Ingo Klöckl

OCHEM

Benutzerhandbuch OCHEM Version 3-0e 2001-04-10

18. April 2001

Ingo Klöckl
Institut für Kernchemie Universität Mainz
Fritz Strassmann-Weg 2
D-55099 Mainz
kloeckl@vkcmzd.chemie.uni-mainz.de
ingo.kloeckl@2k-software.de
Gesetzt mit LaTeX und OCHEM Version 3-0e 2001-04-10 .

Inhaltsverzeichnis

Die Reaktionsschemata

1. Einführung

1.1 Übersicht

Das vorliegende Paket soll den Chemiker in die Lage versetzen, organische Strukturformeln sowie ganze Reaktionsschemata in LATEX-Texte zu integrieren. Ziel ist, anhand einer einfachen Beschreibung der Formeln und Reaktionen die genaue Anordnung der einzelnen Formeln innerhalb des Schemas einem Chemie-Compiler zu überlassen, ähnlich wie TEX anhand der Befehle Textstücke selbständig anordnet. Analog der Erzeugung von Graphiken mit MetaPost werden die chemischen Beschreibungen mit der LATEX-Bearbeitung in eine Textdatei geschrieben, die nach dem Compilieren TEX-Fragmente mit den Formelbildern liefert. Die Fragmente werden automatisch an der Stelle der Chemiebeschreibung in das LATEX-Dokument eingefügt. Es ist nicht erforderlich, diese Fragmente neben den eigentlichen Dokumentdateien aufzuheben, da sie jederzeit durch den Compiler neu erzeugt werden könn-n en, so wie auch Metapost-Graphiken wieder aus den Sourcen generierbar sind. Haben sich die Formeln jedoch nicht geändert, ist eine Neucompilation nicht erforderlich und der Schritt der Compilation kann ausgespart werden. In diesem Falle müssen die Fragmente natürlich noch existieren.

Die Aufmerksamkeit der Formelzeichnung richtet sich vor allem auf eine planare Wiedergabe, worunter auch gängige schematische Stereodarstellungen verstanden werden, wie Sesselkonformationen des Zyklohexans oder Verbrückungen bei Polyzyklen, zum Beispiel des Morphins oder Crassanins. Da das Paket von einer geometrischen Beschreibung der Strukturen ausgeht, können komplizierte Gerüste, wie sie beispielsweise Alkaloiden zu eigen sind, gewisse Probleme aufwerfen, wenn Sie eine räumliche Darstellung anfertigen wollen. Sie müssen in einem solchen Falle über die gewöhnliche Menge an Bindungswinkeln und -längen hinaus individuelle Berechnungen dieser Werte anstellen und fleißigen Gebrauch von `saveXY` und `restoreXY` machen[1]. OCHEM ist in dieser Hinsicht kein molecular modelling-Tool (seufz). Ich glaube, alle zur Beschreibung auch solcher Strukturen erforderlichen (graphischen) Primitive bereitgestellt zu haben; wenn Sie keine Möglichkeit sehen, bestimmte Strukturen zu erzeugen, bin ich über einen Hinweis dankbar.

Mechanistiker können mit Hilfe kleiner gebogener Pfeile Elektronenverschiebungen andeuten. Sie werden sehen, daß die unzähligen Mausbewegungen bei Benutzung eines WYSIWYG-Graphikeditors eine getreue Repräsentierung in Textform besitzen und die Formelbeschreibung rasch anwachsen lassen können. Auch hier gilt: falls jemand eine gute Idee zur Beschreibung der Pfeilchen hat, möge er sie vorbringen (am besten mit Implementierung :-))

Die Formeln werden mit maximaler Qualität in Form eines Fragments mit PostScript-Code in die TEX-Dateien geschrieben, sodaß sie nur mit einem PostScript-Treiber und -drucker sichtbar sind! Verfügen Sie über keinen Zugang zu PostScript-Druckern,

[1] Nichts für Warmduscher und Schattenparker also.

können Sie mit dem Programm GV oder seinen Derivaten allerdings die PS-Ausgabedateien in andere Druckerformate konvertieren. Die Texte, die in den Formeln auftreten, werden mit TEX gesetzt und erfordern somit kein PostScript. Die Entscheidung zugunsten von PostScript hängt (neben der Menge an freier Zeit) vor allem mit den bescheidenen graphischen Fähigkeiten des originalen LATEX zusammen, das keine Linien unter beliebigem Winkel oder Kreise mit freiem Radius zeichnen kann. Der Compiler erzeugt jedoch eine generische Ausgabe, die vom Modul `be.pm` in verschiedene finale Formate umgesetzt wird. Hier ist auch ein LATEX-Ausgabeformat vorgesehen, das nur LATEX-Mittel nutzt, aber aus den erwähnten Gründen im allgemeinen keine so schönen Formeln liefert (vergleichen Sie selbst, in welchen Fällen die Ausgabe hochwertig ist und in welchen Mängel auftreten). Dieses native LATEX-Ausgabeformat soll künftig jedoch verbessert werden, wobei vermutlich das Paket `eepic` zum Einsatz kommen wird.

1.2 Installation

Der Chemie-Compiler besteht aus einem Perl-Script, das die eigentliche Compilierung vornimmt, sowie einigen Modulen, ebenfalls in Perl geschrieben. Sie benötigen die folgenden Dateien in einem Perl-gerechten Verzeichnisbaum:

```
chemie.pl
streambuf.pm
loc.pm
bbox.pm
be.pm
mncyclib.pm
bicyclib.pm
```

Das File `chemie.pl` stellt den eigentlichen Compiler dar und muß in ein Verzeichnis kopiert werden, das im Suchpfad liegt (Variable `PATH` für Windows), die anderen Dateien enthalten Hilfsklassen (jaja, alles sehr objektorientiert, wow!). Die `*lib.pm`-Module sind Bibliotheken für Ringstrukturen. Alle `.pm`-Dateien müssen in ein Verzeichnis im Perl-Baum kopiert werden, in dem Bibliotheksdateien liegen, typischerweise kann dies `/perl/5.005/lib` sein.

Zur Einbindung in LATEX-Dokumente ist die Paketdatei

```
ochem.sty
```

in einem von LATEX durchsuchten Verzeichnis erforderlich. Um die Größen der TEX-Texte zu ermitteln, ruft der Compiler intern LATEX auf. Je nachdem, wie auf Ihrem Rechnern das LATEX-System gestartet wird, müssen Sie den Wert von `$latexcmd` im Script `chemie.pl` ändern. Der Default ist zu folgendem Aufruf von LATEX äquivalent:

```
latex <datei>
```

Die Zeichenfolge `latex` muß zum Beispiel durch `tex386` ersetzt werden, falls Sie LATEX wie folgt starten:

```
tex386 <datei>
```

Möchten Sie die Formeln mit Mitteln von LaTeX setzen, benötigen Sie noch die Pakete `epic` und `eepic`. Sie werden im jeweiligen LaTeX-Dokument hinzugeladen.

Wollen Sie einen Präprozessor für Makroersetzung einsetzen, benötigen Sie auch diesen. Für den GNU-Präprozessor M4 existieren Makros zum Satz von Naturstoffen sowie allgemein-nützlicher Natur in den folgenden Includedateien:

```
natur.inc
utils.inc
```

Diese Include-Dateien müssen ebenfalls in ein Verzeichnis im Suchpfad kopiert werden.

1.3 Einbindung in LaTeX

Die Beschreibung der Formeln und Reaktionsschemata erfolgt in einer speziellen Beschreibungssprache in ASCII-Dateien. Diese Dateien können entweder manuell erstellt oder online aus einem LaTeX-Dokument mit eingebetteten Formeln erzeugt werden. Hier soll hauptsächlich die letzte Methode dargestellt werden, da sie die größere Bequemlichkeit bietet und die Formelbeschreibung und der korrespondierende Text nicht auf mehrere Dateien verteilt sind.

Sie müssen im LaTeX-Dokument das Paket `ochem.sty` laden und erhalten damit die neue Umgebung `chemistry`, innerhalb derer die Formelprogramme stehen:

```
% file beispiel.tex
\documentclass{...}
\usepackage{ochem}

\begin{document}

\begin{chemistry}[phenol]
  formula(C,C)
  { ring(){ 3: bond(90) atom("OH",L); }
  }
\end{chemistry}

\begin{chemistry}
  formula(C,C,"Benzen",HR,24)
  { ring(){ }  }
\end{chemistry}

\end{document}
```

Die vollständige Bearbeitung eines solchen Chemiedokumentes erfordert zwei LaTeX-Bearbeitungen. Im ersten Lauf mit

```
latex beispiel
```

werden die Inhalte aller `chemistry`-Umgebungen in eine Datei `\jobname.chm` geschrieben, und zwar in Form einzelner `schema`-Befehle (siehe unten), die die Formeln und Schemata der einzelnen Umgebungen enthalten. Die Namen der Blöcke werden durch den Parameter der `chemistry`-Umgebungen festgelegt und können bei der Fehlersuche behilflich sein. Lassen Sie diesen Parameter fort, wird eine laufende Nummer eingesetzt.

Die nach dem ersten Lauf resultierende Datei `\jobname.chm` muß nun mit dem Formelcompiler übersetzt werden

```
chemie.pl beispiel.chm
```

und liefert für jede `chemistry`-Umgebung eine Datei, die LaTeX-Code enthält. Der Name dieser Dateien wird ebenfalls aus dem ersten Parameter und der Extension `.ctx` gebildet, für das Beispiel übersetzen Sie also die Datei `beispiel.chm` und erhalten `phenol.ctx`. (Die Extension `.ctx` kann als „chemistry TeX" gedeutet werden; aufgrund der temporären Natur dieser Dateien, die jederzeit durch Neucompilierung erzeug- und überschreibbar sind, müssen sie nicht wie normale TeX-Files verwaltet werden, Befehle wie `ls *.tex` lieferten sonst eine unübersehbare Fülle von Kleinstdateien, auch das Packen aller `.tex`-Dateien in ein Archiv würde diese temporären Dateien miterfassen.) Der Compiler schreibt für jede bearbeitete `chemistry`-Umgebung deren Namen (`phenol` im Beispiel) in eckigen Klammern auf das Terminal:

```
This is OCHEM chemistry compiler version 1.0b 2001-04-10
requires OCHEM.STY 3.0e
[phenol.ctx] [1.ctx]
```

Diese Angabe ist nützlich, da Sie, falls ein Abbruch durch fehlerhaften Chemiecode erfolgt, leicht die Umgebung finden können, die den Fehler enthält.

Die nach der Compilierung erhaltenen `.ctx`-Dateien werden bei der zweiten Bearbeitung mit LaTeX

```
latex beispiel
```

automatisch eingeladen und bilden den vollständigen Formelinhalt. Da in jedem Lauf versucht wird, sie zu laden, erhalten Sie für jede Datei die Warnung, daß sie noch nicht existiere.

LaTeX-Umgebung für Direktbefehle

Bei einigen Gelegenheiten müssen Sie direkt Chemiebefehle des Abschnitts 3.2 in die `.chm`-Zwischendatei schreiben, zum Beispiel um Bibliotheken zu laden oder Makrodefinitionen für den Präprozessor einzufügen. Sie benutzen zu diesem Zwecke die `chemspecial`-Umgebung, die Ihre Befehle ohne Umsetzung in die Programmdatei schreibt:

```
\begin{chemspecial}
  <direkte Befehle>
\end{chemspecial}
```

Umgang mit dem Compiler

Zur Erzeugung der fertigen LaTeX-Fragmente mit den Formelbildern muß das beim ersten LaTeX-Lauf gebildete Programmfile, hier `beispiel.chm`, compiliert werden. Sie benutzen dazu das Perlscript `chemie.pl`:

```
chemie.pl beispiel.chm
```

Eventuell müssen Sie auf Ihrem Rechner, um das Perlscript zu starten, einen Aufruf der folgenden Form vornehmen:

```
perl chemie.pl beispiel.chm
```

Es existieren folgende Compilerswitches:

-type s : Legt das Ausgabeformat fest, `s` kann einen der Werte PS, PSLATEX oder LATEX annehmen. Default: PSLATEX.

-trace lev : Druckt eine Aufrufliste der internen Funktionen bis zu einer maximalen Schachtelungsebene von `lev` aus, mit deren Hilfe eventuell Fehler besser gefunden werden können (vor allem Programmierfehler `:-)`). Default: 0.

-pp : Weist den Compiler an, einen Präprozessorlauf mit der Eingabedatei zu starten, um Makros et cetera aufzulösen. Sie müssen `chemie.pl` für Ihren Präprozessor angepaßt haben. (Default: kein Präprozessorlauf.)

Ausgabedateien im reinen LaTeX-Format werden mit folgendem Aufruf erzeugt:

```
chemie.pl -type LATEX beispiel.chm
```

Eingabekonventionen

Bei der Eingabe der Formeln sind einige Konventionen zu beachten:

- Texte wie Formelnamen oder Bezeichner von Variablen sind mit doppelten Anführungszeichen einzuleiten und zu beenden:

```
set("rXS",24)
formula(C,C,"Formelname",HA,24)
{ ... }
```

Stehen innerhalb der Zeichenkette Anführungszeichen, so sind diese zu verdoppeln:

```
formula(C,C,"Kaurans""aure",HA,24)
{ ... }
```

Die Verdoppelung ist ähnlich wie in FORTRAN ein Hinweis für den Parser, ein literales Anführungszeichen als Textbestandteil zu lesen und es nicht als Ende der Zeichenkette zu interpretieren.

Treten Anführungszeichen zu Beginn des Namens auf, stehen also drei Zeichen " unmittelbar hintereinander, müssen Sie, um Fehlinterpretationen zu vermeiden, folgende Konstruktion eingeben:

```
formula(C,C,"{}""Ostrogen",HA,24)
{ ... }
```

Die leere Klammer {} wird später beim Textsatz mit LaTeX ignoriert, trennt aber das einfache vom doppelten Anführungszeichen.

- Maßangaben wie Längen sind entweder in einer symbolischen Form anzugeben (zum Beispiel N für eine normallange Bindung) oder als Maßzahl *ohne Einheit*, es wird stets Punkt (pt) angenommen.

Kommentare beginnen mit einem Prozentzeichen und gelten bis zum Ende der Zeile, in der dieses steht:

```
% ganze Zeile ist Kommentar

formula(L,R)  % Formel von Atropin
{ ... }
```

Prozentzeichen innerhalb von Anführungszeichen sind Bestandteil der Zeichenkette und werden nicht als Kommentareinleitung betrachtet (es können also Ausbeuteangaben in die Schemata integriert werden).

Da der M4-Präprozessor das Hashzeichen # als Kommentarmarke benutzt, treten zwei Probleme auf:

- Prozentzeichen werden von M4 nicht als Kommentarbeginn erkannt und auch in den Kommentaren unerwünschterweise Ersetzungen durchgeführt.
- Tritt in einer Formel das Zeichen # auf, wie beispielsweise in bond(#1), wäre durch den Präprozessorlauf der dem Hashzeichen folgende Zeilenrest verschwunden, was zu Fehlermeldungen des Chemiecompilers führt.

Zu Beginn eines jeden Includefiles wird daher mit der Zeile

```
changecom(`//')
```

die Folge // als Kommentarmarke für M4 gewählt. Um Zeilen auszukommentieren, schreiben Sie dann

```
formula(L,R) % normaler Kommentar
{  %// TERPEN(O)   M4-Makro auskommentiert
   ...
   Me_ bond(#1)   %// Kommentar
}
```

Der Chemiefont

Der Befehl `\chemfont` spezifiziert Kommandos zur Auswahl des Zeichensatzes, der für den Satz von Textzeichen innerhalb der Formeln zum Einsatz kommen soll. Als Standard ist `\sffamily` vorgesehen, das heißt alle Zeichen sind ebenso groß wie die Buchstaben des Grundtextes, aber serifenlos. Sie können im Vorspann des Dokumentes einen anderen Font einstellen, zum Beispiel die Serifengrundschrift in kleinerem Schriftgrad:

```
\chemfont{\rmfamily\small}
```

Oder die aktuelle Grundschrift:

```
\chemfont{}
```

Ringstrukturmodule

Wie in Abschnitt 2.9 des Tutorial geschildert werden wird, können Sie Bibliotheken mit eigenen Ringstrukturen aufbauen oder bereits vorhandene nutzen. Sie werden bei Bedarf dem LaTeX-Dokument, das sie benötigt, hinzugefügt. Jede verwendete Bibliothek muß *vor ihrer Verwendung* durch einen Befehl in der `chemspecial`-Umgebung geladen werden:

```
\begin{chemspecial}
  require("<Lib>")
\end{chemspecial}
```

Um die mitgelieferte Bibliothek `bicyclib.pm` mit bizyklischen Naturstoffringen einzusetzen, muß Ihr Dokument sinngemäß dem in Schema 1–1 gezeigten entsprechen.

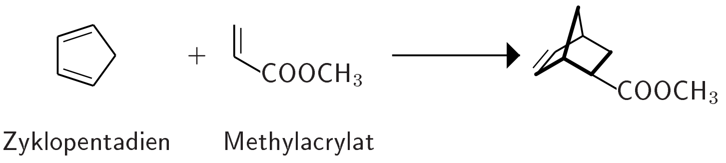
Neue Gleitumgebung

Das Paket enthält eine weitere Umgebung `schema`, die ähnlich der `figure`-Umgebung gleitende Abbildungen mit eigenem Nummernkreis für Reaktionsschemata bereitstellt. Eine einfache Anwendung ist folgende:

```
\begin{schema}
  \begin{chemistry}[phenolsyn]
     ...
  \end{chemistry}
  \caption{Eine einfache Phenolsynthese.\label{phen}}
\end{schema}
```

Im zweispaltigen Layout können Sie die Sternform (`schema*`-Umgebung) verwenden, um die Schemata über beide Spalten zu setzen.

```
\documentclass[...]{...}
\usepackage{ochem}

\begin{document}

\begin{chemspecial}
  require("bicyclib")
\end{chemspecial}

\begin{chemistry}[campher]
multiline(1,L)
{ formula(L,R,"Zyklopentadien")
  { ring(,,H1=3=,,5,0){} }

  formula(L,R){ atom("+") }

  formula(L,R,"Methylacrylat")
  { bond(-90,=U) bond(-30) atom("COOCH$_3$",L) }

  arrow(){}

  formula(L,R)
  { ring("bc221h",,5=){ 2: bond(-30) atom("COOCH$_3$",L); }
  };
}
\end{chemistry}

\end{document}
```



Schema 1–1 Verwendung von Ringstrukturen, hier des Bizyklohexans, aus einer Bibliothek.

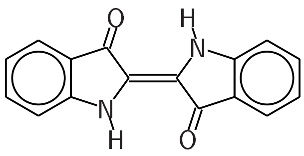
Formeln sind LaTeX-Boxen!

Wie Sie aus der Überschrift entnehmen können, bilden die `chemistry`-Umgebungen normale LaTeX-Boxen, genauer `picture`-Umgebungen oder -boxen, sodaß Sie zur Positionierung der Formeln im Text alles anwenden können, was auch mit LaTeX-Abbildungen gestattet ist, zum Beispiel die Anordnung mehrerer vertikaler schmaler Reaktionsketten nebeneinander oder das Plazieren von Formeln mitten im Textfluß oder in Abbildungen oder eigenen Zeilen.

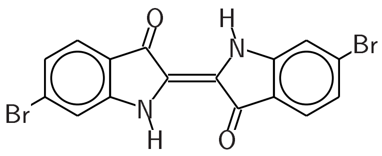
Formeln als EPS-, PNG- ... Graphiken

Die bislang dargestellte Methode, eine Formel mithilfe der `chemistry`-Umgebung in ein LaTeX-Dokument einzubinden, setzt voraus, daSSie stets die vom Formelcompiler erzeugten kleinen LaTeX-Dateien bereit haben. An sich nicht schlimm, wird hierdurch jedoch eine Benutzung dieser Formeln nur in LaTeX-Dokumenten möglich. Sie können jedoch auch durch eine Paketoption `separate` ein Dokument erstellen, das ausschließlich `chemistry`-Umgebungen enthält, die durch die Option automatisch auf eigenen Seiten, je eine Umgebung und damit Formel auf einer Seite, gesetzt werden. Da auch der Seitenstil ëmptyëingesetzt wird, erscheinen keine Seitenzahlen mehr.

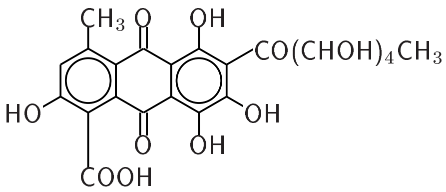
Als Beispiel diene die Datei `formeln.tex`, die die folgenden drei Formeln enthält:



Indigo



6,6'-Dibrom-indigo
Purpur



Carminsäure

```
\documentclass{report}
\usepackage{german}
\usepackage[separate]{ochem}
\begin{document}

\begin{chemistry}[indigo]
  formula(L,R,"Indigo",HR,24)
  {
    ring(,,H,,5,0){
      vertex(,2){};
      0: bond(r,=C,L) ring(,0,H,,5,r){
                        vertex(,2){};
                        1: atom("N") bond(r) atom("H");
                        4: bond(r,=C) atom("O");
                      };
      1: atom("N") bond(r) atom("H");
      4: bond(r,=C) atom("O");
```

```
    }
  }
\end{chemistry}

\begin{chemistry}[purpur]
  formula(L,R,"\shortstack{6,6'-Dibrom-indigo\\Purpur}",HR,24)
  {
    ring(,,H,,5,0){
      vertex(,2){ 3: bond(r) atom("Br"); };
      0: bond(r,=C,L) ring(,0,H,,5,r){
                        vertex(,2){ 3: bond(r) atom("Br"); };
                        1: atom("N") bond(r) atom("H");
                        4: bond(r,=C) atom("O");
                      };
      1: atom("N") bond(r) atom("H");
      4: bond(r,=C) atom("O");
    }
  }
\end{chemistry}

\begin{chemistry}[carmin]
  formula(L,R,"Carmins""aure",HR,24)
  { ring(,,H){
      vertex(,1,4){ 0: bond(r,,Ln) atom("C",C,R) atom("OOH",L);
                    1: bond(r) atom("O",C,L) atom("H",R);
                    3: bond(r) atom("C",C,R) atom("H$_3$",L);
                  };
      vertex(,4,1){ 0: bond(r) atom("O",C,R) atom("H",L);
                    3: bond(r) atom("O",C,R) atom("H",L);
                    4: bond(r) atom("CO(CHOH)$_4$CH$_3$",L);
                    5: bond(r) atom("O",C,R) atom("H",L);
                  };
      0: bond(r,=C) atom("O");
      3: bond(r,=C) atom("O");
    }
  }
\end{chemistry}

\end{document}
```

(Beachten Sie in diesem Beispiel, daß die einzelnen `chemistry`-Umgebungen jeweils Namen `indigo`, `purpur` und `carmin` erhalten haben!)

Dieses Dokument, das ausschließlich noch Formeln enthält, ist die Grundlage, um aus den einzelnen Formeln einzelne EPS-Dateien zu erstellen, die anschließend in ein beliebiges anderes Graphikformat konvertiert werden können, wie JPG oder PNG, um die Formeln in einer Webpräsentation einzusetzen.

Um zum Beispiel die Formel der Carminsäure (dritte Formel oder Formel auf der dritten Seite des Dokumentes) in das PNG-Format zu konvertieren, kann die Befehlsfolge

```
dvips -E -n 1 -p 3 -o carmin.eps formeln.dvi
convert carmin.eps carmin.png
```

eingesetzt werden. Die Option `-E` von dvips, um EPS-Dateien zu erzeugen, ist auf jedem Rechner vorhanden, das Programm convert ist Bestandteil der OpenSource-

Graphikbearbeitungsprogramme Image Magick, die vor allem unter Linux oft installiert sind oder für verschiedene Betriebssysteme leicht aus dem Internet geladen werden können.

Um eine ganze Reihe von Formeln zu konvertieren, kann das mitgelieferte Perl-Script `makePic.pl` sehr hilfreich sein:

```
makePic.pl --n=3 --format=png --outfile=formeln formeln
```

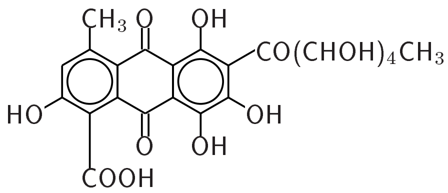
Es erstellt mit diesem Aufruf die Dateien `farben1.eps` und `farben1.png` bis `farben3.eps` und `farben3.png`. Noch einfacher wird die Konversion dadurch, daß die Namen der einzelnen `chemistry`-Umgebungen in eine spezielle Datei `formeln.names` geschrieben werden. Diese Namen können durch den einfachen Aufruf

```
makePic.pl --namefile --format=png formeln
```

als Namen der einzelnen Graphikdateien verwendet werden. Für das gegebene Beispieldokument erhalten Sie die Dateien `indigo.eps` und `indigo.png`, `purpur.eps` und `purpur.png` sowie `carmin.eps` und `carmin.png`.

Die PNG-Graphik kann zur Internet-Publikation eingesetzt, die EPS-Graphik dagegen nach Belieben in ein LaTeX-Dokument eingebunden werden:

```
\fbox{
  \includegraphics{formeln3.eps}
}
```



Carminsäure

Beachten Sie bitte, daß das Paket Image Magick auf Ihrem Rechner installiert sein muß.

1.4 Ausgabeformate

Es ist leicht möglich, anstelle der PostScript-Ausgabe andere Formate generieren zu lassen. Sämtliche Ausgaben werden aus graphischen Primitiven zusammengesetzt, die vom Modul `be.pm` in konkrete Anweisungen übersetzt werden. Diese Primitive umfassen Linien, Kreise, Text, verschiedene Bindungen sowie Reaktionspfeile. Gelingt es Ihnen, nur für diese Grundelemente eine Umsetzung im gewünschten Format zu finden, können sie künftig alle, auch komplexe Formeln und Schemata in dieses Format compilieren lassen. Als Grundlage für solche Eigenentwicklungen können die bereits vorhandenen Ausgabeformate dienen, die durch eine eindeutige Zahl unterschieden werden:

\$be::BE_PSLATEX : Texte werden in LaTeXin einer `picture`-Umgebung gesetzt, die Graphikausgabe erfolgt in PostScript. Die Definition der verschiedenen Bindungsarten und Reaktionspfeile ist in `ochem.sty` als PostScript-Headerdatei enthalten. Die Dimensionen der Textboxen werden ermittelt, indem alle auftretenden Texte in eine temporäre LaTeX-Datei geschrieben werden, die durch den Compiler übersetzt wird und für jeden Text die Größe (Breite, Höhe, Tiefe) liefert. In diesem Modus können Sie die besten Ergebnisse erwarten.

\$be::BE_PS : Dieses Format erzeugt ein reines PostScriptfile im EPS-Format. Achtung, die Größen von Text werden derzeit nicht ermittelt und stets auf Null gesetzt, was in falschen und seltsamen Verschiebungen der Texte resultiert. Dieses Format dient im Moment vor allem zum raschen Testen während der Entwicklung, ohne LaTeX benutzen zu müssen, da die reinen Formelbilder korrekt dargestellt sind. Wenn Sie keinen Text in der Formel haben, können Sie so fertige EPS-Ausgabedateien erzeugen.

\$be::BE_LATEX : Dies ist eine reine Ausgabe mit graphischen LaTeX-Primitiven wie Linien, Pfeilen und Kreisen. Entsprechend den sparsamen Möglichkeiten, werden nicht alle Bindungsformen, Winkel und Kreisradien unterstützt.

Abhängig von den Ausgabeformaten, muß auch die Ermittlung der Textgrößen auf unterschiedliche Weisen geschehen. Da dieser Vorgang unmittelbar nach dem Parsen des Programmcodes erfolgt, finden Sie die entsprechenden Zeilen nicht in `be.pm`, sondern in `streambuf.pm`. Auch in diesem Modul wird anhand des Formattyps entschieden, ob LaTeX zur Ermittlung der Textgröße aufgerufen werden muß. Für eine reine PostScript-Ausgabe existiert derzeit keine Größenermittlung.

1.5 Makrodefinitionen

Beim Compilieren des von LaTeX erzeugten `.chm`-Files kann durch den Schalter `-pp` eine Präprozessorbearbeitung veranlaßt werden, um Makros aufzulösen oder bedingte Bearbeitung zuzulassen. Hierfür müssen Sie in `chemie.pl` bei der Auswertung der Variablen `$preproc` innerhalb der Backticks den Aufruf Ihres bevorzugten Präprozessors einfügen. Der Präprozessor muß den Inhalt der Eingabedatei, deren Name in `$infile` steht, in bearbeiteter Form in die Zwischendatei `$tmpfile` übertragen. Wenn Sie den Gnu M4-Präprozessor benutzen, müssen Sie keine Anpassung vornehmen.

Möchten Sie keinen Präprozessor einsetzen, brauchen Sie nichts anzupassen und dürfen den Compiler nicht mit `-pp` aufrufen.

Der M4-Präprozessor

Hier soll keine Manpage zu M4 gegeben, sondern nur kurz dargestellt werden, wie ein Makro definiert und eingesetzt wird. Jede Makrodefinition muß vor ihrer Benutzung innerhalb einer `chemspecial`-Umgebung auftreten; sowohl Name wie auch Ersetzungstext müssen dabei durch einfaches Backtick respektive Hochkomma geklammert sein. Die aktuellen Argumente sind über die Variablen `$1`, `$2` … verfügbar. Mit der Sequenz

```
\begin{chemspecial}
  define(‘C3’, ‘bond(30) atom("$1") bond(-30)’)
\end{chemspecial}

\begin{chemistry}
  formula()
  { C3(O) C3(NH) }
    % ^-- aquivalent zu
    % bond(30) atom("O") bond(-30) ...
    % ... bond(30) atom("NH") bond(-30)
\end{chemistry}
```

wird die Struktur des Methoxymethyl-methylamins erzeugt.

Sie können auch Makros schreiben, die mit Hilfe von `ifelse` Entscheidungen treffen. Als Beispiel dient hier die Reihe einiger fünfgliedriger Heterozyklen (Thiophen, Furan und Pyrrol), in der die Chalkogenvertreter keinen Wasserstoff tragen:

```
\begin{chemspecial}
  define(‘CP’,‘ring("cpentane",,1=3=)
    {0: atom("$1");
     ifelse($1,‘N’,‘0: bond(-90) atom("H");’)
    }’)
\end{chemspecial}

\begin{chemistry}
  formula(L,R,"Furan",HA,24){ CP(O) }
  formula(L,R,"Pyrrol",HA,24){ CP(N) }
  formula(L,R,"Thiophen",HA,24){ CP(S) }
\end{chemistry}
```

Das Makro vergleicht seinen Parameter mit dem Symbol für Stickstoff und erzeugt, falls der Vergleich positiv ausfällt, eine weitere Bindung zum Wasserstoffatom.

Haben Sie eine Anzahl Makros geschrieben, die Sie immer wieder benötigen, können Sie diese in einem Includefile zusammenfassen und bei Bedarf nachladen – eine Art von Bibliotheksdatei auf der Ebene des Präprozessors. Ausführliche Beispiele von Makros finden Sie in den Abschnitten 3.6 und 3.7. Sie müssen das gewünschte Includefile mit der Folge

```
\begin{chemspecial}
include(‘natur.inc’)
\end{chemspecial}
```

in Ihr Dokument einbinden. `include` ist ein weiterer Präprozessorbefehl von M4, der die angegebene Datei lädt und die darin enthaltenen Makrodefinitionen ausführt.

Um maximalen Nutzen aus Makros zu ziehen, müssen sie möglichst flexibel geschrieben werden. Beim Terpengerüst `TERPEN` lag zum Beispiel die Überlegung zugrunde, daß die natürlichen Terpene dieses Typs meistens an den Kohlenstoffatomen C^1, C^2, C^3 und C^8 substituiert sind. Der seltener auftretende Substituent an C^2 wird als (eventuell leerer) Parameter übergeben. Die anderen Positionen werden gespeichert und können mit `restoreXY` zum Startpunkt von Substituenten werden. Mit Hilfe einiger `ifelse`-Befehle wird das `STEROID`-Makro so gestaltet, daß es die ähnlichen Gerüste wichtiger Steroidgrundkörper erzeugen kann, wobei die Beschreibung der vier Ringe gemeinsam verwendet wird.

Sie sollten beim Schreiben Ihrer Makros ähnliche Überlegungen anstellen, um zu einer Anzahl von wichtigen Schlüsselatomen zu gelangen, an denen Modifikationen möglich sind. Kleine Änderungen können als Parameter übergeben werden, sind komplizierte Formeln zu erwarten, werden diese Parameter leicht unübersichtlich und es ist besser, im Makro solche Schlüsselpositionen zu speichern und später mit `restoreXY` wieder anzuspringen. Variable Gerüstteile können mit `ifelse` eingesetzt werden und auf einer gemeinsamen Struktur aufbauen.

1.6 Ausblick

Ich habe in der vorliegenden Version die Ideen bearbeitet, die mir spontan zum Thema Formelsatz in den Sinn gekommen sind; folgen weitere, werden diese in späteren Versionen ebenfalls zur Verfügung stehen. Da sich viele Änderungen oder Erweiterungen erst beim intensiven Nutzen des Pakets herauskristallisieren, wäre ich über Anregungen oder Kritik aus Nutzerkreisen interessiert.

Da jedem automatisch der Sourcecode zugängig ist, kann unter „Nutzung" auch die *Weiterentwicklung* verstanden werden *smile*! Zunächst auf den Satzmechanismus konzentriert, habe ich hier ein weites Feld zum Bestellen hinterlassen; einige Punkte, die verbessert werden müssen, sind zum Beispiel:

- Verbesserte Ausgabe nur mit Mitteln von LaTeX (Moduskennung `$be::BE_LATEX`).
- Ermittlung der Textgröße auch bei reiner PostScript-Ausgabe mit der Moduskennung `$be::BE_PS`.
- Vielleicht kann auch das Programmdesign insgesamt und von Grund auf besser oder einfacher gestaltet werden.

Geänderter Code sollte analog dem Vorgehen bei TeX/LaTeX durch eine Namensänderung kenntlich gemacht werden, das heißt es darf unter allen Umständen nur eine einzige Version OCHEM Version 3-0e 2001-04-10 existieren! Damit nach konstruktiven Änderungen alle in den Genuß der Bemühungen und weiterentwickelter Versionen kommen können, schlage ich vor, Verbesserungen in Form des Sourcecodes oder detaillierter Bemerkungen an mich zu mailen, damit sie in die derzeit aktuelle Version eingepflegt werden können. Meine EMail-Adresse ist

`kloeckl@vkcmzd.chemie.uni-mainz.de`
`ingo.kloeckl@2k-software.de`

2. Tutorial

Dieses Kapitel soll Ihnen genügend Material an die Hand geben, um erfolgreich mit dem Formelsatz beginnen zu können. Es enthält eine Anleitung für die ersten Schritte, zeigt aber auch versteckte Aspekte einzelner Befehle und besondere Einsatzmöglichkeiten sowie Beispiele aus der Praxis. Eine ausführliche Beschreibung der Kommandos wird in der Befehlsreferenz in Kapitel 3 gegeben.

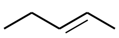
2.1 Grundlegende Elemente

2.1.1 Einfache Formeln

Bei der Beschreibung der Formeln wird ein geometrischer Ansatz verfolgt. Um eine größtmögliche Zahl an Strukturen beschreiben zu können, existiert nur eine minimale Anzahl an Grundelementen wie die Bindung in beliebiger Richtung in verschiedener Erscheinung, das Atomsymbol, Verzweigungen sowie einfache (monozyklische) Ringgerüste. Diese Elemente kombinieren Sie, indem Sie an einem Atom, zum Beispiel eines Ringes, eine Bindung in der gewünschten Richtung anfügen, die zu einem bestimmten Ringatom eines weiteren Ringes führt und so fort. Möchten Sie komplexere Ringe wie zum Beispiel Inden verwenden, liegt Ihnen kein Makro für Indexgerüste vor, sondern Sie kombinieren einen Sechs- und einen Fünfring. Der Vorteil liegt gegenüber einer Sammlung von Makros für jede Struktur klar auf der Hand: es kann eine größere Vielfalt von Strukturen dargestellt werden, obwohl im Compiler selber nur eine Handvoll implementiert werden muß. Auch können wenige Befehle rascher erlernt werden als zahlreiche unterschiedliche.

Der Nachteil: bei manchen Strukturen benötigen Sie mehr Grundelemente als bei Einsatz eines spezialisierten Makros. Da die Bindungen auch durch einen Winkel beschrieben werden, müssen Sie mehr geometrisch als chemisch denken (aber die meisten Strukturen arbeiten ohnehin mit einer immer gleichen Menge von Winkeln). Einige symbolische Winkelangaben im Zusammenhang mit Ringsystemen helfen Ihnen aber, komplizierte Winkelberechnungen zu vermeiden.

Jede Formel wird mit dem Befehl `formula` gesetzt, dessen Parameter weiter unten besprochen werden. Im Rumpf des Befehls steht die geometrische Beschreibung der Struktur. Nachfolgend einige Beispiele für sehr einfache Strukturen, die jedoch die Grundzüge des Verfahrens deutlich machen. Die Formel besteht aus einer einfachen Folge von Bindungen, woraus eine aliphatische Verbindung resultiert:



```
formula(L,R)
{ bond(30) bond(-30) bond(30,=) bond(-30) }
```

Für die Doppelbindung ist die Angabe eines optionalen Parameters erforderlich.

Einfachbindungen entsprechen dem Defaultwert und müssen nicht gesondert notiert werden. Beachten Sie die stets wiederkehrenden Winkelangaben 30°, 150° sowie -30° und -150° für schräge, aber flach verlaufende Bindungen. Diese Winkelwerte harmonieren mit den Kanten von Sechsringen, die aufrecht auf einer Ecke stehen. Bevorzugen Sie auf Kanten liegende Sechsringe, werden Sie die Winkel 60°, 120° sowie -60° und -120° einsetzen. Darüberhinaus benötigen Sie bis auf wenige Ausnahmen nur noch die Hauptachsenwinkel 0°, 90°, 180° und -90°.

Da Ketten einzelner Bindungen sehr häufig auftreten, ist hierfür eine Kurzform erlaubt, in der mehrere Bindungsbeschreibungen, durch Semikola getrennt, zusammengefaßt sind. Obige Formel läßt sich auch schreiben als

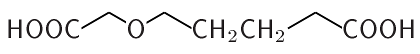
```
formula(L,R){ bond(30;-30;30,=;-30) }
```

Der Hauptnachteil der vorgestellten Beschreibung ist, daß die konkreten Bindungswinkel es erschweren, eine identische Alkylkette in anderer Orientierung, zum Beispiel gedreht, darzustellen. Sie müssten alle Winkel umrechnen. Eine größere Abstraktion wird durch symbolische Winkelangaben erreicht, die vor allem im Zusammenhang mit Ringsystemen von Bedeutung sind und weiter unten genauer vorgestellt werden. Der Beispiel-Aliphat kann damit wie folgt formuliert werden:

```
formula(L,R){ bond(r; r/; r,=; r/) }
```

`r` ist ein spezieller Winkel, der normalerweise der radialen Bindung an Ringen an der aktuellen Ringposition entspricht. Liegt kein Ring in der Beschreibung der Formel vor, wird ein Defaultwert von 30° für `r`, von 90° für `t` und von -90° für `b` eingesetzt. Die (komprimierte) Beschreibung kann damit wie folgt gelesen werden: bilde eine (radiale) Bindung; bilde abknickende Bindung; bilde Bindung in der ursprünglichen (radialen) Richtung als Doppelbindung; bilde wieder eine abknickende Bindung. Eine solche Beschreibung ist unabhängig von konkreten Winkeln und kann somit an verschiedenen Stellen der Formel eingesetzt werden. `ring` setzt selbständig die Winkel auf sinnvolle Werte, bei reinen Alkylketten müssen Sie in seltenen Fällen mit `set` die Winkel manuell setzen (Beispiele in der Referenz zu `bond`). Mit konkreten Winkeln können Sie jederzeit, auch bei intensiver Verwendung der symbolischen Angaben, die gewünschte Richtung erzwingen.

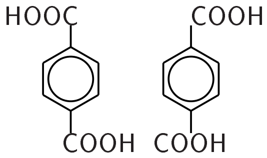
Die zweite und alle weiteren Formeln zeigen, wie Atomsymbole in das Gerüst aus Bindungen integriert werden. Beachten Sie die Positionierungen der einzelnen Atomgruppen: einzelne Atome an terminalen Positionen werden mit der Voreinstellung `C,C` von `atom` zumeist zentriert zum Bindungsende gesetzt, während längere Texte bündig mit einer seitlichen Kante anschließen (ein solcher Kantenabschluß wäre aus Gründen der Einheitlichkeit auch für einzelne Buchstaben wie „O" anzuraten, ist mir aber meistens zu umständlich :-( ).




```
formula(L,R)
{ atom("HOOC",L,R) bond(30;-30) atom("O")
  bond(30;-30) atom("CH$_2$CH$_2$",L,R)
  bond(30;-30) atom("COOH",L,R)
}
```

Ein besonderes Problem tritt auf, wenn Bindungen von 90° oder -90° in langen Texten enden, zum Beispiel in der Carboxyl- oder Hydroxylgruppe. Um diese lange Gruppe nicht seitlich, sondern (schöner) zentriert zum C anzusetzen, wird der Text in einen zentrierbaren (C) und einen seitlich ansetzbaren (OOH) Teil aufgespalten

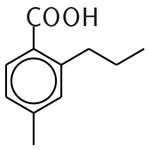
(Formel links). In der Formel rechts können Sie erkennen, welches Aussehen die COOH-Gruppe besäße, wäre sie einfach in einem Stück mit L positioniert worden (Bindung oben). Eine Zentrierung des Textes hätte katastrophale Folgen (COOH-Gruppe unten).




```
formula(L,R)
{ ring()
  { 0: bond(r) atom("C",C,R) atom("OOH",L);
    3: bond(r) atom("C",C,L) atom("HOO",R);
  }
}
formula(L,R)
{ ring()
  { 0: bond(r) atom("COOH",C);
    3: bond(r) atom("COOH",L); }
}
```

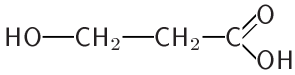
Das Problem tritt nicht in Erscheinung, wenn die Bindungen schräg auf einen langen Textabschnitt treffen und seitlich zum gesamten Text positioniert werden, wie Sie bereits an der Etherkette mit den zwei Carboxylgruppen gesehen haben.

Ringstrukturen werden mit dem Befehl `ring` erzeugt. In seiner einfachsten Form erhalten Sie einen Benzenring, mit verschiedenen Parametern können Sie jedoch auch n-gliedrige Ringe sowie andere wichtige Gerüste (Campher) darstellen. Mit Ringen begegnen Ihnen erstmal Elemente, von denen mehrere Substituenten abzweigen können. Diese Substituenten werden wie normale Formeln beschrieben, müssen jedoch jeweils durch ein `;` angeschlossen werden. Das Ringatom, von dem aus der Substituent abzweigt, wird durch eine Nummer, bei Null beginnend, bezeichnet. Um nicht für jede Ringposition den genauen Winkel, unter dem die Bindungen abzweigen, lernen zu müssen, können Sie bei `bond` eine symbolische Winkelangabe `r` für radiale Bindungen angeben (eine ausführlichere Diskussion symbolischer Winkel finden Sie im nächsten Abschnitt):



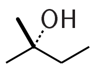

```
formula(L,R)
{ ring()
  { 0: bond(r);
    3: bond(r) atom("C",C,R) atom("OOH",L);
    4: bond(r; r/; r); }
}
```

Eine weitere Form der Verzweigung kann bei Ketten auftreten. `branch` hält die aktuelle Position in anonymer Form fest und erlaubt es, Substituenten anzugeben, die mit `;` abgeschlossen werden und an diesem Punkt beginnen:



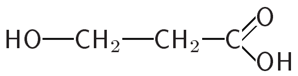

```
formula(L,R)
{ atom("HO",L,R) bond(0)
  atom("CH$_2$",L,R) bond(0)
  atom("CH$_2$",L,R) bond(0)
  branch { bond(-45) atom("OH", L);
           bond(45,=) atom("O",L);
           atom("C");
         }
}
```

Da dieser Punkt nach Abarbeitung des Befehlsrumpfes noch erhalten bleibt, kann von ihm aus die Kette weitergeführt werden. Sie können daher eine Hauptkette formulieren und die von dieser abzweigenden Seitenäste mit `branch` beschreiben:



```
formula(L,R)
{ bond(30)
  branch { bond(120,t);
           bond(60,o) atom("O",C,R) atom("H",L);
         }
  bond(-30; 30)
}
```

Die Befehle `saveXY` und `restoreXY` bieten die Möglichkeit, die aktuelle Position unter einer Nummer zu speichern und an beliebiger Stelle darauf zuzugreifen. Die rekursive Schachtelung, die bei Verzweigungen mit `branch` auftritt, kann so in einen linearen Ablauf umgewandelt werden. Besonders bei tief geschachtelten Verzweigungen kann diese Form vorteilhaft sein.



```
formula(L,R)
{ atom("HO",L,R) bond(0)
  atom("CH$_2$",L,R) bond(0)
  atom("CH$_2$",L,R) bond(0)
  saveXY(#1)     atom("C")
  restoreXY(#1)  bond(-45) atom("OH", L)
  restoreXY(#1)  bond(45,=) atom("O",L)
}
```

2.1.2 Formeln anordnen

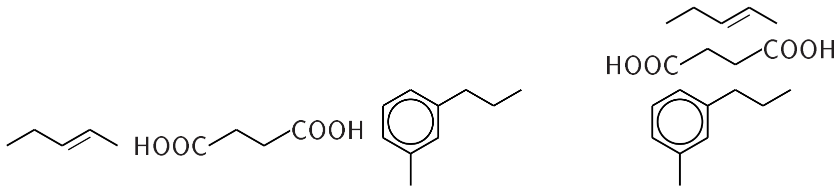
Haben Sie erst einmal eine Formelbeschreibung erstellt, kann diese innerhalb des `formula`-Befehls als Einheit betrachtet und automatisch plaziert werden. An dieser Stelle kommen die Parameter des Befehls ins Spiel: sie geben an, wie die Formel relativ zu einem Startpunkt plaziert werden soll (zentriert, linksbündig, mit einer Ecke oder Kante) und an welcher Stelle der Formel der Anschlußpunkt für nachfolgende Formeln liegt (ebenfalls zentriert, eine Kante oder Ecke). Dieser Anschlußpunkt wird zum Startpunkt für folgende Formeln. Daraus folgt, daß Sie für beide Parameter beliebige Werte, zum Beispiel `C,C`, einsetzen können, wenn es Ihnen nur darum geht, ein Formelbild in den Text einzufügen, da beide Parameter erst im Zusammenspiel mehrerer Formeln ihren Sinn erhalten.

Betrachten Sie als Beispiel das Schema 2–1, in dem in der linken Hälfte Formeln Seite an Seite erscheinen, auf eine gemeinsame vertikale Mitte ausgerichtet mit dem Positionierungsparameter `L,R`. Diesselben Formeln mit `T,B` gesetzt, reihen sich rechts an einer gemeinsamen Mittelkante, aber vertikal auf.

Die beiden Varianten, Formeln aneinanderzufügen, dürften die meistgenutzten sein, da mit ihnen Reaktionsketten aufgebaut werden können. (In einem späteren Abschnitt werden Sie noch lernen, wie neben dem bislang linearen Ablauf Verzweigungen hinzufügen.) Es sind jedoch zwei wichtige Sonderanordnungen zu erwähnen:

- Eine regelmäßige tabellarische Anordnung kann mit den bisherigen Mitteln nicht erreicht werden, da die Position jeder folgenden Formel von der Größe der vorhergehenden abhängig ist. Der Befehl →`gotoXY` setzt den Startpunkt einer Formel

```
\begin{chemistry}[intro2a]
  formula(L,R){ bond(30; -30; 30,=; -30) }

  formula(L,R)
  { atom("HOOC",L,R) bond(30; -30; 30) atom("COOH", L) }

  formula(L,R)
  { ring()
    { 0: bond(r);
      4: bond(r; r/; r); }
  }
\end{chemistry}
\hfil
\begin{chemistry}[intro2b]
  formula(T,B){ bond(30; -30; 30,=; -30) }

  formula(T,B)
  { atom("HOOC",L,R) bond(30; -30; 30) atom("COOH", L) }

  formula(T,B)
  { ring()
    { 0: bond(r);
      4: bond(r; r/; r); }
  }
\end{chemistry}
```



Schema 2–1 Die Positionierungsparameter vermitteln den Zusammenhang zwischen den einzelnen Formeln. In der linken Hälfte von links nach rechts angeordnete Formeln, in der rechten Hälfte übereinandergestellte Formeln (von oben nach unten).

unabhängig vom Kontext auf einen bestimmten Punkt und bietet somit die Grundlage einer regelmäßigen Anordnung. Ein Beispiel finden Sie in der Befehlsreferenz zu `gotoXY`.

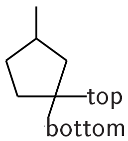
- Eine freie Anordnung von Formeln kann sinnvoll sein, wenn zwei oder mehr Reaktanden zu einem Produkt führen und die Reaktanden nicht hintereinander, durch Pluszeichen verbunden, Schlange vor ihrem Reaktionspfeil stehen sollen, sondern „wolkenartig“ verteilt sein sollen. Dies ist mit dem Befehl →`shiftXY` realisierbar, der den aktuellen Punkt um einen bestimmten Betrag verschiebt. Als Bezugspunkt wird dann in der Regel der Startpunkt der ersten Formel gewählt, alle anderen beziehen sich dann auf diese. Ein Beispiel ist im Tutorial in Abschnitt 2.6 gegeben.

2.2 Komplexere Formeln

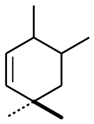
Dieser Abschnitt stellt komplexere Formeln sowie die zu ihrer Beschreibung nötigen Mittel vor. Einige der verwendeten Tricks werden in der alphabetischen Referenz erwähnt, andere nicht, in jedem Falle können Sie in diesem Abschnitt eine ausführlichere Diskussion wichtiger Darstellungsmittel finden, aus welchem Grunde dieser Abschnitt die Lektüre der Referenz nicht ersetzen mag. Haben Sie eine Formel zu setzen, zu deren Beschreibung Ihnen anscheinend die Mittel fehlen, schauen Sie über die hier gezeigten Strukturen, vielleicht erhalten Sie die richtigen Anregungen. Manchmal geht’s auch einfach mit Pfusch und wilden Linien:-)

2.2.1 Symbolische Winkel

Der `ring`-Befehl liefert über die symbolische Angabe `r` hinaus für jedes Ringatom drei Kardinalwinkel, die für radiale Bindungen und Bindungen parallel zu den beiden angrenzenden Kanten verwendet werden. Diese werden von `bond` ausgewertet, wenn Sie einen der „Winkel“ `r`, `t` oder `b` angeben. Sie können so Bindungen erzeugen, ohne sich um den konkreten Winkel, unter dem sie vom Ring abzweigen, Gedanken zu machen. Die nächste Verbindung zeigt die Verwendung aller drei Symbole:



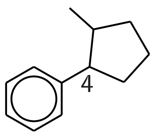

```
formula(L,R)
{ ring(,,H,,5,90)
  { 0: bond(r);
    2: bond(t) atom("top",L);
    2: bond(b) atom("bottom",L); }
}
```



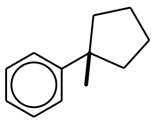
```
formula(L,R)
{ ring(,,H1=)
  { 0: bond(t,t); 0: bond(b,o);
    3: bond(r);
    4: bond(r); }
}
```

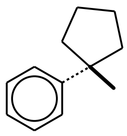
Bei den Ringgerüsten mit räumlicher Intention (`chair`, `bc222o` und so weiter) wird die Bedeutung der `t`- und `b`-Winkel als „top“ und „bottom“ deutlich, Beispiel finden Sie bei der Vorstellung der Bibliotheksmodule und des →`ring`-Befehls.

Mündet eine Bindung mit symbolischem Winkel in einem weiteren Ring, kennen Sie den Winkel nicht, um den der zweite Ring gedreht werden muß, damit die Bindung auch bezüglich dieses zweiten Ringes in einer der drei Relationen `r`, `t` oder `b` steht. Dies ist auch nicht erforderlich, da anstelle des Drehwinkels des zweiten Ringes ebenfalls eine dieser symbolischen Angaben stehen kann. Sie müssen nur die Nummer des Ringatoms angeben, an dem die Bindung endet:



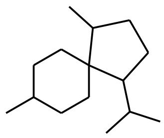
```
formula(L,R)
{ ring(){4: bond(r)
            ring(,4,H,,5,r){ 0: bond(r); };
        }
}
```





```
formula(L,R)
{ ring(){ 4: bond(r)
             ring(,0,H,,5,r){ 0: bond(t,t); }; }
}
formula(L,R)
{ ring(){ 4: bond(r,o)
             ring(,0,H,,5,b){ 0: bond(t,t); }; }
}
```

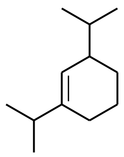
Das Acoran zeigt, daß auch Spiroverbindungen gezeichnet werden können, bei denen die beiden Ringe nicht durch eine Bindung verbunden sind. Der symbolische Winkel `r` ist ein für jede Ringposition spezifischer Default für Drehwinkel:



Acoran

```
formula(L,R,"Acoran",HR,24)
{ ring(,,H)
  {1: bond(r);
   4: ring(,0,H,,5,r)
      { 1: bond(r);
        4: bond(r) branch { bond(r+);
                            bond(r-); };
      };
  }
}
```

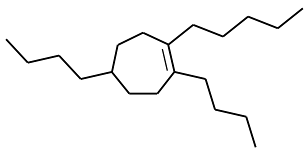
Ein nachgestelltes + oder − vergrößert beziehungsweise verkleinern den Winkel um 60°. Aus der Sicht der zu zeichnenden Bindung entspricht dies einer Drehung um 60° nach links beziehungsweise rechts. Sie können so ohne Kenntnis des genauen Winkels verzweigte Alkyl-, zum Beispiel Isopropylsubstituenten beschreiben:



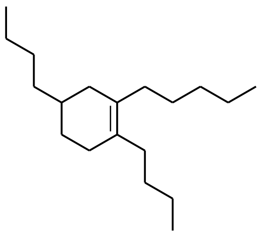
```
formula(L,R)
{ ring(,,H1=)
  { 3: bond(r) branch { bond(r+);  bond(r-); };
    1: bond(r) branch { bond(r+);  bond(r-); }; }
}
```

Weist eine der Ketten der Verzweigung mehr als eine Methyleneinheit auf, ist es geschickter, die Verzweigung mit `rt` oder `r/t` zu formulieren, siehe unten.

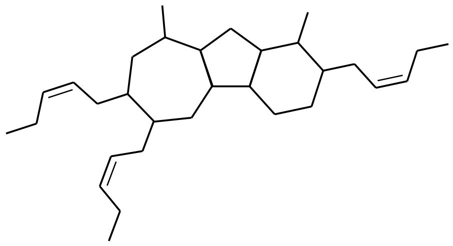
Mit den `+`- und `-`-Angaben können auch längere Alkylketten ohne Wissen um den genauen Winkel formuliert werden. Da die Drehungen relativ zum augenblicklichen Drehwinkel erfolgen, können Sie alle identischen Alkylsubstituenten, an welcher Ringpositionen sie auch auftreten, durch Angabe ein und derselben Beschreibung setzen. Beachten Sie, daß bei diesem Verfahren alle Ketten durch Drehung aus derselben Grundform hervorgehen und die Ketten so in Abhängigkeit ihres Startatoms unterschiedlich orientiert sind:



```
formula(T,B)
{ ring(,,H1=,,7,90)
  { 1: bond(r; r-; r; r-; r);
    2: bond(r; r-; r; r-) ;
    5: bond(r; r-; r; r-) ;  }
}
```

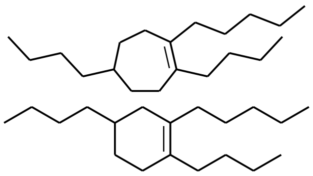


```
formula(T,B)
{ ring(,,H1=,,6,90)
  { 1: bond(r; r-; r; r-; r);
    2: bond(r; r-; r; r-) ;
    5: bond(r; r-; r; r-) ; }
}
```



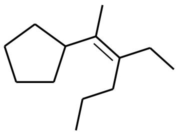
```
formula(L,R)
{ ring(,,H,,5,90)
  { vertex(,1,4,H,6)
    { 0: bond(r);
      1: bond(r; r-; r,=U; r+; r);
    };
    vertex(,3,1,H,7)
    { 0: bond(r);
      4: bond(r; r-; r,=U; r+; r);
      5: bond(r; r-; r,=U; r+; r);
    };
  }
}
```

Einige dieser Ketten werden üblicherweise horizontal orientiert dargestellt. Wünschen Sie eine solche Anordnung, können Sie den symbolischen Winkel `r/` anstelle der festen Rechtsdrehung `r-` benutzen. Diese Angabe legt keine bestimmte Drehrichtung fest, sondern bestimmt diese anhand der augenblicklichen Bindungsrichtung: weist die Bindung nach oben rechts (zum Beispiel 30°), wird die nächste Bindung nach unten rechts (-30°) gezogen. Auf Bindungen nach unten links folgen solche nach links oben und so fort. Diese Anordnung fällt am angenehmsten bei sechsgliedrigen Ringen auf:



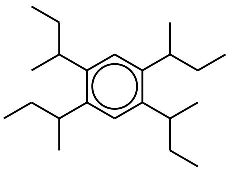
```
formula(T,B)
{ ring(,,H1=,,7,90)
  { 1: bond(r; r/; r; r/; r) ;
    2: bond(r; r/; r; r/) ;
    5: bond(r; r/; r; r/) ; }
}
formula(T,B)
{ ring(,,H1=,,6,90)
  { 1: bond(r; r/; r; r/; r) ;
    2: bond(r; r/; r; r/) ;
    5: bond(r; r/; r; r/) ; }
}
```

Wiederholte Anwendung der `+`- und `-`-Symbole erhöht oder vermindert jeweils um 60°, sodaß auch geschachtelte Alkylketten ohne Kenntnis der genauen Winkel der aktuellen Ringposition gesetzt werden können:



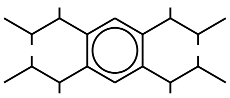
```
formula(L,R)
{ ring(,,H,,5,90)
  { 1: bond(r)
       branch { bond(r+); }
       bond(r-,=)
       branch { bond(r--; r---; r--); }
       bond(r) bond(r-); }
}
```

Das folgende Beispiel diskutiert eine Möglichkeit der Kettenverzweigungen genauer. Durch die Festlegung einer Linksdrehung für die Methylverzweigung (`r+`) resultieren auf der „linken“ Seite des Ringes andere Ketten als auf der rechten, was die Orientierung der Methylgruppen angeht. Die Ketten entstehen durch strenge Rotation:



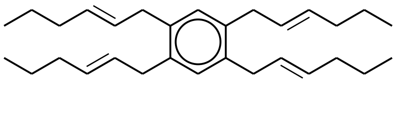
```
formula(L,R)
{ ring()
  { 1: bond(r) branch { bond(r+); } bond(r-;r);
    2: bond(r) branch { bond(r+); } bond(r-;r);
    4: bond(r) branch { bond(r+); } bond(r-;r);
    5: bond(r) branch { bond(r+); } bond(r-;r); }
}
```

Die symbolischen Winkel `rt` und `r/t` versuchen, eine gefälligere, das heißt symmetrischere und an den Koordinatenachsen orientierte Anordnung solcher Querbindungen zu erzielen, indem Links- oder Rechtsdrehungen zugelassen und auch automatisch bestimmt werden:

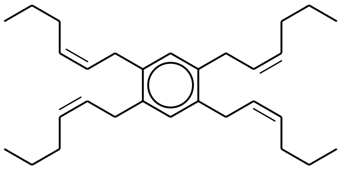


```
formula(L,R)
{ ring()
  { 1: bond(r) branch { bond(rt,,s); }
       bond(r/) branch { bond(r/t,,s); } bond(r);
    2: bond(r) branch { bond(rt,,s); }
       bond(r/) branch { bond(r/t,,s); } bond(r);
    4: bond(r) branch { bond(rt,,s); }
       bond(r/) branch { bond(r/t,,s); } bond(r);
    5: bond(r) branch { bond(rt,,s); }
       bond(r/) branch { bond(r/t,,s); } bond(r); }
}
```

Zur Darstellung der Stereochemie an Doppelbindungen muß zu den bislang genannten noch der symbolische Winkel `r\` hinzutreten, mit dem eine Bindung *entgegen* dem Knickwinkel `r/` gezeichnet wird. Die Auswirkungen sowie die korrekte Formulierung von winkelunabhängigen E/Z-Isomeren ist in der folgenden Formel gezeigt:



(E)-Isomer



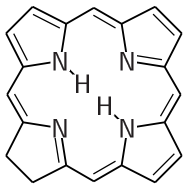
(Z)-Isomer

```
formula(T,B,"(E)-Isomer",HR,24)
{ ring()
  { 1: bond(r; r/; r,=; r/; r; r/);
    2: bond(r; r/; r,=; r/; r; r/);
    4: bond(r; r/; r,=; r/; r; r/);
    5: bond(r; r/; r,=; r/; r; r/); }
}
formula(T,B,"(Z)-Isomer",HR,24)
{ ring()
  { 1: bond(r; r/; r,=; r\; r; r/);
    2: bond(r; r/; r,=; r\; r; r/);
    4: bond(r; r/; r,=; r\; r; r/);
    5: bond(r; r/; r,=; r\; r; r/); }
}
```

2.2.2 Lineare Systeme

Viele Formelbilder lassen sich als lineare Abfolge von einfachen Bindungen oder zumindest identischen oder ähnlichen Einheiten auffassen und daher linear beschreiben.

Der durchaus komplizierte Chlorinring kann trotz seines Ringcharakter als lineares System aufgefaßt werden, das aus vier Pyrrol-/Methyleneinheiten besteht. Jede diese Methylenbrücken endet am Pyrrolring der nächsten Einheit. Der Ring kann somit durch eingeschachtelte nahezu identische Beschreibungen der Grundeinheit beschrieben werden. Der Ringschluß ergibt sich durch korrekte Wahl der Winkel aus Symmetriegründen von selber. (Da die Anschlußpunkte einer Struktureinheit stets von der vorhergehenden abhängig sind, erlangt man hier keinen Gewinn durch den Einsatz einer Linearisierung mit `saveXY` und `restoreXY`. Dieser wäre nur gegeben, wenn die Startpunkte der Partialformeln allesamt zu Beginn verfügbar wären.)

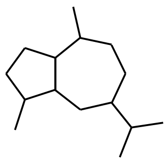


Chlorin

```
formula(L,R,"Chlorin",HR,24)
{ ring(,,H1=3=,,5,-45)
  { 0: atom("N") bond(-45) atom("H");
    4: bond(30) bond(-30,=)
       ring(,1,H2=4=,,5,-135)
       {  0: atom("N");
          4: bond(-60) bond(-120,=)
             ring(,1,H2=,,5,135)
             { 0: atom("N") bond(135) atom("H");
               4: bond(-150,=) bond(150)
                  ring(,1,H0=,,5,45)
                  { 0: atom("N");
                    4: bond(120,=) bond(60);
                  };
             };
       };
  }
}
```

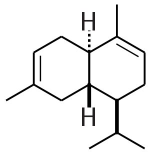
2.2.3 Linear anellierte Ringe

Der einfachste Fall anellierter Ringsysteme liegt dann vor, wenn alle Ringe ein gemeinsames Atom aufweisen. Sie können sich zunutze machen, daß `ring` den aktuellen Startpunkt nicht verschiebt und so, wenn der aktuelle Punkt genau diesem gemeinsamen Atom entspricht, alle Ringe in einem Rutsch dargestellt werden. Alle Ringe liegen dann in derselben Schachtelungsebene, was die Formel möglicherweise übersichtlicher gestaltet als die Angabe von Ringen als Substituenten weiterer Ringe. Damit beide Ringe die gleiche Kantenlänge haben und somit nahtlos zusammenpassen, muß ihre Länge mit `#N` (auf Wunsch auf `#L` oder andere Längen, wesentlich ist das `#`-Zeichen) gleichgesetzt:



Guajan

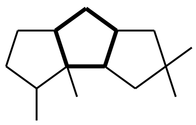
```
formula(L,R,"Guajan",HR,24)
{ ring(,4,H,#N,7,0){ 1: bond(r) branch { bond(r+); bond(r-); };
                     5: bond(r);}
  ring(,2,H,#N,5,180){4: bond(r); }
}
```



β-Cadinen

```
formula(L,R,"$\beta$-Cadinen",HR,24)
{ ring(,5,H1=){ 1: bond(r); }
  ring(,1,H3=){ 0: bond(r,t) branch{ bond(r+);
                                     bond(r-);};
               1: bond(t,t) atom("H");
               2: bond(b,o) atom("H");
               3: bond(r);
              }
}
```

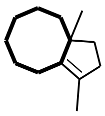
Haben nicht alle Ringe ein gemeinsames Atom, müssen Sie einen Basisring auswählen, der die weiteren Ringe als Substituenten trägt. Um die Übersicht zu bewahren, kann es sinnvoll sein, den am kompliziertesten substituierten Ring als Basis zu wählen, um einfachere Substitution an den Ringen zu erhalten, die selbst Substituent dargestellen. Im Falle des Hirsutans wird der zentrale Fünfring als Basis gewählt, da auf diese Weise zwei einfache Ringe als Substituenten auftreten. Wäre einer der endständigen Ringe gewählt worden, wäre der zentrale Fünfring dessen Substituent, der wiederum einen Ring als Substituent trüge, was die Formel unnötig komplizieren würde. In diesen und den folgenden Formeln ist der Basisring durch größere Strichstärke hervorgehoben:



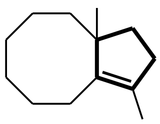
Hirsutan

```
formula(L,R,"Hirsutan",HR,24)
{ ring(,,H,,5,90)
  { vertex(,1,2,H,5){ 0: bond(t); 0: bond(b); };
    vertex(,3,3,H,5){ 0: bond(r); };
    3: bond(t); }
}
```

Die `vertex`-Syntax übernimmt die ansonsten komplizierten Berechnungen der korrekten Startpunkte und Drehwinkel aller anelierter Ringe. Ein weiteres Beispiel zeigt, wie einfach zusammengesetzte Ringsysteme beschreibbar sind. Die beiden gezeigten Varianten demonstrieren, daß die Wahl des Basisringes auch die Gesamtdrehung der Formel beeinflußt: im ersten Falle ist das nicht gedrehte Zyklooktan die Basis, der Fünfring erscheint an der schräg verlaufenden [a]-Kante und verleiht dem Gesamtmolekül so einen gedrehten Eindruck. Wählen Sie, wie im zweiten Falle, dagegen den Fünfring als Basis, entsteht der Eindruck einer horizontal liegenden Formel. Beachten Sie die variable Kantenlänge der Ringe. Sie können jederzeit mit # eine bestimmte Kantenlänge für einen Ring festlegen.

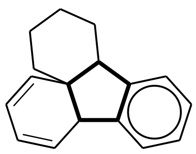


```
formula(L,R)
{ ring(,,H,,8,0)
  { vertex(,0,1,H0=,5){ 0: bond(r); };
    0: bond(t);
  }
}
```

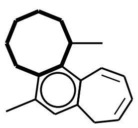


```
formula(L,R)
{ ring(,,H1=,,5,0)
  { vertex(,2,0,H,8){ 0: bond(t); };
    1: bond(r);
  }
}
```

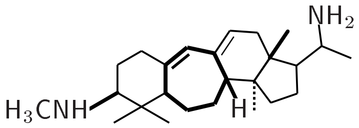
Mehrere Substituenten bauen komplizierte Systeme auf, wobei die ankondensierten Ringe ihrerseits weitere Ringe tragen können, sodaß Systeme entstehen, in denen nicht alle Ringe direkt an einer Kante des Basisringes liegen. Beides wird durch die folgenden Beispiele illustriert:



```
formula(L,R)
{ ring(,,H,,5,90)
  { vertex(,1,0,,6){ };
    vertex(,3,0,H2=4=,6){ };
    vertex(,4,0,H,6){ };
  }
}
```



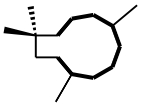
```
formula(L,R)
{ ring(,,H,,8,0)
  { vertex(,1,1,,6)
    { 0: bond(r);
      vertex(,3,0,H2=4=,7){};
    };
    0: bond(r);
  }
}
```



Buxenin-G

```
formula(L,R,"Buxenin-G",HR,24)
{ ring(,,H4=,,7,0)
  { 0: bond(b,t) atom("H");
    vertex(,3,1,H,6){ 3: bond(t); 3: bond(b);
                      4: bond(r,t) atom("H$_3$CNH",R); };
    vertex(,6,4,H5=,6){ 2: bond(t,t);
                        3: bond(-90,o);
                        vertex(,2,2,H,5)
                        { 4: bond(r)
                             branch { bond(r+) atom("N",C,R)
                                               atom("H$_2$",L); }
                             bond(r-);
                        };
                      };
  }
}
```

Nicht immer muß die beiden Ringen gemeinsame Bindung sichtbar sein. Das Humulen besteht aus einem Neunring, kondensiert mit einem Vierring. Der Eindruck der quadratischen Aussparung wird einfach durch je eine unsichtbare Bindung in jedem der Ringe erreicht:



Humulen

```
formula(L,R,"Humulen",HR,24)
{ ring(,,H4s,,9,0)
  { 3: bond(r);
    vertex(,4,3,H3s,4,-45)
       { 2: bond(100,<.); 2: bond(170,<<); };
    8: bond(45);
  }
}
```

2.2.4 Manuelle Erstellung anellierter Systeme

Sie haben im letzten Abschnitt gesehen, wie mit `vertex` mehrkernige Systeme aus streng polygonalen Ringen aufbaut werden können. Diese Syntaxerweiterung ist jedoch nur für den Typ `ring`, also n-Ecke, definiert und kann daher nicht mit anderen Typen wie `cpentane` verwendet werden. Müssen Ringe aus solchen anderen Typen aufgebaut werden, müssen Sie die Berechnungen der korrekten Drehwinkel selber vornehmen. Es folgt eine kleine Anleitung, in der eine allgemeine Formel für den Drehwinkel entwickelt wird, die auf Polygonen basiert, aber auf andere Typen übertragbar ist, wenn Sie deren typische Winkel kennen. Die Basispolygone sind dabei *nicht* gedreht, <p2> besitzt den Wert Null.

Den einzelnen Kanten sind gemäß Tabelle 2.1 Winkel θ_i zugeordnet, die in die allgemeine Formel einzusetzen sind. In der folgenden allgemeinen Formel bedeuten ϕ_{soll} den gewünschten Drehwinkel des Grundkörpers, ϕ den korrekten Drehwinkel des Ringes, der als Substituent dargestellt wird, θ_1 und θ_2 die Kantenwinkel der zusammentreffenden Kanten von Basis- und Substituentenring:

$$\phi = \phi_{soll} + (\theta_1 - \theta_2) + 180$$

Die folgende Verbindung basiert auf dem fünfgliedrigen Pyrrolidinring und wird zur Verdeutlichung mit den Drehwinkeln 0°, 45° und 90° dargestellt. Die Nullpunk-

Tabelle 2.1 Zuordnung des Kantenwinkels zu den einzelnen Kanten für verschiedene Polygone (Zyklopentan, $n = 5$ bis Zyklooktan, $n = 8$). Die Kantennummer ist gleichzeitig die Nummer eines Substituenten.

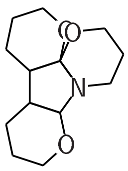
Kante	Kantennummer	$n = 5$	$n = 6$	$n = 7$	$n = 8$
a	0	54	60	64	67
b	1	-18	0	13	23
c	2	-90	-60	-38	-22
d	3	-162	-120	-90	-67
e	4	-234	-180	-141	-112
f	5		-240	-193	-157
g	6			-244	-202
h	7				-247

te der einzelnen Furanringe werden vom Sauerstoffatom eingenommen, der Nullpunkt des Pyrrolidins vom Stickstoff.

Konstruieren wir zunächst den Grundkörper ohne Drehung des Pyrrolidinkerns. Durch folgende Kantenkorrespondenzen ergeben sich die angegebenen Winkel:

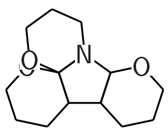
Pyrrolidin		Furan		Winkel
Kante	Substituent i	Kante	Startatom j	
b	1	e	$4 + 1 = 5$	$0 + -18 - -180 + 180 = -18$
d	3	b	$1 + 1 = 2$	$0 + -162 - 0 + 180 = 18$
e	4	e	$4 + 1 = 5$	$0 + -234 - -180 + 180 = -234$

Durch die Wahl der Ringkanten, die übereinstimmen sollen, ist die Lage des Nullpunktes des gedrehten Ringes festgelegt und ebenfalls in der Tabelle angegeben. Es gilt dabei, daß die Substituentenposition i des Basisringes die Nummer der Kante ist (Tabelle 2.1). Das Startatom j, das in der Beschreibungsliste des Substituentenringes eingesetzt werden muß, ist um Eins höher als die Kantennummer.



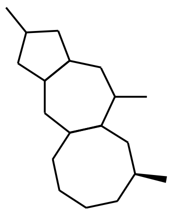

```
formula(L,R)
{ ring(,,H,#N,5,0)
  { 1: ring(,5,H,#N,6,-18){ 0: atom("O");};
    3: ring(,2,H,#N,6,18){ 0: atom("O");};
    4: ring(,5,H,#N,6,-234){ 0: atom("O");};
    0: atom("N"); }
}
```

Bei den gedrehten Systemen muß nun einfach der Drehwinkel ϕ_{soll} zu allen Winkeln addiert werden.



```
formula(L,R)
{ ring(,,H,#N,5,90)
  { 1: ring(,5,H,#N,6,72){ 0: atom("O");};
    3: ring(,2,H,#N,6,108){ 0: atom("O");};
    4: ring(,5,H,#N,6,-144){ 0: atom("O");};
    0: atom("N"); }
}
```

Und schließlich noch ein System aus Fünf-, Sieben- und Achtringen:

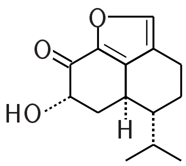


```
formula(L,R)
{ ring(,,H,#N,7,0)
  { 4: ring(,3,H,#N,5,129){ 0: bond(r);};   % 0 + -141 - -90 + 180
    1: ring(,6,H,#N,8,350){ 0: bond(r,<<);}; % 0 + 13 - -157 + 180
    0: bond(r); }
}
```

Die typischen Winkel beim Ringtyp `cpentane` entsprechen denen des Sechsecks, wobei Sie die fortfallende Ecke berücksichtigen müssen.

2.2.5 Angular anellierte Ringe

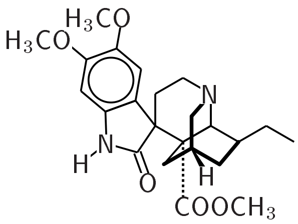
Ringsysteme, in denen die Ringe mehr als eine gemeinsame Bindung haben, können mitunter schwierig zu beschreiben sein, wenn Sie sie aus Polygonen aufbauen, da die Bindungen zum Beispiel eines Fünfringen andere Winkel aufweisen als die der Sechs- oder Siebenringe. Bei kondensierten Sechsringen tritt dieses Problem nicht auf; für den häufigen Fall des Zusammentreffens von Sechs- und Fünfringen kann zur Darstellung der Fünfringe der Typ `"cpentane"` gewählt werden, dessen Winkel nahtlos zu denen der Sechsringe passen.



(+)-Hibiscon A

```
formula(L,R,"($+$)-Hibiscon A",HR,24)
{ ring(,4,H){ 1: bond(r,o) atom("HO",R);
              2: bond(r,=C) atom("O");
              5: bond(b,o) atom("H"); }
  ring(,2,H){ 0: bond(r,o) branch{ bond(r+); bond(r-);}; }
  ring("cpentane",,0=3=){ 2: atom("O");}
}
```

Die Formel des Crassanins zeigt, daß auch das Gerüst des Bizyklo[2.2.2]oktans mit einem Sechsring kombinieren werden kann, da die Bindungslängen im Bizyklus denen der sonstigen Bindungen entspricht. Die Kantenlängen des Fünfrings muß mit der Größenangabe `#N` etwas verringert werden, damit der Benzenring nicht erheblich größer als der Zyklohexanring wirkt:



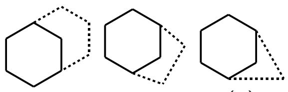
Crassanin

```
formula(L,R,"Crassanin", HR,24)
{ ring("bc222o",,,N)
  { 2: bond(30) bond(-30);
    0: ring(,,H,N)
       { 1: ring(,4,H,#N,5,-70)
            { 0: bond(-70,=C) atom("O");
              1: atom("N") bond(r) atom("H");
              vertex(,2,1){ 4: bond(r) atom("O",C,L) atom("H$_3$C",R);
                            5: bond(r) atom("O",C,L) atom("H$_3$C",R);  };
            };
       };
    0: bond(-90,o,NN) atom("C",C,R) atom("OOCH$_3$",L);
    4: bond(-60,<<) atom("H");
    7: atom("N"); }
}
```

Treten andere Ringgrößen hinzu, ist es am einfachsten, ein Grundgerüst aus Fünf- und Sechsringen aufzubauen und die größeren oder kleineren unregelmäßigen Ringe in Form von Kohlenstoffbrücken hinzuzufügen, was im nächsten Abschnitt angesprochen wird.

2.2.6 (Kohlenstoff-) Brücken

Brücken können in verschiedenen Formen in Erscheinung treten. In den folgenden



Fällen (a) (b) (c) werden die gestrichelten Bindungen als Brücke betrachtet.

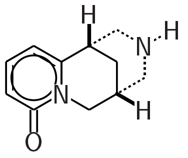
Fall (a) zeichnet sich dadurch aus, daß die Brücke eine Kopie eines Teilstückes des Grundkörpers ist, das nur verschoben wurde. Es treten keine Positionierungsprobleme auf, wenn Sie die entsprechenden Teile des Grundkörpers exakt übernehmen (Parallelverschiebungen sind winkeltreu).

Fall (b) ist problematisch, weil die Brücke kein Pendant im Grundgerüst hat. Wenn Sie die Brücke von einem Brückenkopf her aufbauen, wissen Sie spätestens bei der Bindung, die den Ringschluß herstellt (die letzte der Brücke), nicht, welchen Winkel und welche Länge Sie wählen müssen, um exakt auf den Brückenkopf zu treffen. Hier empfehle ich, die Brücke von beiden Brückenköpfen her aufzubauen und die letzte Position eines Teilstückes zu speichern. Diese Position dient als Ziel einer wilden Linie, die vom anderen Teilstück her mit `bond(#n)` gezeichnet wird. Als wilde Linie wählen Sie am besten diejenige Bindung des Brückenzuges aus, von der Sie am wenigsten wissen, wie Winkel und Länge zu wählen sind.

Fall (c) ist nochmals problematischer, weil die Brücke so kurz ist, daß keine wilde Linie das unbekannte Teilstück überbrücken kann. Hier kann helfen, wenn Sie durch eine unsichtbare Bindung einen Punkt anspringen, der die Brücke symmetrische teilt, dessen Position speichern und dann, wiederum mit `bond(#n)`, zwei wilde Linien von den beiden Brückenköpfen her ziehen. Welchen Punkt Sie als Zielpunkt auswählen, hängt leider etwas von ihrer geometrischen Phantasie ab.

Fall (a)

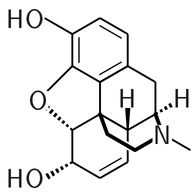
Ein Beispiel für den einfachen Falle, daß die Brücke einem bereits vorhandenen Teilstück entspricht, das nur verschoben ist, ist das Cytisin. Die kopierte und verschobene Brücke ist gestrichelt dargestellt (was in diesem Falle keine stereochemische Aussage sein soll!!!):



(−)-Cytisin

```
formula(L,R,"($-$)-Cytisin",HR,24)
{ ring(,5,1=3=){ 0: bond(r,=C) atom("O");
                 5: atom("N");
                }
  ring(,1,H){ 3: bond(r,t) atom("H");
              5: bond(r,t) atom("H");
              3: bond(30,o; -30,o) atom("N")
                 branch { bond(30,o) atom("H"); }
                 bond(-90,o; -150,o);
            }
}
```

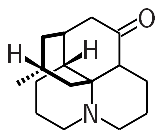
Auch das Morphin weist eine Verbrückung auf, die einem Ausschnitt des Sechsringes gleicht und nach der Verschiebung wieder exakt auf ein Ringatom des Grundkörpers trifft. Im Gegensatz zum Cytisin werden hier aber kurze Bindungen mit „krummen“ Winkeln für die Verschiebung sowie dickere Bindungen für die Brücke benutzt, um einen perspektivischen Eindruck zu erreichen. Der Furanring ist durch drei anellierte Sechsringe sehr verzogen und wird in diesem Falle asymmetrisch wiedergegeben, wobei der Ringschluß durch eine wilde Linie erfolgt:



Morphin

```
formula(L,R,"Morphin",HR,24)
{ ring(,,H1=3=5=)
  { 2: bond(r) atom("HO",R);
    1: bond(-135,,L) atom("O") saveXY(#1);
  }
  ring(,2,H)
  { 1: ring(,3,H5=)
       { 1: bond(r,<.) atom("HO",R);
         2: bond(#1,<.);
         4: bond(t,<<) atom("H");
       };
    5: bond(t,<.) atom("H");
    1: bond(-70,t,S; -30,p; 30,t)
       saveXY(#2)
       atom("N") bond(110,t,S)
       restoreXY(#2) bond(-30);
  }
}
```

Die Brücke im Lycopodin ist ein weiteres Beispiel für verschobene Repliken des Grundkörpers:

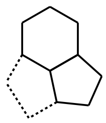


Lycopodin

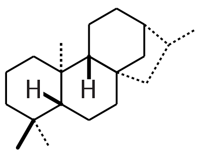
```
formula(L,R,"Lycopodin",HR,24)
{ ring(,2,H){}
  ring(,4,H)
  { 3: bond(b,<<) atom("H");
    5: atom("N");
  }
  ring(,0,H)
  { 4: bond(r,=C) atom("O");
    0: bond(-170,t,S; 150,p) saveXY(#1)
       bond(90,t; 10,t,S)
       restoreXY(#1) bond(-150,<.)
       restoreXY(#1) bond(150,<<) atom("H");
  }
}
```

Fall (b)

Die folgenden beiden Beispiele weisen eine Brücke auf, die durch Annäherung von beiden Brückenköpfen her aufgebaut wird, der Ringschluß erfolgt durch eine wilde Linie (Brücke wiederum gestrichelt gezeichnet):



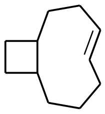
```
formula(L,R)
{ ring(,2,H)
  { 1: bond(-120,o) saveXY(#1);
    vertex(,5,2,H,5){ 1: bond(-150,o; #1,o); };
  }
}
```



(−)-Kauran

```
formula(L,R,"($-$)-Kauran",HR,24)
{ ring(,,H)
  { 3: bond(-90,t) atom("H");
    2: bond(b,o);
    1: bond(b,t) atom("H");
    vertex(,1,1,H,6){3: bond(-120,t); 3: bond(-60,o); };
    vertex(,3,3,H,6){ 1: bond(-40,o) saveXY(#1) bond(30,o);
                      3: bond(-20,o) bond(#1,o);
                    };
  }
}
```

Die Darstellung einer dem Humulen nahe verwandten Struktur bringt Schwierigkeiten, wenn die E/Z-Isomerie, die aus der Doppelbindung im Neunring resultiert, korrekt wiedergegeben werden soll. Am besten fassen Sie den Neunring als Brücke mit einem unbekannten Teilstück auf, das durch eine wilde Linie wiedergegeben wird. (Im Beispiel entsteht eine Asymmetrie, da diese wilde Linie nicht exakt in der vertikalen Mitte des Vierringes endet; wollen Sie eine perfekt symmetrische Formel, müssen Sie Lösungen analog dem Falle (c) suchen):

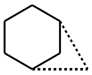


(E)-Isomer

```
formula(L,R,"(E)-Isomer",HR,24)
{ ring(,,H,#N,4,-45)
  { 3: bond(70; 10; -50; -110,=) saveXY(#1);
    0: bond(-70; -10; 50; #1);
  }
}
```

Fall (c)

Die Lösung dieses Problems wurde bereits gegeben, im Beispiel ist es sogar einfach, den Zielpunkt zu bestimmen, da er in radialer Verlängerung des Ringatoms, das sich zwischen den beiden Brückenköpfen befindet, liegt (hier Position 5).



(c)

```
formula(L,R,"(c)",HR,24)
{ ring(,,H){ 5: saveXY(#1,-30,N);
             0: bond(#1,o);
             4: bond(#1,o); }
}
```

Wilde Linien

Aus den vorangegangenen Beispielen ist deutlich geworden, wie nützlich wilde Linien sind (vor allem zum Pfuschen). Sind die Brücken sehr kompliziert und müssen Sie daher unübersichtliche Befehlsketten schreiben, kann es sinnvoll sein, die Position eines oder beider Brückenköpfe mit `saveXY` zu speichern und später, außerhalb der geschachtelten Beschreibungen des Grundkörpers, mit `restoreXY` wieder aufzugreifen.

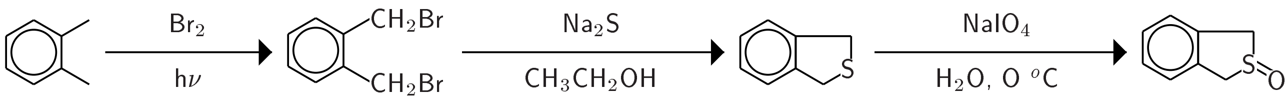
2.3 Reaktionsketten

Das Konzept der automatischen Positionierung von Formeln zueinander kann mit dem weiteren Grundelement des Reaktionspfeils so erweitert werden, daß nicht nur Formeln nebeneinander gestellt, sondern mit zwischengesetzten Pfeilen zu ganzen Reaktionsschemata oder -ketten verbunden werden. Neben einer einfachen linearen Abfolge sind daneben auch Verzweigungen und deren Gegenteil, die Zusammenführung sowie mehrzeilige Schemata möglich.

2.3.1 Horizontale lineare Anordnung

Anstelle der bereits bekannten beiden Positionierungsparameter einer Formel verfügt das Pfeilelement über die Angabe einer Richtung und Länge, um den Anschlußpunkt zu bestimmen. Der Pfeil beginnt dabei stets am aktuellen Startpunkt.Im einfachsten Falle einer horizontal von links nach rechts gerichteten Kette notieren Sie eine Folge von Formeln mit den Positionierungen (L,R) sowie von Pfeilen mit einem Winkel von 0° (Voreinstellung), wie Schema 2–2zeigt:

```
\begin{chemistry}[hor1]
  formula(L,R){ ring(){ 4: bond(r);  5: bond(r); } }

  arrow()
  { text(T,C){ formula(C,C){ atom("Br$_2$") }}
    text(B,C){ formula(C,C){ atom("h$\nu$") }}
  }

  formula(L,R)
  { ring(){ 4: bond(r) atom("CH$_2$Br",L);
            5: bond(r) atom("CH$_2$Br",L); }
  }

  arrow()
  { text(T,C){ formula(C,C){ atom("Na$_2$S") }}
    text(B,C){ formula(C,C){ atom("CH$_3$CH$_2$OH") }}
  }

  formula(L,R)
  { ring(,5){}
    ring("cpentane",1){ 4: atom("S"); }
  }

  arrow()
  { text(T,C){ formula(C,C){ atom("NaIO$_4$") }}
    text(B,C){ formula(C,C){ atom("H$_2$O, 0 $^o$C") }}
  }

  formula(L,R)
  { ring(,5){}
    ring("cpentane",1)
    { 4: atom("S") bond(r,=C) atom("O"); }
  }
\end{chemistry}
```



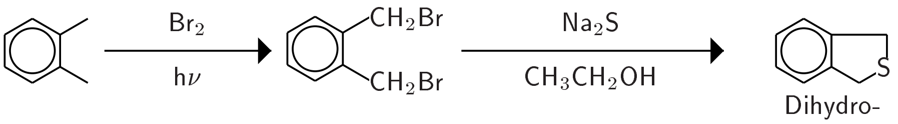
Schema 2–2 Horizontales Reaktionsschema: Darstellung eines Benzothiophens.

Die einzelnen Formeln werden mit dieser Positionierung auf eine gemeinsame Mittellinie, auf der die Reaktionspfeile liegen, zentriert. Es ist auch denkbar, die Ketten von rechts nach links aufzubauen, wobei die Positionierung (R,L) benutzt wird, für die Pfeile ein Winkel von 180°.

Eine Erweiterung der Formelsyntax läßt die Beschriftung der Komponenten zu und ist speziell für den Satz von Formeln in horizontalen Ketten geeignet. Sie müssen dazu den Typ `HA` (horizontal, absoluter Abstand) wählen und den Abstand der Text*unterkante* von der Mittellinie angeben, wie Schema 2–3 zeigt. Der Befehl `\shortstack` erlaubt es, Zeilenumbrüche innerhalb des Textes vorzunehmen:

```
\begin{chemistry}[hor2]
  formula(L,R,"o-Xylen",HA,36)
```

```
  { ring(){ 4: bond(r);  5: bond(r); } }

  arrow()
  { text(T,C){ formula(C,C){ atom("Br$_2$") }}
    text(B,C){ formula(C,C){ atom("h$\nu$") }}
  }

  formula(L,R,"$\alpha,\alpha'$-o-Xylen",HA,36)
  { ring(){ 4: bond(r) atom("CH$_2$Br",L);
            5: bond(r) atom("CH$_2$Br",L); }
  }

  arrow()
  { text(T,C){ formula(C,C){ atom("Na$_2$S") }}
    text(B,C){ formula(C,C){ atom("CH$_3$CH$_2$OH") }}
  }

  formula(L,R,"\shortstack{Dihydro-\\benzo[b]thiophen}",HA,36)
  { ring(,5){}
    ring("cpentane",1){ 4: atom("S"); }
  }
\end{chemistry}
```



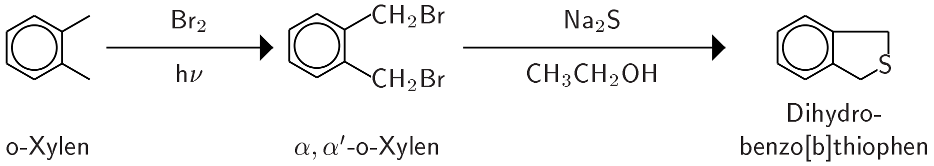
o-Xylen α,α'-o-Xylen Dihydro-benzo[b]thiophen

Schema 2–3 Horizontales Reaktionsschema mit Formelnamen, die auf eine gemeinsame Linie unterhalb des Reaktionspfeils ausgerichtet sind (fester Abstand des Textes zur Mittellinie).

Bei diesem Verfahren ist es erforderlich, den benötigten Abstand der Texte von der Mittellinie abzuschätzen oder durch Versuche zu ermitteln. In beiden Fällen können Sie nur schwer für mehrere Schemata einen gleichmäßigen Abstand erreichen. Der Befehl `multiline`, der speziell für den Satz n-zeiliger horizontaler Reaktionsketten konzipiert ist, trägt dem jedoch Rechnung, indem er in jeder Zeile die am tiefsten liegende Formelunterkante bestimmt und zu dieser einen konstanten Abstand `rTextSep` einhält, um die Höhe der gesamten Textzeile zu berechnen. Alle Formelbeschriftungen in dieser Zeile liegen somit auf gleicher Höhe und halten einen identischen Mindestabstand von der größten Formel ein. Wählen Sie $n = 1$, können Sie das angesprochene Problem lösen (Schema 2–4). Beachten Sie, daß Formeln innerhalb von `multiline` keine Positionierungsparameter für die Beschriftung (`HR` und ähnliche) besitzen dürfen, da diese Werte ja selbständig bestimmt werden:

```
\begin{chemistry}[hor3]
multiline(1)
{ formula(L,R,"o-Xylen")
  { ring(){ 4: bond(r);  5: bond(r); } }

  arrow()
  { text(T,C){ formula(C,C){ atom("Br$_2$") }}
    text(B,C){ formula(C,C){ atom("h$\nu$") }}
  }
```

```
  formula(L,R,"$\alpha,\alpha'$-o-Xylen")
  { ring(){ 4: bond(r) atom("CH$_2$Br",L);
            5: bond(r) atom("CH$_2$Br",L); }
  }

  arrow()
  { text(T,C){ formula(C,C){ atom("Na$_2$S") }}
    text(B,C){ formula(C,C){ atom("CH$_3$CH$_2$OH") }}
  }

  formula(L,R,"\shortstack{Dihydro-\\benzo[b]thiophen}")
  { ring(,5){}
    ring("cpentane",1){ 4: atom("S"); }
  };
}
\end{chemistry}
```



Schema 2–4 Horizontales Reaktionsschema mit Formelnamen, die automatisch in konstantem Abstand auf einer gemeinsamen Linie unterhalb des Reaktionspfeils ausgerichtet sind.

Zur Bestimmung des korrekten Abstands der Texte von der Mittellinie werden nur diejenigen Formeln herangezogen, die auch eine Beschriftung aufweisen. Der Fall, daß ein sehr ausgedehntes Zwischenprodukt (zum Beispiel ein vertikal dargestelltes Isomerengemisch ohne Text) die Höhe der Beschriftungen der gesamten Zeile bestimmt, tritt somit nicht ein, wie Sie in Schema 2–8 sehen können.

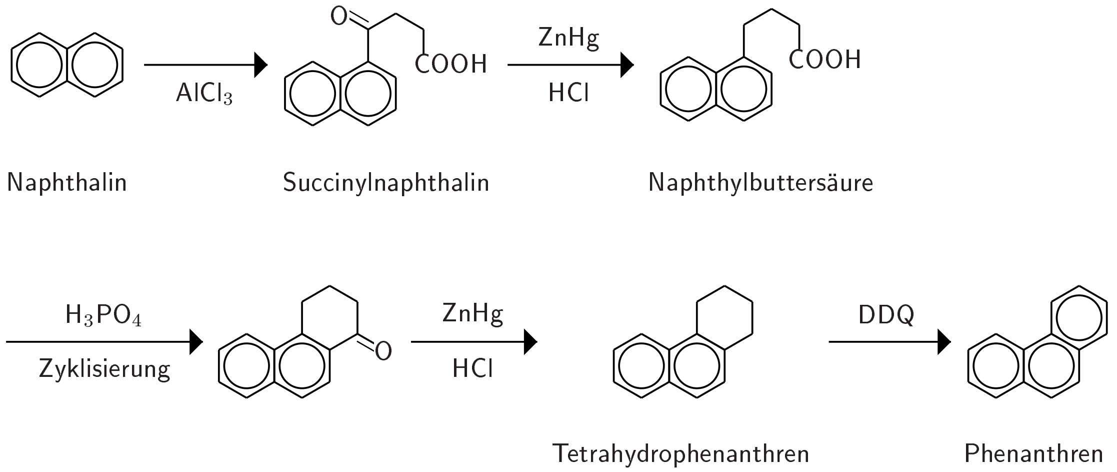
2.3.2 Mehrzeilige Schemata

Die Diskussion ging bislang von einzelnen Reaktionszeilen aus. Es ist jedoch möglich, mehrere solcher Zeilen übereinandersetzen zu lassen, wobei automatisch zwischen den einzelnen Zeilen ein konstanter Abstand `rMultilineSep` gesetzt wird, unter Berücksichtigung eventueller Beschriftungen (Schema 2–5).

```
\begin{chemistry}[multli1]
set("rArrowExtend",12)
multiline(2,L)
{ % line 1
  formula(L,R,"Naphthalin")
  { ring(,5){}  ring(,1){} }

  arrow(){ text(B,C){ formula(C,C){ atom("AlCl$_3$") } } }

  formula(L,R,"Succinylnaphthalin")
```

```
  { ring(,5){}
    ring(,1)
    { 3: bond(r)
         branch { bond(150,=) atom("O"); }
         bond(30; -30; -90) atom("C",C,R) atom("OOH",L); }
  }

  arrow()
  { text(T,C){ formula(C,C){ atom("ZnHg") } }
    text(B,C){ formula(C,C){ atom("HCl") } } }

  formula(L,R,"Naphthylbutters""aure")
  { ring(,5){}
    ring(,1){ 3: bond(r; 30; -30; -90) atom("C",C,R) atom("OOH",L); }
  }
  ;
  %line 2
  arrow()
  { text(T,C){ formula(C,C){ atom("H$_3$PO$_4$") } }
    text(B,C){ formula(C,C){ atom("Zyklisierung") } } }

  formula(L,R)
  { ring(,5){}
    ring(,1){ 3: ring(,1,H){ 5: bond(r,=) atom("O"); }; }
  }

  arrow()
  { text(T,C){ formula(C,C){ atom("ZnHg") } }
    text(B,C){ formula(C,C){ atom("HCl") } } }

  formula(L,R,"Tetrahydrophenanthren")
  { ring(,5){}
    ring(,1){ 3: ring(,1,H){}; }
  }

  arrow(){ text(T,C){ formula(C,C){ atom("DDQ") } } }

  formula(L,R,"Phenanthren")
  { ring(,5){}
    ring(,1){ 3: ring(,1){}; }
  }
  ;
}
\end{chemistry}
```

Sie können auf diese Weise eine lange Reaktionskette in mehrere Zeilen brechen oder auch eine Schar von unabhängigen kurzen Ketten übereinander setzen. Darüberhinaus findet der Befehl Anwendung beim Zusammenfassen mehrerer einzelner Formeln oder größeren Reaktionsausschnitten, Beispiele hierzu sind im Abschnitt 2.3.5 gegeben.



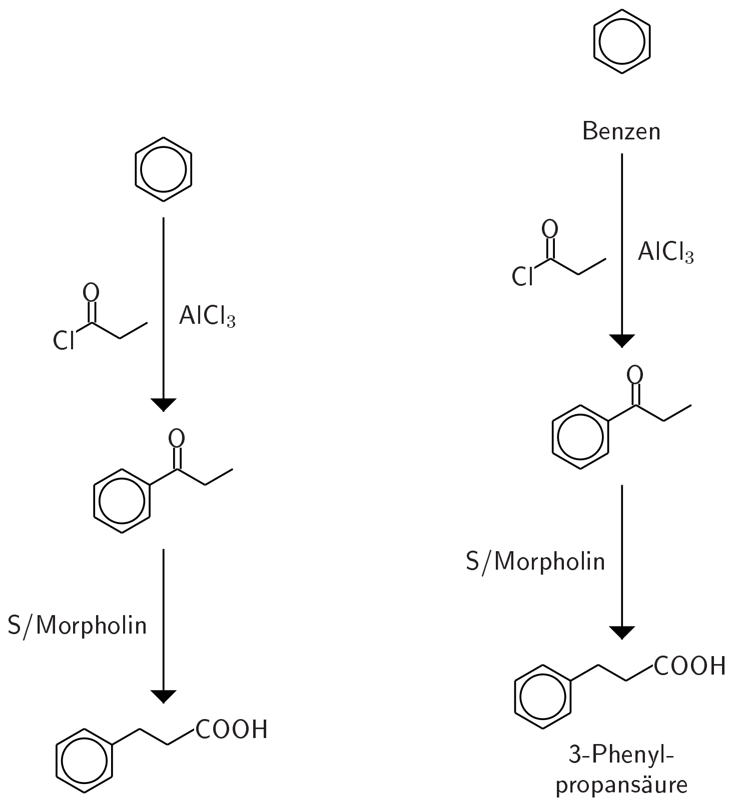
Schema 2–5 Synthese von Phenanthren, mehrzeilig gesetzt mit `multiline`.

2.3.3 Vertikale Anordnung

Ebenso einfach wie horizontale Ketten können vertikal verlaufende Abläufe gesetzt werden, wobei die Hauptrichtung von oben nach unten mit der Positionierung (T,B), seltener von unten nach oben mit (B,T) erreicht wird, wie man anhand des Schemas 2–6 (links) sieht. Die Pfeile weisen nun unter einem Winkel von -90° nach unten:

```
\begin{chemistry}[vert1]
  formula(T,B)
  { ring(){} }

  arrow(-90)
  { text(T,C){ formula(C,C){ atom("Cl") bond(30)
                 branch{ bond(90,=C) atom("O"); }
                         bond(-30; 30) }}
    text(B,C){ formula(C,C){ atom("AlCl$_3$") }}
  }

  formula(T,B)
  { ring()
    { 4: bond(r) branch { bond(r+,=C) atom("O"); }
         bond(r-; r); }
  }

  arrow(-90)
  { text(T,C){ formula(C,C){ atom("S/Morpholin") }} }

  formula(T,B)
  { ring()
    { 4: bond(r; r-; r) atom("COOH",L,R); }
  }
```

```
\end{chemistry}
```



Schema 2–6 Vertikales Reaktionsschema: Darstellung der 7-Oxononansäure. Rechts dasselbe Schema mit Bennenung der Komponenten. Die Texte weisen einen festen Abstand zur Formelunterkante auf.

Auch hierbei ist eine Beschriftung der Formeln möglich, wobei in jedem Falle der Abstand der Textunterkante von der Formelunterkante angegeben werden muß (Typ `V`), da keine horizontale Mittellinie existiert. Auch in diesem Falle können Zeilenumbrüche mit `\shortstack` innerhalb des Textparameters auftreten (Schema 2–6 rechts):

```
\begin{chemistry}[vert2]
  formula(T,B,"Benzen",V,24)
  { ring(){} }

  arrow(-90)
  { text(T,C){ formula(C,C){ atom("Cl") bond(30)
                branch{ bond(90,=C) atom("O"); }
                bond(-30; 30) }}
    text(B,C){ formula(C,C){ atom("AlCl$_3$") }}
  }

  formula(T,B)
  { ring()
    { 4: bond(r) branch { bond(r+,=C) atom("O"); }
         bond(r-; r); }
  }
```

```
  arrow(-90)
  { text(T,L){ formula(C,C){ atom("S/Morpholin") }} }

  formula(T,B,"\shortstack{3-Phenyl-\\propans""aure}",V,24)
  { ring()
    { 4: bond(r; r-; r) atom("COOH",L,R); }
  }
\end{chemistry}
```

Bislang wurde davon ausgegangen, daß die Hauptrichtung der Kette stets dieselbe ist – von links nach rechts oder von oben nach unten. Dies muß nicht immer der Fall sein, es steht Ihnen frei, die Positionierungsparameter einer Formel zum Beispiel auf `L,B` zu setzen, um den Verlauf an dieser Stelle von links-rechts nach oben-unten zu ändern und einen Knick zu erreichen.

2.3.4 Kontext und Verzweigung

Es ist nun an der Zeit, das Konzept des *Kontextes* zu verdeutlichen. Implizit haben Sie es mit dem Zeichnen des ersten Schemas bereits angewandt, ohne es zu wissen. Ein Kontext ist nichts weiter als die Bounding Box einer Formel. Er wird eingesetzt, um die Koordinaten der neuen Kardinalpunkte der Formel (die vier Ecken, die vier Seitenmitten sowie der Mittelpunkt) anhand der Positionierungsparameter `<pos>` und `<Cpos>` zu berechnen. Jeder `formula`-Befehl setzt den aktuellen Kontext auf die Bounding Box der durch ihn gezeichneten Formel.

Unmittelbar sichtbar kann der Kontext durch die Befehle `fbox` oder `bracket` gemacht werden: sie nutzen den aktuellen Kontext, um die zugehörige Formel einzurahmen oder mit eckigen Klammern einzuschließen.

Ein wichtiges Merkmal von Reaktionsketten, das grundlegend auf dem Kontext aufbaut, ist die *Verzweigung*, durch die Sie von einer Formel ausgehend, mehrere Ketten weiterlaufen lassen können. Dabei speichern Sie mit `savecontext` den Kontext und damit die geometrische Ausdehnung der Formel unter einer bestimmten Nummer und können so immer, wenn Sie ihn erneut benötigen, mit `setcontext` unter dieser Nummer darauf zugreifen. Dies ist zu Beginn jeder weiteren Reaktionskette erforderlich, da durch die folgenden Formeln der Kontext der Verzweigungsformel überschrieben ist.

Eine einfache Verzweigung der Reaktionskette wird in der Beschreibung zum Befehl `setcontext` in Schema 3–4 gegeben. Hier soll anhand der Reaktionsmöglichkeiten eines Borans die Mehrfachverzweigung dargestellt werden. Ebenso wie bei der einfachen Verzweigung speichern Sie den Kontext der Formel, die als Verzweigungspunkt dienen soll, ein einziges Mal ab, da er sich nicht ändert. Zum Unterschied vom einfachen Falle verwenden Sie ihn aber mehrmals, um mit `setcontext` einen Anschlußpunkt zu berechnen:

```
\begin{chemistry}[cont2]
  formula(L,R)
  { atom("R") bond(30; -30,=) }

  arrow()
  { text(T,L){ formula(C,C){ atom("H") bond(0) atom("B")
```

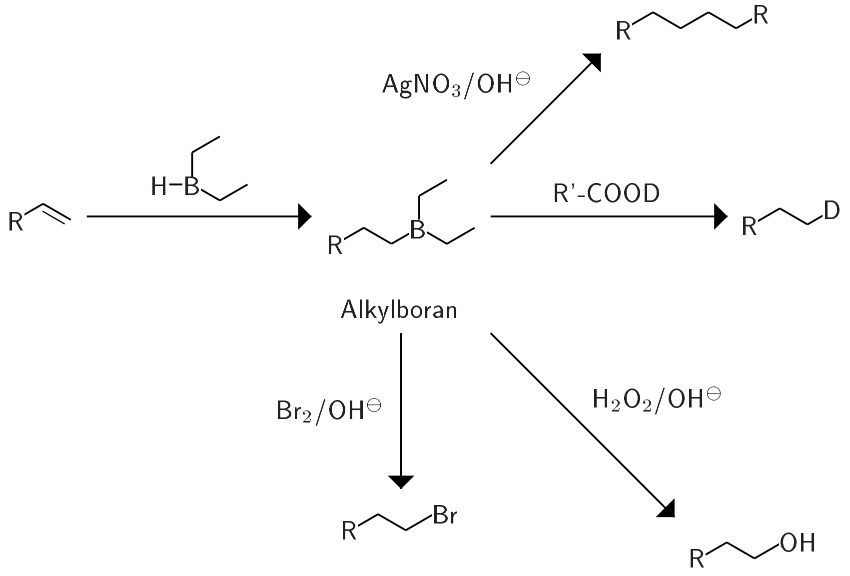
```
      branch{bond(90; 30); bond(-30; 30);} }} }

  formula(L,R,"Alkylboran",HR,24)
  { atom("R") bond(30; -30; 30)
    atom("B")
    branch{bond(90; 30); bond(-30; 30);}
  }
  savecontext(#1)

  arrow()
  { text(T,L){ formula(C,C){ atom("R'-COOD") }} }
  formula(L,R)
  { atom("R") bond(30; -30; 30) atom("D") }

  setcontext(#1,TR)
  arrow(45)
  { text(T,L){ formula(C,C){ atom("AgNO$_3$/OH$^\ominus$") }} }
  formula(BL,R)
  { atom("R") bond(30; -30; 30; -30; 30) atom("R") }

  setcontext(#1,BR)
  arrow(-45)
  { text(T,L){ formula(C,C){ atom("H$_2$O$_2$/OH$^\ominus$") }} }
  formula(TL,R)
  { atom("R") bond(30; -30; 30) atom("OH",L) }

  setcontext(#1,B)
  arrow(-90)
  { text(T,L){ formula(C,C){ atom("Br$_2$/OH$^\ominus$") }} }
  formula(T,R)
  { atom("R") bond(30; -30; 30) atom("Br",L) }
\end{chemistry}
```

Weitere Möglichkeiten der Verzweigungen, wobei mehrere Formeln einen gemeinsamen Kontext bilden und so als Einheit agieren, werden im folgenden Abschnitt vorgestellt.

2.3.5 Reaktionsabschnitte als Einheit betrachten

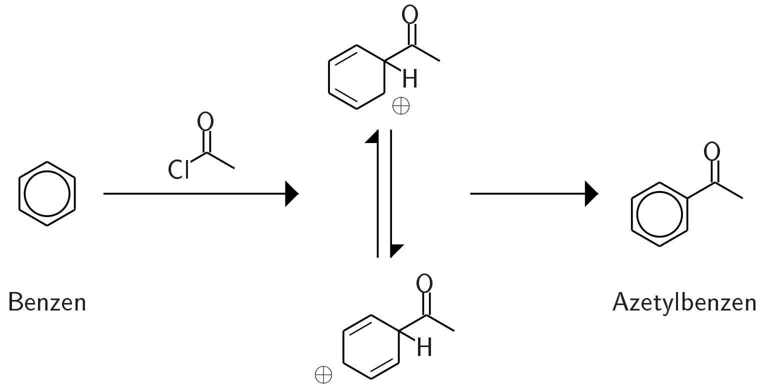
So kann der Wunsch bestehen, innerhalb einer horizontalen Kette ein Zwischenprodukt mit mehreren mesomeren Zuständen vertikal übereinander darzustellen, wie dies in Schema 2–8 zu sehen ist. Hier besteht die Lösung darin, einen `multiline`-Befehl mit einem kleinen vertikal orientierten Schema als *einzeiligen* Inhalt zu verwenden. Die beiden Grenzstrukturen können so als Einheit betrachet werden, die durch die Positionierung `L,R` zur Reaktionskette zentriert ist. Die innere Ausrichtung ist bei einzeiligem Inhalt von `multiline` sinnlos und kann beliebig gewählt werden. Zusätzlich wurde eine `multiline`-Umgebung benutzt, um die Formelbeschriftung auf gleiche Höhe zu bringen:

```
\begin{chemistry}[multli2]
multiline(1)
{ formula(L,R,"Benzen"){ ring(){} }

  arrow()
```



Schema 2–7 Was passiert mit dem Boran? Mehrfachverzweigungen, von derselben Formel ausgehend, können es zeigen. Der Kontext der Verzweigungsstelle, hier des Borans, muß dafür gespeichert werden.

```
  { text(T,L){ formula(C,C){ atom("Cl") bond(30)
      branch{bond(90,=C) atom("O");} bond(-30) } }
  }

  multiline(1,C)
  { formula(T,B)
    { ring(,,H0=2=)
      { 4: bond(30) branch{ bond(90,=C) atom("O");} bond(-30);
        4: bond(b) atom("H");
        5: bond(r,s,S) atom("$\oplus$"); }
    }
    arrow(-90,,<=>){}
    formula(T,B)
    { ring(,,H2=5=)
      { 4: bond(30) branch{ bond(90,=C) atom("O");} bond(-30);
        4: bond(b) atom("H");
        1: bond(r,s,S) atom("$\oplus$"); }
    };
  }

  arrow(){}

  formula(L,R,"Azetylbenzen")
  { ring()
    { 4: bond(r) branch { bond(r+,=C) atom("O");} bond(r-); }
  }
;}
\end{chemistry}
```



Schema 2–8 Ein unabhängiges vertikales Schema wird als einzeiliges `multiline`-Objekt gesetzt und kann so als Einheit in den horizontalen Formelverlauf eingefügt werden.

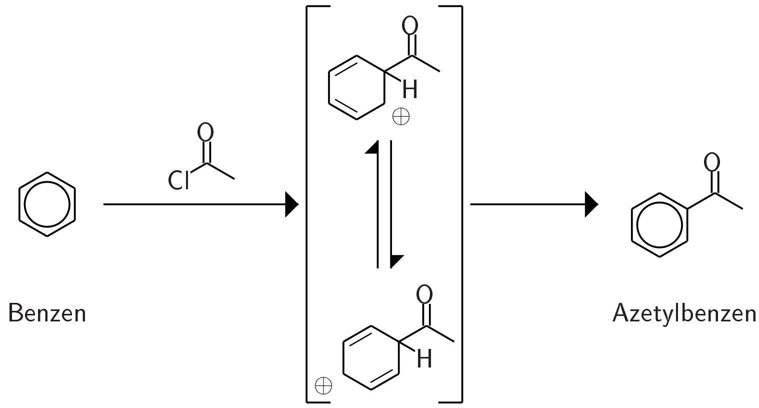
Sie können mit diesem Verfahren ebensogut horizontal angeordnete Grenzstrukturen oder Reaktionsabschnitte innerhalb von vertikalen Ketten anordnen, wobei `T,B` als Positionierung eingesetzt werden muß. Dies ist beim Gemisch verschiedener Chlorbutene im Schema 2–22 benutzt worden.

2.3.6 Kontexte zusammenfassen

`multiline` kann Ihnen auch weiterhelfen, wenn Sie einige Formeln zusammenfassen wollen, um eine Klammerung aller beteiligten Formeln erzielen möchten, zum Beispiel um die mesomeren Zwischenstrukturen des Schemas 2–8. Der Befehl `bracket` klammert jedoch nur die letzte gezeichnete Formel. Eine einzeilige `multiline`-Einheit liefert nach außen hin einen *Superkontext*, das heißt einen Kontext, der alle eingeschachtelten Kontexte enthält und es daher erlaubt, eine Kettenverzweigung oder Einrahmung auf alle beteiligten Formeln zu beziehen. Betrachten Sie als Beispiel für eine Einrahmung Schema 2–9:

```
\begin{schema}
\begin{chemistry}[multli3]
multiline(1)
{ formula(L,R,"Benzen"){ ring(){} }

  arrow()
  { text(T,L){ formula(C,C){ atom("Cl") bond(30)
      branch{bond(90,=C) atom("O");} bond(-30) } }
  }

  multiline(1,C)
  { formula(T,B)
    { ring(,,HO=2=)
      { 4: bond(30) branch{ bond(90,=C) atom("O");} bond(-30);
        4: bond(b) atom("H");
        5: bond(r,s,S) atom("$\oplus$"); }
    }
    arrow(-90,,<=>){}
```

```
    formula(T,B)
    { ring(,,H2=5=)
      { 4: bond(30) branch{ bond(90,=C) atom("O");} bond(-30);
        4: bond(b) atom("H");
        1: bond(r,s,S) atom("$\oplus$"); }
    }
    ;
  }
  bracket()
  arrow(){}

  formula(L,R,"Azetylbenzen")
  { ring()
    { 4: bond(r) branch { bond(r+,=C) atom("O");} bond(r-); }
  }
;}
\end{chemistry}
```

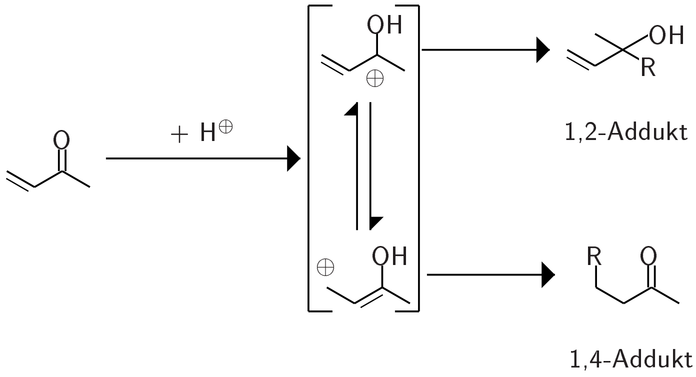


Schema 2–9 Ein Superkontext mit `multiline` erlaubt die Klammerung mehrerer Formeln. Er wird durch ein `multiline`-Objekt gebildet.

Anstelle des `bracket`-Befehls können Sie ebensogut mehrfache Verzweigungen mit `savecontext` und `setcontext` aufbauen, wobei der Superkontext als Grundlage verwendet wird. Unabhängig von diesem können Sie aber auch auf die Einzelkontexte zugreifen, sofern diese *innerhalb* von `multiline` gespeichert werden, wie das Beispiel des Schemas 2–10 zeigt:

```
\begin{chemistry}[multli4]
formula(L,R)
{ bond(-30,=) bond(30)
  branch {bond(90,=C) atom("O");} bond(-30);
}

arrow()
{ text(T,C){ formula(C,C){ atom("+ H$^\oplus$") } } }

multiline(1,C)
{ formula(T,B)
  { bond(-30,=; 30) branch{ bond(-90,s,S) atom("$\oplus$");
                            bond(90) atom("O",C,R) atom("H",L);
                          }
```

```
              bond(-30)
  }
  savecontext(#1)
  arrow(-90,,<=>){}
  formula(T,B)
  { atom("$\oplus$") bond(-90,s,S)
    bond(-30; 30,=)
    branch { bond(90) atom("O",C,R) atom("H",L); }
    bond(-30)
  }
  savecontext(#2)
  ;
}
bracket()

setcontext(#1,R)
arrow(){}
formula(L,R,"1,2-Addukt",HR,24)
{ bond(-30,=; 30)
  branch{ bond(150);
          bond(90,=C) atom("O");
        }
  bond(-30) atom("R")
}
setcontext(#2,R)
arrow(){}
formula(L,R,"1,4-Addukt",HR,24)
{ atom("R") bond(-90)
  bond(-30; 30)
  branch { bond(90,=C) atom("O"); } bond(-30)
}
\end{chemistry}
```

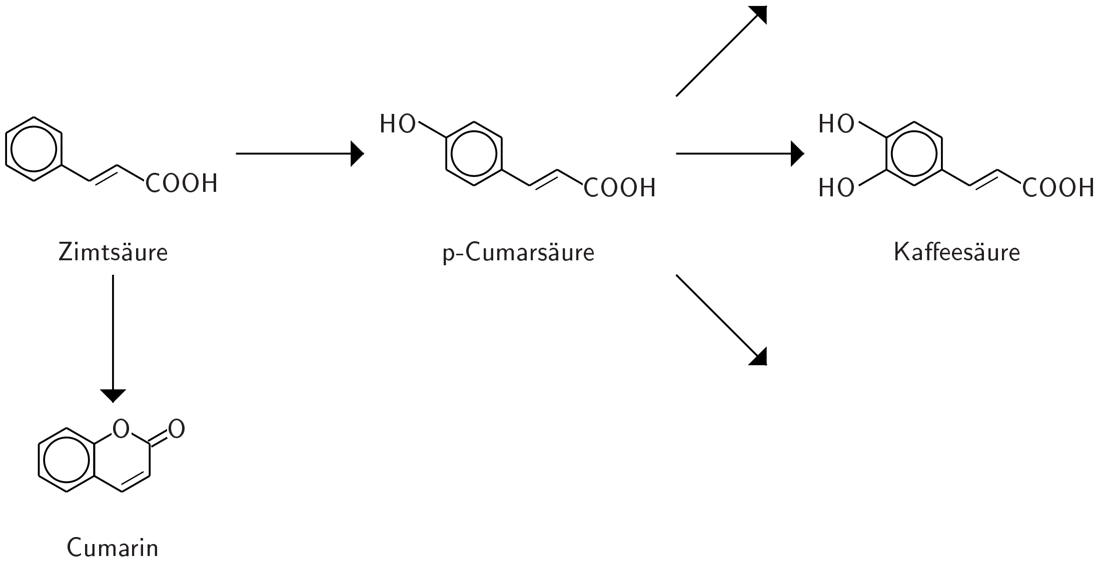


Schema 2–10 Unabhängig vom Superkontext können Verzweigungen auf die einzelnen Teilformeln ausgerichtet werden, wenn deren Kontexte einzeln gespeichert werden.

Das Speichern der Kontexte einzelner Formeln eines `multiline`-Zusammenhanges ist von allgemeiner Bedeutung, wenn zwar `multiline` zum Ausrichten der Formelbeschriftungen benutzt werden soll, aber dennoch die Formeln Verzweigungspunkte darstellen. Hierzu folgt ein weiteres Beispiel (Schema 2–11):

```
\begin{chemistry}[multli5]
```

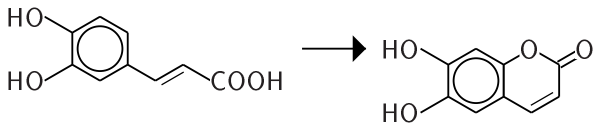
```
multiline(1)
{ formula(L,R,"Zimts""aure")
  { ring(){5: bond(r;r/,=;r) atom("COOH",L); } }
  savecontext(#1)

  arrow(){}
  formula(L,R,"p-Cumars""aure")
  { ring(){5: bond(r;r/,=;r) atom("COOH",L);
           2: bond(r) atom("HO",R); }
  }
  savecontext(#2)

  arrow(){}
  formula(L,R,"Kaffees""aure")
  { ring(){5: bond(r;r/,=;r) atom("COOH",L);
           2: bond(r) atom("HO",R);
           1: bond(r) atom("HO",R); }
  }
;}
setcontext(#1,B) arrow(-90){}
formula(T,B,"Cumarin",V,24)
{ ring(,,H5=)
  { vertex(,1,1){};
    3: atom("O");
    4: bond(r,=C) atom("O"); }
}
setcontext(#2,TR) arrow(45){}
setcontext(#2,BR) arrow(-45){}
\end{chemistry}
```



Schema 2–11 Mit `multiline` angeordnete Formelbeschriftungen und Verzweigungen, ermöglicht durch individuelle Kontexte.

Für jede Formel, die als Verzweigungspunkt dient, wird ein Individualkontext gespeichert. Mit `multiline` erzeugte Formelbeschriftungen gehen dabei in den Kontext mit ein, sodaß Sie bei nach unten führenden Pfeilen wie gezeigt den korrekten Anschlußpunkt erhalten.

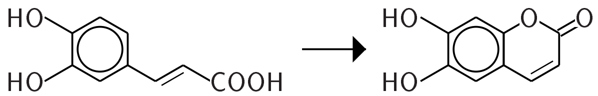
Da der `multiline`-Block als Ganzes betrachtet wird, ergeben sich Positionierungsprobleme, wenn außerhalb des Block eine Formel ohne Beschriftung folgen soll. Deren Mittellinie läge mittig zum gesamten vorangegangegen Block samt Beschriftung:



Kaffeesäure

```
multiline(1)
{ formula(L,R,"Kaffees""aure")
  { ring(){5: bond(r;r/,=;r) atom("COOH",L);
           2: bond(r) atom("HO",R);
           1: bond(r) atom("HO",R); }
          }
          arrow(,12){}
        ;}
        formula(L,R)
{ ring(,,H5=)
  { vertex(,1,1){ 4: bond(r) atom("HO",R);
                  5: bond(r) atom("HO",R); };
    3: atom("O");
    4: bond(r,=C) atom("O"); }
}
```

Die Lösung besteht wiederum darin, den Kontext der letzten Formel zu speichern und nach Abschluß des `multiline`-Blockes zu setzen:



Kaffeesäure

```
multiline(1)
{ formula(L,R,"Kaffees""aure")
  { ring(){5: bond(r;r/,=;r) atom("COOH",L);
           2: bond(r) atom("HO",R);
           1: bond(r) atom("HO",R); }
          }
          savecontext(#1)
        ;}
        setcontext(#1,R)
arrow(,12){}
formula(L,R)
{ ring(,,H5=)
  { vertex(,1,1){ 4: bond(r) atom("HO",R);
                  5: bond(r) atom("HO",R); };
    3: atom("O");
    4: bond(r,=C) atom("O"); }
}
```

2.3.7 Zusammenführung mehrerer Zweige

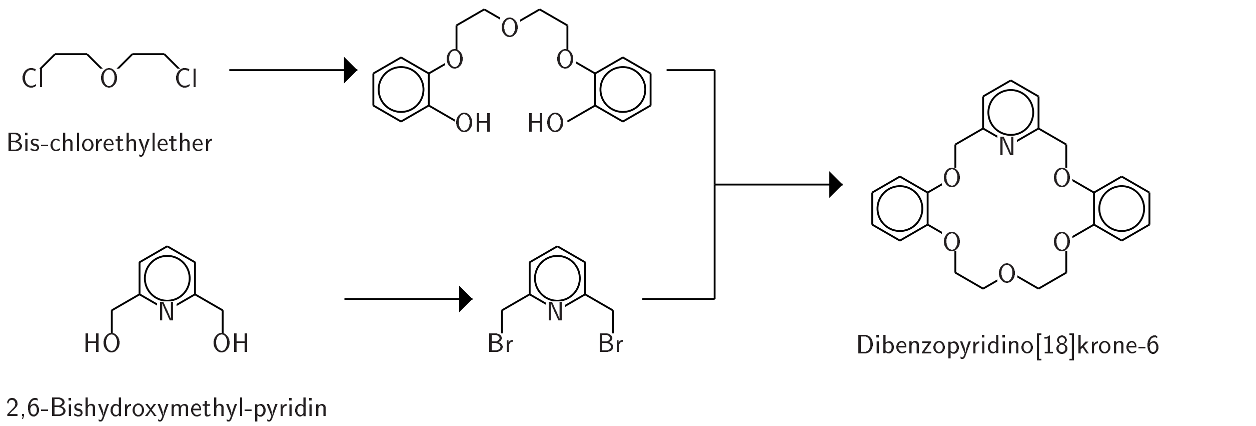
Reaktionsketten können in mehrere Teilzweige aufgespalten werden, aber auch das Gegenteil, die Zusammenführung mehrerer Zweige zu einem einzigen, etwa im Zuge einer konvergenten Synthese, ist oft erforderlich. Je nachdem, ob die Äste horizontal oder vertikal zusammenlaufen sollen, können Sie zwischen den Befehlen `joinh` und `joinv` wählen.

Als Beispiel für ein horizontal ausgerichtetes Schema dient die Synthese eines Kronenethers.

```
\begin{chemistry}[joinh1]
joinh(2,L)
{ % line1
  formula(L,R,"Bis-chlorethylether")
  { atom("Cl") bond(45; 0; -45)
    atom("O") bond(45; 0; -45) atom("Cl")
  }
  arrow(){}
  formula(L,R)
  { ring()
    { 5: bond(r) atom("OH",L);
      4: bond(30) atom("O") bond(90; 30; -30)
         atom("O")
         bond(30; -30; -90) atom("O")
         bond(-30)
         ring(,2){ 1: bond(r) atom("HO",R); }; }
  };

  % line 2
  formula(L,R,"2,6-Bishydroxymethyl-pyridin")
  { ring()
    { 0: atom("N");
      1: bond(r; r+) atom("O",C,L) atom("H",R);
      5: bond(r; r-) atom("O",C,R) atom("H",L); }
  }
  arrow(){}
  formula(L,R)
  { ring()
    { 0: atom("N");
      1: bond(r; r+) atom("Br");
      5: bond(r; r-) atom("Br"); }
  };
}

arrow(){}

formula(L,R,"Dibenzopyridino[18]krone-6",HR,24)
{ ring()
  { 5: bond(-30) atom("O")
       bond(-90; -30; 30) atom("O")
       bond(-30; 30; 90) atom("O") bond(30);
    4: bond(30) atom("O") bond(90; 30)
       ring(,1)
       { 5: bond(-30; -90) atom("O") bond(-30)
            ring(,2) {};
         0: atom("N");
       };
  }
}
\end{chemistry}
```

Auch mehrfache horizontale Zusammenführungen sind möglich, wenn Sie in einen oder mehrere der Zweige weitere `joinh`-Befehle schreiben (Schemata 2–13):

```
\begin{chemistry}[joinhh]
```

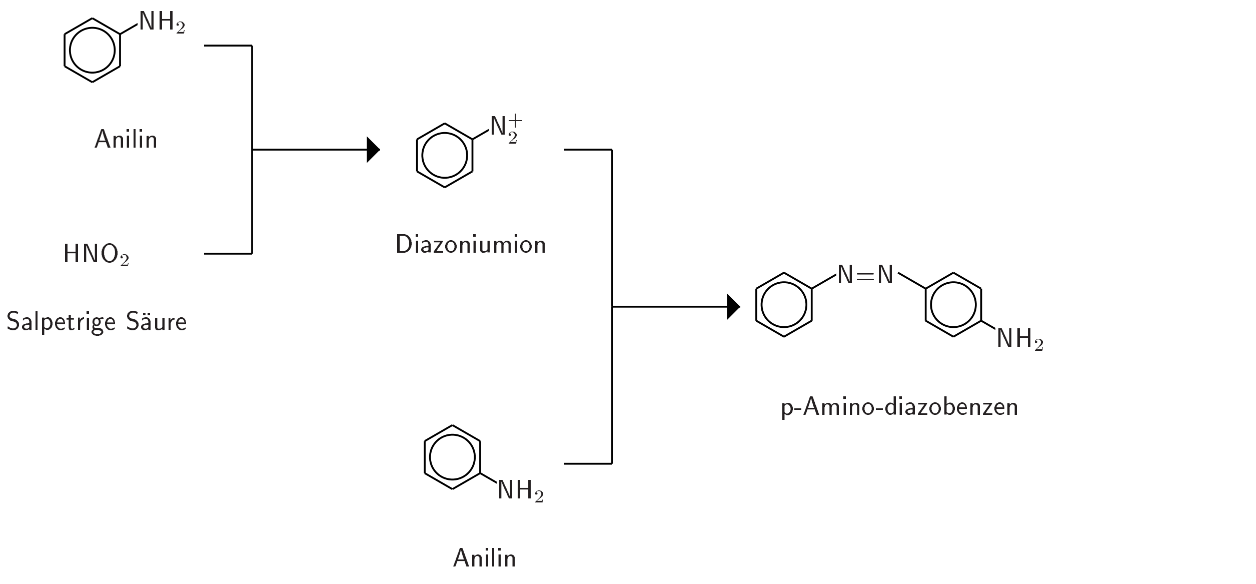


Schema 2–12 Die Synthese eines Kronenethers als Beispiel für die Zusammenführung zweier horizontaler Ketten mit `joinh`.

```
joinh(2,R)
{ joinh(2,R)
  { formula(L,R,"Anilin")
    { ring(){ 4: bond(r) atom("N",C,R) atom("H$_2$",L);} }
    ;
    formula(L,R,"Salpetrige S""aure")
    { atom("HNO$_2$") }
    ;
  }
  arrow(){}
  formula(L,R,"Diazoniumion")
  { ring(){ 4: bond(r) atom("N",C,R) atom("$_2^+$",L);} }
  ;
  formula(L,R,"Anilin")
  { ring(){ 5: bond(r) atom("N",C,R) atom("H$_2$",L);} }
  ;
}
arrow(){}
formula(L,R,"p-Amino-diazobenzen",HR,24)
{ ring(){ 4: bond(r) atom("N=N",L,R) bond(r/)
             ring(,0,,,,r)
             { 3: bond(r) atom("N",C,R) atom("H$_2$",L);};
        }
}
\end{chemistry}
```

Analog können Sie auch vertikale Schemata zusammenführen. Das Schema 2–14 zeigt dies mit zwei ineinandergeschachtelten `joinv`-Einheiten:

```
\begin{chemistry}[joinv1]
set("rArrowExtend",18)
joinv(2,B)
{ joinv(2,B)
  { formula(T,B)
    { bond(30,=; -30; 30)
      branch { bond(150);  bond(30) atom("OH",L); } bond(-30)
```



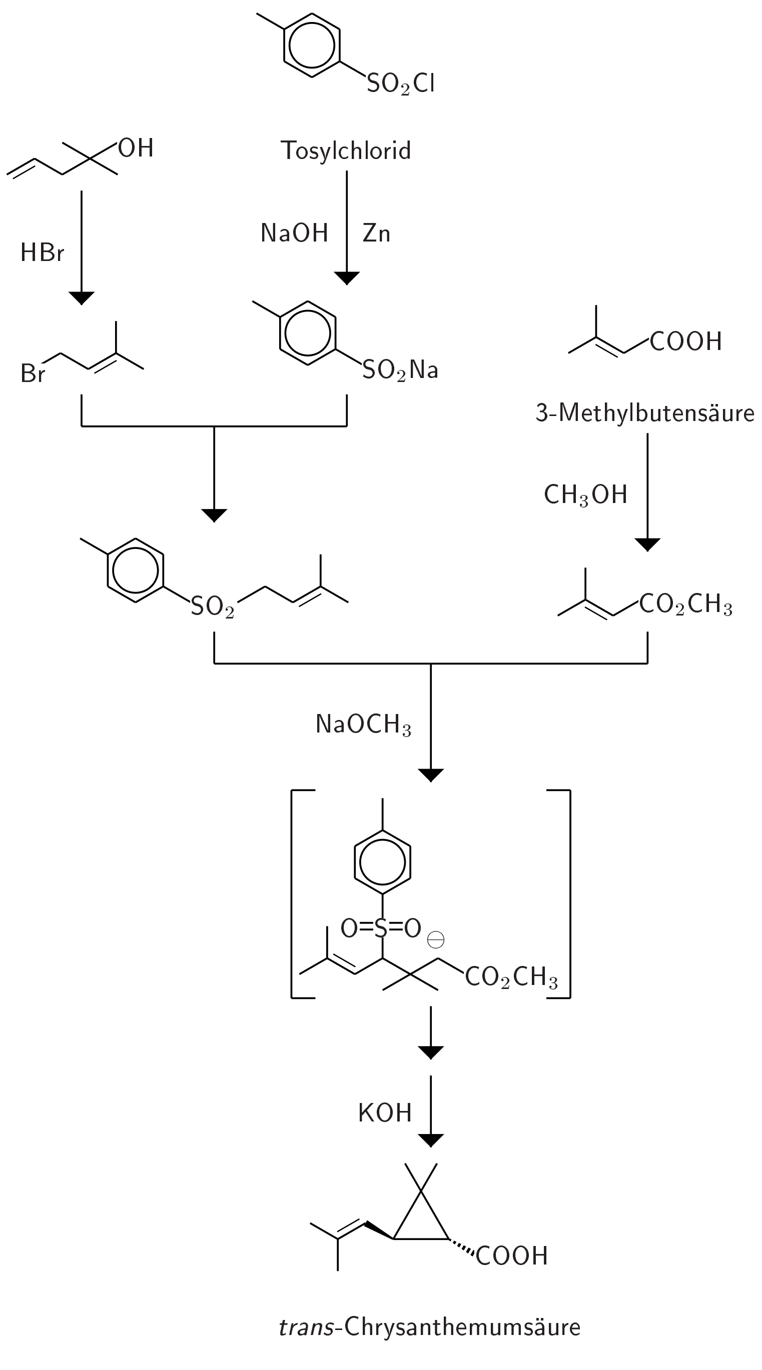
Schema 2–13 Homogene geschachtelte Zusammenführungen mehrerer Reaktionszweige.

```
  }
  arrow(-90){ text(T,L){formula(C,C){atom("HBr")}}}
  formula(T,B)
  { atom("Br") bond(30; -30; 30,=)
    branch { bond(90); } bond(-30)
  };

  formula(T,B,"Tosylchlorid",V,24)
  { ring(){ 2: bond(r); 5: bond(r) atom("SO$_2$Cl",L); }
  }
  arrow(-90)
  {  text(T,L){formula(C,C){atom("NaOH")}}
     text(B,L){formula(C,C){atom("Zn")}}
  }
  formula(T,B)
  { ring(){ 2: bond(r); 5: bond(r) atom("SO$_2$Na",L); }
  };
}
arrow(-90){} % 80%

formula(T,B)
{ ring(){ 2: bond(r);
          5: bond(r) atom("SO$_2$",L,R) bond(30; -30; 30,=)
             branch{bond(90);} bond(-30); }
};

formula(T,B,"3-Methylbutens""aure",V,24)
{ bond(30) branch{ bond(90); } bond(-30,=; 30) atom("COOH",L)
}
arrow(-90){ text(T,L){formula(C,C){atom("CH$_3$OH")}}}
formula(T,B)
{ bond(30) branch{ bond(90); } bond(-30,=; 30) atom("CO$_2$CH$_3$",L)
```

```
  };
}

arrow(-90){ text(T,L){formula(C,C){atom("NaOCH$_3$")}}}

formula(T,B)
{ ring()
  { 3: bond(r);
    0: bond(r) atom("S")
       branch { bond(180,=C) atom("O");
                bond(0,=C) atom("O"); }
       bond(-90)
       saveXY(#1)
       bond(-150) bond(150,=)
         branch { bond(-150); bond(90); }
       restoreXY(#1)
       bond(-30) branch { bond(-150);
                          bond(-30); }
       bond(30)
       branch{ bond(90,s,S) atom("$\ominus$"); }
       bond(-30) atom("CO$_2$CH$_3$",L);
  }
}
bracket()

arrow(-90,10){}
arrow(-90,10){ text(T,L){formula(C,C){atom("KOH")}}}

formula(T,B,"\textit{trans}-Chrysanthemums""aure",V,24)
{ ring(,,H,,3,90)
  { 1: bond(r,<.) atom("COOH",L);
    2: bond(150,<<) bond(-150,=) branch{ bond(-90); bond(150); };
    0: bond(t);
    0: bond(b);
  }
}
\end{chemistry}
```

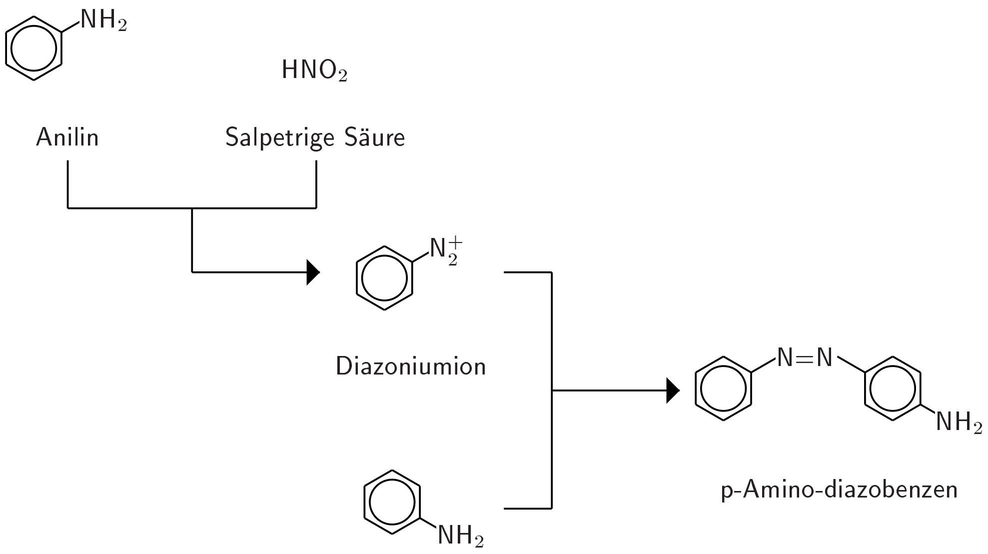
Bei Bedarf können Sie auch mehrfach geschachtelte gemischte Zusammenführungen formulieren, zum Beispiel als horizontales Schema, das ein vertikales Teilschema enthält (Schema 2–15):

```
joinh(2,R)
{ joinv(2,B)
  { formula(T,B,"Anilin",V,24)
    { ring(){ 4: bond(r) atom("N",C,R) atom("H$_2$",L);} }
    ;
    formula(T,B,"Salpetrige S""aure",V,24)
    { atom("HNO$_2$") }
    ;
  }
  arrow(-90,12,-){} nospace arrow(){}
  formula(L,R,"Diazoniumion")
  { ring(){ 4: bond(r) atom("N",C,R) atom("$_2^+$",L);} }
  ;
  formula(L,R)
  { ring(){ 5: bond(r) atom("N",C,R) atom("H$_2$",L);} }
```



Schema 2–14 Synthese der Chrysanthemumsäure, gesetzt mit `joinv`.

```
  ;
}
arrow(){}
formula(L,R,"p-Amino-diazobenzen",HR,24)
{ ring(){ 4: bond(r) atom("N=N",L,R) bond(r/)
             ring(,0,,,,r)
             { 3: bond(r) atom("N",C,R) atom("H$_2$",L);};
        }
}
```

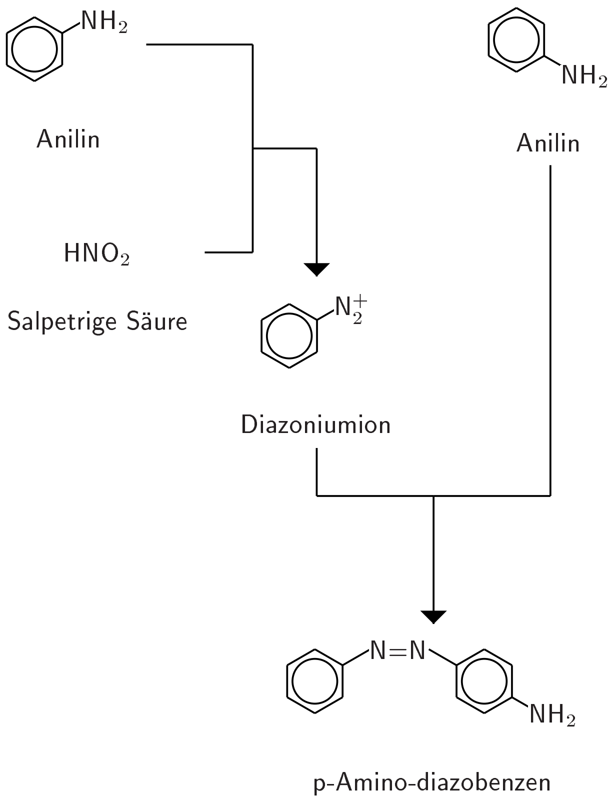


Schema 2–15 Synthese einer Diazoverbindung. Horizontale und vertikale Zusammenfügung gemischt.

Das Pendant mit vertauschten Rollen ist ein vertikales Schema mit horizontalem Anteil (Schema 2–16):

```
\begin{chemistry}[joinvh]
joinv(2,T)
{ joinh(2,L)
  { formula(L,R,"Anilin")
    { ring(){ 4: bond(r) atom("N",C,R) atom("H$_2$",L);} }
    ;
    formula(L,R,"Salpetrige S""aure")
    { atom("HNO$_2$") }
    ;
  }
  arrow(,12,-){} nospace arrow(-90){}
  formula(T,B,"Diazoniumion",V,24)
  { ring(){ 4: bond(r) atom("N",C,R) atom("$_2^+$",L);} }
  ;
  formula(T,B,"Anilin",V,24)
  { ring(){ 5: bond(r) atom("N",C,R) atom("H$_2$",L);} }
  ;
}
arrow(-90){}
formula(T,B,"p-Amino-diazobenzen",V,24)
```

```
{ ring(){ 4: bond(r) atom("N=N",L,R) bond(r/)
             ring(,0,,,,r)
             { 3: bond(r) atom("N",C,R) atom("H$_2$",L);};
        }
}
\end{chemistry}
```



Schema 2–16 Gemischte `joinv`/`joinh`-Befehle.

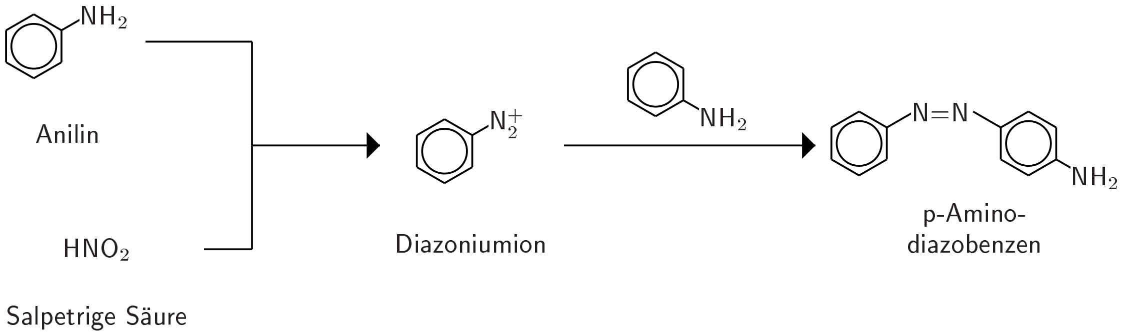
2.3.8 Mischung der Grundelemente

Es ist möglich, die bislang vorgestellten Möglichkeiten zum Satz komplexer Formeleinheiten (mehrzeilige Blöcke und Zusammenführungen) miteinander zu mischen. Die homogene Mischung der Zusammenführung wurde bereits vorgestellt, hier soll gezeigt werden, wie eine Zusammenführung zu Beginn eines einzeiligen `multiline`-Rumpfes eingebettet wird (Schema 2–17):

```
\begin{chemistry}[joinhml]
multiline(1)
{ joinh(2,L)
  { formula(L,R,"Anilin")
    { ring(){ 4: bond(r) atom("N",C,R) atom("H$_2$",L);} }
    ;
    formula(L,R,"Salpetrige S""aure")
    { atom("HNO$_2$") }
```

```
    ;
  }
  arrow(){}
  formula(L,R,"Diazoniumion")
  { ring(){ 4: bond(r) atom("N",C,R) atom("$_2^+$",L);} }

  arrow()
  { text(T,L){formula(L,R)
    { ring(){ 5: bond(r) atom("N",C,R) atom("H$_2$",L);} } } }

  formula(L,R,"\shortstack{p-Amino-\\ diazobenzen}")
  { ring(){ 4: bond(r) atom("N=N",L,R) bond(r/)
              ring(,0,,,,r)
              { 3: bond(r) atom("N",C,R) atom("H$_2$",L);};
          }
  }
;
}
\end{chemistry}
```

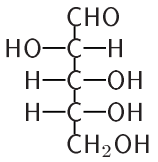
Weitere Zeilen können bei Bedarf der `multiline`-Umgebung hinzugefügt werden.



Schema 2–17 Zusammenführung horizontaler Reaktionsketten und `multiline` gemischt.

2.4 Zyklohexan und Saccharide

In der Literatur sind verschiedene Darstellungen von Sacchariden üblich. Textformeln in der Darstellung nach FISCHER können mit `atom` gesetzt werden, die Verzweigungen nach links und rechts sind mit `branch` nachzubilden:



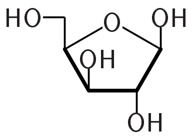
D-Arabinose

```
formula(L,R,"D-Arabinose",HR,24)
{  branch { atom("C",C,R) atom("HO",L); } bond(-90)
   branch { bond(180) atom("HO",R); bond(0) atom("H",L); }
   atom("C",C,C) bond(-90)
   branch { bond(180) atom("H",R); bond(0) atom("OH",L); }
   atom("C",C,C) bond(-90)
   branch { bond(180) atom("H",R); bond(0) atom("OH",L); }
   atom("C",C,C) bond(-90)
   atom("C",C,R) atom("H$_2$OH",L)
}
```

Bei häufigem Einsatz solcher Formeln ist ein Makro empfehlenswert, zum Beispiel:

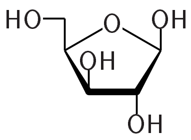
```
\begin{chemspecial}
define(‘TOP_’, ‘branch { atom("C",C,R) atom("$1",L); } bond(-90)’)
define(‘MID_’, ‘branch { bond(180) atom("$1",R);
                         bond(0) atom("$2",L); }
                atom("C",C,C) bond(-90) ’)
define(‘BOTTOM_’, ‘atom("C",C,R) atom("$1",L); }’)
\end{chemspecial}
\begin{chemistry}
  formula(L,R,"D-Arabinose",HR,24)
  {  TOP_(‘HO’)
     MID_(‘HO’, ‘H’)
     MID_(‘H’, ‘OH’)
     MID_(‘H’, ‘OH’)
     BOTTOM(‘H$_2$OH’)
  }
\end{chemistry}
```

Eine schematische räumliche Darstellung (Formel nach HAYWORTH) von Furanosen kann mit dem Ringtyp `furanose` erreicht werden, eventuell mit Angabe einer Bindungsliste zur Verstärkung der perspektivischen Bindungen:



β-D-Xylose

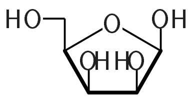
```
formula(C,C,"$\beta$-D-Xylose",HR,24)
{ ring("furanose",0,,L)
  { 0: atom("O");
    1: bond(90) atom("O",C,R) atom("H",L);
    2: bond(-90) atom("O",C,R) atom("H",L);
    3: bond(90) atom("O",C,R) atom("H",L);
    4: bond(90;180) atom("O",C,L) atom("H",R);
  }
}
```



β-D-Xylose

```
formula(C,C,"$\beta$-D-Xylose",HR,24)
{ ring("furanose",0,1<<2b3>>,L)
  { 0: atom("O");
    1: bond(90) atom("O",C,R) atom("H",L);
    2: bond(-90) atom("O",C,R) atom("H",L);
    3: bond(90) atom("O",C,R) atom("H",L);
    4: bond(90;180) atom("O",C,L) atom("H",R);
  }
}
```

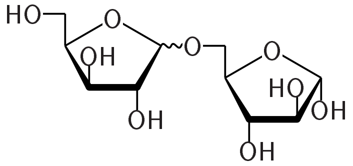
Innerhalb des Ringes kann die Darstellung von *vic*-Hydroxylgruppen in *cis*-Stellung aus Platzgründen problematisch sein. Sie können jedoch das Wasserstoffsymbol mit TEX-Befehlen näher an das Sauerstoffsymbol rücken. Damit Teile des „O“ nicht überdeckt werden, muß beim Setzen des „H“ das Löschen des Hintergrundes abgeschaltet werden:



β-D-Lyxose

```
formula(C,C,"$\beta$-D-Lyxose",HR,24)
{ ring("furanose",0,,L)
  { 0: atom("O");
    1: bond(90) atom("O",C,R) atom("H",L);
    2: bond(90) atom("O",C,L) atom("H\kern-.2em",R,,0);
    3: bond(90) atom("O",C,R) atom("\kern-.2emH",L,,0);
    4: bond(90;180) atom("O",C,L) atom("H",R);
  }
}
```

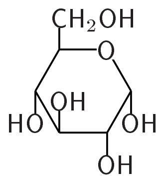
Furanoseringe können wie normale Ringe kombiniert werden, um Oligosaccharide aufzubauen. Im folgenden Beispiel wird der Bindungstyp ˜ verwendet, um ein Anomer zu kennzeichnen:



D-Xylofuranosyl-(1$\to$4)-α-D-arabinose

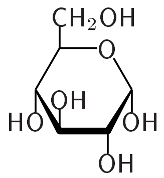
```
formula(C,C,
"D-Xylofuranosyl-(1$\to$4)-$\alpha$-D-arabinose",HR,24)
{ ring("furanose",0,1<<2b3>>,L)
  { 0: atom("O");
    1: bond(0,~) atom("O",C,C) bond(0;-90)
    ring("furanose",4,1<<2b3>>,L)
    { 0: atom("O");
      1: bond(-90) atom("O",C,R) atom("H",L);
      2: bond(90) atom("O",C,L) atom("H",R);
      3: bond(-90) atom("O",C,R) atom("H",L);
    }
    ;
    2: bond(-90) atom("O",C,R) atom("H",L);
    3: bond(90) atom("O",C,R) atom("H",L);
    4: bond(90;180) atom("O",C,L) atom("H",R);
  }
}
```

Auch Pyranosen können in der HAYWORTH-Darstellung wiedergegeben werden. Hierzu ist kein spezieller Ringtyp erforderlich, es kann ein normaler großer Sechsring verwendet werden, eventuell mit Angabe einer Bindungsliste zur Erhöhung des räumlichen Eindrucks:



α-D-Glucopyranose

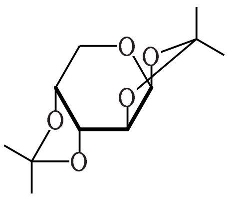
```
formula(L,R,"$\alpha$-D-Glucopyranose",HR,24)
{ ring(,,H,L,,0)
  { 5: atom("O");
    0: bond(-90) atom("O",C,R) atom("H",L);
    1: bond(-90) atom("O",C,R) atom("H",L);
    2: bond(90) atom("O",C,R) atom("H",L);
    3: bond(-90) atom("O",C,L) atom("H",R);
    4: bond(90) atom("C",C,R) atom("H$_2$OH",L);
  }
}
```



α-D-Glucopyranose

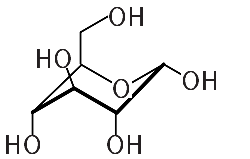
```
formula(L,R,"$\alpha$-D-Glucopyranose",HR,24)
{ ring(,,HO<<1b2>>,L,,0)
  { 5: atom("O");
    0: bond(-90) atom("O",C,R) atom("H",L);
    1: bond(-90) atom("O",C,R) atom("H",L);
    2: bond(90) atom("O",C,R) atom("H",L);
    3: bond(-90) atom("O",C,L) atom("H",R);
    4: bond(90) atom("C",C,R) atom("H$_2$OH",L);
  }
}
```

Mit Hilfe von wilden Linien können Derivate der Saccaride gesetzt werden, wie die 1,2-3,4-Di-O-*sec*-butyliden-β-D-*arabino*-pentopyranose zeigt:

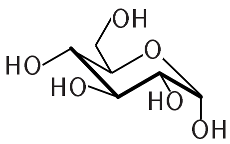


```
formula(L,R)
{ ring(,,HOt1t2t,L,,0)
  { 5: atom("O");
    1: bond(90) atom("O") saveXY(#1);
    0: bond(90) atom("O") bond(20,,L)
         branch{bond(-30); bond(90);} bond(#1);
    2: bond(-90) atom("O") saveXY(#2);
    3: bond(-90) atom("O") bond(-120,,L)
         branch{bond(150); bond(-90);} bond(#2);
  }
}
```

Für die genauere Darstellung der sterischen Verhältnisse der Pyranosen kann der Ringtyp `chair` genutzt werden, der einen Zyklohexanring in Stereodarstellung („Sesselform“) zeigt. Hilfreich sind hierbei die symbolischen Winkelangaben `t` und `b` für axiale und äquatoriale Substituenten. Aus Platzgründen muß die Hydroxylgruppe an Position 5 um 20° aus ihrer normalen `t`-Position gedreht werden:

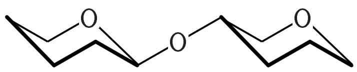



```
formula(L,R)
{ ring("chair",,,L,-1)
  { 0: bond(b) atom("O",C,L) atom("H",R);
    5: bond(20+t) atom("O",C,L) atom("H",R);
    4: bond(b) atom("O",C,R) atom("H",L);
    3: bond(b) atom("O",C,R) atom("H",L);
    2: atom("O");
    1: bond(t; t-) atom("OH",L);
  }
}
```



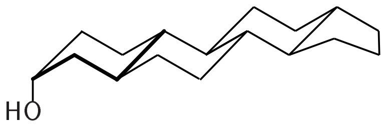

```
formula(L,R)
{ ring("chair",,,L)
  { 0: bond(b) atom("HO",R);
    5: bond(t) atom("HO",R);
    4: bond(b) atom("O",C,L) atom("H",R);
    3: bond(b) atom("O",C,R) atom("H",L);
    2: atom("O");
    1: bond(t; t-) atom("OH",L);
  }
}
```

Das Zyklohexangerüst wurde so konstruiert, daß die 1→6-glykosidische Bindung von Di- und Polysacchariden mit mit einem bond-Befehl doppelt-normaler Länge (NN) oder – mit Ethersauerstoff – zwei aufeinanderfolgenden Bindungen normaler Länge erzeugt werden kann. Der zweite Ring liegt dann auf der gleichen Höhe wie der erste, sodaß Oligosaccharide horizontal verlaufen:



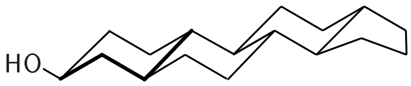

```
formula(L,R)
{ ring("chair",,,L)
  { 2: atom("O");
    3: bond(t,,L) atom("O") bond(t,,L)
       ring("chair",,,L){ 2: atom("O"); };
  }
}
```

Durch die Symmetrie des chair-Gerüstes können mehrere Ringe aneinander gesetzt werden, um zum Beispiel ein Steroidgerüst räumlich zu veranschaulichen. Für den fünfgliedrigen D-Ring müssen Sie eine „wilde" Verbindungslinie ziehen:



3-α-Hydroxysteran

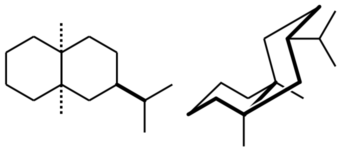
```
formula(L,R,"3-$\alpha$-Hydroxysteran",HR,24)
{ ring("chair",,,L,-1)
  { 0: bond(b)
       atom("O",C,L) atom("H",R);
       vertex("chair",3,0,3-4-5-,-1)
       { vertex("chair",2,5,3-4-5-,-1)
         { 3: bond(b,,L) saveXY(#1);
           4: bond(t,,L; -t|,,L; #1);
         };
       };
  }
}
```



3-β-Hydroxysteran

```
formula(L,R,"3-$\beta$-Hydroxysteran",HR,24)
{ ring("chair",,,L,-1)
  { 0: bond(t)
       atom("O",C,L) atom("H",R);
       vertex("chair",3,0,3-4-5-,-1)
       { vertex("chair",2,5,3-4-5-,-1)
         { 3: bond(b,,L) saveXY(#1);
           4: bond(t,,L; -t|,,L; #1);
         };
       };
  }
}
```

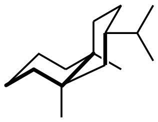
Im Falle des Valerans ergibt sich durch das Verschmelzen zweier Ringkanten mit normalerweise unterschiedlicher Länge eine kleine Unschönheit, da der eine Ring entsprechend vergrößert werden muß, um die Ringkanten anzugleichen:



Valeran

```
formula(L,R)
{ ring(,4,H){}
  ring(,2,H){ 1: bond(t,o);
              2: bond(b,o);
              5: bond(r,t) branch { bond(r+); bond(r-); }; }
}

formula(L,R,"Valeran",HR,24)
{ ring("chair",,,,-1)
  { 3: bond(b);
    vertex("chair",3,0,5p)
    { 4: bond(0) branch{bond(0++); bond(0+-);}; };
    4: bond(-90); }
}
```

Sie können das Formelbild durch manuelle Bearbeitung etwas verbessern:



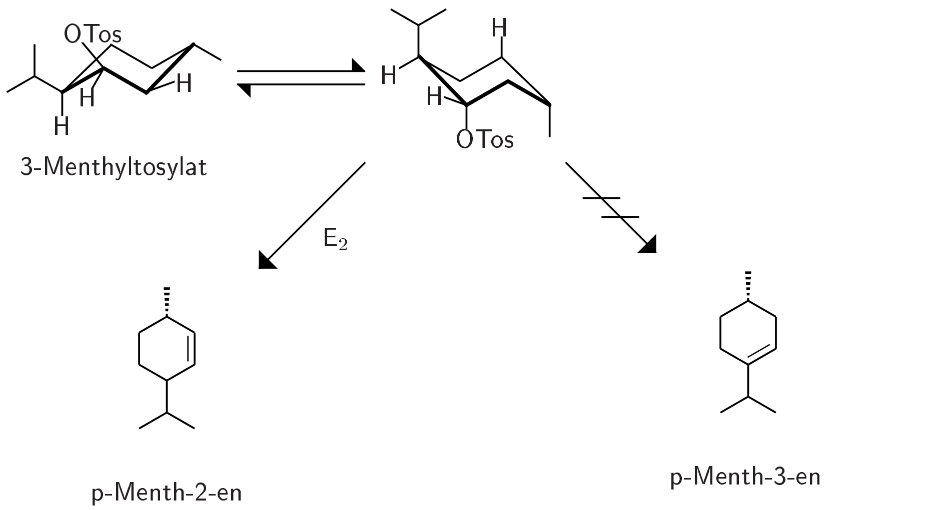
Valeran

```
formula(L,R,"Valeran",HR,24)
{ ring("chair",,,,-1)
  { 3: bond(b);
    4: saveXY(#1);
    3: bond(t; 30; -120)
       saveXY(#2) bond(-90,p; #1)
       restoreXY(#2) bond(0) branch{bond(0++); bond(0+-);};
    4: bond(-90); }
}
```

Der variable Parameter des Zyklohexangerüstes erlaubt es, zwischen den beiden denkbaren Hauptkonformeren auszuwählen. Schema 2–18 kann damit den Verlauf einer Eliminierungsreaktion erklären:

```
\begin{chemistry}[conf]
  formula(L,R,"3-Menthyltosylat",HA,36)
  { ring("chair",,,L,-1)
    { 0: bond(t) branch{ bond(t-); bond(t+);} ;
      0: bond(b) atom("H");
      5: bond(130,,L) atom("O",C,R) atom("Tos",L);
      5: bond(b) atom("H");
      4: bond(t) atom("H",L);
      3: bond(b); }
  }

  arrow(,,<=>){}

  formula(L,BR)
  { ring("chair",,,L)
    { 0: bond(t) branch{ bond(t+); bond(t-);} ;
      0: bond(b) atom("H");
      2: bond(t) atom("H");
      3: bond(b);
      5: bond(b) atom("O",C,R) atom("Tos",L);
      5: bond(t) atom("H"); }
  }
  savecontext(#1)

  arrow(-45,,-|>){}
```

```
  formula(TL,C,"p-Menth-3-en",HA,54)
  { ring(,,H5=){ 0: bond(r) branch { bond(r-); bond(r+); };
                 3: bond(r,<.); }
  }

  setcontext(#1,BL)
  arrow(-135,,->){ text(T,L){ formula(C,C){atom("E$_2$") } } }
  formula(TR,C,"p-Menth-2-en",HA,54)
  { ring(,,H4=){ 0: bond(r) branch { bond(r-); bond(r+); };
                 3: bond(r,<.); }
  }
\end{chemistry}
```



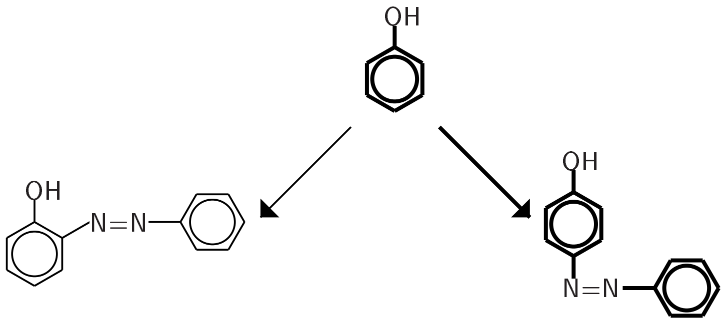
Schema 2–18 Das Gleichgewicht zwischen den beiden Hauptkonformeren des Zyklohexans nutzt die verschiedenen Konformeren des Zyklohexans.

2.5 Änderung von Parametern

Als Beispiel soll im Schema 2–19 der Hauptreaktionsweg durch verstärkte Linien hervorgehoben werden. Die Lösung besteht darin, den Programmcode für diesen Weg in ein `save`/`restore`-Befehlspaar einzuschachteln und innerhalb dieser Klammer den Parameter `$rLW` für die Linienstärke zu ändern. Der seitlich abzweigende Nebenweg darf nicht innerhalb dieser Klammerung notiert werden:

```
\begin{chemistry}[param1]
  save(#1)
  set("rLW", 1.5)
  formula(L,BR)
  { ring(){ 3: bond(r) atom("O",C,R) atom("H",L); } }
  savecontext(#1)
  arrow(-45){}
  formula(L,R)
  { ring(){ 3: bond(r) atom("O",C,R) atom("H",L);
```

```
            0: bond(r) atom("N=N",L,R) bond(0)
               ring(,3,,,,r){}; }
  }

  restore(#1)

  setcontext(#1,BL)
  arrow(-135){}
  formula(R,R)
  { ring(){ 3: bond(r) atom("O",C,R) atom("H",L);
            4: bond(r) atom("N",C,R) atom("=N",L,R) bond(0)
               ring(,3,,,,r){}; }
  }
\end{chemistry}
```



Schema 2–19 Hauptreaktionen können durch höhere Linienbreite von Nebenwegen abgehoben werden. Die Linienbreite wird mit `set` eingestellt.

Wollen Sie alle Parameter mit einem gleichen Faktor multiplizieren, können Sie anstelle mehrerer `set`-Befehle auch →`scale` einsetzen.

2.6 Elektronendarstellungen

Ein wesentlicher Bestandteil vor allem mechanistischer Formeldarstellungen ist die Wiedergabe der Elektronenkonfigurationen und der Verschiebungen von Elektronen.

2.6.1 Elektronenkonfiguration

Die Wiedergabe der Elektronenkonfiguration an einem Atom ist zwar nicht unmittelbarer Bestandteil von OCHEM, sie läßt sich jedoch mit dem LaTeX-Paket `echem.sty`, das in der OCHEM-Distribution enthalten ist, erreichen. Es enthält die beiden Makros `\vd{<Atom>}` und `\vdd{<Atom>}`, um neben dem genannten Atom ein resp. zwei Elektronen vertikal übereinanderzustellen. Die Elektronen werden durch Punkte symbolisiert. Das Atomsymbol dient nur zur Festlegung des Raumes, um den die Elektronen herum angeordnet werden sollen, es wird selber nicht dargestellt und muß, wie die Beispiele zeigen, nach dem Makro folgen. Je nach der Reihenfolge des Atoms und der Makros erhalten Sie die Elektronen vor oder nach dem Atomsymbol:

•Li, Li•, :Be, Be:, :B•

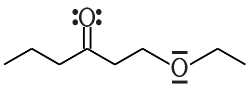
```
\vd{Li}Li,
Li\vd{Li},
\vdd{Be}Be,
Be\vdd{Be},
\vdd{B}B\vd{B}
```

Für eine horizontale Anordnung stehen vier Makros zur Verfügung: für Elektronen oberhalb des Atomsymbols die Makros `\hdu{<Atom>}` und `\hddu{<Atom>}`, für Elektronen unterhalb des Atomsymbols die Makros `\hdl{<Atom>}` und `\hddl{<Atom>}`:

Li, Li, C

```
\hdu{Li}Li,
\hdl{Li}Li,
\hddu{C}\hddl{C}C
```

Für die Einbettung in eine OCHEM-Formel muß das Paket in einer `package`-Anweisung innerhalb einer `chemspecial`-Umgebung sowie im Dokument selber geladen werden:



```
\usepackage{ochem,echem}
\begin{chemspecial}
package("echem")
\end{chemspecial}
...
\begin{chemistry}[elekt]
formula(L,R)
{
   bond(30;-30;30)
   branch { bond(90,=C)
            atom("\vdd{O}O\vdd{O}");
          }
   bond(-30;30;-30)
   atom("\hdl[\echhbar]{O}\hdu[\echhbar]{O}O",L,R)
   bond(30;-30)
}
\end{chemistry}
```

2.6.2 Elektronenverschiebungen

Als wichtiges Detail bei mechanistischen Betrachtungen treten kleine Pfeile innerhalb von Formeln oder -schemata auf, die die Wanderung von Elektronen veranschaulichen. Zwar ist dies fast ein Anwendungsfall für chemische CAD-Programme, ich habe jedoch versucht, einen sinnvollen Befehl für genau diese Elektronenverschiebungen zu implementieren. Da Start- und Zielpunkt nicht einfach durch eine Gerade verbunden werden können, sind einige Parameter zur genauen Beschreibung des Bogens erforderlich. Grundlage des Verschiebungspfeils ist ein kubischer Spline, der neben Start- und Zielpunkt durch die Winkel, die die Tangenten an diesen beiden Punkten bilden, sowie zwei Kontrollpunkte beschrieben wird. Da die Aus- und Einfallswinkel meist feststehen, sind es vor allem die Kontrollpunkte (beziehungsweise deren Entfernung von den Zielpunkten), mit denen der Kurvenverlauf gesteuert werden kann. Hier ist jedoch Erfahrung und Gefühl nötig, um nicht zu verzweifeln, weil die Kurve immer falsch geknickt ist :-)

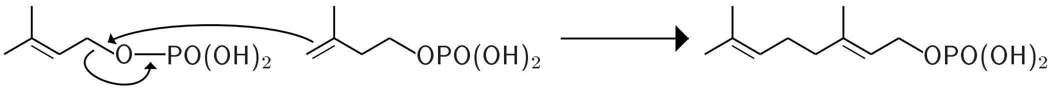
Da die Zielpunkte der Verschiebungsbewegung meist in der Mitte von Bindungen liegt, die Bindungen selber aber in ganzer Länge dargestellt werden sollen, kann zum Speichern dieser mittleren Position, vom Startpunkt der Bindung ausgehend, `saveXY` in seiner erweiterten Syntax mit Winkel und Längenangabe eingesetzt werden. Diese Syntax gestattet es, die Koordinaten des Punktes zu speichern, der erreicht würde, zöge man vom aktuellen Punkt eine Bindung mit dem angegebenen Winkel und der Länge. Da der aktuelle Punkt dabei nicht verschoben wird, kann sofort nach `saveXY` ein `bond`-Kommando folgen. Bei `saveXY` kommen entsprechend der Mittelposition häufig die Kleinbuchstaben als Längenangaben (`n`) in Betracht. Das Beispiel zeigt, wie Sie eine Mittelposition samt Bindung speichern und zeichnen:

```
saveXY(#1,30,n)  % Winkel 30 Grad, halbe Normallaenge
bond(30)        % Winkel 30 Grad, volle Normallaenge
```

Die Nummern, unter der die Punkte gesichert werden, sind Eingabeparameter für `emove`. Start- und Zielpunkt können innerhalb einer Formel liegen, müssen aber nicht, entscheidend ist in jedem Falle der aktuell unter der betreffenden Nummer gespeicherte Punkt (heißt also, daß Sie Ihre `saveXY`-Kommandos konsequent fortlaufend numerieren sollten).

Betrachten Sie als Beispiel den kettenverlängernden Schritt der Isoprenoid-Biosynthese in Schema 2–20. Es werden vier Punkte gespeichert entsprechend einer intramolekularen Verschiebung und einer intermolekularen Elektronenbewegung:

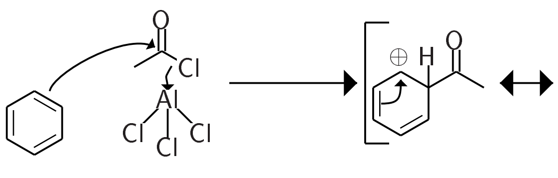
```
formula(L,R)
{ bond(30) branch { bond(90); } bond(-30,=; 30)
  saveXY(#2,-30,n) bond(-30)
  atom("O",L,R) saveXY(#4,0,n) bond(0) atom("PO(OH)$_2$",L)
}
emove(#2,-150,20,#4,-90,20)
space(R)
formula(L,R)
{ saveXY(#3,30,n)
  bond(30,=) branch { bond(90);} bond(-30; 30; -30)
  atom("OPO(OH)$_2$",L)
}
emove(#3,150,20,#2,30,20)

arrow(){}

formula(L,R)
{ bond(30) branch { bond(90); } bond(-30,=; 30; -30; 30)
  branch { bond(90); }
  bond(-30,=; 30; -30) atom("OPO(OH)$_2$",L)
}
```



Schema 2–20 Der kettenverlängernde Schritt der Biosynthese der Isoprenoide wird durch Elektronenverschiebungen verdeutlicht.

Ein weiteres Beispiel, die Friedel-Crafts-Acylierung, ist in Schema 2–21 dargestellt. Auch hier werden intra- und intermolekulare Verschiebungen illustriert. Die `shiftXY`-Befehle dienen dazu, die drei Reaktanden in der richtigen relativen Lage zu positionieren. Durch die Einschachtelung in den Rumpf von `multiline` kann der anschließende Reaktionspfeil auf dieses Formeltriplet als Einheit bezogen werden:

```
multiline(1)
{ formula(C,C)
  { ring(,,H1=3=5=){ 3: saveXY(#1,-30,n); }
  }
  shiftXY(50,30)
  formula(C,C)
  { bond(30) saveXY(#2)
    branch { bond(90,=C) atom("O"); }
    saveXY(#3,-30,n) bond(-30) atom("Cl");
  }
  shiftXY(0,-30)
  formula(C,C)
  { saveXY(#4) atom("Al")
    branch{ bond(-45,,L) atom("Cl");
            bond(-90,,L) atom("Cl");
            bond(-135,,L) atom("Cl"); }
  }
  emove(#1,80,10,#2,150,15)
  emove(#3,-120,10,#4,90,10)
  ;
}

arrow(){}

formula(L,R)
{ ring(,,H1=5=)
  { 1: saveXY(#5,90,n);
    3: saveXY(#6) bond(90,s,n) atom("$\oplus$");
    4: bond(r) branch{ bond(r+,=C) atom("O"); }
       bond(r-);
    4: bond(t) atom("H");
  }
}
emove(#5,0,10,#6,-90,10)
bracket([)

arrow(0,10,<->){}
```



Schema 2–21 Die Elektronenverschiebung zeigt deutlich den Verlauf der Friedel-Crafts-Acylierung eines Aromaten.

2.7 Beispiele

Sie haben nun alle Elemente, die in Formeln verwendet werden können, kennengelernt. Zum Abschluß des Tutorials werden sie kombiniert, um zwei komplexe Reaktionsschemata zu programmieren.

2.7.1 Nylonsynthese

Schema 2–22 zeigt die Darstellung der Ausgangsprodukte für die Nylon/Perlonsynthese. Die Idee dahinter ist, eine möglichst lange Folge von Formeln und Pfeilen zu finden, die einfach beschrieben werden kann und die Zahl der benötigten Verzweigunsstellen minimiert. Hiermit kann die Übersicht gewahrt werden. Ein anderer Ansatz könnte sein, die zusammenhängenden Reaktionsstränge nach chemischen Gesichtspunkten zusammenzufassen, was im vorliegenden Falle jedoch unmöglich ist, da *zwei* Startpunkte existieren: einer beim Benzen und ein anderer beim Butadien. Es wäre nicht möglich, die beiden Stränge in der gezeigten Anordnung zu einer gemeinsamen Formel zu führen. (Eine solche Zusammenfassung kann von `joinh` oder `joinv` geleistet werden, erfordert aber eine komplette Umgestaltung des ganzen Schemas.)

Der Hauptstrang beginnt beim Benzen und endet beim Butadien – aber halt, erfolgen die letzten Schritte nicht „gegen die Fahrtrichtung"? Nicht aus der Sicht des Compilers, für den das Erscheinungsbild eines Pfeiles unerheblich ist. Reaktionen „gegen den Strich" werden eben nicht mit dem `->`-Pfeil, sondern mit dem `<--`-Pfeil formuliert.

Den Richtungsänderungen, die im Verlaufe des Hauptstranges auftreten, wird durch eine entsprechende Positionsparametrisierung der Formeln und der Pfeilwinkel Rechnung getragen.

Die Chlorierung des Butadiens liefert ein Gemisch zweier Dichlorbutene, die mit Hilfe eines einzeiligen `multiline`-Blockes zur Achse der vertikalen Reaktionsfolge zentriert werden können. Das Pluszeichen ist eine verkappte „Formel", die lediglich den Text „+" wiedergibt. Der Name, der unter beiden Formeln zentriert stehen soll, wird durch eine leere Formel erzeugt, die nur die Beschriftung enthält.

Der Hauptstrang präsentiert sich als Programm wie folgt:

```
\begin{chemistry}[nylon]
  formula(L,R,"Benzen",HR,24){ ring(){} }

  arrow()
  { text(T,C){ formula(C,C){ atom("H$_2$/Pt, Al$_2$O$_3$") } }
    text(B,C){ formula(C,C){ atom("36 bar") } }
  }

  formula(L,R){ ring(,,H){} }
  savecontext(#1)
```

```
arrow(,36)
{ text(T,C){ formula(C,C){ atom("O$_2$") } }
  text(B,C){ formula(C,C){ atom("Co(OAc)$_3$") } }
}

formula(L,B)
{ atom("HOOC",L,R) bond(30;-30;30;-30;30;-30)
  atom("COOH", L)
}
fbox

arrow(-90)
{ text(T,C){ formula(C,C){ atom("$\Delta$") } }
  text(B,C){ formula(C,C){ atom("NH$_3$") } }
}

formula(T,B)
{ atom("H$_2$NCO",L,R) bond(30;-30;30;-30;30;-30) atom("CONH$_2$", L) }

arrow(-90){}

formula(T,B)
{ atom("NC",L,R) bond(30;-30;30;-30;30;-30) atom("CN", L) }

arrow(-90)
{ text(T,C){ formula(C,C){ atom("H$_2$/Ni") } }
  text(B,C){ formula(C,C){ atom("130 $^o$C, 240 bar") } }
}

formula(T,L)
{ atom("H$_2$N",L,R) bond(30;-30;30;-30;30;-30) atom("NH$_2$", L) }
fbox savecontext(#2)

arrow(180,,<-){} nospace
arrow(-90,,-)
{ text(T,C){ formula(C,C){ atom("NH$_3$") } }
  text(B,C){ formula(C,C){ atom("H$_2$/Kat") } }
}

formula(T,B)
{ atom("NC",L,R) bond(30;-30;30,=;-30;30;-30) atom("CN", L) }

arrow(-90,,<-)
{ text(T,C){ formula(C,C){ atom("CuCN") } }
  text(B,C){ formula(C,C){ atom("CN$^\ominus$") } }
}

multiline(1,C,T,B)
{ formula(L,R)
  { atom("Cl") bond(30;-30;30,=;-30;30) atom("Cl")
  }
  formula(L,R){atom("+") }
  formula(L,R)
  { bond(30,=;-30) branch { bond(-90) atom("Cl"); }
    bond(30;-30) atom("Cl")
  };
```

```
  }
  formula(T,B,"Chlorbuten",V,12){}

  arrow(-90,,<-)
  { text(T,C){ formula(C,C){ atom("Cl$_2$") } }  }

  formula(T,B,"Butadien",V,24)
  { bond(30,=; -30; 30,=)   }
```

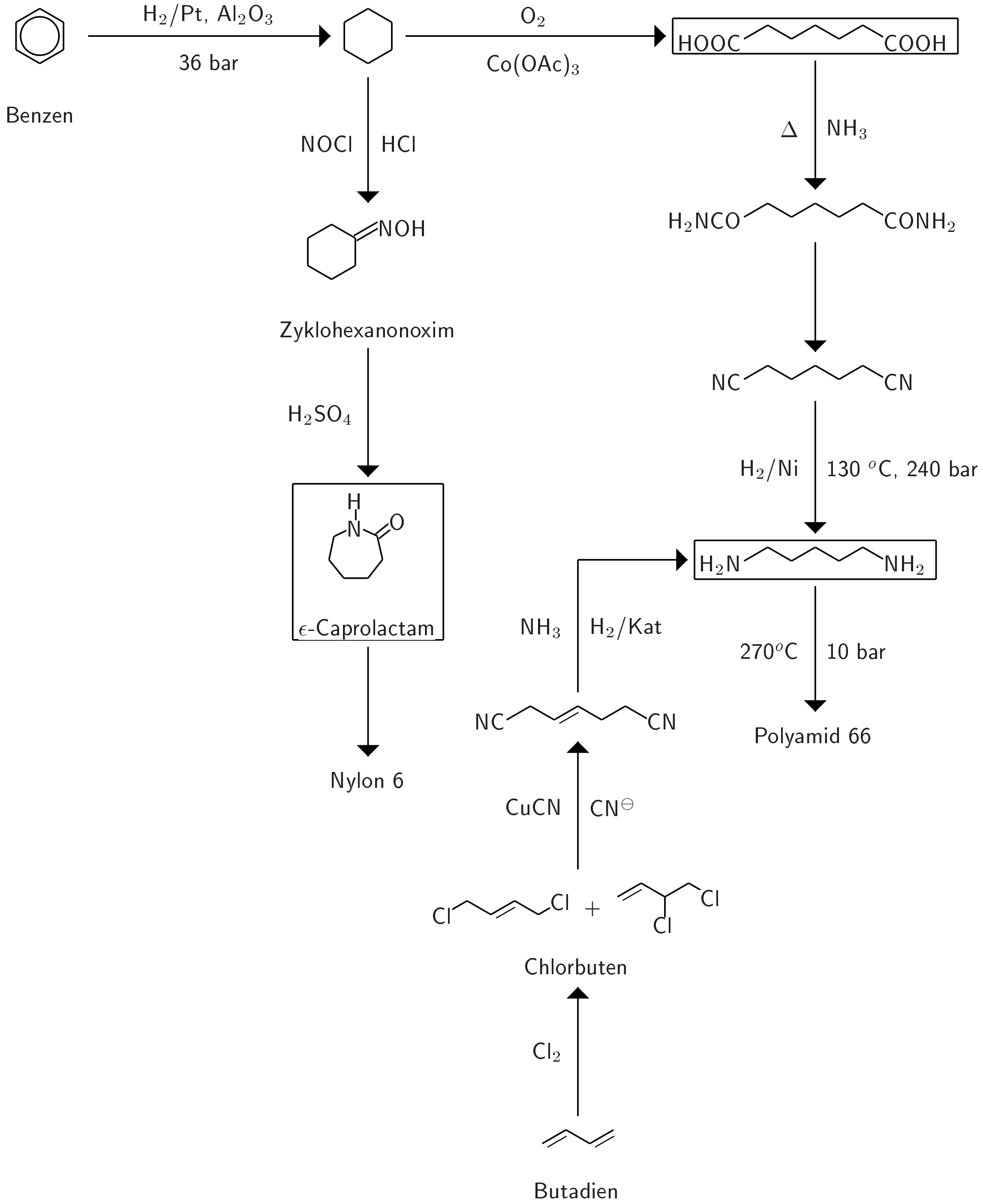
Durch die Verwendung eines Hauptstranges verbleiben insgesamt zwei Verzweigungen, die zu den Endprodukten führen und keine Besonderheiten aufweisen:

```
  setcontext(#1,B)
  arrow(-90)
  { text(T,C){ formula(C,C){ atom("NOCl") } }
    text(B,C){ formula(C,C){ atom("HCl") } }
  }

  formula(T,B, "Zyklohexanonoxim",V,24)
  { ring(,,H){ 4: bond(r,=C) atom("NOH",L); }  }

  arrow(-90)
  { text(T,C){ formula(C,C){ atom("H$_2$SO$_4$") } }  }

  formula(T,B,"$\epsilon$-Caprolactam",V,24)
  { ring(,,H,,7,90)
    { 0: atom("N") bond(r) atom("H");
      1: bond(r,=C) atom("O"); }
  }
  fbox

  arrow(-90){}

  formula(T,B){ atom("Nylon 6")  }

  setcontext(#2,B)
  arrow(-90)
  { text(T,C){ formula(C,C){ atom("270$^o$C") } }
    text(B,C){ formula(C,C){ atom("10 bar") } }
  }

  formula(T,B){ atom("Polyamid 66")  }
\end{chemistry}
```

2.7.2 Terpen-Biosynthese

Die Biosynthese der Hauptvertreter bizyklischer Monoterpene ist als weiteres komplexes Beispiel vorgesehen und in Schema 2–23 dargestellt. Es enthält Verzweigungen, verschiedene Ringgerüste sowie fett gedruckte Formeln.

```
  save(#1)       % save parameter
  formula(L,R)
  { bond(30) saveXY(#1) bond(90,=; 150)
```



Schema 2–22 Die Nylonsynthese als Beispiel für ein komplexes Schema.

```
  bond(90; 30) saveXY(#2) bond(-30,=)
  bond(-90; -45,s,n) atom("$\oplus$")
  restoreXY(#1) bond(-30)  restoreXY(#2) bond(90)
}

arrow(){}

formula(L,R)
{ ring(,,H3=)
  { 0: bond(r) saveXY(#1) bond(r-)
       restoreXY(#1) bond(r+)
       restoreXY(#1) bond(r,s,n) atom("$\oplus$");
    3: bond(r); }
}
savecontext(#1)

arrow(){}
set("rLW",1.5)
formula(L,R,"Terpineol",HR,24)
{ ring(,,H3=)
  { 0: bond(r)
       branch{ bond(r-); bond(r+); }
       bond(r) atom("O",C,R) atom("H",L);
    3: bond(r); }
}
restore(#1)

% to thujene
setcontext(#1,B) arrow(-90,60){}
formula(T,R)
{ ring(,,H3=)
  { 0: bond(r) branch { bond(r-); bond(r+); };
    0: bond(r,s) atom("$\oplus$");
    3: bond(r); }
}

arrow(){}
formula(L,R)
{ ring(,,H)
  { 0: bond(r) branch { bond(r-); bond(r+); };
    3: bond(r);
    3: bond(45,s) atom("$\oplus$");
    4: saveXY(#2);
    0: bond(#2); }
}

arrow(){}
set("rLW",1.5)
formula(L,R,"3-Thujen",HR,24)
{ ring(,,H2=)
  { 0: bond(r) branch { bond(r-); bond(r+); };
    3: bond(r);
    4: saveXY(#2);
    0: bond(#2); }
}
restore(#1)
```

```
% to pinene
setcontext(#1,TR) arrow(60,36){}

formula(L,R)
{ ring(,,H)
  { 3: bond(r);
    3: bond(45,s) atom("$\oplus$");
    0: bond(90) branch { bond(30); bond(90,,n);
                         bond(-150,,n);};
  }
}

formula(L,R)
{ ring("bc311h")
  { 0: bond(t); 0: bond(b);
    1: bond(r); }
}

arrow(){}
set("rLW",1.5)
formula(L,R,"$\beta$-Pinen",HR,24)
{ ring(,,H)
  { 3: bond(r,=C);
    0: bond(90) branch { bond(30); bond(90,,n);
                         bond(-150,,n);};
  }
}
restore(#1)

% to camphene
setcontext(#1,T)
arrow(90,60){}

formula(B,R)
{ ring("bc221h")
  { 4: bond(r);
    3: bond(r,s,n) atom("$\oplus$");
    6: bond(t); 6: bond(b); }
}

arrow(){}
set("rLW",1.5)
formula(L,R,"Campher",HR,24)
{ ring("bc221h")
  { 4: bond(r);
    3: bond(r,=C) atom("O");
    6: bond(t); 6: bond(b); }
}
restore(#1)

% to carene
setcontext(#1,BR) arrow(-60,36){}

formula(L,R)
{ ring(,,H)
  { 3: bond(r);
    0: bond(t) saveXY(#1) bond(90; 45,s) atom("$\oplus$")
```

```
        restoreXY(#1) bond(-90) restoreXY(#1) bond(30);
  }
}

arrow(){}
set("rLW",1.5)
formula(L,R,"3-Caren",HR,24)
{ ring(,,H2=)
  { 3: bond(r);
    0: bond(t) saveXY(#1) bond(90)
        restoreXY(#1) bond(-90) restoreXY(#1) bond(30);
  }
}
restore(#1)
```

2.8 Weitere Verbindungsklassen

Viele Verbindungsklassen wurden in diesem Tutorial nicht behandelt. Die meisten von ihnen können aber mit den besprochenen Mitteln erzeugt werden und bedürfen daher keiner Diskussion. Die Arbeit mit immer gleichen oder sehr ähnlichen Formeln kann jedoch erheblich erleichtert werden, wenn man sich einen Satz von Makros schreibt, die auf diesen Ähnlichkeiten basieren. Anregungen für solche Makros erhalten Sie im Abschnitt 3.6. Dort werden folgende Klassen von Stoffen näher angesprochen:

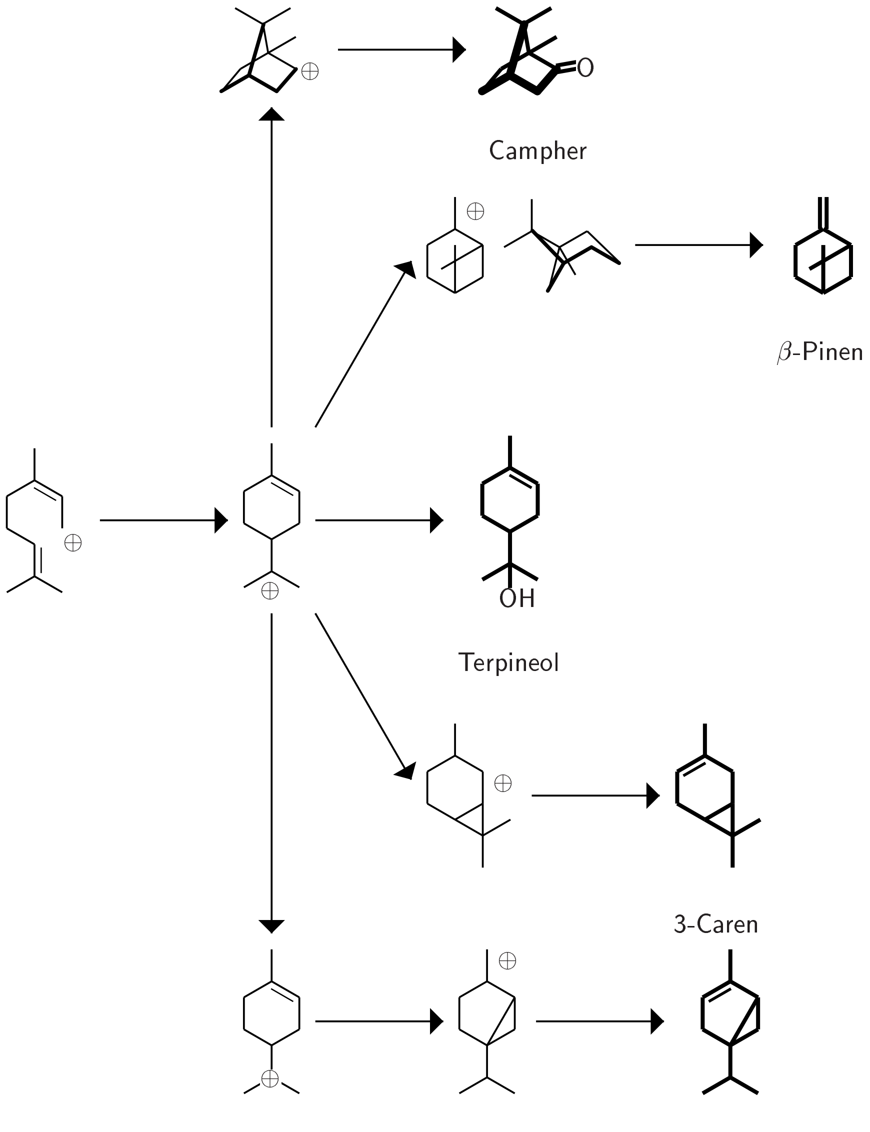
- Steroide
- Isoprene
- Aminosäuren und Oligopeptide

2.9 Entwicklung eigener Ringstrukturen

Um den Compiler nicht mit einer Vielzahl von denkbaren Ringstrukturen, die nur selten benötigt werden, zu belasten, wurden nur die unter `ring` aufgelisteten Grundstrukturen in das Core aufgenommen. Es ist jedoch möglich, in einem Perlmodul eigene Ringstrukturen zu definieren und dieses mit Hilfe von `require` in dem Dokument, das diese Strukturen benötigt, nachzuladen.

Sie finden als Beispiel für den Aufbau eines solchen Moduls die beiden Dateien `mncyclib.pm` und `bicyclib.pm`, die über den Grundwortschatz hinaus mono- und bizyklische Gerüste bereitstellen. Sie können diese Dateien, deren Listings in den Abschnitten **??** und **??** abgedruckt sind, als Templates benutzen; was Sie im wesentlichen in einer Strukturdefinition bereitstellen müssen, sind

- die Koordinaten der einzelnen Atome im Array `@$XYa`,
- die Bindungstypen (einfache, doppelte, projektive) im Array `@$BT`,



Schema 2–23 Die Biosynthese der bizyklischen Monoterpene aus der geradkettigen Vorstufe Farnesol.

- eine Liste von Atomen, die als Startpunkt für Bindungen dienen, im Array @$BLa,
- eine Liste mit den Winkeln der Bindungen im Array @$rThetaa,
- eine Liste mit den Bindungslängen im Array @$rLena,
- die Anzahl der Atome im Gerüst in $$iAnz,
- die Ringgröße, im allgemeinen die Länge einer typischen Ringkante in $$rLen,
- Information, ob ein Aromatenring gezeichnet werden soll, in $$bAromat sowie sein Radius in $$rRadius,
- gegebenenfalls Werte für die beiden freien Parameter.

Für jedes Gerüst müssen Sie zwei Funktionen bereitstellen:

<type>_default_ : Diese Funktion gibt die Defaultwerte vor, die der Befehl ring nach außen bereitstellt. Hierzu zählen vor allem die Anzahl der Atome und die Liste mit den Bindungstypen, da diese von einer eventuell folgenden Bindungsliste überschrieben werden können sollen.

<type>_ : Diese Funktion enthält die eigentliche Definition des Gerüstes. Alle Parameter, auch die beiden variablen Parameter <p1> und <p2> sind bekannt und können benutzt werden.

Das Array @$XYa enthält die Koordinaten der Atome in Punkt (pt), die das Gerüst aufbauen. Sie können diese auf einen beliebigen Nullpunkt beziehen und entweder frei angeben oder – für geometrische Gerüste – berechnen lassen. Versuchen Sie dabei nach Möglichkeit, den Längenparameter in die Koordinaten einfließen zu lassen, damit Ihre Struktur in der Größe skalierbar ist! In den Grundgerüsten bezeichnet er die Länge einer Kante und bestimmt damit die Gesamtgröße.

Die Liste @$BLa enthält eine Folge von Nummern der Atome, von denen Bindungen ausgehen. Die i-te Bindung ist mit der i-ten Atomnummer in dieser Liste verknüpft und wird durch die Daten beschrieben, die jeweils an i-ter Stelle in den einzelnen Eigenschaftenlisten stehen. Monozyklische Systeme werden durch eine einfache Liste @$BLa beschrieben, die aus den Zahlen von Null bis (Anzahl Atome−1) besteht, analog enthalten die Eigenschaftenlisten soviele Einträge wie Atome und damit Bindungen zwischen ihnen vorhanden sind. Die Liste @$BT ist einfach aufzubauen und enthält einfach die den Bindungstypen entsprechenden Nummern (können in streambuf.pm bei der Funktion GetBond nachgelesen werden). Die anderen beiden Listen können unter Umständen Schwierigkeiten bei der erforderlichen Berechnung der Winkel und Längen bereiten. Sie können jedoch die Funktionen

```
getangle(<p1>, <p2>)
getlength(<p1>, <p2>)
```

benutzen, um Winkel und Länge einer Bindung zwischen den Atomen mit den Koordinaten <p1> und <p2> zu bestimmen, wobei der Startpunkt der Bindung am Atom <p1> liegt.

```
getpos(<p1>, <angle>, <len>)
```

kann dagegen den Endpunkt einer Bindung vom Punkt `<p1>` liefern, die eine gegebene Richtung und Länge hat. Je nachdem, ob Sie die Koordinaten der Gerüstatome kennen oder es anhand bekannten Bindungswinkel und -längen aufbauen, kommen Sie mit einem der beiden Funktionensätze rasch zu den fehlenden Daten.

3. Alphabetische Befehlsreferenz

Dieses Kapitel enthält die komplette Beschreibung aller Befehle und ihrer Syntax. Für die einzusetzenden Werte gilt:

- Angaben wie Bindungstypen oder -längen werden durch Symbole festgelegt, deren Aussehen der späteren Erscheinung möglichst ähnlich ist.
- Alle Winkel werden in Altgraden gemessen. Drehsinn ist der mathematisch negative (Uhrzeigersinn), Bezugsachse die positive x-Achse.
- Absolut angebbare Längen werden in Punkt (pt) *ohne* Einheit notiert.

Die meisten Parameter der Befehle sind optional und werden bei Auslassung mit einem Defaultwert belegt, der dem häufigsten Anwendungsfalle entspricht und die Schreibarbeit beträchtlich erleichern kann. Nach dem letzten konkret angegebenen Parameter kann bereits die schließende Klammer gesetzt werden, Notation der restlichen Parameter oder der trennenden Kommata ist unnötig. Soll ein später in der Liste auftretender Parameter angegeben werden, ist es zur Kenntlichmachung seiner Position erforderlich, alle vorhergehenden Argumente ebenfalls zu notieren. Möchten Sie keine Werte einsetzen, ist die Angabe der Kommata ausreichend.

Es bedeuten in der Syntaxbeschreibung:

```
<fliste> := atom | bond | branch | orbital | saveXY | ring |
            save | restore | set

<sliste> := arrow | formula |
            savecontext | setcontext | gotoXY | shiftXY |
            emove |
            save | restore | set | scale
            bracket | fbox
```

3.1 Benutzerbefehle

Der folgende Abschnitt listet alle Befehl auf, die innerhalb der `chemistry`-Umgebung (Einsatz im LaTeX-Dokument) beziehungsweise des `schema`-Befehls (manuelle Erstellung des Programmfiles) auftreten können.

3.1.1 arrow

```
arrow([ [<phi>] [, [<len>] [, [<type]]]])
{ [ text(T, <Tpos>){ <fliste> } ]
  [ text(B, <Tpos>){ <fliste> } ]
}
[ nospace ]

<Tpos> := L | C | R

Defaults: <phi>  = 0
          <len>  = rArrowExtend
          <type> = 1
```

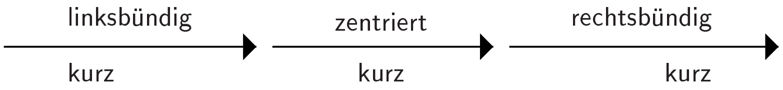
Mit diesem Kommando erzeugen Sie Reaktionspfeile, mit denen die einzelnen Formeln zu Reaktionsketten verbunden werden. Die Gestalt des Pfeils wird von den Parametern <phi>, <len> und <type> bestimmt. Die Richtung des Pfeils wird mit den Winkel <phi> in Altgrad und im mathematisch positiven Sinne gegenüber der positiven x-Achse gemessen.

<type> legt die Art des Pfeils fest (Gleichgewichtspfeil, „führt zu"-Pfeil, durchgestrichener Pfeil) und wird durch ein Symbol gemäß der folgenden Aufstellung spezifiziert:

Der Parameter <len> modifiziert die Länge des Pfeils und gibt die Strecke an, die nach dem Start und vor der Spitze auf jeden Fall existieren muß. Die Mindestlänge eines Pfeils beträgt somit 2<len>. Die Angabe dieser Strecke über die halbe Länge wird verständlich, wenn man die Möglichkeit, den Pfeil zu beschriften, berücksichtigt. Die Gesamtlänge eines Pfeiles bestimmt sich nämlich aus dem Anteil <len> vor der Beschriftung, der Länge der Beschriftung selbst und einem folgenden Anteil <len>.

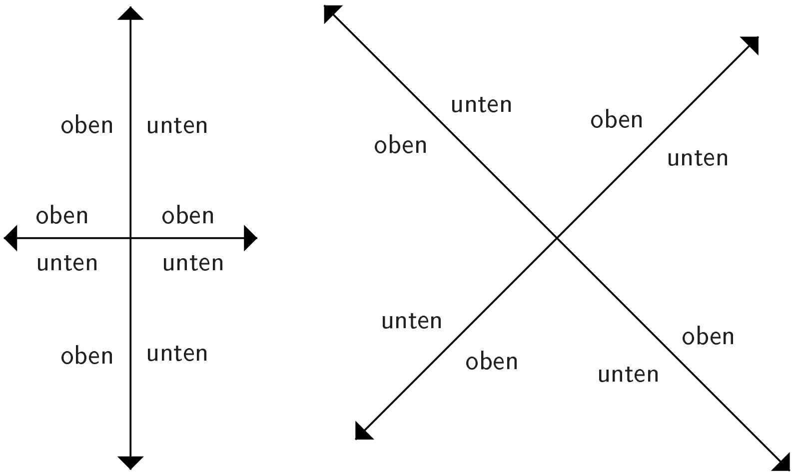
Die Beschriftung kann sowohl oberhalb wie auch unterhalb des Pfeils auftreten. Dem wird durch die Positionierungsparameter T und B im Befehl text Rechnung getragen. Da als Beschriftung sowohl Texte wie auch Formeln möglich sind, wird anstelle von <fliste> eine formula-Beschreibung oder eine Reaktionssequenz erwartet. Normale Texte müssen daher mit dem Befehl atom innerhalb einer Formeln gesetzt werden. Einige einfache Beispiele mögen dies verdeutlichen:

Der Parameter <Tpos> wird bedeutsam, wenn Sie Formeln oder Texte sowohl über als auch unter den Pfeil setzen wollen. In diesem Falle können Sie mit den Werten L, C und R beide Beschriftungen linksbündig zueinander, zentriert oder rechtsbündig ausrichten, wie dies im Schema 3–1 zu sehen ist.



Schema 3–1 Anordnung der oberen und unteren Beschriftung mit verschiedener Ausrichtung.

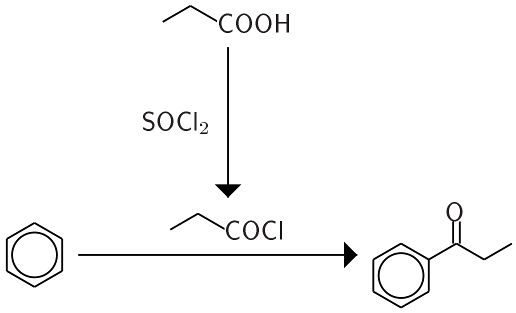
Im Schema 3–2 sehen Sie, wie die Beschriftung bei verschiedenen Winkeln des Pfeils positioniert wird.



Schema 3–2 Die Anordnung der oberen und unteren Beschriftung bei verschiedenen Winkeln des Pfeils.

Wie bereits angesprochen, können anstelle einfacher Texte auch Reaktionsschemata auf einem Pfeil aufgesetzt werden, wie das Beispiel des Schemas 3–3 zeigt, in dem neben der Hauptreaktion auch die Darstellung des Reagenzes skizziert ist:

```
\begin{chemistry}[arrow2]
  formula(L,R)
  { ring(){} }

  arrow()
  { text(T,L)
    { formula(T,B){ bond(30; -30)atom("COOH",L) }
      arrow(-90)
      { text(T,L) { formula(C,C){ atom("SOCl$_2$") } } }
      formula(T,B){ bond(30; -30)atom("COCl",L) }
    }
  }

  formula(L,R)
  { ring()
    { 4: bond(r) branch { bond(r+,=C) atom("O"); }
         bond(r-; r); }
  }
\end{chemistry}
```



Schema 3–3 Komplette Reaktionsketten können als „Beschriftung“ auf Pfeile aufgesetzt werden.

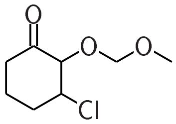
Normalerweise wird nach einem Pfeil in der Richtung des Fortsetzungspunktes ein kleiner Zwischenraum gesetzt, um das Aneinanderpicken der einzelnen Elemente zu vermeiden. Ist dies aber beabsichtigt, können Sie unmittelbar nach `arrow` den Befehl `nospace` einfügen.

3.1.2 atom

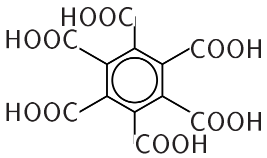
```
atom("<Text>" [, [<pos>] [, [<Cpos>]]])

<pos> := C | L | R | T | B | TL | TR | BL | BR
<Cpos> := C | L | R | T | B | TL | TR | BL | BR

Defaults: <pos>  = C
          <Cpos> = C
```

Mit diesem Befehl können Sie an der aktuellen Position einen beliebigen TEX-Text einfügen, der gemäß der Positionierungsvorgabe `<pos>` zum aktuellen Punkt ausgerichtet wird. Der aktuelle Punkt kann durch die Dimension der Textbox beeinflußt werden, die Positionierung des Fortsetzungspunkte wird durch den Parameter `<Cpos>` bestimmt.

Nachfolgend sind einige Formeln dargestellt, in denen die verschiedenen Möglichkeiten, Atomsymbole zu verwenden, illustriert sind. Im einfachsten Falle, einem terminalen Atom, können Sie die Positionierungen `C` verwenden, um das Symbol zentriert zum Endpunkt der Bindung zu setzen. Bei längeren Texten wie COOH ist es empfehlenswerter, je nach Richtung der Bindung eine links- oder rechtsbündige Anordnung vorzunehmen. Da in beiden Fällen keine Fortsetzung von diesem Atom ausgeht, ist die Angabe `<Cpos>` unerheblich und kann fortgelassen werden. Erst wenn von diesem Atomsymbol aus weitere Bindungen abzweigen, gewinnt dieser Parameter seine Bedeutung, wie Sie an der Methylengruppe sehen, die von rechts erreicht wird (`<pos>` ist `R`) und von der aus eine Bindung nach links weiterführt (`<pos>` ist `L`):

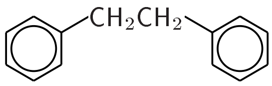



```
formula(L,R)
{ ring(,,H)
  { 3: bond(r,=C) atom("O");
    4: bond(r) atom("O") bond(r-; r)
       atom("O") bond(r-);
    5: bond(r) atom("Cl");
  }
}
```




```
formula(L,R)
{ ring()
  { 0: bond(r) atom("COOH", L);
    1: bond(r) atom("HOOC", R);
    2: bond(r) atom("HOOC", BR);
    3: bond(r) atom("HOOC", R);
    4: bond(r) atom("COOH", L);
    5: bond(r) atom("COOH", TL);
  }
}
```

Sie sehen, daß lange Textstücke wie „COOH“ unschön plaziert werden, im Tutorial auf Seite 16 können Sie erfahren, wie Sie in solchen Fällen besser vorgehen.




```
formula(L,R)
{ ring()
  { 2: bond(r) atom("CH$_2$CH$_2$", R, L)
       bond(-150) ring(,4){};
  }
}
```

Da die Texte und Atomsymbole durch TEX gesetzt werden, sind Sie nicht auf Buchstaben beschränkt, sondern können auch mathematische Formatierungshilfsmittel einsetzen, um Indizes, die Symbole $\oplus$ und $\ominus$ sowie griechische Zeichen zu erhalten. Auch benutzerdefinierte Befehle können als Text auftreten, in diesem Fall müssen die Definitionen allerdings in einer Paketdatei gesammelt werden. Das Vorgehen ist in Abschnitt 3.2.2 beschrieben.

3.1.3 bond

```
bond(<phi> [, [<btype>] [, [<len>]]])
bond([<phi>] [+|-] r|t|b|l|v [+|-] [, [<btype>] [, [<len>]]])
bond(#<n> [, [<btype>] ]])

bond(<args1...>; <args2...>; ... )
<btype> := - -> <- <-> <. << 3 t s >> >. p b ~
<len> := S | N | L | s | n | l [...]

Defaults: <btype> = 0
          <len>   = N
```

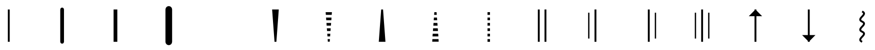
Dieses Kommando erzeugt das grundlegende graphische Primitiv „Bindung“. Die Syntaxvarianten decken verschiedene Anwendungsfälle ab.

Normalform

Sie können dabei mit dem Parameter `<phi>` den Winkel der Bindung gegenüber der positiven x-Achse vorgeben, mit Hilfe von `<btype>` zwischen mehreren Bindungsformen wählen sowie mit `<len>` die Länge der Bindung festlegen. Die Bindung beginnt am aktuellen Startpunkt und verschiebt den aktuellen Endpunkt, der zum folgenden Startpunkt wird, um ihre eigene Länge und Richtung, sodaß Sie mehrere `bond`-Befehle aneinanderhängen können, um Atomketten zu bilden oder einfach – in der komprimierten Syntax – mehrere Beschreibungen mit Semikola verketten können.

Bindungstypen

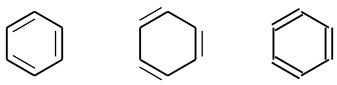
In allen Syntaxvarianten werden die verschiedenen Typen von Bindungen gemäß der folgenden Aufstellung mit Hilfe von Symbolen für `<btype>` ausgewählt (alle Bindungen sind mit einem Winkel von 90º gezeichnet):



`-  t  p  b  s  <<  <.  >>  >.  o  =C  =U  =  3  ->  <-  ~`

- : Diese Bindungsform entspricht der normalen Bindung, die meisten in den Formeln verwendet wird.

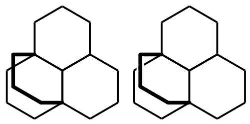
=, =C, =U : Doppelbindungen können mit drei verschiedenen Typen gesetzt werden, die sich in der Lage der einzelnen Bindungen unterscheiden. Die normale Form ist `=` und wird eingesetzt, wenn die zweite Bindung im Uhrzeigersinn rechts von der Hauptbindung liegen soll. Innerhalb von (Benzen-) Ringen ist dies die Form, die ein ausgewogenes Aussehen aufweist, da die kleineren zweiten Bindungen innerhalb des Ringes liegen. `=U` erzeugt eine zweite Bindung links der Hauptbindung und wird innerhalb von Ringen benutzt, die gegen den Uhrzeigersinn gezeichnet werden, damit die Bindung wiederum im Inneren des Ringes liegt. `=C` schließlich zeichnet zwei gleichwertige Bindungen, die zentriert zu einer gedachten Mittellinie liegen. Sie eignet sich nicht für Ringbindungen, wird aber benutzt, wenn eine Doppelbindung zu einem terminalen Atomsymbol führt. Die Unterschiede der einzelnen Formeln werden in den folgenden Formeln sichtbar:



```
gotoXY(0,0)
  formula(C,C){ ring(,,H0=2=4=){} }
gotoXY(50,0)
  formula(C,C){ ring(,,H0=U2=U4=U){} }
gotoXY(100,0)
  formula(C,C){ ring(,,H0=C2=C4=C){} }
```

3 : Diese Form entspricht einer Dreifachbindunng.

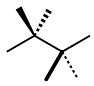
t, p, << : Hiermit wird eine dickere Bindung (t), eine dickere Bindung mit weißem Rand (p) oder eine sich verbreiternde Bindung (<<) gezeichnet. Die Formen t und << eignen sich gleichermaßen zur Darstellung von Bindungen aus der Papierebene heraus, wie das Beispiel zum nächsten Stichwort zeigt. Bindungen mit t sind darüberhinaus in polyzyklischen Systemen geeignet zur Darstellung der räumlich aus dem Papier ragenden Verbrückungen. Zur Erhöhung dieses perspektivischen Eindrucks zeichnet p noch einen weißen Rand um die breitere Bindung. An der folgenden Verbindung ist dies dargestellt, wobei p nur bei den Bindungen eine Wirkung zeigt, die andere Bindungen schneiden:



```
formula(L,R)
{ ring(,4,H){}
  ring(,2,H){}
  ring(,0,H)
  { 1: bond(180,t,S; -90,t; -30,t; 0,t,S); }
}
formula(L,R)
{ ring(,4,H){}
  ring(,2,H){}
  ring(,0,H)
  { 1: bond(180,t,S; -90,p; -30,t; 0,t,S); }
}
```

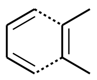
Wenn Sie diesen Trick anwenden, müssen Sie beachten, daß die *Reihenfolge* der Bindungen entscheidend ist, Bindungen mit Aussparungen (Typ p) müssen immer *zuletzt* gezeichnet werden! Im obigen Beispiel darf die Reihenfolge, in der die Ringe gezeichnet werden, nicht geändert werden, wenn die perspektivische Wirkung erhalten bleiben soll.

<., o : Diese Bindungsformen entsprechen einer gestrichelten, sich verbreiternden Bindung beziehungsweise einer gestrichelten Bindungen und eignen sich alle zur Darstellung von Bindungen in die Papierebene hinein:



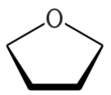
```
formula(C,C)
{ bond(30)
  branch { bond(120,<<); bond(60,<.); }
  bond(-30)
  branch { bond(-120,t); bond(-60,o); }
  bond(30)
}
```

Die fein gepunkteten Linie o eignet sich auch dazu, Bindungen in statu nascendi oder Wasserstoffbrückenbindungen anzudeuten:



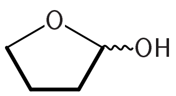
```
formula()
{ ring(,,H0=2=4=U3o5o)
  { 4: bond(r);
    5: bond(r); }
}
```

>>, >., b : Diese Bindungsformen entsprechen sich verjüngenden ausgefüllten, sich verjüngenden gestrichelten oder breiten Bindungen und eignen sich zur Darstellung von Bindungen eines Atom über der Papierebene in diese hinein. Die Breite der Bindung vom Typ `b` (einstellbar mit dem Parameter `rBW`) entspricht der Maximalbreite der sich verbreiternden oder verjüngenden Bindungen, sodaß alle drei Formen leicht kombiniert werden können, ein Beispiel ist die räumliche Wiedergabe nach HAYWORTH:



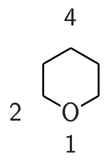
```
formula(C,C)
{ ring("furanose",0,1<<2b3>>,L)
  { 0: atom("O");
  }
}
```

˜ : Diese Bindungsformen wird benutzt, wenn die Stereochemie an einem Atom nicht weiter spezifiziert werden soll oder ein Anomerengemisch vorliegt, wie ein Beispiel aus der Zuckerchemie zeigt:



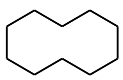
```
formula(C,C)
{ ring("furanose",,,L)
  { 0: atom("O");
    1: bond(0,~) atom("OH",L);
  }
}
```

s : Diese unsichtbare Bindung kann benutzt werden, um Punkte in der Umgebung eines Atoms zu erreichen. Ein Beispiel wäre die Numerierung eines Atomgerüstes:



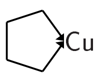
```
formula(C,C)
{ ring(,,H)
  { 0: atom("O");
    0: bond(r,s) atom("1");
    1: bond(r,s) atom("2");
    3: bond(r,s) atom("4"); }
}
```

Weiterhin können in einem Ringgerüst einige Bindungen ausgespart werden, wie etwa das Zyklodekan zeigt:



```
formula(C,C)
{ ring(,4,H4s)
  { }
  ring(,2,H1s)
  { }
}
```

->, <- : Diese Bindungsform eignet sich zur Darstellung von Komplexbindungen mit Angabe des Atoms, das das Elektronenpaar bereitstellt:



```
formula(C,C)
{ ring(,,H0<-4->,,5,0)
  { 0: atom("Cu", L); }
}
```

Bindungslängen

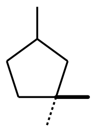
Die Länge der Bindung wird durch die Buchstaben `S` (short), `N` (normal) und `L` (long) kodiert. Ein Kleinbuchstabe kodiert eine Länge, die halb so groß wie die des entsprechenden Großbuchstabens ist. Jeder höhere Buchstabe bedeutet eine um 50% erhöhte Länge. Das mehrfache Erscheinen eines Buchstabens bedeutet, daß die Bindung um ein entsprechendes Vielfaches der Länge dieses Buchstabens verlängert wird, beispielsweise bedeutet `NN` die doppelte Normallänge. Als Folge hiervon können einige Längen auf mehrere Arten dargestellt werden: `nn` ist dasselbe wie `N`, `NNN` dasselbe wie `LL`. Die tatsächliche Länge der einzelnen Bindungen ist in den Variablen `rLenS`, `rLenN` und `rLenL` gespeichert (→`set`). Die besondere Längenangabe `0` erzeugt eine Bindung der Länge Null, was sinnvoll sein kann, wenn Sie ein allgemeines Makro geschrieben haben und eine der Bindungen nicht immer vorhanden sein soll.

r/t/b/l/v-Form

Diese Syntax ist mit der Normalform in allen Punkten identisch, lediglich die Winkelangabe ist abweichend. Die Syntax gibt vor allem eine Hilfestellung beim Ansatz von Seitenketten an Ringsysteme und ist demzufolge innerhalb eines `ring`- oder `vertex`-Befehls am sinnvollsten einsetzbar. Die Buchstaben stellen dann eine symbolische Winkelangabe dar, deren konkreter Wert vom Ringsystem abhängig ist.

`r` : Eine radiale Bindung, also eine Bindung, deren Richtung durch die Gerade Zentrum-Ringatom bestimmt ist.

`t`, `b` : Winkel für Bindungen ober- und unterhalb der Papierebene (top und bottom). Besonders beim Ringtyp `chair` und verschiedenen Konformerengerüsten ist diese Bedeutung offensichtlich, bei rein planaren Formeln (ebene Ringe) werden für diese Bindungen häufig die Verlängerungen der an das Ringatom anstoßenden beiden Kanten verwendet. Welche Winkel für diese Variante tatsächlich benutzt werden, wird in Form von schematischen Formeln bei der Besprechung der einzelnen Ringe (Befehl `ring` und Vorstellung der Bibliotheken) gezeigt.



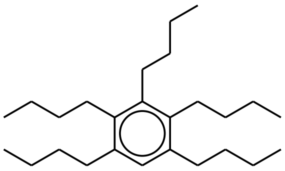
```
formula(L,R)
{ ring(,,H,,5,90)
  { 0: bond(r);
    2: bond(t,t);
    2: bond(b,o); }
}
```

`v(<n>)` : Winkel, unter dem die Bindung vom Ringatom mit der Nummer `<n>` zum nächsten Ringatom gezeichnet wird, entspricht also dem Winkel der Ringkante `<n>`.

`l` : Entspricht dem Winkel, unter dem die letzte Bindung gezeichnet wurde.

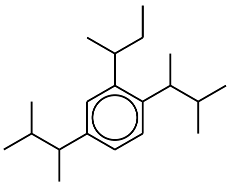
Diese Grundwinkel können durch verschiedene Symbole modifiziert werden, die in dieser Reihenfolge auftreten und kombiniert werden können:

`/` : Diese Angabe verwandelt den Winkel eines `r`-, `t`- oder `b`-Winkels in seinen Komplementärwinkel. Dies ist der Winkel, der dazu führt, daß eine Kette mit der Winkelfolge `r r/ r r/ r` in Zickzackform verläuft und sich dabei an der horizontalen respektive vertikalen Koordinatenachse orientiert. Der Innenwinkel der Knickstellen entspricht dann stets 60°.



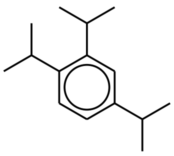
```
formula(L,R)
{ ring()
  { 1: bond(r; r/; r; r/);
    2: bond(r; r/; r; r/);
    3: bond(r; r/; r; r/);
    4: bond(r; r/; r; r/);
    5: bond(r; r/; r; r/); }
}
```

`t` : Hiermit erzeugen Sie einen Winkel, der dem `/`-Winkel gerade entgegengesetzt ist und so eine Verzweigungsbindung hervorbringt. Dieser Winkel tritt somit üblicherweise innerhalb von `branch` auf. In der Kombination `/t` erhalten Sie den Abzweigwinkel zur „anderen" Seite der Zickzackkette:



```
formula(L,R)
{ ring()
  { 1: bond(r) branch{ bond(rt); }
       bond(r/)  branch{ bond(r/t); }
       bond(r);
    3: bond(r) branch{ bond(rt); }
       bond(r/)  branch{ bond(r/t); }
       bond(r);
    4: bond(r) branch{ bond(rt); }
       bond(r/)  branch{ bond(r/t); }
       bond(r); }
}
```

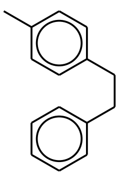
`+`, `-` : Diese Zeichen können mehrfach auftreten und erhöhen beziehungsweise vermindern den Zeichenwinkel um jeweils 60°. Dies ist der übliche Winkel von Alkanen in Zickzackform oder der Hauptwinkel bei Sechsringen. Die Wirkung entspricht einer Drehung aus der augenblicklichen Richtung nach links beziehungsweise nach rechts. Sie erzeugen damit Änderungen in der Kettenrichtung oder Isopropylverzweigungen:



```
formula(L,R)
{ ring()
  { 2: bond(r) branch { bond(r+); bond(r-); };
    3: bond(r) branch { bond(r+); bond(r-); };
    5: bond(r) branch { bond(r+); bond(r-); };
  }
}
```

Für längere Ketten mit Methylverzweigung ist die Bestimmung des Abzweigwinkels mit `rt` oder `r/t` besser geeignet.

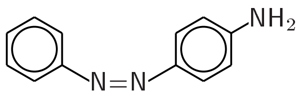
`|` : Dieses Zeichen spiegelt die Bindung an der vertikalen Achse. Sie können so einfach symmetrisch gebaute Moleküle erzeugen:



```
formula(L,R)
{ ring()
  { 4: bond(r; r+; r|)
       ring(,0,,,,r)
       { 3: bond(r); };
  }
}
```

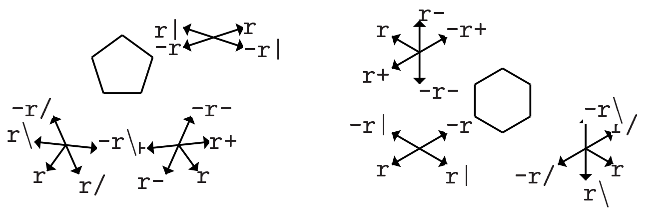
Es ist möglich, eine Summe aus einem konkreten Winkel `<phi>` und dem durch die aufgeführten symbolischen Angaben erhaltenen Winkel zu bilden, um zum Beispiel um 10° von einer radialen Bindung abzuweichen. Der Operator `+` bildet die gewöhnliche Summe der Winkel, `-` dagegen liefert die Summe aus dem konkreten Winkel und der dem symbolischen Winkel *entgegengesetzten* Richtung.

- kann auch ohne konkrete Zahlangabe benutzt werden, um Bindungen entgegen einer Richtung zu zeichnen:

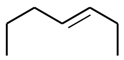



```
formula(L,R)
{ ring()
  { 5: bond(r) atom("N=N",L,R) bond(-r|)
       ring(,0,,,,r)
       { 3: bond(r) atom("N",C,R)
                    atom("H$_2$",L); };
  }
}
```

Eine ausführliche Besprechung der einzelnen Varianten sowie zahlreiche Beispiele sind im Tutorial, Abschnitt 2.2.1 gegeben. Es folgt hier eine kurze Zusammenfassung einiger Richtungen, die durch Kombination verschiedener Winkelsymbole erhalten werden können.

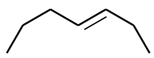


Zu Beginn einer Formel werden diese symbolischen Angaben mit Defaultwinkeln assoziiert, und zwar 30° für `r`, 90° für `t` und -90° für `b`. Sie können diese Angaben somit auch außerhalb eines Ringsystems verwenden:



```
formula(L,R)
{ bond(t; r; r/; r,=; r/; b) }
```

Um die Startwinkel zu verändern, können Sie →`set` einsetzen:



```
formula(L,R)
{ set("iAngleR",30)
  set("iAngleT",60)
  set("iAngleB",-60)
  bond(t; r; r/; r,=; r/; b)
}
```

#-Form

Die Form mit #<n> (auch als #-Syntax bezeichnet) dient zur Erzeugung einer Bindung, deren Richtung und Länge nicht spezifiziert wird, sondern die zu einem zuvor mit `saveXY` gespeicherten Atom führt. Sie können damit das aktuelle Atom mit dem Atom <n> verbinden. Beispiele sind bei →`saveXY` und →`restoreXY` zu finden.

Verkettete Bindungslisten

Die Bindungsbeschreibungen mehrerer `bond`-Befehle können, durch Semikola getrennt, in einem einzigen `bond`-Kommando zusammengefaßt werden und dadurch viel Schreibarbeit sparen. Es sind dabei alle Syntaxvarianten sowie Kombinationen möglich. Einige Äquivalenzen:

```
bond(30) bond(0,=,L) bond(-30) bond(30)
 == bond(30; 0,=,L; -30; 30)

bond(r) bond(r/) bond(r)
 == bond(r;r/;r)
```

3.1.4 bracket

```
{formula | multiline} bracket() | bracket([) | bracket(])
```

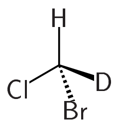
Mit Hilfe dieses Befehls fügen Sie eckige Klammern um die letzte mit `formula` erzeugte Formel oder mit `multiline` erzeugten Formelblock. Wird dieser Befehl nach einem anderen als den angegebenen Kommandos ausgeführt, können die Ergebnisse von den Erwartungen abweichen! Sie können angeben, ob eine linke oder rechte Klammer gewünscht wird oder ob – mit einer leeren Parameterliste – beide Klammern erscheinen sollen. Die Klammern dienen vor allem zum Zusammenfassen mesomerer Grenzstrukturen oder eines Zwischenproduktes (-gemisches).

Um Näheres über die Klammersetzung in komplexen Zusammenhängen zu erfahren, lesen Sie bitte Abschnitt 2.3.5 des Tutorials.

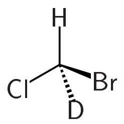
3.1.5 branch

```
branch{ <aliste_1>;
        <aliste_2>;
        ...
      }
```

Mit diesem Befehl können Sie ausgehend von der aktuellen Position, Verzweigungen aufbauen. Jede Formelkette `<aliste>` beginnt aufs Neue am Verzweigungspunkt. Als Beispiel mögen folgende Formeln dienen, in denen vom zentralen Kohlenstoffatom je drei Ketten abzweigen. Da der Befehl nach seinem Abschluß den Startpunkt unverändert läßt, müssen nur zwei Ketten im Rumpf von `branch` formuliert werden:



```
formula(L,R)
{ atom("Cl") bond(30,,L)
  branch { bond(90,,L) atom("H");
           bond(-20,<<,L) atom("D");
         }
  bond(-70,<.,L) atom("Br")
}
```



```
formula(L,R)
{ atom("Cl") bond(30,,L)
  branch { bond(90,,L) atom("H");
           bond(-20,<<,L) atom("Br");
         }
  bond(-70,<.,L) atom("D")
}
```

Dieser Befehl eignet sich durch seine hierarchisch-rekursive Arbeitsweise hervorragend für strukturiert denkende Mitmenschen; gehören Sie dagegen zu den Freigeistern, wird Ihnen die Möglichkeit, mit →`saveXY` Punkte zu speichern und später darauf zurückzukommen (→`restoreXY`, →`bond`), sehr willkommen sein, da so Verzweigungen ohne zusätzliche Schachtelungsebenen, dafür mit hintereinander auftretenden linearen Abschnitten, simuliert werden.

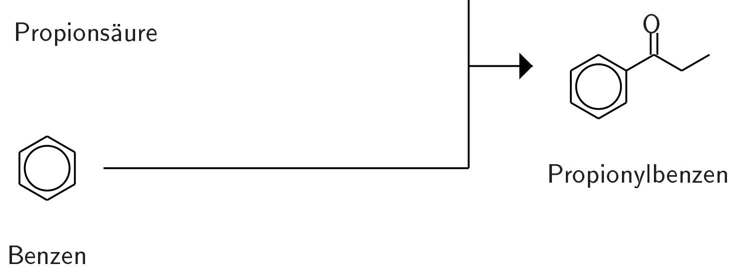
3.1.6 joinh

```
joinh(<n>, <pos>)
{ <rliste_1>;
  ...
  <rliste_n>;
}

<pos> := L | C | R
```

Der Befehl dient zum Zusammenführen von `<n>` horizontal laufenden Reaktionsketten. Die einzelnen Ketten werden so angeordnet, daß sie in einem gleichen Abstand übereinander verlaufen. Der Parameter `<pos>` legt fest, ob sie dabei linksbündig, zentriert oder rechtsbündig zueinander stehen. Eventuell auftretende Formelbeschriftungen werden dabei mit einem konstanten Abstand von der Mittellinie gesetzt, sodaß der vertikale Mindestabstand dem Wert des Parameters `rTextSep` entspricht. Aus diesem Grunde dürfen beschriftete Formeln *keine* Abstandsparameter besitzen! Formeln, die keine Beschriftung aufweisen, beeinflußen durch ihre Größe den Textabstand nicht.

Der Anschlußpunkt für weiterführende Reaktionspfeile liegt auf der rechten Seite in der Mitte des Formelblockes. Das nachfolgende Schema zeigt als Beispiel eine Synthese, bei der zwei getrennte Reaktionsketten zum Addukt zusammengeführt werden. Typischerweise folgt nach `joinh` ein `arrow`-Befehl, da `joinh` keine Pfeilspitzen zeichnet (die beim Ineinanderschachteln mehrerer `joinh`-Befehle hinderlich wäre). Durch die Positionierung `L` beginnen alle Teilketten linksbündig:

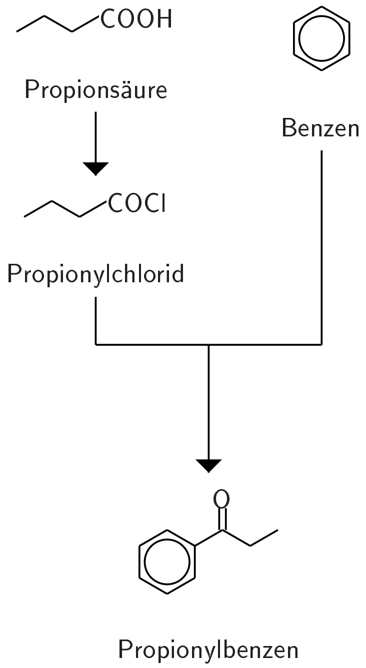
```
joinh(2,L)
{ % upper line
  formula(L,R,"Propions""aure"){ bond(30;-30;30) atom("COOH",L) }
  arrow(,12){}
  formula(L,R){ bond(30;-30;30) atom("COCl",L) }
  ;
  % lower line
  formula(L,R,"Benzen"){ ring(){} }
  ;
}
arrow(,12){}
formula(L,R,"Propionylbenzen",HR,24)
{ ring()
  { 4: bond(r) branch{ bond(rt,=C) atom("O"); }
       bond(r/;r); }
}
```



3.1.7 joinv

```
joinv(<n>, <pos>)
{ <rliste_1>;
  ...
  <rliste_n>;
}

<pos> := T | C | B
```

Ähnlich →`joinh` für horizontale Ketten, führt dieser Befehl `<n>` vertikal von oben nach unten laufende unabhängige Reaktionsketten an einem Punkt in der Mitte der Unterkante des gesamten Blockes zusammen, von dem aus ein Reaktionspfeil nach unten angesetzt werden kann. Die einzelnen Ketten werden gemäß dem Parameter `<pos>` mit ihren Oberkanten, Mitten oder Unterkanten zueinander ausgerichtet. Die einfache Anwendung des Befehls erfolgt analog zu →`joinh`, das dortige Beispiel präsentiert sich in der vertikalen Form vermöge `T` bündig zur oberen Kante:




```
joinv(2,T)
{ % left line
  formula(T,B,"Propions""aure",V,24)
  { bond(30;-30;30) atom("COOH",L) }
  arrow(-90,12){}
  formula(T,B,"Propionylchlorid",V,24)
  { bond(30;-30;30) atom("COCl",L) }
  ;

  % right line
  formula(T,B,"Benzen",V,24)
  { ring(){} }
  ;
}

arrow(-90){}

formula(T,B,"Propionylbenzen",V,24)
{ ring()
  { 4: bond(r) branch{ bond(rt,=C) atom("O"); }
       bond(r/;r); }
}
```

3.1.8 emove

```
emove(#<i>, <iAngle>, <iCtrl>, #<j>, <jAngle>, <jCtrl>)
```

(Nur mit PostScript-Ausgabe verfügbar!) Der Befehl erzeugt kleine gerade oder gebogene Pfeile, mit denen Wanderungsbewegungen von Elektronen dargestellt werden. Pfeile können innerhalb einer Formel verlaufen oder Punkte in verschiedenen Formeln miteinander verbinden. Der Befehl emove selbst muß wie formula oder arrow außerhalb eines Formelrumpfes stehen.

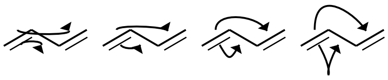
Der Start- und Endpunkt der Pfeile wird von Punkten gebildet, die mit Hilfe zweier saveXY-Befehle gespeichert wurden, die zugeordnete Nummer tritt hier als Parameter <i> beziehungsweise <j> in Erscheinung. Es wird ein Abstand der Größe rEmove zwischen Pfeil und spezifiziertem Punkt eingefügt, damit der Pfeil nicht direkt an der Bindung klebt. Die genaue Gestalt der Pfeile wird durch die beiden Winkelangaben <iAngle> und <jAngle> sowie zwei reelle Zahlen <iCtrl> und <jCtrl bestimmt. Die Winkel geben an, unter welchem Winkel Pfeilende und -spitze die spezifizierten Punkten verlassen beziehungsweise erreichen. Diese Winkel sind den Angaben bei bond analog. Die Linien, die in Verlängerung die Start- und Endpunkte unter den gegebenen Winkeln verlassen, können als Tangenten an den Pfeilbogen angesehen werden. Auf ihnen liegen, in den Abständen <iCtrl> beziehungsweise <jCtrl>, zwei Kontrollpunkte.

Die Menge der Start-, End- sowie der beiden Kontrollpunkte steuert dann den genauen Verlauf eines PostScript-Splines (kubischer Spline), an dessen Ende eine kleine Pfeilspitze aufgesetzt ist.

Will man eine konkrete Elektronenbewegung zeichnen, ist die Wahl der Winkel meist kein Problem, da sie durch die geplanten Aus- und Einfallwinkel bestimmt werden. Fingerspitzengefühl und Erfahrung erfordert dagegen die Wahl der Kontrollparameter, da sehr leicht zu krumme oder zu gerade oder falsch gebogene Bögen zustandekommen.

Einige Beispiele sollen den Einfluß der Kontrollparameter zeigen:

```
formula(L,R)
{ saveXY(#1,30,n) bond(30,=)
  saveXY(#3,-30,n) bond(-30)
  saveXY(#2,30,n) bond(30,=)
}
emove(#1,90,1,#2,120,1)
emove(#1,-60,1,#3,-120,1)

emove(#1,90,5,#2,120,5)
emove(#1,-60,5,#3,-120,5)

emove(#1,90,10,#2,120,10)
emove(#1,-60,10,#3,-120,10)

emove(#1,90,20,#2,120,10)
emove(#1,-60,20,#3,-120,20)
```



Sie erkennen an den Beispielen zweierlei: a) der Einfluß der Kontrollparameter auf die Kurvenkrümmung ist umso stärker, je größer diese sind, das heißt je weiter entfernt die Kontrollpunkte sind. Die vier Steuerungspunkt bilden ein Viereck, innerhalb dessen die Splinekurve vollständig liegt, wobei jeder Kontrollpunkt die Kurve gleichsam an sich ziehen möchte. Je weiter er entfernt ist, desto mehr wird auch die Kurve in seine Richtung gebogen. Am vierten Beispiel können Sie sehen, daß auf diese Weise sogar Verschlingungen entstehen können. b) Die meisten Startpunkte von Elektronensplines liegen in der Mitte von Bindungen. Hier vereinfacht der Einsatz der erweiterten `saveXY`-Syntax die Formelbeschreibung erheblich, da mit nur zwei Befehlen der Bindungsmittelpunkt gesichert und die Bindung selbst gezeichnet werden kann. Zur Angabe der halben Bindungslänge wird die Längenangabe `n` statt `N` bei normalen Bindungen eingesetzt. Im Tutorial, Abschnitt 2.6, werden konkrete Anwendungsfälle beschrieben.

3.1.9 fbox

```
{formula | multiline} fbox
```

Mit diesem Befehl erzeugen Sie einen Rahmen um die zuletzt mit `formula` gezeichnete Formel oder mit `multiline` erzeugten Formelblock.

Näheres über die Rahmung komplexer Zusammenhänge ist in Abschnitt 2.3.5 des Tutorials zu erfahren.

3.1.10 formula

```
formula(<pos>, <Cpos>, ["<Text>", HA|HR|V, <dy>])
{ <fliste> }
[ nospace ]

<pos> := C | L | R | T | B | TL | TR | BL | BR
<Cpos> := C | L | R | T | B | TL | TR | BL | BR

Defaults: <pos>  = L
          <Cpos> = R
```

Dies ist der zentrale Befehl des Chemiepaketes. Mit ihm bilden Sie anhand grundlegender Elemente Formeln, die mit diesem Befehl zu einer festen Einheit verwandelt werden, die automatisch bezüglich weiterer Formeln plaziert wird und in Zusammenhang mit Pfeilen zu Reaktionsschemata führt. Auch weitere abstraktere Plazierungsbefehle wie →`multiline` oder →`joinh` erfordern Formeln als Basis.

Die Formel selbst wird im Rumpf des Befehls `<fliste>` durch eine Folge von Grundelementen (→`atom`, →`bond`, →`ring`, →`branch`) gebildet. Diese Einheit wird so plaziert, daß sie in der Relation `<pos>` zum Startpunkt steht. Der Anschlußpunkt für die folgenden Formeln wird durch den Parameter `<Cpos>` bestimmt. Die Bedeutung der Parameter wird im Tutorial detailliert gezeigt.

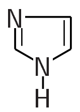
Normalerweise wird nach einer Formel in der Richtung des Anschlußpunktes ein Zwischenraum gesetzt, damit die einzelnen Elemente nicht direkt aneinanderhängen. Ist ebendies beabsichtigt, fügen Sie unmittelbar nach `formula` den Befehl `nospace` ein.

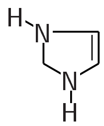
3.1.11 gotoXY

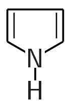
```
gotoXY([<x>] [, [<y>]])

Defaults: aktuelle x- und y-Werte
```

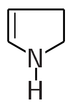
Mit diesem Befehl kann der aktuelle Punkt auf einen bestimmten absoluten Punkt gesetzt werden. Dies widerspricht zunächst der Forderung nach einer selbständigen Ausrichtung aller Elemente eines Schemas, kann aber sinnvoll sein, wenn Sie Formeln an festen Stellen erscheinen lassen wollen, unabhängig vom Zufall der jeweiligen Größe, etwa in einer tabellarischen Anordnung:

```
\begin{chemistry}[gotoxy1]
  gotoXY(0,0)
  formula(C,C,"Pyrrol",HA,36)
  { ring("cpentane",,1=3=)
    { 0: atom("N") bond(r) atom("H"); }
  }
  gotoXY(50,0)
  formula(C,C,"Pyrrolin",HA,36)
  { ring("cpentane",,1=)
    { 0: atom("N") bond(r) atom("H"); }
  }
  gotoXY(0,100)
  formula(C,C,"Imidazol",HA,36)
  { ring("cpentane",,1=3=)
    { 0: atom("N") bond(r) atom("H");
      2: atom("N"); }
  }
  gotoXY(50,100)
  formula(C,C,"Imidazolin",HA,36)
  { ring("cpentane",,3=)
    { 0: atom("N") bond(r) atom("H");
      2: atom("N") bond(r) atom("H"); }
  }
\end{chemistry}
```


Imidazol


Imidazolin


Pyrrol


Pyrrolin

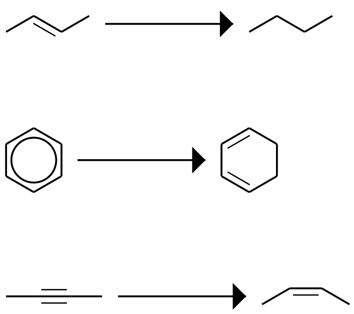
Eine relative Anordnung wird durch den Einsatz des Befehls →`shiftXY` ermöglicht.

3.1.12 multiline

```
multiline(<n> [, [<iPos>] [, [<pos>] [, [<cPos>]]]] )
{ <rliste_1>;
  ...
  <rliste_n>;
}

<iPos> := L | C | R
<pos> := C | L | R | T | B | TL | TR | BL | BR
<Cpos> := C | L | R | T | B | TL | TR | BL | BR

Defaults: <iPos> = L
          <pos>  = L
          <cPos> = R
```

Der Befehl dient dazu, `<n>` Zeilen mit Formeln oder Reaktionsketten übereinander zu setzen, wobei zwischen den einzelnen Zeilen ein konstanter Abstand verbleibt. Die einzelnen Zeilen werden gemäß `<iPos>` linksbündig, mittig oder rechtsbündig zueinander ausgerichtet. Der `<n>`-zeilige Block als Ganzes wird gemäß `<pos>` auf den aktuellen Punkt ausgerichtet, der Anschlußpunkt für folgende Formeln gemäß `<cPos>` bestimmt (ähnlich dem Positionierungsverfahren für `formula`). Beschriftungen von Formeln werden in einem selbständig berechneten, konstanten Abstand gesetzt, sodaß alle Texte einer Zeile in gleicher Höhe aufscheinen. Sie dürfen innerhalb der beschrifteten `formula`-Befehle keine Abstandsangaben für den Textparameter vorgeben!

Im einfachsten Falle wird der Befehl benutzt, um längere Reaktionsschemata auf mehrere Zeilen aufzuteilen oder mehrere kurze Ketten übereinanderzusetzen:



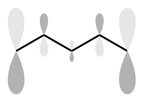
```
\begin{chemistry}
  multiline(3,L)
  { formula(L,R){ bond(30; -30,=;30) }
    arrow(){}
    formula(L,R){ bond(30; -30;30) }
    ;
    formula(L,R){ ring(){} }
    arrow(){}
    formula(L,R){ ring(,,H0=2=){} }
    ;
    formula(L,R){ bond(0; 0,3; 0) }
    arrow(){}
    formula(L,R){ bond(30;0,=;-30) }
    ;
  }
\end{chemistry}
```

Der Spezialfall `<n>` gleich Eins kann jedoch eingesetzt werden, um mehrere Formeln als Einheit zu betrachten und zu plazieren, einzurahmen oder als Verzweigung dienen zu lassen. Lesen Sie hierzu Abschnitt 2.3.5 des Tutorials.

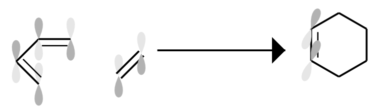
3.1.13 orbital

```
orbital([<iAngle>] [, [<rWeight>]])

Default:
         <rWeight> = 1
```

(Derzeit nur mit einem PostScript-Ausgabeformat!) Mit Hilfe dieses Befehls können Orbitallappen oder -keulen dargestellt werden, wie Sie gelegentlich zur Illustration der Vorgänge bei Zykloadditionen verwendet werden. Der Parameter `<iAngle>` ist der Winkel, den die Längsachse der Keule und die positive x-Achse einschließen, `<rWeight>` ist ein Skalierungsfaktor, der die Größe des Orbitals angibt. Die obere und die untere Hälfte des Orbitals unterscheiden sich durch ihren Grauwert. Einige Beispiele:



```
formula(L,R)
{ orbital(90,2) bond(30)
  orbital(-90,1) bond(-30)
  orbital(90,0.5) bond(30)
  orbital(-90,1) bond(-30)
  orbital(90,2)
}
```



```
multiline(1)
{ formula(L,R)
  { orbital(90,1) bond(135,=)
    orbital(-90,1) bond(45)
    orbital(-90,1) bond(0,=) orbital(90,1)
  }
  shiftXY(12,0)
  formula(L,R)
  { orbital(90,1) bond(45,=C) orbital(90,1) }
;}
arrow(){}
formula(L,R)
{ ring(,,H1=)
  { 1: orbital(-110,1);
    2: orbital(-110,1);  }
}
```

3.1.14 restore

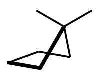
```
restore(#<n>)
```

Mit diesem Befehl wird der unter der ganzen Zahl `<n>` gespeicherte Parametersatz reaktiviert (→`save`).

3.1.15 restoreXY

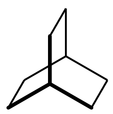
```
restoreXY(#<n>)
```

Mit diesem Befehl wird der aktuelle Punkt auf die zuvor mit →`saveXY` unter der ganzen Zahl `<n>` gespeicherte Position gesetzt. Sie können so Verzweigungen in einer Struktur erreichen, ohne wie bei →`branch` rekursiv denken zu müssen (sozusagen „wilde“ Verzweigungen:-)). Beispiele:



Dimethyl-zyklopentan

```
formula(C,C,"Dimethyl-zyklopentan",HA,36)
{ saveXY(#1) bond(170; -160; -10,t; 60,t,L)
  saveXY(#2) bond(#1)
  restoreXY(#2) bond(30) restoreXY(#2) bond(150)
}
```



Bizyklo[2.2.2]oktan

```
formula(C,C,"Bizyklo[2.2.2]oktan",HA,36)
{ bond(30,t,L) saveXY(#1) bond(-30,t,L; 60; 150,,L)
  saveXY(#2) bond(-150,,L; -120)
  restoreXY(#1) bond(90,t,L; 60) bond(#2)
}
```

Die Beispiele des `branch`-Befehls können mit `saveXY` und `restoreXY` folgendermaßen formuliert werden:

```
\begin{chemistry}
  formula(L,R)
  { atom("Cl") bond(30,,L)
    saveXY(#1)
    bond(90,,L) atom("H")
    restoreXY(#1) bond(-20,<<,L) atom("D")
    restoreXY(#1) bond(-70,<.,L) atom("Br")
  }

  formula(L,R)
  { atom("Cl") bond(30,,L)
    saveXY(#1)
    bond(90,,L) atom("H")
    restoreXY(#1) bond(-20,<<,L) atom("Br")
    restoreXY(#1) bond(-70,<.,L) atom("D")
  }
\end{chemistry}
```

Es sei noch erwähnt, daß das Bizyklo[2.2.2]oktan als Grundgerüst in `bicyclib.pm` vorhanden ist. Verschiedene Zyklopentankonformeren sind in `mncyclib.pm` enthalten.

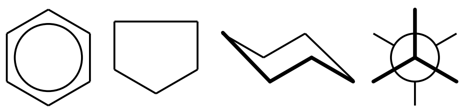
3.1.16 ring

```
ring([ <typ> [, [<start>] [, [<bliste>]
     [, [[#]<len>] [, [<p1>] [, [<p2>]]]]]]])
{ [ <aliste_1>;
    <aliste_2>;
    ... ]

  % nur bei <typ> = "ring", "cpentane", "chair":
  [ vertex([<kante_1>] [, [<kante_2>]
           [, [<bliste_1>] [, [<p1>] ]]]])
    { [<aliste_11>;
       <aliste_12>;
       ... ]
    }
    ... ]
}

<start> := 0..<n-1>
<bliste> := [H|O] [ 0|1|...|<b><btype> <bliste>]
<aliste_i> := C|0|1|...|<n-1> : <fliste>

Defaults: <typ>    = "ring"
          <start>  = 0
          <bliste> = O
          <len>    = N
          <p1>     = 6
          <p2>     = -90
```

Der Befehl dient zur Erzeugung vorgefertigter Ringstrukturen. Die erweiterte Syntax mit `vertex` ist nur bei den angegebenen Ringtypen möglich.

Fixe Parameter

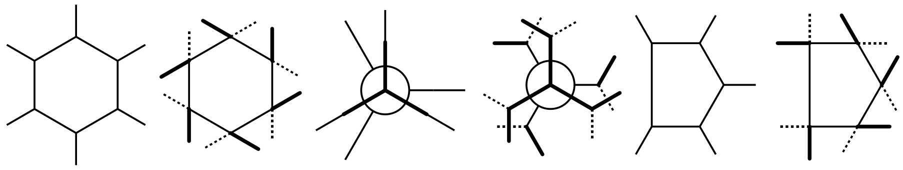
Der Parameter `typ` bestimmt gemäß nachfolgender Kodierung, welches Grundgerüst gezeichnet wird:



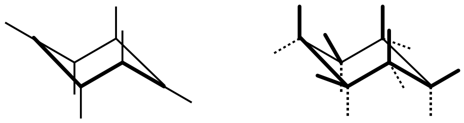
ring cpentane chair newman

Obige Liste enthält die im Compilercore definierten Ringstrukturen; mit Hilfe des Befehls →`require` können Sie jedoch eine Bibliothek aus extern definierten Ringstrukturen aufbauen und bei Bedarf hinzuladen. Möglichkeiten, fünfgliedrige Ringe in Stereodarstellung ähnlich `chair` zu erzeugen, sind bei den Steroiden in Abschnitt 2.4 des Tutorials sowie der Befehlsreferenz zu `restoreXY` gezeigt worden.

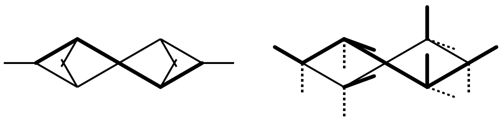
Die Anordnung der Bindungen bei den Richtungstypen `r`, `t` und `b` des Befehls `bond` wird in den nachfolgenden Formeln gezeigt. `t`-Bindungen werden durch dicke, `b`-Bindungen durch gestrichelte Linien symbolisiert.



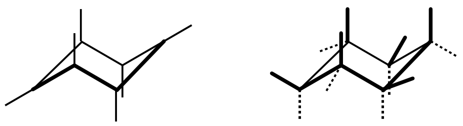
radial | top/bottom | radial | top/bottom | radial | top/bottom



radial, p=1 | top/bottom, p=1

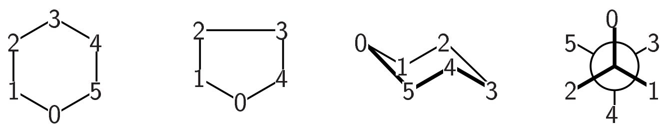


radial, p=0 | top/bottom, p=0



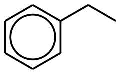
radial, p=-1 | top/bottom, p=-1

Einige der Parameter besitzen eine feststehende Bedeutung, bei anderen variiert diese je nach dem zugrundeliegenden Gerüst. Fixe Bedeutung kommt den Parametern `<start>` und `<bliste>` zu. Mit dem ersten Wert, der zwischen Null und der Anzahl der Atome im Gerüst minus Eins variieren kann, legen Sie die relative Lage des Gerüstes zum aktuellen Startpunkt fest, das heißt, welches Atom am augenblicklichen Zeichenpunkt liegt. Ist die Ringstruktur das erste Element der Formel, ist dieser Parameter ohne Bedeutung, da der Ring das Basiselement der Formel wird. Haben Sie jedoch bereits Bindungen erzeugt, geben Sie hiermit das Zielatom der letzten Bindung an. Die Numerierung der einzelnen Gerüste ist im folgenden gezeigt:

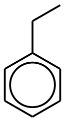


Als Grundregel für einfache n-gliedrige Ringe und die meisten anderen Grundgerüste gilt: die Nummern werden im Uhrzeigersinne größer, das Atom mit der Nummer Null liegt auf der positiven x-Achse oder ist um einen Winkel, der typspezifisch ist (-90° für den Typ `ring`), im Uhrzeigersinne gedreht. Bei einigen Grundgerüsten kann die Reihenfolge der Nummern abweichen.

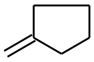
Beachten Sie bei den folgenden Anwendungsmöglichkeiten, daß `ring` den aktuellen Startpunkt nicht verändert, was in der letzten Formel ausgiebig verwandt wird, um die drei Ringe zu zeichnen, ohne einige von ihnen als Substituenten der anderen betrachten zu müssen:



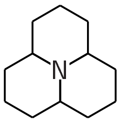
```
formula(C,C)
{ bond(150; -150) ring(,4){} }
```



```
formula(C,C)
{ bond(-150; -90) ring(,3){} }
```



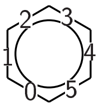
```
formula(C,C)
{ bond(30,=C) ring("cpentane",1){} }
```

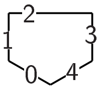


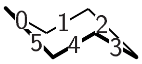

```
formula(C,C)
{ ring(,4,H){} ring(,2,H){} ring(,0,H){}
  atom("N")
}
```

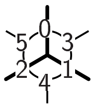
Die Numerierung dient auch zur Kenntlichmachung, welcher Substituent an welchem Ringatom beginnt (Nummern in `<aliste>`).

Mit der *Bindungsliste* `<bliste>` variieren Sie die Bindungsarten, die beim Aufbau des Gerüstes eingesetzt werden. Sie besteht aus einer Folge von Paaren aus Bindungsnummer und Bindungstyp. Die Bindungsnummer beginnt bei Null und kann bis zu Eins kleiner als die Anzahl der Bindungen im Ring überhaupt sein. Die Zuordnung der Bindungsnummer zu den einzelnen Bindungen ist folgende:

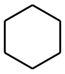




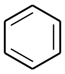




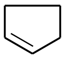
Die Angabe des Bindungstyps kennen Sie bereits vom Befehl `bond` her. Da die Angabe eines oder mehrerer Bindungstypen nicht automatisch das Zeichnen eines Aromatenringes unterbindet, müssen Sie dies gegebenenfalls zusätzlich anweisen. Für diesen Zweck stehen die zusätzlichen Typen `H` für gesättigte (Ring wird nicht gezeichnet) und `O` für aromatische Kerne (Ring wird gezeichnet) zur Verfügung. Diese Angaben müssen vor einer eventuellen Bindungsliste stehen! Einige Beispiele mögen die Bindungsliste verdeutlichen:



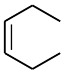
```
formula(C,C){ ring(,,H){} }
```



```
formula(C,C){ ring(,,H0=2=4=){} }
```



```
formula(C,C){ ring("cpentane",,0=){} }
```



```
formula(C,C){ ring(,,H1=4s){} }
```

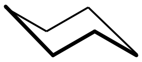
Der Parameter `<len>` kodiert die Größe des Ringes. Die Defaulteinstellung `N` entspricht bei Polygonen dem Radius des zugrundeliegenden Kreises, bei anderen Gerüsten der Länge der Hauptbindungen. Ist es geometrisch möglich, erhalten alle Bindungen diese Länge, ansonst die größtmögliche Anzahl. Bei Polygonen folgt aus der Festlegung des Radius als Grundlage, daß die Kantenlänge bei Dreiecken erheblich größer ist als bei Neunecken. Wünschen Sie die *Kantenlänge* festzulegen, müssen Sie die

Größenvariante mit # verwenden, siehe unten.

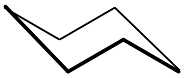
Eine Abweichung vom Normalwert der Ringgröße ist besonders bei komplizierten Gerüsten wie dem Zyklohexan sinnvoll:

```
formula(C,C) { ring("chair",,,S){} }
```

```
formula(C,C) { ring("chair",,,N){} }
```

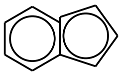


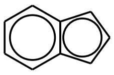
```
formula(C,C) { ring("chair",,,L){} }
```



```
formula(C,C) { ring("chair",,,NN){} }
```

Die Variante mit einem # vor der Bindungslänge ist nur bei Gerüsten des Typs `ring` erlaubt und legt fest, daß `<len>` nicht dem Radius des Kreises, sondern der Kantenlänge des Polygons entspricht. Polygone unterschiedlicher Eckenzahl weisen dann die festgelegte, gleiche Kantenlänge auf. Sinnvoll ist dies, wenn kondensierte Ringe unterschiedliche Gliederzahl aufweisen, wie der Vergleich zeigt:





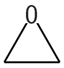
```
gotoXY(0,0)
formula(C,C) { ring(,5){}
                 ring(,2,,,5,0){}
               }

gotoXY(60,0)
formula(C,C) { ring(,5,,#N){}
                 ring(,2,,#N,5,0){}
               }
```

In der Grundform sind die Kantenlängen eines Sechsringes etwas kürzer als die eines Fünfringes, was beim Kondensieren der Systeme deutlich auffällt (noch extremer bei Dreiringen). In einem solchen Falle muß deswegen jeder der beteiligten Ringe eine konstante Kantenlänge aufweisen, mit `#N` angezeigt.

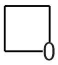
Variable Parameter

Die Angaben `<p1>` und `<p2>` haben je nach dem Gerüsttyp unterschiedliche Bedeutung.

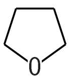
ring : Bei diesem Grundgerüst haben sie die Bedeutung der Anzahl der Ecken des Ringes (Default ist 6 für einen Benzenring) und eines Winkels in Altgraden, um den der Ring gedreht ist. Bezugsachse ist hierbei die positive x-Achse, der Drehsinn mathematisch negativ (Uhrzeigersinn). Der Defaultwert von -90° sorgt dafür, daß der Benzenring auf der Ecke mit der Nummer 0 steht. Die wichtigsten alizyklischen Grundkörper können wie folgt gesetzt werden:



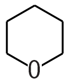

```
formula(C,C) { ring(,,H,,3,90)
                 { 0: atom("O"); } }
```



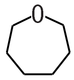

```
formula(C,C) { ring(,,H,,4,-45)
                 { 0: atom("O"); } }
```




```
formula(C,C) { ring(,,H,,5,-90)
                 { 0: atom("O"); } }
```

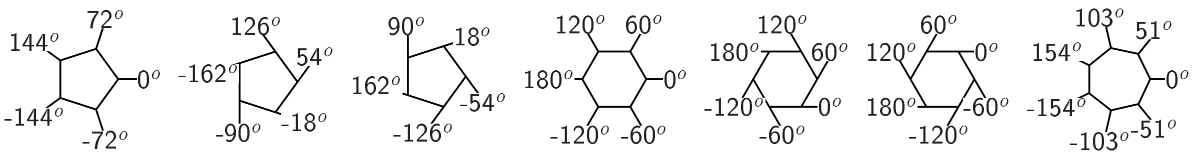


```
formula(C,C) { ring(,,H,,6,-90)
               { 0: atom("O"); } }
```



```
formula(C,C) { ring(,,H,,7,90)
               { 0: atom("O"); } }
```

Einige wichtige Winkel im Zusammenhang mit diesen grundlegenden Polygonen sind in den folgenden Formeln angegeben. Die Winkel gelten für die Bindung in Pfeilrichtung vom nächsten Ringatom aus. Sind die Ringe gedreht, so ist der Drehwinkel zu den angegebenen Werten zu addieren.

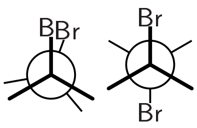


cpentane : Der Parameter <p1> ist ohne Bedeutung, <p2> stellt wie beim Grundgerüst `ring` den Drehwinkel des gesamten Ringes dar.

chair : <p2> stellt wie beim Grundgerüst `ring` den Drehwinkel des gesamten Ringes dar. <p1> kodiert mit den drei möglichen Werten 1, 0 und −1 die beiden Sessel- und twist-Konformeren:

$p_1 = 1$ $p_1 = 0$ $p_1 = -1$

newman : Dieses Gerüst entspricht dem axialen Blick auf eine Einfachbindung (Darstellung in der Newman-Projektion). Vom vorderen Atom ausgehende Bindungen zweigen unter den festen Winkeln 90°, 210° und 330° ab. Die Bindungen des hinteren Atoms werden um den Winkel `p1` verdreht dargestellt:

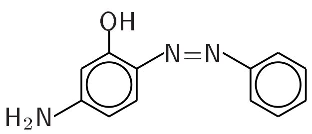


syn anti

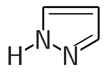
```
formula(L,R,"syn",HA,36)
{ ring("newman",,,L,-20){ 0: atom("Br");
                          3: atom("Br");} }

formula(L,R, "anti",HA,36)
{ ring("newman",,,L,180){ 0: atom("Br");
                          3: atom("Br");} }
```

Substituenten

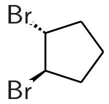
Mit den einzelnen <aliste_i>-Komponenten im Rumpf des Befehls spezifizieren Sie Substituenten, die ihren Ursprung im i-ten Ringatom haben. Die Numerierung der Ringatome wurde bereits im Zusammenhang mit <start> vorgestellt. Die Substituenten können einzelne Bindungen, weitere Ringe oder Kombinationen aus allen Formelelementen sein. Beispiele:

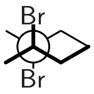


```
formula(L,R)
{ ring()
  { 1: bond(r) atom("H$_2$N",R);
    4: bond(r) atom("N=N",L,R) bond(r-) ring(,2){};
    3: bond(r) atom("O",C,R) atom("H",L);
  }
}
```



```
formula(L,R)
{ ring("cpentane",,2=4=)
  { 0: atom("N");
    1: atom("N") bond(r) atom("H");
  }
}
```



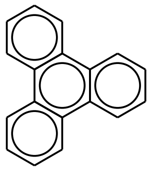


```
formula(L,R)
{ ring(,,H,,5,0)
  { 2: bond(r,<<) atom("Br");
    3: bond(r,<.) atom("Br");
  }
}
formula(L,R)
{ bond(150)
  ring("newman",3)
  { 0: atom("Br");
    1: bond(30,t);
    4: atom("Br");
  }
}
```

Der `vertex`-Befehl

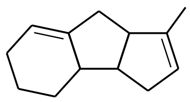
Bei den Ringtypen, die einen Drehwinkel unterstützen, ist eine erweiterte Syntax mit `vertex`-Befehlen möglich. Diese dienen dazu, weitere Ringe über eine Kante mit dem aktuellen Ring zu verknüpfen. `<kante_1>` und `<kante_2>` kodieren die betroffenen Kanten des Basisringes (des aktuellen Ringes) und des zweiten Ringes in Form der Zahlen, die auch zum Aufbau der Bindungsliste benutzt werden. `<bliste_1>` ist die Bindungsliste für den neuen Ring, der ebenfalls nur vom Typ `ring` sein darf. `<p1>` schließlich ist die Ringgröße des neuen Ringes.

Der Rumpf des `vertex`-Befehls ist identisch zum Rumpf von `ring`, das heißt es können, durch Positionszahlen spezifiziert, Substituenten oder auch weitere `vertex`-Befehle eingesetzt werden. Einige Beispiele sollen den Einsatz verdeutlichen.

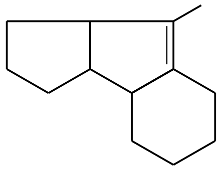


Triphenylen

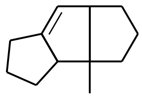
```
formula(L,R,"Triphenylen",HR,24)
{ ring(){ vertex(,0){};
          vertex(,2){};
          vertex(,4){}; }
}
```



```
formula(L,R)
{ ring(,,H,,5,90)
  {
    vertex(,1,3,H0=,5){ 0: bond(r); };
    vertex(,3,1,H0=,6){};
  }
}
```



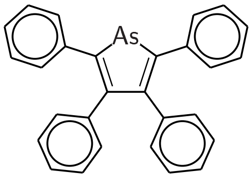
```
formula(L,R)
{ ring("cpentane",,,L)
  { vertex("cpentane",3,1,H3=)
    {3: bond(r); vertex(,4,0,H){};};
  }
}
```



```
formula(L,R)
{ ring("cpentane",,,,,0)
  { vertex("cpentane",2,2,H0=)
    {3: bond(b); vertex(,4,0,H,5){};};
  }
}
```

Symbolische Drehwinkel

Bei den Ringtypen `ring` und `cpentane` sind anstelle ganzer Zahlen auch die symbolischen Angaben `r`, `t` und `b` für den Drehwinkel <p2> möglich. Ähnlich den analogen Angaben des `bond`-Befehls, werden auch an dieser Stelle die Winkel der radialen beziehungsweise tangentialen Bindungen kodiert: der Drehwinkel des Ringes wird so bestimmt, daß die letzte gezeichnete Bindung, die zum spezifizierten Startatom führt, in der symbolischen Relation zum Ring steht:



Tetraphenylarsol

```
formula(L,R,"Tetraphenylarsol",HR,24)
{ ring(,,H1=3=,,5,90)
  { 0: atom("As");
    1: bond(r) ring(,,,,,r){};
    2: bond(r) ring(,,,,,r){};
    3: bond(r) ring(,,,,,r){};
    4: bond(r) ring(,,,,,r){}; }
}
```

Weitere Beispiele für symbolische Drehwinkel sind im Tutorial zu finden.

3.1.17 save

```
save(#<n>)
```

Dieser Befehl sichert die beim Formelsatz benutzten graphischen Parameter unter der ganzen Zahl `<n>`. Sie verwenden den Befehl, wenn Sie mit →`set` lokale Änderungen am Parametersatz vornehmen wollen und die Originalwerte wieder benötigen (→`restore`). Beispiele zu diesem Befehl finden Sie im Tutorial, Abschnitt 2.5.

3.1.18 savecontext

```
savecontext(#<n>)
```

Mit diesem Befehl wird der Kontext, der durch die zuletzt dargestellte Formel festgelegt wird, unter einer Nummer `<n>` gespeichert. Sie können so später auf diesen Formelkontext zurückgreifen (→`setcontext`), um zum Beispiel Abzweigungen im Schema zu erzeugen, auch wenn bereits andere Formeln gezeichnet wurden.

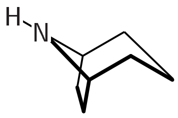
3.1.19 saveXY

```
saveXY(#<n>)
saveXY(#<n>, <iAngle>, <Len>)
```

Der Befehl speichert in der oberen Syntaxvariante die aktuellen Koordinaten unter der Nummer `<n>`. Die erweiterte, untere Syntax speichert die Koordinaten eines Punktes, der vom aktuellen Punkt um die Länge `<Len>` in Richtung des Winkels `<iAngle>` verschoben ist. Beide Angaben entsprechen denen des `bond`-Befehls. Der aktuelle Punkt wird dabei *nicht* verschoben.

Einfache Syntax

Sie benutzen den gespeicherten Punkt, um später mit `bond(#<n>)` eine Bindung von einem anderen Atom zu diesem Punkt zu zeichnen, ohne die Richtung und Länge der Bindung angeben zu müssen. Sie können so auf einfache Weise polyzyklische und irreguläre Gerüste, wie sie für Naturstoffe typisch sind, generieren. Das Tropin und Nor-Pinan sind als Grundgerüste in `bicyclib.pm` vorhanden, hier soll jedoch gezeigt werden, wie Sie sich behelfen können, falls ein Gerüst nicht in einer Bibliothek zu finden ist:

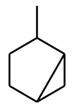


Desoxy-Tropin

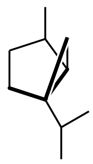
```
formula(C,C,"Desoxy-Tropin",HA,36)
{ ring("chair",,,L)
  { 0: atom("N") bond(150) atom("H");
    1: bond(-100) saveXY(#1);
    5: bond(-100,t; #1,t); }
}
```

Nor-Pinan

```
formula(L,R,"Nor-Pinan",HA,36)
{ ring(,,H)
  { 2: saveXY(#1);
    0: bond(90; #1); }
}
space(R)
formula(L,R)
{ ring("chair",,,L,-1)
  { 2: saveXY(#1);
    4: bond(-60,t; #1); }
}
```

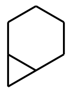


7,8,9-Trinor-thujan



Thujan

```
formula(L,R,"7,8,9-Trinor-thujan",HR,24)
{ ring(,,H)
  { 3: bond(r);
    4: saveXY(#1);
    0: bond(#1); }
}
space(R)
formula(L,R,"Thujan",HR,24)
{ ring(,,H0t1t,,5,0)
  { 4: bond(90);
    1: saveXY(#1);
    0: bond(90; #1,p; -60)
       branch { bond(30); bond(-90); };
  }
}
```

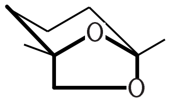


Nor-Caran

```
formula(C,C,"Nor-Caran",HA,36)
{ ring(,,H)
  { 1: saveXY(#1);
    0: bond(-150; #1);  }
}
```

Da die Thujan- und Carangerüste in planarer Darstellung reguläre Bindungswinkel und -längen aufweisen, können sie einfach wie folgt formuliert werden:

```
\begin{chemistry}[savexy2]
  formula(L,R,"Nor-Pinan",HA,36)
  { ring(,,H){ 0: bond(90; 150); } }

  formula(L,R,"Nor-Caran",HA,36)
  { ring(,,H){ 0: bond(-150; 90); } }
\end{chemistry}
```



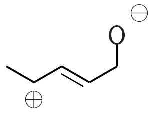
(-)-(1S,5R)-Frontalin

```
formula(C,C,"(-)-(1S,5R)-Frontalin",HA,36)
{ ring("chair",,,L)
  { 3: bond(t);
    3: bond(b,t) atom("O") saveXY(#1);
    4: atom("O");
    5: bond(t);
    5: bond(b,t; #1,t); }
}
```

Die Zusammenarbeit mit dem Befehl →`restoreXY` erweitert die Möglichkeiten, irreguläre Naturstoffgerüste aufzubauen, erheblich; siehe Befehlsbeschreibung zu `restoreXY`. Beachten Sie, daß der Gültigkeitsbereich der gespeicherten Punkte mit der aktuellen Formel nicht automatisch endet, sodaß eine spätere irrtümliche Verwendung dieses Punktes ohne erneute Zuweisung auf eine dann falsche Formel verweist! Sie können sich dies natürlich zunutze machen, wenn Sie zwei Formeln miteinander verknüpfen wollen, zum Beispiel den Elektronenübergang darstellen oder etwas besonders Trickreiches planen.

Erweiterte Syntax

Diese Befehlsvariante ist vor allem bei der Darstellung von Elektronenverschiebungen von Interesse, entsprechende Beispiele sind bei `emove` sowie im Tutorial, Abschnitt 2.6 zu finden. In einem anderem Zusammenhang ist diese Variante brauchbar, wenn Sie unkompliziert Punkte neben der Hauptlinie der Formel markieren müssen, ein Beispiel wären Ladungssymbole neben einer Alkankette:



```
formula(L,R)
{ bond(-30) saveXY(#1,-90,n)
  bond(30; -30,=; 30)
  bond(90) saveXY(#2,45,N)
  atom("O")
  restoreXY(#1) atom("$\oplus$")
  restoreXY(#2) atom("$\ominus$")
}
```

3.1.20 scale

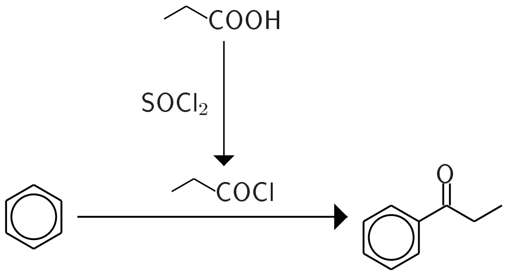
```
scale(<p>)
```

Der Befehl skaliert alle internen graphischen Parameter mit dem Faktor `<p>`. `<p>` ist eine Angabe des Bruchteils; um die Formeln auf 70% zu verkleinern, schreiben Sie daher

```
scale(0.7)
```

Zahlen größer als 1 führen zu einer Vergrößerung, Zahlen unter 1 zu einer Verkleinerung der folgenden Zeichnung.

Der Befehl wird im allgemeinen innerhalb eines →`save`/`restore`-Paares angewandt, um die Originalwerte wieder reaktivieren zu können. Als Beispiel soll eine Hilfsreaktion auf einem Reaktionspfeil verkleinert dargestellt werden:




```
\begin{chemistry}[scale]
  formula(L,R)
  { ring(){} }

  arrow()
  { text(T,L)
    { save(#1)
      scale(0.8)
      formula(T,B){ bond(30; -30)atom("COOH",L) }
      arrow(-90)
      { text(T,L) { formula(C,C){ atom("SOCl$_2$") } } }
      formula(T,B){ bond(30; -30)atom("COCl",L) }
      restore(#1)
    }
  }

  formula(L,R)
  { ring()
    { 4: bond(r) branch { bond(r+,=C) atom("O"); }
         bond(r-; r); }
  }
\end{chemistry}
```

3.1.21 shiftXY

```
shiftXY([<dx>] [, [<dy>]])

Default: <dx> = 0
         <dy> = 0
```

Der Befehl erlaubt es, außerhalb von Formeln den aktuellen Punkt um die Größe `<dx>` auf der x-Achse, um `<dy>` auf der y-Achse zu verschieben. Lassen Sie Angaben fort, findet auf der entsprechenden Achse keine Verschiebung statt.

Hauptnutzen des Befehls ist es, mehrere Reaktanden, die nicht streng nacheinander aufgefädelt erscheinen sollen, sondern eher eine Art „Wolke“ bilden, richtig gegeneinander anzuordnen. Im Gegensatz zu →`gotoXY` werden jedoch keine festen Punkte angegeben, sondern nur Relationen. Verschiebungen des Bezugspunktes (aktueller Punkt) betreffen gleichartig alle relativen Positionierungen. Es ist zweckmäßig, aber nicht notwendig, die Formeln mit den Positionierungsparametern `C,C` zu setzen. Ein Beispiel ist im Tutorial, Abschnitt 2.6, gegeben. Dort sind drei Reaktanden so zu setzen, daß zwischen ihnen die Elektronenübergänge darstellbar sind. Die Zusammenfassung aller „wild verteilten“ Reaktanden zu einer Einheit gelingt wieder einmal mit einem einzeiligen `multiline`-Kommando.

Tabelle 3.1 Interne Satzparameter, die mit `set` geändert werden können.

Parameter	Bedeutung
`rLW`	Liniendicke für alle Linien wie Bindungen, Kreise, Pfeile und sonstige Linien.
`rBW`	Maximale Dicke bei projektiven Bindungsformen.
`rAW`	Dicke der Spitze von Reaktionspfeilen.
`rBD`	Strichelungsmuster für projektive Bindungen.
`rXS`	Horizontaler Abstand von Formeln und Pfeilen zueinander.
`rXY`	Vertikaler Abstand von Formeln und Pfeilen zueinander.
`rArrowExtend`	Zusätzlicher Abstand, um den ein Reaktionspfeil auf beiden Seiten über die Beschriftung hinausreicht.
`rArrowSkip`	Abstand der Pfeilbeschriftung vom Pfeil selbst.
`rChainSep`	Abstand von Reaktionsketten zueinander bei den Befehlen `joinh` und `joinv`.
`rMultilineSep`	Abstand von Reaktionsketten zueinander beim Befehl `multiline`.
`fboxsep`	Abstand des Rahmens/Klammer von der Formel.
`rTextSep`	Vertikaler Abstand der Formellegenden von der Unterkante der tiefsten Formel.
`iAngleR`	Winkel für radiale Bindungen, Symbol `r`.
`iAngleT`	Winkel für radiale Bindungen, Symbol `t`.
`iAngleB`	Winkel für radiale Bindungen, Symbol `b`.

3.1.22 set

```
set(<cPar>, <iVal>)
```

Mit diesem Befehl können Sie jeweils einen der in Tabelle 3.1 aufgelisteten Parameter `<cPar>`, die das Layout des Formelsatzes steuern, ändern und ihn auf den Wert `<iVal>` setzen. Die Wirkung bezieht sich teilweise auf Satzparameter wie Abstände von Formelzeilen zueinander, teilweise auf Druckparameter wie Linienbreite und Pfeilstärke. Änderungen an Parametern der letztgenannten Gruppe werden in die Ausgabedatei geschrieben.

Mit diesem Befehl vorgenommene Änderungen sind permament, es ist gegebenenfalls erforderlich, sie lokal durchzuführen, indem der Parametersatz vor dem Abschnitt des Programmes, für den sie Gültigkeit besitzen sollen, unter einer bestimmten Nummer gespeichert und nach Ende des Gültigkeitsbereichs unter dieser Nummer wieder aktiviert wird (Befehle `save` und `restore`):

```
...
save(#1)
set("rLW", 2)  % erhoehe Linienstaerke
formula...
...
restore(#1)    % hole alte Werte
...            % wieder normale Linienbreite
```

Ein Beispiel für eine Anwendung ist im Tutorial, Abschnitt 2.5 gegeben.

3.1.23 setcontext

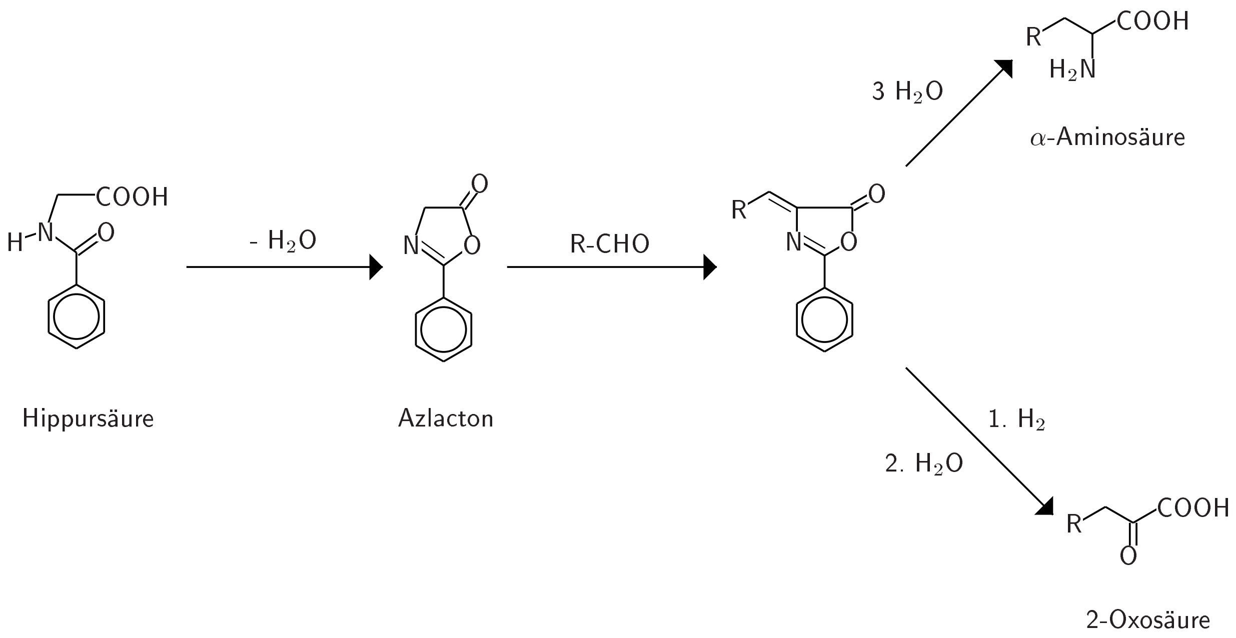
```
setcontext(#<n>, <pos>)
[nospace]

<pos> := C | L | R | T | B | TL | TR | BL | BR
```

Der Befehl legt den unter der Nummer `<n>` zuvor gespeicherten Kontext als aktuellen Kontext ab und ermittelt einen Fortsetzungspunkt, der sich aus der Kontext-Bounding Box und dem Positionierungsparameter `<pos>` ergibt. Haupteinsatzgebiet des Befehls ist das Bilden von Verzweigungen in Reaktionsketten: in Schema 3–4 wird, vom Azlakton ausgehend, eine Reaktionskette „normal" notiert in der Folge (L,R) – (L,R) – (L,TR). Damit vom Azlakton noch ein zweiter Kettenstrang abzweigen kann, muß zunächst nach Satz der Laktonformel deren Kontext gespeichert (→`savecontext`) und nach Ende der ersten Kette wieder aktiviert werden. Da die zweite Kette nach rechts unten verlaufen soll, wird als Positionierung dabei `BR` angegeben:

```
\begin{chemistry}[cont1]
multiline(1)
{ formula(L,R, "Hippurs""aure")
  { ring(,,H3s4=C,,5)
    { 0: bond(r) ring(,3){};
      1: atom("N") bond(r) atom("H");
      3: atom("COOH",L,C);
      4: atom("O"); }
  }

  arrow()
  { text(T,L) { formula(C,C){ atom("- H$_2$O") } } }

  formula(L,R, "Azlacton")
  { ring(,,HO=,,5)
    { 0: bond(r) ring(,3){};
      1: atom("N");
      3: bond(r,=C) atom("O");
      4: atom("O"); }
  }
  savecontext(#1);
}

setcontext(#1,R)
arrow(){ text(T,L) { formula(C,C){ atom("R-CHO") } } }

formula(L,TR)
{ ring("cpentane",,O=)
  { 0: bond(r) ring(,3){};
    1: atom("N");
    2: bond(r,=U) bond(-150) atom("R");
    3: bond(r,=C) atom("O");
    4: atom("O");
  }
}
```

```
savecontext(#2)

arrow(45)
{ text(T,L){ formula(C,C){ atom("3 H$_2$O") } }
}

formula(L,R,"$\alpha$-Aminos""aure", HR, 24)
{ atom("H$_2$",C,R) atom("N",L,C) bond(90)
  branch
  { bond(150; -150) atom("R");
    bond(30) atom("COOH",L,R);
  }
}

setcontext(#2,BR)
arrow(-45)
{ text(T,L){ formula(C,C){ atom("1. H$_2$") } }
  text(B,L){ formula(C,C){ atom("2. H$_2$O") } }
}

formula(L,R,"2-Oxos""aure", HR, 24)
{ atom("O") bond(90,=C)
  branch
  { bond(150; -150) atom("R");
    bond(30) atom("COOH",L,R);
  }
}
\end{chemistry}
```

Sie können durch Speichern mehrerer Kontexte und durch Verwendung mehrerer `setcontext`-Befehle verschiedene Kettenverzweigungen formulieren sowie von einer Formel mehrere Ketten abzweigen lassen, was im Tutorial, Abschnitt 2.3.4, ausführlich erläutert wird.



Schema 3–4 Mehrere Reaktionsketten zweigen von derselben Formel ab. Der Kontext dieser Formel muß dafür gespeichert werden.

3.1.24 space

```
space(<pos>)

<pos> := C | L | R | T | B | TL | TR | BL | BR
```

Mit diesem Befehl können Sie zusätzlichen Zwischenraum in der durch `<pos>` spezifizierten Richtung einfügen – oder in Form eines gezielten Rückschrittes wieder entfernen. (Nach einem Pfeil oder einer Formel ist dies in der Fortsetzungsrichtung mit `nospace` möglich.)

3.2 Sonderbefehle

Die in diesem Abschnitt enthaltenen Kommandos dürfen keinesfalls innerhalb einer `chemistry`-Umgebung in LaTeX-Dokumenten auftreten. Einige von ihnen sind Bestandteil des eigentlichen Sourcefiles, das während der Bearbeitung von LaTeX erzeugt oder von Ihnen per Hand geschrieben werden kann. Ihnen entsprechen gemäß dem folgenden Schema LaTeX-Befehle, sodaß keine Notwendigkeit besteht, die grundlegenden Befehle einzusetzen:

```
\begin{chemistry}{...}↔schema("...")
\end{chemistry}       ↔endschema
\chemfontname         ↔font
```

Es gibt jedoch auch einige Sonderbefehle, die während der Initialisierung von OCHEM benötigt werden, nicht jedoch innerhalb jeder einzelnen `chemistry`-Umgebung. Diese können innerhalb der `chemspecial`-Umgebung eingefügt werden. Diese Umgebung übernimmt den eingeschlossenen Text ohne Änderung in das eigentliche Sourcefile und steht in der Regel zu Beginn des Dokumentes. Zu den Befehlen, die sinnvoll in einem LaTeX-Dokument chemischen Inhalts eingesetzt werden können, zählen `require` und `package`.

Wenn Sie sich das nach einem LaTeX-Lauf erstellte `.chm`-File anschauen, können Sie sehen, wie Ihr Dokument in das Sourcefile übertragen wurde.

3.2.1 font

```
font("<name>")
```

Dieses Kommando enthält die LaTeX-Befehle, die den für chemische Textsymbole des `atom`-Befehls zu verwendenden Font einstellen. Diese Form des Befehls dient nur internen Zwecken, da für den LaTeX-Anwender der Befehl `\chemfontname` bereitsteht.

3.2.2 package

```
package("<Paket>")
package("[<Optionen>]<Paket>")
```

Lädt das angegebene Paket, ggf. mit Optionen, während der LaTeX-Bearbeitung der Textstücke der Formeln. Der Befehl kann mehrfach auftreten, muß jedoch vor der ersten Formel stehen.

Die Berechnung der Größe, die ein Text, z. B. ein Atomsymbol, einnimmt, wird für ein LaTeX-Ausgabeformat dem LaTeX-Compiler übertragen. Dieser wird mit einer temporären Datei aufgerufen, die alle im Reaktionsschema benutzten Textstücke enthält. Falls in diesen Texten Makros verwendet werden, müssen diese auch für die LaTeX-Bearbeitung der temporären Datei bekannt sein. OCHEM verwendet daher folgendes Verfahren: alle Makros, die Sie innerhalb von Textpositionen in Formeln verwenden, müssen Sie in einem (oder mehreren) separaten Paketen, z. B. `biochem.sty`, sammeln. Die Namen dieser Pakete werden über den `package`-Befehl von der temporären Datei geladen und somit verfügbar. Auch Paketoptionen können übergeben werden:

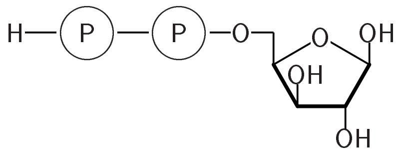
```
\begin{chemspecial}
package("biochem")
package("[biochem]chemmacros")
\end{chemspecial}
```

Ein Beispiel wäre ein P in einem Kreis, das in der Biochemie häufig als Symbol für aktiviertes Phosphat benutzt wird. Sie können das Symbol mit Hilfe eines LaTeX-Makros erzeugen. Die Makrodefinition schreiben Sie in eine Datei `biochem.sty`:

```
% biochem.sty

\newcommand{\Phosphat}
{{\setlength{unitlength}{1pt}
 \begin{picture}(20,20)
   \put(10,10){\makebox(0,0){P}\circle{20}}
 \end{picture}
}}
```

Das Makro können Sie dann in Formeln wie folgt verwenden:

```
\usepackage{biochem}

\begin{chemspecial}
package("biochem")
\end{chemspecial}

...

\begin{chemistry}
formula(C,C,"Xylose-4-diphosphat",HR,24)
{ ring("furanose",,,L)
  { 0: atom("O");
    1: bond(90) atom("O",C,R) atom("H",L);
    2: bond(-90) atom("O",C,R) atom("H",L);
    3: bond(90) atom("O",C,R) atom("H",L);
    4: bond(90;180) atom("O") bond(180)
         atom("\Phosphat",R,L) bond(180)
         atom("\Phosphat",R,L) bond(180)
         atom("H",R);
  }
}
\end{chemistry}
```



Xylose-4-diphosphat

Wie das Beispiel zeigt, muß das Paket mit den eigenen Makros auch im LaTeX-Dokument geladen werden, da dieses den Text in der endgültigen, vom Chemiecompiler errechneten Positionierung setzen muß und daher ebenfalls auf die Makrodefinitionen zugreift.

3.2.3 require

```
require(<File>)
```

Dieser Befehl lädt das Perl-Modul `<File>.pm`. Er muß vor dem ersten `schema`-Befehl stehen, kann jedoch mehrfach auftreten. Sein primärer Zweck ist nicht, „echte“ Perl-Module zu laden (wiewohl dies möglich wäre), sondern eigene Ringstrukturen, die als Perl-Code vorliegen müssen, verfügbar zu machen. Sie können so eine Vielzahl an seltener gebrauchten Strukturen bereitstellen, ohne jede einzelne im Compilercode verankern zu müssen und ihn so zu überfrachten. Auch ist es so möglich, Strukturen zu entwickeln, ohne automatisch eine eigene Version des Compilers zu erzeugen, Sie bleiben somit kompatibel und können den Compiler jederzeit gegen die neuste Version austauschen, ohne Ihrer Definitionen verlustig zu gehen.

Der Aufbau von Gerüststrukturen wird im Tutorial, Abschnitt 2.9, geschildert, einige wie `bicyclib.pm`, sind in der OCHEM-Distribution standardmäßig enthalten. Möchten Sie dieses Modul laden, um das Gerüst `tropin` (Abschnitt 3.4) zu laden, schreiben Sie

```
\begin{chemspecial}
  require("bicyclib")
\end{chemspecial}

\begin{chemistry}
  formula(C,C)
  { ring("tropin",,,L)
    { ... }
  }
\end{chemistry}
```

3.2.4 schema

```
schema("<name")
<sliste>
endschema
```

Dieses Kommando kennzeichnet Anfang und Ende einer Formel oder eines ganzen Reaktionsschemas. `<sliste>` wird als Einheit übersetzt und unter dem Dateinamen `<name>` im Ausgabeformat gespeichert. Der Befehl stellt die logische Grundeinheit des Chemiepaketes dar und korrespondiert zur `chemistry`-Umgebung auf LaTeX-Benutzerebene.

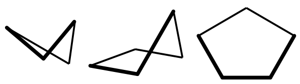
3.3 Die Bibliothek `mncyclib.pm`

In diesem Abschnitt werden die Ringstrukturen erläutert, die durch das Bibliotheksmodul `mncyclib.pm` verfügbar gemacht werden. Es wird durch die Zeilen

```
\begin{chemspecial}
  require("mncyclib")
\end{chemspecial}
```

im Vorspann eines LaTeX-Dokumentes verfügbar gemacht. Die folgenden Erklärungen folgen dem Schema, das bereits bei →`ring` angewandt wurde, für nähere Erklärungen siehe die Beschreibung des Befehls.

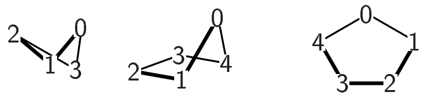
Die Bibliothek enthält die folgenden monozyklischen Strukturen und ihre Kodierung `<type>`:



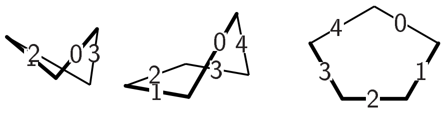
cb cp furanose

Sie können mit den Gerüsten dieser Bibliothek monozyklische Strukturen in schematisierter Stereodarstellung erzeugen.

Die Numerierung der Atome ist durch die folgenden Formeln beschrieben:



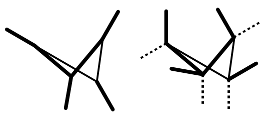
Die Numerierung der Bindungen erfolgt nach den folgenden Schemata



Die Bedeutung der beiden variablen Parameter `<p1>` und `<p2>` ist folgende:

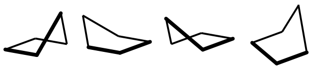
cb : (Zyklobutan-Gerüst) Der Parameter `<p1>` entfällt, `<p2>` stellt den Drehwinkel des gesamten Gerüstes dar.

Die Anordnung der Bindungen bei den Richtungstypen `r`, `t` und `b` wird in den nachfolgenden Formeln gezeigt. `t`-Bindungen werden durch dicke, `b`-Bindungen durch gestrichelte Linien symbolisiert.



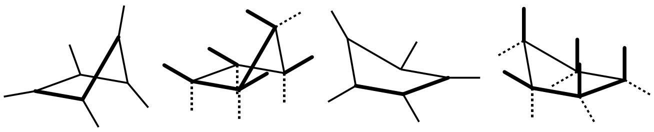
radial top/bottom

cp : (Zyklopentan-Gerüst) Der Parameter `<p2>` stellt wiederum den Gerüstdrehwinkel dar. `<p1>` ermöglicht die Auswahl zwischen den folgenden Konformeren:

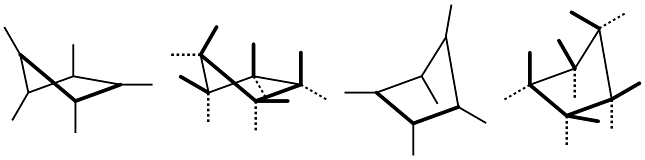


p1=0 p1=0 p1=0 p1=0

Die Anordnung der Bindungen bei den Richtungstypen `r`, `t` und `b` wird in den nachfolgenden Formeln gezeigt. `t`-Bindungen werden durch dicke, `b`-Bindungen durch gestrichelte Linien symbolisiert.



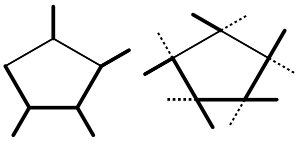
radial top/bottom radial top/bottom



radial top/bottom radial top/bottom

furanose : (Furanose-Gerüst) Die Parameter sind hier bedeutungslos.

Die Anordnung der Bindungen bei den Richtungstypen `r`, `t` und `b` wird in den nachfolgenden Formeln gezeigt. `t`-Bindungen werden durch dicke, `b`-Bindungen durch gestrichelte Linien symbolisiert.



radial top/bottom

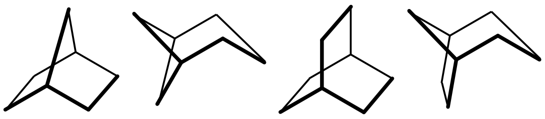
3.4 Die Bibliothek `bicyclib.pm`

In diesem Abschnitt werden die Ringstrukturen erläutert, die durch das Bibliotheksmodul `bicyclib.pm` verfügbar gemacht werden. In einem LaTeX-Dokument laden Sie es mit den Zeilen

```
\begin{chemspecial}
  require("bicyclib")
\end{chemspecial}
```

im Vorspann. Die folgenden Erklärungen folgen dem Schema, das bereits bei →`ring` angewandt wurde, für nähere Erklärungen siehe die Beschreibung des Befehls.

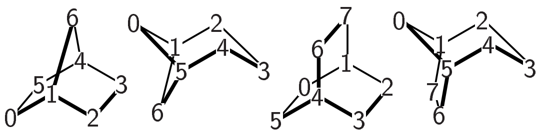
Die Bibliothek enthält die folgenden bizyklischen Strukturen und ihre Kodierung `<type>`:



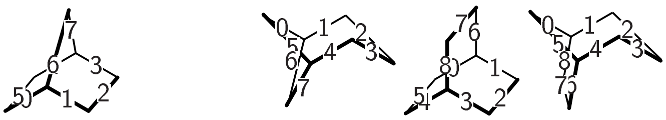
`bc221h` `bc311h` `bc222o` `bc321o`

Die Kodierung stellt dabei eine Abkürzung für **B**i**c**yclo[**n**.**m**.**o**]**h**eptan und **B**i**c**yclo[**n**.**m**.**o**]**o**ktan dar. Sie erfassen mit den Strukturen die Gerüste von Tropin, Norpinan und Norbornan.

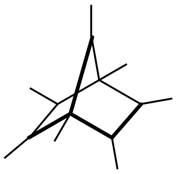
Die Numerierung der Atome ist durch die folgenden Formeln beschrieben:

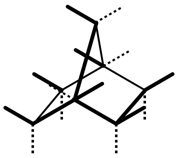


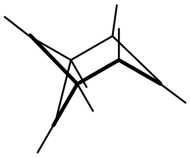
Die Numerierung der Bindungen erfolgt nach den folgenden Schemata.

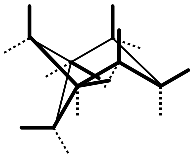


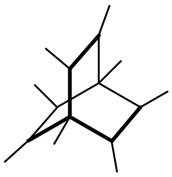
Die Anordnung der Bindungen bei den Richtungstypen `r`, `t` und `b` wird in den nachfolgenden Formeln gezeigt. `t`-Bindungen werden durch dicke, `b`-Bindungen durch gestrichelte Linien symbolisiert.

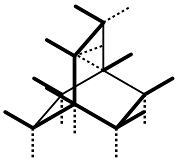

radial

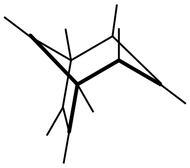

top/bottom

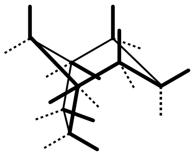

radial


top/bottom


radial


top/bottom


radial


top/bottom

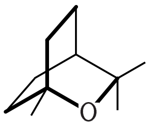
Die Bedeutung der beiden variablen Parameter <p1> und <p2> ist folgende:

bc221h : (Norbornan-Gerüst) Beide Parameter entfallen ersatzlos. (Nach neuen IUPAC-Regeln muß dieses Gerüst als 8,9,10-Trinorbornan bezeichnet werden.)

bc311h : (Norpinan-Gerüst) Beide Parameter entfallen ersatzlos.

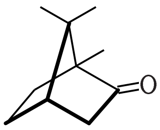
bc321o : (Tropin-Gerüst) Beide Parameter entfallen ersatzlos.

Nachfolgend sind einige bizyklische Naturstoffe aus den Gerüsten der Bibliothek aufgebaut worden.



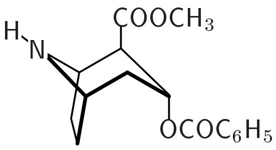
Cineol

```
formula(C,C,"Cineol",HA,48)
{ ring("bc222o",,,L)
  { 2: bond(t);  2: bond(b);
    3: atom("O");
    4: bond(r);
  }
}
```



Campher

```
formula(C,C,"Campher",HA,48)
{ ring("bc221h",,,L)
  { 3: bond(r,=C) atom("O");
    4: bond(r);
    6: bond(t);  6: bond(b);
  }
}
```



Kokain

```
formula(C,C,"Kokain",HA,48)
{ ring("bc321o",,,L)
  { 2: bond(t) atom("C",C,R) atom("OOCH$_3$",L);
    3: bond(b) atom("O",C,R) atom("COC$_6$H$_5$",L);
    0: atom("N") bond(r) atom("H");
  }
}
```

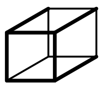
3.5 Die Bibliothek `polycyclib.pm`

In diesem Abschnitt werden die polyzyklischen Ringstrukturen erläutert, die durch das Bibliotheksmodul `polycyclib.pm` verfügbar gemacht werden. In einem LaTeX-Dokument laden Sie es mit den Zeilen

```
\begin{chemspecial}
  require("polycyclib")
\end{chemspecial}
```

im Vorspann. Die folgenden Erklärungen folgen dem Schema, das bereits bei →`ring` angewandt wurde, für nähere Erklärungen siehe die Beschreibung des Befehls.

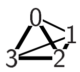
Die Bibliothek enthält die folgenden polyzyklischen Strukturen und ihre Kodierung `<type>`:

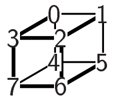



tetrahedrane cubane

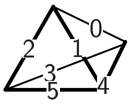
Sie erfassen mit den Strukturen die Gerüste von Tetrahedran und Kuban.

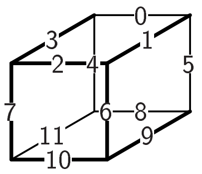
Die Numerierung der Atome ist durch die folgenden Formeln beschrieben:



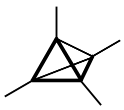
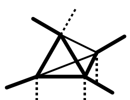
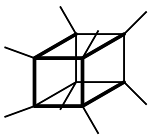
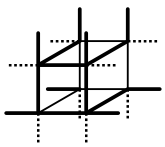


Die Numerierung der Bindungen erfolgt nach den folgenden Schemata.





Die Anordnung der Bindungen bei den Richtungstypen `r`, `t` und `b` wird in den nachfolgenden Formeln gezeigt. `t`-Bindungen werden durch dicke, `b`-Bindungen durch gestrichelte Linien symbolisiert.

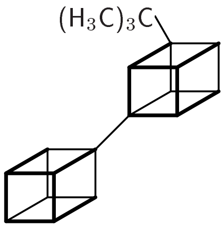

radial

top/bottom

radial

top/bottom

Die Bedeutung der beiden variablen Parameter `<p1>` und `<p2>` ist folgende:

tetrahedrane : Der Parameter <p1> ist bedeutungslos, wohingegen <p2> den Drehwinkel des gesamten Gerüstes darstellt.

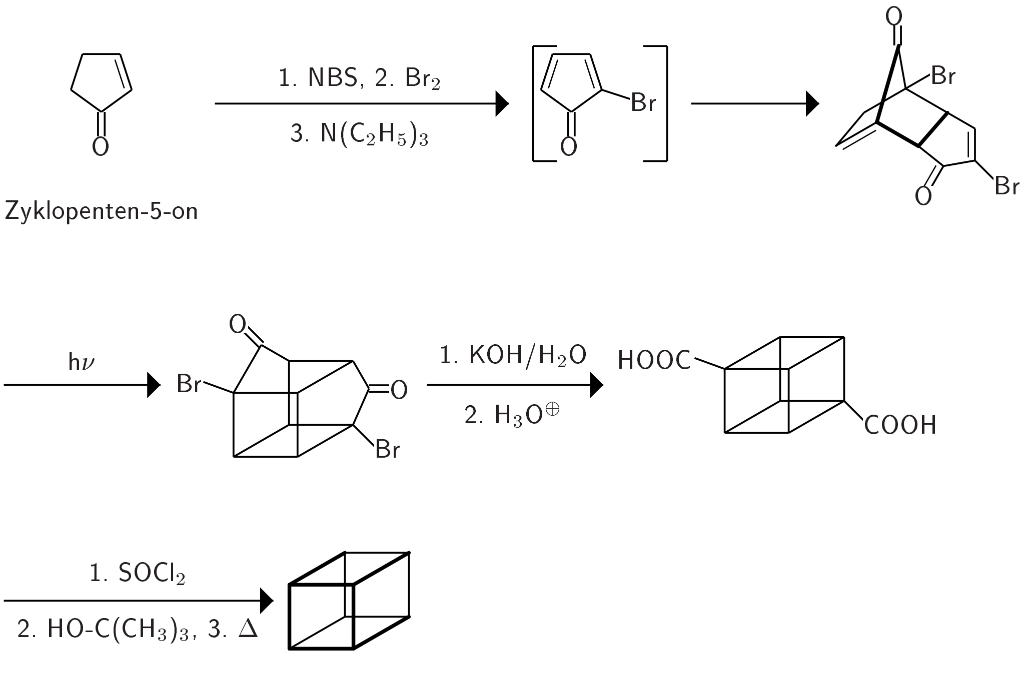
cubane : Der Parameter <p1> ist bedeutungslos, wohingegen <p2> den Drehwinkel des gesamten Gerüstes darstellt.

Als Beispiel folgt die Formel des 2-*tert*-Butylcubylcubans.




```
\begin{chemistry}
    ring("cubane",,,L){
      1: bond(r,,L)
         ring("cubane",7,,L,,0){
             0: bond(r) atom("(H$_3$C)$_3$C",R);
         };
    }
```

Als umfangreicheres Beispiel folgt die Synthese des Kubans:



Kuban

```
multiline(3,L)
{ % upper line
  formula(L,R,"Zyklopenten-5-on"){
    ring(,,H3=,,5,-90){
      0: bond(r,=C) atom("O");
    }
  }
  arrow(){
    text(T,C){ formula(C,C){ atom("1. NBS, 2. Br$_2$") }}
    text(B,C){ formula(C,C){ atom("3. N(C$_2$H$_5$)$_3$") }}
  }
  space(R)
```

```
  formula(L,R){
    ring(,,H1=3=,,5,-90){
      0: bond(r,=C) atom("O");
      4: bond(r) atom("Br",L);
    }
  }
  bracket()
  space(R)
  arrow(){}
  formula(L,R){
    ring("bc221h",,0=,L){
      6: bond(r,=C) atom("O");
      4: bond(r) atom("Br",L);
      3: bond(-30;-90,=) branch { bond(-45) atom("Br",L); } bond(-170)
         saveXY(#1) bond(-120,=) atom("O");
      2: bond(#1);
    }
  }
  ;

  % middle line
  arrow(){
    text(T,C){ formula(C,C){ atom("h$\nu$") }}
  }
  formula(L,R){
    ring("cubane",,1-2-3s5s6-7-9-10-11-,NN){
      3: saveXY(#1); 0: bond(150) branch { bond(135,=C) atom("O"); } bond(#1);
      3: bond(r) atom("Br", R);
      5: saveXY(#2); 1: bond(-60) branch { bond(0,=C) atom("O"); } bond(#2);
      5: bond(r) atom("Br", L);
    }
  }
  arrow(,5){
    text(T,C){ formula(C,C){ atom("1. KOH/H$_2$O") }}
    text(B,C){ formula(C,C){ atom("2. H$_3$O$^\oplus$") }}
  }
  formula(L,R){
    ring("cubane",,1-2-3-6-7-9-10-11-,NN){
      3: bond(r) atom("HOOC", R);
      5: bond(r) atom("COOH", L);
    }
  }
  ;

  % lower line
  arrow(,5){
    text(T,C){ formula(C,C){ atom("1. SOCl$_2$") }}
    text(B,C){ formula(C,C){ atom("2. HO-C(CH$_3$)$_3$, 3. $\Delta$") }}
  }
  formula(L,R,"Kuban"){
    ring("cubane",,,NN){}
  }
  ;
}
```

3.6 Das Includefile natur.inc

Dem Chemiepaket liegt eine kleine Includedatei natur.inc bei (Listing in Abschnitt ??), in der eine gewiße (größer werdende) Anzahl an Makros zusammengefaßt ist, die die Arbeit mit bestimmten Verbindungsklassen, vor allem Naturstoffen, erleichtern sollen. Sie wird wie folgt in ein LaTeX-Dokument eingebunden:

```
\begin{chemspecial}
include('natur.inc')
\end{chemspecial}
```

Es ist wahrscheinlich, daß Sie die Makros in der vorliegenden Form noch nicht besonders nützlich finden; da es aber schwer ist, Makros für alle denkbaren Anwendungsfälle zu schreiben, ohne die Makros mit 50 Parametern zu versehen, schlage ich vor, sie als Fundus zu betrachten und für die eigenen Bedürfnisse Anpassungen vorzunehmen. Wie immer, sind auch in dieser Beziehung Anregungen sehr willkommen, um für nachfolgende Distributionen eine eventuell sinnvollere Datei zu liefern:-)

Die einzelnen Makros werden im folgenden besprochen. Sie erzeugen jeweils ein typisches Gerüst für eine bestimmte Verbindungsklasse. Wichtige Schlüsselpositionen werden dabei abgespeichert und können für Modifikationen mittels restoreXY angesprungen werden; die Nummern der betreffenden Punkte wird jeweils im folgenden Text mitgeteilt. Einige Makros erwarten Parameter, die das Grundgerüst ebenfalls modifizieren, zum Beispiel androstane oder cholane für das Steroidmakro, um Androstan- oder Cholangerüste zu erzeugen. Weitere Modifikationen sind möglich, indem Bindungslisten für Ringe als Parameter übergeben werden. Schließlich können kleine Formelbeschreibungen als Parameter auftreten. Anhand der Beispiele sollten Sie jedoch die Anwendung der Makros verstehen können.

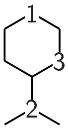
Beachten Sie, daß diese Makros nicht beliebig verallgemeinert sind, es werden daher immer Strukturen auftreten, die nicht unmittelbar mit diesen Makros zu setzen sind; in diesen Fällen können Sie das Makro kopieren und modifizieren oder – in komplizierten Fällen – die ganze Struktur ohne Makro beschreiben, was unter Umständen einfacher sein kann.

TERPEN

Dieses Makro erzeugt ein Menthangerüst, das als Grundlage für einige monozyklische Monoterpene dienen kann. Die allgemeine Form lautet

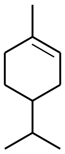
```
TERPEN_(<bliste>, <C2-Subs>)
```

<bliste> ist dabei die Bindungsliste für den Ring, <C2-Subs> ist die Beschreibung eines Substituenten an C^2. Ist kein Substituent vorhanden, schreiben Sie einfach ‘’ als leere Liste. Die Positionen von C^1, C^8 und C^3 werden in den Koordinaten Eins bis Drei gespeichert. Die folgende Formel zeigt Grundgerüst und Lage der gelieferten Punkte:



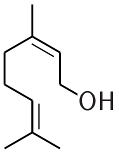
TERPEN_

Einige Beispiele zeigen die Anwendung der Bindungsliste, des Substituenten sowie der gespeicherten Punkte:



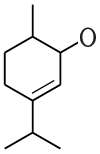
1-p-Menthen

```
formula(L,R,"1-p-Menthen",HR,24)
{ TERPEN_(H3=) restoreXY(#1) bond(90) }
```



Nerol

```
formula(L,R,"Nerol",HR,24)
{ TERPEN_(H3=5s)
  restoreXY(#1) bond(90)
  restoreXY(#3) bond(-30) atom("O",C,R) atom("H",L)
  restoreXY(#2) bond(90,=)
}
```



Carvenon

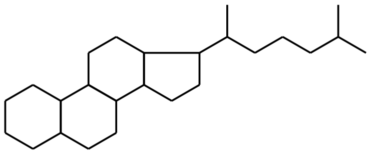
```
formula(L,R,"Carvenon",HR,24)
{ TERPEN_(H5=,bond(30) atom("O"))
  restoreXY(#1) bond(90)
}
```

STEROID

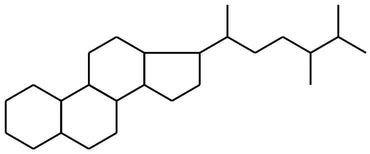
Dieses Makro hat den Aufbau

```
STEROID_(<bliste>, <type>)
```

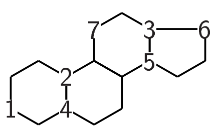
und erzeugt Steroidstrukturen gemäß `<type>`. Die möglichen Typen entsprechen den wichtigsten Grundkohlenwasserstoffen und sind in der folgenden Aufstellung enthalten. Die Ziffern an manchen Atomen entsprechen der Nummer des betreffenden gespeicherten Koordinatenpaares:



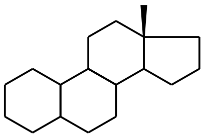
cholestane



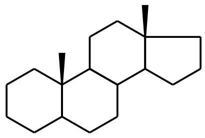
ergostane



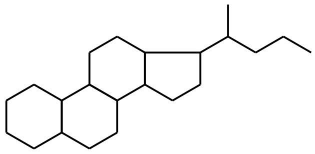
STEROID_



estrane



androstane

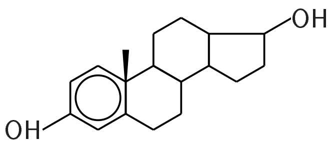


cholane

<bliste> ist eine Bindungsliste, die zum Ring A gehört. Mit ihrer Hilfe kann der Ring aromatisiert oder mit Doppelbindungen versehen werden. Zur weiteren Modifikation können die folgenden Koordinaten benutzt werden, die auch in der Übersicht oben durch Ziffern wiedergegeben wurden:

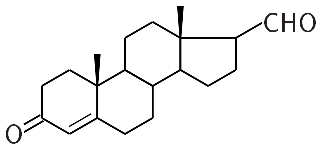
Nummer	Atom	Nummer	Atom
1	C^{3}	2	C^{10}
3	C^{13}	4	C^{5}
5	C^{14}	6	C^{17}
7	C^{11}		

Einige Beispiele sollen die Anwendung zeigen.



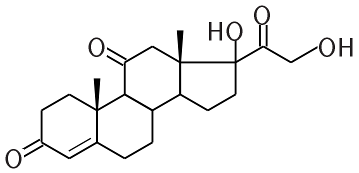
Östradiol

```
formula(L,R,"{}""Ostradiol",HR,24)
{ STEROID_(O)
  restoreXY(#2) bond(90,<<)
  restoreXY(#1) bond(-150) atom("OH",R)
  restoreXY(#6) bond(30) atom("OH",L)
}
```



Testosteron

```
formula(L,R,"Testosteron",HR,24)
{ STEROID_(H5=, androstane)
  restoreXY(#1) bond(-150,=C) atom("O")
  restoreXY(#6) bond(30) atom("CHO",L)
}
```



Cortison

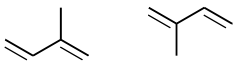
```
formula(L,R,"Cortison",HR,24)
{ STEROID_(H5=, androstane)
  restoreXY(#1) bond(-150,=C) atom("O")
  restoreXY(#7) bond(150,=C) atom("O")
  restoreXY(#6) bond(30) branch{ bond(90,=C) atom("O"); }
                bond(-30; 30) atom("OH",L)
  restoreXY(#6) bond(90) atom("O",C,L) atom("H",R)
}
```

ISOPREN

Dieses Makro erzeugt eine Isopreneinheit in zwei verschiedenen Formen. Sie dient zum Aufbau von Polyprenketten Die allgemeine Form lautet

```
ISOPREN_(<bond_len>)
ISOPREN1_(<bond_len>, <bond_type>)
```

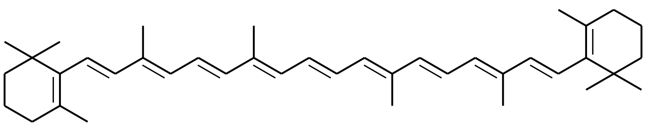
`<bond_len>` ist die Bindungslänge einer Bindung, die zum Startpunkt der eigentlichen Einheit führt. Sie muß mit der Angabe '0' auf Null gesetzt werden für eine isolierte Einheit und kann mit '' leer gelassen oder ignoriert werden für den Fall aneinanderhängender Einheit für Polyprene. Mit `<bond_type>` können Sie die Bindungsform für eine der Doppelbindungen angeben (Verwendung bei der terminalen ψ-Einheit des Carotens). Die folgenden Formeln zeigen die Grundgerüste mit `<bond_len>` gleich Null:



ISOPREN_ ISOPREN1_

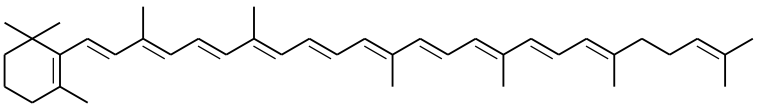
```
formula(L,R,"I{}SOPREN\_",HR,24)
{ ISOPREN_(0) }
formula(L,R,"I{}SOPREN1\_",HR,24)
{ ISOPREN1_(0) }
```

Einige Beispiele zeigen die Anwendung der Einheiten zum Aufbau von Carotenen:



β,β-Caroten

```
formula(T,B,"$\beta,\beta$-Caroten",HR,24)
{ ring(,,H4=)
  { 3: bond(t); 3: bond(b);
    5: bond(r);
    4: ISOPREN_
       ISOPREN_ bond(r; r/,=; r)
       ISOPREN1_
       ISOPREN1_ ring(,0,H0=,,,r)
                 { 1: bond(r);
                   5: bond(t); 5: bond(b);
                 };
  }
}
```



β,ψ-Caroten

```
formula(T,B,"$\beta,\psi$-Caroten",HR,24)
{ ring(,,H4=)
  { 3: bond(t); 3: bond(b);
                5: bond(r);
                4: ISOPREN_
                ISOPREN_ bond(r; r/,=; r)
                ISOPREN1_
       ISOPREN1_
       ISOPREN1_(,‘-’)
       bond(r/,=) branch { bond(r/t); } bond(r);
  }
}
```

AA

Dieser Baustein erleichtert die Darstellung von Aminosäuren in einer textuellen Art und Weise ohne Berücksichtigung der Stereostruktur. Die allgemeine Syntax

```
AA_(<N-Terminus>, <C-Terminus>, <Seitenkette>)
```

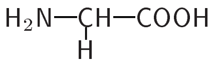
sieht zwei Parameter vor, die den N- beziehungsweise C-Terminus beschreiben. Die Angaben `H` beziehungsweise `OH` erzeugen die terminalen Sequenzen H_2N- und -COOH. Diese Positionen können frei bleiben und erlauben auf diese Weise die Darstellung von Oligopeptiden. Sie können aber auch beliebige Formelsequenzen enthalten, um ungewöhnliche Endgruppen zu zeichnen. Der dritte Parameter erzeugt die Seitenkette der Aminosäure.

```
<N-Terminus>NH—CH—CO<C-Terminus>
                |
                <Seitenkette>
```

AA_

Die folgenden Beispiele zeigen den Einsatz des Makros. Zur besseren Übersicht wird dabei die Includedatei `utils.inc`, die im Abschnitt 3.7 beschrieben wird, eingesetzt.

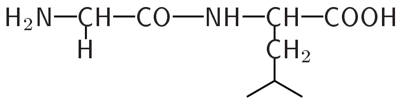
Die Grundform einer Aminosäure wird durch die Termini H und OH erreicht:



Glycin

```
formula(L,R,"Glycin",HR,24)
{ AA_(H,OH,atom("H")) }
```

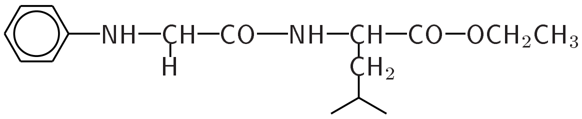
Für ein einfaches Dipeptid darf die erste Aminosäure keinen C-Terminus, die zweite keinen N-Terminus erhalten:



H-Gly-Val-OH

```
formula(L,R,"H-Gly-Val-OH",HR,24)
{ AA_(H,,atom("H"))
  bond(0)
  AA_(,OH,branch{atom("C",C,R) atom("H$_2$",L);} iPr_(-90))
}
```

Um eine ungewöhnliche Absättigung der terminalen Atome anzuzeigen, können auch vollkommen andere Formelelemente spezifiziert werden:



Ph-Gly-Val-OiPr

```
formula(L,R,"Ph-Gly-Val-OiPr",HR,24)
{ AA_(ring(,0,,,,0){} bond(0),,atom("H"))
  bond(0)
  AA_(,bond(0) OEt_,
    branch{atom("C",C,R) atom("H$_2$",L);}
    iPr_(-90))
}
```

3.7 Das Includefile `utils.inc`

Eine weitere Includedatei ist `utils.inc` bei (Listing in Abschnitt ??), in der sehr kleine Makros zu finden sind, die immer wiederkehrende Schreibarbeiten abkürzen helfen. Sie wird wie folgt in ein LaTeX-Dokument eingebunden:

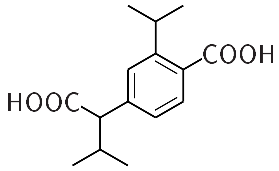
```
\begin{chemspecial}
include(‘utils.inc’)
\end{chemspecial}
```

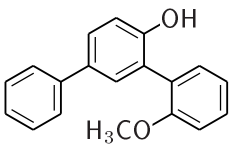
Sie enthält die folgenden Makros.

```
Me_             % erzeugt -CH3
eM_             % dto H3C-
Et_             % erzeugt -CH2CH3
tE_             % dto H3CH2C-
iPr_(<dir>)     % dto -CH(CH3)2

OH_               % erzeugt -OH
HO_               % dto HO-
NH2_              % erzeugt -NH2
H2N_              % dto H2N-
NH2p_             % erzeugt -N(+)H2
pH2N_             % dto H2N(+)-
NMe2_             % erzeugt -N(CH3)2
Me2N_             % dto (H3C)2N-
NMe2p_            % erzeugt -N(+)(CH3)2
pMe2N_            % dto (H3C)2N(+)-
OMe_              % dto -OCH3
MeO_              % dto H3CO-
OEt_              % dto -OCH2CH3
EtO_              % dto H3CH2CO-

COOH_             % dto -COOH
HOOC_             % dto HOOC-
COOMe_            % dto -COOCH3
MeOOC_            % dto H3COOC-
COOEt_            % dto -COOCH2CH3
EtOOC_            % dto H3CH2COOC-

SO3H_             % dto -SO3H
HO3S_             % dto HO3S-
SO3m_             % dto -SO3(-)
mO3S_             % dto (-)O3S-

Ph_{...}          % Phenyl
Ph_(<angle>){...} % Phenyl
```

Der Rumpf von `Ph_` entspricht dem Rumpf eines normalen `ring`-Befehls und kann alle dort erlaubten Operationen enthalten oder leer bleiben. `iPr_` erzeugt einen abzweigenden Isopropylrest, dessen Richtung mit dem Argument festgelegt werden muß. Dieses besteht aus einer Winkelangabe.

Einige Beispiele zeigen die Anwendungsmöglichkeiten. Das Makro zur Erzeugung des Phenylringes weist einen optionalen Parameter zur Angabe des Drehwinkels des Ringes auf, der zur Darstellung von Benzen als Grundkörper der Struktur verwendet wird, in den Beispielen 90°. Der Defaultwinkel `r` für diesen Parameter führt zur Darstellung von Phenylsubstituenten:



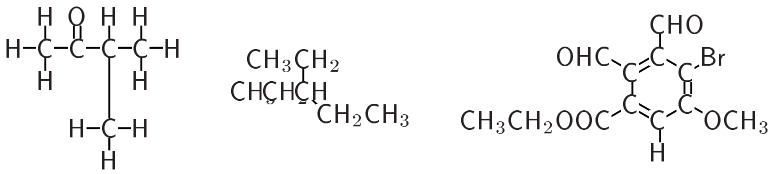



```
formula(L,R)
{ Ph_(90)
  { 0: iPr_(r);
    1: bond(r) COOH_;
    4: bond(r) iPr_(rt) bond(r/) HOOC_; }
}
formula(L,R)
{ Ph_(90)
  { 1: bond(r) OH_;
    4: bond(r) Ph_{};
    2: bond(r) Ph_{ 5: bond(r) MeO_; };
  }
}
```

3.8 Includefiles für Strukturen nach KEKULE: condensed.inc und cyclohexanes.inc

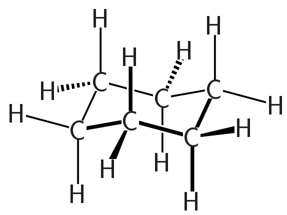
Diese Includedateien wurden freundlicherweise von Herrn Dr. Kenvard Vaughan bereitgestellt und sind im Unterverzeichnis `kv` zu finden. Sie werden wie die anderen Includedateien auch eingebunden und enthalten Makros, die den Satz von Strukturformeln nach KEKULE bedeutend vereinfachen. Beispiele aus der Dokumentation:

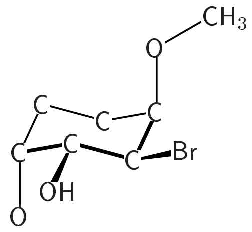
```
\begin{chemspecial}
define(‘bndlen_’, ‘N’)
include(‘condensed.inc’)
include(‘cyclohexanes.inc’)
\end{chemspecial}

\begin{chemistry}
  formula(L,R){_h(0) hk_(_h) co_ hk_(_h, bond(l,,L) vk_(_h) _h) hk_(_h) _h}
  space(R) space(R)
  formula(L,R){_ch3(,0) ch2_ ch_(,90, _ch2ch3(-), -45) _ch2ch3 }
  space(R) space(R)
  formula(L,R){_cho(-,-30) _kPh(_cho,_br,_och3,,_cooch2ch3(-)) }
\end{chemistry}
```



Beispiele für Zyklohexane:

```
\begin{chemistry}
  defaulz
  formula(L,R){ chex_(C)
     c1_(_h,_h)
     c2_(_h,_h)
     c3_(_h,_h)
     c4_(_h,_h)
     c5_(_h,_h)
     c6_(_h,_h)
  }
  space(R) space(R) space(R) space(R)
  formula(L,R){ chex_(C)
     c1_(_o)
     c2_(,_oh)
     c3_(,_br)
     c4_(o_(r) _ch3)
  }
\end{chemistry}
```





Zu den beiden Includedateien existieren die ausführlichen Dokumentationen `condensed_doc.ps` und `cyclohexanes_doc.ps`.

A. Das Programm

A.1 History

20-OCT-1999 : (`chemie.pl` version 1.0a 1999-10-20, `be.pm`)
Setting of `\unitlength` enclosed in a group.

09-JAN-2000 : (`ochem.sty` version 3-0b 2000-01-09, `streambuf.pm`)
Additional bond types `>>`, `>.` and `b`.

17-JAN-2000 : (`mncyclib.pm` version 1.1)
`furanose` ring type for saccarid chemistry.

19-JAN-2000 : (`ochem.sty` version 3-0c 2000-01-19, `streambuf.pm`, `chemie.pl`)
Additional bond type `~`. The length of a bond can now be marked by a various number of markers N,n,L,l,S and s, representing additive terms. Command `package` allows to pass packages to the intermediate LaTeX file, supporting macros in text position of `atom`.

19-APR-2000 : Some undesired spaces are eliminated thanks to S. Seckinge, Freiburg (`ochem.sty` version 3-0d 2000-04-19, `be.pm`).

10-APR-2001 : Creation of EPS, JPG and PNG supplied. Minor changes in the perl code.

A.2 Danke schön!

Einigen Benutzer (Herr Dr. Kenward Vaughan sowie Herrn Oliver Bürschaper) verdanke ich wichtige Anregungen (fehlende Bindungstypen, Furanoseringe, die Möglichkeit, Makros in `atom`-Befehlen einzusetzen) und Meldungen von Fehlern.

Herrn Dr. Kenward Vaughan danke ich fernerhin für seinen Einsatz bei der Entwicklung der Makros zur Darstellung von Formeln in der Kekule-Darstellung, die im Unterverzeichnis `kv` enthalten sind.

B. Mehr Stoff für Chemiesüchtige

Wer Formeln zeichnet, muß sie möglicherweise auch benennen, was unter Umständen beliebig diffizil werden kann. Auf den Internetseiten der Firma `acdlabs` ist jedoch eine online-Version des IUPAC-Reglements zu finden:

`http://www.acdlabs.com/iupac/nomenclature/`

Eine schöne Einführung in die korrekte Nomenklatur organischer Verbindungen ist D. Hellwinkel, *Die systematische Nomenklatur organischer Verbindungen*, Springer-Verlag Heidelberg.

Als WYSIWYG-Konkurrenzprodukt zum vorliegenden LaTeX-Paket ist das chemische Zeichenprogramm ChemSketch der oben genannten Firma als Freeware-Version für Dosenrechner verfügbar:

`http://www.acdlabs.com/download`

Es kann allerdings nicht nur Formeln zeichnen (sehr schön sogar), sondern auch verschiedene Berechnungen durchführen und dreidimensionale Stab- oder Kugerl-(Kalotten)modelle zeichnen. Außerdem besitzt es einen umfangreichen Katalog mit Substanzen, Eigenschaften ....

Index