

Ingo Klöckl



User's manual OCHEM Version 4/0d

October 3, 2001

Ingo Klöckl
Institute for Nuclear Chemistry, University of Mainz
Fritz Strassmann-Weg 2
D-55099 Mainz
kloeckl@vkcmzd.chemie.uni-mainz.de
ingo.kloeckl@2k-software.de
Typeset with \LaTeX and OCHEM Version 4/0d.

OCHEM Version 4/0d
© 1999–2001 Ingo Klöckl
This program can be redistributed and/or modified under the terms of the LaTeX
Project Public License Distributed from CTAN archives in directory
macros/latex/base/lpp1.txt; either version 1 of the License, or any later ver-
sion.

Contents

1	Introduction	1
1.1	Some words of the translator ...	1
1.2	Overview	1
1.3	Installation	2
1.4	Post-installation steps	3
1.5	Usage in \LaTeX	4
1.6	Output formats	12
1.7	Macro definitions	13
1.8	Examples, the formula catalogue	14
1.9	Further perspective	15
2	Tutorial	17
2.1	Your first steps into a chemist's world	17
2.1.1	Simple formulae	17
2.1.2	Alignment of formulae	20
2.2	More complex formulae	21
2.2.1	Symbolic angles	22

2.2.2	Linear systems	26
2.2.3	Linear annellated rings	26
2.2.4	Manual construction of annellated systems	29
2.2.5	Angular annellated ring systems	31
2.2.6	(Carbon-)bridges	32
<hr/>		
2.3	Reaction chains	36
<hr/>		
2.3.1	Horizontally linear sequences	36
2.3.2	Multi-line schemes	39
2.3.3	Vertical alignment	41
2.3.4	Context and branching	43
2.3.5	Parts of reactions as units	44
2.3.6	Joining contexts	46
2.3.7	Merging of several branches	51
2.3.8	Mixing of basic elements	57
<hr/>		
2.4	Cyclohexane and carbohydrates	59
<hr/>		
2.5	Modification of parameters	65
<hr/>		
2.6	Depiction of electrons	66
<hr/>		
2.6.1	Configurations of electrons	66
2.6.2	Movements of electrons	67
<hr/>		
2.7	Cutting off bonds and molecule parts	69
<hr/>		
2.8	Examples	71
<hr/>		
2.8.1	Nylon synthesis	71
2.8.2	Terpene biosynthesis	74
<hr/>		
2.9	More compound classes	77
<hr/>		
2.10	Development of ring structures	77
<hr/>		
3	Alphabetic command reference	81
<hr/>		
3.1	User commands	81

3.1.1	arrow	82
3.1.2	atom	85
3.1.3	bond	87
3.1.4	bracket	95
3.1.5	branch	96
3.1.6	cutline	97
3.1.7	emove	98
3.1.8	fbox	99
3.1.9	formula	99
3.1.10	gotoXY	100
3.1.11	joinh	101
3.1.12	joinv	102
3.1.13	multiline	103
3.1.14	orbital	104
3.1.15	restore	104
3.1.16	restoreXY	105
3.1.17	ring	106
3.1.18	save	114
3.1.19	savecontext	114
3.1.20	saveXY	115
3.1.21	scale	118
3.1.22	set	119
3.1.23	setcontext	120
3.1.24	shiftXY	122
3.1.25	space	122

3.2	Special commands	123
-----	------------------	-----

3.2.1	font	124
3.2.2	package	124
3.2.3	require	125
3.2.4	schema	126

3.3	The library <code>mncyclib.pm</code>	127
-----	--------------------------------------	-----

3.4	The library <code>bicyclib.pm</code>	129
-----	--------------------------------------	-----

3.5	The library <code>polycyclib.pm</code>	131
-----	--	-----

3.6	The include file <code>natur.inc</code>	134
-----	---	-----

3.7	The include file <code>utils.inc</code>	140
-----	---	-----

3.8	Include files for KEKULE structures: <code>condensed.inc</code> and <code>cyclohexanes.inc</code>	142
-----	---	-----

A The program 143

A.1 History 143

A.2 A lot of thanks goes to ... 143

B More stuff for chemistry addicts 145

Index 147

The reaction schemes

1-1	Usage of ring structure, here the bicyclohexane, from a library. . . .	9
2-1	The positioning parameter describes the relations between the formulae. Left: formulae aligned left to right, right: formulae aligned top to bottom.	21
2-2	Horizontal reaction scheme: synthesis of a benzothiophene.	37
2-3	Horizontal reaction scheme with compound labels which are aligned toward a common line underneath the reaction arrow (absolute distance of the text to the center-line).	38
2-4	Horizontal reaction scheme with compound labels which are automatically aligned in a constant distance on a common line underneath the reaction arrow.	39
2-5	Synthesis of phenanthrene, set in multiple lines with <code>multiline</code> . . .	41
2-6	Vertical reaction scheme: synthesis of 7-Oxononanic acid. On the right the same scheme with labeling of the compounds. All texts possesses a fixed distance from the lower formula edges.	42
2-7	What happens to the borane? Multiple branchings, derived from one and the same formula, can show it. The context of the branching point, here the borane, must be stored.	45
2-8	An independent vertical scheme is set as a single-line <code>multiline</code> object and can thus be inserted as a unity into the horizontal formula stream.	46
2-9	A super-context, built up with <code>multiline</code> , permits to bracket all formulae, contained in the super-context.	47
2-10	Branchings can be applied to each single formula independently from the super-context, if the contexts are stored separately.	48
2-11	With <code>multiline</code> aligned formula titles and branching, attained with individual contexts.	49
2-12	The synthesis of a crown-ether as an example for the merging of two horizontal chains with <code>joinh</code>	52
2-13	Horizontal merging of two horizontally merged reaction sequences.	53
2-14	Synthesis of Chrysanthemums" aure, arranged with <code>joinv</code>	56
2-15	Synthesis of a diazo compound. Mixed horizontal and vertical merging.	57
2-16	Mixed <code>joinv/joinh</code> commands.	58
2-17	Merging of horizontal reaction sequences with <code>multiline</code>	59

2-18 The balance between the two main conformers of cyclohexane uses the various representations of the same ring type.	65
2-19 Main reactions can be separated from minor reactions by a thicker line. The line width can be chosen by the command <code>set</code>	66
2-20 The chain-extending step of the biosynthesis of isoprenoids is demonstrated by electron shifts.	69
2-21 The electron movement illustrated the process of the Friedel-Crafts-acylation of an aromatic.	69
2-22 Nylon synthesis serves as an example for a complex scheme.	75
2-23 Biosynthesis of bicyclic monoterpenes out of the straight-chain precursor farnesol.	78
3-1 Alignment of two text blocks above and below an arrow.	83
3-2 Positioning of the text blocks above and below an arrow, due to different angles of the arrow.	84
3-3 Complete reaction chains act like labels if they are set above arrows.	84
3-4 Several reaction chains branch from one formula. The context of this formula must be stored.	121
3-5 The synthesis of cubane.	133

1. Introduction

1.1 Some words of the translator . . .

This translation was made by a person who never wishes to be made responsible for it. The epitaph is:

Моя хата с краю
я ничево не знаю

(Lots of thank to this unknown person from the author of OCHEM for the nice translation which could never be achieved by the OCHEM programmer!)

1.2 Overview

This package is intended to support the typesetting of organic structural formulae and reaction schemes with \LaTeX . The aim of this package is to provide a \TeX -like mechanism for positioning individual formulae within a scheme by giving a simple description of the geometry of the formulae. In short, this package provides a “chemistry compiler” which should do all the painstaking work for you. During a first \LaTeX run, the chemical descriptions are written into an auxiliary text file. This file is then compiled and produces \TeX fragments containing the formulae, which then in turn will be read during a second \LaTeX run. Thus, the graphic fragments automatically replace the descriptions in the output \LaTeX document. Depending on the chosen output format (LaTeX, PSLATEX and PS), the compiler produces text fragments consisting of pure \LaTeX , \LaTeX with embedded PostScript code or pure PostScript.

Changing the chemistry material’s description requires all the steps depicted above. If changes do not influence the chemistry descriptions, the compiled graphic fragments can easily be reused.

The main focus of this package and its compiler lies on planar representations of simple text formulae, line drawings as well as simple and common three-dimensional stereo formulae like chair conformations of cyclohexane or bridges in polycyclic compounds like morphinee. A geometric description of very complicated structures, e. g. polycyclic alkaloids, can cause problems. Beside the common bond angles and lengths, all others have to be calculated individually, and extensive use of `saveXY` and `restoreXY` is necessary. Quite unfortunately, OCHEM is not a molecular modelling tool (sniff). The author welcomes any suggestions or comments for this package; especially if it is felt that some basic elements for a proper and easy description of structural elements are missing.

Chemists who need to show the reaction mechanisms may use small arrows to il-

illustrate the movement of electrons. In order to draw such arrows, it becomes clear that there are as many commands which have to be typed in a text editor as there are countless mouse movements using WYSIWYG editors. In both ways, the size of the formula description grows considerably. So, if anybody makes a good suggestion to solve this nuisance, you are heartily welcome (please don't forget the implementation).

The formulae are embedded with maximum output quality as a fragment with PostScript code in the \LaTeX files. This means, that they are visible only with a PostScript viewer and printer. If a PostScript printer is not available, the PostScript previewer `ghostview` can be used to convert the output file in another, more suitable format. `ghostview` may also be used for printing the PostScript documents on any ink jet printer.

Texts within formulae are set with \LaTeX and require therefore no PostScript. As the graphical capabilities of \LaTeX are truly limited, the page description language PostScript becomes a natural choice. The compiler produces a generic output that is translated by the module `be.pm` into several final formats. There is also a \LaTeX output format that uses only \LaTeX means, but because of the facts mentioned above, it produces no high-quality formulae (you may compare for yourself in which cases the output is acceptable). It is intended to improve this native \LaTeX output format in connection with the package `eepic`.

If you do not want to use PostScript documents in conjunction with `ghostview`, a truly good way is to use \LaTeX text with PostScript drawing (default output mode) and produce the formulae as EPS files which subsequently can be included in another \LaTeX document with `\includegraphics`. In this way, you would have one document, containing only your formula descriptions and a second one with the real text, including the EPS pictures. (You may also choose the generated PNG or JPG pictures for other publication media.)

1.3 Installation

To install OCHEM, unzip the distribution into a temporary directory, which can be removed after the installation process has been completed. Take care that your UNZIP program keeps the directory structure. If you have unzipped the files, go to the `install` subdirectory. Windows users may perform the following steps:

```
c:\> mkdir tmp
c:\> cd tmp
c:\tmp> copy <source>chdist.zip .
c:\tmp> pkzip25 -extra -dir chdist
c:\tmp> cd install
```

while UNIX users might type

```
/tmp> mkdir tmp
/tmp> cd tmp
/tmp/tmp/> cp <source>chdist.zip .
/tmp/tmp/> unzip chdist
/tmp/tmp/> cd install
```

Now edit the file `install.cfg` using an editor, e. g. Windows NOTEPAD or emacs or whatever editor you prefer. Change all entries in the configuration file according to your intended directory structure. After editing the configuration file, start the installation process, for Windows use:

```
c:\tmp\install\> edit install.cfg.WIN32
c:\tmp\install\> install.pl WIN32
c:\tmp\install\> cd ..\..
c:\rd tmp/s
```

Unix users may type

```
/tmp/tmp/install\> vi install.cfg.UNIX
/tmp/tmp/install\> install.pl UNIX
/tmp/tmp/install\> cd ../..
/tmp> rm -r tmp
```

The chemistry compiler consists of a few perl scripts and modules. At least the scripts should be placed in a directory (key BINDIR) which lies in your PATH variable for execution. The modules (key MODULDIR) may or may not reside in this directory, too. If you have a preprocessor, the path to it is in the key M4BINDIR. Include files for M4 lie within a directory specified by the key INCDIR.

The key STYLEDIR codes a directory which is searched by \LaTeX . To help DVIPS find a prologue file, the key DVIPSDIR must point to a directory where DVIPS looks for files.

1.4 Post-installation steps

By internally calling \LaTeX , the compiler determines the sizes of \TeX texts which belong to formulae (e. g. element symbols). Depending on \LaTeX 's call syntax on your computer, you may have to change the value of the parameter `LaTeXCMD` in the configuration file `ochem.cfg`. This file is found wherever your BINDIR key points to. The default value is equivalent to the following call to \LaTeX :

```
latex <file>
```

The default command string `latex` in the configuration has to be replaced with `tex386` for example, if you start \LaTeX as follows:

```
LaTeXCMD=tex386
```

If a preprocessor for macro expansion should be used, the appropriate package from a third-party source must be acquired. The GNU M4 preprocessor is supported by the following macro packages for simple typesetting of natural compounds and commonly useful abbreviations:

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

If you want to use another preprocessor, you have to change some entries in the configuration file `ochem.cfg`. For a normal call of your preprocessor in the way:

```
YourPP.exe infile.c_raw outfile.c
```

use the following value in the configuration file:

```
PPCMD=YourPP.exe %INFILE% %TMPFILE%
```

1.5 Usage in \LaTeX

The descriptions of formulae and reaction schemes are given in a special language in ASCII files. These files may be independently and manually written from a running \LaTeX system, or created online from a \LaTeX document with embedded chemistry code. The latter method is described here due to a bigger convenience (text and formulae are not splitted across several files).

You have to load the package `ochem.sty` to get the new environment `chemistry` which encloses the formula description:

```
% file example.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}
```

A complete cycle of such a chemistry document requires two \LaTeX runs. In the first one, initiated with

```
latex example
```

the contents of all `chemistry` environments are written into a file `\jobname.chm`. Within this file, each environment is resolved into a `schema` command, which derives its name from the optional parameter of the corresponding environment. If the parameter has been omitted, subsequent numbers are used. After compilation, this name is kept as the name of the fragment file with an extension `.ctx` and therefore makes debugging easier.

The result from the first L^AT_EX run, `\jobname.chm`, is compiled by the chemistry compiler

```
chemie.pl example.chm
```

producing for each `chemistry` environment a file containing L^AT_EX code which build the graphical appearance of the formula. In the example above, you compile the file `example.chm` and get `pheno1.ctx`. (The extension `.ctx` can be interpreted as “chemistry T_EX”; they are not considered to be as important as other “normal” T_EX files because these fragmentary files can easily be recreated. The different extension in regard to the usual T_EX files allows for a more convenient archiving and handling.)

The compiler writes the name for each `chemistry` environment (`pheno1` in the example), enclosed in brackets, to the terminal:

```
This is OCHEM chemistry compiler version 1.0c 2001-05-25
requires OCHEM.STY 3.0e
[pheno1.ctx] [1.ctx]
```

This is especially helpful if an error interrupts the compiler run. The error must be found in the environment corresponding to the last name printed.

The `.ctx` files resulting from compilation are automatically loaded during the second run of L^AT_EX:

```
latex example
```

They form the complete and final graphical representation of the formulae. A warning is written to the log file, if these load attempts fail. This is always the case during the very first L^AT_EX run or later whenever new formulae are inserted into the document.

L^AT_EX environment for direct commands

Sometimes material must be added to the intermediate `.chm` file using chemistry commands described in section 3.2, e. g. to load libraries or preprocessor macros. This is done in the `chemspecial` environment which writes its content literally into the `.chm` file:

```
\begin{chemspecial}
  <direct commands>
\end{chemspecial}
```

The compiler man page

To achieve L^AT_EX fragments with the final formulae appearances, you have to compile the source file, created in the first run of L^AT_EX. The following command compiles `example.chm`:

```
chemie.pl example.chm
```

You may find it useful to call the perl script as follows, if you have not installed perl in a proper way:

```
perl chemie.pl example.chm
```

Currently, there are the following compiler switches:

- type *s* : Specifies the output format, *s* may be one of the values PS, PSLATEX oder LATEX. Default value is PSLATEX.
- trace lev : Prints a list of all internal calls up to the level depth specified with *lev*. You can use this switch to find nauseating programming bugs. The default trace depth is 0.
- pp : Tells the compiler to start a preprocessor before compiling the source file for resolving macros. You are expected to have successfully configured `chemie.pl` for your favorite preprocessor. Default: pre-process the input file.

Output files in pure \LaTeX can be achieved by typing

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

Note that this output type is not yet really supported due to graphical reasons. This possibility is expected to be better realized in future versions of OCHEM.

Configuration

Some compiler and appearance settings are kept in a configuration file in INI file format. This file is called `ochem.cfg` and can be used to change permanently the values of important parameters. The compiler settings are usually changed only once (at least if you finally got a running system ;-), and the appearance settings like stroke width or bond lengths remain unchanged, too, and can also be set in the \LaTeX document with the `set` command. But the configuration file is a good place for changes without touching the compiler code to support a site-specific appearance.

Coding conventions

The following items have to be considered while coding your favorite compounds:

- texts like names of formulae or variable identifiers are to be enclosed in double quotes:

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

If you want to embed double quotes as part of the string itself, you have to duplicate the double quotes (may causing double trouble):

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

Double double quotes are – as it is in FORTRAN – hints for the parser to read literal double quotes as part of the string, not as delimiters.

The strange construction "" at the beginning of a string is probably interpreted by an unmindful author as a representation of a string, starting with a double quote. In fact, this triple irritates the parser and you have to write it as follows:

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

The empty braces {} are later ignored by L^AT_EX, but effectively separate the starting delimiter and the double double quotes.

- Units, like length, are either specified in a symbolic manner (e. g. N for a normal bond) or as number without the unit symbol. For the unit of length, the point (pt) is assumed.

Comments starts with a percent sign, their scope ends at the end of this line:

```
% whole line is a comment

formula(L,R) % formula of atropinee
{ ... }
```

Percent signs within double quotes are parts of the string and will not be interpreted as the start of a comment. This allows you to publish your impressively high yields of the tenth step in a big synthesis.

From the M4 preprocessor's point of view, the percent signs do *not* specify the begin of a comment, instead the hash sign # is used. In this case, you can safely use the sequence %#:

```
formula(L,R) % normal comment
{ %# TERPEN(0) M4 macro commented out
}
```

Since the M4 preprocessor uses the hash sign # as comment marker, there arise two problems:

- percent signs are not recognized by M4 as the begin of a comment which leads to unacceptable replacements in the comments.
- If the hash sign # appears in a formula, as for example in `bond(#1)`, the rest of the line following the hash sign would disappear during the preprocessor run which would then lead to error messages of the chemistry compiler.

At the beginning of each include file, with the line

```
changeocom('//')
```

the sequence // is set as mark of comment for M4. To comment out lines, you note

```

formula(L,R) % normal comment
{ // TERPEN(0) M4 macro commented out
  ...
  Me_ bond(#1) // comment
}

```

The chemistry font

The command `\chemfont` specifies L^AT_EX font switches for choosing the font used for typesetting text symbols within formulae. The default is `\sffamily`; this means, all characters are as big as the document's text, but without serifs. You can set another font in the preamble of the document, e.g. the normal serif font in smaller size:

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

Or the document's normal font:

```
\chemfont{}
```

Ring structure modules

As section 2.10 of the tutorial explains in more detail, you can build library files with individual ring structures and use such which already exist by adding them to the document. Each library has to be loaded explicitly with a command in a `chemspecial` environment before it can be used:

```

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

```

The library `bicyclib.pm` provides bicyclic natural compounds and is shipped out with the package. In order to use this library (scheme 1–1), your document has to look similar to the following:

```

\documentclass[...]{...}
\usepackage{chem}

\begin{document}

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

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

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

```

```

formula(L,R,"methyacrylate")
{ 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}

```

Scheme 1–1 Usage of ring structure, here the bicyclohexane, from a library.

New floating environment

This package offers you a new environment `schema` to typeset floating figures which offer its own consecutive numbering for reaction schemes, just as the `figure` environment does in a similar way for illustrations. A simple example is the following:

```

\begin{schema}
  \begin{chemistry}[phenolsyn]
    <sophisticated synthesis>
  \end{chemistry}
  \caption{A sophisticated synthesis.\label{phen}}
\end{schema}

```

In two-column mode, you can feel free to use the star form of the environment (`schema*`) to span a reaction scheme over both columns.

formulae are L^AT_EX boxes!

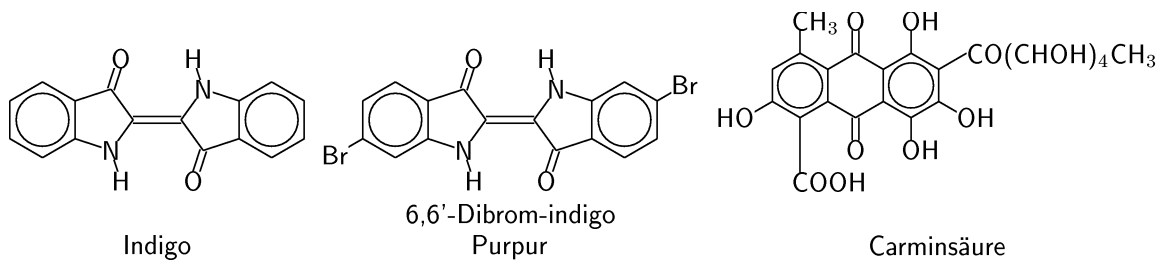
The `chemistry` environments represent normal L^AT_EX boxes, more exactly, `picture` environments or boxes. Therefore, you are able to exploit all positioning hacks used for L^AT_EX illustrations, e. g. the placement of multiple vertically oriented reaction schemes side by side or the embedding of formulae in the middle of the text, in figures or separate lines.

formulae as EPS, PNG ... graphics

The method depicted so far of including a formula by help of the `chemistry` environment into a L^AT_EX document prerequisites that the small L^AT_EX files are always produced beforehand by the formula compiler. Thus, the use of these formulae becomes possible only in L^AT_EX documents. By applying the package option `separate`

you can produce a document that consists solely of chemistry environments. They are automatically set by the option on separate pages, that is, each environment on its own page. With the page style `empty` which is automatically set, the page numbering is left out leaving only the formula visible on the page.

The example `formeln.tex` contains the following three formulae:



```

\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\\purple}",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}[carminsaeure]
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}

```

(Please pay attention in this example to the fact that each `chemistry` environment gets its own name `indigo`, `purpur` and `carminsaeure`!)

This document which solely consists of formulae provides the basis to generate single EPS files out of the single formulae. The EPS files may be turned into any other graphics format like JPG or PNG in order to present the formulae on web pages.

To illustrate this, the formula of the carminic acid (which is the third formula or formula on the third page of the document) is converted into the PNG format with the help of the shown command sequence:

```

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

```

Every computer provides the option `-E` of `dvips` to create EPS files. The program “convert” is part of the open-source graphic tool `image magick`, which is mainly installed under Linux or can be found for various systems in the internet.

The included perl script `makePic.pl` supports the conversion of more than one formula at a time:

```

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

```

This call of the script generates the files `farben1.eps` and `farben1.png` to `farben3.eps` and `farben3.png`. The conversion gets yet easier by the fact that the name of each `chemistry` environment is preserved in a special file `formeln.names`. These names can be used by the call

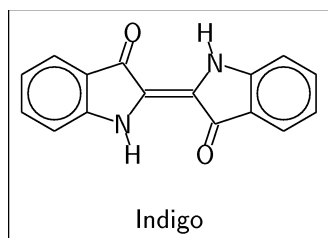
```

makePic.pl -namefile -format=png formeln

```

as names for the resulting graphics files. For the given example we get the files `indigo.eps` and `indigo.png`, `purpur.eps` and `purpur.png` as well as `carminsaeure.eps` and `carminsaeure.png`.

The PNG graphic may be put to use in internet publications, while the EPS graphic may be included in L^AT_EX documents:



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

If you are only interested in the EPS graphics, no conversion should take place. In this case, you call the script as follows:

```
makePic.pl -namefile -noconvert formeln
```

Please note that the package image magick must be installed on your computer, if you use the conversion feature!

1.6 Output formats

It is easily possible to generate output in other formats than PostScript. All output is composed of graphical primitives, using generic commands. These commands are interpreted in the module `be.pm` according to the specified output format. The primitives are lines, circles, text, bonds and arrows. If you find an implementation for all of these basic elements in your favourite output format, you can further compile all your chemical documents into this format.

In the current version, the following output formats, identified by an integer number, are available:

- `$be::BE_PSLATEX` : texts are typeset using \LaTeX in a `picture` environment, the graphical output is written in PostScript. The required definitions of bond and arrow types are contained in a header file in `ochem.sty`, implemented using `\special`. The dimensions of text blocks are determined by writing all occurring text fragments into a temporary file, applying \LaTeX to it and reading back the text box sizes (width, height and depth). This output format currently achieves the best output quality.
- `$be::BE_PS` : This output format produces pure PostScript in the EPS format. Caution: the sizes of text fragments are not determined and always set to zero, achieving poor results when the formulae contain texts, not only graphical structures. This format is mainly used for testing purposes during development without a need for a running \LaTeX system. The graphical presentation of the formulae themselves is always correct. If you are not using text in the formulae, you can use this mode without restrictions for EPS-generation.
- `$be::BE_LATEX` : This is a pure \LaTeX output format. The primitives like lines and circles will use the `epic` package in future, so that some radii and bond shapes are not supported. Note that this output type is not well supported at the moment.

In fact, sophisticated primitives as cubic splines used by `emove` are in the current version only available in PostScript. The pure \LaTeX format is not yet completely devel-

oped and therefore misses possible features. The results may be looking ugly sometimes. This is definitely a TO DO.

Depending on the output formats, the determination of text sizes has to be done in different ways. Text size calculation is done immediately after reading the chemical source code, so that the code will be found in module `streambuf.pm` instead of `be.pm`. In this module, the output format code decides how to calculate text sizes. Currently, the only way is to call `\TeX` for that (there is no way for pure PostScript output with text).

1.7 Macro definitions

The switch `-pp` calls a preprocessor during compilation of the `.chm` file generated by `\TeX` to resolve macro definitions and to handle conditioned branches in the chemistry source. You have to configure the call of your favorite preprocessor in `chemie.pl`, when the variable `$preproc` is evaluated within back ticks. The preprocessor has to process the content of the input file, its name held in `$infile`, and to transfer the processed file into the intermediate file `$tmpfile`. If your top-one preprocessor is GNU's M4, you do not have to configure anything.

If your are not planning to use any preprocessor, you do not have to configure the modules. In this case, you are expected not to use the `-pp` switch.

The M4 preprocessor

This is not M4's man page, but a short introduction how to define and use macros within a `\TeX` document. Each definition, located in a `chemspecial` environment, has to appear before its first use. The macro name and the replacement text are to be enclosed with a backtick-tick sequence. The actual arguments are available via the variables `$1`, `$2` and so on. The sequence

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

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

produces the structure of methoxymethyl-methylamine.

It is possible to write macros handling conditions. As an example, a macro for presentation of five-member heterocycles (thiophene, furan, pyrane) is to be written. The ring systems with chalcogens should not carry hydrogen at the hetero atom:

```
\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,"Pyrrole",HA,24){ CP(N) }
formula(L,R,"Thiophene",HA,24){ CP(S) }
\end{chemistry}

```

The macro compares its argument with the symbol for nitrogen and generates a further bond to a hydrogen atom, if the comparison succeeds.

If you have written some macros for recurrent use, you can collect them in an include file and load this file as needed – some kind of library at the preprocessor stage. Detailed examples of macros can be found in sections 3.6 and 3.7. You have to load the desired include file into your document as follows:

```

\begin{chemspecial}
include('natur.inc')
\end{chemspecial}

```

`include` is another preprocessor command of M4, loading the given file and executing the commands in it.

To use macros in the most powerful way, you should design them flexible. The idea behind the structure TERPEN was that most natural compounds of this type are substituted at the carbon atoms C^1 , C^2 , C^3 and C^8 . The rare substitution at C^2 is given as parameter, eventually being empty. The other important positions are saved and can be used as starting points of substituents with `restoreXY`. The macros' STEROID flexibility bases on the use of some `ifelse` commands, modifying the same basic structure by adding some side chains as needed. The basic structure is described once for all derivatives.

Try to find out which atoms are at key positions in your basic structure, and will often be substituted. Small modifications can be given as parameters. If complex formulae can be expected, the macro call can easily become unclear. In these cases, the storage of the key atom's position with `saveXY` may be better. A common structure with modifications can be built up with common code and some `ifelse` commands.

1.8 Examples, the formula catalogue

In this manual, many examples of coding numerous types of structures are given. Furthermore, in the directory `catalog`, you find a PostScript document `catalog.ps` and an input file with the \LaTeX coding, which contains a collection of many predefined formulae, all listed in an index under several names. The catalog is divided into two parts:

- The first part contains simple to complex basic structures which can serve as starting

point for your changes. If you need, for example, a phenanthrene derivate, you can quickly copy the phenanthrene formula and change it as you need it. In the case of complex formulae, this may help you a lot.

- The second part contains numerous final formulae for basic and complex natural compounds. The found structure can directly be copied as EPS file from the directory `catalog/eps`, if you find the formula suitable for your needs. Again, you might want to change the OCHEM code a bit to fit your needs.

I would be glad if users would contribute formulae to this small collection to create a wide fund of structures of every type. For this, the structures are given not in a \LaTeX document, but in the file `catalog.raw`. This file contains the OCHEM codings as well as some meta-information about the formula. The `.raw` file is processed by simply running `createCatalog.pl`. This run creates two \LaTeX documents, `catalogeps.tex` and `catalog.tex`. The first contains only the formulae for EPS/PNG creation, the latter the catalog which includes the generated EPS for smarter browsing the collection. Summarized, the catalog is created as follows:

```
cd <OCHEMROOT>/catalog

createCatalog.pl

latex catalogeps
chemie.pl catalogeps.chm
latex catalogeps
cp *.eps eps % DOS: copy *.eps eps
makePics.pl -noconvert -namefile catalogeps

latex catalog
latex catalog
makeindex catalog
latex catalog
dvips catalog
rm *.eps % DOS: del *.eps
```

1.9 Further perspective

In the current version of this package I have worked on some ideas about typesetting chemical formulae. It is considered as a living project, meaning that it should be improved, if it finds attention (critical as well as acclaiming, naturally, whereby the latter would be better:-)) from users, and support (better code or parts of it or include files or ...) from programmers. I trust on you to report errors and to contribute fine extensions. Some interesting TO DOs currently open are (the following list is not complete):

- Improvement of the output with means of \LaTeX (mode `$be: :BE_LATEX`). This includes better appearance of bond shapes as well as sophisticated features such as cubic splines.
- Calculation and typesetting of text in pure PostScript output (mode `$be: :BE_PS`).

- Improvement of the input stream parser and the general design.

Changed code should – analogous to the procedure T_EX/L_AT_EX uses – be renamed and therefore marked as modified, yielding only one version OCHEM Version 4/0d. This offers a clear and well-known catalogue of features (and anti-features as well). In order to successfully distribute the improved versions, it is suggested to you to send comments, criticism and improved code by email to me. I will then do my best to integrate the enhancements into the current code. Working code is always preferred :-). The email addresses are

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

The newest version of the package is always available at my OCHEM homepage at

```
http://www.2k-software.de/ingo/ochem.html
```

If you encounter problems or have suggestions, please have a look there to see if the topic in question was already addressed.

2. Tutorial

This chapter is intended to give you enough information to successfully start with your first steps in typesetting chemical equations with \LaTeX . It contains a tutorial, but also some useful hints, special aspects of some of the commands and possibilities of usage. A more detailed description of the commands is given in the reference section, chapter 3.

2.1 Your first steps into a chemist's world

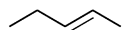
2.1.1 Simple formulae

The description of formulae follows a fairly geometric approach. But be calm, unlikely your scholarly math lessons, which may have left horrible traces in your memory, you are not expected to deal with trigonometric equations, attractors and singularities, with virtuosity. To describe a maximum number of different structures, there exists only a minimum number of basic elements as bonds in any directions and different shapes, atomic symbols and text blocks, branches and simple (monocyclic) ring structures. You combine these constructional parts, e.g. by attaching a bond to a specified atom of a ring structure. The bond gets a specified direction and style and can carry another bond, resulting in a carbon chain, or ends to a ring atom of another ring, yielding connected rings. There are no basic units for structures like indane. In these cases, you start from a six-membered ring and combine a certain vertex with the corresponding vertex of a five-membered ring.

The advantages over a collection of powerful high-level macros for each structure type can clearly be seen: much more structures can be constructed out of a very limited set of elements (and so, lucky for me, only a few elements have to be implemented in the compiler). A few simple commands can also be mastered more rapidly than a large number.

There are also less obvious facts (called disadvantages by some people): complicated structures require more simple elements. Due to the fact that bond directions have to be given (there is really no molecular modelling calculation in this little perl script) by angles, you have to think rather more like mathematicians do than chemists. But most of your structures use nearly always the same set of angles. Some symbolic angles help you to abstract from concrete angles in degrees.

Each formula is set within a `formula` command. The parameters of this command are described later in conjunction with the building of reaction chains. The body consists of the geometric description of the structure. Some simple examples will demonstrate this. The structure is an aliphatic chain:



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

For the double bond the notation of an optional parameter is required. Simple bonds correspond to the default value and need not be especially noted. Attention must be paid to the recurring angles 30° , 150° , -30° and -150° for slanted bonds. These angles harmonize with the edges of six-membered rings standing upright on a vertex. If rings are preferred which rest on their edges, then the angles 60° , 120° , -60° and -120° are used. For other bonds only the main angles 0° , 90° , 180° and -90° come into play (with minor exceptions).

Since chains of single bonds quite often appear, a short notation is possible in which several descriptions of bonds, separated by semicolon, are summarized. The above formula may also be noted as

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

The main disadvantage of the presented description is that the actual angles do not allow to show an identical carbon chain in a different, for example turned, orientation without recalculating all angles. A higher abstraction will be achieved through symbolic angles which are especially important in conjunction with ring systems. They will be introduced further down. Thus, the example aliphate can be noted as follows:

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

r is a special angle which normally corresponds to a bond branching away in radial direction of a ring system. If the description of the formula contains no ring system, a default value of 30° for r , 90° for t and -90° for b is set.

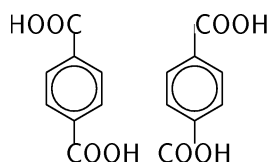
The (compressed) description can therefore be read as follows: create a radial bond; create a branching bond; create a bond in the original (radial) direction as double bond; create again a branching bond. Such a description is independent of actual angles and may be placed therefore at different positions within the formula without recalculating all angles. The command `ring` sets itself appropriate values for the symbolic angles, in case of pure alkyl chains the manual setting of angles with the command `set` may be seldomly required (examples in the reference section \rightarrow `bond`). With actual angles it is always possible to enforce the desired direction, even if symbolic angles are intensely used.

The second and all further formulae show the way in which atomic symbols are incorporated into the set of bonds. Note the positioning of the atom groups: single elements at terminal positions are mostly centered to the end of the bond with the default `C,C` of the command `atom`, while longer texts fit closely with an edge (such a fitting edge would be advisable on the grounds of unity for such letters like "O", but causes naturally a good deal of effort).



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

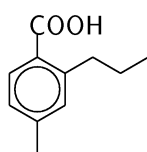
A special problem arises if bonds of 90° or -90° end in long texts, for example in the carboxyl or hydroxyl group. To position these long groups not aside but (better) centered toward the "C", the text is split in a centered (C) and a sideways positioned part (OOH) (formula left). The formula on the right shows the appearance of the carboxyl group, if it would have been positioned in one piece with the option L (bond above). Centering of the text would have resulted in a catastrophe (bond below).



```
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); }
}
```

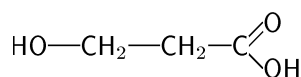
The problem will not occur if slanted bonds meet with a long text and are positioned at the side of the text, as it was demonstrated with the ether chain and the two carboxyl groups.

Ring structures can be drawn with the command `ring`. In the simplest case, you get a benzene ring. Various parameters are used to typeset n-membered rings and other basic structures (camphore and others) and also to modify the bond types (unsaturated, for example). With `rings`, we have for the first time elements from which several substituents can branch. These substituents are described like normal formulae, they must be terminated by `;`. The ring atom from which the substituent branches is identified by an integer number, starting from zero and shown in tables in the reference section for the command `ring`. It is possible (and recommended) to use symbolic angles, e. g. `r` for radial bonds instead of actual angles (symbolic angles are discussed in detail in the next section):



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

Chains can provide another variant of branching. `branch` stores the current position and allows to describe side chains, each one starting at the stored position and terminated with `;`:



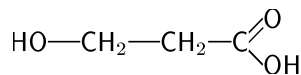
```
formula(L,R){
  atom("HO",L,R) bond(0)
  atom("CH$_2$",L,R) bond(0)
  atom("CH$_2$",L,R) bond(0)
  branch { bond(-45) atom("OH", L);
           bond(45,=) atom("O",L);
           atom("C");
         }
}
```

Since this position remains unchanged after the command's body is executed, the main line of the chain can be continued, and only the side chains must be described within the command `branch`:



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

The commands `→saveXY` and `→restoreXY` allow you to store the current position, identified by an integer number, and later to restore this position. The recursive nesting of structure descriptions using the command `branch` can be transformed into a linear description. This may be useful especially on deep nesting levels.



```
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 Alignment of formulae

When you have described a formula, you can use it within the command `formula` as a prebuilt unit and let it automatically be aligned in respect to other reaction chain elements (other formulae and arrows). The parameters of the command specify how the formulae should be aligned relative to a starting point (centered, left-aligned, with edge or corner) and where the connection should be found. The connection point is the starting point for the next element in the chain (formula or arrow). If you are only interested in inserting a formula into a text, both parameters can be chosen arbitrarily in this case and their default values will work fine. The parameters become meaningful only in the interaction of several formulae.

Scheme 2–1 gives an example for the above said. In the left part, the formulae appear side by side, vertically aligned to a common center line. The position parameters are `L,R`, meaning that the *left* formula lies on the starting point (the formula extends to the right) and the base point for the second formula is on the *right* side of the first one. In the right part, the same formulae are positioned with `T,B`, resulting in a top-to-bottom chain, again aligned to a common, vertical center line:

```
\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()
}
```

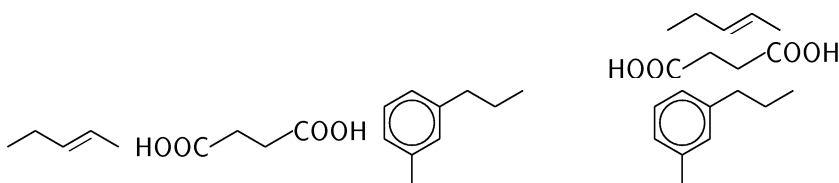
```

    { 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}

```



Scheme 2–1 The positioning parameter describes the relations between the formulae. Left: formulae aligned left to right, right: formulae aligned top to bottom.

Both variants to join formulae are probably the most typical ones, because they build linear reaction chains. Later, it is shown how to build up branches in the chains. However, two important special cases have to be mentioned here:

- A regular table-like position cannot be achieved by the means shown so far, because the position of each following formula depends on the size of all previous ones. The command `→gotoXY` sets the starting point of a formula independent of the context to a given position and therefore forms the basis for a regular placement. Examples can be found in the reference of `gotoXY` on page 100.
- A free placement of formulae may be appropriate when two or more educts lead to a product, but should not appear in single row, one after the other joined through plus signs, but instead they should be more freely distributed “cloud-like”. This may be realized with `→shiftXY` which shifts the current point by a given amount of space. Examples are given in the tutorial in section 2.6.

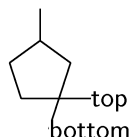
2.2 More complex formulae

This section introduces complex formulae and the means required for their description. Some of the used hacks are mentioned in the alphabetically sorted command reference, others aren't, so it is recommended that you read the reference section,

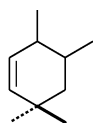
too. If you think you cannot describe a given structure, look over the formulae shown here, maybe you get an idea how to solve your problem. Sometimes and as a last means, you can try it with “wild” lines :-)

2.2.1 Symbolic angles

The command `ring` sets three important angle values for the symbolic angles `r`, `t` and `b`, used for radial bonds and bonds tangential to the ring vertices. These values are processed by the command `bond`, if you use one of the mentioned symbolic angles. In this way you can easily describe formulae, especially with rings, without caring about trigonometric calculations. The next two compounds show the usage of all three symbols:



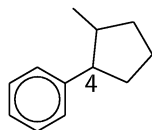
```
formula(L,R){
  ring(,,H,,5,90){
    0: bond(r);
    2: bond(t) atom("top",L);
    2: bond(b) atom("bottom",L); }
}
```



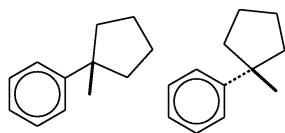
```
formula(L,R){
  ring(,,H1=){
    0: bond(t,t); 0: bond(b,o);
    3: bond(r);
    4: bond(r); }
}
```

In structures with stereoscopic intention (ring types `chair`, `bc222o` and so on), the meaning of the angles `t` and `b` as “top” and “bottom” (or above and below) becomes clear, examples can be found in the introductory section for the library modules and the `→ring` command.

When a bond, given as symbolic angle, terminates at a second ring, the angle is unknown by which the second ring has to be turned around in order to achieve one of the relations `r`, `t` or `b` for the bond and the second ring. This knowledge is unnecessary, because the turning angle for the second ring can be replaced by a symbolic angle, too. Only the number of the ring atom being the bond’s target must be specified:



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

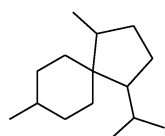


```

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);
    };
  }
}

```

As it can be deduced from acorane, spiro compounds can also be drawn connecting the two rings directly, not by a bond. The symbolic angle r is a default for turning angles:



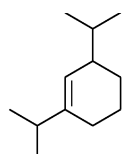
Acoran

```

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-); };
    };
  }
}

```

A following + or - increases or decreases the angle by 60° , respectively. From the view point of the bond to be drawn, this correlates to a turn left (right) about 60° . You can therefore easily describe branched alkyl compounds, especially isopropyl substituents:



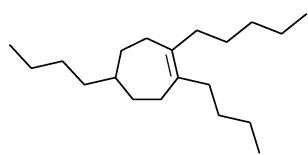
```

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

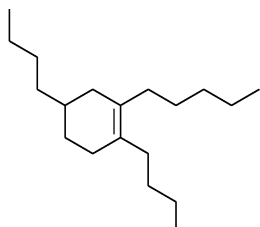
```

If one or both of the branches consists of more than one methylene unit, it is better to describe the branch with rt or r/t , see below.

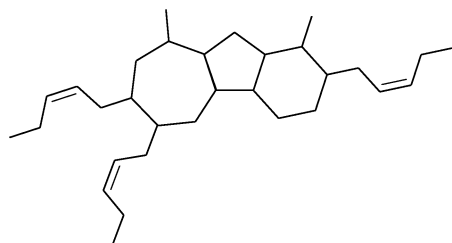
With the modifiers + and -, longer carbon chains can be typeset without knowledge of the exact angles. Since the turn is relative to the current direction, all identical carbon chains can be portrayed with the same description, independently of the ring position they start from. But note that all chains are rotated clones of one and the same chain and are therefore oriented differently, depending on their starting position in the ring:



```
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-) ; }
}
```

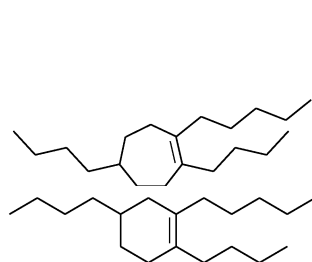


```
formula(T,B){
  ring(,,H1=,,6,90){
    1: bond(r; r-; r; r-; r);
    2: bond(r; r-; r; r-) ;
    5: bond(r; r-; r; r-) ; }
}
```

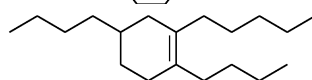


```
formula(L,R){
  ring(,,H,,5,90){
    vertex(,1,4,H,6){
      0: bond(r);
      1: bond(r; r-; r,=U; r+; r);
    };
    vertex(,3,1,H,7){
      0: bond(r);
      4: bond(r; r-; r,=U; r+; r);
      5: bond(r; r-; r,=U; r+; r);
    };
  }
}
```

Some of these chains are usually depicted with a horizontal orientation. If such a construction is desired, you can apply the symbolic angle $r/$ instead of the fixed turn $r-$. This declaration does not enforce a turn by a certain amount, but takes the current direction of the bond into consideration: if the bond points to the top-right (e.g. 30°), then the next bond will be drawn to the bottom-right (-30°). Bonds at the bottom-left are followed by bonds at the top-left and so forth. This construction is most agreeable for six-membered rings:

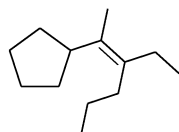


```
formula(T,B){
  ring(,,H1=,,7,90){
    1: bond(r; r/; r; r/; r) ;
    2: bond(r; r/; r; r/) ;
    5: bond(r; r/; r; r/) ; }
}
```



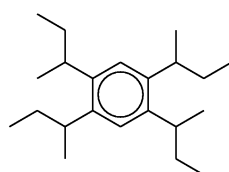
```
formula(T,B){
  ring(,,H1=,,6,90){
    1: bond(r; r/; r; r/; r) ;
    2: bond(r; r/; r; r/) ;
    5: bond(r; r/; r; r/) ; }
}
```

Repeated usage of the + und - symbols increases or decreases the current direction by 60° , so that even nested carbon chains can be organized without information about the exact angles of the current ring-position:



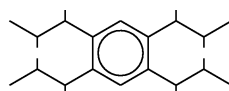
```
formula(L,R){
  ring(,,H,,5,90){
    1: bond(r)
      branch { bond(r+); }
      bond(r-,=)
      branch { bond(r--; r---; r--); }
      bond(r) bond(r-); }
}
```

The following example discusses a possibility of chain-branching in more detail. As a result of the fixation of a left-turn for the methyl branch (r+), we get on the “left side” of the ring different chains from those on the “right side” in respect to the orientation of the methyl groups. The chains arise through strict rotation:



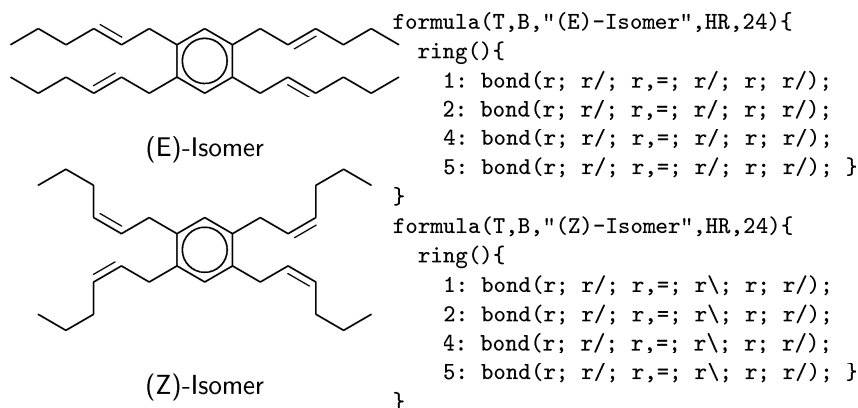
```
formula(L,R){
  ring(){
    1: bond(r) branch { bond(r+); } bond(r-;r);
    2: bond(r) branch { bond(r+); } bond(r-;r);
    4: bond(r) branch { bond(r+); } bond(r-;r);
    5: bond(r) branch { bond(r+); } bond(r-;r); }
}
```

The symbolic angles rt and r/t try to achieve a more appropriate, that means, more symmetrical and at the coordinate axes oriented placement of such attachments. Left- and right-turns are allowed and are automatically resolved (in the example, the bonds to some of the inner methyl groups must be shortened to prevent them from overlapping):



```
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); }
}
```

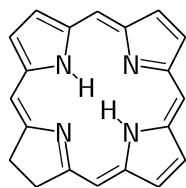
For the delineation of stereo chemistry at double bonds, the symbolic angle $r\backslash$ must be included. With this angle, a bond against the angle $r/$ will be drawn. The effects and the correct notation of E/Z isomerie independent of actual angles is demonstrated in the following formula:



2.2.2 Linear systems

Many formula drawings can be expressed as linear sequences of simple bonds or identical resp. similar construction units.

The relatively complex ring of chlorin may be viewed in spite of its ring construction as a linear system of four pyrrole methylene units (another method of description of this system is mentioned on page 93). Each of these methylene bridges ends at the pyrrole ring of the next unit. The ring may be therefore described by the nested and almost identical depiction of the basic unit. The formation of the ring results out of the correct choice of the angles on symmetrical grounds. (Since the continuation points of a structural unit are always dependent of its predecessor, the usage of linearization with `saveXY` and `restoreXY` brings no advantage. A benefit could only be achieved if the starting points of the structural units would be available at the very beginning as in the construction shown on page 93.)



Chlorin

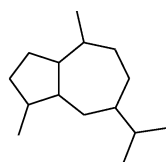
```

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);
        };
      };
    };
  };
}

```

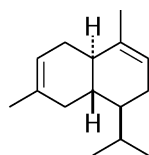
2.2.3 Linear annellated rings

In the simplest case of annellated ring systems, all rings share a common atom. It proves useful that the command `ring` does not shift the actual starting point and thus, if the actual point corresponds to the shared atom, all rings can be drawn by subsequent `ring` commands. This approach, however, brings up some problems, e.g. with bond length of rings with different numbers of atoms. Although it is possible to set all rings that exist in the same nesting level with subsequent `ring` commands, varying the starting points, it is better to use the `vertex` command for simple ring systems, too. Note that a certain edge length is enforced due to the use of a length specification of `#N`:



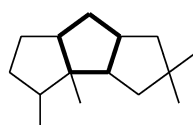
Guajan

```
formula(L,R,"guajane",HR,24){
  ring(, ,H,#N,7,0){
    1: bond(r) branch { bond(r+); bond(r-); };
    5: bond(r);
    vertex(,3,2,H,5){ 4: bond(r); };
  }
}
```

 β -Cadinen

```
formula(L,R,"$\beta$-Cadinen",HR,24){
  ring(, ,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);
    vertex(,1,4,H1=){ 1: bond(r); };
  }
}
```

If not all rings possess a shared atom, a basic ring must be chosen which carries the other rings as annellated substituents. For simplicity's sake, it may be advisable to choose the most complex substituted ring as basis. In this case, simpler substitution at the annellating rings is gained. In the case of hirsutan, the central five-membered ring is chosen as basis, because in this way two simple rings appear as substituents. If one of the terminating rings would be chosen, the central five-membered ring would become its substituent which in turn would carry another ring as substituent. And this would make the formula really complicated! In this and the following formulae the basic ring is emphasized by a larger stroke width:

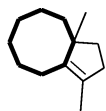


Hirsutan

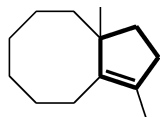
```
formula(L,R,"Hirsutan",HR,24){
  ring(, ,H, ,5,90){
    vertex(,1,2,H,5){ 0: bond(t); 0: bond(b); };
    vertex(,3,3,H,5){ 0: bond(r); };
    3: bond(t);
  }
}
```

The `vertex` syntax is responsible for the otherwise complicated calculations of the correct starting points and turning angles of all annellated rings. A further example

demonstrates how easy assembled ring systems may be described. Both variants illustrate that the choice of the basic ring influences the overall turning of the formula: in the first case, the unturned cyclooctane forms the basis. The five-membered ring is attached to the slanted [a] edge and leads to the impression of the molecule to be turned slightly. In the second case, with the five-membered ring as basis, the probably more desired impression of a horizontally oriented formula arises:

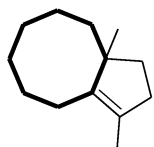


```
formula(L,R){
  ring(,,H,,8,0){
    vertex(,0,1,H0=,5){ 0: bond(r); };
    0: bond(t);
  }
}
```

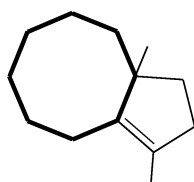


```
formula(L,R){
  ring(,,H1=,,5,0){
    vertex(,2,0,H,8){ 0: bond(t); };
    1: bond(r);
  }
}
```

In the above examples, the variable length of edges of the rings should be noted. This is due to the fact that the regular ring systems possess a common diameter so that the bond lengths of higher ring systems decrease. The choice of the basic ring system influences the lengths of all annellated rings. With a size specification like L, other sizes can be chosen. To enforce the bonds to have a certain length, you can use the # size syntax, e. g. #L to have all ring vertices have the length of a L bond:

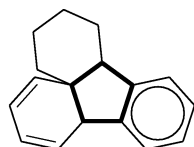


```
formula(L,R){
  ring(,,H,L,8,0){
    vertex(,0,1,H0=,5){ 0: bond(r); };
    0: bond(t);
  }
}
```

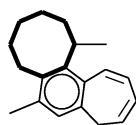


```
formula(L,R){
  ring(,,H,#L,8,0){
    vertex(,0,1,H0=,5){ 0: bond(r); };
    0: bond(t);
  }
}
```

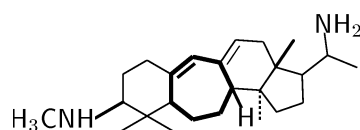
Several substituents lead to complicated systems in which the annellated rings themselves carry more rings so that systems come into existence in which not all rings are adjoined to an edge of the basic ring. This will be illustrated by the following examples:



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



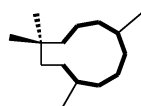
```
formula(L,R){
  ring(,,H,,8,0){
    vertex(,1,1,,6){
      0: bond(r);
      vertex(,3,0,H2=4=,7){};
    };
    0: bond(r);
  }
}
```



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-);
      };
    };
  }
}
```

The common bond needs not always to be visible. The humulene consists out of a nine-membered ring condensed with a four-membered ring. The impression of the square-shaped hole is simply caused by an invisible bond in each of the rings:



Humulene

```
formula(L,R,"Humulene",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 Manual construction of annellated systems

You have seen in the last section how easily polynuclear systems can be build with `vertex`. Unfortunately, this syntax is only implemented for ring systems of the type `ring` (polygons with n vertices). Using this syntax with rings of e. g. type `cpentane` is impossible. In these cases, you have to calculate all data yourself, but here is a brief description of this procedure. It bases on polygons, too (due to their simplicity), but can easily be used for all types of ring systems if you know their typical angles. The polygons serving as base systems are not rotated in the examples, their parameter `<p2>` is zero.

Table 2.1 Correlation of vertex angles to individual vertices for different polygons (cyclopentane, $n = 5$ to cyclooctane, $n = 8$). The vertex number is identical to the number of a substituent.

Edge	edge number	$n = 5$	$n = 6$	$n = 7$	$n = 8$
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

To each vertex, an angle θ_i corresponds to it as shown in table 2.1. These values have to be inserted into the general formula. In the formula, the symbols have the following meaning: ϕ_{soll} is the desired rotating angle of the base system, ϕ the correct rotating angle of the substituting ring, θ_1 and θ_2 the vertex angles of the colliding vertices of base and substituting ring:

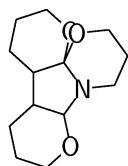
$$\phi = \phi_{soll} + (\theta_1 - \theta_2) + 180$$

The following compound bases on the five-membered pyrrolidin and is shown rotated by 0° , 45° and 90° . The origins of the furan rings are indicated by the oxygen, the origin of pyrrolidin by nitrogen.

At the beginning, we build the structure without turning the pyrrolidin nucleus around. Due to the following correspondences between the vertices we get the angles:

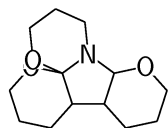
pyrrolidine		furan		angle
vertex	substituent i	vertex	starting atom j	
b	1	e	$4 + 1 = 5$	$0 + -18 - -180 + 180 = -18$
d	3	b	$1 + 1 = 2$	$0 + -162 - 0 + 180 = 18$
e	4	e	$4 + 1 = 5$	$0 + -234 - -180 + 180 = -234$

The choice of the vertices which shall correspond to each other determines the position of the origin of the rotated ring. The origin is given in the table. The vertex label is the position number i of a substituent of the base ring (table 2.1). The starting atom j that has to be inserted in the description list of the substituting ring is the vertex label incremented by one.



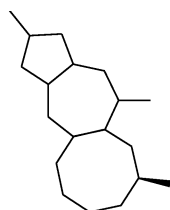
```
formula(L,R){
  ring(, ,H,#N,5,0){
    1: ring(,5,H,#N,6,-18){ 0: atom("O");};
    3: ring(,2,H,#N,6,18){ 0: atom("O");};
    4: ring(,5,H,#N,6,-234){ 0: atom("O");};
    0: atom("N"); }
}
```

In all rotated systems, the rotating angle ϕ_{soll} has to be added to all angles.



```
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"); }
}
```

At last a compound, consisting of a five-, seven- and eight-membered ring:

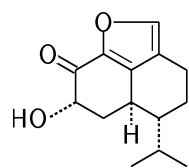


```
formula(L,R){
  ring(, ,H,#N,7,0){
    4: ring(,3,H,#N,5,129){ 0: bond(r);}; % 0 + -141 - -90 + 180
    1: ring(,6,H,#N,8,350){ 0: bond(r,<<);}; % 0 + 13 - -157 + 180
    0: bond(r); }
}
```

The typical angles of a ring type like `cpentane` are those of a regular six-membered ring, where the missing vertex has to be considered in the subsequent labeling.

2.2.5 Angular annellated ring systems

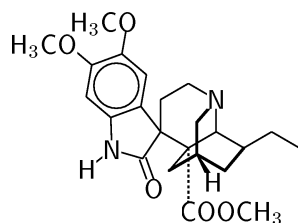
It may be difficult to describe polynuclear systems with bonds shared by several rings, if the number of ring members differs. The angles of a six-membered ring are different (smaller) in comparison to that of a five-membered system. There is no problem with condensed six-membered rings, in the common case of six- and five-membered rings, you can choose the ring type `cpentane` for the latter, whose angles fit to the cyclohexane's angle.



(+)-Hibiscin A

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

The crassanine shows the combination of bicyclo[2.2.2]octane with a six-membered ring. The bond lengths in the bicyclus fit to all others. The length of the five-membered ring's bonds have to be reduced a bit with the specification `#N`, prohibiting a too large benzene ring in comparison to the cyclohexane.



Crassanin

```

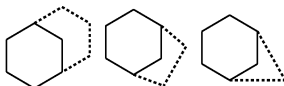
formula(L,R,"crassanin", HR,24){
  ring("bc222o",,,N){
    2: bond(30) bond(-30);
    0: ring(,,H,N){
      1: ring(,4,H,#N,5,-70){
        0: bond(-70,=C) atom("O");
        1: atom("N") bond(r) atom("H");
        vertex(,2,1){
          4: bond(r) atom("O",C,L) atom("H$_3C",R);
          5: bond(r) atom("O",C,L) atom("H$_3C",R);
        };
      };
    };
    0: bond(-90,o,NN) atom("C",C,R) atom("OOCCH$_3",L);
    4: bond(-60,<<) atom("H");
    7: atom("N"); }
}

```

If other ring sizes are added, it is the easiest way to build first a basic construction of five- and six-membered rings. Then, the larger or smaller irregular rings are added as carbon bridges, as it is shown in the next section.

2.2.6 (Carbon-)bridges

Bridges may occur in various forms. In the following cases, the dashed bonds are regarded as bridges.



(a) (b) (c)

Case (a) is characterized by the fact that the bridge is a copy of a part of the basic ring system which has been shifted. Problems of positioning the compound are avoided if the corresponding part of the compound is exactly copied (parallel shifts keep the angles).

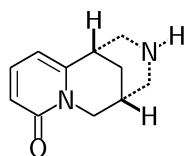
Case (b) is more problematic because the bridge possesses no counterpart within the basic compound. If the bridge starts from one bridge head, it is difficult to calculate the angle and the length of the bond which closes the ring (the last bond of the bridge) in order to hit the other bridge head. Thus, I recommend to build the bridge simultaneously from both bridge heads and to store the last position of one of the bridge parts. This position will serve as aim of a "wild line" which will be drawn from the other bridge part with `bond(#n)`. For the wild line, the bond of the bridge will be best chosen of which you know least how angle and length are to be set.

Case (c) is even more problematic, because the bridge consists of only one single atom, so you have to calculate the position of this atom instead of using wild lines. It may be helpful to jump by means of an invisible bond to a point where the bridge is symmetrically parted. This position must be stored. Then you can draw with `bond(#n)` two wild lines from both bridge heads to this stored position. Which point you choose as a target for the bridge atom depends somewhat from your geomet-

rical imagination.

The construction of a symmetrically positioned target will be a lot easier, if the extended mathematical expressions `bond(=<expr>)` are applied, because the calculation of central positions in relation to known starting positions (for example ringatoms as bridge heads) will be performed by the computer. Some examples follow.

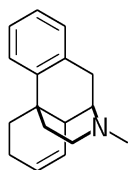
Case (a)



(-)-Cytisin

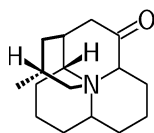
```
formula(L,R,"($-)-Cytisin",HR,24) {
  ring(,,H){
    vertex(,1,4,H1=3=){
      0: bond(r,=C) atom("O");
      5: atom("N");
    };
    3: bond(r,t) atom("H");
    5: bond(r,t) atom("H");
    3: bond(30,o; -30,o) atom("N")
      branch { bond(30,o) atom("H"); }
      bond(-90,o; -150,o);
    }
  }
}
```

The morphinee also possesses a bridge which looks like a sector of the six-membered ring and which hits exactly after the shift again on a ring atom of the basic compound. In contrast to cytisine, here short bonds with unusual angles for the shift and thicker bonds for the bridge in order to achieve a more three-dimensional impression are used. Since the furan ring is distorted because of the three annellated rings (case (c)), first a morphinee derivative will be shown which possesses only the discussed, shifted bridge (the complete construction of the morphinee is shown under case (c)):



```
formula(L,R) {
  ring(,,H){
    vertex(,2,5,H1=3=5=){};
    vertex(,0,3,H5=){};
    1: bond(-70,t,S; -30,p; 30,t) atom("N")
      branch { bond(-30); }
      bond(110,t,S);
    }
  }
}
```

The bridge in lycopodine is another example for shifted copies of the basic structural unit:

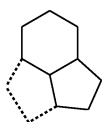


Lycopodin

```
formula(L,R,"Lycopodin",HR,24) {
  ring(,0,H) {
    vertex(,0,3,H){
      4: atom("N");
    };
    vertex(,5,2,H){};
    1: bond(30,<<) atom("H");
    4: bond(r,=C) atom("O");
    0: bond(-170,t,S; 150,p)
      branch { bond(-150,<.); bond(150,<<) atom("H"); }
      bond(90,t; 10,t,S);
    }
  }
}
```

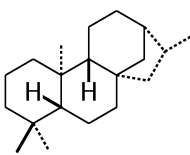
Case (b)

The following two examples show a bridge which is constructed by the approximation from both bridge heads, while the closing of the ring is achieved by a wild line (bridge again drawn as a dashed line):



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

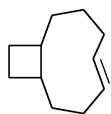
In the case of the kaurane, a symmetrical construction of the bridge is implemented through the possibility to add or subtract constant angles (here 20°) to bond angles:



(-)-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){
      3: bond(-120,t); 3: bond(-60,o);
    };
    vertex(,3,0,H){
      4: bond(-20+v(3),o) saveXY(#1) bond(30,o);
      0: bond(20-v(0),o) bond(#1,o);
    };
  }
}
```

The depiction of a humulene-related structure results in difficulties, if the E/Z isomers caused by the double bond in the nine-membered ring shall be correctly drawn. With the help of wild lines, the following construction can be utilized: the nine-membered ring can be understood as a bridge with an unknown part in it which can be represented by a wild line. In the example, we get an asymmetry, because this wild line does not end in the vertical center of the four-membered ring. But you will achieve a construction for a precise symmetrical formula under case (c).



Humulenderivat wild

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

Case (c)

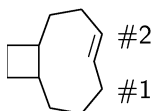
The solution for this problem was already given; in the first example it is easy to determine the target because it lies in a radial prolongation of a ring atom, which is between the two bridge heads (here position 5).



(c)

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

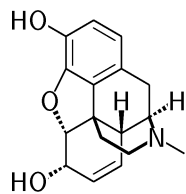
With the help of the extended formula of the `bond` command a precise solution for the already mentioned problems under case (b) can be attained. The exact construction of the humulene derivative builds two symmetrical bridges, starting from the four-membered ring. They extend to the last, unknown part and their destination points are stored as positions 1 and 2. The last atom is now found exactly half between the distance from position 2 to position 1 ($0.5 * (\#2, \#1, 0.5)$), shifted by a half bond length to the left ($-\{n, 0\}$):



Humulenderivat exakt

```
formula(L,R,"Humulenderivat exakt",HR,24) {
  ring(, ,H,#N,4,-45) {
    0: bond(-70; -10; 50) saveXY(#1);
    3: bond(70; 10; -50) saveXY(#2)
    bond(=0.5*(#1-#2)-{n,0},=; #1);
  }
}
```

In a similar way the morphinee can be constructed. The two important positions, namely point 1 and 2, between which the ether oxygen lies, are each located in one of the annellated rings:



Morphin

```

formula(L,R,"morphine",HR,24) {
  ring(.,H){
    vertex(.,2,5,H1=3=5){
      1: saveXY(#1);
      2: bond(r) atom("O",C,L) atom("H",R);
    };
    vertex(.,0,3,H5=){
      1: bond(r,<.) atom("O",C,L) atom("H",R);
      2: bond(=0.5*(#1-#cur)-{N,0},<.) atom("O") bond(#1);
    };
    0: bond(90,<<) atom("H");
    5: bond(t,<.) atom("H");
    1: bond(-70,t,S; -30,p; 30,t) atom("N")
      branch { bond(-30); }
      bond(110,t,S);
    }
  }
}

```

Wild lines

The above shown examples hopefully fulfilled the purpose to demonstrate how very helpful wild lines may be, they are most important for tweaking the structures into the desired form ;-). If the bridges are very complicated, long and often confusing command procedures have to be written. It may be helpful to store the position of one or both bridge heads with the command `saveXY` and to come back to them later, outside of the nested description of the basic structural unit, with the command `restoreXY`.

2.3 Reaction chains

The concept of the automatical arrangement of formulae can be extended with another basic element, the reaction arrow. Thereby, formulae can not only be arranged next to each other, but with interposed arrows be combined to complete reaction schemes or chains. Beside a simple linear sequence, branches and their opposites, joins and multi-line schemes are possible.

2.3.1 Horizontally linear sequences

Instead of the two already described positioning parameters of a formula, the arrow is applied with a direction and length in order to determine the starting point for the next element in the chain. The arrow always starts at the current point. In the simplest case of a chain directed horizontally from left to right, a sequence of formulae with the positionings (L,R) and arrows with an angle of 0° (default) is noted, as shown in scheme 2-2:

```

\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); }
}

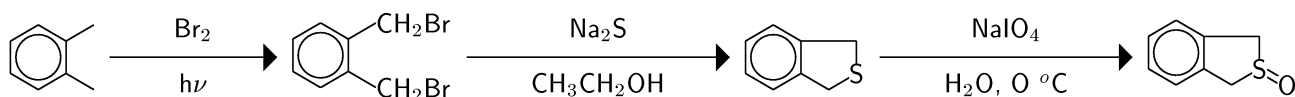
arrow()
{ text(T,C){ formula(C,C){ atom("Na$_2$S") }}
  text(B,C){ formula(C,C){ atom("CH$_3$CH$_2$OH") }}
}

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

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

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

```



Scheme 2–2 Horizontal reaction scheme: synthesis of a benzothiophene.

With this positioning, each formula is vertically aligned to a common center-line on which the arrows lie. It is also possible to build the chains right-to-left, whereby the positioning (R,L) is used and an angle of 180° for the arrows.

An extension of the formula syntax allows for the labeling of the compounds and is especially appropriate for the setting of formula in horizontal chains. For this, you must choose the type HA (horizontal, absolut distance) and specify the distance from the text's baseline to the center-line, as it is shown in scheme 2–3. The command `\shortstack` provides the possibility of setting line breaks within the text:

```

\begin{chemistry}[hor2]
formula(L,R,"o-xylene",HA,36)
{ ring(){ 4: bond(r); 5: bond(r); } }

arrow()

```

```

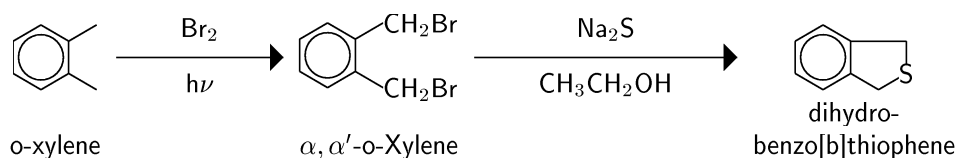
{ text(T,C){ formula(C,C){ atom("Br$_2$") }}
  text(B,C){ formula(C,C){ atom("h$\nu$") }}
}

formula(L,R,"$\alpha,\alpha'$-o-Xylene",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]thiophene}",HA,36)
{ ring(){
  vertex("cpentane",4,1,H){ 4: atom("S"); };
}
}
\end{chemistry}

```



Scheme 2–3 Horizontal reaction scheme with compound labels which are aligned toward a common line underneath the reaction arrow (absolute distance of the text to the center-line).

For this method, it is necessary to estimate the required distance of the text to the center line. It is quite difficult to set an equal distance for several independent schemes. This circumstance is taken into account with the command `multiline`, which is especially designed for the setting of multi-line horizontal reaction chains. It determines in every line the lowest formula edge and observes a constant distance (of size `rTextSep`) for calculating the height of the whole text line. All of the compound labels in this line are equally aligned and hold an identical minimum distance `rTextSep` from the largest formula. With $n = 1$, you can solve the discussed problem (scheme 2–4). It is to be noticed that formulae within `multiline` must not have positioning parameter for the label (HR and similar ones) because these values are automatically determined:

```

\begin{chemistry}[hor3]
multiline(1)
{ formula(L,R,"o-xylene")
  { 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-xylene")
  { ring(){ 4: bond(r) atom("CH$_2$Br",L);

```

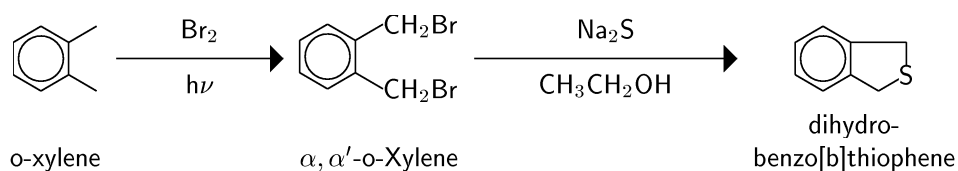
```

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]thiophene}")
{ ring(){
  vertex("cpentane",4,1,H){ 4: atom("S"); };
}
}
;
}
\end{chemistry}

```



Scheme 2-4 Horizontal reaction scheme with compound labels which are automatically aligned in a constant distance on a common line underneath the reaction arrow.

To determine the correct distance of the texts from the center line, only those formulae are used which are also labeled. Therefore, the case does not arise that a very large intermediate (for example a vertically depicted mixture of isomers without labeling) influences the height of the labels of the whole line, as you can see in scheme 2-8.

2.3.2 Multi-line schemes

So far, only single reaction lines have been discussed. It is however possible to set several such lines one above the other, whereby automatically a constant distance `rMultilineSep` between each line pair is set (the possibility of labels is taken into account), see scheme 2-5.

```

\begin{chemistry}[multli1]
set("rArrowExtend",12)
multiline(2,L)
{ % line 1
  formula(L,R,"naphthalene)
  { ring(, ,H0=2=4=){
    vertex(,4,1,H3=5=){
    };
  }
}
}

```

```

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

formula(L,R,"Succinyl-naphthalene")
{ ring(, ,H0=2=4=){
  vertex(,4,1,H3=5=){
    3: bond(r)
    branch { bond(150,=C) 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,"naphthyl-butanoic acid")
{ ring(, ,H0=2=4=){
  vertex(,4,1,H3=5=){
    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("cyclization") } } }

formula(L,R)
{ ring(, ,H0=2=4=){
  vertex(,4,1,H3=5=){
    vertex(,3,0,H){
      5: bond(r,=C) atom("O");
    };
  };
}
}

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

formula(L,R,"tetrahydrophenanthrene")
{ ring(, ,H0=2=4=){
  vertex(,4,1,H3=5=){
    vertex(,3,0,H){
    };
  };
}
}

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

formula(L,R,"phenanthrene")
{ ring(, ,H0=2=4=){
  vertex(,4,1,H3=5=){

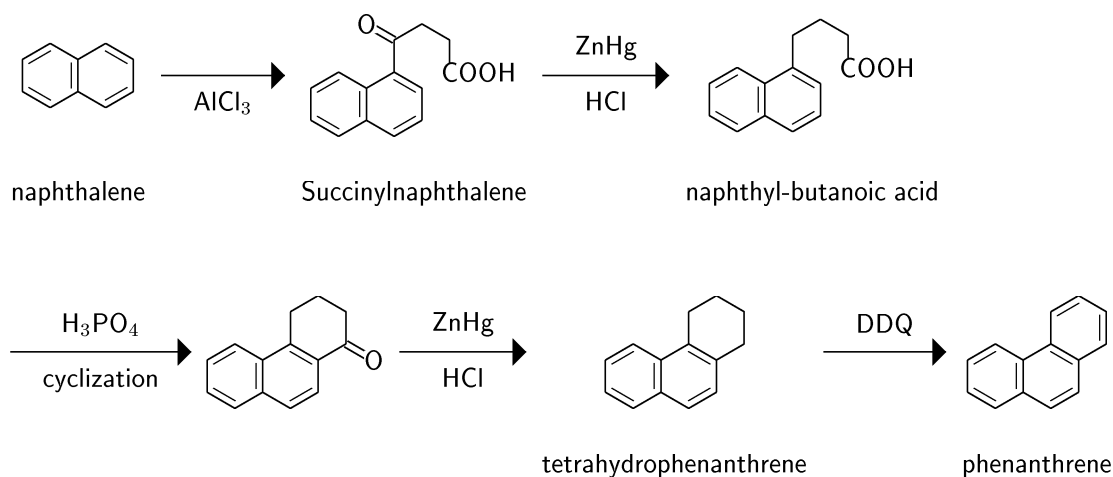
```



```

        vertex(,3,0,H2=4=){
      };
    };
  }
}
;
}
\end{chemistry}

```



Scheme 2-5 Synthesis of phenanthrene, set in multiple lines with `multiline`.

In this way, you can break up a long reaction chain into several lines or set a crowd of independent, short chains one above the other. Furthermore, the command is used to combine several single formula or bigger part of reactions. Examples are given in section 2.3.5.

2.3.3 Vertical alignment

Reactions running from top to bottom can be as easily set as chains from left to right. The main direction is set with the positioning (T,B), in rarer cases from bottom-to-top with (B,T), as is demonstrated in scheme 2-6 (left). The arrows now turn down with an angle of -90° :

```

\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$_3$") }}
  }

```

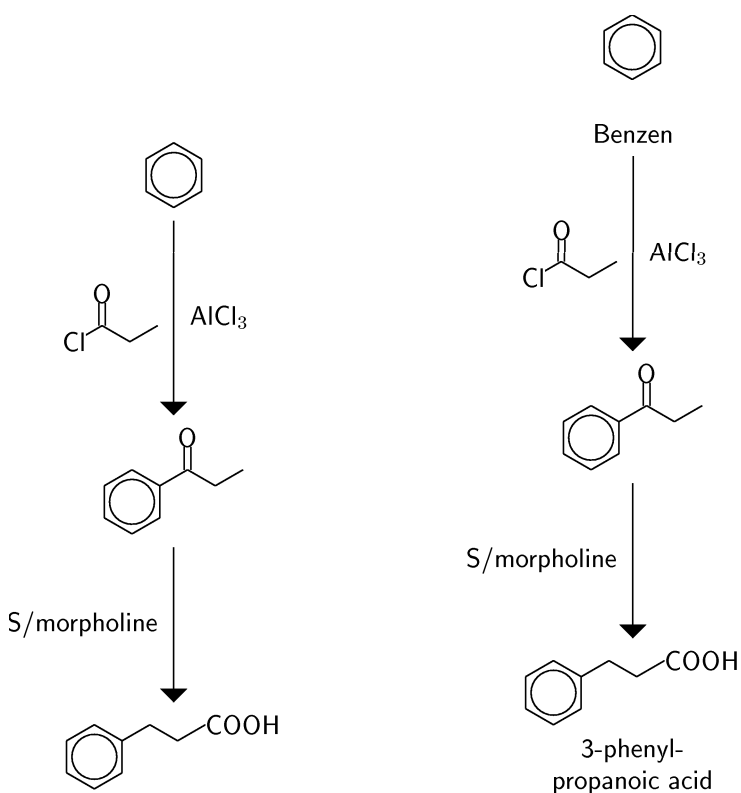
```

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/morpholine") }} }

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

```



Scheme 2-6 Vertical reaction scheme: synthesis of 7-Oxononanic acid. On the right the same scheme with labeling of the compounds. All texts possesses a fixed distance from the lower formula edges.

Again, the labeling of the formulae is possible, but the distance from the text's baseline to the lower formula edge must be noted (type V), because the horizontal centerline does not exist. In this case also line breaks can be set within the \LaTeX command `\shortstack` (scheme 2-6, right):

```

\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/morpholine") }} }

formula(T,B,"\shortstack{3-phenyl-\propanoic acid}",V,24)
{ ring()
  { 4: bond(r; r-; r) atom("COOH",L,R); }
}
\end{chemistry}

```

Up to this point, we assumed that the main direction of the chain is always the same (left-to-right or top-to-bottom). This does not always have to be the case. It depends on your preference to set the positioning parameters of a formula to (L,B), for example. This changes the sequence at this point from left-to-right to top-to-bottom.

2.3.4 Context and branching

It is now time to introduce the concept of *contexts*. Implicitly, we have already used it with the drawing of the very first scheme. A context is nothing more than the bounding box of a formula. It is applied to calculate the coordinates of the new and most important points of the formula (which are the four corners, the centers of the four edges and the center itself) with the help of the positioning parameters `<pos>` and `<Cpos>`. Each `formula` command sets the current context to the bounding box of the formula which was created by it.

The context can be made directly visible through the commands `fbox` or `bracket`: they use the actual context to frame the corresponding formula or to enclose it with brackets.

An important characteristic of reaction chains which bases on the context, is the *branching*. Starting from one and the same formula, we can create several chains. The context and thereby the geometrical dimension of the formula is saved with the command `savecontext` under a certain number. So this context can be repeatedly recalled with `setcontext`, again using its identification number. This is necessary at the beginning of each further reaction chain, because the context of the branching formula will be overwritten by each following formula.

A simple branching of the reaction chain is given in the reference to the command `→setcontext` in scheme 3–4. The reaction possibilities of a borane form an impressive illustration for multiple branchings. As was the case with simple branching, the context of the central formula which serves as branching point is stored only

one time, because it does not change. In contrast to the simple case, however, it is used several times in order to calculate (with the command `setcontext`) the starting point of each following chain:

```
\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,"Alkylborane",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") }

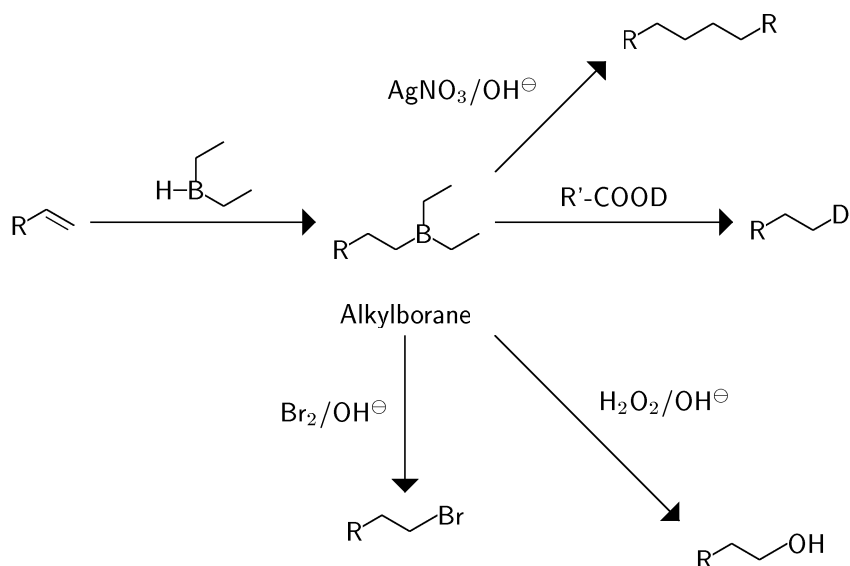
  setcontext(#1,BR)
  arrow(-45)
  { text(T,L){ formula(C,C){ atom("H$_2$O$_2$/OH$^\ominus$") }} }
  formula(TL,R)
  { atom("R") bond(30; -30; 30) atom("OH",L) }

  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}
```

Other possibilities of branching, where several formulae build up a context, working as a unit, will be discussed in the following section.

2.3.5 Parts of reactions as units

It may arise the wish to display an intermediate product with several mesomeric states vertically one above the other within one horizontally organized chain (see scheme 2–8). The solution is to apply a `multiline` command with a small vertically oriented scheme as single-line content. Thus, both border structures can be regarded as a unity, which is centered by help of the positioning parameter `L,R`, be-



Scheme 2-7 What happens to the borane? Multiple branchings, derived from one and the same formula, can show it. The context of the branching point, here the borane, must be stored.

ing a “reaction chain”. The inner alignment becomes senseless in the case of single-line content of the `multiline` command and can therefore freely be chosen. Additionally, another `multiline` command was used for the complete reaction chain to align the formula names:

```

\begin{chemistry}[multli2]
multiline(1)
{ formula(L,R,"benzene"){ 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(,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("$\oplus$"); }
};
}

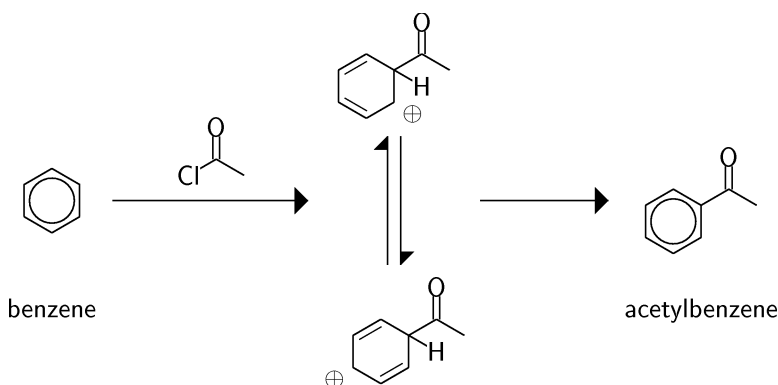
arrow(){}

```

```

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

```



Scheme 2-8 An independent vertical scheme is set as a single-line `multiline` object and can thus be inserted as a unity into the horizontal formula stream.

You can organize with this method horizontally aligned border structures as well as reaction parts within vertically ordered chains. `T,B` must be used for positioning. This has been applied for the mixture of different chloro-butenes in scheme 2-22.

2.3.6 Joining contexts

`multiline` may also be of help to you if you want to contract some formulae, i. e. to achieve a bracketing of all participating formulae, as it is shown in scheme 2-8. The command `bracket`, however, will only bracket the last drawn formula. A single-line `multiline` unity provides a *super-context* to the outside. This is a context which includes all nested individual contexts. It offers the possibility to use it as a reference to multiple formulae for branching or bracketing. Scheme 2-9 may serve as a good example for the bracketing of two formulae:

```

\begin{schema}
\begin{chemistry}[multli3]
multiline(1)
{ formula(L,R,"benzene"){ 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=)

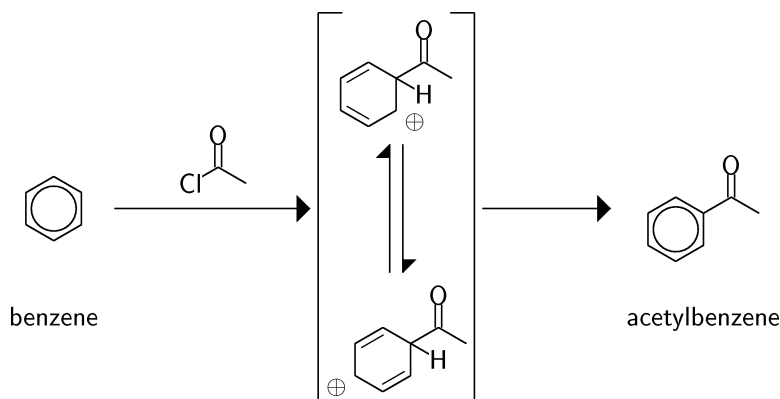
```

```

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

```



Scheme 2-9 A super-context, built up with `multiline`, permits to bracket all formulae, contained in the super-context.

Besides the `bracket` command you can also build up multiple branchings, using `savecontext` and `setcontext`. The super-context may serve as a base for these commands as well as a normal context. Regardless of the super-context, you can manipulate the single contexts, if these are stored *within* the `multiline` command, as demonstrated by scheme 2-10:

```

\begin{chemistry}[multi4]
formula(L,R)
{ bond(-30,=) bond(30)
  branch {bond(90,=C) atom("O");} bond(-30);
}

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

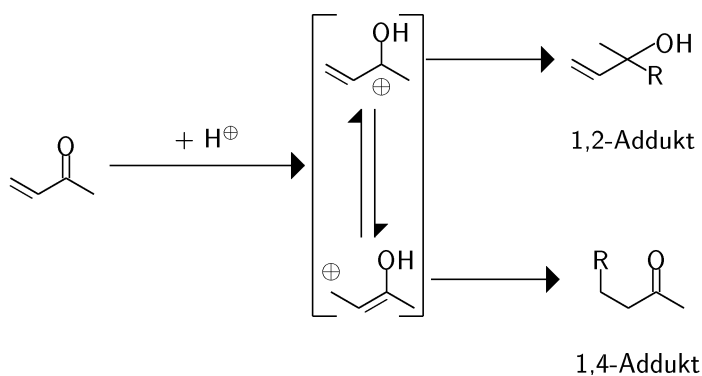
```

```

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}

```



Scheme 2–10 Branchings can be applied to each single formula independently from the super-context, if the contexts are stored separately.

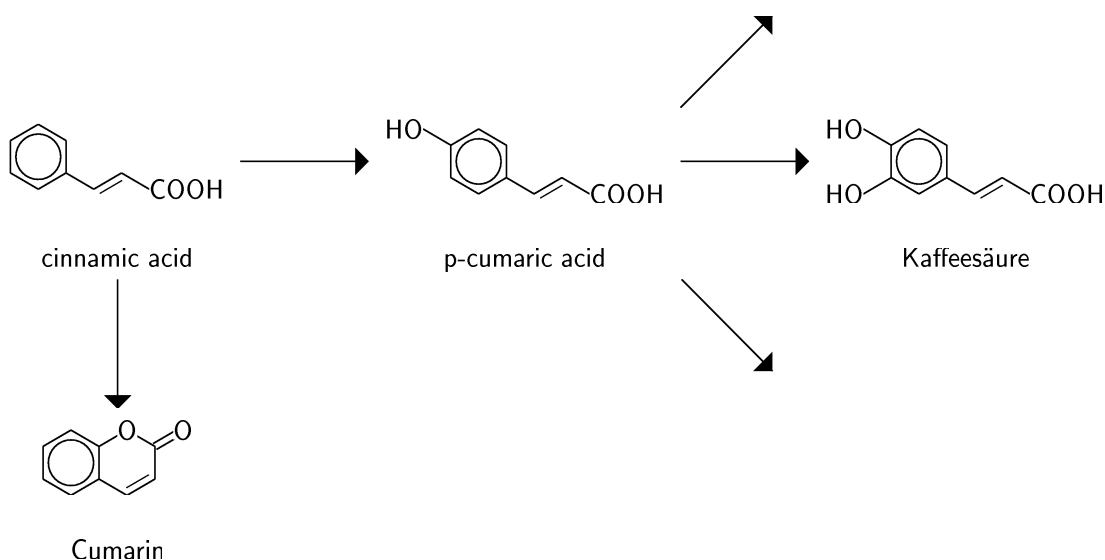
The storage of the contexts of single formulae within `multiline` is important, if `multiline` is used for the alignment of the formula titles and the formulae them-

selves are branching points, too. See the following example (scheme 2–11):

```
\begin{chemistry}[multli5]
multiline(1)
{ formula(L,R,"cinnamic acid")
  { ring(){5: bond(r;r/;=;r) atom("COOH",L); } }
  savecontext(#1)

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

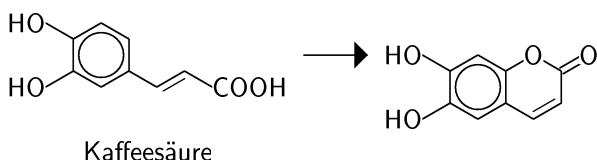
  arrow(){
  formula(L,R,"Kaffees\\""aure")
  { ring(){5: bond(r;r/;=;r) atom("COOH",L);
    2: bond(r) atom("HO",R);
    1: bond(r) atom("HO",R); }
  }
  }
; }
setcontext(#1,B) arrow(-90){}
formula(T,B,"Cumarin",V,24)
{ ring(,,H5=)
  { vertex(,1,1){};
    3: atom("O");
    4: bond(r,=C) atom("O"); }
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



Scheme 2–11 With `multiline` aligned formula titles and branching, attained with individual contexts.

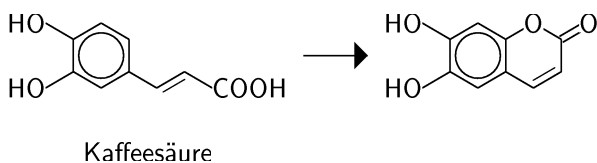
For each formula which acts as a branching point, an individual context is stored. Formula titles which were produced with `multiline` move into the context. In the case of arrows pointing down, you will be rewarded with the correct connection point.

Since the `multiline` block is regarded as a whole, positioning problems may arise if a formula without a title should follow the `multiline` block. This formula's centerline would be set centered in relation to the whole preceding block, including the titles:



```
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");
  }
}
```

The solution is realized by storing the context of the last formula and setting it after the closing of the `multiline` block:



```
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 Merging of several branches

Reaction chains may be split up into several parts, but also quite often the opposite is the case. The merging of several branches to a single one is often necessary, as in the course of a convergent synthesis. You can choose either the command `joinh` or `joinv`, depending on whether the branches should merge horizontally or vertically.

The synthesis of a crown-ether may serve as a demonstration for a horizontally aligned merging:

```

\begin{chemistry}[joinh1]
joinh(2,L)
{ % line1
  formula(L,R,"Bis-chloroethylether")
  { 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()

```

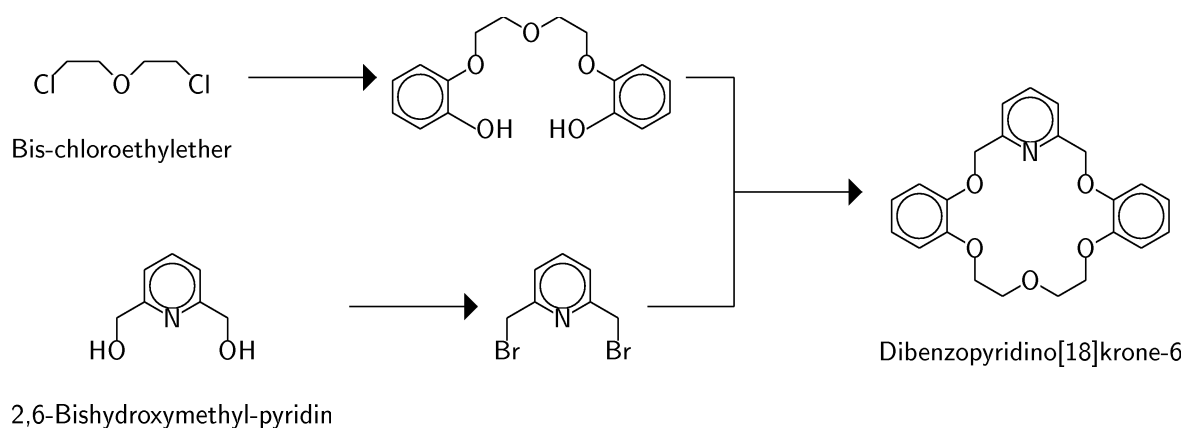
```

    { 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}

```



Scheme 2–12 The synthesis of a crown-ether as an example for the merging of two horizontal chains with `joinh`.

Multiple horizontal mergings are also possible, if you write additional (nested) `joinh` commands into one or more branches of an outer `joinh` command (scheme 2–13):

```

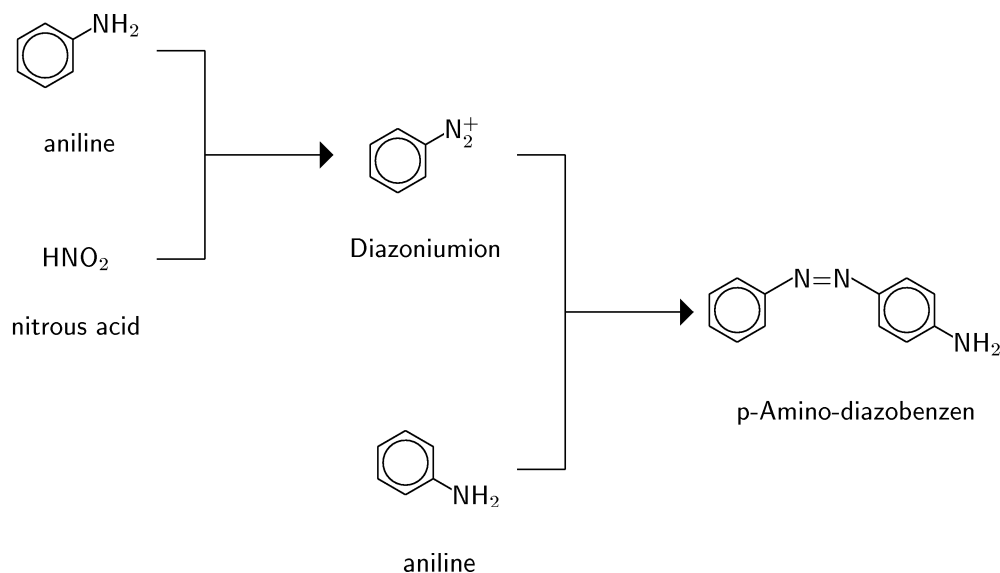
\begin{chemistry}[joinhh]
joinh(2,R)
{ joinh(2,R)
  { formula(L,R,"aniline")
    { ring(){ 4: bond(r) atom("N",C,R) atom("H$_2$",L); } }
    ;
    formula(L,R,"nitrous acid")
    { atom("HNO$_2$") }
    ;
  }
}
arrow(){

```

```

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

```



Scheme 2–13 Horizontal merging of two horizontally merged reaction sequences.

In the same way you can combine vertical schemes. Scheme 2–14 displays this with two nested `joinv` units:

```

\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)
}
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,"tosylchloride",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-methylbutanoic acid",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}-chrysanthemum acid",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}

```

If needed, you can also nest horizontally merging sequences into vertically merging chains and vice versa. An example is scheme 2–15, which shows a horizontal scheme, containing a vertical sequence:

```

joinh(2,R)
{ joinv(2,B)
  { formula(T,B,"aniline",V,24)
    { ring(){ 4: bond(r) atom("N",C,R) atom("H$_2$",L);} }
    ;
    formula(T,B,"nitrous acid",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);} }
  ;
}
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);} }
}
}

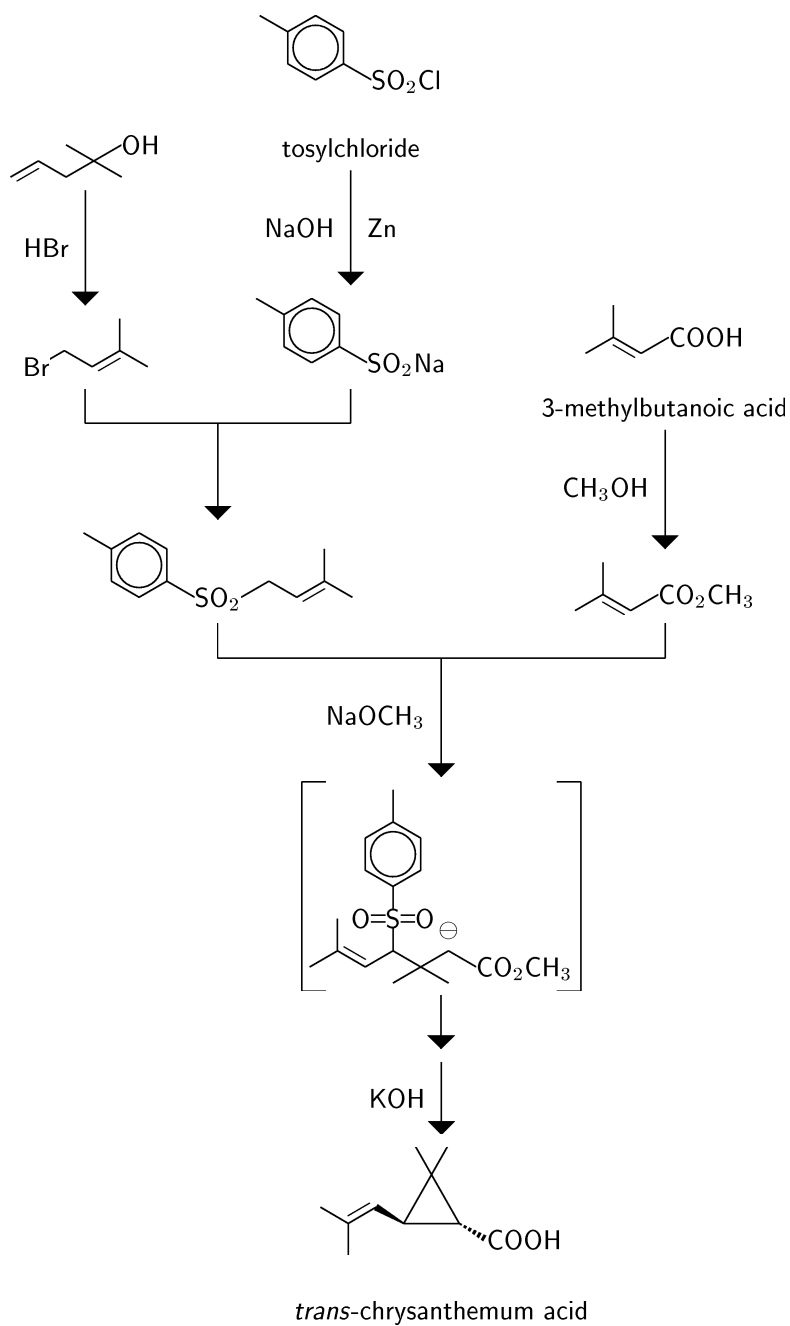
```

The counterpart with juxtaposed roles is a vertical scheme with a horizontal part, as scheme 2–16 shows:

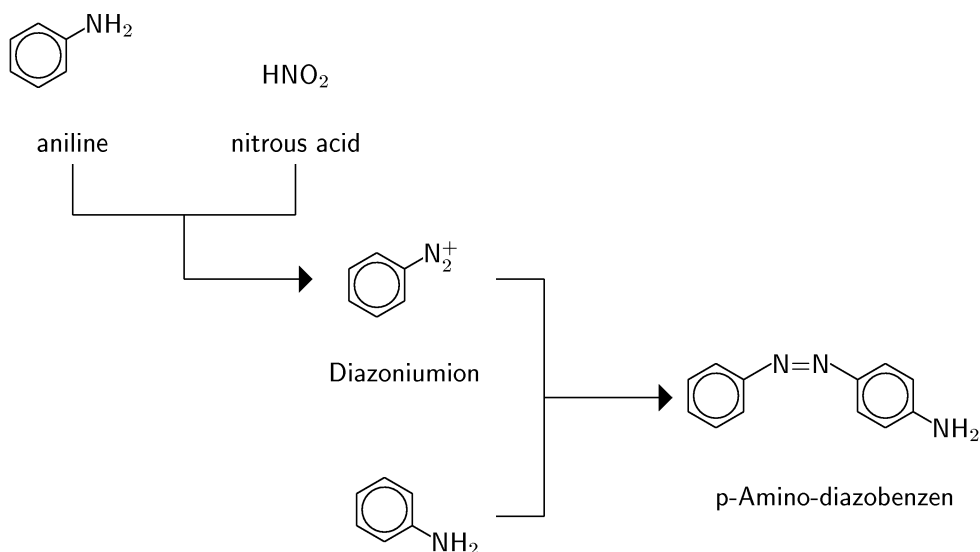
```

\begin{chemistry}[joinvh]
joinv(2,T)
{ joinh(2,L)
  { formula(L,R,"aniline")
    { ring(){ 4: bond(r) atom("N",C,R) atom("H$_2$",L);} }
    ;
    formula(L,R,"nitrous acid")
    { 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,"aniline",V,24)
  { ring(){ 5: bond(r) atom("N",C,R) atom("H$_2$",L);} }
  ;
}

```



Scheme 2-14 Synthesis of Chrysanthemums[®]aure, arranged with joinv.



Scheme 2–15 Synthesis of a diazo compound. Mixed horizontal and vertical merging.

```

}
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}

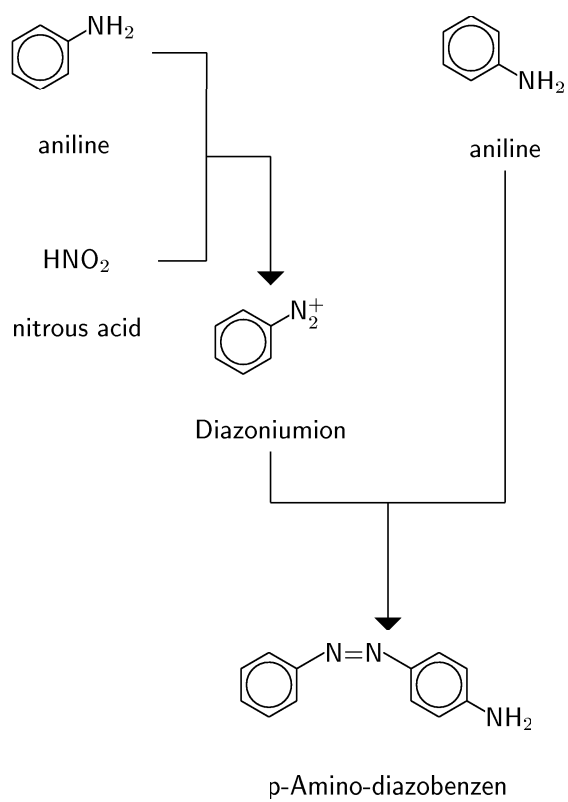
```

2.3.8 Mixing of basic elements

It is possible to mix the various methods for the setting of complex formula units (multi-line blocks and merging), which have been introduced so far. The homogeneous mixture of the merging has already been discussed. In the following lines, we depict the way in which a merging will be embedded at the beginning of a single-line multiline body (scheme 2–17):

```

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



Scheme 2-16 Mixed joinv/joinh commands.

```

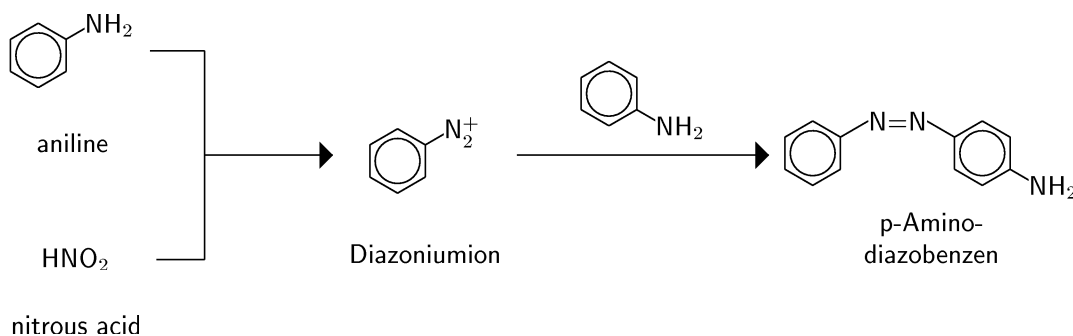
formula(L,R,"Diazoniumion")
{ ring(){ 4: bond(r) atom("N",C,R) atom("$_2^+$",L);} }

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}

```

More lines can be added to the `multiline` environment if necessary.



Scheme 2–17 Merging of horizontal reaction sequences with `multiline`.

2.4 Cyclohexane and carbohydrates

In literature, we find different representations of carbohydrates. Structural formulae following FISCHER's representation may be set with `atom`, the branchings to the left and right are constructed with the command `branch`:

```

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)
}

```

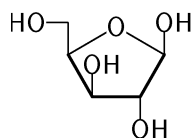
$\begin{array}{c} \text{CHO} \\ \text{HO}-\text{C}-\text{H} \\ \text{H}-\text{C}-\text{OH} \\ \text{H}-\text{C}-\text{OH} \\ \text{CH}_2\text{OH} \end{array}$

D-arabinose

In the case of frequent use of such formulae a macro may appear helpful, for example:

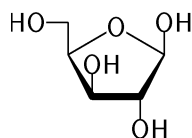
```
\begin{chemspecial}
define('TOP_', 'branch { atom("C",C,R) atom("$1",L); } bond(-90)')
define('MID_', 'branch { bond(180) atom("$1",R);
                      bond(0) atom("$2",L); }
                      atom("C",C,C) bond(-90) ')
define('BOTTOM_', 'atom("C",C,R) atom("$1",L); ')
\end{chemspecial}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

A schematic three-dimensional delineation according to HAYWORTH can be attained with the ring type `furanose`, maybe with the additional notification of a bonding list to reinforce the perspective bondings:



β -D-xylose

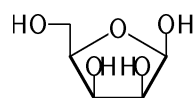
```
formula(C,C,"$\beta$-D-xylose",HR,24){
ring("furanose",0,,L){
0: atom("O");
1: bond(90) atom("O",C,R) atom("H",L);
2: bond(-90) atom("O",C,R) atom("H",L);
3: bond(90) atom("O",C,R) atom("H",L);
4: bond(90;180) atom("O",C,L) atom("H",R);
}
}
```



β -D-xylose

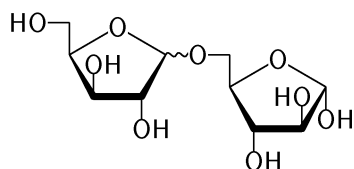
```
formula(C,C,"$\beta$-D-xylose",HR,24){
ring("furanose",0,1<<2b3>>,L){
0: atom("O");
1: bond(90) atom("O",C,R) atom("H",L);
2: bond(-90) atom("O",C,R) atom("H",L);
3: bond(90) atom("O",C,R) atom("H",L);
4: bond(90;180) atom("O",C,L) atom("H",R);
}
}
```

Within the ring, the representation of *vic*-hydroxyl groups in the *cis* position may become problematic. The hydrogen symbol can however be shifted nearer to the oxygen symbol with the help of T_EX commands. To avoid the covering of parts of the "O", the deleting of the background has to be put off until the "H" is set:

 β -D-lyxose

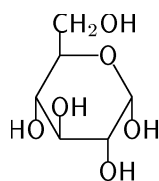
```
formula(C,C,"$\beta$-D-lyxose",HR,24){
  ring("furanose",0,,L){
    0: atom("O");
    1: bond(90) atom("O",C,R) atom("H",L);
    2: bond(90) atom("O",C,L) atom("H\kern-.2em",R,,0);
    3: bond(90) atom("O",C,R) atom("\kern-.2emH",L,,0);
    4: bond(90;180) atom("O",C,L) atom("H",R);
  }
}
```

Furanose rings can be combined in the same way as it is done with normal rings to build up oligosaccharides. In the following example, the bondig type \sim is used to denote an anomere:

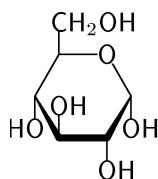
D-xylofuranosyl-(1→4)- α -D-arabinose

```
formula(C,C,
"D-xylofuranosyl-(1\to4)-$\alpha$-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);
}
}
```

Pyranoses also can be delineated following the HAYWORTH method. For this, there is no special ring type required. A normal big six-membered ring can be utilized, perhaps with a bonding list to strengthen the three-dimensional impression:

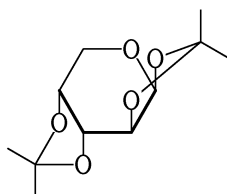
 α -D-glucopyranose

```
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

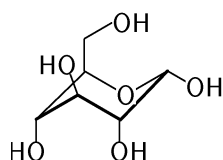
```
formula(L,R,"$\alpha$-D-glucopyranose",HR,24){
  ring(, ,H0<<1b2>>,L,,0){
    5: atom("O");
    0: bond(-90) atom("O",C,R) atom("H",L);
    1: bond(-90) atom("O",C,R) atom("H",L);
    2: bond(90) atom("O",C,R) atom("H",L);
    3: bond(-90) atom("O",C,L) atom("H",R);
    4: bond(90) atom("C",C,R) atom("H$_2$OH",L);
  }
}
```

Wild lines provide the possibility to describe derivations of carbohydrates, as is the case with 1,2-3,4-di-O-sec-butyliden- β -D-arabino-pentopyranose:

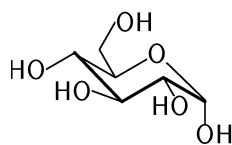


```
formula(L,R){
  ring(, ,H0t1t2t,L,,0){
    5: atom("O");
    1: bond(90) atom("O") saveXY(#1);
    0: bond(90) atom("O") bond(20, ,L)
      branch{bond(-30); bond(90);} bond(#1);
    2: bond(-90) atom("O") saveXY(#2);
    3: bond(-90) atom("O") bond(-120, ,L)
      branch{bond(150); bond(-90);} bond(#2);
  }
}
```

For a more detailed depiction of steric properties of the pyranoses, the ring type `chair` may be used. It shows a cyclohexane ring in stereo delineation (in chair conformation). The symbolic angles t and b for axial and equatorial substituents prove helpful. For spatial reasons, the hydroxyle group at position 5 must be turned 20° from the normal t position:



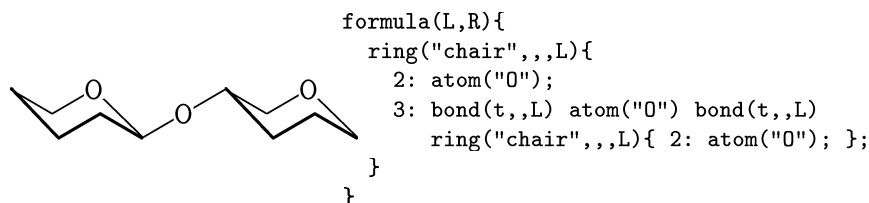
```
formula(L,R){
  ring("chair", , ,L,-1){
    0: bond(b) atom("O",C,L) atom("H",R);
    5: bond(20+t) atom("O",C,L) atom("H",R);
    4: bond(b) atom("O",C,R) atom("H",L);
    3: bond(b) atom("O",C,R) atom("H",L);
    2: atom("O");
    1: bond(t; t-) atom("OH",L);
  }
}
```



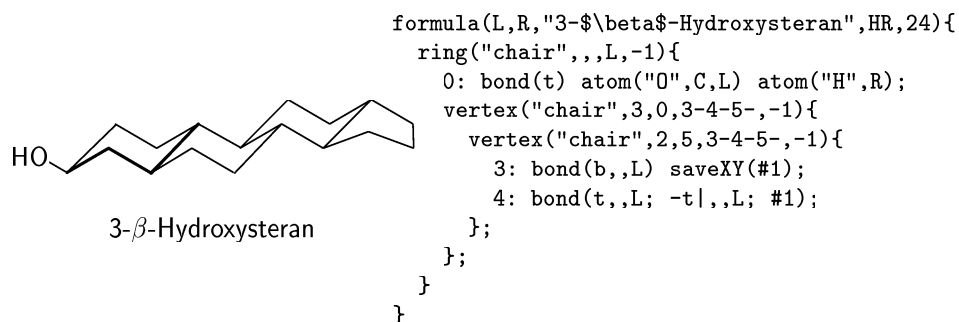
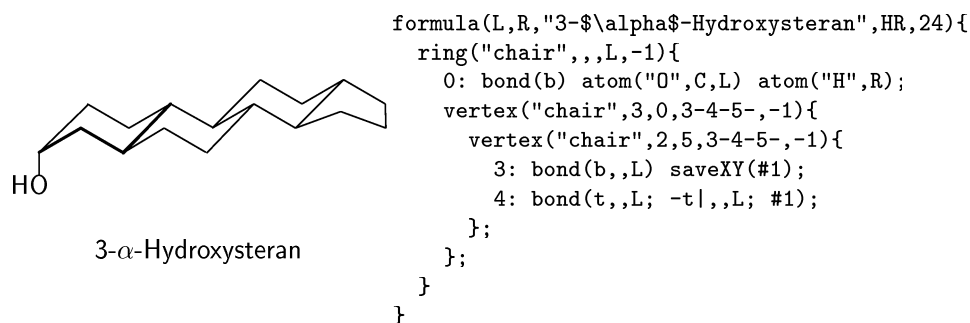
```
formula(L,R){
  ring("chair", , ,L){
    0: bond(b) atom("HO",R);
    5: bond(t) atom("HO",R);
    4: bond(b) atom("O",C,L) atom("H",R);
    3: bond(b) atom("O",C,R) atom("H",L);
    2: atom("O");
    1: bond(t; t-) atom("OH",L);
  }
}
```

The cyclohexane was constructed in such a way that the 1 \rightarrow 6 glycosid bonds of

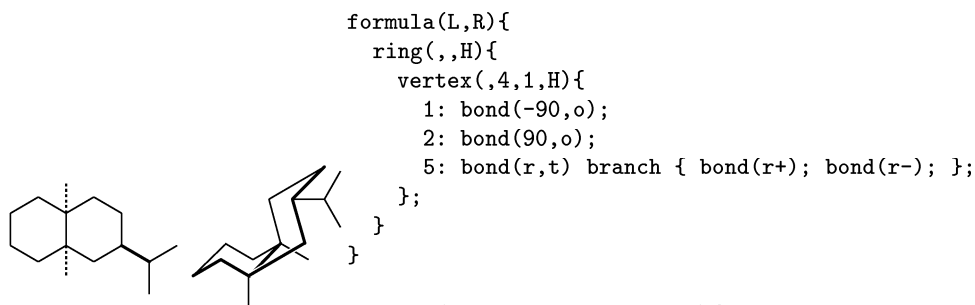
di- and polycarbohydrates should be drawn twice as long as normal with a `bond` command (length NN). Alternatively, the more common case is that you will place an ether oxygen between the two rings. This can be done by drawing a normal bond, followed by an oxygen and another bond. The second ring is set on the same height as the first one, so that oligo-saccharides are horizontally aligned:



Because the chair conformation is symmetrical, several rings may be joined together to delineate, for example, a steroid structure three-dimensionally. For the five-membered D ring, a wild bonding line must be drawn:



In the case of valerane, the melting of two ring edges with a usually different length turns out to be disharmonious:



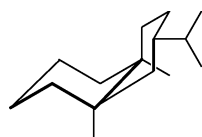
Valeran

```

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

```

The appearance of the formula is manually a little bit improved by the following method which mimicks the ring system B:



Valeran

```

formula(L,R,"valerane",HR,24){
  ring("chair",,,L,-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); }
  }
}

```

The flexible parameter of the cyclohexane structure offers the opportunity to choose between the possible main conformeres. Scheme 2–18 explains thereby the process of an elimination reaction:

```

\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);
    }
  }

```



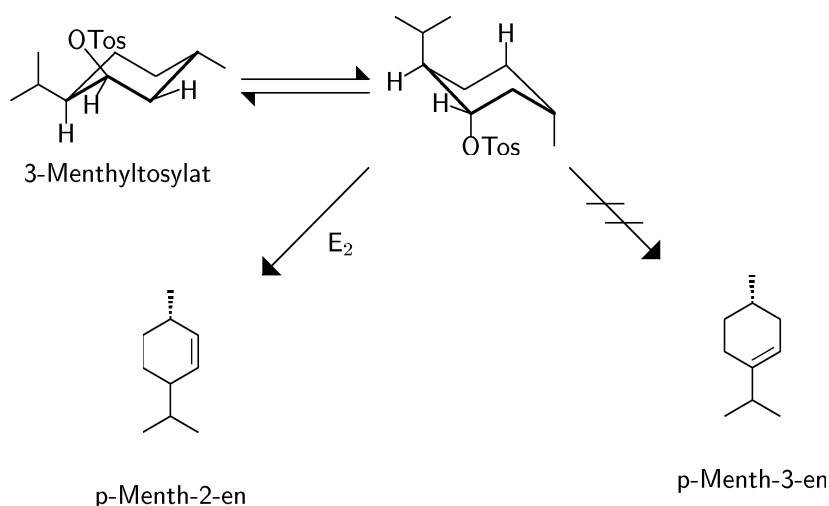
```

    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}

```



Scheme 2–18 The balance between the two main conformeres of cyclohexane uses the various representations of the same ring type.

2.5 Modification of parameters

As an example, the main reaction process in scheme 2–19 should be emphasized by thickened lines. The solution shows how the code for this technique is nested into a `save/restore` command pair. Within this, we change the parameter `$rLW`, which is responsible for the line width. The branching minor process must not be noted within the grouping:

```

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

```

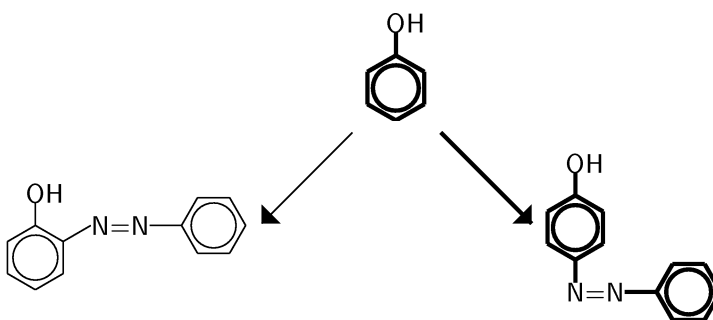
```

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(O)
          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(O)
          ring(,3,,,r){}; }
}
\end{chemistry}

```



Scheme 2–19 Main reactions can be separated from minor reactions by a thicker line. The line width can be chosen by the command `set`.

If you wish to multiply all parameters by the same factor, you can apply `→scale` instead of multiple `set` commands.

2.6 Depiction of electrons

An essential part of mechanistical formula representations is the rendering of the electronic configuration and the movement of electrons.

2.6.1 Configurations of electrons

The depiction of electronic configurations in regard to an atom is not really a part of OCHEM. But it can be achieved with the \LaTeX package `echem.sty`, which is included in the OCHEM distribution. It includes two macros, `\vd{<Atom>}` and `\vdd{<Atom>}`, which set beside the atom one or two electrons vertically one above the other. The electrons are symbolized by points. The atom symbol is only used to

determine the space around which the electrons should be positioned. The atom itself will not be drawn and it must (as the examples demonstrate) follow after the macro. Depending on the arrangement of the atom and the macros, the electrons will be set before or after the atom symbol:

•Li, Li•, :Be, Be•, :B•

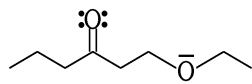
```
\vd{Li}Li,
Li\vd{Li},
\vdd{Be}Be,
Be\vdd{Be},
\vdd{B}B\vdd{B}
```

For a horizontal arrangement, we can exploit four macros. Macros for electrons above the atom symbol are `\hdu{<Atom>}` and `\hddu{<Atom>}`, macros for electrons below the atom symbol are `\hdl{<Atom>}` and `\hddl{<Atom>}`:

Li, Li, C̄

```
\hdu{Li}Li,
\vddu{Li}Li,
\vddu{C}\hddl{C}C
```

To include these into an OCHEM formula, the package must be loaded with an explicit `package` command into a `chemspecial` environment and into the document itself:



```
\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 Movements of electrons

An important detail arises in the case of mechanistical depictions. Little arrows within formulae or formula schemes are used to illustrate the shifts of electrons. Of course, this is almost an application for special CAD programs, but I tried to implement a useful command for precisely these electron movements. The matter is a little bit complicated, because the start and destination point cannot be linked just by a straight line. Therefore we need a bunch of parameters to describe the arc. The base for the shift arrow is a cubic spline, which is described firstly by the start and destination point, secondly by the angles, produced by the tangents at these two points, and lastly by two control points. The angles of inflection resp. reflection are in the most cases known so that chiefly the control points (that is, their distance from

the destination points) determine the bending of the arc. However, a good portion of experience and skillfulness is necessary to avoid a sense of despair because the arc gets always awkwardly bent.

The destination points of shifts are mostly placed in the center of bonds. This poses the problem that the bonds themselves should be drawn in their full length. To store this central position, beginning from the start point of the bond, we can apply `saveXY` with its extended syntax, i. e. angle and length. This syntax makes it possible to save the coordinates of a point which would be reached if a bond of a specified angle and length was drawn from the current position. The current position, however, is not changed so that the `saveXY` command may be followed directly by a `bond` command. Due to the central position, small caps (n) are used quite often with `saveXY`. The example illustrates the way in which a central position including the bond is stored and drawn:

```
saveXY(#1,30,n) % angle 30 degree, half the normal length
bond(30)        % angle 30 degree, normal length
```

The stored points are indicated by numbers. These serve as parameters for `emove`. Start and destination point may or may not appear not within one formula. More decisive is at any rate the actual point which is stored under the relevant number. So the `saveXY` commands should be continually numbered.

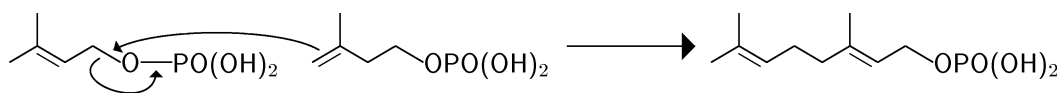
Take for example the chain-extending step of isoprenoid biosynthesis in scheme 2–20. Four points are saved in regard to an intramolecular transition and an intermolecular electron shift:

```
\begin{chemistry}
  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)
  }
\end{chemistry}
```

Another instance is exemplified by the Friedel-Crafts-acylation shown by scheme 2–21. In this case also we find intra- and intermolecular movements. The `shiftXY` commands help to position the three reactands in the correct relative location. The following reaction arrow is nested into the body of `multiline` and can thereby be directed at this formula triplet as a whole:



Scheme 2-20 The chain-extending step of the biosynthesis of isoprenoids is demonstrated by electron shifts.

```

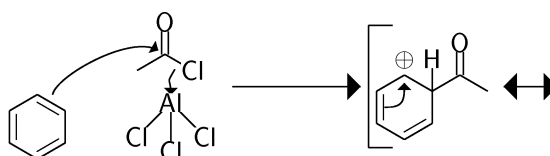
multiline(1)
{ formula(C,C)
  { ring(,,H1=3=5=){ 3: saveXY(#1,-30,n); }
  }
  shiftXY(50,30)
  formula(C,C)
  { bond(30) saveXY(#2)
    branch { bond(90,=C) atom("O"); }
    saveXY(#3,-30,n) bond(-30) atom("Cl");
  }
  shiftXY(0,-30)
  formula(C,C)
  { saveXY(#4) atom("Al")
    branch{ bond(-45,,L) atom("Cl");
            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,<->){}

```

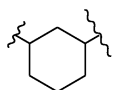


Scheme 2-21 The electron movement illustrated the process of the Friedel-Crafts-acylation of an aromate.

2.7 Cutting off bonds and molecule parts

An essential part of mechanistical formula representations is the rendering of the electronic configuration and the movement of electrons.

In a simple case, you just want to cut a bond to show only the interesting part of the molecule and indicate that there are more atoms which should not be shown. For this, you simply draw the bond (usually, this is done with half the length of a standard bond) and add another `bond` command with the bond type `l`. This type draws a cut-off line after the already drawn bond. The length of this line can be varied by using the length parameter of the `bond` command:



```
formula(L,R) {
  ring(,H){
    2: bond(r,,n) bond(r,l);
    4: bond(r,,n) bond(r,l,L);
  }
}
```

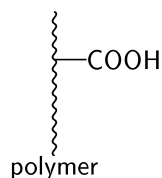
This method works fine if you want to cut off several independent bonds. But another common case is to cut off bigger parts of the molecule which are connected by two bonds to the visible part. Here, you use the command `cutline`, which also draws a cut-off line, but connects two given positions with this line. As an example, the right ring of naphthalene is cut off by this code:



```
formula(L,R) {
  ring(,H){
    4: bond(r,,n) saveXY(#1);
    5: bond(r,,n) cutline(#1,#cur);
  }
}
```

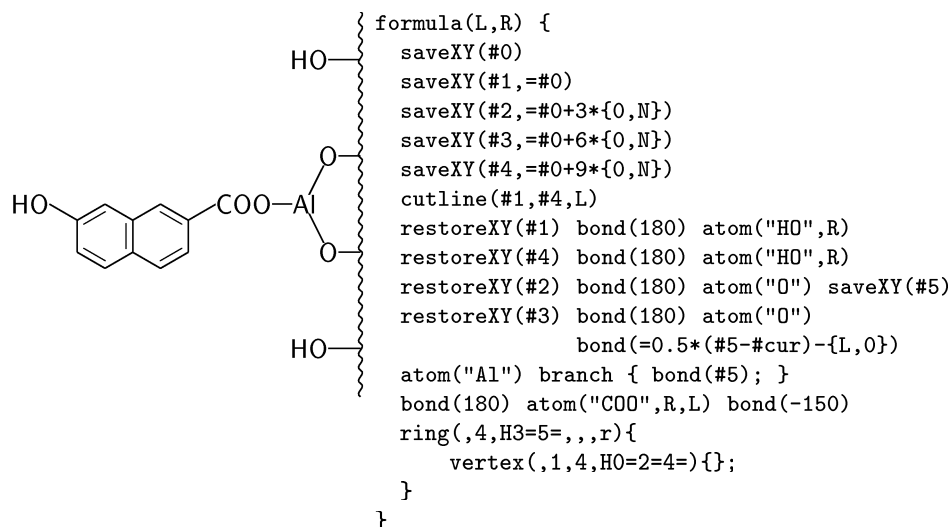
The optional parameter of `→cutline` allows you to specify the length by which the cut-off line extends the given positions as well as the line type.

The presented commands can not only be utilized to cut off molecule parts, but also to draw backbones of bigger molecules which should only be symbolized without concrete chemical structure, as i. e. in polymer backbones:



```
formula(L,R) {
  atom("polymer",C,T)
  bond(90,~,LL)
  branch {
    bond(0) atom("COOH",L);
  }
  bond(90,~,L)
}
```

The positioning possibilities offered by the `saveXY` and `restoreXY` commands allow you to draw backbones with several groups attached to the backbone:



2.8 Examples

At this stage, all elements usable in formulae are now familiar to you. Many of them are combined in the following section to describe two complete and fairly complex reaction schemes.

2.8.1 Nylon synthesis

Scheme 2–22 shows how the precursors of the nylon/perlon synthesis are prepared from simple compounds. The basic idea is to find a very long sequence of formulae and arrows which can be described easily and which minimizes the number of branches required. This can help you to keep the overview. Another attempt may be to summarize consecutive reaction chains due to chemical aspects. In the present example, this is not possible due to the existence of *two* start points: one chain starts from benzene, the other from butadiene. It would not be possible to lead both threads to a common formula in the arrangement shown here (with `joinh` or `joinv` this could be achieved, but would require a repositioning of the whole scheme).

The main line here starts from benzene and ends at butadiene – but stop, do not run the last two steps into the wrong direction? Not from the compiler's view which ignores the geometric representation of the elements. Reaction sequences against the direction are simply described with the arrow type `<-` instead of `->`!

The changes of directions occurring in the main line are properly described by suitable positioning parameters of the `formula` commands and the appropriate angles of arrows.

The chlorination of butadiene yields a mixture of two dichlorbutenes, being centered to the vertical axis of the reaction sequence with a single-line `multiline` block.

The plus sign is a hidden “formula”, containing only the text “+”. The name, which should be written centered below both formulae, can be produced with the help of an empty formula, only containing the name.

The main line is described as follows:

```

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

  arrow()
  { text(T,C){ formula(C,C){ atom("H$_2$/Pt, A1$_2$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("HOC",L,R) bond(30;-30;30;-30;30;-30)
    atom("COH", 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 $^{\circ}$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,"chlorobuten",V,12){}

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

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

```

Through the use of a main process we have two branches which lead to the end products and possess no specialities:

```

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

formula(T,B, "cyclohexanonoxime",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$^\circ$C") } }
  text(B,C){ formula(C,C){ atom("10 bar") } }
}

```

```

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

```

2.8.2 Terpene biosynthesis

The biosynthesis of the main representatives of bicyclic monoterpenes is another complex instance, which is depicted in scheme 2–23. It contains branches, various ring compounds and bold set formulae.

```

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

```

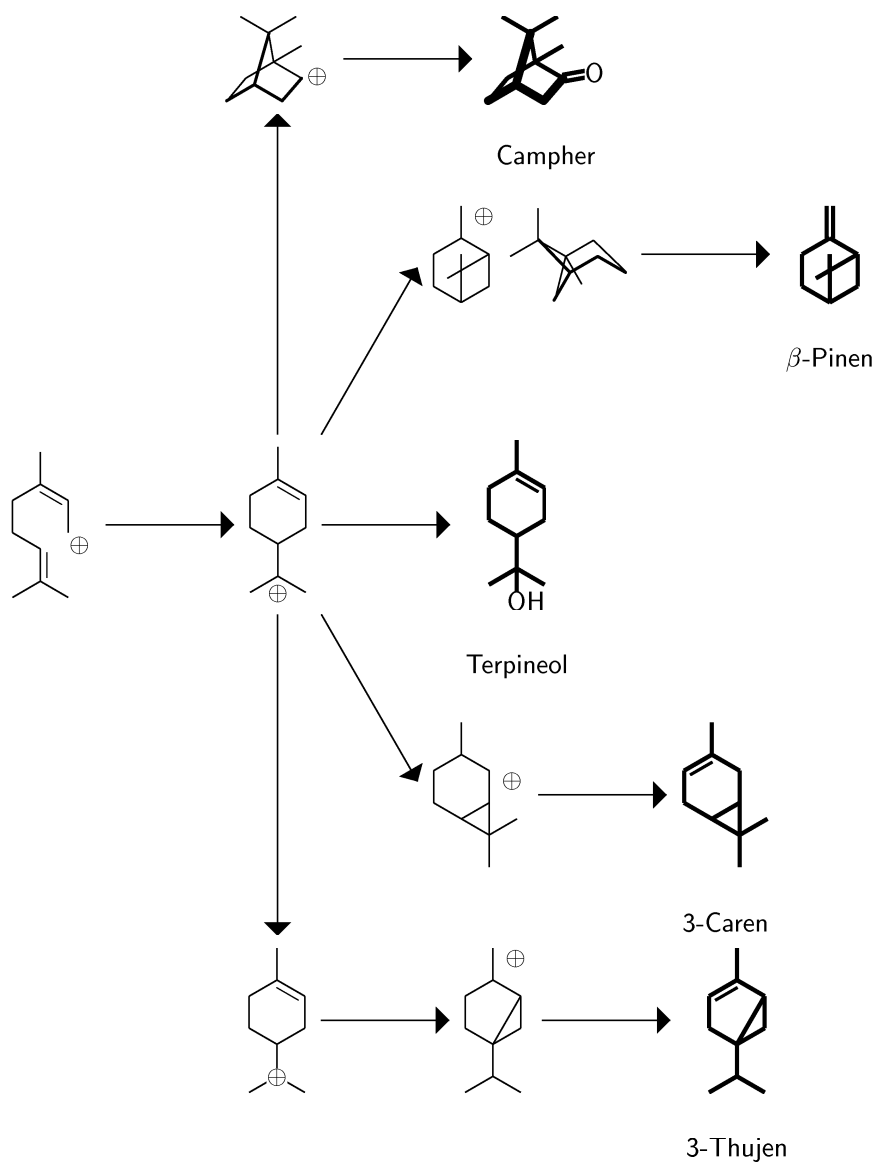
2.9 More compound classes

Many compound classes have not been discussed in this tutorial. Most of them can be realized with the means already explained. However, the repeated work with the same or similar formulae can be simplified a lot if a set of macros is prepared which is based on those similarities. Some hints for such macros are given in section 3.6. The following classes of compounds are described in more detail there:

- steroides
- isoprenes
- amino acids and oligopeptides

2.10 Development of ring structures

To avoid overloading the compiler with a lot of seldomly used ring structures, only basic structures listed under `ring` have been taken into the core. It is however pos-



Scheme 2–23 Biosynthesis of bicyclic monoterpenes out of the straight-chain precursor farnesol.

sible to define individual ring structures within a perl module. With the help of `require`, these can be provided in a document.

To illustrate the construction of such a module, you find the two files `monocyclib.pm` and `bicycliclib.pm`. Beside the basic vocabulary they also provide mono- and bicyclic compounds. You can use these files as templates; what must be primarily supplied in a structure definition is:

- the coordinates of each atom in the array `@$XYa`,
- the bond types (single, double, projective ...) in the array `@$BT`,
- a list of atoms which serve as start points for bonds, in the array `@$BLa`,
- a list of angles of the bonds in the array `@$rThetaa`,
- a list of bond lengths in the array `@$rLena`,
- the amount of atoms in a compound in `$$iAnz`,
- the ring size, usually the length of a typical ring edge, in `$$rLen`,
- information whether an aromatic ring shall be drawn, in `$$bAromat`, also its radius in `$$rRadius`,
- if necessary the settings for the two free parameters.

For each ring structure you must denote two functions:

`<type>_default_` : This function presets the default values offered by the `ring` to the outside. Among these, the amount of the atoms and the list with the bond types are most important. The latter should be able to be overwritten by a possibly following bond list.

`<type>_` : This function contains the actual definition of the ring structure. All parameters, including the two flexible parameters `<p1>` and `<p2>`, are known and can be applied.

The array `@$XYa` contains the coordinates of the atoms measured in point (pt), which build up the compound. These can be put in relation to a freely choosable origin point and can be denoted with either fixed coordinates or – for geometrical compounds – be calculated. It is always recommended to include the length parameter into the coordinates because structure sizes remain scalable. In basic compounds, it denotes the length of an edge and thereby determines the overall size.

The list `@$BLa` contains a series of numbers of the atoms from which bonds take their start. A certain bond is linked with a corresponding atom number in this list. The link is determined by the sequence in which the members of both lists appear. The bond is described by data which occurs at the corresponding position in several properties lists. Monocyclic systems are described by a simple list `@$BLa` which consists out of figures from zero to the number of atoms minus one. In an analogous way, the property lists contains as many items as there are atoms. The list `@$BT` is easily built and simply contains the numbers corresponding to the bond types (they can be looked up in `streambuf.pm` in the function `GetBond`). The two other lists may occasionally create difficulties if the need for calculation of angles and lengths arises. However, the functions

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

may come handy to find out the angle and length of a bond between the atoms with the coordinates <p1> and <p2>. The start point of the bond is attached to the atom <p1>.

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

may deliver the destination point of a bond from point <p1>, which possesses a given direction and length. Wether you know the coordinates of the compound's atoms or construct it with the help of given bond angles and lengths, in any way you attain the missing data by using one of the two function sets.

3. Alphabetic command reference

This chapter contains a complete description of all commands and their syntax. Considerations for the free parameters are:

- Specifications as bond types or bond lengths are given by symbols, with their appearances as close as possible to their typeset counterparts.
- Angles are measured in degrees (“Altgrade”) in the mathematically positive sense (counter-clockwise). They are related to the positive x-axis.
- Absolut lengths are measured in points (pt) *without* unit.

Most of the parameters are optional ones, having commonly a default value. This simplifies the writing of code considerably. After the last parameter with a given value the closing bracket can be written, a notation of later parameters or commas is not necessary. Note that in case you want to specify a later parameter’s value, you have to write all separating commas to define the position of the parameter correctly. It is sufficient to write only the commas, if you do not want to specify values for these parameters.

Symbols in the syntax descriptions stand for:

```
<flist> := atom | bond | branch | orbital | saveXY | ring |  
         save | restore | set  
  
<slist> := arrow | formula |  
         savecontext | setcontext | gotoXY | shiftXY |  
         emove |  
         save | restore | set | scale  
         bracket | fbox
```

3.1 User commands

This section contains all commands that are allowed to appear in a \LaTeX document’s `chemistry` environment or `schema` command, respectively.

3.1.1 arrow

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

<Tpos> := L | C | R

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

This command creates reaction arrows, joining formulae to yield reaction chains. The arrow's appearance is determined by the parameters <phi>, <len> and <type>. The direction of the arrow is measured with an angle <phi> in degrees and in mathematically positive sense in regard to the positive x axis.

<type> determines the type of the arrow (balance arrow, "leads to" arrow, strikethrough arrow) according to the following list:

 <=>	 <->
 <-	 < -
 ->	 - >
 -	

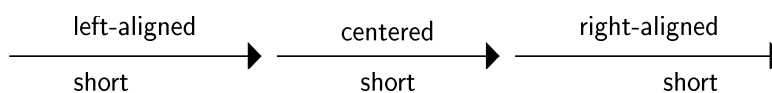
The parameter <len> modifies the length of the arrow and specifies the distance which must at least be present after the start and before the end of the arrow. An arrow's minimum length is therefore 2<len>. The specification of the length via a half of the length becomes clear, considering the possibility of labeling the arrow: the total arrow length is one half of <len> before the text, the length of the text itself and one half of <len> after the text.

Text blocks or formulae may appear above or below the arrow. The position is specified with the positioning parameters T and B in the command `text`. Due to the possible appearance of text blocks or formulae, a `formula` description or reaction sequence is expected. Simple text blocks are therefore to be set with the command `atom` within a formula.

The parameter <Tpos> becomes important, if you typeset formulae or text blocks above and below the arrow. In this case, you can specify the alignment of both formulae with the values L, C und R, thereby creating left-aligned, centered or right-aligned text blocks, as you can see in scheme 3-1.

Scheme 3-2 shows how the arrow's text blocks are positioned due to different arrow angles.

As mentioned above, it is possible to use complete reaction schemes instead of simple text blocks over an arrow, as you can see from scheme 3-3. This figure illustrates a side reaction, leading to the actual adduct:



Scheme 3-1 Alignment of two text blocks above and below an arrow.

```

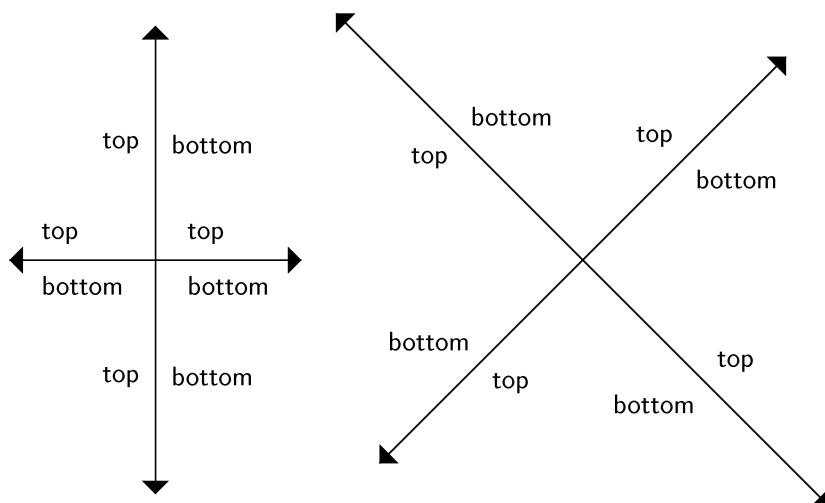
\begin{chemistry}[arrow2]
  formula(L,R)
  { ring(){} }

  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) }
    }
  }

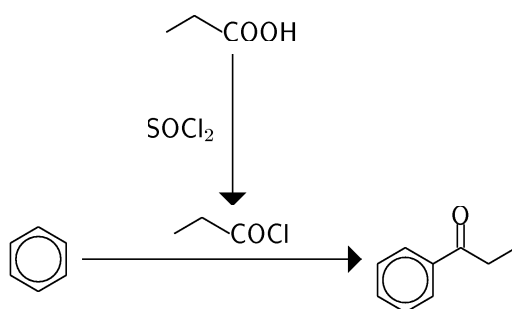
  formula(L,R)
  { ring()
    { 4: bond(r) branch { bond(r+,=C) atom("O"); }
      bond(r-; r); }
  }
\end{chemistry}

```

Usually, a small space is set after an arrow in the direction of the continuation point to prevent several arrows from sticking too densely together. If this effect is intended, you may denote the command `noindent` immediately after `arrow` to inhibit this behaviour and set arrows and formulae without additional space.



Scheme 3-2 Positioning of the text blocks above and below an arrow, due to different angles of the arrow.



Scheme 3-3 Complete reaction chains act like labels if they are set above arrows.

3.1.2 atom

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

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

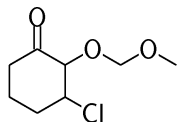
```
<Cpos> := C | L | R | T | B | TL | TR | BL | BR
```

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

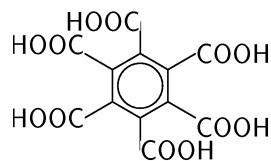
```
<Cpos> = C
```

This command inserts arbitrary TeX text at the actual position with an alignment specified by <pos> relative to the actual position. This actual position may be influenced by the text's size due to the continuation directive given in <Cpos>.

The following formulae are examples, showing the different possibilities of text symbols. The simplest case is an element symbol which terminates a bond and is therefore center-aligned with C at the bond's end. With longer texts like COOH is it advisable to right- or left-align the text. In both cases, no continuation from this atom is required, the parameter <Cpos> is therefore unused. If further bonds branch from this atom, the parameter gains importance: nicely demonstrated by the methylene group which is reached from the right (<pos> is R) and from which a bond leads on to the left (<pos> is L):

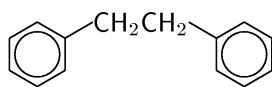


```
formula(L,R){
  ring(,H){
    3: bond(r,=C) atom("O");
    4: bond(r) atom("O") bond(r-; r)
      atom("O") bond(r-);
    5: bond(r) atom("Cl");
  }
}
```



```
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);
  }
}
```

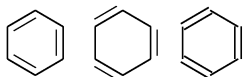
It is easily seen that long text blocks like COOH tend to be placed quite disharmoniously. In the tutorial on page 19 some information about such cases is given.



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

Typesetting symbols and texts with \TeX means that you are not limited to simple letters. Mathematical formatting, indices, the symbols \oplus and \ominus as well as greek letters can be used. Even user-defined commands can appear as text, in this case it is necessary to collect the definitions in advance in a package file. The method is described in section 3.2.2.

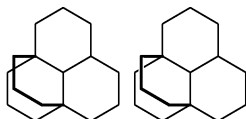
two bonds equally spaced and positioned to the virtual center of the bond. They are not designed for working in ring structures, but for the representation of exocyclic bonds and bonds leading to terminating atom symbols. The differences between the formulae are shown in the following figure:



```
formula(L,R){ ring(, ,H0=2=4=){} } space(R)
formula(L,R){ ring(, ,H0=U2=U4=U){} } space(R)
formula(L,R){ ring(, ,H0=C2=C4=C){} }
```

3 : This is a triple bond.

t, p, << : Here we have the following types of thick bonds (t); thick bonds with an additional white border (p) and a bond becoming larger (<<). The types t and << are both suitable to represent perspective bonds, coming out of the paper plane, as the examples for the next bond types show. The bond type t is also applicable to symbolize bridges, lying above the paper plane in complicated polycyclic systems. To increase illusion, type p draws a white border around the bond. If you regard the following compound, you can compare both variants. The effect of p is only visible if bonds cross other bonds:



```
formula(L,R) {
  ring(, ,H){
    vertex(,4,1,H){};
    vertex(,3,0,H){};
    3: bond(180,t,S; -90,t; -30,t; 0,t,S);
  }
}
formula(L,R) {
  ring(, ,H){
    vertex(,4,1,H){};
    vertex(,3,0,H){};
    3: bond(180,t,S; -90,p; -30,t; 0,t,S);
  }
}
```

When using this hack, you have to consider the *sequence* of drawing bonds. Bonds of type p only become effective when drawn at *last!* To keep the illusion of the perspective in the example above, you cannot change the sequence of the rings.

<., o : This type is a dashed, self-broadening bond or a dashed bond, respectively. They are used for the perspective representation of bonds lying behind the paper plane:



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

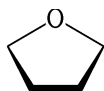
The fine-dashed line o is suitable for bonds in statu nascendi or to illustrate hydrogen bridge bonds:



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

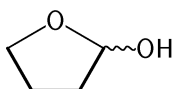
>>, >., b : These bond types correspond to narrowing, filled-out, narrowing dashed or broad bonds and are useful for the depiction of bonds of an atom, lying above the paper

plane with bonds stretching into it. The broadness of the bond of type *b* (tuned by the parameter *rBW*) correlates to the maximum broadness of the broadening or narrowing bonds so that all three forms can easily be combined. See for example the three-dimensional delineation following HAYWORTH:



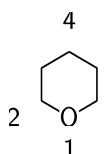
```
formula(C,C){
  ring("furanose",0,1<<2b3>>,L){
    0: atom("O");
  }
}
```

- ~ : This type of bond will be used if the stereochemistry of an atom should not further be specified or if there is a mixture of anomeres, as an example from the saccharide chemistry shows:



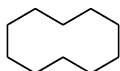
```
formula(C,C){
  ring("furanose",,,L){
    0: atom("O");
    1: bond(0,~) atom("OH",L);
  }
}
```

- s : This is an invisible bond and can be used to reach points near an atom. An example would be the numbering of an atom compound:



```
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"); }
}
```

As you can see from the cyclodecane, invisible bonds can also be used to rub out lines in a polycyclic system:



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

- >, <- : These arrow-like types are used to represent complex bonds. The atom is denoted which spends the electron pair:



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

- | : This bond type indicates a non-saturated bond which still continues. It can be used to show only parts of a molecule, but still to give a hint of the rest, like for example in the β headpart of the carotene nomenclature:



```
formula(C,C){
  ring(,H4=){
    3: bond(b);
    3: bond(t);
    5: bond(r);
    4: bond(r,,n;r,l);
  }
}
```

Bond lengths

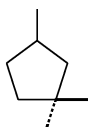
The length of the bond is coded by the letters S (short), N (normal) and L (long). A small letter codes a length half as long as that of the corresponding uppercase letter. Each larger letter signifies a length increased by 50%. If the same letter appears more than once, then the bond will be prolonged the corresponding amount of length of this letter. NN exemplifies the double normal length. As a consequence, some lengths may be represented in several ways: nn is the same as N, NNN relates to LL. The actual length of each bond is stored in the variables `rLenS`, `rLenN` and `rLenL` (\rightarrow set). The special length 0 supplies a bond of the length zero, which may be useful if you have written a general macro in which one of the bonds should not be always present.

r/t/b/l/v symbolic angles

This syntax is identical to the normal form in all points except the specification of the angle. This syntax variant is intended to help constructing side chains on ring systems and is mostly used in `ring` or `vertex` commands. The letters are symbolic angle specifications with their concrete value depending on the actual ring system and the ring position the side chain is starting from.

r : symbolizes a radial bond, meaning a bond with its direction determined by a line center of ring-ring atom.

t, b : Angles for bonds above (top) or below (bottom) the paper plane. Especially useful with the ring type `chair` and other conformeres. These bond types are also applicable in planar formulae to draw bonds prolonging ring vertices. Which angles are really used for these variant is discussed in schematic form when presenting the individual ring types (command `ring` and introduction of the libraries).



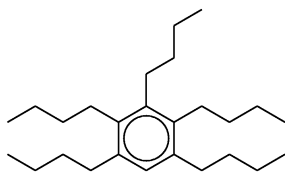
```
formula(L,R){
  ring(,H,,5,90){
    0: bond(r);
    2: bond(t,t);
    2: bond(b,o); }
}
```

v(<n>) : The angle of this bond is determined by the bond from the ring atom labeled <n> to the next atom in the ring.

1 : Equals to the angle of the last-drawn bond.

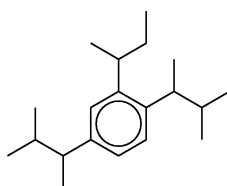
These basic angles may be modified by different symbols in the following sequence (single or in any combination):

/ : This specification transforms the angle of a **r**, **t** or **b** symbol into its complementary angle. This is the angle leading to the “zick-zack” presentation of linear chains when combined with its normal angle in a sequence `r r/ r r/ r`. The resulting chain is oriented at the x and y axis, respectively. The inner angle at the corner is always 60°.



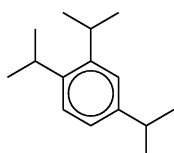
```
formula(L,R){
  ring(){
    1: bond(r; r/; r; r/);
    2: bond(r; r/; r; r/);
    3: bond(r; r/; r; r/);
    4: bond(r; r/; r; r/);
    5: bond(r; r/; r; r/); }
}
```

t : This produces an angle which is the counterpart of the angle /, therefore yielding a branch. The combination /t supplies a branch to the opposite side of the zick-zack-chain:



```
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); }
}
```

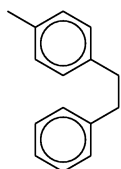
+, - : These specifications may be present more than once and increase or decrease the bond angle by 60° each. This is the normal angle which builds alkane chains in zick-zack-representation and also the main angle in six-member rings. The effect is a turn out of the actual direction to the left or to the right. You get changes in the chain's directions or isopropyle branching:



```
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-); };
  }
}
```

To symbolize longer, methyl-branched chains, the specification of the angles is better done with `rt` or `r/t`.

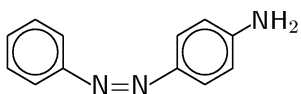
| : This symbol mirrors a bond at the vertical axis. You can easily get symmetric molecules:



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

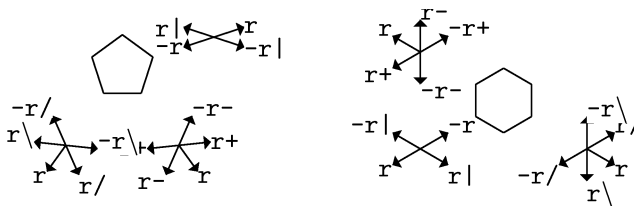
It is possible to summarize a given angle $\langle \text{phi} \rangle$ and the angle determined by a combination of the aforementioned symbols. This can be used to e. g. describe a 10°-turn out of the radial direction. The operator + returns the normal sum of both angles, - returns the sum of the given angle and a symbolic angle running into the *opposite* direction.

- can also be used to draw bonds in opposite to a given direction:

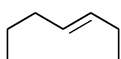


```
formula(L,R){
  ring(){
    5: bond(r) atom("N=N",L,R) bond(-r|)
    ring(,0,,,r){
      3: bond(r)
        atom("N",C,R) atom("H$_2$",L);
    }
  }
}
```

A more detailed explanation of all these variants as well as numerous examples are given in the tutorial section 2.2.1. A short conclusion of some important directions, yielded by combinations of symbolic angles follows here:

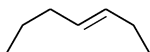


At the beginning of a formula environment, the symbolic angles are associated with the default angles 30° for `r`, 90° for `t` and -90° for `b`. This makes it possible to use the symbolic angles out of a ring system, too:



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

To change the default values, use `→set`:



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

#-syntax

The syntax variant starting with `#<n>` supports bonds of which the direction and length is not specified, but calculated automatically so that the resulting bond connects the actual atom and a position `<n>`. The latter position was formerly saved with `saveXY`. Examples can be found in `→saveXY` and `→restoreXY`.

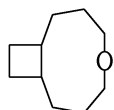
Chained bonds

The descriptions of several subsequent `bond` commands can be compressed into a single `bond` command, wherein each individual bond is separated by a semicolon. All syntax variants can be used as well as combinations of them. As example, the following sequences are equivalent:

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

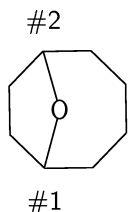
```
bond(r) bond(r/) bond(r)
== bond(r;r/x)
```

Complex 2D-terms with the ==-form



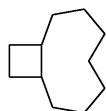
```
formula(L,R) {
  ring(,H,#N,4,-45) {
    0: bond(-70; -10; 50) saveXY(#1);
    3: bond(70; 10; -50)
      bond(=0.5*(#1-#cur)) atom("O") bond(#1);
  }
}
```

To calculate the central point and to shift it to the left at the very same time, we can use the addition of the central point and a certain amount of movement (by the length of a normal bond length):



```
formula(L,R) {
  ring(,H,#L,8,-22) {
    2: saveXY(#1);
    5: saveXY(#2) bond(=0.5*(#1-#2)+{N,0}) atom("O") bond(#1);
  }
}
```

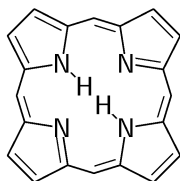
The second position does not need to be explicitly stored because it can be easily extracted with #cur (this stands for "current"):



```
formula(L,R) {
  ring(,H,#N,4,-45) {
    0: bond(-70; -10; 50) saveXY(#1);
    3: bond(70; 10; -50)
      bond(=0.5*(#1-#cur)-{N,0} ;#1);
  }
}
```

The complex example illustrates how the porphine compound can be built up. Because of its symmetry, an undefined origin is chosen and the rings are at the four points {N,N}, {N,-N}, {-N,-N} and {-N,N} arranged in such a way that the ring atom 1 with nitrogen is attached at this position. (Please note that the notification {N,-N} is not allowed, the correct form is {N,0}-{0,N}.) `saveXY(#cur,<expr>)` means that the value of <expr> (which is the location of an atom) will be used as actual position (#cur instead of, for example, #1).

The atoms 2 and 5 of each ring which form later on the bridge heads, are stored in the positions 1 to 8 to provide for the delineation of the four bridges.



```

formula(L,R) {
  saveXY(#0)
  restoreXY(#0) saveXY(#cur,={N,N})
  ring(,0,H2=4=,,5,-135){
    0: atom("N");
    1: saveXY(#1); 4: saveXY(#2);
  }
  restoreXY(#0) saveXY(#cur,={N,0}-{0,N})
  ring(,0,H2=,,5,135){
    0: atom("N") bond(135) atom("H");
    1: saveXY(#3); 4: saveXY(#4);
  }
  restoreXY(#0) saveXY(#cur,={0,0}-{N,N})
  ring(,0,H0=2=,,5,45){
    0: atom("N");
    1: saveXY(#5); 4: saveXY(#6);
  }
  restoreXY(#0) saveXY(#cur,={0,N}-{N,0})
  ring(,0,H1=3=,,5,-45){
    0: atom("N") bond(-45) atom("H");
    1: saveXY(#7); 4: saveXY(#8);
  }
  restoreXY(#8) bond(=0.5*(#1-#8)+{0,n}; #1,=)
  restoreXY(#2) bond(=0.5*(#3-#2)+{n,0}; #3,=)
  restoreXY(#4) bond(=0.5*(#5-#4)-{0,n},=; #5)
  restoreXY(#6) bond(=0.5*(#7-#6)-{n,0},=; #7)
}

```

3.1.4 bracket

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

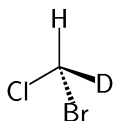
This command inserts brackets around the last single formula created with `formula` or the last partial scheme created with `multiline`. When executed after commands other than the two mentioned above, the results are unpredictable! You can choose whether you want to get a left bracket, a right bracket or both, in which case you have to specify an empty parameter list. The brackets are useful to comprise several individual mesomeric structures or a mixture of intermediates.

Details about bracketing in complex situations can be found in the section 2.3.5 of the tutorial.

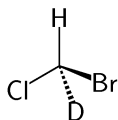
3.1.5 branch

```
branch{ <alist_1>;
        <alist_2>;
        ...
      }
```

This command allows you to build up branched compounds. Each branch `<alist>` starts at the current position. In the following example, three chains branch from the central carbon atom. The command lets the current point unchanged, so it is possible to continue from the central point with the third (last) chain, and only two branches have to be described within `branch`:



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



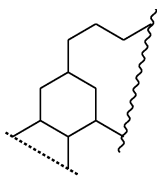
```
formula(L,R){
  atom("Cl") bond(30,,L)
  branch { bond(90,,L) atom("H");
           bond(-20,<<,L) atom("Br");
         }
  bond(-70,<.,L) atom("D")
}
```

This command is exceptionally suitable for a structured human mind, because of its way to work hierarchically-recursively. If you are more of an unconventional type, you might prefer to save a position (`→saveXY`) and later return to this position (`→restoreXY`, `→bond`) to continue with another branch. This transforms hierarchical branches in a linear sequence wherein each branching point has to be saved.

3.1.6 cutline

```
cutline(#<n>, #<m> [, [<extend>] [, <type>]])
```

This command draws a line with (bond) type <type> from point A, specified by #<n>, to point B, specified by #<m>. The default bond type is ~. For each point, the identifier #cur can be used instead of an integer, specifying the current point. At each end of the drawn line, an extent of size <extend> can be specified as a bond length expression (default is the normal bond length N). The following formula gives a short example, showing decaline with a partially cutted ring:



```
formula(L,R) {
  ring(,H){
    3: bond(r; r-; r--; r-) saveXY(#3);
    0: bond(r) saveXY(#1);
    1: bond(r) cutline(#1,#cur,s,o);
    5: bond(r) cutline(#cur,#3);
  }
}
```

The command is used to indicate that parts of the molecule are not shown here. You can also symbolize parts of i. e. a macromolecular chain with this command. Details are explained in the tutorial, section 2.7.

3.1.7 emove

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


(Only available in PostScript output format!) The command draws small, straight or arched arrows to illustrate electron movements. Arrows can start in one formula and end in the same formula or connect points in two different formulae. An arrow tip is drawn at the arrow's end. The command `emove` itself is only allowed to appear outside a formula body, similar to `formula` or `arrow`.

The start and end points of the arrows are previously saved (`saveXY`) points, identified by their numbers `<i>` and `<j>` respectively. A distance `rEmove` is inserted between arrow and specified point to avoid the circumstance that the arrow directly sticks to a bond. The arrow is described by a cubic spline, the shape is fine-tuned by two angles `<iAngle>` and `<jAngle>` and two real numbers `<iCtrl>` and `<jCtrl>`. The angles describe the angle under which the tail or tip of the arrow leaves or enters a target position. The lines, leaving the target atoms under the given angles, can be thought of as tangents to the arrow in the given points. In the distances `<iCtrl>` and `<jCtrl>` respectively, lie the two control points necessary for a PostScript spline. The amount of the start, end and the two control points controls the exact drawing of the spline at which end a little arrow tip is attached.

To draw a concrete electron transfer, the two angles are most easy to choose, because they are determined by the desired input and output angles at the target atoms. Experience (and maybe good luck) however, is required to find values for the control parameters to achieve an aesthetically arched arrow.

Some examples may show you the influence of the control parameter:

```
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)
```



Two topics can be seen: a) the control parameters' influence on the degree the arrow is arched grows the greater the distance becomes between them (the control points lie farther away from the start and end points). All four main points (start point, end point and the two control points) form a trapezoid, in which the spline curve is embedded, whereby each control point wants to attract the arc. The fourth ex-

ample shows an extreme case, the curve is intertwined. b) Most of the target points of electron movements lie in the center of bonds. The expanded syntax of `saveXY` simplifies the formula description to a large extent, because only two commands are necessary to save the bond's center point and to draw the bond itself. A symbolic representation of the half of a bond length is `n` instead of `N` for normal bonds. The tutorial, section 2.6, shows more examples.

3.1.8 fbox

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

The command draws a frame around the last formula, created with `formula`, or the last block of formulae, created with `multiline`. You can get further information about frames around complex formulae in the tutorial, section 2.3.5.

3.1.9 formula

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

This is the central command of the package. It builds formulae out of basic structural elements, which are subsequently transformed into an enclosed unit. This unit is then positioned automatically in relation to other formulae, which builds whole reaction schemes with the element `→arrow`. Some more advanced commands like `→multiline` or `→joinh` require formulae as base.

The formula itself, sitting in the command's body as `<flist>`, is described by a sequence of basic elements (`→atom`, `→bond`, `→ring`, `→branch`). This unit is positioned in such a way that it has the relation `<pos>` to the start point. The connection point for all following formulae is determined by the parameter `<Cpos>`. The meaning of all parameters is shown in the tutorial in more detail.

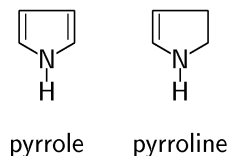
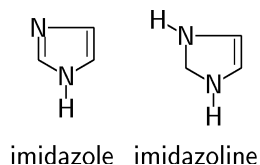
Usually, a space is left after a formula in the direction of the connection point to prevent the formulae following directly each other. If you intend your formulae to behave exactly this way, insert `nospace` immediately after `formula`.

3.1.10 gotoXY

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

Defaults: current x- and y-values

This command sets the current point to the specified absolute position. This seems to be in contrast to wishes for automatic positioning of formulae and elements in a reaction scheme and in fact, it is! But it may make sense, if you want some formulae to appear at certain places, independent of the randomness of scheme sizes. Think about table-like arrangements:



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

A relative arrangement can be achieved by command `→shiftXY`.

3.1.11 joinh

```

joinh(<n>, <pos>)
{ <rlist_1>;
  ...
  <rlist_n>;
}

<pos> := L | C | R

```

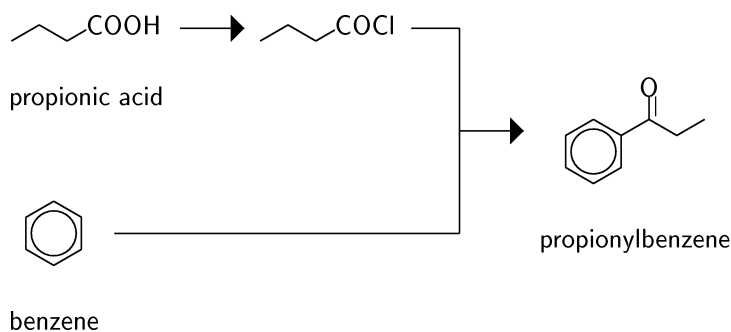
This command joins $\langle n \rangle$ horizontal reaction chains. The individual chains are positioned in such a way that they run one above the other in an equal distance. The parameter $\langle pos \rangle$ controls the alignment of all lines: left-, centered or right-aligned. If labels for formulae occur, they will be set with a constant distance from the center line. The vertical minimum distance corresponds to the value of the parameter `rTextSep`. Due to this automatic text positioning feature, named formulae are *not* allowed to have distance parameters! Unnamed formulae do not influence the distance of text to the center axis.

The connection point for subsequent reaction arrows lies on the right side in the middle of the formula block. The following example shows a convergent synthesis, joining two separate chains to a single one. Typically, `joinh` is followed by an `arrow` command, because `joinh` does not draw arrow tips (if several `joinh` commands would be nested, superfluous arrow tips would occur otherwise). All part chains begin left-aligned due to the given parameter L:

```

\begin{chemistry}
  joinh(2,L){
    % upper line
    formula(L,R,"propionic acid"){
      bond(30;-30;30) atom("C",C,R) atom("OOH",L) }
    arrow(,12){}
    formula(L,R){ bond(30;-30;30) atom("C",C,R) atom("OCl",L) }
    ;
    % lower line
    formula(L,R,"benzene"){ ring(){} }
    ;
  }
  arrow(,12){}
  formula(L,R,"propionylbenzene",HR,24){
    ring(){ 4: bond(r) branch{ bond(rt,=) atom("O"); }
            bond(r/;r); }
  }
\end{chemistry}

```



3.1.12 joinv

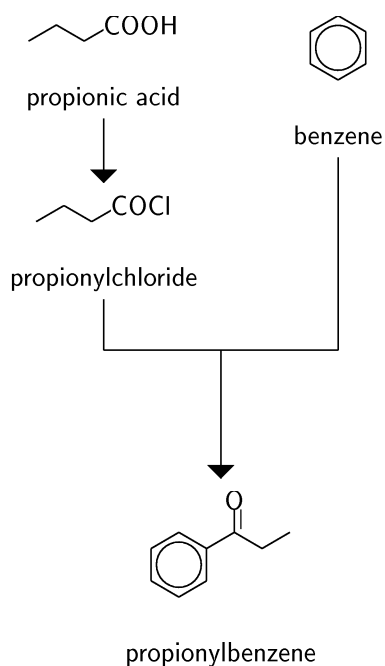
```

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

<pos> := T | C | B

```

Similar to \rightarrow joinh, this command joins <n> individual vertically oriented chains into a single one. The individual chains start at the top and are collected at the bottom; the continuation point lies in the middle of the lower border of the resulting block. Usually, you continue with an arrow tip here. All chain's top borders are aligned according to the parameter <pos>: top-aligned, centered or bottom-aligned. The \rightarrow joinh example is shown vertically oriented (top-aligned due to the T positioning):



```

joinv(2,T)
{ % left line
  formula(T,B,"propionic acid",V,24)
  { bond(30;-30;30) atom("C",C,R) atom("OOH",L) }
  arrow(-90,12){}
  formula(T,B,"propionylchloride",V,24)
  { bond(30;-30;30) atom("C",C,R) atom("OCl",L) }
  ;

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

arrow(-90){}

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

```

3.1.13 multiline

```

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

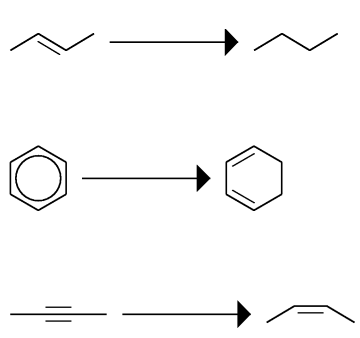
<iPos> := L | C | R
<pos> := C | L | R | T | B | TL | TR | BL | BR
<cPos> := C | L | R | T | B | TL | TR | BL | BR

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

```

This command typesets <n> lines with formulae or reaction schemes one below the other, keeping a constant distance between all neighbored lines. The first line <rliste_1> is typeset at the top of the stack. All lines are aligned according to <iPos> (left-aligned, centered or right-aligned). The block of <n> lines is completely positioned in relation to the current point according to <pos>. The connection point for subsequent formulae is determined by <cPos> (like the positioning of `formula`). Texts below formulae are typeset bottom-aligned in an individual line, keeping a fixed minimum distance from the highest formula in the line. Due to this automatism, no distance parameters are allowed for formulae with text!

In the easiest case, the command is used to split long reaction schemes into several lines, or to typeset several short chains in a stack:



```

\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}

```

The special case <n> is equal to 1 can be used to collect several formulae into a single unit, which can be placed or framed or serve as branching point. See section 2.3.5 in the tutorial.

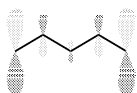
3.1.14 orbital

```
orbital([<i>Angle</i>] [, [<rWeight>]])
```

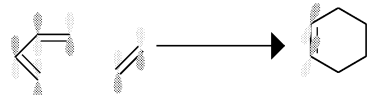
Default:

```
<rWeight> = 1
```

(Currently only available in the PostScript output format!) The command allows you to draw p-orbitals, e. g. to illustrate some quantum mechanical influences of a cyclo addition. The parameter <i>Angle</i> is the angle between the axis of the orbital and the positive x axis, <rWeight> is a scaling factor, specifying the size of the orbital. The upper and the lower half of the orbital are identified by different gray shades. Some examples:



```
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); }
}
```

3.1.15 restore

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

This command restores a parameter set, previously stored under the integer number <n> (→save).

3.1.16 restoreXY

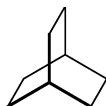
```
restoreXY(<#<n>)
```

The command restores a point which was previously stored with `→saveXY`, labeled `<n>` and then sets the current point to it. So you achieve branches in structures without the need of recursive thinking, as `→branch` requires. We call this “wild branches” :-) Examples:



dimethyl-cyclopentane

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



Bicyclo[2.2.2]octane

```
formula(C,C,"Bicyclo[2.2.2]octane",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)
}
```

The examples of the branch command can be written with `saveXY` and `restoreXY` as follows:

```
\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}
```

It should be mentioned that bicyclo[2.2.2]octane is contained as a basic compound in the library module `bicyclib.pm`. Different conformers of cyclopentane are found in the module `mncyclib.pm`.

3.1.17 ring

```

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

  % nur bei <typ> = "ring", "cpentane", "chair":
  [ vertex([<typ2>], [<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|O|1|...|<n-1> : <fliste>

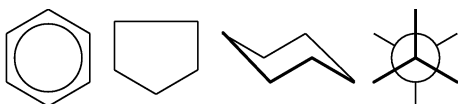
Defaults: <typ>      = "ring"
          <start>    = 0
          <bliste>    = 0
          <len>      = N
          <p1>       = 6
          <p2>       = -90

```

This command produces pre-fabricated ring structures. The extended syntax with `vertex` is only available in conjunction with the ring types mentioned above.

Fixed parameters

The parameter `typ` specifies which basic compound is to be drawn according to the following code:

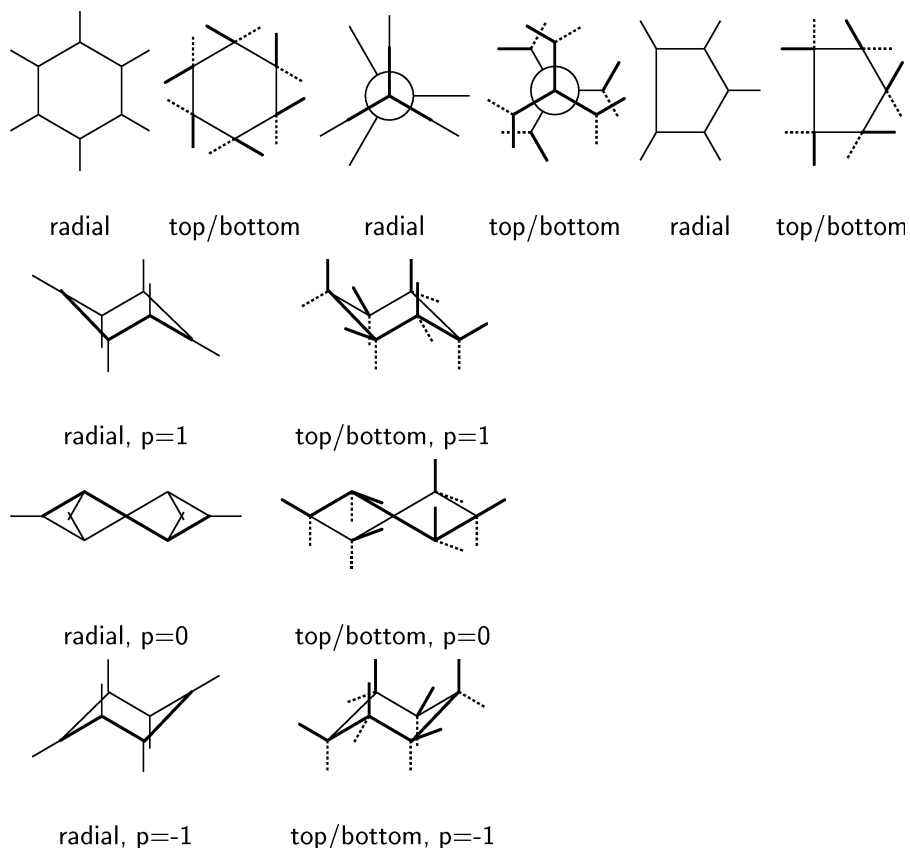


ring cpentane chair newman

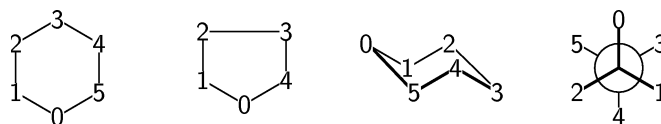
The list above contains all structures defined in the compiler core; with the aid of `→require`, you can build up libraries of externally defined ring structures, which are loadable when needed. Some possibilities to produce five-membered rings in stereographical representation like `chair` are shown in section 2.4 of the tutorial (steroids) and in the command reference under `→restoreXY`.

The position of bonds with the direction types `r`, `t` and `b` of the command `bond` are

shown in the following formulae. **t** bonds are symbolized with thick lines, **b** bonds with dashed lines.

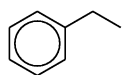


Some of the parameters have a fixed meaning, the meaning of others depends on the ring type. Fixed parameters are `<start>` and `<bliste>`. The first, which can vary between Zero and the number of atoms in the structure minus one, specifies the relation of the structure to the current point. This means, it determines which atom of the ring structure lies at the current point. In the case the ring is the first element of a formula, the parameter does not make sense because the ring becomes the basic element of the formula. But if bonds are already drawn, you specify the target atom of the last bond. The labeling of the structures is as follows:

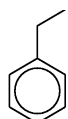


The basic rule for simple *n*-membered rings and most of the other basic structures is: the labels increase clock-wise, the atom labeled with Zero lies on the positive x axis or is clock-wise rotated by a type-specific angle (-90° for type `ring`). The labeling order of some basic structures may differ from this scheme.

Note in the following examples that `ring` does not move the current point. You can use this to draw all three rings without needs for substituting one ring with another, as shown in the last formula (nevertheless, it is recommended to use `vertex` for the rings B and C due to several reasons):



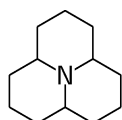
```
formula(C,C){
  bond(150; -150) ring(,4){}
}
```



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



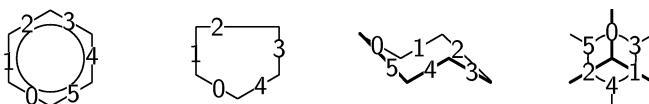
```
formula(C,C){
  bond(30,=C) ring("cpentane",1){}
}
```



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

The labeling serves as indication which substituent starts at which ring atom, too (labels in `<alist>`).

With the *bond list* `<blist>`, you can vary the bond types used while building the structure. It consists of a sequence of pairs of bond number (or label) and bond type. The bond label starts with 0 and ranges to the number of bonds in the structure, decremented by one. The correspondence of bond labels to individual bonds is shown in the following diagram:



The specification for the bond type is the same the command `bond` uses. The drawing of a circle, indicating aromaticity, is not automatically inhibited by bond type specifications, you have to specify explicitly whether this aromaticity ring should be present or not. For this purpose, the additional types `H` (meaning saturated systems without circle) and `O` (aromatic systems with circle) are available. These two specifications have to appear at the beginning of the bond list! Some examples may illustrate the use of bond lists:



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



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



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



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

The parameter `<len>` codes the size of the ring. The default `N` is the radius of the building circle for polygons and the length of the main bond for all other systems. All or at least the highest possible number of bonds are of this length, if geometrically possible.

Due to the fact that the *radius* is the base length of polygons, the length of the real bonds between the atoms varies as function of the number of vertices (bonds of triangles are significantly longer than bonds of nine-membered rings). To set a constant *bond length* you have to specify the size with `#`, see below.

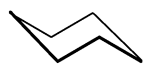
An explicit specification of the ring size is appropriate especially in conjunction with complex systems, e. g. cyclohexane:



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



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

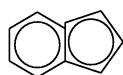


```
formula(C,C){
  ring("chair",,,L){}
}
```

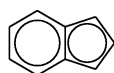


```
formula(C,C){
  ring("chair",,,NN){}
}
```

The variant with `#` before the bond length specification is allowed only for the structure type `ring` and means that `<len>` does not specify the radius of the circle, but the length of the real bonds themselves. Polygons with different numbers of atoms then have equal bond lengths. This must be considered if you condense two different rings as shown:



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



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

Usually, the bonds of the six-membered ring are shorter by a small amount than those of a five-membered ring. This becomes clearly visible when condensing both rings (more extreme when using three-membered rings). Therefore, each of the participating rings' sizes have to be specified with `#N`.

Variable parameters

The parameters <p1> and <p2> have different meanings depending on the basic structure type:

ring : For this structure, they code the number of vertices (or atoms) of the ring (default is 6 for benzene) and an angle in degree, by which the ring is rotated. The reference is the positive x axis, the rotation is done clock-wise. Due to the default of -90° for the angle, the benzene ring stands upright on the atom labeled 0. The most important alicyclic basic compounds can be typeset as follows:



```
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"); }
}
```

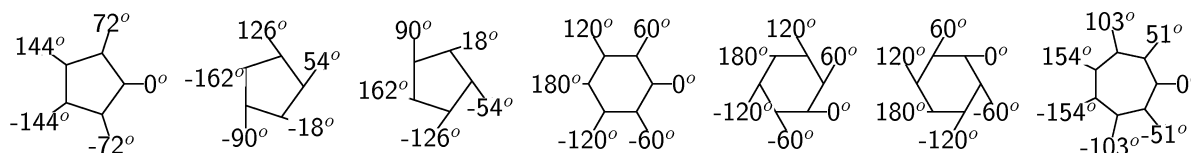


```
formula(C,C){
  ring(,H,,6,-90){ 0: atom("O"); }
}
```



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

Some important angles are given for the most basic polygonal structures in the following formulae. They code the angle of a line parallel to a polygon side in the indicated direction. If the polygons are rotated, the rotation angle has to be added to the angles given here.



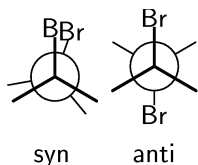
cpentane : Parameter <p1> has no meaning, <p2> is the angle by which the whole compound is rotated.

chair : <p2> is the angle by which the structure is rotated. <p1> codes with values 1, 0 und -1 the two chair- and the twist conformation of cyclohexane:



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

newman : This structure corresponds to an axial view on a single bond (representation in the NEWMAN projection). Bonds starting from the nearer atom have fixed angles of 90° , 210° und 330° . The bonds of the farer atom are rotated by an additional angle p1:



```

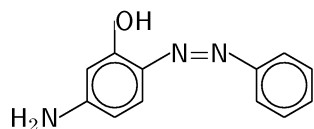
formula(L,R,"syn",HA,36){
  ring("newman",,,L,-20){
    0: atom("Br");
    3: atom("Br");
  }
}

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

```

Substitution

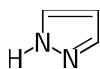
The individual `<alisti>` components in the command's body specify the substitution at the *i*.th ring atom. The labeling of the ring atoms has been shown earlier (see parameter `<start>`). Substituents can consist of single bonds, further ring structures or combinations of all formula elements. Some examples:



```

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("OH",L);
  }
}

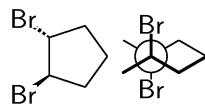
```



```

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");
  }
}

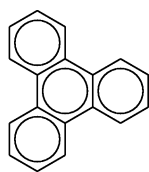
formula(L,R){
  bond(150)
  ring("newman",3){
    0: atom("Br");
    1: bond(30,t);
    4: atom("Br");
  }
}

```

 The vertex command

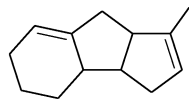
Ring structures supporting rotation of the structure offer an extended syntax with `vertex` commands. These commands condense further structures to the current structure. `<vertex_1>` and `<vertex_2>` code the concerned edges of the basic ring (the current ring) and the second ring in the form of numbers. These numbers are identical to those used for bond list specification, see above. `<blist_1>` is the bond list for the new structure. `<p1>` codes the size of the new structure.

The body of the `vertex` command is identical to the body of `ring`, meaning that substitution is possible, using the labeling scheme of the incoming structure. Condensation of further structures with more `vertex` commands is also possible. Some examples may show you proper usage.

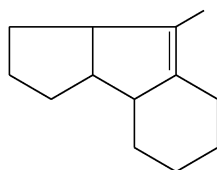


triphenylene

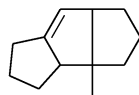
```
formula(L,R,"triphenylene",HR,24){
  ring(){
    vertex(,0){};
    vertex(,2){};
    vertex(,4){};
  }
}
```



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



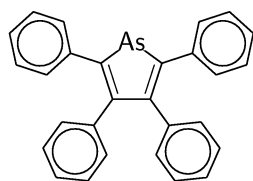
```
formula(L,R){
  ring("cpentane",,,L){
    vertex("cpentane",3,1,H3=){
      3: bond(r);
      vertex(,4,0,H){};
    };
  }
}
```



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

Symbolic rotation angles

The ring structures `ring` and `cpentane` support the symbolic angles `r`, `t` and `b` for the rotation angle `<p2>` instead of integer numbers. Similar to the related specifications of the `bond` command, these symbols code the angles of the radial and tangential bonds. The ring's rotation is calculated so that the last bond drawn, leading to the specified start atom, possesses the specified relation to the ring:



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){}; }
}
```

More examples of symbolic angles can be found in the tutorial.

3.1.18 save

`save(#<n>)`

This command saves the graphical parameter used for typesetting formulae under the number <n>. You use this command if you want to locally change a parameter's value and you want to reset it to the original value (\rightarrow `restore`). Examples of this are given in the tutorial, section 2.5.

3.1.19 savecontext

`savecontext(#<n>)`

This command saves the context of the last drawn formula under a number <n>. You can later reinstall this context (\rightarrow `setcontext`) for building up branches in a reaction chain or to reuse a formula's context several times.

3.1.20 saveXY

```

saveXY(<#<n>)
saveXY(<#<n>, <iAngle>, <Len>)
saveXY(<#<n>, =<2dexpr>)

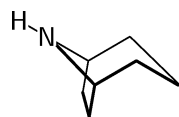
2dexpr := 2dexpr' + 2dexpr' | 2dexpr' - 2dexpr' | 2dexpr'
2dexpr' := realnum * 2dexpr | (2dexpr) | #<n> | #cur | {<x>,<y>} |
           philen(<phi>, <len>)

```

This command saves in the syntax variant above the current coordinates under a number <n>. The extended syntax below saves the coordinates of a point, shifted by the length <Len> into the direction of the angle <iAngle> from the current point. Both parameters are identical to those of the `bond` command. The current point is *not* shifted. Corresponding to the `bond` command, there is the opportunity to denote with the = syntax a complex formula in order to calculate a storable position (for an example, see the command `bond` on page 93).

Simple syntax

You use a saved point to draw later with `bond(<#<n>)` a bond from other atoms to it regardless of the exact bond length and angle. In this way, you can easily build up complex and irregular polycyclic compounds, typical for natural compounds. tropine and Nor-Pinane are available in the library `bicyclib.pm`. Here a demonstration is given how you can implement a structure not yet available in the library:



Desoxy-Tropin

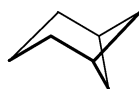
```

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

```



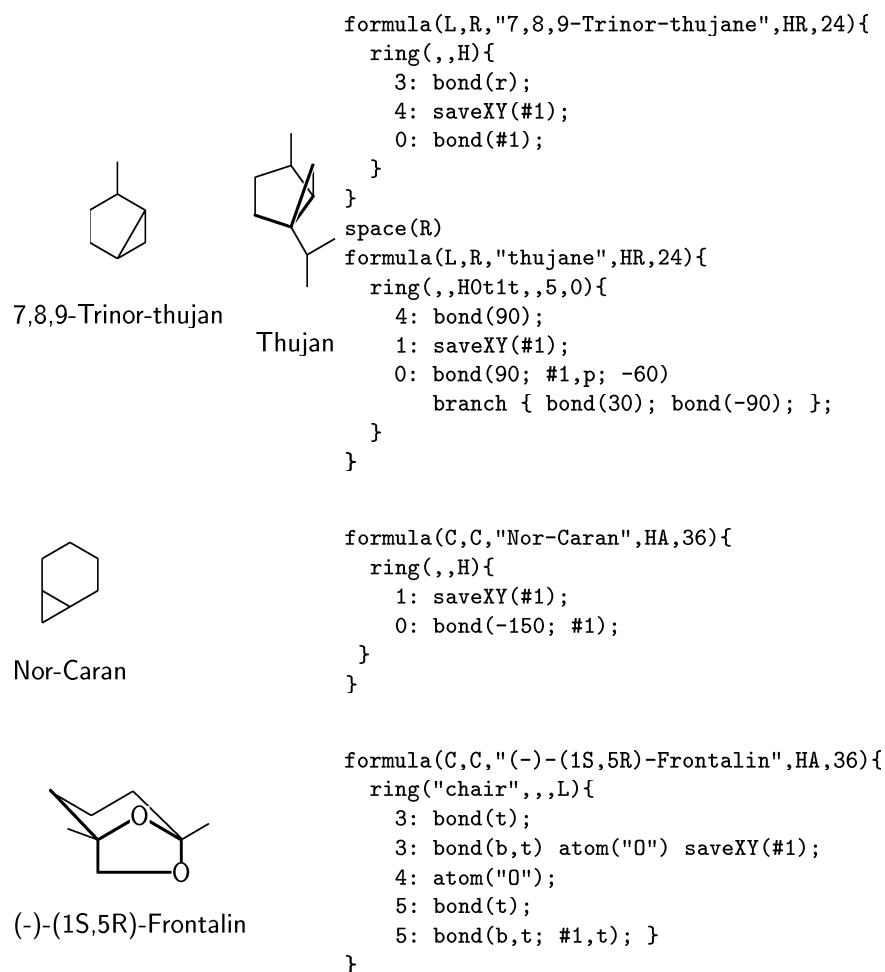
Nor-Pinan



```

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);
  }
}

```



In a planar representation, the pinane and carane structure behave like regular six-membered rings, so you can describe them simply as follows:

```

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

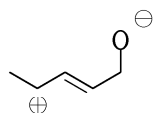
  formula(L,R,"Nor-Caran",HA,36)
  { ring(,H){ 0: bond(-150; 90); } }
\end{chemistry}

```

In conjunction with `→restoreXY`, this syntax greatly expands the possibilities of typesetting irregular natural compounds; see the description of `restoreXY`. Note that the scope of the saved points is not limited to the formula, so that a later reference to an already saved position points erroneously to the wrong formula! You can exploit this behaviour to illustrate the electron transfer between two different formulae, as shown in section ??.

Extended syntax

This syntax variant is especially useful for the illustration of electron movements, examples can be found at `→emove` and in the tutorial, section 2.6. Another application is to mark points nearby the formula's main line to typeset labels or symbols. Examples would be charge symbols beside an alkane chain:



```
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$")  
  restoreXY(#2) atom("\ominus$")  
}
```

3.1.21 scale

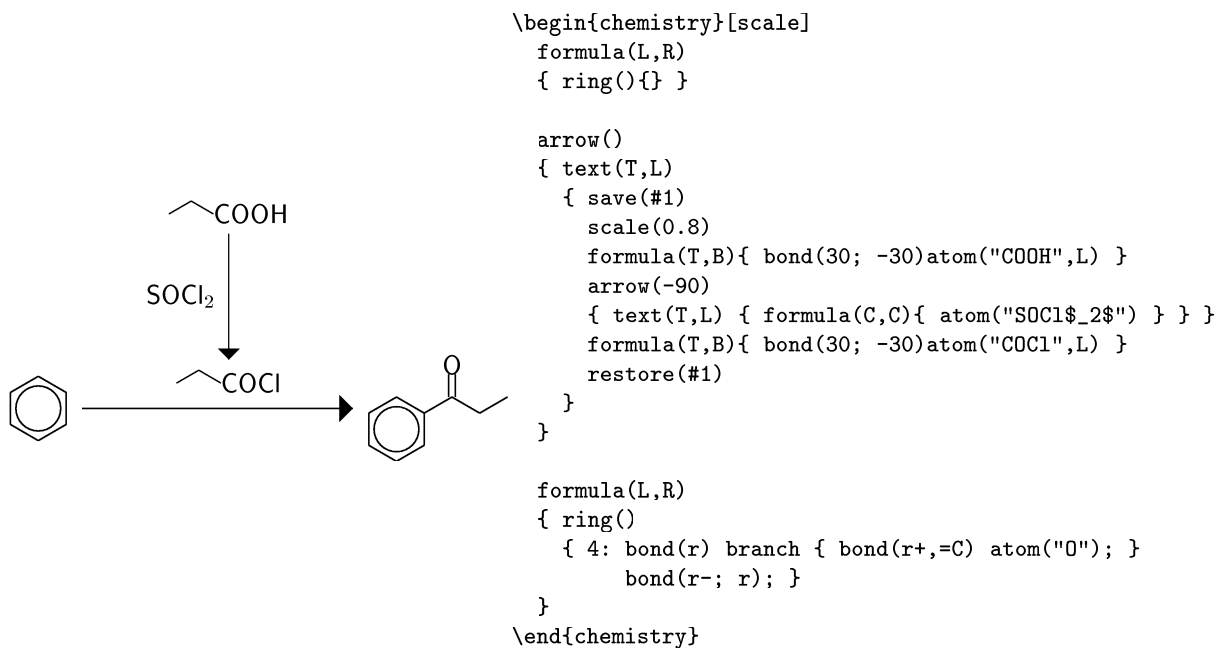
`scale(<p>)`

The command scales all internal graphical parameter by a factor <p>. <p> is a fraction; to scale a formula to 70%, write

`scale(0.7)`

Fractions greater than 1 result in a magnification, fractions less than 1 in a smaller drawing.

You use the command habitually within a `→save/restore` pair to recover easily the original values. As an example, a helper reaction will be shown above a reaction arrow:



3.1.22 set

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

With this command, you can adapt one of the parameters in table 3.1 which determine the layout of the formula, and set it to the value <iVal>. Its values influence partly the settings like the distance of a formula from another, partly the appearance like line width and arrow thickness. Changes of parameters of the latter kind will be written into the output file.

Changes made with this command are permanent, so it may be necessary to adapt them locally. For this, the parameter set is stored under a certain number right before the section of the program in which they will be changed. After the end of the section they will be reactivated with the help of this number (commands `→save` and `→restore`):

```
...
save(#1)
set("rLW", 2) % increase line width
formula...
...
restore(#1) % restore old values
... % normal line width
```

A delineation of an application is found in the tutorial, section 2.5.

Table 3.1 Internal parameters which can be changed with `set`.

Parameter	Meaning
rLW	Width of all lines in bonds, circles, arrows and others.
rBW	Maximum width of projective bonds.
rAW	Width of the arrow tip of reaction arrows.
rBD	Dash pattern for projective bonds.
rXS	Horizontal distance of formulae and arrows.
rXY	Vertical distance of formulae and arrows.
rArrowExtend	Additional distance a reaction arrow stands out on both sides in relation to a label.
rArrowSkip	Distance of the arrow label from the arrow itself.
rChainSep	Distance of the reaction chains joined together by the commands <code>joinh</code> and <code>joinv</code> .
rMultilineSep	Distance of the reaction chains stacked on top of each other by the command <code>multiline</code> .
fboxsep	Distance of the frame/bracket from the formula.
rTextSep	Vertical distance of the formula label from the bottom edge of the lowest formula.
iAngleR	Angle for radial bonds, symbol <code>r</code> .
iAngleT	Angle for "top" bonds, symbol <code>t</code> .
iAngleB	Angle for "bottom" bonds, symbol <code>b</code> .

3.1.23 setcontext

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

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

This command sets the already under a number <n> stored context as the current context and calculates a continuation point out of the context's bounding box and the positioning parameter <pos>. An important application for the command is the construction of branches in reaction chains: in scheme 3–4 a reaction scheme is noted, starting from azlactone, in the "usual" sequence (L,R) – (L,R) – (L,TR). In order to achieve a second branch from the azlactone, its context must be stored (command `→savecontext`) after the typesetting of the lactone formula. This context must be reactivated after the end of the first chain. The second chain should run down to the right, therefore BR as positioning parameter will be denoted:

```
\begin{chemistry}[cont1]
multiline(1){
  formula(L,R, "hippuric acid"){
    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(,,H0=,,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",,0=){
    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,"$\alpha$-Aminoacid", HR, 24){
  atom("H$_2$",C,R) atom("N",L,C) bond(90)
```



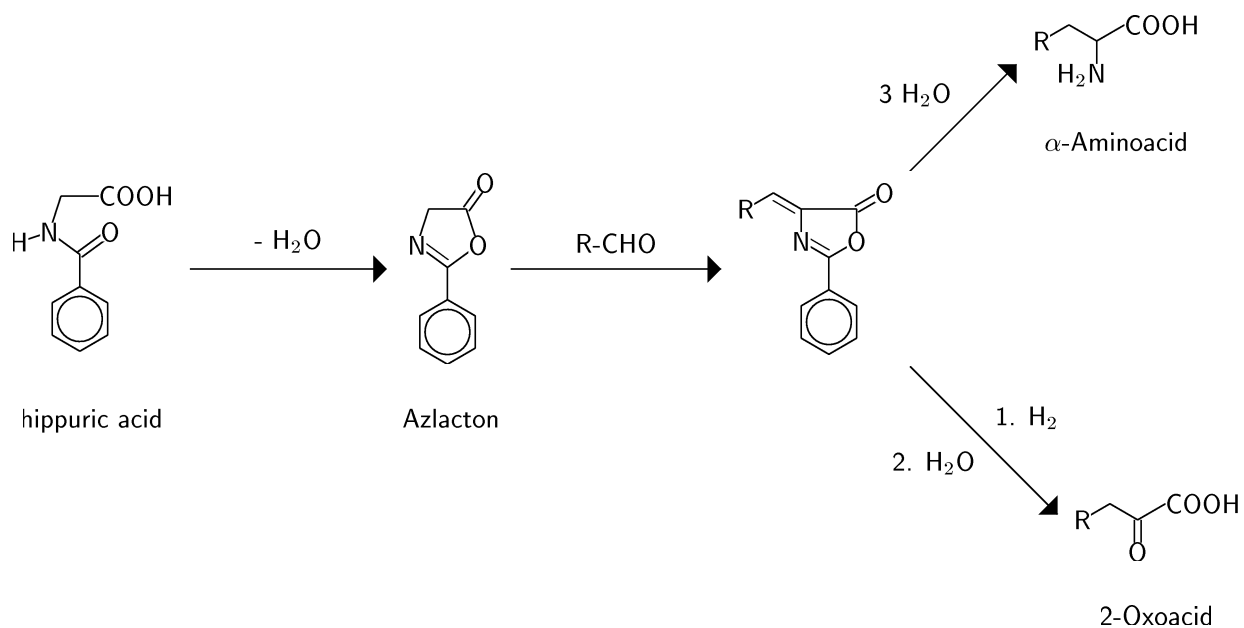
```

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

```

You can construct different branches by saving a number of contexts and applying several `setcontext` commands. Furthermore, by this method several chains can branch from one formula, which is explained in detail in the tutorial, section 2.3.4.



Scheme 3-4 Several reaction chains branch from one formula. The context of this formula must be stored.

3.1.24 shiftXY

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

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

This command shifts the current point by <dx> on the x axis and by <dy> on the y axis outside a formula. If a value is missing, no shift takes place at the corresponding axis.

The main usage of the command is to place several educts not in a regular kind of “string”, but in a “cloud”. In the opposite to `→gotoXY`, no fixed points are given but relative distances. Shifts of the overall reference point affect all educts equally. It may be useful (but it is not obligatory) to use the placement parameter `C,C`. An example is given in section 2.6. There, three educts are to be placed so that you can illustrate the electron movements between them. The collection of all “wild” distributed educts to one unit proves successful once again with a single-line `multiline` command.

3.1.25 space

```
space(<pos>)
```

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

This command inserts additional space into the direction specified by <pos>. In choosing a direction contrary to the main one, you get a “backspace”, which means that space is eliminated. (After an `arrow` or `formula` command, this elimination of additional space is possible with the postfix command `nospace`.)

3.2 Special commands

This section introduces commands which must not at all appear within a `chemistry` environment in \LaTeX documents. Some of them are part of the actual source files which are produced during a \LaTeX run or which may be coded by you manually. According to the following scheme, \LaTeX commands correspond to them so that it is not necessary to use the basic commands:

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

However, there are some special commands which are required during the initialization of OCHEM, but not within each single chemistry environment. Those may be included in the `chemspecial` environment. This environment passes the enclosed text without changes to the actual source file and is usually placed at the beginning of the document. Among the commands which can be usefully applied in a \LaTeX document `require` and `package` should be mentioned.

If you look over the `.chm` file produced by a \LaTeX run, you can see how your document has been processed into the source file.

3.2.1 font

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

This command contains the \LaTeX commands which adjust the font for the chemical text symbols of the `atom` command. This form of the command only serves for internal purposes, because the command `\chemfont` has been provided for the \LaTeX user (see page 8 for details).

3.2.2 package

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

This command loads the required package with possible options during the \LaTeX processing of the formulae' labels. It may appear more than once, but must always appear before the first formula.

The calculation of the size taken by a text, i. e. an atom symbol, is transmitted to the \LaTeX compiler. The compiler starts with a temporary file which contains all texts used in the reaction schemes. If macros are used in these texts, they must be available for the \LaTeX processing of the temporary file. OCHEM therefore applies the following method: all macros which are in use within text positions in formulae must be collected in one or more separate packages, i. e. `biochem.sty`. The names of these packages are loaded with the `package` command by the temporary file and become then available. Package options can also be communicated:

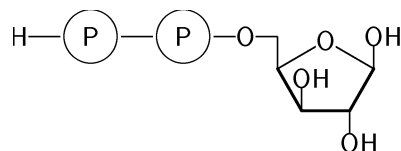
```
\begin{chemspecial}
package("biochem")
package("[biochem]chemmacros")
\end{chemspecial}
```

An illustration for this would be the letter "P" in a circle which is often used in biochemistry for activated phosphate. You can produce the symbol with the help of a \LaTeX macro. The macro definition is written in a file called `biochem.sty`:

```
% biochem.sty

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

The macro is then used in formulae as follows:



Xylose-4-diphosphate

```

\usepackage{biochem}

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

...

\begin{chemistry}
formula(C,C,"Xylose-4-diphosphate",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("\Phospat",R,L) bond(180)
      atom("\Phospat",R,L) bond(180)
      atom("H",R);
  }
}
\end{chemistry}

```

The example demonstrates how the package which contains your own macros must be included into the \LaTeX document. This must arrange the text in the final positioning calculated by the chemistry compiler. The macro definitions must therefore be accessible for \LaTeX as well as OCHEM.

3.2.3 require

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

This command loads the perl module `<File>.pm`. It must appear before the first `schema` command, and can be used multiple times. Its primary purpose is not to load “real” perl modules (which would be also possible), but to make available self-defined ring structures, implemented in perl code. In this way, you can keep a variety of rarely needed structures available without introducing them directly into the compiler core and without overloading it. It is also possible to develop structures without automatically producing a new version of the compiler. The compiler is kept compatible and can be replaced all the time by newer versions without losing your self-defined extensions. Furthermore, the module with your definitions can be forwarded to others without a compiler replacement.

The development of structures is depicted in the tutorial, section 2.10. Some structures are already defined in the library `bicyclib.pm`, for example, which is included in the OCHEM distribution. If the module should be loaded to create the structure of tropinee (section 3.4), then you have to code:

```

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

\begin{chemistry}

```

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

3.2.4 schema

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

This command encloses a complete formula or reaction scheme. `<slite>` is compiled as a unit and saved in the selected output format under the file name `<name>`. This environmental command contains the logic basic units of the chemistry package and corresponds to the `chemistry` environment in \LaTeX .

3.3 The library `mncyclib.pm`

This section explains the ring structures contained in the library file `mncyclib.pm`. You load this library into the preamble of a \LaTeX document as follows

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

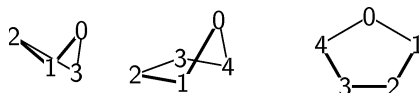
The explanations follow the schema already used in the `→ring` section. There you will find further explanations.

The library contains the following monocyclic structures and their coding `<type>`:



cb cp furanose

With the basic structures of this library, you can draw monocyclic compounds in a pseudo three-dimensional manner. The numbering scheme is shown in the following formulae:



The bonds are numbered according to the following formulae:



The meaning of the variable parameters `<p1>` and `<p2>` is as follows:

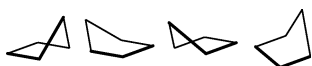
`cb` : (cyclobutane) parameter `<p1>` is unused, `<p2>` is the rotation angle of the complete structure.

The positions of bonds with the direction types `r`, `t` and `b` are shown in the following formulae. `t`-bonds are symbolized by thick lines, `b`-bonds by dashed lines.



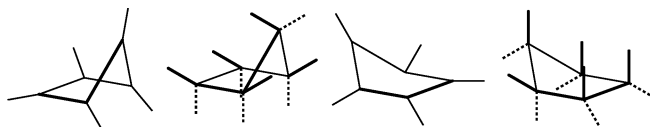
radial top/bottom

`cp` : (cyclopentane) again, parameter `<p2>` is the rotation angle of the complete structure. `<p1>` allows for the selection of different conformers:



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

The positions of bonds with the direction types **r**, **t** and **b** are shown in the following formulae. **t**-bonds are symbolized by thick lines, **b**-bonds by dashed lines.

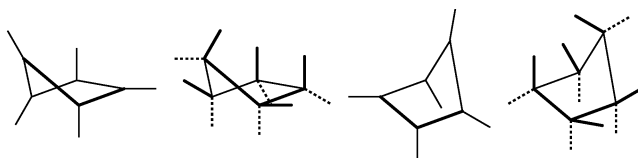


radial

top/bottom

radial

top/bottom



radial

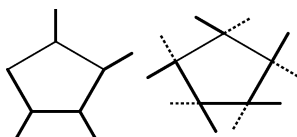
top/bottom

radial

top/bottom

furanose : (furanose structure) The parameter here remain unused.

The arrangement of bonds with the direction types **r**, **t** and **b** is delineated by the following schemes. **t** bonds are drawn by thick lines, **b** bonds by dashed lines:



radial

top/bottom

3.4 The library bicyclib.pm

This section explains the ring structures contained in the library file `bicyclib.pm`. You load this library into the preamble of a \LaTeX document as follows

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

The explanations follows the schema already used in the `→ring` section. There you will find further explanations.

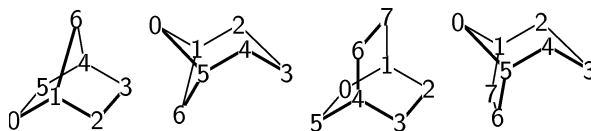
The library contains the following bicyclic structures and their coding `<type>`:



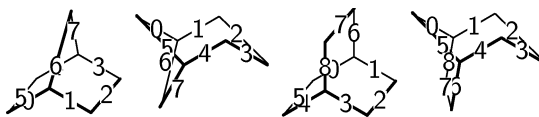
bc221h bc311h bc222o bc321o

The type code is an abbreviation for **bicyclo[n.m.o]heptane** and **bicyclo[n.m.o]octane**. You get the structures of tropine, norpinane and norbornane with these types.

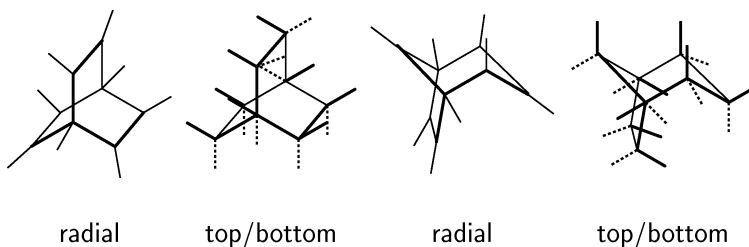
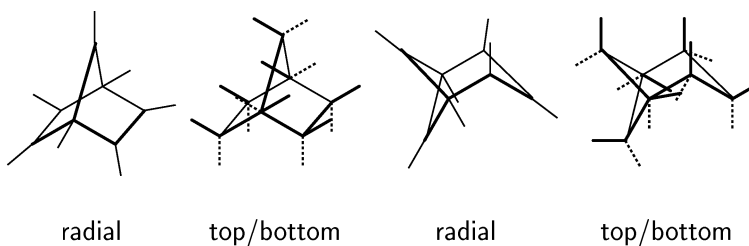
The numbering scheme is shown in the following formulae:



The bonds are numbered according to the following schemes:



The positions of bonds with the direction types **r**, **τ** and **b** are shown in the following formulae. **τ** bonds are symbolized by thick lines, **b** bonds by dashed lines.



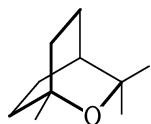
The meaning of the variable parameters <p1> and <p2> is the following:

bc221h: (norbornane) Both parameters are unused. (According to the new IUPAC rules, this structure has to be named 8,9,10-trinorbornane.)

bc311h: (norpinane) Both parameters are unused.

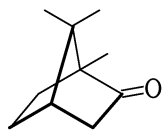
bc321o: (tropinee) Both parameters are unused.

As examples, some bicyclic natural compounds are built with the library's structures.



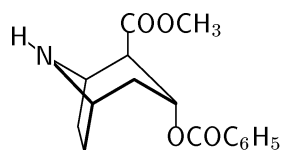
Cineol

```
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,"cocaine",HA,48){
  ring("bc321o",,,L){
    2: bond(t) atom("C",C,R) atom("OOC$3",L);
    3: bond(b) atom("O",C,R) atom("COC$6$H$5",L);
    0: atom("N") bond(r) atom("H");
  }
}
```

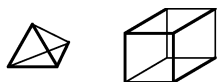
3.5 The library polycyclib.pm

This section explains the polycyclic ring structures provided by the library module `polycyclib.pm`. It is loaded into the preamble of a \LaTeX document by the subsequent lines:

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

The descriptions follow the scheme which has been applied to the command `\rightarrowring`. More detailed explanations can be found there.

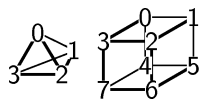
The library contains the following polycyclic structures and their coding `<type>`:



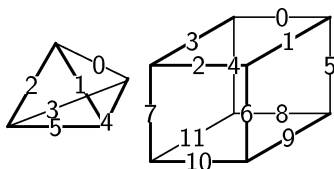
tetrahedrane cubane

Structures like tetrahedrane and cubane can be depicted.

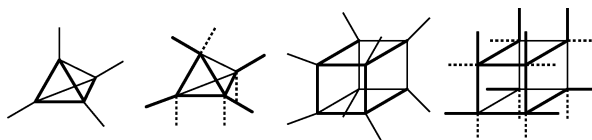
The numbering of atoms is illustrated by the following formulae:



The bonds are numbered according to the schemes given:



The arrangement of bonds with the direction types `r`, `t` and `b` is displayed by the following formulae. `t` bonds are indicated by thick lines, `b` bonds by dashed lines.



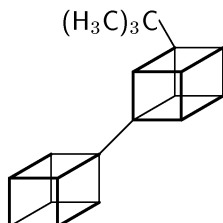
radial top/bottom radial top/bottom

The significance of the two variable parameters `<p1>` and `<p2>` is:

tetrahedrane : The parameter `<p1>` is unused, while `<p2>` represents the rotation angle of the whole structure.

cubane : The parameter <p1> is insignificant, whereas <p2> determines the rotation angle for the whole structure.

For the case in point the formula of the 2-*tert*-Butylecubylecubane is stated as an example of usage:



```
formula(L,R){
  ring("cubane",,,L){
    1: bond(r,,L)
      ring("cubane",7,,L,,0){
        0: bond(r) atom("(H$_3$C)$$_3$C",R);
      };
    }
  }
}
```

For a more complex case, the synthesis of cubane is presented in scheme 3–5:

```
multiline(3,L)
{ % upper line
  formula(L,R,"cyclopenten-5-one"){
    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);
    }
  }
  }
  ;

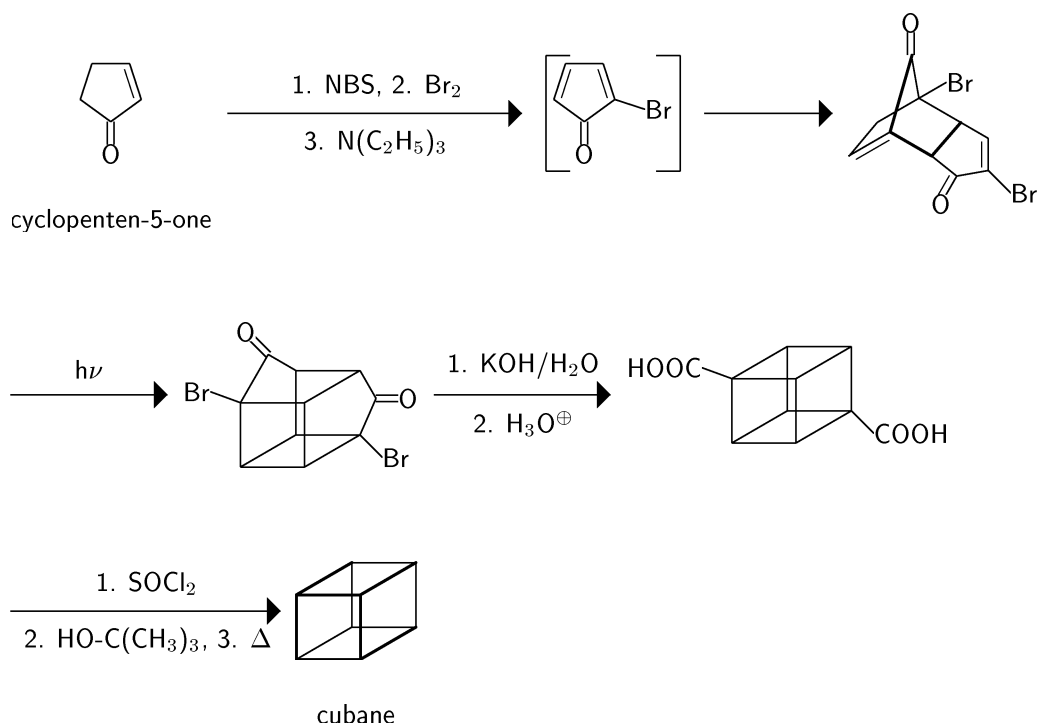
  % middle line
  arrow(){
    text(T,C){ formula(C,C){ atom("h$\nu$") }}
  }
  formula(L,R){
    ring("cubane",,1-2-3s5s6-7-9-10-11-,NN){
      3: saveXY(#1); 0: bond(150) branch { bond(135,=C) atom("O"); } bond(#1);
    }
  }
}
```

```

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$^+$") }}
}
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,"cubane"){
ring("cubane",,,NN){
}
}
}
;
}

```



Scheme 3-5 The synthesis of cubane.

3.6 The include file `natur.inc`

The OCHEM package is shipped with a small include file `natur.inc`, containing a few, but growing number of macros, which simplify the work with some substance classes, especially natural compounds. It is incorporated as follows into your \LaTeX document:

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

You might find the macros not really useful in their present form; but it is difficult to generalize macros without using about 50 parameters for all possible and impossible cases. You are expected to see these macros as a fund from which you derive basic structures and modify them according to your needs. As usual, I am happy to hear from you in such cases and get a copy of such a better-written include file :-)

The macros are now described in detail. They all draw basic structures for certain classes of substances. Important key positions are saved, so you can reuse them with `restoreXY`. The coordinate pair's numbers are given for each macro. Some macros expect parameters, which also modify the basic structure, e. g. `androstane` or `cholane` for the steroid macro to accomplish the androstane and cholane structure. Further modifications can be made with bond lists, describing basic ring elements in the structure. At last, complete descriptions of side chains can be employed as arguments. The examples show all these possibilities.

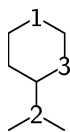
Please note that these macros are not totally general, you will always find structures that you cannot typeset with them. In these cases, you can copy the macros and modify it or, in really difficult cases, describe the complete resistant structure without any macro!

TERPEN

This macro creates a menthane structure, which serves as basis for some monocyclic monoterpenes. The general syntax is

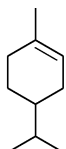
```
TERPEN_(<blist>, <C2-Subs>)
```

`<blist>` is the bond list for the ring, `<C2-Subs>` is the description of a substituent at C^2 . If there is no such substituent, you can specify an empty description with `'`. The positions of C^1 , C^8 and C^3 are saved in the coordinate pairs 1 to 3. The following formula shows the basic structure and the position of the saved atoms:



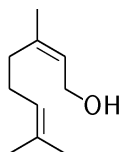
TERPEN_

Some examples show the handling of the bond list, the substituent's description and the saved points:



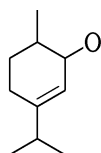
1-p-Menthen

```
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

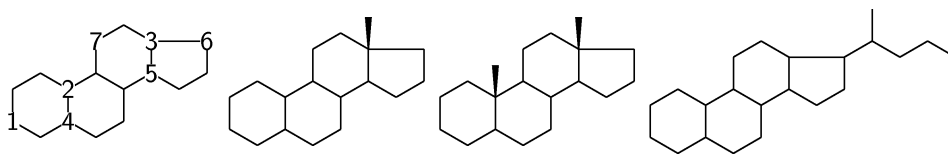
```
formula(L,R,"Carvenon",HR,24){
  TERPEN_(H5=,bond(30) atom("O"))
  restoreXY(#1) bond(90)
}
```

STEROID

This macro has the general syntax

```
STEROID_(<blist>, <type>)
```

and generates steroid structures according to <type>. The possible types correspond to the most important basic hydrocarbons and are shown in the following overview. The numbers at some atom positions match the number of the saved coordinate pairs:

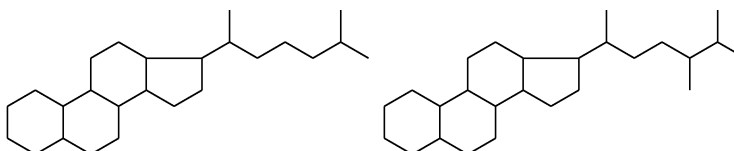


STEROID_

estrane

androstane

cholane



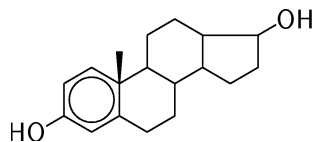
cholestane

ergostane

<blist> is a bond list, describing ring A. You can use it to specify aromaticity or double bonds for this ring. Further modifications can be done by using the subsequent coordinates, represented in the overview below by numbers:

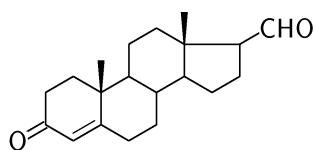
number	atom	number	atom
1	C ³	2	C ¹⁰
3	C ¹³	4	C ⁵
5	C ¹⁴	6	C ¹⁷
7	C ¹¹		

Some examples may show you the macro's purpose:



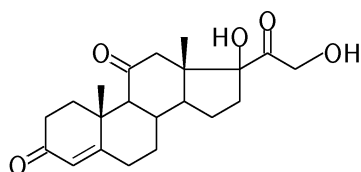
"Ostradiol

```
formula(L,R,"{}""Ostradiol",HR,24){
  STEROID_(0)
  restoreXY(#2) bond(90,<<)
  restoreXY(#1) bond(-150) atom("O",C,L) atom("H",R)
  restoreXY(#6) bond(30) atom("O",C,R) atom("H",L)
}
```



Testosteron

```
formula(L,R,"Testosteron",HR,24){
  STEROID_(H5=, androstane)
  restoreXY(#1) bond(-150,=C) atom("O")
  restoreXY(#6) bond(30) atom("C",C,R) atom("HO",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

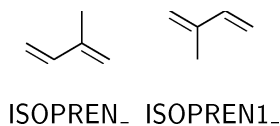
This macro draws an isoprene unit in two different forms, allowing the construction of polyprene chains. The general syntax is

```

ISOPREN_(<bond_len>)
ISOPREN1_(<bond_len>, <bond_type>)

```

<bond_len> is the length of a bond, leading to the start point of the unit. For an isolated unit, you must specify a length of '0'. For subsequent units in polyprene chains, this parameter may remain empty with '. With <bond_type>, you specify the bond type for one of the double bonds (usage at the terminal ψ -unit of the Caroten). The following formulae show the basic unit with <bond_len> equal to zero:



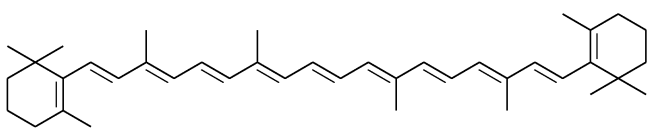
ISOPREN_ ISOPREN1_

```

formula(L,R,"I{}SOPREN\_",HR,24){
  ISOPREN_(0)
}
formula(L,R,"I{}SOPREN1\_",HR,24){
  ISOPREN1_(0)
}

```

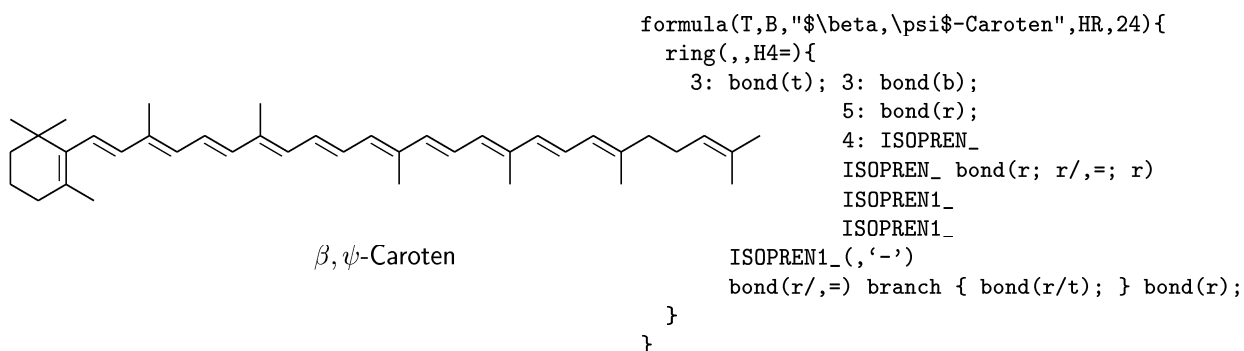
Some examples show the construction of carotene chains:

 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);
    };
  };
}

```



AA

This macro simplifies the representation of amino acids in a text-like mode without considering the three-dimensional structure. The general syntax

`AA_(<N-terminus>, <C-terminus>, <side chain>)`

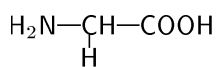
offers two parameters, describing the N- and C-terminus respectively. The arguments H or OH generate the terminal sequences H₂N- and -COOH. These positions may remain empty and allow therefore the construction of oligopeptides. You can also specify any formula elements to depict abnormally terminated amino acids. The third parameter describes the amino acid's side chain.

`<N-terminus>NH—CH—CO<C-terminus>`
 |
 <side chain>

AA_

The following examples show the usage of the macro. For convenience, the include file `utils.inc`, described in section 3.7, is utilized.

The basic form of an amino acid is generated by the termini H und OH:



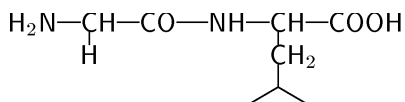
Glycin

```

formula(L,R,"Glycin",HR,24){
  AA_(H,OH,atom("H"))
}

```

In a simple dipeptide, neither is the first amino acid allowed to have a C-terminus, nor the second a N-terminus:



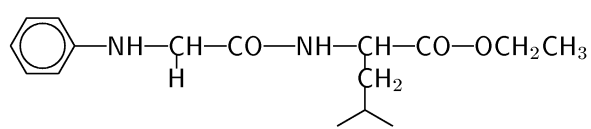
H-Gly-Val-OH

```

formula(L,R,"H-Gly-Val-OH",HR,24){
  AA_(H,atom("H"))
  bond(O)
  AA_(,OH,branch{atom("C",C,R) atom("H$_2$"},L);} iPr_(-90)
}

```

To depict an abnormal saturation of the terminal atoms, you can also specify totally different formula elements:



```

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 The include file `utils.inc`

Another include file is `utils.inc` which contains very small macros. They are intended to support the coding of recursively used structures. They are added to a \LaTeX document as follows:

```
\begin{chemspecial}
include('utils.inc')
\end{chemspecial}
```

The file contains the following macros:

```
H_           % creates H
N_           % creates N
O_           % creates O
S_           % creates S
Cl_          % creates Cl

Me_          % creates -CH3
eM_          % dto H3C-
Et_          % creates -CH2CH3
tE_          % dto H3CH2C-
iPr_(<dir>)  % dto -CH(CH3)2

OH_          % creates -OH
HO_          % dto H0-
ONa_         % creates -ONa
NaO_         % dto NaO-
Om_          % creates -O$^{\ominus}$
mO_          % dto $^{\ominus}$O-
OR_(<R>)     % creates -O<R>
RO_(<R>)     % dto <R>O-
NH2_        % creates -NH2
H2N_        % dto H2N-
NH2p_       % erzeugt -N(+)H2
pH2N_       % dto H2N(+)-
NMe2_       % creates -N(CH3)2
Me2N_       % dto (H3C)2N-
NMe2p_      % creates -N(+) (CH3)2
pMe2N_      % dto (H3C)2N(+)-
OMe_        % dto -OCH3
MeO_        % dto H3CO-
OEt_        % dto -OCH2CH3
EtO_        % dto H3CH2CO-
CH2OH_      % dto -CH2OH
HOCH2_      % dto H2HOC-
NO2_        % creates -NO2
O2N_        % dto O2N-

CHO_        % dto -CHO
OHC_        % dto OHC-
```

```

COOH_          % dto -COOH
HOOC_          % dto HOOC-
COONa_         % dto -COONa
NaOOC_         % dto NaOOC-
COOm_         % dto -COO$^\ominus$
mOOC_         % dto $^\ominus$OOC-
COOMe_        % dto -COOCH3
MeOOC_        % dto H3COOC-
COOEt_        % dto -COOCH2CH3
EtOOC_        % dto H3CH2COOC-

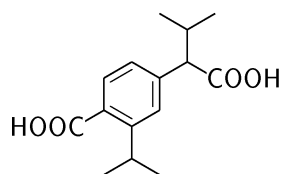
SO3H_         % dto -SO3H
HO3S_         % dto HO3S-
SO3m_         % dto -SO3(-)
mO3S_         % dto (-)O3S-

Ph_{...}      % phenyl
Ph_(<angle>){...} % phenyl

```

The body of `Ph_` matches the body of a normal `ring` command and may contain the same legal operations or remain empty. `iPr_` creates a branching isopropyle rest whose direction must be determined by the argument. It consists of an angle.

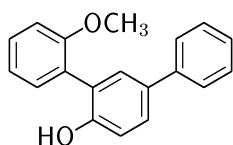
Some examples illustrate the possible applications. The macro which generates the phenyl ring possesses an optional parameter for denoting the rotation angle of the ring. It is employed as the basic element of the structure for the depiction of benzene. The rotation angle in the examples is -90° . The default angle \mathbf{r} for this parameter leads to the delineation of phenyl substituents:



```

formula(L,R) {
  Ph_(-90)
  { 0: iPr_(r);
    1: bond(r) HOOC_;
    4: bond(r) iPr_(rt) bond(r/) COOH_; }
}

```



```

formula(L,R) {
  Ph_(-90)
  { 1: bond(r) HO_;
    4: bond(r) Ph_{};
    2: bond(r) Ph_{ 5: bond(r) OMe_; };
  }
}

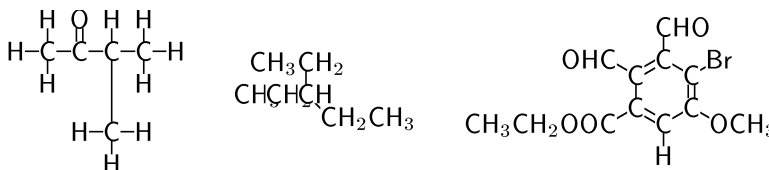
```

3.8 Include files for K_EK_UL_E structures: condensed.inc and cyclohexanes.inc

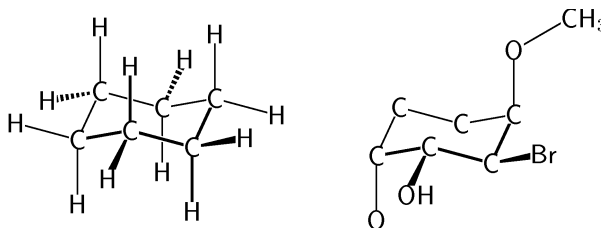
These include files have been graciously provided by Dr. Kenvard Vaughan. They can be found in the subdirectory kv. The files are enclosed in the same way as all other include files are and contain macros which significantly simplify the setting of structure formulae according to K_EK_UL_E. Corresponding to the two include files, there are the two comprehensive documentations condensed_doc.ps and cyclohexanes_doc.ps. Some examples from the documentation:

```
\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(1,,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}
```



```
\begin{chemistry}
defaultz
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}
```



A. The program

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 carbohydrate 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 variety 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.
- 25-MAY-2001 : Extended syntax for 2D expressions in `bond` and `saveXY`.
- 17-SEP-2001 : Installation script created.
- 19-SEP-2001 : Compound catalog created.
- 03-OCT-2001 : English manual finished.

A.2 A lot of thanks goes to ...

my wonderful wife Claudia for her patience with which she has translated the german manual. Not only has she translated it, but only brought into a readable and useful form.

Some users (like Dr. Kenward Vaughan and Oliver Bürschaper) have given me important suggestions (missing bond types, furanose rings, the possibility to apply macros in `atom` commands) and bug reports.

Special thanks again to Dr. Kenward Vaughan for his dedication to the development of those macros for the depiction of formulae according to Kekule which are included in the subdirectory `kv`.

B. More stuff for chemistry addicts

Who intends to typeset formulae must be aware of also naming them, a fact, which can become really complicated. Fortunately, on the web pages of the company ACDLABS, an online version of the IUPAC rules can be found:

<http://www.acdlabs.com/iupac/nomenclature/>

For an agreeable introduction into the precise nomenclature of organic compounds the textbook of D. Hellwinkel, *Die systematische Nomenklatur organischer Verbindungen*, Springer-Verlag Heidelberg, is recommended (in german language).

There is a WYSIWYG product competing with the presented \LaTeX package. It is the chemical drawing program "ChemSketch" by the above mentioned company. It is available as freeware version for DOSes (i. e. Windows computers):

<http://www.acdlabs.com/download>

It can however not only draw formulae (even very attractive ones), but also handle different calculations and construct three-dimensional rod- or sphere models. Furthermore, it possesses a considerable catalogue of substances, features ...

Index

A

AA, 138
active phosphate, 124
amino acids, 138
angle
 r+, 23
 r-, 23
 r/, 24, 25
 r/t, 25
 rt, 25
 symbolic, 22, 90
anomer, 61
anomere
 bond, 89
appearance
 configuration, 6
atom, 85
atropinee, 115, 130

B

bicyclib.pm, 129
bicyclic compounds, 129, 130
bond, 87
 angle, 90
 anomere, 61, 89
 connecting two atoms, 92
 r+, 23
 r-, 23
 r/, 24, 25
 r/t, 25
 rt, 25
Bornan, 130
bracket, 95
branch, 96
branching
 in formulae, 96

C

Cadinen, 27
camphor, 130
carbohydrates, 59
 derivatives, 62

Carminsure, 10
Caroten, 137
catalogue, 14
chain
 reaction, 36
chemistry (environment), 4
Chlorin, 26
Cineol, 130
cocaine, 130
commands
 atom, 85
 bond, 87
 bracket, 95
 branch, 96
 cutline, 97
 emove, 98
 fbox, 99
 font, 124
 formula, 99
 gotoXY, 100
 joinh, 101
 joinv, 102
 multiline, 103
 nospace, 83, 99
 orbital, 104
 package, 124
 require, 125
 restore, 104
 restoreXY, 105
 ring, 106
 save, 114
 savecontext, 114
 saveXY, 115
 scale, 118
 schema, 126
 set, 119
 setcontext, 120
 shiftXY, 122
 space, 122
comment line, 7
configuration
 electronic, 66

of appearance, 6
of compiler, 6
connection bond, 92
context, 43, 120
 super-, 46
crassanine, 31
cubane, 132
 synthesis, 132
cutline, 97
cyclobutane, 127
 spheric representation, 127
cyclohexane, 106
 spherical representation, 106
cyclopentane, 63, 127
 spheric representation, 127
 spherical representation, 63, 127

D

distance, 122

E

electrons
 configuration, 66
 shift, 67
emove, 98
environment
 chemistry, 4
EPS format, 9
examples, 14

F

fbox, 99
Five-membered rings, 60
font, 124
formula, 99
 as a unit, 44
furanose, 60

G

gotoXY, 100
Graphics formats (EPS, JPG, PNG), 9
guajane, 27

H

Hayworth representation, 60, 89
Hirsutan, 27
Humulen, 29

I

Indigo, 10
Internet
 Publication in the, 9
Isoprene, 137

J

joinh, 101
joinv, 102
JPG format, 9

K

Kauran, 34
Kekule representation, 142

L

library
 bicyclib.pm, 129
 development of, 77
 load, 8
 mncyclib.pm, 127
 polycyclib.pm, 131
lycopodine, 33

M

macro, 13
 condition, 13
 isoprene, 137
 steroides, 135
 Terpene, 134
macros
 AA, 138
mncyclib.pm, 127
monocyclic compounds, 127
morphinee, 35
multiline, 103

N

natural compounds, 105, 115, 129, 131
New rings, 125
Norbornan, 130
norpinane, 130
nospace, 83, 99

O

ochem.cfg, 6
orbital, 104

P

package, 124
phenyl substituent, 113
Phosphate
 active, 124
pinane, 130
PNG format, 9
polycyclib.pm, 131
polycyclic compounds, 105, 115, 131
Porphin, 26
preprocessor, 13
Publication
 in the Internet (WWW), 9
purple, 10
pyranose, 62
Pyranosen, 61

R

r+, 23
r-, 23
r/, 24, 25
r/t, 25
reaction
 stack, 103
reaction chain, 36
 branched, 43
representation
 Kekule, 142
 spheric
 cyclobutane, 127
 cyclopentane, 127
 spherical
 cyclopentane, 63
require, 8, 125
restore, 104
restoreXY, 105
ring, 106
rings, 59
 five-membered, 60
 irregular, 105, 115
 user-defined, 77, 125
rt, 25

S

save, 114
savecontext, 114
saveXY, 115
scale, 118
schema, 126
sequence
 of reactions, 36
 branched, 43
set, 119
setcontext, 120
shift
 of electrons, 67
shiftXY, 122
space, 122
spherical representation, 105, 127
 cyclohexane, 106
splitting
 of reaction sequences, 43
steroides, 63, 135
structural formula
 Kekule representation, 142
structures
 user-defined, 77
substituent
 phenyl, 113
sugar, 59
super context, 46
symbolic angles, 22, 90
synthesis
 cubane, 132

T

Terpene, 134
tetrahedrane, 131
thujane, 115
tropine, 130
tropinee, 130
Tutorial, 17

U

user-defined rings, 125

V

valerane, 63

W

Web format, 9
WWW
 Publication in the, 9