Kohlenstoffatome und weiße Wasserstoffatome dargestellt und, unter Berücksichtigung der zuvor berechneten Atomkoordinaten, auf dem Bildschirm in einer dreidimensional wirkenden Darstellung abgebildet. In Abbildung 5-1 ist die grafische Benutzeroberfläche des Programms WinSmiles - 3D bei der Berechnung der Struktur von Propan abgebildet.

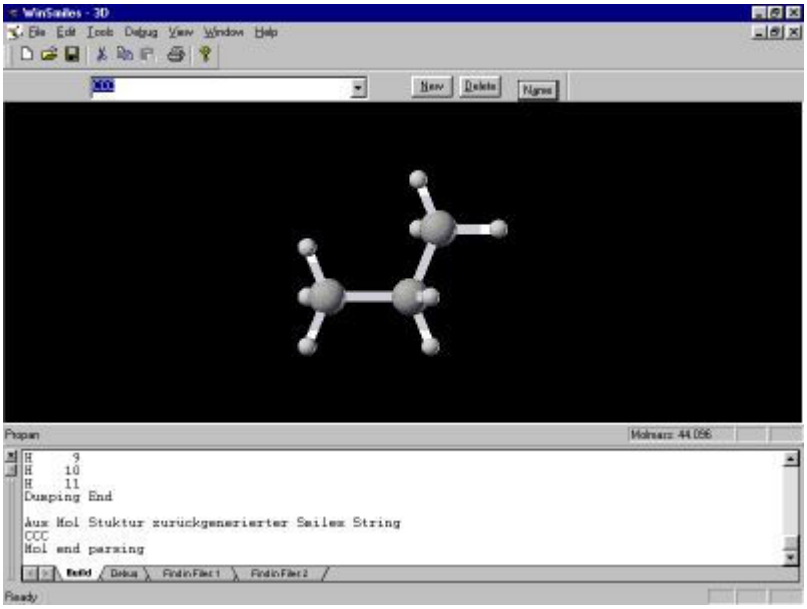


Abbildung 5-1: Darstellung von Propan durch das Programm WinSmiles - 3D

Der weitere Aufbau der virtuellen Molekülstruktur von längeren Kohlenstoffketten ist nachfolgend am dem Alkan Butan veranschaulicht. Die Verbindung Butan kann unter anderem durch die linearen Notationen "CCCC" oder "C*4" in der SMILES - 3D Notation codiert werden. Das Programm generiert aus den unterschiedlichen Eingaben die virtuelle Molekülstruktur des Butans und weist in Analogie zu dem Propan den Kohlenstoffatomen eine tetraedrische Zelle zu.

Zum Aufbau der dreidimensionalen Struktur der Kohlenstoffkette wird die erste Bindungsstelle der Zelle des Add-Atoms mit der zweiten Bindungsstelle der Zelle des From-Atoms verbunden und die Bindungsstellen der Zellen stagged zueinander ausgerichtet. Das Programm ordnet die Kohlenstoffatome alle in einer Ebene an, so dass eine lange Kohlenstoffkette erzeugt wird. Die übrigen Bindungsstellen der tetraedrischen Zellen werden mit der Bindungsstelle der monohedralen Zellen der automatisch ergänzten Wasserstoffatome verbunden.

Zur Veranschaulichung der erzeugten Molekülstruktur ist in Abbildung 5-2 die grafische Benutzeroberfläche des Programms WinSmiles - 3D bei der Auswertung der linearen Notation "C*4" dargestellt.

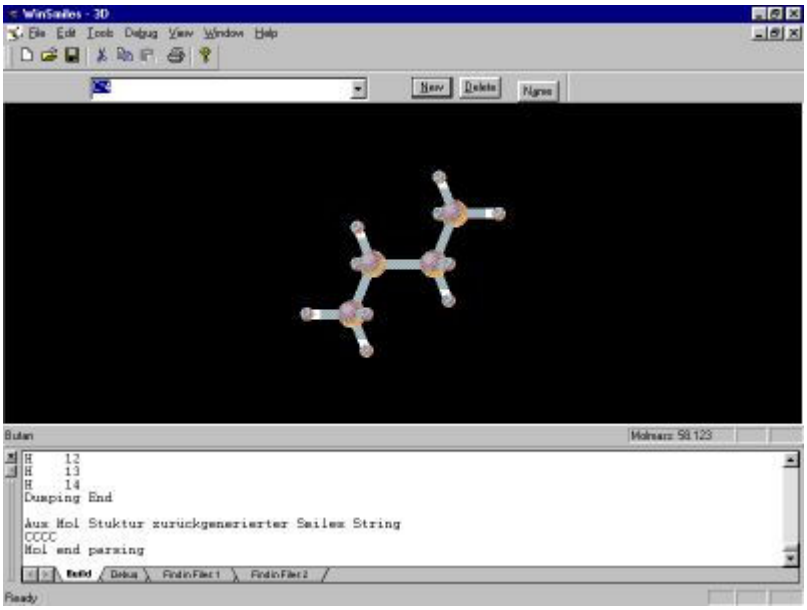


Abbildung 5-2: Darstellung von Butan durch das Programm WinSmiles - 3D

Die Software WinSmiles - 3D kann lineare Alkane bis zu einer Kettenlänge von 121 Kohlenstoffatomen berechnen. Allerdings werden die visualisierten Molekülstrukturen von langen Alkanketten sehr komprimiert dargestellt und die strukturellen Details können insofern nur auszugsweise vergrößert auf dem Bildschirm abgebildet werden.

Die automatische Generierung der linearen Notation erzeugt aus der virtuellen Molekülstruktur des Propans und des Butans, unabhängig von der Eingabe, die entsprechende Anzahl an Kohlenstoffsymbolen. Der optional mögliche Multiplikator der Kohlenstoffatome wird bei der automatischen Generierung der linearen Notation nicht erzeugt.

Die Anordnung der Bindungsstellen zweier tetraedrischer Zellen zueinander und die Reihenfolge in der die Bindungsstellen in der linearen Notation dargestellt werden ist nachfolgend an der Verbindung 2-Chlor-1-fluorethan erläutert. Beim Aufbau der Molekülstruktur werden die tetraedrischen Zellen der Kohlenstoffatome entsprechend dem SMILES - 3D Konzept stagged miteinander verbunden und die

erste Bindungsstelle der Zelle des From-Atoms trans zu der zweiten Bindungsstelle der Zelle des To-Atoms orientiert.

Die Verbindung kann unter anderem durch die Zeichenfolge "FCCCl" codiert werden. Beim Aufbau der dreidimensionalen Struktur positioniert diese Notation das Fluoratom an die erste Bindungsstelle der Zelle des ersten Kohlenstoffatoms, sowie das Chloratom an die zweite Bindungsstelle der Zelle des zweiten Kohlenstoffatoms in der linearen Zeichenfolge. Hieraus resultiert, dass die Kohlenstoffatome und die Halogenatome in einer Ebene angeordnet werden und das Fluoratom im Bezug auf die CC-Bindung trans zum Chloratom orientiert ist.

Durch die explizite Angabe von Wasserstoffatomen in der linearen Notation kann die Reihenfolge bei der Besetzung der Bindungsstellen der Zelle variiert und die Anordnung der Halogenatome zueinander verändert werden. Die lineare SMILES - 3D Notation "FCCHCl" codiert ebenfalls die Verbindung 2-Chlor-1-fluorethan. Beim Aufbau der Molekülstruktur positioniert diese Notation das Fluoratom an der ersten Bindungsstelle der Zelle des ersten Kohlenstoffatoms und das Chloratom an die dritte Bindungsstelle der Zelle des zweiten Kohlenstoffatoms in der linearen Notation. Die zweite Bindungsstelle der Zelle des zweiten Kohlenstoffatoms wird mit dem explizit in der Zeichenfolge angegebenen Wasserstoffatom besetzt. Diese Besetzung der Bindungspositionen ergibt die in Abbildung 5-3 dargestellte Anordnung der Atome.

Des weiteren kann die Verbindung durch die Zeichenfolge "FCCHHCl" codiert werden. Diese Notation platziert beim Aufbau der dreidimensionalen Struktur das Fluoratom an die erste Bindungsstelle der Zelle des ersten Kohlenstoffatoms und das Chloratom an die vierte Bindungsstelle der Zelle des zweiten Kohlenstoffatoms. Die zweite und dritte Bindungsstelle der Zelle werden mit den explizit in der Notation angegebenen Wasserstoffatomen besetzt. Zur Veranschaulichung der unterschiedlichen Notationen sind in der nachfolgenden Abbildung 5-3 die entsprechenden dreidimensionalen Molekülstrukturen dargestellt.

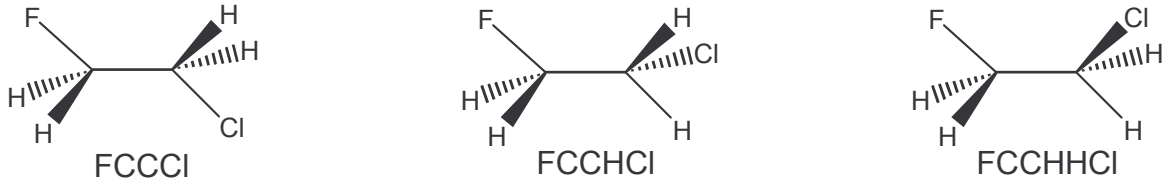
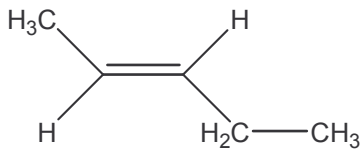


Abbildung 5-3: Modifikation der dreidimensionalen Struktur durch die explizite Angabe von Wasserstoffatomen

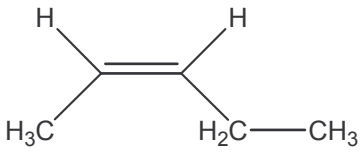
Die linearen Alkene werden in Analogie zu den linearen Alkanen durch die Notation der Alkylketten und durch die explizite Angabe der Doppelbindung durch das Symbol = codiert und in die Anwendung eingegeben. Die Verbindung 2-Penten kann also entweder durch die Zeichenfolge "CC=CCC", oder durch die Zeichenfolge "CC=C*3" in die Anwendung eingegeben werden. Die Software WinSmiles - 3D realisiert aus den unterschiedlichen linearen SMILES Eingaben den Aufbau der linearen Alkenkette. Zur Berechnung der Atomkoordinaten wird den sp^3-hybridisierten Kohlenstoffatomen eine tetraedrische Zelle und den sp^2-hybridisierten Kohlenstoffatomen der CC-Doppelbindung eine trigonale Zelle zugewiesen. Die Bindungsstellen der Zelle der Kohlenstoffatome der Doppelbindung werden in einer Ebene zueinander angeordnet und ein Bindungswinkel von 120° zwischen den Bindungsstellen des Atoms simuliert. Die Alkylketten werden standardmäßig an der Doppelbindung trans zueinander angeordnet.

Die cis Anordnung der Alkylketten erfolgt entweder durch die explizite Zuweisung der Wasserstoffatome an die entsprechenden Bindungsstellen der sp^2-hybridisierten Kohlenstoffatome in der linearen Notation, oder durch die optionale Spezifikation des Dihedralwinkels der CC-Doppelbindung. Die Verbindung cis 2-Penten kann in der linearen SMILES - 3D Notation sowohl durch die Zeichenfolge "CC=CHCC", als auch durch die Zeichenfolge "CC[180 =]CCC" codiert und die Molekülstruktur in dem Programm entsprechend berechnet werden. In Abbildung 5-4 ist die unterschiedliche Codierung der dreidimensionalen Struktur von cis- und trans-2-Penten dargestellt.



SMILES: CC=CCC

trans - Form



SMILES: CC=CHCC

cis - Form

Abbildung 5-4: Codierung von cis- und trans-2-Penten

In Analogie zum Aufbau der Molekülstruktur der Alkane werden die Zellen der sp^3-hybridisierten Kohlenstoffatome an der CC-Bindung stagged zueinander angeordnet, so dass eine lange Alkylkette entsteht. Die freien Bindungsstellen des codierten Alkens werden in der virtuellen Molekülstruktur automatisch mit terminalen Wasserstoffatomen besetzt. In dem Programm wird die CC-Doppelbindung bei der Berechnung der Atomkoordinaten mit einer Bindungslänge von 149,4 pm simuliert.

Die Berechnung von Molekülstrukturen mit mehreren Doppelbindungen ist problemlos möglich. Bei einer Alkenstruktur wird den sp-hybridisierten Kohlenstoffatomen eine dihedrale Zelle zugewiesen und die benachbarten Doppelbindungen als lineare Struktureinheiten berechnet.

Das Programm erzeugt automatisch die lineare SMILES Zeichenfolge der codierten Alkene durch die entsprechende Anzahl an Kohlenstoffsymbolen und dem Zeichen = an der entsprechenden Bindungsposition, unabhängig von der Eingabe.

Die linearen Alkine werden in Analogie zu den Alkenen codiert und die Dreifachbindung durch das Zeichen # in die Software eingegeben. Den sp-hybridisierten Kohlenstoffatomen wird eine dihedrale Zelle zugeordnet und die Zellen an der CC-Dreifachbindung linear zueinander mit einem Bindungswinkel von 180° miteinander verbunden. Bei der Berechnung der dreidimensionalem virtuellen Molekülstruktur wird die CC-Dreifachbindung mit einer Bindungslänge von 142,0 pm berechnet.

Die Software WinSmiles - 3D erzeugt bei der automatischen Generierung der linearen SMILES Zeichenfolge die Alkine durch die entsprechende Reihenfolge der

Kohlenstoffsymbole und das Dreifachbindungssymbol # an der entsprechenden Bindungsposition in der Notation.

In der organischen Chemie sind die Kohlenstoffketten häufig durch unterschiedliche Gruppen funktionalisiert. Die einfachste Modifikation einer virtuellen Alkankette ist die Integration von terminalen Atomen, z. B. Fluor oder Chlor in die Struktur. Im folgenden ist die Darstellung der Verbindung 1,2-Difluorethan in dem Programm WinSmiles - 3D erläutert.

Das Molekül wird in der linearen SMILES - 3D Notation durch die lineare Zeichenfolge "FCCF" codiert und entsprechend in die Software eingegeben. Das Programm erzeugt aus der Zeichenfolge zwei miteinander verbundene sp^3-hybridisierte Kohlenstoffatome, deren räumliche Orientierung der Bindungsstellen durch zwei tetraedrische Zellen, die stagged miteinander verbunden sind, dargestellt wird. Bei der Berechnung der Atomkoordinaten werden die beiden Fluoratome, entsprechend der Konvention zur Zellverknüpfung in einer Ebene und im Bezug zu der CC-Bindung trans zueinander angeordnet. Eine andere Anordnung der Atome zueinander kann durch die optionale Spezifikation des dihedralen Bindungswinkels erreicht werden. Die räumliche Orientierung der Bindungsstellen des Fluoratoms wird ebenfalls durch eine tetraedrische Zelle realisiert. Die erste Bindungsstelle der Zelle des Fluoratoms wird mit der Zelle des Kohlenstoffatoms verbunden. Die übrigen drei Bindungsstellen der Zelle des Fluoratoms werden mit den freien Valenzelektronen des Fluors besetzt, so dass bei der Visualisierung der dreidimensionalen Struktur diese Bindungsstellen nicht dargestellt werden. Bei der Berechnung der Atomkoordinaten wird die CF-Bindung mit einer Bindungslänge von 141,3 pm berechnet. Das Programm visualisiert das Fluoratom in der dreidimensionalwirkenden Abbildung standardmäßig durch eine farblich grüne Atomdarstellung.

Bei der automatischen Generierung der linearen SMILES Zeichenfolge erzeugt das Programm, ausgehend vom ersten Fluoratom der virtuellen Molekülstruktur, das Fluorsymbol sowie die Atomsymbole der Kohlenstoffatome, und beendet die Erzeugung der Zeichenfolge mit dem Atomsymbol des zweiten Fluoratoms.

Des weiteren sind in der organischen Chemie eine Vielzahl an Verbindungen bekannt, die Sauerstoffatome enthalten. Im folgenden wird die Codierung und die Berechnung der Molekülstruktur der allgemein bekannten Verbindungen Ethanol (Hydroxyethan) und Diethylether (Ethoxyethan) veranschaulicht.

Die Verbindung Ethanol wird in der SMILES - 3D Notation unter anderem durch die Zeichenfolge „CCO“ codiert und entsprechend in die Software eingegeben. Das Programm erzeugt aus der Eingabe das dreidimensional wirkende Abbild der Molekülstruktur von Ethanol. Zum Aufbau der dreidimensionalen Struktur weist das Programm den sp^3-hybridisierten Kohlenstoffatomen und dem Sauerstoffatom jeweils eine tetraedrische Zelle zu. Die Zellen werden so miteinander verbunden, dass die Bindungsstellen der Zellen stagged zueinander orientiert sind und eine längliche Atomkette entsteht. Zwei Bindungsstellen der tetraedrischen Zelle des Sauerstoffatoms sind mit den freien Elektronenpaaren besetzt und werden in der Abbildung nicht explizit visualisiert. Das Wasserstoffatom der Hydroxylgruppe wird, wie die übrigen Wasserstoffatome in der virtuellen Molekülstruktur, automatisch ergänzt. Bei der Berechnung der dreidimensionalen Struktur wird die CO-Bindungslänge mit 144,9 pm und die OH-Bindungslänge mit 102,9 pm angenommen. In der visualisierten Darstellung wird das virtuelle Sauerstoffatom farblich rot dargestellt.

Das Programm erzeugt automatisch die SMILES - 3D Notation aus der virtuellen Molekülstruktur, indem es die Atomsymbole der Kohlenstoffatome und des Sauerstoffatoms erzeugt und die Zeichen in der entsprechenden Reihenfolge ausgibt.

Eine weitere allgemein bekannte und nicht verzweigte Verbindung ist der Diethylether, der durch die Zeichenfolge „CCOCC“ codiert werden kann. Die Software WinSmiles - 3D konvertiert diese Zeichenfolge in die entsprechende dreidimensionale virtuelle Molekülstruktur. Bei der Erzeugung dieser virtuellen Struktur wird den sp^3-hybridisierten Kohlenstoffatomen und dem Sauerstoffatom von dem Programm jeweils eine tetraedrische Zelle zugewiesen und die Bindungsstellen der miteinander verbundenen Zellen stagged zueinander angeordnet, so dass eine längliche Molekülstruktur realisiert wird. Die freien Elektronenpaare des

Sauerstoffatoms werden auf die dritte bzw. vierte Bindungsstelle der tetraedrischen Zelle des Sauerstoffatoms verteilt und nicht visualisiert. Hieraus resultiert am virtuellen Sauerstoffatom ein COC-Bindungswinkel von 109°; die CO-Bindung wird mit einer Länge von 144,9 pm simuliert. Die freien Bindungsstellen der Zellen der virtuellen Molekülstruktur werden mit den monohedralen Zellen der in der Molekülstruktur ergänzten Wasserstoffatome verbunden.

Das Programm WinSmiles - 3D erzeugt aus der virtuellen Struktur des Diethylethers die lineare Notation “CCOCC“, was der ursprünglichen Eingabe entspricht.

Die dreidimensionale Struktur von linearen Molekülen, in denen das Sauerstoffatom durch ein höheres homologes Element (z.B. Schwefel) der VI. Hauptgruppe substituiert ist, werden entsprechend analog berechnet.

Verschiedene anorganische Verbindungen können ebenfalls durch eine lineare Notation codiert und von dem Programm WinSmiles - 3D simuliert werden.

Die Verbindung Magnesiumoxid wird durch die lineare SMILES - 3D Notation Mg=O codiert. Das Programm berechnet aus der linearen Zeichenfolge die virtuelle Molekülstruktur. Bei der Berechnung der Struktur wird dem Magnesiumatom eine monohedrale Zelle und dem sp^2-hybridisierten Sauerstoffatom eine trigonale Zelle, zur Simulation der räumlichen Orientierung der Bindungsstellen, zugewiesen. Die von dem Programm visualisierte Struktur erscheint als lineares Molekül, da die zweite bzw. die dritte Bindungsstelle der Zelle des Sauerstoffatoms mit den freien Valenzelektronenpaaren des Sauerstoffatoms besetzt sind und diese Bindungsstellen bei der Visualisierung nicht abgebildet werden. Bei der Berechnung der Atomkoordinaten wird die Doppelbindung mit einer Bindungslänge von 191,0 pm angenommen. Das Magnesiumatom wird bei der Visualisierung von dem Programm standardmäßig in der Farbe Dunkelgrün dargestellt.

Die automatische Generierung der Notation erzeugt die Zeichenfolge aus den Atomsymbolen und dem Doppelbindungssymbol in der entsprechenden Reihenfolge unabhängig von der Eingabe.

Die Hydride der chemischen Elemente werden in der SMILES - 3D Notation durch das jeweilige Atomsymbol codiert, da die Wasserstoffatome in der virtuellen Molekülstruktur automatisch ergänzt werden. Das Magnesiumhydrid wird durch die Zeichen „Mg“ codiert und die Molekülstruktur berechnet. Zum Aufbau der Struktur wird dem Magnesiumatom eine dihedrale Zelle zugewiesen und die Bindungsstellen der Zelle mit der Bindungsstelle der monohedralen Zellen der Wasserstoffatome verbunden. Das visualisierte Magnesiumhydrid wird als lineares Molekül mit einem Bindungswinkel von 180° dargestellt und die Bindungslänge mit 170,3 pm simuliert.

Die automatische Generierung des Atomsymbols ist für alle Hydride realisiert.

Die virtuelle Molekülstruktur von Magnesiumchlorid wird unter anderem durch die Zeichenfolge ClMgCl codiert. In Analogie zu dem Magnesiumhydrid wird dem virtuellem Magnesiumatom eine dihedrale Zelle zugewiesen. Bei dem Aufbau der Struktur werden die Bindungsstellen der dihedralen Zelle jeweils mit der ersten Bindungsstelle der tetraedrischen Zelle der Chloratome verbunden. Hierdurch wird das Magnesiumchlorid als lineares Molekül dargestellt und die Bindung mit einer Länge von 233,4 pm simuliert.

Das Programm WinSmiles - 3D erzeugt automatisch die Atomsymbole der Verbindung als lineare Zeichenfolge.

5.2 Codierung und Darstellung verzweigter Molekülstrukturen

In diesem Kapitel wird, im Sinne des SMILES - 3D Konzeptes, die Codierung von verzweigten, nicht cyclischen Molekülstrukturen erläutert und die Umsetzung in der Software WinSmiles - 3D überprüft. Nachfolgend sind verschiedene Verbindungen aus der organischen und anorganischen Chemie exemplarisch ausgewählt und die linearen Notationen dieser Verbindungen in das Programm WinSmiles - 3D eingegeben worden.

Der tertiäre Alkohol *tert*.-Butanol (2-Methyl-2-propanol) kann unter anderem durch die lineare Zeichenfolge „CC(C)(C)O“ codiert und die virtuelle Molekülstruktur durch die entsprechende Eingabe in das Programm erzeugt werden. Bei der Berechnung der dreidimensionalen Struktur weist das Programm allen sp^3-hybridisierten Atomen eine tetraedrische Zelle zu und verbindet die Zellen unter Berücksichtigung der Bindungsverhältnisse miteinander. Das Programm beginnt dabei mit der Zelle des Atoms, das durch das erste Zeichen der linearen Notation codiert ist, und verbindet es mit der Zelle des Atoms, das durch das nächste Zeichen der linearen Notation codiert ist. Im weiteren Ablauf verbindet das Programm die freien Bindungsstellen des ersten Kohlenstoffatoms mit den terminalen Zellen der Wasserstoffatome und berechnet in nächsten Schritt die Bindungen des zweiten Kohlenstoffatoms. Bei der Berechnung der Bindungen des zweiten Kohlenstoffatoms wird zunächst die Bindung zu dem Kohlenstoffatom, das durch das dritte, in runden Klammern stehende Atomsymbol codiert ist, aufgebaut. Anschließend werden die Bindungen des dritten Kohlenstoffatoms zu den terminalen Wasserstoffatomen realisiert, bevor die weiteren Bindungen des zweiten Kohlenstoffatoms berechnet werden.

Dieser Vorgang wird für alle weiteren Atome der linearen Notation wiederholt bis sämtliche Bindungen der Molekülstruktur berechnet sind.

Beim Aufbau der virtuellen Molekülstruktur eines verzweigten Moleküls werden die linearen Seitenketten an einer Verzweigung immer zunächst vollständig berechnet, bevor die Berechnung an der Verzweigungsstelle fortgesetzt wird.

In Abbildung 5-5 ist die grafische Benutzeroberfläche der Software WinSmiles - 3D abgebildet und die Visualisierung von *tert*.-Butanol gezeigt.

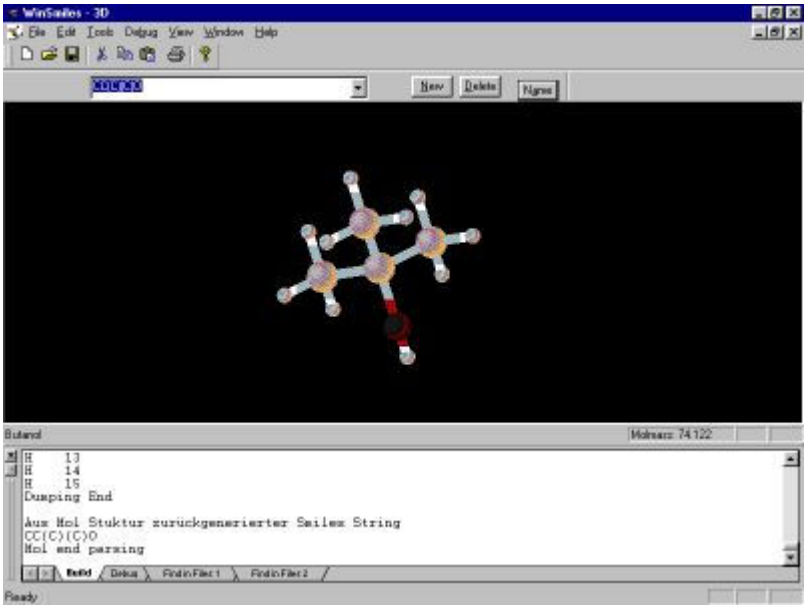


Abbildung 5-5: Visualisierung von *tert.*-Butanol in dem Programm WinSmiles - 3D

Die essentielle Aminosäure Alanin ist, im Sinne der SMILES - 3D Notation, ein weiteres Beispiel für eine verzweigte, nicht cyclische Struktur. Die Verbindung L-Alanin kann durch die lineare Notation „CC(N)C=OO“ und die Verbindung R-Alanin durch die Zeichenfolge "CC(N)HC=OO" codiert werden. In Abbildung 5-6 sind die L- und die R-Form des Alanin dargestellt.

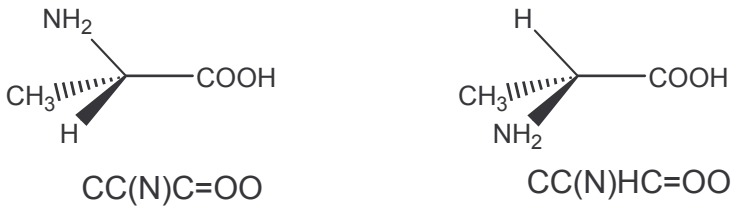


Abbildung 5-6: Struktur von L- und R-Alanin

Durch die Eingabe der unterschiedlichen Zeichenfolgen in das Programm wird unter Berücksichtigung der Stereoisomerie die virtuelle Molekülstruktur generiert und die dreidimensional wirkende Darstellung von Alanin erzeugt. Zum Aufbau der Struktur wird den sp^3-hybridisierten Kohlenstoffatomen, dem Stickstoffatom und dem Sauerstoffatom der Hydroxylgruppe eine tetraedrische Zelle zugewiesen. Des

weiteren ordnet das Programm dem sp^2-hybridisierten Kohlenstoffatom und dem Sauerstoffatom der Carbonylgruppe eine trigonal planare Zelle zu. Die Zellen der virtuellen Atome werden entsprechend der Bindungsverhältnisse und der Konvention zur Zellverknüpfung miteinander verbunden und bei der Berechnung der Atomkoordinaten eine CN-Bindungslänge von 145,8 pm, sowie eine NH-Bindungslänge von 103,8 pm angenommen. Standardmäßig wird das Stickstoffatom bei der Visualisierung in blau dargestellt.

Die automatische Generierung der linearen SMILES Zeichenfolge erzeugt aus der virtuellen Molekülstruktur der Aminosäure Alanin die ursprünglich eingegebene Zeichenfolge.

Die Borsäure kann, entsprechend des SMILES - 3D Konzeptes, entweder durch die Zeichenfolge „B(O)(O)O“ oder alternativ durch die Zeichenfolge „OB(O)O codiert werden. Die linearen Zeichenfolgen der Borsäure verdeutlichen, dass die Notationen, im Sinne des SMILES - 3D Konzeptes, eine verzweigte Molekülstruktur codieren.

Zum Aufbau der dreidimensionalen Struktur wird dem Boratom eine trigonal-planare Zelle und den Sauerstoffatomen eine tetraedrische Zelle zugewiesen. Die Zellen werden an den Bindungsstellen entsprechend den Bindungsverhältnissen und unter Berücksichtigung des dihedralen Winkels miteinander verbunden. Die Wasserstoffatome werden in der virtuellen Molekülstruktur automatisch ergänzt und den terminalen Atomen eine monohedrale Zelle zugewiesen. Bei der Berechnung der Atomkoordinaten wird die BO-Bindung mit einer Bindungslänge von 151,6 pm und die OH-Bindung mit einer Bindungslänge von 102,9 pm simuliert. Bei der Visualisierung der Molekülstruktur wird das Boratom standardmäßig in der Farbe hell grün dargestellt.

In Abhängigkeit vom Anfangsatom bei dem Aufbau der virtuellen Molekülstruktur werden die beiden unterschiedlichen linearen Notationen der Borsäure automatisch generiert.

5.3 Codierung und Darstellung von cyclischen Molekülstrukturen

In diesem Kapitel wird die Codierung von cyclischen Molekülstrukturen beschrieben und die Erzeugung der virtuellen Molekülstruktur in der Software erläutert.

In der linearen SMILES - 3D Notation werden cyclische Molekülstrukturen durch die formale Spaltung einer Ringbindung auf eine lineare Struktur reduziert und die gespaltene Ringbindung durch die Verwendung des Zeichens % und einen nachfolgenden numerischen Bindungsindex in der SMILES - 3D Notation codiert.

Die dreidimensionale Struktur der kleinen und sehr gespannten Ringsysteme, wie zum Beispiel die Molekülstrukturen von Cyclopropan oder von Cyclobutan, können nicht richtig codiert werden. Die Ringbindungswinkel der sp^3-hybridisierten Kohlenstoffatome weichen deutlich von dem tetraedrischen Winkel ab. In der dreidimensionalen Struktur von Cyclopropan beträgt der Ringbindungswinkel 60° und in der Struktur von Cyclobutan beträgt der Ringbindungswinkel 88.5°.[66] Zur Darstellung von derartigen räumlichen Anordnungen der Bindungen sind keine entsprechenden Zelltypen in das SMILES - 3D Konzept implementiert.

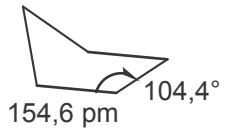
Die Molekülstruktur von Cyclopentan wird unter anderem durch die Zeichenfolge %1C[150 -]C[180 -]C[210 -]C[-]CH%1 codiert. Das Programm WinSmiles - 3D berechnet aus der Zeichenfolge die dreidimensionale virtuelle Molekülstruktur und simuliert die CC-Bindung mit einer Länge von 152,6 pm, sowie die CH-Bindungen mit einer Länge von 110,6 pm. Zur Generierung der dreidimensionalen Struktur wird den sp^3-hybridisierten Kohlenstoffatomen eine tetraedrische Zelle und den Wasserstoffatomen eine monohedrale Zelle zugewiesen.

Zum Aufbau der cyclischen Struktur berechnet das Programm zunächst alle Bindungen des virtuellen Atoms, das durch das erste Atomsymbol in der linearen Notation codiert ist, mit Ausnahme der Bindung, die mit dem Zeichen % gekennzeichnet ist. Die Bindungsposition wird an dem virtuellen Atom zunächst nicht besetzt und die Bindungsstelle durch den Index der cyclischen Ringbindung markiert. Zum weiteren Aufbau der dreidimensionalen Struktur berechnet das Programm die

Bindungen der virtuellen Atome, die in der linearen Notation durch die nachfolgenden Atomsymbole codiert sind und verbindet die Bindungsstellen der entsprechenden Zellen unter Berücksichtigung der Bindungsverhältnisse miteinander. Dieser Vorgang wird solange wiederholt bis das Programm die Bindungsstelle ermittelt, die durch den gleichen cyclischen Ringbindungsindex gekennzeichnet ist wie die Ringbindung des ersten Atoms und erzeugt die cyclische Struktur durch Verbinden dieser beiden Atome an den entsprechenden Bindungsstellen. Abschließend werden die übrigen Bindungen der weiteren virtuellen Atome berechnet und die virtuelle Molekülstruktur weiter aufgebaut.

Diese Vorgehensweise beim Aufbau der dreidimensionalen Molekülstruktur macht deutlich, dass die Ringstruktur durch den Anwender in der linearen Notation codiert werden muss und dass das Programm die Atome nicht automatisch cyclisch anordnet.

Die Ringstruktur des Cyclopentan wird durch die explizite Angabe des von dem Standardwert abweichenden Dihedralwinkels der CC-Bindung realisiert und die tetraedrischen Zellen jeweils um den definierten Winkel um die CC-Bindungsachse zueinander gedreht. Hierdurch wird eine möglichst realistische Darstellung von Cyclopentan erreicht, bei der vier C-Atome in einer Ebene angeordnet werden und ein C-Atom etwas aus der Ringebene herausragt. In Abbildung 5-7 ist die Struktur der "envelope"-Form von Cyclopentan gezeigt und die experimentell bestimmten Bindungswinkel und Bindungslängen von Cyclopentan angegeben.[66]



envelope-Form

Abbildung 5-7: Struktur von Cyclopentan

Das Programm erzeugt aus der virtuellen Molekülstruktur die Zeichenfolge %1C[150.0 -]C[180.0 -]C[210.0 -]CCH%1 als die lineare SMILES - 3D Notation von Cyclopentan.

Die Verbindung Cyclohexan ist in unterschiedlichen Konformationsisomeren bekannt, z. B. die Sesselform und die Wannenform, wie in Abbildung 5-8 dargestellt.

Wannenform Sesselform

Abbildung 5-8: Konformationsisomere des Cyclohexan

Diese Isomere können in der SMILES - 3D Notation durch unterschiedliche Zeichenfolgen codiert werden. Die Wannenform wird durch die Zeichenfolge %1C[120 -]C[180 -]C[240 -]C[120 -]C[180 -]C%1 und die Sesselform durch die Zeichenfolge %1C[120 -]C[240 -]C[120 -]C[240 -]C[120 -]C%1 codiert.

Das Programm WinSmiles - 3D simuliert beim Aufbau der dreidimensionalen Molekülstruktur die CC-Bindung mit einer Bindungslänge von 152.6 pm und die CH-Bindung mit einer Länge von 110,6 pm. Den sp^3-hybridisierten Kohlenstoffatomen wird eine tetraedrische Zelle und den terminalen Wasserstoffatomen eine monohedrale Zelle zugeordnet. Die Molekülstrukturen der verschiedenen Konformationsisomere werden durch die explizite Angabe des dihedralen Winkels der CC-Bindungen in der linearen Notation erreicht.

Das Programm erzeugt, in Abhängigkeit von der codierten Konformation des Cyclohexans, automatisch die entsprechende lineare SMILES - 3D Zeichenfolge aus der virtuellen Molekülstruktur.

Die Molekülstruktur von Benzol kann in der SMILES - 3D Notation durch unterschiedliche lineare Zeichenfolgen codiert werden. Die Struktur kann unter anderem durch die Zeichenfolge %1C=CHC=CHC=CH%1 dargestellt werden. In dieser Form der linearen Notation wird die Reihenfolge, in welcher die Bindungsstellen der trigonalen Zelle der sp^2-hybridisierten Kohlenstoffatome miteinander verbunden werden, durch die explizite Angabe der Wasserstoffatome festgelegt. Durch die Angabe der Doppelbindungen wird die alternierende Bindungsreihenfolge codiert und der Aufbau der dreidimensionalen Molekülstruktur erreicht.

Des weiteren kann durch die Zeichenfolge %1C[180 =]C[180 -]C[180 =]C[180 -]C[180 =]C%1 ebenfalls die planare Ringstruktur von Benzol codiert werden. In dieser Form der linearen Notation wird die Position der Doppel- und Einfachbindungen in der cyclischen Struktur festgelegt und durch die explizite Angabe des Dihedralwinkels der CC-Bindungen die dreidimensionale Struktur aufgebaut.

Die aromatische Struktur von Benzol kann ferner durch die Zeichenfolge %1:C[180 :]C[180 :]C[180 :]C[180 :]C[180 :]C%1 codiert werden. Bei dieser Form der linearen Notation wird die Bindung zwischen den Kohlenstoffatomen direkt als aromatische Bindung durch das Zeichen **:** gekennzeichnet. Zum Aufbau der dreidimensionalen Struktur wird den sp^2-hybridisierten Kohlenstoffatomen jeweils eine trigonal-planare Zelle zugewiesen und die Bindungsstellen der Zellen unter Berücksichtigung des dihedralen Winkels miteinander verbunden, so dass die planare Ringstruktur des Benzols erzeugt wird.

Bei der automatischen Generierung der linearen Notation wird, in Abhängigkeit vom Aufbau der virtuellen Molekülstruktur und der verwendeten Bindungstypen, die entsprechende Form der linearen Notation erzeugt.

5.4 Optionale Spezifikation der Atome und der Bindungen

In diesem Kapitel werden die optionalen Spezifikationsmöglichkeiten der virtuellen Atome und der virtuellen Bindungen erläutert und die Umsetzung in der Software WinSmiles - 3D überprüft.

Die SMILES - 3D Notation bietet die Funktionalität, die Masse eines virtuellen Atoms zu modifizieren. Die optionale Masse eines Isotops wird direkt durch die Eingabe des numerischen Wertes vor dem Atomsymbol codiert. Das Programm verwendet standardmäßig das Atomgewicht der codierten Atome zur Berechnung des Molekülgewichtes. Hierbei wird die optional modifizierte Masse des codierten Atoms entsprechend berücksichtigt. Das Programm zeigt das Molekülgewicht der virtuellen Molekülstruktur in der rechten Ecke der unteren Statuszeile auf der grafischen Benutzeroberfläche an.

Bei der automatischen Erzeugung der SMILES - 3D Notation berücksichtigt das Programm die modifizierte Atommasse und gibt den numerischen Wert der Masse des virtuellen Atoms mit vier Stellen nach dem Komma vor dem codierten Atomsymbol aus.

Die explizite Angabe der Anzahl der Bindungsstellen eines Atoms, ermöglicht dem Anwender Molekülstrukturen zu erzeugen, die von den vordefinierten Standardwerten der Atome abweichen. Die Modifikation der Anzahl der Bindungen eines Atoms wird im Folgenden an zwei Schwefelverbindungen demonstriert.

Dem virtuellen Schwefelatom wird standardmäßig eine tetraedrische Zelle zugeordnet. Das Programm verteilt die sechs Valenzelektronen des Schwefelatoms, entsprechend dem SMILES - 3D Konzept, auf die vier Bindungsstellen der Zelle, so dass die dritte bzw. vierte Bindungsstelle jeweils mit einem Elektronenpaar voll und die erste bzw. die zweite Bindungsstelle der Zelle jeweils mit einem Valenzelektron besetzt sind. Das Schwefelatom verfügt somit über zwei Bindungsstellen die mit einem tetraedrischen Winkel zueinander orientiert sind. Diese Bindungsanordnung wird zum Beispiel beim Schwefelwasserstoff verwendet.

Die Schwefelwasserstoffverbindung wird in der SMILES - 3D Notation einfach durch das Atomsymbol S codiert. Das Programm erzeugt aus der Eingabe die dreidimensionale Struktur und simuliert die SH-Bindung mit einer Länge von 135,9 pm.

In der Software WinSmiles - 3D kann die Verbindung Schwefelhexafluorid durch die optionale Spezifikation der Anzahl der Bindungsstellen durch die Zeichenfolge [S 6](F)*6 codiert werden. Beim Aufbau der dreidimensionalen Struktur wird dem spezifizierten Schwefelatom eine oktraedrische Zelle zugewiesen. In diesem Fall verteilt das Programm die sechs Valenzelektronen auf die Bindungsstellen der Zelle, so dass jede Bindungsstelle mit einem Valenzelektron besetzt ist und das Schwefelatom somit über sechs freie Bindungsstellen verfügt. Die oktraedrisch angeordneten Bindungsstellen des Schwefelatoms werden mit den tetraedrischen Zellen der Fluoratome verbunden und die SF-Bindungen mit einer Länge von 166,6 pm simuliert. In Abbildung 5-9 ist die Visualisierung der virtuellen

Molekülstruktur von Schwefelhexafluorid durch das Programm WinSmiles - 3D dargestellt.

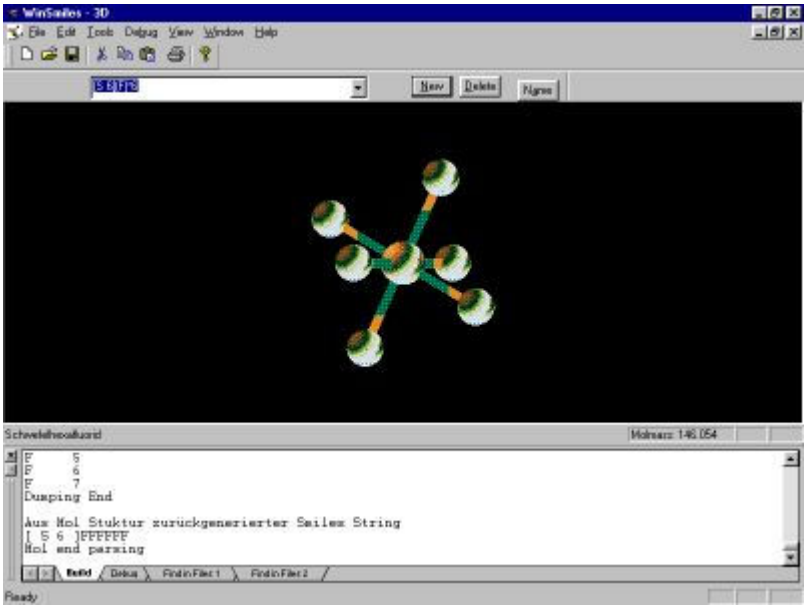


Abbildung 5-9: Visualisierung von SF_6 in dem Programm WinSmiles - 3D

Bei der automatischen Generierung der linearen Notation aus der virtuellen Molekülstruktur des Schwefelhexafluorid wird die Zeichenfolge [S 6]FFFFFF erzeugt. Der optionale Multiplikator der virtuellen Fluoratome wird nicht automatisch generiert.

Das SMILES - 3D Konzept erlaubt dem Anwender die optionale Angabe von Atomladungen und ermöglicht somit die Codierung von geladenen molekularen Teilstrukturen. Die Spezifikation der Atomladung wird im folgenden an dem Hydroxylion und dem H_3O^+-Ion erläutert.

Das einfach negativ geladene Hydroxylion wird in der linearen Notation durch die Zeichenfolge [O -] dargestellt. Das Programm weist dem virtuellen Sauerstoffatom eine tetraedrische Zelle zu und verteilt die sieben Valenzelektronen, sechs Elektronen des Sauerstoffatoms und ein ladungsbedingtes Elektron auf die vier Bindungsstellen der Zelle. Aus der Verteilung der Elektronen resultiert, dass die erste Bindungsstelle mit einem Elektron und die übrigen Bindungsstellen der Zelle jeweils mit einem Elektronenpaar besetzt sind. Die tetraedrische Zelle des Sauerstoffatoms verfügt also über eine freie Bindungsstelle die mit der monohedralen Zelle eines automatisch erzeugten Wasserstoffatoms verbunden wird. Bei der

Berechnung der Atomkoordinaten wird die OH-Bindung mit einer Länge von 101.9 pm simuliert.

Bei der automatischen Generierung der linearen Notation aus der virtuellen Struktur des Hydroxylions erzeugt das Programm die eingegebene Zeichenfolge.

Das einfach positiv geladene H_3O^+-Ion wird durch die lineare Zeichenfolge [O +] codiert und entsprechend in die Software eingegeben. Das Programm weist dem positiv geladenen Sauerstoffatom, in Analogie zu dem negativ geladenen Sauerstoffatom, eine tetraedrische Zelle zu und verteilt die fünf Valenzelektronen auf die vier Bindungsstellen der Zelle. Aus der Verteilung ergibt sich, dass die erste bis dritte Bindungsstelle jeweils mit einem Elektron und die vierte Bindungsstelle der Zelle mit einem Elektronenpaar besetzt ist. Die tetraedrische Zelle hat somit drei freie Bindungsstellen, die jeweils mit einer monohedralen Zelle der automatisch generierten Wasserstoffatome verbunden werden. Bei der Berechnung der dreidimensionalen Struktur des H_3O^+-IonS wird eine OH-Bindungslänge von 101.9 pm angenommen.

Das Programm WinSmiles - 3D erzeugt aus der virtuellen Molekülstruktur automatisch die ursprünglich eingegebene Notation.

In dem Programm WinSmiles - 3D kann der Benutzer bei der Verwendung eines virtuellen Atoms mit einer pentagonalen Zelle die Position der ersten Bindungsstelle festlegen. Zur Verwendung der axialen Position als erste Bindungsstelle wird das virtuelle Atom durch den optionalen Buchstaben „a“ gekennzeichnet. Die Software verbindet eine derartig gekennzeichnete pentagonale Zelle immer mit der axialen Position als erste Bindungsstelle.

Das Programm berücksichtigt die optionale Spezifikation der axialen Bindungsposition bei der automatischen Erzeugung der linearen Notation und erzeugt in der Zeichenfolge den Buchstaben a.

5.5 Definition und Integration von Templates

In die weiterentwickelte Software WinSmiles - 3D ist die Funktionalität zur Integration von vordefinierten Templates in die Notation der Molekülstrukturen implementiert.

Zur Verwendung dieses Features ist es erforderlich, das zu verwendende Template zunächst in dem OGL-Document der Anwendung zu definieren oder das Document vom Typ „SMILES Molecules File“ (*.smi), in welchem das entsprechende Template bereits gespeichert ist, zu öffnen. Anschließend kann das Template, unter Berücksichtigung der freien Bindungsstellen, in die Notation weiterer Molekülstrukturen integriert werden. Bei der Definition von Templates ist es sinnvoll, die freien Bindungspositionen der Molekülstruktur in der Notation des Templates explizit anzugeben und mit dem Buchstaben L zu kennzeichnen, da dies die Transparenz der linearen Notation für den Anwender erhöht.

In dem Programm WinSmiles - 3D sind die dreidimensionalen Strukturen der zwanzig häufigsten natürlichen Aminosäuren und die Molekülstrukturen verschiedener organischer Ringsysteme als Templates vordefiniert. Die Templates der Aminosäuren sind in der Datei Amino.smi, sowie die Templates der Ringsysteme in der Datei Ring.smi definiert. Nach dem Öffnen der Dateien können die vordefinierten Strukturen entweder durch die Definition von weiteren Templates ergänzt, oder in neue Molekülstrukturen integriert werden.

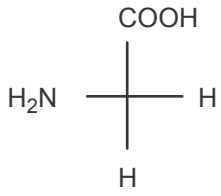
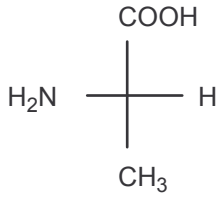
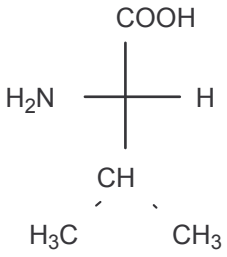
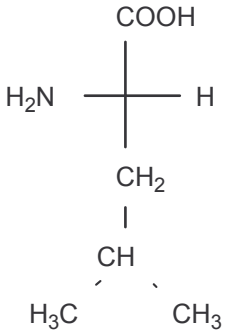
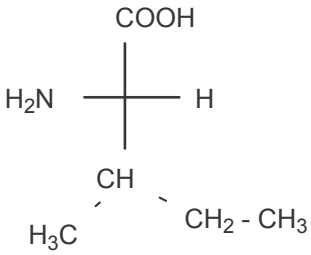
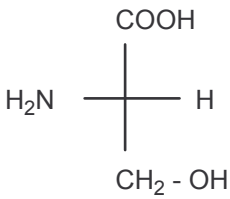
Im folgenden wird die Notation der codierten Templates erläutert und der Aufbau der vordefinierten Molekülstrukturen beschrieben.

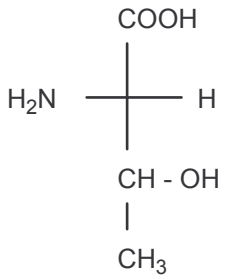
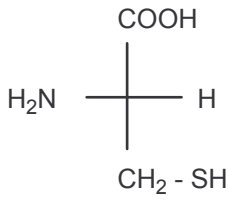
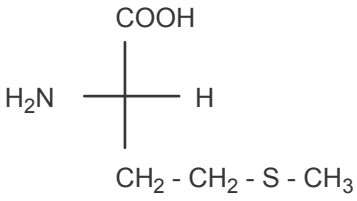
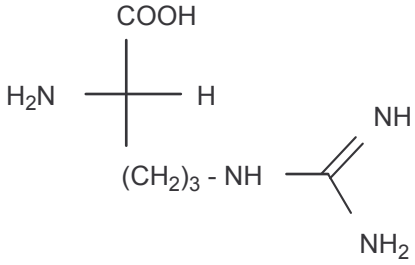
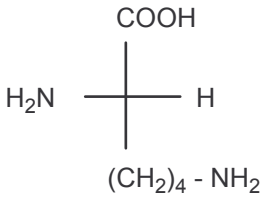
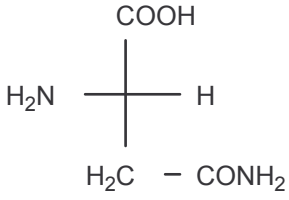
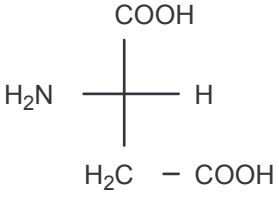
Die Templates der Aminosäuren verfügen alle über zwei freie Bindungspositionen, die bei der Integration der vordefinierten Molekülstrukturen selektiv besetzt werden. Die erste freie Bindungsposition ist die Hydroxylgruppe der Carboxylgruppe, der allgemein als Sauerstoff terminaler Teil der Aminosäure bezeichnet wird, und die zweite freie Bindungsposition ist die zweite Bindungsstelle der Aminogruppe, die allgemein als Stickstoff terminaler Teil bezeichnet wird. Diese Anordnung der freien Bindungspositionen in den Templates der Aminosäuren

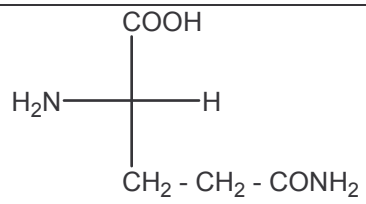
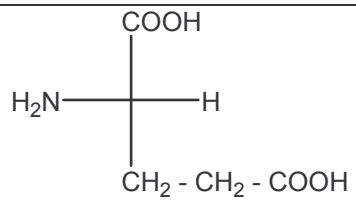
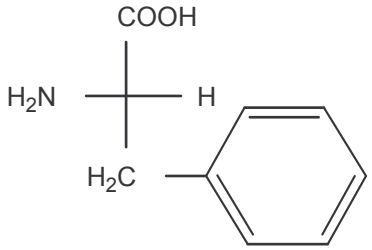
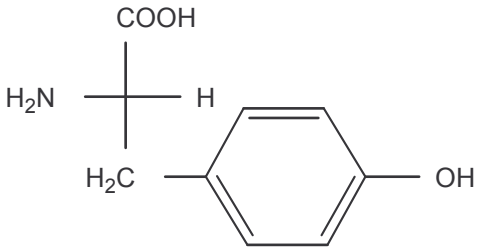
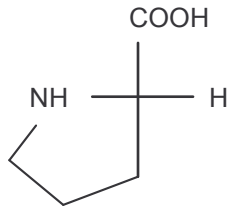
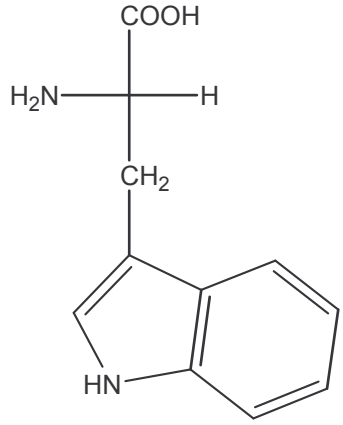
ermöglicht zum Beispiel den effizienten Aufbau der dreidimensionalen Struktur von Peptidsequenzen durch die Aneinanderreihung der entsprechenden Aminotemplates in der linearen Notation.

Die natürliche Aminosäure (L)-Alanin ist durch die lineare Zeichenfolge L1C=OC(NL2)C als Template codiert und unter dem Namen Ala gespeichert. Die Molekülstruktur wird aus den tetraedrischen Zellen der sp^3-hybridisierten Kohlenstoffatome und des Stickstoffatoms, sowie der trigonalen Zelle des sp^2-hybridisierten Kohlenstoffatoms der ursprünglichen „Carboxylgruppe“ aufgebaut. Die Bezeichnung „Carboxylgruppe“ wird hier lediglich zum besseren Verständnis verwendet, da in der linearen Notation die Hydroxylgruppe der Carboxylgruppe durch die freie Bindungsposition substituiert ist. Die terminalen Wasserstoffatome und die Liganden zur Kennzeichnung der freien Bindungspositionen des Templates werden in der Struktur durch monohedrale Zellen dargestellt. Die Bindungsstellen der jeweiligen Zellen werden entsprechend dem SMILES - 3D Konzept unter Berücksichtigung des Dihedralwinkels miteinander verbunden und die freien Bindungsstellen der Struktur automatisch mit Wasserstoffatomen besetzt. Die L-Konfiguration des chiralen Kohlenstoffatoms wird durch die Besetzung der ersten Bindungsstelle der tetraedrischen Zelle mit dem Kohlenstoffatom der ursprünglichen „Carboxylgruppe“, der zweiten Bindungsstelle mit dem Stickstoffatom der Aminogruppe und der dritten Bindungsstelle der Zelle mit dem Kohlenstoffatom der Methylgruppe erreicht. Die vierte Bindungsstelle, die nicht explizit in der linearen Zeichenfolge codiert ist, wird automatisch mit einem Wasserstoffatom besetzt. Die erste freie Bindungsstelle des definierten Templates an dem Kohlenstoffatom der ursprünglichen „Carboxylgruppe“ ist in der linearen Zeichenfolge durch die Buchstaben L1 und die zweite freie Bindungsstelle an dem Stickstoffatom durch die Zeichen L2 gekennzeichnet.

Die Molekülstrukturen der übrigen vordefinierten Aminosäuren sind aus der gleichen Grundstruktur aufgebaut und unterscheiden sich lediglich im Aufbau der funktionellen Seitenkette. In der nachfolgenden Tabelle 5-1 sind die linearen Notationen und die Namen der Templates der vordefinierten Aminosäuren dargestellt.

Amino-Säure	Zweidimensionale Strukturformel	Template-Name	SMILES - 3D Notation
Glycin		Gly	L1C=OCNL2
Alanin		Ala	L1C=OC(NL2)C
Valin		Val	L1C=OC(NL2)C(C)C
Leucin		Leu	L1C=OC(NL2)CC(C)C
Isoleucin		Ile	L1C=OC(NL2)C(C)CC
Serin		Ser	L1C=OC(NL2)CO

Threonin		Thr	L1C=OC(NL2)C(O)C
Cystein		Cys	L1C=OC(NL2)CS
Methionin		Met	L1C=OC(NL2)CCCNC(N)=N
Arginin		Arg	L1C=OC(NL2)CCCN(N)=N
Lysin		Lys	L1C=OC(NL2)CCCCN
Asparagin		Asn	L1C=OC(NL2)CC=ON
Asparagin-säure		Asp	L1C=OC(NL2)CCC=OO

Glutamin		Gln	L1C=OC(NL2)CCC=ON
Glutamin-säure		Glu	L1C=OC(NL2)CCC=OO
Phenylalanin		Phe	L1C=OC(NL2) CC%1[180 :] C[180 :]C[180 :]C[180 :]C[180 :] C%1
Tyrosin		Tyr	L1C=OC(NL2)CC%1[180 :] C[180 :]C[180 :]C[180 :](C [180 :]C%1)O
Prolin		Pro	L1C=OC[180 -](NL2[180 -] C[180 -]C[180 -]C%1)%1
Tryptophan		Trp	L1C=OC(NL2)CCC=(C[180 -] N[180 -]C=%2[180 -]C[180 =] C[180 -]C[180 =]C[180 -] C%2%1)%1

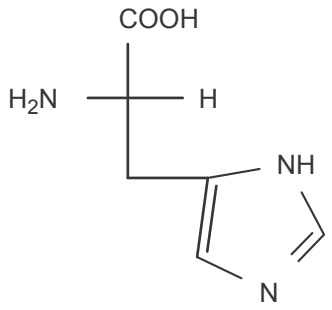
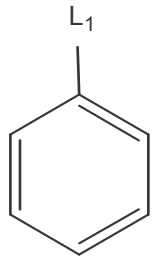
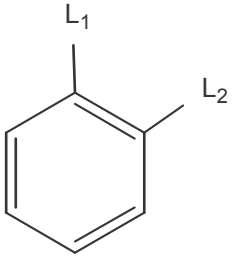
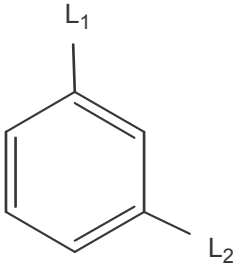
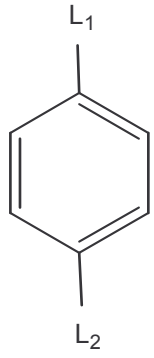
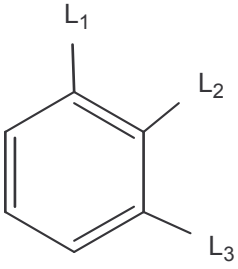
Histidin		His	L1C=OC(NL2)CC=(C[180 -]N[180 =]C[180 -]N%1)%1

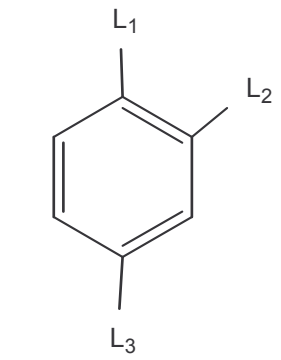
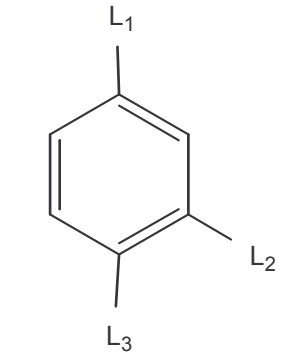
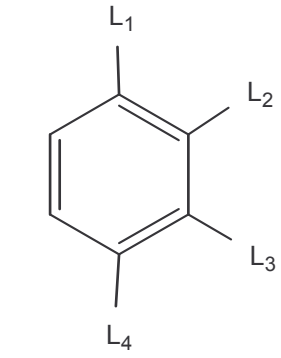
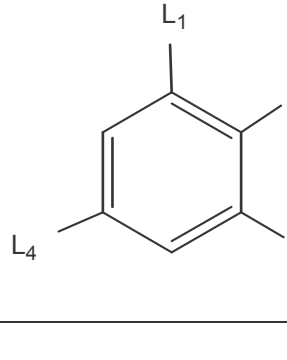
Tabelle 5-1: Die Templates der Aminosäuren

In der Software WinSmiles - 3D sind die Molekülstrukturen einiger häufig vorkommender organischer Ringsysteme in Templates vordefiniert. Die flexible Integration der Templates in die Notation ermöglicht dem Anwender neue und komplexe Strukturen schnell und effizient zu codieren. Im folgenden wird der Aufbau der vordefinierten Ringstrukturen und die lineare Notation der Templates erläutert.

Die Phenylgruppe kommt in zahlreichen chemischen Verbindungen vor. Die planare Ringstruktur und die tetraedrische Anordnung der Bindungen des sp^3-hybridisierten Kohlenstoffatoms wird in der SMILES - 3D Notation durch die Zeichenfolge L1CC%1[180 :]C[180 :]C[180 :]C[180 :]C[180 :]C%1 codiert und die Molekülstruktur unter dem Templatenamen „_Ph“ gespeichert. Die freie Bindungsposition an dem sp^3-hybridisierten Kohlenstoffatom ist in dem Template in einer Ebene mit dem aromatischen Ring angeordnet.

Die aromatische Ringstruktur des Benzols kommt in einer Vielzahl chemischer Verbindungen vor. In den unterschiedlichen Verbindungen ist die planare Ringstruktur häufig durch ein, zwei oder mehrere Substituenten an den Bindungspositionen der Wasserstoffatome funktionalisiert. Zur effizienten Integration derartig substituierter aromatischer Ringsysteme in die lineare Notation, ist es sinnvoll die unterschiedlichen cyclischen Strukturen, die über die entsprechenden, freien Bindungspositionen verfügen, in Templates vorzudefinieren. In der nachfolgenden Tabelle 5-2 sind die Templates der Benzolderivate, die in dem Programm vordefiniert sind, dargestellt und die lineare Notation angegeben.

Zweidimensionale Strukturformel	Template-Name	SMILES - 3D Notation
	_Bz	LC%1[180 :]C[180 :]C[180 :]C[180 :]C[180 :]C%1
	_2Bz	L1C%1[180 :]C[180 :](C[180 :]C[180 :]C[180 :]C%1)L2
	_3Bz	L1C%1[180 :]C[180 :]C[180 :](C[180 :]C[180 :]C%1)L2
	_4Bz	L1C%1[180 :]C[180 :]C[180 :]C[180 :](C[180 :]C%1)L2
	_23Bz	L1C%1[180 :]C[180 :](C[180 :](C[180 :]C[180 :]C%1)L3)L2

	_24Bz	L1C%1[180 :]C[180 :](C[180 :]C[180 :](C[180 :]C%1)L3)L2
	_35Bz	L1C%1[180 :]C[180 :]C[180 :](C[180 :]C[180 :](C%1)L3)L2
	_234Bz	L1C%1[180 :]C[180 :](C[180 :](C[180 :](C[180 :]C%1)L4)L3)L2
	_235Bz	L1C%1[180 :]C[180 :](C[180 :](C[180 :]C[180 :](C%1)L4)L3)L2

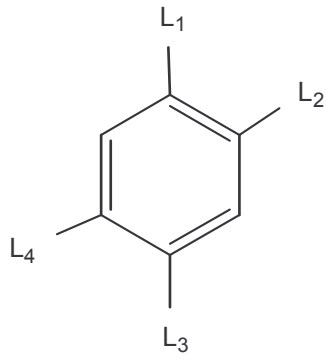
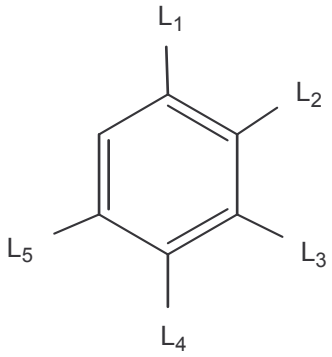
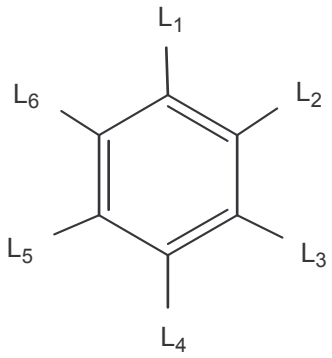
	_245Bz	L1C%1[180 :]C[180 :](C[180 :]C[180 :](C[180 :](C%1)L4)L3)L2
	_2345Bz	L1C%1[180 :]C[180 :](C[180 :](C[180 :](C[180 :](C%1)L5)L4)L3)L2
	_23456Bz	L1C%1[180 :]C[180 :](C[180 :](C[180 :](C[180 :](C%1L6)L5)L4)L3)L2

Tabelle 5-2: Vordefinierte Molekülstrukturen der Benzolderivate

Die Namen der Benzoltemplate beginnen alle mit dem Zeichen _ , einer oder mehrerer Zahlen und den Zeichen Bz als Abkürzung für Benzol. Der Unterstrich kennzeichnet immer die freie Bindungsstelle am ersten Kohlenstoffatom des Ringes und die nachfolgenden Zahlen von 2 bis 5 kennzeichnen die Kohlenstoffatome, die eine weitere freie Bindungsposition haben. Bei der Verwendung der Templates mit mehreren freien Bindungspositionen werden die freien Bindungsstellen in der numerischen Reihenfolge besetzt.

6. Diskussion und Ausblick

Im Rahmen dieser Arbeit ist das SMILES - 3D Konzept, ausgehend von der linearen Wiswesser Notation[30, 31] und auf Basis des von Weininger et al. entwickelten SMILES Konzeptes[6], vorgestellt und die Umsetzung der linearen Notation in der anwenderorientierten Software WinSmiles - 3D erläutert worden.

Das ursprünglich von Hinze et al. publizierte Broad SMILES Konzept[7] ist in dieser Arbeit zu dem SMILES - 3D Konzept zur Codierung von dreidimensionalen Molekülstrukturen in einer linearen Notation weiterentwickelt und die Zeichenfolge der Notation simplifiziert worden.

In dem SMILES - 3D Konzept ist die Festlegung des Standardwertes des dihedralen Winkels zwischen verbunden Zellen durch die Einführung des Prinzips, dass die niedrigst mögliche Bindungsstelle des Add-Atoms in einer Ebene trans orientiert ist zu der niedrigst möglichen Bindungsposition des From-Atoms, entscheidend vereinfacht und die Definition der Standardwerte des Dihedralwinkels neu überarbeitet worden.

Die lineare SMILES - 3D Notation ist durch den Verzicht auf die optionale Spezifikationsmöglichkeit des Positionsindex eines codierten Atoms in vielen Codierungsfällen verkürzt und die lineare Zeichenfolge, durch die eventuell erforderliche explizite Angabe eines Atoms an einer Bindungsstelle, in ihrem Erscheinungsbild vereinheitlicht worden. Weiterhin resultiert aus dieser Vereinfachung die Harmonisierung der Konvention der Standardwerte des Dihedralwinkels, da durch den optionalen Positionsindex eines Atoms weitere Regeln erforderlich sind.

Des weiteren ist in dem weiterentwickelten SMILES - 3D Konzept auf die Verwendung des Zeichens ^, zur Kennzeichnung der ekliptischen Stellung der Bindungspositionen zweier miteinander verbundener tetraedrischer Zellen verzichtet worden, so dass diese Information nunmehr einheitlich in der Zeichenfolge durch die explizite Angabe des Dihedralwinkels der Bindung codiert wird.

Im Rahmen dieser Arbeit ist das weiterentwickelte SMILES - 3D Konzept zur Codierung von zahlreichen linearen, verzweigten und cyclischen Molekülen erfolgreich verwendet worden. Die Anwendung zeigt, dass dreidimensionale Molekülstrukturen, in denen die räumliche Bindungsanordnung der Atome nicht wesentlich von der Orientierung der Bindungsstellen der Zellen abweicht, einfach und effektiv in der linearen Zeichenfolge codiert werden können. Zur Codierung von Molekülstrukturen in denen die räumliche Anordnung der Bindungen entscheidend von der Orientierung der Bindungsstellen der verwendeten Zelltypen abweicht ist es erforderlich die Bindungswinkel explizit zu modifizieren. Hierzu wäre eine Erweiterung der optionalen Spezifikationsmöglichkeiten des Atoms nötig durch die der Bindungswinkel direkt codiert wird. Diese Erweiterung der SMILES - 3D Notation würde allerdings die lineare Zeichenfolge verlängern und die Lesbarkeit beeinträchtigen. Zur Codierung von Atomen mit mehr als sechs Elektronengruppen ist es unumgänglich weitere Zelltypen, z. B. eine Pentagonalbipyramidale Zelle zur Darstellung von 7 Elektronengruppen oder eine quadratisch-antiprismatische Zelle zur Darstellung von 8 Elektronengruppen, in die SMILES - 3D Notation zu implementieren.

Die im Rahmen dieser Arbeit weiterentwickelte Software WinSmiles - 3D interpretiert die lineare Zeichenfolge entsprechend dem modifizierten SMILES - 3D Konzept und generiert als interne Repräsentation die sogenannte virtuelle Molekülstruktur. Auf Basis der internen virtuellen Molekülstruktur berechnet das Programm, unter Berücksichtigung der implementierten Zelltypen zur räumlichen Orientierung der Bindungsstellen und der implementierten kovalenten Atomradien, die dreidimensionale Struktur des codierten Moleküls.

Die berechnete Molekülstruktur kann von dem Programm in unterschiedlichen Darstellungen, z. B. einer dreidimensional wirkenden Abbildung, auf der grafischen Benutzeroberfläche veranschaulicht werden. Die Visualisierung auf dem Bildschirm ist durch die Integration des OpenGL-Interface in die Software realisiert worden.

Komplexe und große Molekülstrukturen können durch die Integration von vordefinierten Molekülstrukturen, sogenannten Templates, effizient in einer linearen Notation codiert werden. Die Funktionalität des Programms ist daher erweitert und entsprechende Algorithmen zur Integration von Templates in die Software eingebettet worden.

Des weiteren ist in die Software ein Interface implementiert worden, dass unabhängig von der Eingabe, ausgehend von der internen Repräsentation die lineare SMILES - 3D Notation erzeugt und Funktionen zur Generierung der virtuellen Molekülstruktur bereit stellt.

Dieses Interface bildet die Grundlage für die weitere Entwicklung der Software in der die interne virtuelle Molekülstruktur durch die Eingabe über die grafische Benutzeroberfläche durch die Maus realisiert wird und die gegenwärtig noch nicht vollständig in das Programm implementiert ist.

Die erweiterte Funktionalität der Software kann zukünftig zur kompakten Codierung von dreidimensionalen Molekülstrukturen in einer linearen Notation, unabhängig von der Erzeugung der Zeichenfolge, verwendet werden. Das Programm bietet die Möglichkeit Molekülfragmente, z. B. die unterschiedlichen Aminosäuren, in Templates zu speichern und aus diesen Molekülfragmenten die dreidimensionale Struktur größerer Moleküle, z. B. die Struktur von Peptiden, aufzubauen. Des weiteren kann die dreidimensionale Struktur, auch von größeren Molekülen auf eine lineare Zeichenfolge reduziert werden und hierdurch die Molekülstruktur zwischen Computern, z. B. in internen Netzwerken oder im Internet, äußerst effizient übermittelt werden.

Ferner kann die lineare SMILES - 3D Notation zur Suche von dreidimensionalen Molekülstrukturen oder von Substrukturen in entsprechenden Datenbanken verwendet werden.

Weiterhin eignet sich die von dem Programm WinSmiles - 3D berechnete dreidimensionale Molekülstruktur als geometrische Ausgangsstruktur in quantenchemischen oder molekülmechanischen Programmen zur Berechnung exakter Molekülstrukturen.

Zur Generierung von eineindeutigen bijektiven linearen SMILES - 3D Notationen ist es notwendig weitere Regeln, die den Anfang der linearen Notation festlegen, in das Konzept zu implementieren. Dies ist gegenwärtig noch nicht realisiert und könnte zukünftig, in Anlehnung an die in Kapitel 2.3.1 beschriebene Wiswesser Notation, durch die Integration von hierarchischen Regeln gelöst werden.

7. Literaturverzeichnis

[1] Sadowski, J.; Gasteiger, J.,Chem. Rev. **1993**, 93, 2567-2581.

[2] Hehre, W. J.; Radom, L.; Schleyer, P. v. R.; Pople J. *Ab Initio Molecular Orbital Theory*; John Wiley and Sons, New York, 1986.

[3] Stewart, J. J. P. in *Reviews in Computational Chemistry*; Lipkowitz, K. B.; Boyd, D. B., Eds.; VCH Publishers; New York, 1990, Vol. 1, 45-82.

[4] Dykstra, E. C..; Augspurger, J, D.; Kritman, B.; Malik, D. in *Reviews in Computational Chemistry*; Lipkowitz, K. B.; Boyd, D. B., Eds.; VCH Publishers; New York, 1990, Vol. 1, 83-118.

[5] Burkert, U.; Allinger, N. L. *Molecular Mechanics*,; ACS Monograph 177; American Chemical Society; Washington, DC, 1989.

[6] Weininger, D., Weininger, A., Weininger, J. L.; Chem. Des. Autom. News **1986**, 1, 2-15.

[7] Hinze, J.; Welz, U. in Gasteiger, J; Eds. *Software-Entwicklung in der Chemie;* Gesellschaft Deutscher Chemiker, Frankfurt am Main, 1996.

[8] Goodson, A. L.; J. Chem. Inf. Comput. Sci. **1980**, 20, 167.

[9] International Union of Pure and Applied Chemistry; *Nomenclature of Inorganic Chemistry*, 2. ed., Pergamon Press; Oxford, 1981.

[10] International Union of Pure and Applied Chemistry, *Nomenclature of organic Chemistry*, Rigaudy, J.; Pergamon Press; Oxford, 1979.

[11] International Union of Pure and Applied Chemistry (IUPAC), http://www.iupac.org.

[12] Chemical Abstracts; *Naming and indexing of chemical substances for Chemical abstracts*, reprint of Appendix IV from the Chemical abstracts 1987 index guide; Chemical Abstracts Services; Columbus, Ohio, 1987.

[13] Chemical Abstracts Services (CAS), http://www.cas.org.

[14] Beilstein-Institut; *Kennen Sie Beilstein? Erläuterungen zu Beilsteins Handbuch der organischen Chemie*, Beilstein-Institut für Literatur der organischen Chemie (Frankfurt, Main), Springer Verlag, Berlin, 1979.

[15] Morgan, H. L.; J. Chem. Doc. **1965**, 5, 107-113.

[16] Dittmar, P. G.; Farmer, N. A.; Fisanick, W.; Haines, R. C.; Mockus, J.; J. Chem. Inf. Comput. Sci. **1983**, 23, 93-102.

[17] Wipke, W. T.; Rogers, D.; J. Chem. Inf. Comput. Sci. **1984**, 24, 255-262.

[18] Attias, R.; J. Chem. Inf. Comput. Sci. **1983**, 23, 102-108.

[19] King, R.B. *Graph theory and Topology in Chemistry*, Elsevier, 1987.

[20] Stobaugh, R. E.; J. Chem. Inf. Comput. Sci. **1985**, 25, 271-275.

[21] Balaban, A. T.; J. Chem. Inf. Comput. Sci. **1985**, 25, 334-343.

[22] Wiswesser, W. J.; J. Chem. Inf. Comput. Sci. **1985**, 25, 258-263.

[23] Wiswesser, W. J.; *A Line-Formula Chemical Notation*, Thomas Crowell, New York, 1954.

[24] Bremser, W.; Anal. Chim. Acta. **1978**, 103, 355-365.

[25] Barnard, J. M.; Jochum, C. J.; Welford, S. M.; ACS Symposium Series 400; American Chemical Society; Washington, D. C., 1989, 76-81.

[26] Huixiao, H.; Xinquan, X.; J. Chem. Inf. Comput. Sci.**1992**, 32, 116-120.

[27] Herndon, W. C.; Bertz, S. H.; J. Comput. Chem. **1987**, 4, 367-374.

[28] Read, R. C.; J. Chem. Inf. Comput. Sci. **1983**, 23, 135-149.

[29] Read, R. C.; J. Chem. Inf. Comput. Sci. **1985**, 25 116-128.

[30] Smith, E. G. *The Wiswesser Line-Formula Chemical Notation*, McGraw-Hill, New York, 1966.

[31] Vollmer, J.; J. Chem. Educ. **1988**, 60, 192-196.

[32] Wiswesser, W. J.; J. Chem. Inf. Comput. Sci. **1982**, 22, 88-93.

[33] Wiswesser, W. J.; J. Inf. Sci. **1982**, 4, 69-77.

[34] Weininger, D.; J. Chem. Inf. Comput. Sci. **1988**, 28, 31-36.

[35] Weininger, D.; Weininger, A.; Weininger, J. L.; J. Chem. Inf. Comput. Sci. **1989**, 29, 97-101.

[36] Wilcox, C. S.; Levinson, R. A. in *Artifical intelligence applications in Chemistry*; Pierre, T. H., Hohne, B. A., Eds.; ACS Symposium Series 306; American Chemical Society; Washington, DC, 1986.

[37] Ghoshal, N.; J. Chem. Inf. Comput. Sci.**1990**, 30, 308-311.

[38] Ash, S.; Cline M. A.; Homer, R. W.; Hurst, T.; Smith, G. B.; J. Chem. Inf. Comput. Sci. **1997**, 37, 71-79.

[39] TRIPOS, Inc., St. Louis, MO, http://www.tripos.com

[40] Exographics, Newark, NJ.

[41] Entwickelt von Dr. R. Pearlman, University of Texas at Austin.

[42] Fujita, S.; Comput. Chem. **1994**, 18, 109-116.

[43] Fujita, S.; Tanka, N.; J. Chem. Inf. Comput. Sci. **1999**, 39, 903-914.

[44] Gakh, A. A.; Burnett, M. N.; J. Chem. Inf. Comput. Sci. **2001**, 41, 1494-1499.

[45] Ash, J. E.; Hyde, E.; *Chemical Information Systems*, Ellis Horwood, Chichester, 1975.

[46] Granito, C. E.; Rosenbeerg, M. D.; J. Chem. Doc. **1971**, 11, 251-256.

[47] Garfield, E.; Revesz, G. S.; Granito, C. E.; Dorr, H. A.; Calderon, M. M.; Warner, A.; J. Chem. Doc. **1970**, 10, 54-58.

[48] Sorter, P. T.; Granito, C. E.; Gilver, J. C.; Gelberg, A; Metcalf, E. A.; J. Chem. Doc. **1964**, 4(1), 56-60.

[49] Fritts, L. E.; Schwind, M. M.; J. Chem. Inf. Comput. Sci. **1992**, 22, 106-109.

[50] Weininger, D.; J. Chem. Inf. Comput. Sci. **1990**, 30, 237-243.

[51] Walters, W. P.; Yalkowsky, S. H.; J. Chem. Inf. Comput. Sci. **1996**, 36, 1015-1017.

[52] Rowley, R. J.; Oscarson, J. L.; Rowley, R.L.; Wilding, W. V.; J. Chem. Eng. Data **2001**, 46 (5), 1110-1113.

[53] Siani, M. A.; Weininger, D.; Blaney, J. M.; J. Chem. Inf. Comput. Sci. **1994**, 34, 588-593.

[54] Siani, M. A., Weininger, D.; James, C. A.; Blaney, j. M.; J. Chem. Inf. Comput. Sci. **1995**, 35, 1026-1033.

[55] Daylight Chemical Information Systems, 419 East Palace, Santa Fe, New Mexico 87501, http://www.daylight.com

[56] Bone, R. G. A.; Firth, M. A.; Sykes, R. A.; J. Chem. Inf. Comput. Sci. **1999**, 39, 846-860.

[57] Gillespie, R. J., Nyhohm, R. S. Quart. Rev. II, **1957**, 339-380.

[58] Gillespie, R. J., J. Chem. Educ. **1963**, 40, 295-301.

[59] Microsoft Corporation, http://www.microsoft.com .

[60] Silicon Graphics, Inc.

[61] Microsoft Developer Network, http://www.msdn.microsoft.com

[62] Stöcker, H. *Taschenbuch mathematischer Formeln und moderner Verfahren*, 4. korrigierte Aufl., Verlag H. Deutsch, Frankfurt a. Main, 1999.

[63] Mark J. Kilgard, *The OpenGl Utility Toolkit (GLUT) Programming Interface API Version 3*, **1996**, http://www.opengl.org

[64] Pauling, L.; Corey, R. B., Nature **1953**, 171, 345.

[65] Koltun, W. L.; Biopolymers **1965**, 3, 665-679.

[66] Vollhardt, K. P. C. *Organische Chemie*, 1. Aufl., VCH Verlag, Weinheim, 1988.

8. Anhang

8.1 Methoden und Membervariablen der Klasse Atom

Der nachfolgende Abschnitt enthält eine alphabetisch geordnete, detaillierte Beschreibung jeder einzelnen Methode und Membervariablen, die in die Klasse Atom implementiert ist.

Atom ()

Der Konstruktor initialisiert die Membervariablen mit den Eigenschaften des Atoms, realisiert die Referenzverknüpfung zwischen den Atomen und initialisiert die Bindungen des Atoms.

Atom (AtomTable *at, Atom *atom)

Parameter:

at

Referenz auf das zu verwendende Objekt vom Typ AtomTable, das die Standardwerte der Atome enthält.

atom

Referenz auf ein Objekt vom Typ Atom, die auf das zuvor erzeugte Objekt vom Typ Atom verweist.

Bemerkungen:

Zunächst wird der Membervariablen m_at der Parameter at und den Membervariablen M_sym, m_oz, m_ele, m_bind, m_oxistate und m_mass die entsprechenden Werte aus dem Objekt at zugewiesen. Den Membervariablen m_hybridstring, m_varname wird der Wert NULL, den Membervariablen m_noligand, m_charge der Wert 0 und der Membervariablen m_isautoatom der Wert False zugewiesen. Des weiteren wird den Variablen m_typeset und m_case der Membervariablen m_cell vom Typ Cell der Wert 0, sowie der Variablen m_typ des Objektes der korrespondierende Wert aus dem Parameter at zugewiesen.

Zur Erzeugung der Referenzverknüpfung der Objekte vom Typ Atom wird das Objekt des Parameters atom mit dem in dieser Funktion erzeugten Objekt über die Referenzen m_next des Parameters und der Membervariablen n_prev des Objektes miteinander verknüpft. Hat die Funktion keinen gültigen Parameter atom so wird den Membervariablen m_next und m_prev der Wert NULL zugewiesen.
Abschließend werden die Bindungen des neu erzeugten Objektes initialisiert, dazu wird den Variablen m_toatom, m_fromatom und m_backbond des jeweiligen Elementes der Membervariablen m_bond vom Typ Bond der Wert Null zugewiesen und den Variablen M_frombond und m_tobond der Elemente der Standardswert –1.

~Atom()

Destruktor zur Zerstörung der aufrufenden Instanz.

~Atom()

Parameter:
Der Destruktor benötigt keine Parameter.

Rückgabewert:
Der Destruktor hat keinen Rückgabewert.

Bemerkungen:

Der Destruktor löscht und gibt den Speicherplatz der Membervariablen m_sym, m_varname und m_hybridstring frei.

Count ()

Die Funktion erzeugt die Nummerierung der Atome.

void Count ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion ermittelt zunächst das erste Objekt vom Typ Atom durch den Aufruf der Funktion First. Anschließend wird die fortlaufende Nummerierung der Objekte vom Typ Atom, die aus der Referenzverknüpfung m_next der jeweiligen Objekte vom Typ Atom ermittelt werden, der Membervariablen m_no zugewiesen. Die Nummerierung beginnt mit dem Wert 1.

delete ()

Die Funktion zerstört Objekte vom Typ Atom.

void delete ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion ermittelt zunächst aus der Membervariablen m_next ein eventuelles weiteres Objekt vom Typ Atom und ruft die Funktion Delete der ermittelten Instanz auf. Abschließend wird der Destruktor der eigenen Instanz aufgerufen.

Draw ()

Funktion zur Visualisierung des Atoms.

void Draw ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion ermittelt zunächst den Wert der Membervariablen m_next und ruft bei einem gültigen Wert der Variablen die Funktion Draw der Instanz m_next rekursiv auf.

Anschließend wird die durchschnittliche Bindungslänge des Atoms durch Division der Summe der Bindungslängen aller Bindungen des Atoms, ermittelt aus den Variablen m_rad der Elemente der Membervariablen m_bond, durch die Anzahl der ToAtome berechnet. Im weiteren Funktionsablauf wird die durchschnittliche Bindungslänge mit dem Faktor 2,5 bei der Darstellung der Atome als Kugeln skaliert und bei der Darstellung der Atome durch Atomsymbole bzw. Atomsymbole die durch Linien miteinander verbunden sind, wird die zuvor berechnete Bindungslänge nicht skaliert. In Abhängigkeit von dem Wert der Variablen Drawmode des Objektes g_Drawmode wird der Ausgabestring in der inversen Atomfarbe erzeugt und an der Position des

Atoms, ermittelt aus den Membervariablen m_x, m_y und m_z, durch die Funktion SetText ausgegeben.
Anschließend wird durch den Aufruf der Funktion glPushMatrix die aktuelle Matrix festgelegt und eine Translationsmatrix mit den Werten der Membervariablen m_x, m_y und m_z durch den Aufruf der Funktion glTranslated durch Matrixmultiplikation erzeugt. Nachdem die Farbe zur Darstellung der Atome durch den Aufruf der Funktion glColor4d mit den Werten der Variablen r, g und b der Membervariablen m_at initialisiert ist, wird in Abhängigkeit von dem Ausgabemodus entweder eine Kugel durch den Aufruf der Funktion GlutSolidSphere zur Darstellung eines Atoms erzeugt, oder die Funktion Draw aller Elemente der Membervariablen m_bond vom Typ Bond zur Darstellung einer Bindung aufgerufen. Abschließend wird durch den Aufruf der Funktion glPopMatrix die aktuelle Matrix wieder freigegeben.

First ()

Die Funktion ermittelt das zuerst erzeugte Atom eines Moleküls.

*Atom First()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion gibt eine Referenz auf das Objekt vom Typ Atom zurück, die auf das erste Objekt vom Typ Atom in der Referenzverknüpfung verweist.

Bemerkungen:

Die Funktion ermittelt aus der Membervariablen m_prev, in der die Referenzverknüpfung der Objekte vom Typ Atom realisiert ist, die Referenz auf das erste Objekt vom Typ Atom und gibt die Referenz auf das ermittelte Objekt zurück.

FreeBondsCount()

Die Funktion ermittelt die Anzahl der freien Bindungen eines Atoms.

int FreeBondsCount()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion liefert einen Rückgabewert vom Typ int, der der Anzahl der freien Bindungsstellen des Atoms entspricht.

Bemerkungen:

Zunächst wird die Anzahl der Bindungselektronen durch Subtraktion der Membervariablen m_charge von der Membervariablen m_ele und die Anzahl der Orbitale aus der Variablen m_type der Membervariablen m_cell vom Typ Cell ermittelt. Anschließend werden die Elektronen auf die Orbitale, die Elemente eines lokalen Feldes vom Typ int, mit Hilfe des Modulus Operators verteilt. Aus allen Elementen der Membervariablen m_bond vom Typ Bond, deren Variable m_toatom ein Objekt oder deren Variable m_prozent nicht 0 ist, wird aus der Variablen m_type die Anzahl der Elektronen der To-Atome ermittelt und ebenfalls auf die Elemente des lokalen Feldes vom Typ int verteilt. Abschließend wird die Anzahl der Orbitale, die nur ein Elektron enthalten, aus den Elementen des lokalen Feldes vom Typ int, deren Wert 1 ist, ermittelt und die so bestimmte Anzahl der freien Bindungsstellen zurückgegeben.

GetAtom ()

Die Funktion ermittelt die Referenz auf das im Parameter spezifizierte Atom.

*Atom GetAtom(int no)

Parameter:
no Variable vom Typ int, deren Wert den Index des Atoms angibt.

Rückgabewert:
Die Funktion gibt die Referenz auf das ermittelte Objekt vom Typ Atom zurück. Kann kein Objekt ermittelt werden, gibt die Funktion den Wert NULL zurück.

Bemerkungen:

Die Funktion überprüft zunächst ob die Membervariable m_no mit dem Parameter no übereinstimmt. Ist dies der Fall, so wird die Funktion mit der Rückgabe der Referenz auf das Objekt vom Typ Atom, zu dem die aufrufende Memberfunktion gehört,

beendet. Ist dies nicht der Fall, so wird die Funktion GetAtom mit dem gleichen Parameter no der Membervariablen m_next aufgerufen. Falls die Membervariable m_next nicht existiert wird der Wert NULL zurückgegeben.

IsBondOK ()

Die Funktion überprüft, ob der im Parameter spezifizierte Bindungstyp möglich ist.

bool IsBondOK(int bondtype)

Parameter:
bondtype
Variable vom Typ int, deren Wert der doppelten Anzahl der Bindungselektronen entspricht.

Rückgabewert:
Die Funktion gibt den Wert true zurück, wenn das Atom eine ausreichende Anzahl an freien Bindungsstellen hat. Hat das Atom keine ausreichende Anzahl an freien Bindungsstellen, so liefert die Funktion den Wert false.

Bemerkungen:

Die Funktion ruft die Memberfunktion FreeBondsCount auf und gibt für alle Rückgabewerte der Memberfunktion, die größer sind als der halbe Wert des Parameters bondtype, den Wert true zurück.

Last()

Die Funktion ermittelt das letzte Atom aus der Referenzverknüpfung.

*Atom Last()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion liefert die Referenz auf das letzte Objekt vom Typ Atom aus der Referenzverknüpfung der Objekte vom Typ Atom.

Bemerkungen:

Die Funktion ermittelt aus der Membervariablen m_next, mit der die Referenzverknüpfung der Atome realisiert ist, das Objekt vom Typ Atom dessen Membervariable m_next kein gültiges Objekt enthält und gibt die Referenz auf das ermittelte Objekt zurück.

Read ( )

Die Funktion liest Daten eines Objektes vom Typ Atom aus einer Datei und speichert die Werte der Eigenschaften des Atoms in den entsprechenden Membervariablen.

void Read(FGetS &fgs)

Parameter:
fgs
Objekt vom Typ FGetS, das die zu lesenden Daten enthält.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion liest aus dem im Parameter fgs übergebenen Objekt vom Typ FgetS die Daten und weist die Werte der Eigenschaften den Membervariablen m_sym, m_hybridstring, m_oxistate, m_charge, m_ele, m_m, m_x, m_y und m_z der aufrufenden Instanz zu.
Abschließend ruft die Funktion die Methode Read der Membervariablen m_cell und für jede Bindung des Atoms die Methode Read aller Elemente der Membervariablen m_bond mit dem Parameter fgs auf.

SetAtom(Atom *a)

Die Funktion weist den Eigenschaften des Atoms die entsprechenden Werte zu.

Parameter:
a
Referenz auf ein Objekt vom Typ Atom, die auf das Atom verweist dessen Werte in die aufrufende Instanz übertragen werden.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion weist zunächst den Membervariablen m_oz, m_ele, m_bind, m_oxistate, m_no, m_noligand, m_mass, m_charge, m_x, m_y, m_z, m_autoatom, m_at und m_varcount die entsprechenden Variablen des Parameters a zu. Des weiteren speichert die Funktion das Atomsymbol in der Membervariablen m_syn. Ist der Parameter a ein Template, so wird der Name in der Membervariablen m_varname gespeichert, der sonst der Wert NULL zugewiesen wird. Den Variablen m_type, m_typeset und m_case der Membervariablen m_cell vom Typ Cell werden die entsprechenden Variablen des Parameters a zugewiesen. Abschließend werden alle Bindungen, die Elemente der Membervariablen m_bond, initialisiert, dabei wird den Variablen m_toatom, m_fromatom und m_backbond der Wert NULL zugewiesen.

SetBondFlag ()

Die Funktion steuert das flag einer Bindung zur Kennzeichnung des Bearbeitungszustandes der Bindung.

void SetBondFlag (int flag)

Parameter:
flag
Variable vom Typ int.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion weist der Variablen m_flag aller Elemente der Membervariablen m_bond vom Typ Bond, die jeweils eine Bindung darstellen, den Wert des Parameters flag zu. Abschließend wird für alle Atome, die aus der Referenzverknüpfung m_next ermittelt werden, die Funktion SetBondFlag mit dem Parameter flag aufgerufen.

SetCell ()

Die Funktion ermittelt die Koordinaten der Bindungsstellen in Abhängigkeit vom verwendeten Zelltyp.

void SetCell ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Zunächst weist die Funktion den Elementen der Variablen m_xyzref der Membervariablen m_cell die Werte für den Ursprung und zwei Referenzpunkte zu. Aus der Variablen m_type der Membervariablen m_cell wird der Zelltyp ermittelt und entsprechend des Zelltyps die Werte der Koordinaten der Bindungsstellen der Zelle den Elementen der Variablen m_xyz der Membervariablen m_cell zugewiesen.
Die Zuweisung erfolgt dabei so, dass drei aufeinander folgende Elemente vom Typ double die Werte für die Koordinaten in der Reihenfolge x, y und z enthalten.
Die Werte der Koordinaten der Bindungsstellen sind so gewählt, dass der Mittelpunkt der Zelle im Ursprung und die erste Bindungsstelle auf der x-Achse bei dem Wert 1 ist, sowie dass der Wert des Abstandes aller Bindungsstellen vom Ursprung 1 ist. Bei

dem linearen Zelltypen ist die zweite Bindungsstelle ebenfalls auf der x-Achse bei dem Wert –1. Die Anordnung von drei Bindungsstellen erfolgt in der xy-Ebene, die zweite Bindungsstelle hat einen positiven y-Wert bzw. die dritte Bindungsstelle einen negativen y-Wert. Bei der tetraedrischen Anordnung sind die zweite bis vierte Bindungsstelle rotationsförmig um die x-Achse angeordnet, die zweite Bindungsstelle hat einen positiven y-Wert und einen negativen z-Wert, die dritte Bindungsstelle ist in der xz-Ebene und die vierte Bindungsstelle hat negative Werte.

Bei der oktraedrischen Anordnung sind alle Bindungsstellen auf den Achsen. Der x-Wert der zweiten Bindungsstelle ist –1, der y-Wert der dritten Bindungsstelle ist 1, der z-Wert der vierten Bindungsstelle ist 1, der y-Wert der fünften Bindungsstelle ist –1 und der z-Wert der sechsten Bindungsstelle ist –1.

Abschließend wird der Variablen m_n die Anzahl der Bindungsstellen und der Variablen m_flag, zur Kennzeichnung der Berechnung der Zellkoordinaten, der Wert 1 zugewiesen.

Smiles()

Die Funktion erzeugt aus den Atomem und Bindungen den SMILES String.

*char Smiles (Atom *atom, Bond *bond)

Parameter:

atom

Referenz auf ein Objekt vom Typ Atom die auf das Atom verweist, dessen SMILES String erzeugt werden soll.

bond

Referenz auf ein Objekt vom Typ Bond aus der das Atom ermittelt wird, dessen SMILES String generiert werden soll.

Rückgabewert:

Referenz auf das Objekt vom Typ char, die auf den in dieser Funktion generierten SMILES String verweist.

Bemerkungen:

Zunächst wird überprüft, ob der Parameter bond existiert und der Wert der Variablen m_flag des Objektes bond, zur Kennzeichnung des Bearbeitungszustandes, ermittelt. Ist die Bindung bereits bearbeitet, so wird die Funktion mit der Rückgabe der Referenz auf den Speicher beendet; ansonsten wird aus der Variablen m_toatom des Objektes bond das zu analysierende Atom ermittelt und der Variablem m_flag, zur Kennzeichnung dass die Bindung bearbeitet ist, der Wert 1 zugewiesen.

Existiert der Parameter atom, so wird zunächst das Flag zur Kennzeichnung des Bearbeitungszustandes einer Bindung durch den Aufruf der Funktion SetBondFlag auf 0 gesetzt und der Speicher mit dem Wert 0 initialisiert.

Hat die Variable m_autoatom des Parameters atom, zur Kennzeichnung eines automatisch generierten Wasserstoffatoms, einen gültigen Wert, so wird die Funktion mit der Rückgabe der Referenz auf den Speicher verlassen.

Ist der Parameter atom das erste Objekt vom Typ Atom in der Referenzverknüpfung der Objekte vom Typ Atom und hat an der ersten Bindungsstelle eine cyclische Bindung, so wird das Zeichen % und der Wert der Variablen m_prozent des nullten Elementes der Variablen m_bond des Parameters atom in den SMILES String geschrieben. Anschließend wird das Symbol der Bindung aus der globalen Variablen g_bondstring entsprechend des Wertes der Variablen m_type des 0. Elementes der Variablen m_bond des Parameters atom ermittelt und ebenfalls in den SMILES String geschrieben. Sind die Werte des Bindungswinkels, gekennzeichnet durch die Variable m_rot, oder der Wert der Bindungslänge, gekennzeichnet durch die Variable m_radinput, für die 0. Bindung spezifiziert, so wird der Wert der Variablen m_rot vor dem Bindungssymbol und der Wert der Variablem m_rad nach dem Bindungssymbol in den SMILES String gespeichert, dabei wird der gesamte Ausdruck für die Bindung in eckige Klammern gefasst.

Das Atomsymbol wird aus der Variablen m_syn des Parameters atom ermittelt und gegebenenfalls in eckigen Klammern mit dem von der Standardmasse abweichenden Wert der Variablen m_mass vor dem Atomsymbol bzw. dem Wert der Variablen m_noligand, der Nummer des Ligandens spezifiziert, sowie nach dem Atomsymbol mit den Werten des verwendeten Zelltypes, aus der Variablen m_typeset des objektes m_cell des Parameters atom, der Ladung des Atoms, aus der Variablen m_charge des Parameters atom, und dem Buchstaben A, als Kennzeichnung einer axialen Position, ermittelt aus der Variablen m_case des Objektes m_cell des Parameters atom, spezifiziert.

Anschließend wird für alle Bindungen des Atoms, den Elementen der Variablen m_bond des Parameters atom, deren Variabel m_toatom auf ein gültiges Atom verweist oder deren Variabel m_prozent eine gültige Ringbindungsnummer hat, das Bindungssymbol aus der globalen Variablen g_bondstring entsprechend des Wertes der Variablen m_typeset der Bindung ermittelt. Enthält die Variable m_aromatic den Wert true, so wird ein Doppelpunkt als Bindungssymbol für eine aromatische Bindung ausgegeben. Sind die Werte des Bindungswinkels, gekennzeichnet durch die Variable m_rot, oder der Wert der Bindungslänge, gekennzeichnet durch die Variable m_radinput, für die jeweilige Bindung spezifiziert, so wird der Wert der Variablen m_rot vor dem Bindungssymbol und der Wert der Variablem m_rad nach dem Bindungssymbol in den SMILES String geschrieben, dabei wird der gesamte Ausdruck für die Bindung in eckige Klammern gefasst. Ist die Bindung eine cyclische Bindung, gekennzeichnet durch die Variable m_prozent, so wird das Zeichen % und der Wert der Variablen m_prozent in den SMILES String geschrieben.

Anschließend wird der Variablen m_flag der Rückbindung, ermittelt aus der Variablen m_backbond der jeweiligen Bindung, zur Kennzeichnung als bearbeitete Bindung der Wert 1 zugewiesen.
Für jede Bindung wird die Funktion Smiles mit der jeweiligen Bindung als Parameter aufgerufen und der Rückgabewert in dem SMILES String gespeichert.
Abschließend wird die Referenz auf den in dieser Funktion generierten SMILES String zurückgegeben.

StripH ()

Die Funktion entfernt die automatisch erzeugten Wasserstoffatome aus der virtuellen Molekülstruktur.

void StripH ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion ermittelt zunächst aus der Referenzverknüpfung, durch den Aufruf der Memberfunktion First, das erste Objekt vom Typ Atom und überprüft, ob es sich um ein automatisch generiertes Objekt vom Typ Atom, gekennzeichnet durch den Wert true der Membervariablen m_IsAutoAtom, handelt.
Ist das ermittelte Objekt ein automatisch generiertes Wasserstoffatom, so wird aus dem nullten Element der Variablen bond, ein Objekt vom Typ Bond, des ermittelten Wasserstoffatoms die Referenz m_backbond ermittelt und aus der Bindung die Referenz auf das From-Atom, an das das Wasserstoffatom gebunden ist, bestimmt. Anschließend werden die Eigenschaften der Bindung auf die Standardwerte zurückgesetzt, dazu wird den Variablen m_toatom, m_fromatom und m_backbond der Wert NULL, den Variablen M_tobond und m_frombond der Wert –1 zugewiesen, sowie den Variablen m_type und m_rad der Wert 0.
Das automatisch erzeugte Wasserstoffatom wird aus der Referenzverknüpfung der Objekte vom Typ Atom entfernt, sowie die Referenzverknüpfung der anderen Objekte vom Typ Atom aktualisiert und das automatisch generierte Objekt durch den Aufruf der Funktion delete zerstört.

Abschließend wird aus der Referenzverknüpfung der Objekte vom Typ Atom das nächste Atom ermittelt und die Überprüfung und Entfernung von automatisch generierten Wasserstoffatomen wiederholt.

Write(FILE *fp)

Funktion zur Speicherung der Daten eines Objektes vom Typ Atom.

bool Write(FILE *fp)

Parameter:
fp
Referenz auf ein Objekt vom Typ FILE, die auf die Datei verweist in der die Daten gespeichert werden.

Rückgabewert:
Die Funktion gibt bei fehlerfreier Speicherung der Daten den Wert true zurück.

Bemerkungen:

Sämtliche Daten werden durch den Aufruf der Funktion fprintf in das im Parameter fp referenzierte Objekt vom Typ FILE geschrieben. In der Datei wird der Anfang der Daten eines Atoms durch die Schlüsselworte „// BeginAtom“ gekennzeichnet, gefolgt vom Atomsymbol dessen Wert aus der Membervariablen m_sym ermittelt wird. Hat die Membervariable m_hybridstring einen gültigen Wert, so wird dieser nach dem Schlüsselwort „//HybridString“ in die Datei geschrieben. Anschließend werden die Eigenschaften des Atoms nach dem Schlüsselworten „// no, oxistate, oz, charge, ele, bind, mass, x,y,z“ entsprechend der Werte der Membervariablen m_no, m_oxistate, m_oz, m_charge, m_ele, m_bind, m_mass, m_x, m_y, und m_z in die Datei geschrieben. Zur Speicherung der Werte des Objektes m_cell wird die Funktion write des Objektes vom Typ Cell mit dem Parameter fp aufgerufen. Die Daten der Bindungen werden durch den Aufruf der Funktion write der Elemente der Membervariablen m_bond, deren Variable m_toatom einen gültigen Wert hat, gespeichert. Die Elemente der Membervariablen m_bond, deren Variable m_toatom keinen gültigen Wert hat, werden mit dem Schlüsselwort „//NoBond“ gekennzeichnet. Das Ende eines Datensatzes eines Objektes vom Typ Atom wird mit den Schlüsselwort „//EndAtom“ gekennzeichnet.

m_at

Öffentliche Referenz auf ein objekt vom Typ AtomTable, das die Standartwerte des Atoms enthält.

m_bind

Variable vom Typ int, die die Anzahl der Bindungen des Atoms enthält.

m_bond

Öffentliche Variable, die entsprechend des Wertes der Konstanten MAX_BOND Elemente vom Typ Bond enthält und die Bindungen des Atoms darstellt.

m_cell

Öffentliche Variable vom Typ Cell, die die Zelle des Atoms enthält.

m_charge

Öffentliche Variable vom typ int, deren Wert die Ladung des Atoms darstellt und bei der Erzeugung eines Objektes vom Typ Atom mit dem Wert 0 initialisiert wird.

m_ele

Öffentliche Variable vom Typ int, deren Wert die Anzahl der Elektronen darstellt.

m_hybridstring

Öffentliche Referenz auf eine Variable vom Typ char.

m_isautoatom

Öffentliche boolsche Variable, die bei der Erzeugung eines Objektes vom Typ Atom mit dem Wert false initialisiert wird. Der Wert true zeigt an, dass das Objekt vom Typ Atom automatisch generiert ist.

m_mass

Öffentliche Variable vom Typ double, deren Wert die Masse des Atoms darstellt.

m_next, m_prev

Öffentliche Referenzen auf das vorherige bzw. nächste Objekt vom Typ Atom. Mit den Referenzen m_next und m_prev wird die Referenzverknüpfung zwischen den Objekten vom Typ Atom realisiert.

m_no

Öffentliche Variable vom Typ int, die bei der Erzeugung eines Objektes vom Typ Atom mit dem Wert 0 initialisiert wird. Die Variable enthält den Wert der Nummerierung aller Atome einer virtuellen Molekülstruktur.

m_noligand

Öffentliche Variable vom Typ int, die bei der Erzeugung eines Objektes vom Typ Atom mit dem Wert 0 initialisiert wird. Der Wert der Variablen gibt die fortlaufende Nummer eines Liganden an.

m_oxistate

Öffentliche Variable vom Typ int, deren Wert ist die Oxidationsstufe des Atoms.

m_oz

Öffentliche Variable vom Typ int deren Wert die Ordnungszahl des Atoms ist.

m_sym

Öffentliche Referenz auf eine Variable vom Typ char, die das Atomsymbol enthält.

m_varcount

Öffentliche Variable vom Typ int, die mit dem Wert 0 bei der Erzeugung eines Objektes vom Typ Atom initialisiert wird. Der Wert der Variablen ist der Index gleicher Templates in einer virtuellen Molekülstruktur.

m_varname

Öffentliche Referenz auf eine Variable vom Typ char, die den Namen eines Templates enthält.

m_x, m_y, m_z

Öffentliche Variablen vom Typ double, die die Koordinaten des Atoms enthalten.

8.2 Methoden und Membervariablen der Klasse Bond

Der nachfolgende Abschnitt enthält eine alphabetisch geordnete, detaillierte Beschreibung jeder einzelnen Methode und Membervariablen, die in die Klasse Bond implementiert ist.

Bond ()

Konstruktor zur Erzeugung eines Objektes vom Typ Bond.

Parameter:
Die Funktion benötigt keine Parameter.

Bemerkungen:

Zunächst wird den Membervariablen m_type, deren Wert den Bindungstyp spezifiziert, und m_flag, deren Wert den Bearbeitungszustand der Bindung kennzeichnet, der Wert 0 zugewiesen. Die Membervariablen m_toatom und m_fromatom werden mit dem Wert NULL initialisiert, sowie den Membervariablen m_tobond und m_frombond der Wert –1 zugewiesen. Die Referenz auf die Rückbindung m_backbond wird mit dem Wert NULL initialisiert, sowie den Membervariablen m_aromatic und m_radimput der Wert false zugewiesen. Abschließend wird den Membervariablen m_rad, m_prozent und m_rot der Wert 0 zugewiesen.

Draw ()

Funktion zur visuellen Darstellung von Bindungen.

void Draw ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Hat die Membervariable m_toatom den Wert NULL oder die Membervariable m_flag einen Wert der die Bindung als bereits bearbeitet kennzeichnet, so wird die Funktion beendet.

Zunächst wird der Membervariablen m_flag der aufrufenden Instanz und der

Variablen m_flag der Membervariable m_backbond zur Kennzeichnung der Bindung als bearbeitet der Wert 1 zugewiesen. Aus der Membervariablen m_rad wird die Bindungslänge des from-Atoms, sowie aus der Variablen m_rad der Membervariablen m_backbond die Bindungslänge des to-Atoms ermittelt und falls diese gleich 0 ist, wird der Wert aus der globalen Variablen gKovRad durch den Aufruf der Funktion getKovRad mit dem entsprechenden Atom als Parameter ermittelt. Anschließend wird die Farbe der Bindung des From-Atoms, ermittelt aus den Variablen r, g und b der Variablen m_at des From-Atoms, durch den Aufruf der Funktion glColor3d festgelegt, sowie die Breite der Linie der Bindung durch den Aufruf der Funktion glLineWidth auf den Wert 10 festgelegt.
Zur Visualisierung der Bindung wird der Vektor ermittelt, der das From-Atom in den Ursprung projiziert und in Richtung des To-Atoms orientiert ist. Nach dem Aufruf der Funktion glBegin mit dem Parameter GL_LINES wird als Anfangspunkt der Linie zur Darstellung des Bindungsanteiles des From-Atoms der Ursprung und als Endpunkt der Punkt auf der Bindungslinie zwischen From-Atom und To-Atom entsprechend der Bindungslänge des From-Atoms, ermittelt aus dem Wert der Membervariablen m_rad, festgelegt und mit dem Aufruf der Funktion glEnd abgeschlossen. Mit den gleichen Funktionsaufrufen wird die Linie zur Darstellung des Bindungsanteils des To-Atoms erzeugt, nachdem die Farbe der Bindung des To-Atoms aus den Variablen r, g und b der Variablen m_at des To-Atoms ermittelt ist.

Read ()

Die Funktion liest Daten eines Objektes vom Typ Bond aus einer Datei und speichert die Werte der Eigenschaften der Bindung in den entsprechenden Membervariablen.

void Read(FGetS &fgs)

Parameter:
fgs
Objekt vom Typ FgetS, das die zu lesenden Daten enthält.

Rückgabewert*:*
Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion liest aus dem im Parameter fgs übergebenen Objekt vom Typ FgetS die Daten einer Bindung und speichert die Werte der Eigenschaften der Bindung in den Membervariablen m_type, m_tobond, m_frombond, m_aromatic und m_rad der aufrufenden Instanz des Objektes vom Typ Bond.

Write(FILE *fp)

Funktion zur Speicherung der Daten eines Objektes vom Typ Bond.

bool Bond:Write(FILE *fp)

Parameter:
fp
Referenz auf ein Objekt vom Typ FILE, die auf die Datei verweist in der die Daten gespeichert werden.

Rückgabewert:
Die Funktion gibt bei fehlerfreier Speicherung der Daten den Wert true zurück.

Bemerkungen:

Sämtliche Daten werden durch den Aufruf der Funktion fprintf in das im Parameter fp referenzierte Objekt vom Typ FILE geschrieben.
In der Datei wird der Anfang der Daten eines Objektes vom Typ Bond durch die Schlüsselworte „// BeginBond“ gekennzeichnet. Anschließend werden die Eigenschaften der Bindung nach den Schlüsselworten „fp,"// Bond: type, tobond, frombond, aromatic, rad, toatom, fromatom“ entsprechend der Werte der Membervariablen m_type, m_tobond, m_frombond, m_aromatic, m_rad, m_toatom->m_no und m_fromatom->m_no in die Datei geschrieben.
Das Ende eines Datensatzes eines Objektes vom Typ Bond wird mit dem Schlüsselwort „//EndBond“ gekennzeichnet.

m_aromatic

Öffentliche boolsche Variable zur Kennzeichnung einer aromatischen Bindung.

m_backbond

Öffentliche Referenz auf ein Objekt vom Typ Bond, die auf die Rückbindung verweist.

m_flag

Öffentliche Variable vom Typ int zur Kennzeichnung des Bearbeitungszustandes der Bindung.

m_fromatom

Öffentliche Referenz auf ein Objekt vom Typ Atom, die auf das „From-Atom“ der Bindung verweist.

m_frombond

Öffentliche Variable vom Typ int, deren Wert die Bindungsstelle des From-Atoms bezeichnet.

m_prozent

Öffentliche Variable vom Typ int, deren Wert die Bindungsnummer einer cyclischen Bindung bezeichnet.

m_rad

Öffentliche Variable vom Typ double, deren Wert den Radius des Atoms bezeichnet.

m_radimput

Öffentliche boolsche Variable zur Kennzeichnung einer optional spezifizierten Bindungslänge.

m_rot

Öffentliche Variable vom Typ double, deren Wert den dihedralen Winkel einer Bindung bezeichnet.

m_toatom

Öffentliche Referenz auf ein Objekt vom Typ Atom, die auf das To-Atom der Bindung verweist.

m_tobond

Öffentliche Variable vom Typ int, deren Wert die Bindungsstelle am To-Atom der Bindung bezeichnet.

m_type

Öffentliche Variable vom Typ int, deren Wert den Bindungstyp bezeichnet.

8.3 Methoden und Membervariablen der Klasse Cell

Der nachfolgende Abschnitt enthält eine alphabetisch geordnete, detaillierte Beschreibung jeder einzelnen Methode, Membervariablen und der Aufzählungsvariablen die in die Klasse Cell implementiert sind.

Cell ()

Konstruktor eines Objektes vom Typ Cell.

Parameter:
Die Funktion benötigt keinen Parameter.

Bemerkungen:

Bei der Konstruktion eines Objektes vom Typ Cell wird der Membervariablen m_flag inline der Wert false zugewiesen.

Read ()

Funktion zum Lesen von Daten eines Objektes vom Typ Cell.

void Read (FGetS &fgs)

Parameter:
fgs
Objekt vom Typ FgetS, das die zu lesenden Daten enthält.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion liest aus dem Parameter fgs die Daten eines Objektes vom Typ Cell und speichert die Werte der Eigenschaften der Zelle in den Membervariablen ",m_type,m_case,m_flag und m_n. Des weiteren speichert die Funktion die Werte der Koordinaten der Bindungsstellen der Zelle in den entsprechenden Elementen der Membervariablen m_xyz, sowie die Werte des Referenzpunktes in den Elementen der Membervariablen m_xyzref.

Write ()

Funktion zur Speicherung der Daten eines Objektes vom Typ Cell.

bool Write (FILE *fp)

Parameter:
fp
Referenz auf ein Objekt vom Typ FILE, die auf die Datei verweist in der die Daten gespeichert werden.

Rückgabewert:
Die Funktion gibt bei fehlerfreier Speicherung den Wert true zurück.

Bemerkungen:

Die Daten werden durch den Aufruf der Funktion fprintf in das im Parameter fp referenzierte Objekt vom Typ FILE geschrieben.
In der Datei wird der Anfang der Daten eines Objektes vom Typ Cell durch die Schlüsselworte „// BeginCell“ gekennzeichnet. Anschließend werden die Eigenschaften der Zelle nach den Schlüsselworten "// type, case, flag, n // n points x,y,z danach 3 point xyzref“ entsprechend der Werte der Membervariablen m_type, m_case, m_flag, m_n, sowie die Werte der Elemente der Membervariablen m_xyz und m_xyzref in die Datei geschrieben.
Das Ende eines Datensatzes eines Objektes vom Typ Cell wird mit dem Schlüsselwort „//EndCell“ gekennzeichnet.

m_case

Private Variable vom Typ char, deren Wert entweder die Position zweier Zelltypen zueinander oder die Bindungsposition einer Zelle festlegt.

Bemerkungen:

Die Variable verwendet die Werte der Aufzählungsvariablen Cases.

m_flag

Private Variable vom Typ char zur Kennzeichnung des Bearbeitungszustandes eines Objektes vom Typ Cell.

m_n

Private Variable vom Typ int, deren Wert die Anzahl der Bindungsstellen einer Zelle angibt.

m_type

Private Variable vom Typ char, deren Wert den verwendeten Zelltyp bezeichnet.

Bemerkungen:

Die Variable verwendet die Werte der Aufzählungsvariablen Types.

m_typeset

Private Variable vom Typ char, deren Wert den im SMILES String definierten Zelltypen bezeichnet.

m_xyz

Variable, deren Elemente die Koordinaten der Zelle enthalten.

Bemerkungen:

Die Variable hat entsprechend der Koordinatenachsen x, y und z 3*MAX_BOND Elemente vom Typ double, die die Koordinaten der Zelle repräsentieren.

m_xyzref

Variable, deren Elemente die Referenzpunkte enthält.

Bemerkungen:

Die Variable hat entsprechend der Koordinatenachsen x, y und z 3*4 Elemente vom Typ double, die die Referenzpunkte repräsentieren.

Cases

Öffentliche Aufzählungsvariable vom Typ enum, die den Werten NORMAL, STAGGERED, ECLIPSED die natürlichen ganzen Zahlen von 0 bis 2 und dem Wert AXIAL den Wert 4 zuweist.

Types

Aufzählungsvariable der unterschiedlichen Zelltypen.

Bemerkungen:

Die Variable ist eine öffentliche Aufzählungsvariable vom Typ enum. Bei der Initialisierung wird den Aufzählungsvariablen NOCELL, SINGLE, DIGONAL, TRIGONAL, TETRAEDRIC, PENTAGONAL und OCTAEDRIC ganzzahlige Werte zugewiesen. Dem Wert NOCELL wird der Wert 0 zugewiesen.

8.4 Methoden und Membervariablen der Klasse Mol

Der nachfolgende Abschnitt enthält eine alphabetisch geordnete, detaillierte Beschreibung jeder einzelnen Methode und Membervariablen, die in die Klasse Mol implementiert ist.

CleanAllUp ()

Die Funktion gibt den Speicherplatz der Membervariablen frei und aktualisiert die Referenzverknüpfung der Objekte vom Typ Mol.

void CleanAllUp ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Zunächst wird der Destruktor der Membervariablen m_bitmap aufgerufen und der Speicherplatz der Membervariablen m_smiles, m_name und m_varname freigegeben. Die Membervariable m_atom, die die Referenzverknüpfung der Objekte vom Typ Atom enthält, wird durch den Aufruf der Funktion delete des ersten Objektes vom Typ Atom zerstört.

Abschließend wird die Referenzverknüpfung der Objekte vom Typ Mol in den Variablen m_next des vorherigen Objektes und m_prev in dem nächsten Objekt vom Typ Mol aktualisiert.

Delete ()

Die Funktion zerstört die aufrufende Instanz vom Typ Mol und alle Objekte vom Typ Mol, auf die in der Referenzverknüpfung verwiesen wird.

void Delete ()

Parameter
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion ruft den Destruktor der aufrufenden Instanz auf und gibt den Speicherplatz frei. Der Aufruf wird für alle Objekte vom Typ Mol, auf die in der Membervariablen m_next verwiesen wird, wiederholt.

Draw ()

Die Funktion realisiert die visuelle Darstellung des Moleküls.

void Draw ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:
Die Funktion ruft zunächst die Funktion SetBondFlag und anschließend die Funktion Draw der Membervariablen m_atom, die die Referenzverknüpfung der Objekte vom Typ Atom enthält, auf.

GetAtom ()

Die Funktion ermittelt entsprechend der im Parameter no übergebenen Zahl ein Objekt vom Typ Atom.

*Atom GetAtom (int no)

Parameter:
no
Variable vom Typ int, die die Nummer des zu ermittelnden Objektes vom Typ Atom bezeichnet.

Rückgabewert:
Die Funktion gibt die Referenz auf das ermittelte Objekt vom Typ Atom zurück.

Bemerkungen:

Die Funktion ermittelt aus der Membervariablen m_atom, die die Referenzverknüpfung der Objekte vom Typ Atom realisiert, das Objekt vom Typ Atom, dessen Variable m_no mit der im Parameter no übergebenen Zahl übereinstimmt und gibt die Referenz auf das Objekt zurück.

GetAtomTable ()

Die Funktion ermittelt die aktuelle Atomtabelle.

*CAtomTable GetAtomTable ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion gibt eine Referenz auf ein Objekt vom Typ CAtomTable zurück.

Bemerkungen:

Die Funktion liefert als Rückgabewert die Referenz auf die Membervariable m_atomtable vom Typ CAtomTable.

GetName ()

Die Funktion ermittelt entweder den Namen oder den SMILES String des Moleküls und gibt die Referenz auf den ermittelten Wert zurück.

*char GetName ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion gibt die Referenz auf eine Variable vom Typ char zurück, die entweder auf den Namen des Moleküls oder auf den SMILES String verweist.

Bemerkungen:

Die Funktion gibt entweder die Referenz auf die Variable m_name, die den Namen des Moleküls enthält, oder falls kein Name existiert die Referenz auf die Variable m_smiles, die den SMILES String enthält, zurück.

IsOk ()

Die Funktion ermittelt den Bearbeitungszustand der aufrufenden Instanz.

bool IsOk ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion gibt die boolsche Variable m_ok zurück.

Bemerkungen:

Die Funktion gibt die boolsche Membervariable m_ok der aufrufenden Instanz, die den Bearbeitungszustand des Objektes bezeichnet, zurück.

MakeBond ()

Die Funktion realisiert die Bindung zwischen zwei Atomen durch die Spezifikation der Objekte vom Typ Bond.

void MakeBond (Atom *a1, Atom *a2, int bondtype, bool flag, bool onlyatom, int prozent)

Parameter:
a1
Referenz auf ein Objekt vom Typ Atom, das mit dem „ToAtom“, der Referenz auf das Objekt a2, verbunden wird.

a2
Referenz auf ein Objekt vom Typ Atom, das mit dem „FromAtom“, der Referenz auf das Objekt a1, verbunden wird.

bondtype
Variable vom Typ int, deren Wert den Typ der Bindung entsprechend der Reihenfolge der Bindungssymbole in der globalen Variablen g_bondstring bezeichnet.

flag
Variable vom Typ bool, die standardmäßig inline mit dem Wert TRUE initialisiert wird und den Zugriff auf das erste Element des Objektes m_bond vom Typ Bond, das zu dem Objekt der Referenz a1 gehört, steuert. Mit dem Wert False kann auf das Element zugegriffen werden.

onlyatom
Variable vom Typ bool, die standardmäßig inline mit dem Wert FALSE initialisiert wird und die Überprüfung des Objektes a1 auf ein Template steuert. Der Wert true analysiert das Objekt a1 als ein Atom.

prozent
Variable vom Typ int, deren Wert die Bindungsstelle in einem cyclischen System eindeutig kennzeichnet.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Zunächst wird durch den Aufruf der zu dem Objekt a1 gehörenden Funktion Count die Nummerierung der Referenzverknüpfung des Objektes a1 in der Variablen m_no aktualisiert.

Ist das durch die Referenz a1 gekennzeichnete Objekt vom Typ Atom ein Template, das durch die zum Objekt gehörenden Variablen m_varname und m_varcount charakterisiert ist, so wird aus dem Template die Referenz auf ein Objekt vom Typ Atom ermittelt, das entweder durch den Wert in der zum Objekt gehörenden Variablen m_noligand charakterisiert ist oder das erste Objekt vom Typ Atom innerhalb des Templates ist. Das ermittelte Objekt vom Typ Atom wird aus der Referenzverknüpfung der Objekte entfernt, die Referenz auf das Objekt vom Typ Bond, mit dem das Objekt mit dem vorherigen verknüpft ist, ermittelt und die Eigenschaften dieses Objektes auf die Initialisierungswerte zurückgesetzt; nachdem zuvor aus den zum Objekt gehörenden Variablen m_fromattom und m_frombond die Werte für die weitere Bindung ermittelt wurden.

Durch Aufruf der Funktion FreebondCount, des entweder im Parameter a1 referenzierten Objektes oder des aus dem Template ermittelten Objektes, wird überprüft, ob die durch den Parameter bondtypespezifizierte Bindung erlaubt ist und gegebenenfalls die niedrigste freie Bindung aus den Elementen der Variablen m_bond des Atoms ermittelt.

Ist die Referenz a2 auf ein Objekt vom Typ Atom ein Template, so finden die gleichen Algorithmen wie zuvor bei dem Parameter a1 beschrieben Anwendung, nur mit dem Unterschied, dass die Referenz auf ein Objekt vom Typ Atom, das durch den Wert 1 in der zum Objekt gehörenden Variablen m_noligand gekennzeichnet ist, ermittelt wird. Aus der zu diesem Objekt gehörenden Variablen m_bond wird das Objekt vom Typ Bond, das das Objekt vom Typ Atom mit dem vorherigen Objekt verbindet, ermittelt und aus den zum Objekt gehörenden Variablen m_frombond und m_fromatom die Eigenschaften für die in dieser Funktion erzeugten Bindung bestimmt. Das ermittelte Objekt vom Typ Atom wird aus der Referenzverknüpfung der Objekte entfernt und zerstört.

Anschließend wird die Variable m_bond, der aus den Parametern a1 und a2 ermittelten Objekte vom Typ Atom, bestimmt und die zuvor ermittelten Werte den objekteigenen Variablen m_frombond, m_tobond, m_fromatom, m_toatom und

m_type zugewiesen. Der Variablen m_rot wird der Wert der Membervariablen m_parse_bondangle und der Variablen m_rad der halbe Wert der Membervariablen m_parse_bondlength zugewisen. Existiert die Variable m_parse_bondlength mit einem gültigen Wert, so wird der Variablen m_radinput der Wert true zugewiesen. Abschließend werden die beiden Membervariablen wieder auf den Wert 0 zurückgesetzt.

Existiert ein gültiger Parameter prozent, so werden nur die Variablen des Objektes vom Typ Bond, das zu der Referenz des Parameters a1 gehört, wie zuvor spezifiziert und der Variablen m_prozent zusätzlich der Wert des Parameters prozent zugewiesen.

Mol ()

Der Konstruktor initialisiert die Membervariablen mit den Standartwerten, realisiert die Referenzverknüpfung zwischen den Objekten vom Typ Mol und initiiert die Analyse und Visualisierung des SMILES Strings.

Mol (const char *smiles, CAtomTable *cat, MolList *mollist, bool debug)

Parameter:

smiles

Referenz auf ein Objekt vom Typ const char, das auf den eingegebenen und zu analysierenden SMILES String verweist.

cat

Referenz auf ein Objekt vom Typ CAtomTable, das auf die zu verwendende Atomtabelle verweist.

Mollist

Referenz auf ein Objekt vom Typ MolList, das auf die verwendete Molliste verweist und dem eine auf das konstruierte Objekt vom Typ Mol verweisende Referenz hinzugefügt wird.

Debug

Boolsche Variable, die die Ausgabe von internen Informationen im Debugmodus steuert. Mit dem Wert „true“ können weitere Informationen ausgegeben werden.

Rückgabewert:

Der Konstruktor hat keinen Rückgabewert.

Bemerkungen:

Der Konstruktor eines Objektes vom Typ Mol initialisiert zunächst die Membervariablen mit den jeweiligen Standartwerten. Aus der Kopie des im Parameter smiles bezeichneten gültigen und existenten Objektes werden die Leerzeichen, Tabulatoren, Zeilenumbrüche und Returnbefehle, mit Ausnahme der Zeichenfolge, die durch eckige Klammern begrenzt sind, entfernt.
Anschließend erfolgt die weitere Analyse des SMILES Strings in der Funktion DoParse(), deren Ergebnis der Referenz m_aton zugewiesen wird. Ist die Referenz m_atom gültig, so erfolgt der Aufruf der Funktion SetUp() zur weiteren Visualisierung des Moleküls.

~Mol()

Der Destruktor zerstört die aufrufende Instanz.

Parameter:
Der Destruktor benötigt keine Parameter.

Rückgabewert:
Der Destruktor hat keinen Rückgabewert.

Bemerkungen:

Der Destruktor der aufrufenden Instanz vom Typ Mol ruft die Funktion CleanAllUp auf.

MolMass()

Die Funktion berechnet die Masse des Moleküls.

double Mol::MolMass()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion gibt eine Variable vom Typ double zurück.

Bemerkungen:

Die Funktion berechnet die Molekülmasse durch Summierung der Variablen mass des Objektes m_at vom Typ CAtomTable für jedes Objekt vom Typ Atom, das in der Referenzverknüpfung m_atom enthalten ist und gibt den berechneten Wert zurück.

DoParse ()

Die Funktion erzeugt in Abhängigkeit von der Analyse des SMILES Strings die Objekte der virtuellen Molekülstruktur.

Atom * DoParse (Atom *a, int bondexpected)

Parameter:

a
Referenz auf ein Objekt vom Typ Atom.

bondexpected
Variable vom Typ int, die den erwarteten Bindungstyp enthält.

Rückgabewert:
Die Funktion gibt die Referenz auf ein in der Funktion erzeugtes Objekt vom Typ Atom zurück.

Bemerkungen:

Die Funktion initialisiert zunächst die Membervariablen m_parse_noligand, m_parse_prozent, m_parse_bondangle, m_parse_bondlength, m_parse_charge, m_parse_mass und m_parse_axial mit dem Wert 0. Die Membervariable m_parse_celltype mit dem Wert –1, die Membervariable m_parse_lastbond mit dem Wert –2 und die Membervariable m_parse _multi mit dem Wert 1.

Die weitere Analyse des SMILES Strings erfolgt durch den Aufruf der Memberfunktion Lex und die Unterscheidung der Rückgabewerte.

Liefert die Funktion Lex den Wert ATOM, so wird ein neues Objekt vom Typ Atom mit dem Parameter einer Referenz auf ein Objekt vom Typ AtomTable entsprechend der Membervariablen m_buf erzeugt. Den Variablem m_mass, m_charge und m_noligand des neu erzeugten Objektes vom Typ Atom, sowie den Variablen m_type und m_case des Objektes m_cell vom Typ Cell werden die entsprechenden Membervariablen zugewiesen.
Ist das erzeugte Objekt vom Typ Atom das erste innerhalb des Aufrufes der Funktion DOParse erzeugte Objekt des Types, so wird überprüft ob es Bestandteil in einem cyclischen System ist und gegebenenfalls die Funktion MakeBond zur Erzeugung einer Bindung aufgerufen. Ist innerhalb der Funktion DoParse bereits mindestens ein Objekt vom Typ Atom erzeugt, so wird überprüft ob das zuletzt erzeugte Objekt Bindungen zulässt und gegebenenfalls die Bindung durch den Aufruf der Funktion MakeBond zwischen den Objekten erzeugt.
Durch Den Wert des Multiplikators in der Membervariablen m_parse_multi wird die entsprechende Anzahl gleicher Objekte vom Typ Atom erzeugt und verbindet das jeweils zuletzt erzeugte Objekt mit dem neu erzeugten Objekt durch die Ausführung der zuvor beschriebenen Schritte.

Liefert die Funktion Lex den Wert PROZENT, so wird das zuletzt erzeugte Objekt vom Typ Atom durch den Aufruf der Funktion MakeBond mit dem Objekt verbunden, das durch die gleiche Membervariable m_parse_prozent gekennzeichnet ist.

Liefert die Funktion Lex den Rückgabewert VAR, so wird zunächst die Referenz auf ein Objekt vom Typ Mol aus der Membervariablen m_mollist entsprechend des Variablennamens in der Membervariablen m_buf erzeugt und die in dem Objekt enthaltenen Referenzen auf die Objekte vom Typ Atom in eine lokale Referenz vom Typ Atom kopiert. Aus dem lokalen Objekt Atom wird durch den Aufruf der klasseneigenen Funktion StripH die Wasserstoffatome entfernt. Ist innerhalb der Funktion DoParse bereits ein Objekt vom Typ Atom erzeugt, so wird überprüft ob ein Objekt bereits den Variablennamen enthält und gegebenenfalls die Membervariable m_varcount um eins inkrementiert, sowie der Variablenname in der Variablen m_var des neuen Objektes vom Typ Atom gespeichert. Das zuletzt von der Funktion DoParse erzeugte Objekt vom Typ Atom wird mit einer Kopie des lokalen Objektes vom Typ Atom durch den Aufruf der Funktion MakeBond verbunden und anschließend das lokale Objekt zerstört.

Liefert die Funktion Lex den Rückgabewert ’(’ so wird zunächst die Membervariable m_parse_parcounter um eins inkrementiert und die Membervariablen m_parse_bondlength und m_parse_bondangle in lokalen Variablen gespeichert. Die weitere Analyse der Rückgabewerte der Funktion Lex erfolgt durch den rekursiven Aufruf der Funktion DoParse. Eine existierende lokale Kopie der Referenz auf ein Objekt vom Typ Atom, die von der rekursiv aufgerufenen Funktion DoParse zurückgegeben wird, wird durch den Aufruf der Funktion MakeBond mit der eventuell existierenden Referenz auf ein Objekt vom Typ Atom, das zuvor in dieser Funktion erzeugt worden ist, verbunden. Dieser Vorgang wird entsprechend des Wertes der Membervariablen m_parse_multi wiederholt und auf den Wert 1 zurückgesetzt.

Liefert die Funktion Lex den Rückgabewert ’)’, so wird die Membervariable m_parse_parcounter um eins dekrementiert und die Funktion mit der Rückgabe einer Referenz auf das lokal erzeugte Objekt vom Typ Aton beendet.

Liefert die Funktion Lex den Rückgabewert VARASSAIGN, so wird der lokalen boolschen Variablen varassign der Wert TRUE zugewiesen.

Liefert die Funktion Lex den Rückgabewert FEHLER, so wird der Membervariablen m_err eine Fehlermeldung zugewiesen und die Funktion Error aufgerufen.

Lex ()

Die Funktion analysiert den SMILES String und interpretiert die Zeichen entsprechend der Stellung in der linearen Zeichenfolge.

int Lex (Atom *delatom)

Parameter:

delatom
: Referenz auf ein Objekt vom Typ Atom.

Rückgabewerte:

Die Funktion gibt die folgenden Werte vom Typ static int zurück:

END
: Der Wert END wird zurückgegeben, wenn das Ende des zu analysierenden SMILES String erreicht ist.

ATOM
: Der Wert ATOM wird zurückgegeben, wenn die lexikologische Analyse einen großen Buchstaben bzw. einen großen Buchstaben und einen nachfolgenden kleinen Buchstaben des SMILES Strings als ein Atom identifiziert hat. Die als ATOM charakterisierten Buchstaben werden in der Membervariablen m_buf gespeichert.

VAR
: Der Wert VAR wird zurückgegeben, wenn die Buchstaben als ein Template identifiziert sind. Die mit VAR charakterisierten Buchstaben werden in der Membervariablen m_buf gespeichert.

VARASSIGN
: Der Wert VARASSIGN wird zurückgegeben, wenn der analysierte String durch die Zeichenfolge „=:“ als die Definition eines neuen Templates identifiziert ist und die Membervariable m_varname den Wert NULL hat. Der Name des Templates wird in der Membervariablen m_varname gespeichert.

PROZENT
: Der Wert PROZENT wird zurückgegeben, wenn das Zeichen '%' und eine nachfolgende Zahl im SMILES String identifiziert worden sind. Die dem Prozentzeichen folgende Zahl wird der Variablen m_parse_prozent zugewiesen.

Referenz auf das Zeichen „(„ bzw. „)“
: Wird bei der Analyse des Smiles Strings das Zeichen „(„ bzw. „)“ identifiziert, so wird eine Referenz auf das Zeichen als Rückgabewert zurückgegeben und der dem Zeichen folgende SMILES String in der Membervariablen m_buf gespeichert.

Bemerkungen:

Die Analyse der Zeichen des SMILES Strings, der durch die Referenz m_smilesp gekennzeichnet ist, erfolgt durch die Inkrementierung der lokalen Referenzen. Die durch die lokalen Referenzen gekennzeichneten Zeichen werden unter Berücksichtigung der Stellung im SMILES String interpretiert und in den entsprechenden Membervariablen gespeichert, sowie falls erforderlich in die entsprechenden Typen konvertiert.

Die Funktion zur lexikologischen Analyse des SMILES Strings überprüft bei den Rückgabewerten ATOM, VAR und der Referenz auf das Zeichen „)“, sowie nach dem Zeichen „]“ auf einen Multiplikator der durch das Zeichen „*“ gekennzeichnet ist und speichert die dem Multiplikationszeichen folgende Zahl in der Membervariablen m_parse_multi.

Wird bei der Analyse des SMILES Strings ein Zeichen das eine Bindung symbolisiert (:, -, =, #) identifiziert, so wird das Bindungssymbol entsprechend der Reihenfolge der Bindungssymbole in der globalen Variablen g_bondstring in eine Zahl vom Typ int konvertiert und in der Membervariablen m_parse_lastbond gespeichert.

Enthält der zu analysierende SMILES String das Zeichen „[„, so wird überprüft ob dem Zeichen eine Zahl folgt. Diese Zahl wird durch die nachfolgenden Bindungssymbole bzw. Buchstaben zur Symbolisierung eines Atoms charakterisiert und entsprechend der Charakterisierung in der Membervariablen m_parse_bondangle bzw. m_parse_mass gespeichert. Eine dem Bindungssymbol folgende Zahl wird in der Membervariablen m_parse_bondlength gespeichert. Symbolisieren die Zeichen ein Atom so wird die Funktion Lex rekursiv aufgerufen und die dem Atomsymbol folgende Zahl in der Membervariablen m_parse_celltype gespeichert. Die Symbole „+“ bzw. „-„ und eine optional folgende Zahl werden in den Typ int konvertiert und der Membervariablen m_parse_charge zugewiesen. Bei der weiteren Spezifikation eines Atoms innerhalb der eckigen Klammern mit den Buchstaben „a“ wird der Membervariablen m_parse_axial der Wert 1 zugewiesen.

SetAtomTable ()

Die Funktion weist den Atomen die im Parameter spezifizierte Atomtabelle zu.

void SetAtomTable(CAtomTable *atomtable)

Parameter:

atomtable
: Referenz auf ein Objekt vom Typ CAtomTable, das auf die zu verwendende Atomtabelle verweist.

Rückgabewert:
: Die Funktion ist vom Typ void.

Bemerkungen:

Die Funktion zerstört zunächst den Inhalt der Membervariablen m_atomtable und weist der Membervariablen anschließend das neu erzeugte Objekt vom Typ CAtomTable, auf das der Parameter atomtable verweist, zu. Abschließend wird die Variable m_at aller Objekte vom Typ Atom, die aus der Referenzverknüpfung der Membervariablen m_atom ermittelt werden, durch den Aufruf der Funktion GetEntry der Membervariablen m_atomtable aktualisiert.

SetCoord ()

Die Funktion berechnet die Koordinaten der Atome und der Bindungsstellen der Zellen unter Berücksichtigung der Position der Zellen der Nachbaratome.

bool SetCoord (Bond *bond)

Parameter:
bond
: Referenz auf ein Objekt vom Typ Bond, das auf die erste zu berechnende Bindung verweist.

Rückgabewert:
: Die Funktion vom Typ BOOL liefert bei Bindungen die einen Ringschluss erzeugen oder deren Koordinaten bereits berechnet sind bzw. in der Funktion fehlerfrei berechnet werden, so wie bei Bindungen die kein To-Atom enthalten den Rückgabewert true.

Bemerkungen:

Ist der Wert der Variablen m_prozent oder m_flag des Objektes bond nicht gleich 0 oder enthält die Variable m_toatom kein Objekt, so wird die Funktion ohne weitere Berechnungen mit dem Rückgabewert true beendet.

Aus der als Parameter übergebenen Referenz auf das Objekt bond werden die Objekte m_toatom, m_fromatom und m_backbond, sowie die Werte der Variablen m_frombond und m_tobond ermittelt und in lokalen Variablen gespeichert. Zur Kennzeichnung, das die Berechnung der Koordinaten ausgeführt ist, wird der Variablen m_flag des Objektes bond der Wert 1 zugewiesen.
Ist der Wert der Variablen m_rad der Objekte vom Typ Bond für die Hin- bzw. Rückbindung gleich 0, so wird der Variablen der Wert für die Länge der Standardbindung aus der globalen Variablen g_kovrad zugewiesen.

Durch den Aufruf der zu dem Objekt m_fromatom gehörenden Funktion SetCell werden die Koordinaten der Bindungsstellen der jeweils verwendeten Zelle ermittelt und anschließend eine Matrix konstruiert, die die Koordinaten der in dem Parameter bond spezifizierten Bindungsstelle des Objektes m_fromatom auf die x-Achse transformiert, die Koordinaten um 180° um die Y-Achse rotiert und entsprechend der Werte für die Bindungslänge in der Variablen m_rad der Objekte m_fromatom und m_toatom auf der x-Achse verschiebt. Des weiteren wird in der Matrix die Rotation innerhalb der Bindung, die durch die Variable m_rot des Objektes m_fromatom spezifiziert ist, integriert, sowie die Rotation um die X-Achse der Zelle des Objektes m_fromatom realisiert. Dabei wird bei den Zelltypen Trigonal und Tetraeder die zweite Bindungsstelle bzw. beim Oktaeder die dritte in die XZ-Ebene gedreht.

Anschließend wird aus der Variablen m_x des Objektes m_toatom die Koordinaten des Atoms und aus der Variablen m_xyz die Koordinaten der in dem Parameter bond spezifizierten Bindungsstelle ermittelt. Aus diesen Koordinaten wird eine Matrix erzeugt, die die Koordinaten der Bindungsstelle auf die X-Achse projiziert und in den Ursprung verschiebt, sowie die Zelle des Objektes m_toatom um die X-Achse dreht. Dabei wird die 0. Bindungsstelle der Zelltypen Trigonal und Tetraeder in die XZ-Ebene gedreht.

Anschließend wird durch Matrixmultiplikation der zuvor erzeugten Matrizen eine Matrix erzeugt, die die Gesamttransformation der neu zu berechnenden Koordinaten enthält. Bei der Berechnung der Koordinaten wird der Ursprung des Koordinatensystems durch Multiplikation mit der Gesamttransformationsmatrix in den Mittelpunkt des Atom m_fromatom transformiert und die Werte der Atomkoordinaten in den Variablen m_x, m_y und m_z des Objektes m_fromatom gespeichert. Die Werte der Bindungsstellen werden ebenfalls durch Matrixmultiplikation berechnet und in der Variablen m_xyz der Membervariablen m_cell des Objektes m_fromatom gespeichert.
Abschließend wird die Funktion SetCoord rekursiv für alle Bindungen des Objektes m_fromatom aufgerufen.

SetMinMaxCoord ()

Die Funktion ermittelt die minimalen, die maximalen und die mittleren Werte der Koordinaten des Moleküls aus allen Atomen.

void Mol::SetMinMaxCoord ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Zunächst wird dem ersten Element der Membervariablen m_mincoord und m_maxcoord der Wert der Variablen m_x, dem zweiten Element der Variablen m_mincord bzw. m_maxcoord wird der Wert der Variablen m_y, sowie dem dritten Element der Variablen m_mincoord bzw. m_maxcoord der Wert der Variablen m_z des Objektes m_atom zugewiesen.
Anschließend wird aus allen Objekten vom Typ Atom, die aus der Referenzverknüpfung m_next des jeweiligen Objektes vom Typ Atom ermittelt werden, die minimalen bzw. maximalen Werte der Koordinaten bestimmt und den entsprechenden Elementen der Membervariablen m_mincoord bzw. m_maxcoord zugewiesen.
Abschließend wird aus der Summe der jeweiligen Elemente der Membervariablen m_mincoord und m_maxcoord das arithmetische Mittel berechnet und dem entsprechenden Elementen der Membervariablen m_midcoord zugewiesen.

SetName ()

Die Funktion weist dem Molekül den Namen zu.

void SetName(const char *name)

Parameter:
name
Konstante Variable vom Typ char, die den Namen des Moleküls enthält.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:
Die Funktion gibt zunächst den Speicherplatz der Membervariablen m_name frei und speichert anschließend den im Parameter name übergebenen Wert in der Membervariablen m_name.

SetUp ()

Die Funktion realisiert aromatische Bindungen, aktualisiert die Nummerierung von Liganden und erzeugt den Ringschluss durch den Aufbau einer Bindung zwischen zwei Atomen in cyclischen Molekülsystemen. Des weiteren berechnet die Funktion die zu verwendenden Zelltypen der Atome, besetzt freie Orbitale mit Wasserstoffatomen und bestimmt die Koordinaten der Atome im Molekül.

void SetUp ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion ist vom Typ void.

Bemerkungen:

Zunächst werden die Objekte vom Typ Bond, die eine aromatische Bindung zwischen den Objekten vom Typ Atom darstellen, ermittelt und der Bindungstyp in der zu dem Objekt Bond gehörenden Variablen m_type von dem Wert 3, für eine aromatische Bindung, in eine alternierende Reihenfolge der Werte –2 bzw. 4, für eine Einfach- bzw. Doppelbindung geändert.

Anschließend wird die Nummerierung der Liganden in der Variablen m_noligand des Objektes vom Typ Atom ermittelt und gegebenenfalls durch eine kontinuierliche natürliche aufsteigende Zahlenreihe aktualisiert.

Nachdem die Variablen m_x, m_y, m_z, für die Koordinaten der Atome, der Objekte vom Typ Atom mit dem Wert 0 initialisiert sind wird das Objekt a1 vom Typ Atom ermittelt, dessen Bindung durch die Variable m_prozent des Objektes vom Typ Bond und der Variablen m_toatom, das kein Objekt enthält, charakterisiert ist. Anschließend wird das Objekt a2 vom Typ Atom ermittelt dessen Bindung durch das gleiche Objekt vom Typ Bond dargestellt wird und in der Variable m_prozent den gleichen Wert wie das zuvor ermittelte Objekt a1 hat. Den Variablen m_type, m_toatom, m_tobond, m_fromatom, m_frombond, für die Eigenschaften der zuvor ermittelten Bindungen, werden die lokal bestimmten Werte zugewiesen und in der Variablen m_backbond die Referenzverknüpfung der Objekte für die Hin- bzw. Rückbindung realisiert.

Anschließend wird aus dem, zu dem Objekt a1 vom Typ Atom gehörenden, Objekt m_cell vom Typ Cell die Anzahl der Orbitale ermittelt und die Anzahl der Elektronen, aus den zu dem Objekt a1 gehörenden Variablen m_ele und m_charge, bestimmt und auf die Orbitale, die Elemente eines lokalen Feldes, vom Typ int verteilt. Aus diesem lokalen Feld wird die Anzahl der vollbesetzten Orbitale bestimmt.

Liefert die Funktion FreeBondCount des Objektes a1 einen Wert größer 0, so werden die freien Bindungsmöglichkeiten mit Wasserstoffatomen besetzt. Dazu wird ein

neues Objekt vom Typ Atom erzeugt und mit der Funktion MakeBond an das vorhandene Objekt gebunden, sowie der Variablen m_autoatom des neu erzeugten Objektes der Wert true zugewiesen.

Aus der Summe der vollbesetzten Orbitale und den Bindungen des Objektes a1, deren Objekte vom Typ Bond durch die Variablen m_toatom oder m_type charakterisiert sind, wird der Zelltyp des Atoms a1 bestimmt und in der Variablen m_type des Objektes m_cell vom Typ Cell gespeichert.

Die Berechnung der Koordinaten der Atome und der Bindungsstellen der Zellen erfolgt durch den Aufruf der Funktion SetCoord, nachdem zuvor der Variablen m_flag des Objektes vom Typ Bond für alle Bindungen durch den Aufruf der Funktion SetBondFlag des Objektes vom Typ Atom der Wert 0 zugewiesen ist. Abschließend werden die minimalen und maximalen Koordinaten der Atome, sowie das arithmetische Mittel der Koordinaten durch den Aufruf der Funktion SetMinMaxCoord bestimmt.

Smiles ()

Die Funktion ermittelt den SMILES String.

char *Smiles ()

Parameter:
Die Funktion benötigt keine Parameter.

Rückgabewert:
Die Funktion gibt die Referenz auf die Membervariable m_smiles vom Typ char zurück.

Bemerkungen:

Die Funktion liefert die Referenz auf die Membervariable m_smiles die den SMILES String enthält.

Write ()

Funktion zur Speicherung der Daten eines Objektes vom Typ Mol.

bool Write(FILE *fp)

Parameter:
fp
Referenz auf ein Objekt vom Typ FILE, die auf die Datei verweist in der die Daten gespeichert werden.

Rückgabewert:
Die Funktion liefert den Wert true bei fehlerfreier Speicherung bzw. den Wert false bei fehlerhafter Speicherung.

Bemerkungen:

Zur Kennzeichnung der gespeicherten Daten wird zunächst das Schlüsselwort „// BeginMol“ in das im Parameter fp referenzierte Objekt vom Typ FILE geschrieben. Anschließend wird die Membervariable m_smiles, die den SMILES String enthält, in das FILE Objekt geschrieben und durch die Schlüsselworte „// BeginSmilesString“ bzw. „// EndSmilesString“ gekennzeichnet. Hat die Membervariable m_name, die den Namen des Moleküls enthält, einen gültigen Wert, so wird die Membervariable m_name nach dem Schlüsselwort „// Name“ in die Datei geschrieben. Abschließend wird für alle Objekte vom Typ Atom, die aus der Referenzverknüpfung in der Membervariablen m_atom ermittelt werden, die Funktion Wirte der Objekte der Instanz Atom aufgerufen und zum Abschluss das Schlüsselwort „// EndMol“ in die Datei geschrieben.

m_atom

Referenz auf ein Objekt vom Typ Atom.

Bemerkung:

Die öffentliche Referenz auf ein Objekt vom Typ Atom wird bei der Erzeugung eines Objektes vom Typ Mol mit dem Wert NULL initialisiert und anschließend das Ergebnis der Funktion DoParse() zugewiesen.

m_atomtable

Referenz auf die verwendete Atomtabelle.

Bemerkung:

Die öffentliche Referenz auf ein Objekt vom Typ CAtomTable wird bei der Erzeugung eines Objektes vom Typ Mol mit dem Wert NULL initialisiert und anschließend der Wert der entsprechenden, im Konstruktor als Parameter übergebenen, Referenz zugewiesen.

m_buf

Membervariable zum Zwischenspeichern von Zeichen.

Bemerkungen:

In der öffentlichen Membervariablen vom Typ char können maximal 1024 Zeichen gespeichert werden.

m_debug

Boolsche Variable zur Steuerung der Ausgabe von zusätzlichen Informationen im Debug – Modus.

Bemerkungen:

Die öffentliche boolsche Variable wird bei der Erzeugung eines Objektes vom Typ Mol im Konstruktor als Parameter übergeben, mit dem Wert „true“ können zusätzliche Informationen ausgegeben werden.

m_kovrad

Referenz auf die verwendete Tabelle der Atomradien.

Bemerkung:

Die öffentliche Referenz auf ein Objekt der Klasse KovRad wird bei der Erzeugung eines Objektes vom Typ Mol mit dem Wert NULL initialisiert.

m_mollist

Referenz auf die zu verwendende Molekülliste.

Bemerkung:

Die öffentliche Referenz auf ein Objekt vom Typ MolList wird bei der Erzeugung eines Objektes vom Typ Mol mit dem Wert NULL initialisiert und anschließend auf den im Konstruktor übergebenen Parameter gesetzt.

m_name

Referenz auf die Variable die den Namen des Moleküls enthält.

Bemerkung:

Die öffentliche Referenz auf ein Objekt vom Typ char wird bei der Erzeugung eines Objektes vom Typ Mol mit NULL initialisiert.

m_next, m_prev

Referenzen auf das vorherige bzw. nächste Objekt vom Typ Mol.

Bemerkungen:

Die öffentlichen Referenzen auf ein Objekt vom Typ Mol werden bei der Erzeugung eines Objektes vom Typ Mol zunächst mit NULL initialisiert.

m_parse_axial

Die Membervariable zeigt an, ob das zu analysierende Atom mit der axialen oder der äquatorialen Position verknüpft wird.

Bemerkungen:

Die öffentliche Membervariable vom Typ char wird in der Funktion DoParse mit dem Wert 0 initialisiert.

m_parse_bondangle

Die Membervariable enthält den dihedralen Winkel zwischen zwei benachbarten Zellen.

Bemerkungen:

Die Membervariable vom Typ double wird in der Funktion DoParse mit dem Wert 0 initialisiert.

m_parse_bondlength

Die Membervariable enthält den Wert der Bindungslänge des zu analysierenden Atoms.

Bemerkungen:

Die öffentliche Membervariable vom Typ double wird in der Funktion DoParse mit dem Wert 0 initialisiert.

m_parse_celltype

Die Membervariable charakterisiert den Zelltyp des zu analysierenden Atoms.

Bemerkungen:

Die öffentliche Membervariable vom Typ int wird in der Funktion DoParse mit dem Wert –1 initialisiert.

m_parse_charge

Die Membervariable stellt den Wert der Ladung des zu analysierenden Atoms dar.

Bemerkungen:

Die öffentliche Membervariable vom Typ int wird in der Funktion DoParse mit dem Wert 0 initialisiert.

m_parse_lastbond

Der Wert der Membervariablen zeigt die nächste freie Bindungsstelle am zu analysierenden Atom an.

Bemerkung:

Die öffentliche Membervariable vom Typ int zeigt die Nummer der nächsten freien Bindungsstelle an und wird in der Funktion DoParse mit dem Wert –2 initialisiert.

m_parse_mass

Der Wert der Membervariablen zeigt die Masse des zu analysierenden Atoms an.

Bemerkungen:

Die Membervariable vom Typ double wird in der Funktion DoParse mit dem Wert 0 initialisiert.

m_parse_noligand

Die Membervariable kennzeichnet die Reihenfolge der terminalen Liganden L in einem Template.

Bemerkungen:

Die öffentliche Variable vom Typ int wird in der Funktion DoParse mit dem Wert 0 initialisiert.

m_parse_parcounter

Membervariable zur Überprüfung der Anzahl der öffnenden und schließenden Klammern innerhalb eines SMILES Strings.

Bemerkungen:

Öffentliche Variable vom Typ int, die bei der Erzeugung eines Objektes vom Typ mol mit dem Wert 0 initialisiert wird. Ein Wert größer Null zeigt an, dass der zu analysierende String mehr öffnende Klammern als schließende Klammern enthält bzw. ein Wert kleiner als Null bedeutet, dass der String mehr schließende als öffnende Klammern enthält. Ein Wert ungleich von Null wird durch eine Fehlermeldung angezeigt.

m_parse_prozent

Membervariable zur Kennzeichnung einer cyclischen Bindung in einem Molekül.

Bemerkungen:

Die öffentliche Membervariable vom Typ int wird in der Funktion DoParse mit dem Wert 0 initialisiert.

m_on_err_delete_atom1, m_on_err_delete_atom2

Referenzen auf Objekte vom Typ Atom

Bemerkungen:

Die öffentlichen Referenzen auf Objekte vom Typ Atom werden bei der Erzeugung eines Objektes vom Typ Mol mit NULL initialisiert.

m_selected

Der Wert der Membervariablen kennzeichnet den Auswahlstatus des Objektes vom Typ Mol aus der Molekülliste.

Bemerkung:

Die boolsche Variable wird bei der Erzeugung eines Objektes vom Typ Mol mit dem Wert false initialisiert.

m_smiles, m_smilesp

Referenzen auf den eingegebenen SMILES – String.

Bemerkungen:

Öffentliche Referenzen die auf ein Objekt vom Typ char zeigen und bei der Erzeugung eines Objektes vom Typ Mol auf ein Objekt gesetzt werden, das den im Konstruktor übergebenen Wert beinhaltet.

m_varname

Referenz auf den Namen eines Templates.

Bemerkungen:

Öffentliche Referenz auf ein Objekt vom Typ char, die bei der Erzeugung eines Objektes vom Typ Mol mit dem Wert NULL initialisiert wird und der in der Funktion Lex der Name des Templates zugewiesen wird.

8.5 Oxidationsstufen der Atome

1 H -I, I																	2 He
3 Li I	4 Be II											5 B III	6 C -IV, II,IV	7 N -II, -I, II bis V	8 O -II, -I I	9 F -I	10 Ne
11 Na I	12 Mg II											13 Al III	14 Si -II, II, IV	15 P -III, III, V	16 S -II, II, IV, VI	17 Cl -I, I, III, V, VII	18 Ar
19 K I	20 Ca II	21 Sc III	22 Ti -I, II bis IV	23 V -I, -III I bis V	24 Cr -II, -I, I bis VI	25 Mn -III, -II, -I, I bis VII	26 Fe -II, -I, I bis VI	27 Co -I, I bis V	28 Ni -I, I bis IV	29 Cu I bis IV	30 Zn II	31 Ga I - III	32 Ge II, IV	33 As -III, III, V	34 Se -II, IV, VI	35 Br -I, I, III, V, VII	36 Kr II
37 Rb I	38 Sr II	39 Y III	40 Zr I bis IV	41 Nb -I, I bis V	42 Mo -II, -I, I bis VI	43 Tc -III, -I, I bis VII	44 Ru -II, I bis VIII	45 Rh -I, I bis VI	46 Pd II, III, IV	47 Ag I bis III	48 Cd I, II	49 In I - III	50 Sn II, IV	51 Sb -III, III, V	52 Te -II, II, IV, VI	53 J -I, I, III, V, VII	54 Xe II, IV, VI, VIII
55 Cs I	56 Ba II	72 Hf I, III, IV	73 Ta -I, I bis V	74 W -II, -I, I bis VI	75 Re -III, -I, I bis VII	76 Os -II, I bis VIII	77 Ir -I, I bis VI	78 Pt II, IV, V, VI	79 Au I, II, III, V	80 Hg I, II	81 Tl I, III	82 Pb II, IV	83 Bi III, V	84 Po	85 At	86 Rn	
87 Fr I	88 Ra II																

Lanthanoide: III
Actinoide: III

8.6 Zelltypen der virtuellen Atome

1 H mo																	2 He
3 Li mo	4 Be di											5 B tr	6 C di,tr,te	7 N di,tr,te	8 O di, tr	9 F te	10 Ne
11 Na mo	12 Mg di											13 Al tr	14 Si te	15 P te, pe	16 S te, oc	17 Cl te	18 Ar
19 K mo	20 Ca di	21 Sc tr	22 Ti te	23 V pe	24 Cr oc	25 Mn pe	26 Fe te	27 Co tr	28 Ni di	29 Cu mo	30 Zn di	31 Ga tr	32 Ge te	33 As te, pe	34 Se te, oc	35 Br te	36 Kr
37 Rb mo	38 Sr di	39 Y tr	40 Zr te	41 Nb pe	42 Mo oc	43 Tc pe	44 Ru te	45 Rh tr	46 Pd di	47 Ag mo	48 Cd di	49 In tr	50 Sn te	51 Sb te, pe	52 Te te, oc	53 J te	54 Xe
55 Cs mo	56 Ba di		72 Hf te	73 Ta pe	74 W oc	75 Re pe	76 Os te	77 Ir tr	78 Pt di	79 Au mo	80 Hg di	81 Tl tr	82 Pb	83 Bi	84 Po	85 At	86 Rn
87 Fr mo	88 Ra di																

Lanthanoide: tr
Actinoide: tr

mo : linear (monohedral)
di : linear (dihedral)
tr : trigonal
te : tetragonal
pe : pentagonal
oc : octahedral

8.7 Farbwerte der virtuellen Atome

1 H R: 255 G: 255 B: 255																	2 He R: 255 G: 192 B: 203
3 Li R: 178 G: 34 B: 34	4 Be R: 0 G: 200 B: 0											5 B R: 0 G: 255 B: 34	6 C R: 200 G: 200 B: 200	7 N R: 143 G: 143 B: 255	8 O R: 250 G: 0 B: 0	9 F R: 218 G: 165 B: 32	10 Ne R: 255 G: 160 B: 171
11 Na R: 0 G: 0 B: 255	12 Mg R: 34 G: 139 B: 34											13 Al R: 128 G: 128 B: 144	14 Si R: 176 G: 165 B: 64	15 P R: 255 G: 165 B: 0	16 S R: 255 G: 200 B: 50	17 Cl R: 0 G: 255 B: 0	18 Ar R: 255 G: 128 B: 139
19 K R: 32 G: 32 B: 255	20 Ca R: 68 G: 138 B: 68	21 Sc R: 60 G: 120 B: 60	22 Ti R: 155 G: 155 B: 155	23 V R: 155 G: 187 B: 123	24 Cr R: 128 G: 128 B: 144	25 Mn R: 100 G: 100 B: 136	26 Fe R: 255 G: 165 B: 32	27 Co R: 128 G: 128 B: 255	28 Ni R: 165 G: 74 B: 74	29 Cu R: 165 G: 132 B: 132	30 Zn R: 160 G: 60 B: 60	31 Ga R: 96 G: 96 B: 112	32 Ge R: 144 G: 166 B: 64	33 As R: 255 G: 155 B: 0	34 Se R: 255 G: 200 B: 40	35 Br R: 170 G: 40 B: 40	36 Kr R: 255 G: 96 B: 139
37 Rb R: 0 G: 32 B: 255	38 Sr R: 32 G: 106 B: 34	39 Y R: 40 G: 80 B: 40	40 Zr R: 123 G: 123 B: 123	41 Nb R: 123 G: 187 B: 89	42 Mo R: 96 G: 98 B: 112	43 Tc R: 68 G: 68 B: 104	44 Ru R: 223 G: 133 B: 0	45 Rh R: 96 G: 96 B: 223	46 Pd R: 134 G: 42 B: 42	47 Ag R: 135 G: 100 B: 100	48 Cd R: 150 G: 40 B: 40	49 In R: 64 G: 64 B: 80	50 Sn R: 144 G: 166 B: 32	51 Sb R: 255 G: 144 B: 0	52 Te R: 255 G: 200 B: 30	53 J R: 160 G: 32 B: 240	54 Xe R: 255 G: 64 B: 107
55 Cs R: 0 G: 16 B: 208	56 Ba R: 255 G: 165 B: 0		72 Hf R: 89 G: 89 B: 89	73 Ta R: 89 G: 187 B: 57	74 W R: 66 G: 66 B: 80	75 Re R: 36 G: 36 B: 72	76 Os R: 191 G: 101 B: 0	77 Ir R: 64 G: 64 B: 191	78 Pt R: 102 G: 10 B: 10	79 Au R: 103 G: 67 B: 67	80 Hg R: 130 G: 30 B: 30	81 Tl R: 32 G: 32 B: 48	82 Pb R: 144 G: 168 B: 0	83 Bi R: 255 G: 116 B: 0	84 Po R: 255 G: 200 B: 20	85 At R: 128 G: 32 B: 208	86 Rn R: 255 G: 32 B: 75
87 Fr R: 0 G: 8 B: 200	88 Ra R: 223 G: 132 B: 8																

Abkürzungen :
R : Rot
G : Grün
B : Blau

8.8 Die Werte der kovalenten Radien

Die nachfolgende Tabelle Zeigt die Werte der kovalenten Atomradien, die das Programm WinSmiles - 3D den virtuellen Atomen zuweist.

Atomsymbol	Oxidationsstufe	Zelltyp	Kovalenter Atomradius [pm]
L	I	mo	34,3
H	I	mo	34,3
Li	I	mo	150,2
Na	I	mo	179,3
K	I	mo	219,8
Rb	I	mo	237,4
Be	II	di	88,0
Mg	II	di	136,0
Ca	II	di	174,0
Sr	II	di	191,0
B	III	di	83,1
B	III	tr	83,1
Al	III	tr	125,0
Ga	III	tr	125,0
In	III	tr	150,0
C	IV	di	71,0
C	IV	tr	74,7
C	IV	te	76,3
C	II	tr	75.0
Si	IV	tr	108,0
Si	IV	te	110,6
Si	II	tr	108.0
Ge	IV	tr	117,0
Ge	IV	te	119,2
Sn	II	tr	135,0
Sn	IV	te	137,9
N	III	tr	68,6
N	III	te	69,5
P	III	tr	101,0
P	III	te	101,8
P	V	te	99,0
P	V	pe	100,6
As	III	te	121,0
As	V	pe	110,0
Sb	III	te	141,0
Sb	V	pe	130,0
O	II	te	68,6
S	II	te	101,6
S	VI	te	96,0
S	VI	pe	97,0
S	VI	oc	102,9
Se	II	te	117,0
Se	IV	pe	103,0
Te	II	te	137,0
Te	IV	pe	126,0
F	I	te	66,0
Cl	I	te	97,4
cl	III	pe	100,0
Cl	VII	te	95,0
Cl	VII	oc	102,0
Br	I	te	108,4
Br	III	pe	111,0
Br	V	oc	108,0
Br	VII	te	103,0
I	I	te	128,4
I	V	oc	123,0
I	VII	oc	120,0

Abkürzungen:

mo	monohedral
di	dihedral
tr	trigonal
te	tetrahedral
pe	pentagonal
oc	octahedral

Atomsymbol	Oxidationsstufe	Zelltyp	Kovalenter Atomradius [pm]
L	I	mo	34,3
H	I	mo	34,3
Li	I	mo	150,2
Na	I	mo	179,3
K	I	mo	219,8
Rb	I	mo	237,4
Be	II	di	88,0
Mg	II	di	136,0
Ca	II	di	174,0
Sr	II	di	191,0
B	III	di	83,1
B	III	tr	83,1
Al	III	tr	125,0
Ga	III	tr	125,0
In	III	tr	150,0
C	IV	di	71,0
C	IV	tr	74,7
C	IV	te	76,3
C	II	tr	75.0
Si	IV	tr	108,0
Si	IV	te	110,6
Si	II	tr	108.0
Ge	IV	tr	117,0
Ge	IV	te	119,2
Sn	II	tr	135,0
Sn	IV	te	137,9
N	III	tr	68,6
N	III	te	69,5
P	III	tr	101,0
P	III	te	101,8
P	V	te	99,0
P	V	pe	100,6
As	III	te	121,0
As	V	pe	110,0
Sb	III	te	141,0
Sb	V	pe	130,0
O	II	te	68,6
S	II	te	101,6
S	VI	te	96,0
S	VI	pe	97,0
S	VI	oc	102,9
Se	II	te	117,0
Se	IV	pe	103,0
Te	II	te	137,0
Te	IV	pe	126,0
F	I	te	66,0
Cl	I	te	97,4
cl	III	pe	100,0
Cl	VII	te	95,0
Cl	VII	oc	102,0
Br	I	te	108,4
Br	III	pe	111,0
Br	V	oc	108,0
Br	VII	te	103,0
I	I	te	128,4
I	V	oc	123,0
I	VII	oc	120,0

Abkürzungen:

mo	monohedral
di	dihedral
tr	trigonal
te	tetrahedral
pe	pentagonal
oc	octahedral

Gedruckt auf alterungsbeständigem,
holz- und säurefreiem Papier.
(DIN-ISO 9706)