ChemFig

v1.0g

A TeX package for drawing molecules

Taxotere
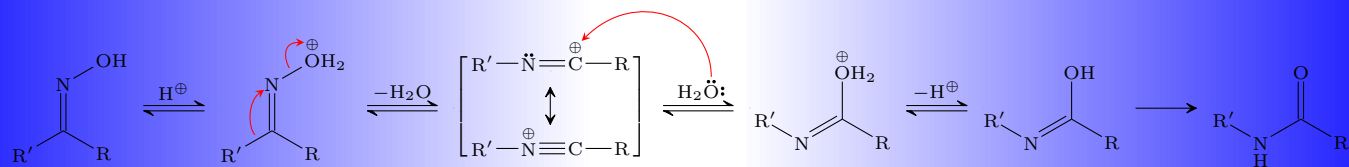
The Beckmann rearrangement

Contents

PART I

Introduction

1   Foreword

This package has seen the light of day thanks to the assistance of Christophe Casseau, who had the idea after being confronted with the complexity of the syntax of the **ppchtex** package.

Throughout the writing of the code, he helped me find interesting features. He always encouraged me to write more advanced features even though I was sometimes (nearly always?) reluctant; if ChemFig has the features it does, it is in large part thanks to him. I thank him as well for his testing of beta versions of this package, and for his contributions to the writing of this manual.

> *Experience shows that is it has been difficult to combine drawing of molecules with the typographic quality of a program like LaTeX, leaving little choice for the user who wishes to have a vector format for these drawings. After having abandoned the* ppchtex *package (developed for conTeXt and available under LaTeX) because of the complexity of its syntax, I turned to the world of programs outside LaTeX. The difficulty in this case is finding a compromise between quality and price. After many unsuccessful attempts I found the only option was a new package, and I would like to thank Christian Tellechea for bringing it to life. To meet my requirements,* ChemFig *needed to be easy to use but still have advanced features, something well nigh impossible. Yet he was able to put together a very flexible TeX code which makes it a pleasure for me to write my molecules. I hope it will be the same for you readers looking for a package useful in the field of chemistry.*

> *Christophe* Casseau

Finally, I wish to warmly thank Theo Hopman and more recently Nikola Castillo for offering to translate this manual into English.

2   Presenting ChemFig

To use this package, start by adding the following code to the preamble:

- `\input chemfig.tex` with εTEX;
- `\usepackage{chemfig}` with LATEX;
- `\usemodule[chemfig]` with ConTEXt.
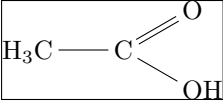
In all cases, the ti*kz* package, if not loaded before, is loaded by ChemFig.

The most important command for drawing molecules is \chemfig{<code>}. The argument code is a set of characters describing the structure of the molecule according to the rules which are described in this manual.

Care has been taken to make it possible to draw the greatest possible number of molecular configurations, while maintaining a simple, flexible, and intuitive syntax. Despite this, the <code> which describes the 2D structure of the molecule increases in complexity in proportion to that of the molecule being drawn.

The command \chemfig draws a molecule using the commands provided by the ti*kz* package, placed inside a `tikzpicture` environment. The choice of ti*kz* implies that:

- the user has a choice of compilation method: pdfLATEX can be used equally well in dvi mode (tex $\longrightarrow$ dvi $\longrightarrow$ ps $\longrightarrow$ pdf) or in pdf mode (tex $\longrightarrow$ pdf). In effect ti*kz*, via the underlying pgf, gives identical graphical results in the two modes;
- the bounding box is automatically calculated by ti*kz* and the user need not worry about any overlap with the text. However, care must be taken with alignment when the molecule is drawn in a paragraph. In the following example, we have drawn the bounding box for the molecule: . ChemFig always places the first atom of the molecule on the baseline of the preceding code.

PART II

Operation of ChemFig

This part is devoted to describing the most common features of ChemFig. The user will find here explanations sufficient to draw most molecules. The presentation of features is done from a theoretical angle, and the goal of this part is not to draw real molecules but to give the user a formal description of the functionality of ChemFig. The "Advanced usage", page 23, will be more practical and will illustrate advanced features for the most demanding uses. It will also highlight methods of building real molecules, page 33. Finally, the last part will give examples of molecules and the code used to draw them.

1   Groups of atoms
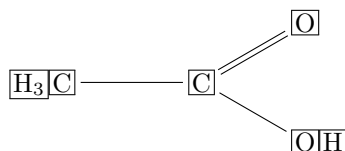
Drawing a molecule consists inherently of connecting groups of atoms with lines. Thus, in the molecule $O = O$, there are two groups of atoms, each consisting of a single atom "O".

However, in this molecule

$$H_3C \longrightarrow C \overset{\displaystyle O}{\underset{\displaystyle OH}{\Big\langle}}$$
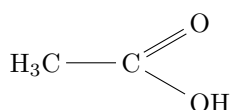
there are four groups of atoms: "H_3C", "C", "O" and "OH". For reasons which we shall see later, ChemFig splits each group into single atoms. Each atom extends up to the next capital letter or one of these special characters: ⊟ ⊟ ~ ⟨ ! ⊛ ⊲ ⊳ @. ChemFig ignores all characters inside braces when splitting groups into atoms.

Therefore the first group of atoms "H_3C" is split into two atoms: $\boxed{H_3}$ and $\boxed{C}$. In terms of chemistry, of course, these are not real atoms; H_3, for example, consists of three hydrogen atoms. In what follows the word atom refers to ChemFig's definition. Thus ChemFig sees the preceding molecule as follows:

$$\boxed{H_3}\boxed{C} \longrightarrow \boxed{C} \overset{\displaystyle \boxed{O}}{\underset{\displaystyle \boxed{O}\boxed{H}}{\Big\langle}}$$

A space is ignored when at the begining of a group of atoms.

2   Different types of bonds

For ChemFig, bonds between two atoms are one of nine types, represented by the characters ⊟, ⊟, ~, ⊳, ⊲, ⊳:, ⊲:, ⊳| and ⊲| :

Bond #	Code	Result	Bond type
1	\chemfig{A-B}	A —— B	Single
2	\chemfig{A=B}	A == B	Double
3	\chemfig{A~B}	A ≡ B	Triple
4	\chemfig{A>B}	A ▶ B	right Cram, plain
5	\chemfig{A<B}	A ◀ B	left Cram, plain
6	\chemfig{A>:B}	A ⅢⅢ· B	right Cram, dashed
7	\chemfig{A<:B}	A ·ⅢⅢ B	left Cram, dashed
8	\chemfig{A>|B}	A ▷ B	right Cram, hollow
9	\chemfig{A<|B}	A ◁ B	left Cram, hollow

The command \setdoublesep{<dim>} adjusts the spacing between the lines in double or triple bonds. This spacing is 2pt by default.

We must understand that when a bond is made between two atoms, these atoms are contained within invisible rectangular boxes. The centres of these two rectangles are separated by an adjustable distance Δ called the "interatomic distance". Furthermore, bonds do not connect to the exact edges of the rectangles: a length δ, also adjustable, separates the edges of the rectangles and the beginning and end of the bond line. The rectangular boxes are made visible in the diagram below to help understanding.
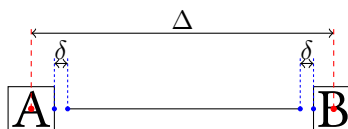
The macro \setatomsep{<dimension>} adjusts the interatomic distance Δ. If the <dimension> is empty, it takes the default value of 3em. This command, like all other settings commands, affects all the following molecules.

Interatomic distance

```
\setatomsep{2em}\chemfig{A-B}\par
\setatomsep{50pt}\chemfig{A-B}
```
A $-$ B

A $=====$ B

The command \setbondoffset{<dimension>} sets the spacing δ between the bond line and the atom. If the <dimension> is empty, δ takes the default value of 2pt.

Trimming bonds

```
\setbondoffset{0pt}\chemfig{A-B}\par
\setbondoffset{5pt}\chemfig{A-B}
```
A$——$B

A $—$ B

If one bond is followed immediately by another, then **ChemFig** inserts an empty group {}. Around this empty group the separation δ is zero:

Empty Groups

```
\chemfig{A-B=-=C}
```
A $—$ B $\equiv\!\!\!==\!\!\!=$ C

The \setbondstyle{<tikz code>} command sets the style for all the bonds drawn thereafter. The <tikz code> is empty by default. To custom a single bond, see page 10.

Style of Bonds

```
\setbondstyle{line width=1pt,red}
\chemfig{A-B=C>|D<E>:F}
```
A $—$ B $=$ C $\rhd$ D $\blacktriangleleft$ E $\cdots$ F

The spacing δ for just one bond can be specified with the character #. This character must be placed *immediately* after the bond symbol and has one required argument between parentheses of the form "#(<dim1>,<dim2>)", where <dim1> is the spacing δ at the beginning of the bond and <dim2> is the that at the end. If <dim2> is omitted, the spacing at the end of the bond takes the value of δ in effect at that time. One can see in the example how the shortening, set to 4pt to be more visible, is nullified for the bond arriving at "B", then for the one leaving "B", and finally for both:

Fine adjustment of bond shortening

```
\setbondoffset{4pt}
\chemfig{A-B-C}\par
\chemfig{A-#(,0pt)B-C}\par
\chemfig{A-B-#(0pt)C}\par
\chemfig{A-#(,0pt)B-#(0pt)C}
```
A $—$ B $—$ C

A $—$B $—$ C

A $—$ B$—$ C

A $—$B$—$ C

By default, all atoms within groups of atoms are typeset in math mode (spaces are ignored). They may therefore contain math mode specific commands such as subscripts or superscripts[1]:

Math mode

```
\chemfig{A_1B^2-C _ 3 ^ 4}
```
$A_1 B^2 — C_3^4$

---

[1]There is a problem with the placement of groups of atoms containing exponents or subscripts. See page 25.

There are settings specifically for Cram bonds. This syntax is used:

$$\setcrambond\{<dim1>\}\{<dim2>\}\{<dim3>\}$$

Any empty argument takes its default value. The three arguments are:

- `<dim1>` is the size of the base of the triangle, and is 1.5pt by default;
- `<dim2>` is the thickness of the dots, and is 1pt by default;
- `<dim3>` is the spacing between the dots, and is 2pt by default.

Here is an example where the three dimensions are changed:

Modified Cram bonds

```
\setcrambond{10pt}{0.4pt}{1pt}
\chemfig{A>B>:C>|D}
```
---

3   Bond angle

Each bond takes an optional argument in brackets. This optional argument can adjust every aspect of a bond, and consists of five optional fields separated by commas. The first of these fields defines the bond angle. Angles increase counterclockwise, and are relative to the horizontal. If the angle field is empty, the angle takes its default value of $0°$. We will see later how to change this default.

There are several ways of specifying the bond angle.

3.1   Predefined angles

When the angle field contains an integer, this represents the angle the bond makes relative to the horizontal, in multiples of $45°$. For example, `[0]` specifies an angle of $0°$, `[1]` is $45°$, and so on up to `[7]` which specifies an angle of $315°$. The integer may lie outside the interval $[0, 7]$, in which case the angle is reduced to the interval $[0, 360)$.

Predefined angles

```
\chemfig{A-B-[1]C-[3]-D-[7]E-[6]F}
```
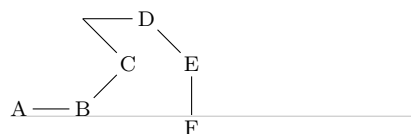
These angles remain valid if the atoms are empty, and this is the case for all the features we will see below:

Predefined angles with empty groups

```
\chemfig{--[1]-[3]--[7]-[6]}
```

3.2   Absolute angles

If one wishes to specify an angle in degrees relative to the horizontal, then the optional angle field must take this form: `[:<absolute angle>]`. If necessary, the `<absolute angle>` is reduced to the interval $[0, 360)$:

Absolute angles

```
\chemfig{A-[:30]B=[:-75]C-[:10]D-[:90]>|[:60]-[:-20]E-[:0]~[:-75]F}
```

3.3 Relative angles

It is often useful to specify a bond angle relative to the preceding bond. This syntax must be then be used: [::<relative angle>]. The sign of the <relative angle> can be omitted if it is a +.

Here is a molecule where the first bond has an absolute angle of $-5°$, and the rest of the bond angles are incremented by $20°$:

Result of relative angles

\chemfig{A-[:-5]-[::+20]-[::20]B-[::+20]-[::20]C-[::20]}

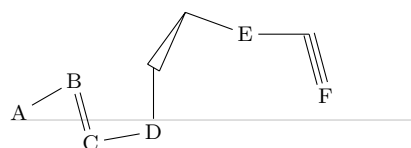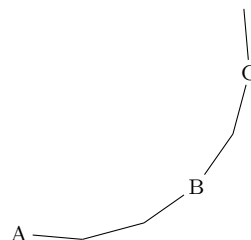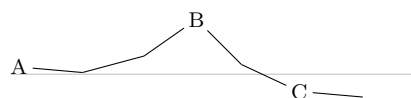One can "break" a chain of relative angles by putting an absolute or predefined angle where desired. Here, atom "B" is followed by a bond at an absolute angle of $315°$.

Result of relative angles followed by absolute

\chemfig{A-[:-5]-[::20]-[::20]B-[7]-[::20]C-[::20]}

4 Length of a bond

Rather than speaking of length of a bond, we should use the term interatomic spacing. If effect, only the interatomic spacing is adjustable with \setatomsep as we have seen on page 6. Once this parameter is set, the length of a bond depends on the content of atoms and, to a lesser extent, the angle the bond makes with the horizontal. It should be obvious that two "slimmer" atoms will have larger edge separations than two which are larger. This can be seen easily in the following example where an "I" atom is narrower than an "M" atom, which means that the bond between the "I" atoms is longer than that between the "M" atoms:

Influence of the size of atoms

```
\chemfig{I-I}\par
\chemfig{M-M}
```

I——I
M——M

This aspect of the size of atoms becomes particularly acute when the atom involves subscripts or superscripts. In this example, the bond is extremely short, to the point of confusion with a negative sign $-$:

Too-short bond

\chemfig{A^{++}_{2}-B^{-}_3}

$A_2^{++} - B_3^-$
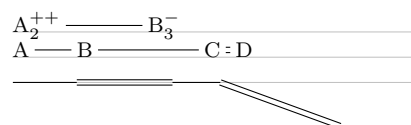
It is important to note that the exponent – is *put inside braces*. If this were not done, ChemFig would stop the atom on this character, which is a bond character. The atom would then be "B^", which would lead to unexpected results.

We see in the example above that is it sometimes necessary to increase (or perhaps reduce) the interatomic distance associated with a bond. For this, the optional argument to bonds is actually made up of several comma-separated fields. As we have seen, the first field specifies the angle. The second field, if it is not empty, is a coefficient which multiplies the default interatomic distance Δ. Thus, writing -[,2] asks that this bond have the default angle (first field is empty) and that the atoms it connects be separated by twice the default distance.

Modified bond length

```
\chemfig{A^{++}_{2}-[,2]B^{-}_3}\par
\chemfig{A-B-[,2]C=[,0.5]D}\par
\chemfig{-=[,1.5]-[,0.75]=[:-20,2]}
```

$A_2^{++} ——— B_3^-$
$A — B ——— C \colon D$

We can change the size of molecules by altering the font size or the argument of \setatomsep, possibly on both [2], being careful to confine these changes within a group if we want to limit the scope:

How to modify the size of molecule

```
\normalsize        \chemfig{H-[:30]O-[:-30]H}\par
\setatomsep{2.5em}\chemfig{H-[:30]O-[:-30]H}\par
\small             \chemfig{H-[:30]O-[:-30]H}\par
\footnotesize      \chemfig{H-[:30]O-[:-30]H}\par
\scriptsize        \chemfig{H-[:30]O-[:-30]H}\par
\tiny              \chemfig{H-[:30]O-[:-30]H}}
```

5   Departure and arrival atoms

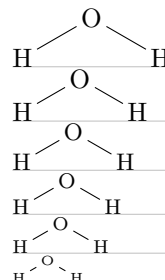A group of atoms can contain several atoms. Suppose we want to connect the group "ABCD" to the group "EFG" with a bond. ChemFig calculates which atom of the first group and which of the second group are to be connected by looking at the angle of bond relative to the horizontal. If the angle is between (but not including) $-90°$ and $90°$ (modulo $360°$) then the bond is made between the last atom of the first group and the first atom of the second group. In all other cases, the bond is made between the first atom of the first group and the last atom of the second group.

Here are some examples where the bond is in the interval $(-90, 90)$, and where the bond is made between D and E:

Default atom connections
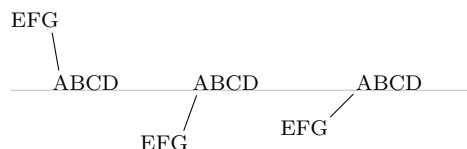
```
\chemfig{ABCD-[:75]EFG}\quad
\chemfig{ABCD-[:-85]EFG}\quad
\chemfig{ABCD-[1]EFG}
```
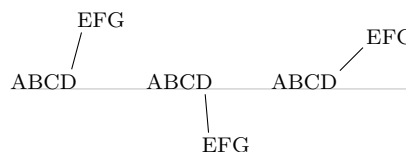
In the following examples, the angles are in the interval $[90, 270]$ and so the bond is made between A and G:

Default atom connections

```
\chemfig{ABCD-[:100]EFG}\quad
\chemfig{ABCD-[:-110]EFG}\quad
\chemfig{ABCD-[5]EFG}
```

One may sometimes want the bond partners to be atoms other than those determined by ChemFig. The departure and arrival atoms can be set with the optional bond argument by writing:

$$[,,<integer\ 1>,<integer\ 2>]$$

where <integer 1> and <integer 2> are the numbers of the desired departure and arrival atoms. These atoms must exist, otherwise an error message will be given.

Specified atom connections

```
\chemfig{ABCD-[:75,,2,3]EFG}\qquad
\chemfig{ABCD-[:75,,,2]EFG}\qquad
\chemfig{ABCD-[:75,,3,2]EFG}
```

---

[2]You can also use the second optional argument of \chemfig, see page 24.

6   Customization of bonds

There is a fifth and last optional argument for bonds which is found after the fourth comma:

$$[\,,\,,\,,\texttt{<tikz code>}]$$

This `<tikz code>` is passed directly to ti*kz* when the bond is drawn. There one can put characteristics such as colour (`red`), dash type (`dash pattern=on 2pt off 2pt`), thickness (`line width=2pt`), or even decoration if the ti*kz* decoration library has been loaded. A bond can be made invisible by writing "`draw=none`". To set several attributes, the syntax of ti*kz* is used, separating them by a comma:

```
┌─────────────────── Passing tikz code ───────────────────┐
│ \chemfig{A-[,,,,red]B}\par                        A —— B │
│ \chemfig{A-[,,,,dash pattern=on 2pt off 2pt]B}\par A ---- B │
│ \chemfig{A-[,,,,line width=2pt]B}\par             A ▬ B   │
│ \chemfig{A-[,,,,red,line width=2pt]B}             A ▬ B   │
└─────────────────────────────────────────────────────────┘
```

Numerous ti*kz* decoration libraries are available. For example, one can use the "`pathmorphing`" library by putting `\usetikzlibrary{decorations.pathmorphing}` in the preamble in order to draw wavy bonds:

```
┌─────────────────────── Wavy bonds ───────────────────────┐
│ \chemfig{A-[,3,,,decorate,decoration=snake]B}   A〜〜〜〜 B │
└──────────────────────────────────────────────────────────┘
```

Cram bonds ignore thickness and dash settings.

7   Default values

At the beginning of each molecule, the default values for the optional arguments are initialized. They are:

- $0°$ for the bond angle;
- 1 for the length multiplication coefficient;
- `<empty>` for the numbers of the departure and arrival atoms, which lets ChemFig calculate these based on the bond angle;
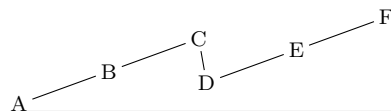- `<empty>` for the parameters passed to ti*kz*.

These default values can be changed for the whole molecule by beginning the molecule code with

$$[\texttt{<angle>},\texttt{<coeff>},\texttt{<n1>},\texttt{<n2>},\texttt{<tikz code}]$$

Thus, if the code of a molecule begins with `[:20,1.5]`, then all the bonds will be at angle of $20°$ by default, and the interatomic distances will have a length 1.5 times the default length. These default values can be overridden at any time by giving an optional argument, such as for the bond which follows atom "C" in this example:

```
┌─────────────────── Overriding default values ───────────────────┐
│                                                                  │
│ \chemfig{[:20,1.5]A-B-C-[:-80,0.7]D-E-F}                         │
│                                                                  │
└──────────────────────────────────────────────────────────────────┘
```

If something odd like `[1,1.5,2,2,red,thick]` is written, then unless otherwise indicated all the bonds will have an angle of $45°$, the interatomic distances will be 1.5 times the default distance, the bonds will begin and end on the second atom of each group, and the bonds will be red and thick:

```
┌─────────────────────── Default values ───────────────────────┐
│                                                                │
│ \chemfig{[1,1.5,2,2,red,thick]ABC-DEF=GHI}                    │
│                                                                │
└────────────────────────────────────────────────────────────────┘
```
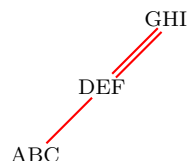
8   Branches

8.1   Principle

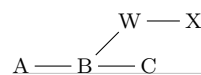Up to now, all the molecules have been linear, which is rare. A sub-molecule can be attached to an atom by following the atom with <code> in parentheses. This <code> is the code of the submolecule which will be attached to the atom.

In this example, the sub-molecule "-[1]W-X" will be attached to atom "B":
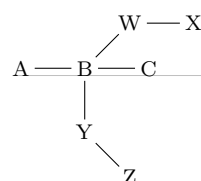
A Branch

```
\chemfig{A-B(-[1]W-X)-C}
```

There can be several sub-molecules which are to be attached to the same atom. Just have several parentheses containing the code for each sub-molecule:
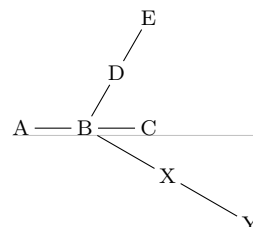
Multiple Branches

```
\chemfig{A-B(-[1]W-X)(-[6]Y-[7]Z)-C}
```

The code of each sub-molecule can define its own default values, which will be valid throughout the whole sub-molecule. Here a sub-molecule "[:60]-D-E" is attached to atom "B", with a default angle of 60° absolute. A second sub-molecule "[::-60,1.5]-X-Y" is attached to "B" with a default bond angle 60° less than that of the preceding bond (which will be the one between "A" and "B") and with an interatomic distance 1.5 times the default value:

Default values in Branches
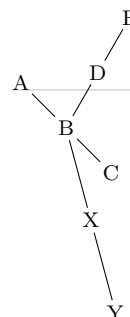
```
\chemfig{A-B([:60]-D-E)([::-30,1.5]-X-Y)-C}
```

Observe what happens if, at the beginning of the main molecule, one writes "[:-45]":

Effect of the default Bond angle

```
\chemfig{[:-45]A-B([:60]-D-E)([::-30,1.5]-X-Y)-C}
```

We see that the angle between the bond B-C and the bond B-X stays at 30° because it is a relative angle for the sub-molecule "-X-Y". By contrast, the branch "-D-E" stays inclined at 60° to the horizontal, and does not follow the rotation given by the −45° angle at the beginning; this is expected because "-D-E" has an absolute angle. It is essential that all the angles be relative in order to rotate the whole molecule.

8.2   Nesting

Sub-molecules may be nested, and the rules seen in the preceding paragraphs stay in force:

Nested branches

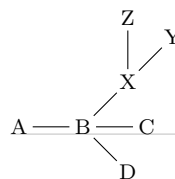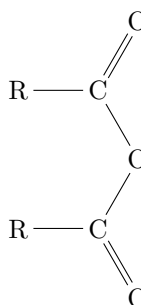`\chemfig{A-B([1]-X([2]-Z)-Y)(-[7]D)-C}`

8.3   Method

Suppose now that we want to draw an acid anhydride molecule: R —— C

The best way to get this is to find the longest chain. Here, for example, we can draw the chain R–C–O–C–R taking into account angles and using only relative angles:

Acid anhydride structure

`\chemfig{R-C-[::-60]O-[::-60]C-[::-60]R}`
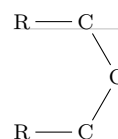
To this structure we just have to add two "=O" sub-molecules to each of the carbon atoms:

Acid anhydride
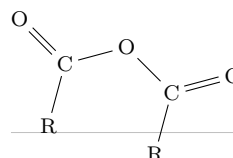
`\chemfig{R-C(=[::+60]O)-[::-60]O-[::-60]C(=[::+60]O)-[::-60]R}`
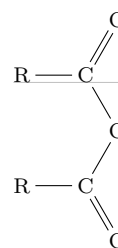
Because we used only relative angles, we can rotate this molecule by giving a default angle of e.g. 75°:

Rotation of a molecule

`\chemfig{[:75]R-C(=[::+60]O)-[::-60]O-[::-60]C(=[::+60]O)-[::-60]R}`

9   Connecting distant atoms

We have seen how to connect atoms *which are adjacent in the code.* It is often necessary to connect atoms which are not next to each other in the code. Let's call these particular bonds "distant bonds".

Let's take this molecule:

<div style="border:1px solid">

Branched structure

`\chemfig{A-B(-[1]W-X)(-[7]Y-Z)-C}`

```
              W —— X
              /
A —— B —— C
              \
              Y —— Z
```

</div>

and suppose that we want to connect the atoms X and C. In this case, **ChemFig** allows a "hook" to be placed *immediately* after the atom of interest. The character used for a hook is "?" because of its similarity to a hook. So, if one writes X? then the atom X will have a hook. Later in the code, all atoms followed by a ? will be connected to X:

<div style="border:1px solid">

Distant bond

`\chemfig{A-B(-[1]W-X?)(-[7]Y-Z)-C?}`

```
              W —— X
              /     \
A —— B —— C
              \
              Y —— Z
```

</div>

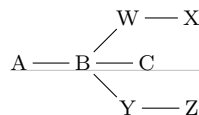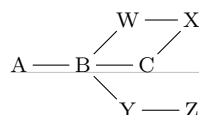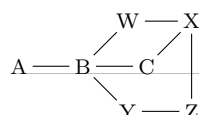We could connect other atoms to X by following them with ?. Here it's the atoms C and Z:

<div style="border:1px solid">

Several distant bonds

`\chemfig{A-B(-[1]W-X?)(-[7]Y-Z?)-C?}`

```
              W —— X
              /     |
A —— B —— C  |
              \     |
              Y —— Z
```

</div>

Now imagine if we were to leave the distant bonds X–C and X–Zwhile adding another: A–W. We must therefore ask for two *different* hooks, one on A and the other on X. Fortunately the character ? has an optional argument:

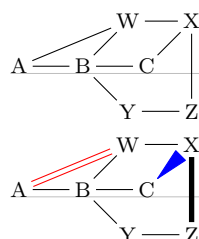<p align="center">?[<name>,<bond>,<tikz>]</p>

where each field takes its default value if it is empty:

- The <name> is the name of the hook: all alphanumeric characters (a...z, A...Z, 0...9) are allowed[3]. The name is a by default. In the first occurrence of the hook with this name, only this field is used.
- <bond> specifies how the atom with the current occurrence of the named hook is to be bonded to the atom with the first occurrence of the hook. There are two ways this can be done. First, this field can be an integer representing the desired bond type: 1=single bond, 2=double bond, etc. (See the table on page 5 for the bond codes.)
  Second, the field can be one of the bond character codes, provided that this character is *between braces*.
- <tikz> will be passed directly to ti*kz* as we have seen with regular bonds.

Here is our molecule with the required distant bonds, then with the bond A–W and X–C customized:

<div style="border:1px solid">

Multiple distant bonds
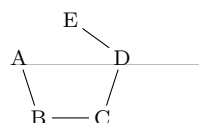
`\chemfig{A?[a]-B(-[1]W?[a]-X?[b])(-[7]Y-Z?[b])-C?[b]}\par\medskip`
`\chemfig{A?[a]-B(-[1]W?[a,2,red]-X?[b])(-[7]Y-`
`Z?[b,1,{line width=2pt}])-C?[b,{>},blue]}`

```
                   W —— X
                  / \    |
A —— B —— C  |
                  \      |
                  Y —— Z
```

```
                   W —— X
                 ≡  \    |
A ≡ B —— C  |
                  \      |
                  Y —— Z
```

</div>

Several different hooks can be written after an atom. Suppose that in this unfinished pentagon, we wish to connect A–E, A–C and E–C:

<div style="border:1px solid">

An incomplete ring

`\chemfig{A-[:-72]B-C-[:72]D-[:144]E}`

```
          E
         / \
A      D
 \     /
  B —— C
```

</div>

Then we must do this:

---

[3]This is not exactly right. Actually all the characters that can be put between \csname...\endcsname are allowed.

Multiple distant bonds

`\chemfig{A?[a]-[:-72]B-C?[a]?[b]-[:72]D-[:144]E?[a]?[b]}`

---

10 Rings

The preceding example shows how to draw a regular polygon, but the method used is tedious because the angles depend on the number of sides of the polygon.
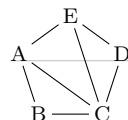
10.1 Syntax

ChemFig can easily draw regular polygons. The idea is to attach a ring to an `<atom>` outside the ring with this syntax:
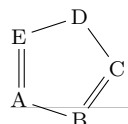
<atom>*<n>(<code>)

`<n>` is the number of sides of the polygon and the `<code>` describes the bonds and groups of atoms which make up its edges and vertices. This code *must* begin with a bond because the atom is outside the ring.

Here is a 5-ring, attached to the atom "A":

5-ring
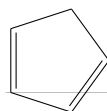
`\chemfig{A*5(-B=C-D-E=)}`

A ring can also be drawn with one, several, or all the groups of atoms empty, as is the case for diagrams outside rings:

5-ring with empty groups
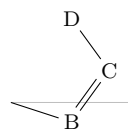
`\chemfig{*5(-=---=)}`

A ring can be incomplete:

Incomplete 5-ring
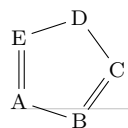
`\chemfig{*5(-B=C-D)}`

If a ring has a code which contains too many bonds and atom groups for the given number of vertices, all the bonds and groups over the maximum allowed are ignored:

Truncated 5-ring

`\chemfig{A*5(-B=C-D-E=F-G=H-I)}`

It is possible to draw a circle or an arc in the inside of a ring. To do so, the following syntax is used:

<atom>**[<angle 1>,<angle 2>,<tikz>]<n>(<code>)

where each field of the optional argument takes its default value if it is empty:

- `<angle 1>` and `<angle 2>` are the absolute angles of the start and finish of the arc. These default to 0° and 360° respectively so that a circle is drawn by default;
- `<tikz>` is the code that will be passed to ti*kz* for drawing the arc.

Rings and arcs

```
\chemfig{**6(------)}\quad
\chemfig{**[30,330]5(-----)}\quad
\chemfig{**[0,270,dash pattern=on 2pt off 2pt]4(----)}
```
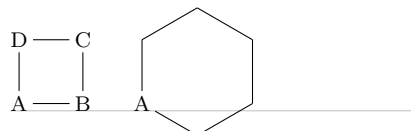
10.2   Angular position

10.2.1   At the start

As can be seen in the examples above, the rule is that the attachment atom "A" is always at the south-west of the ring. Furthermore, the ring is always constructed counterclockwise, and the last bond descends vertically onto the attachment atom:
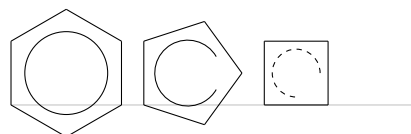
Angular position of rings
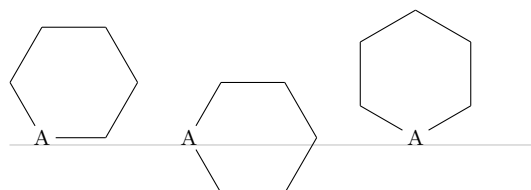
```
\chemfig{A*4(-B-C-D-)}\qquad\chemfig{A*6(------)}
```

If this angular position is not convenient, it is possible to specify another angle using the optional argument at the beginning of the molecule. Here is a 6-cycle which has been rotated by +30°, by −30°, and lastly by +60°:

Rotation of rings

```
\chemfig{[:30]A*6(------)}\qquad
\chemfig{[:-30]A*6(------)}\qquad
\chemfig{[:60]A*6(------)}
```

10.2.2   After a bond

When a ring does not begin a molecule and one or more bonds have already been drawn, the default angular position changes: the ring is drawn is such a way that the bond ending on the attachment atom bisects the angle formed by the first and last sides of the ring.

Here is a simple case:

Bond ending on a ring

```
\chemfig{A-B*5(-C-D-E-F-)}
```
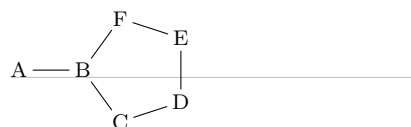
The rule remains valid, whatever the angle of the preceding bond:

Bonds ending on a ring

```
\chemfig{A-[:25]B*4(----)}\vskip5pt
\chemfig{A=[:-30]*6(=-=-=-)}
```
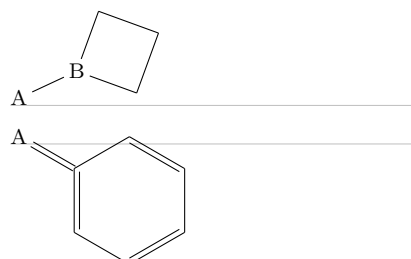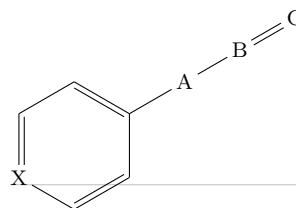
10.3   Branches on a ring

To have branches attached to the vertices of a ring, we use the syntax we have already seen:

<atom>(<code>)
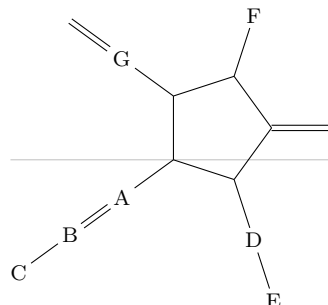
where the <code> is that of the sub-molecule and the <atom> is at the vertex. Unique to rings, the default angle of the sub-molecule is not 0° but is calculated so that it will bisect the sides leaving the vertex:

```
\chemfig{X*6(-=-(-A-B=C)=-=-)}
```
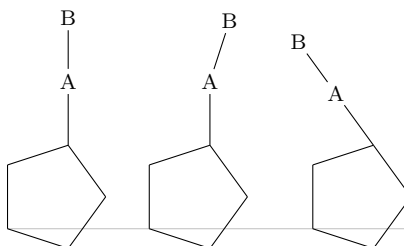
A sub-molecule can be attached to the first vertex of a ring, just like the other vertices:

```
\chemfig{*5((-A=B-C)-(-D-E)-(=)-(-F)-(-G=)-)}
```

If one wants the bond leaving a vertex not to be the bisector of its sides, one can tinker with the optional global parameter or the optional bond parameter:
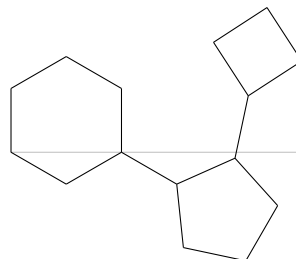
```
\chemfig{*5(---([:90]-A-B)--)}\qquad
\chemfig{*5(---(-[:90]A-B)--)}\qquad
\chemfig{*5(---([::+0]-A-B)--)}
```

It is worth noting that in the third example, where a relative angle of 0° was given, the bonds of the branch are drawn in line with the preceding bond in the ring. This is the rule on page 8 which specified that the reference angle was that of the bond last drawn.

We can now connect together rings with bonds:

```
\chemfig{*6(--(-*5(----(-*4(----))-))----)}
```
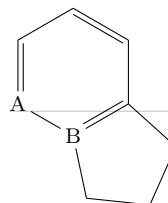
10.4   Nested rings

To "glue" two rings together, the syntax is only slightly different: the vertex is specified where the other ring is going to start. Simply follow this vertex by the usual syntax for a ring. Here for example is a 5-ring which is attached to the second vertex of a 6-ring:


```
\chemfig{A*6(-B*5(----)=-=-=)}
```

Note that the ring which is going to be attached to the main ring has an angular position such that two of the rings' sides coincide. In addition, the 5-ring has only four bonds "----". In effect, the fifth will be useless because it is the second side of the 6-ring, which has already been drawn.

It is quite possible to glue multiple rings together:
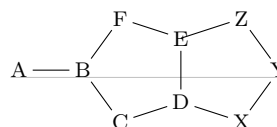

```
\chemfig{*5(--*6(-*4(-*5(----)--)----)---)}
```
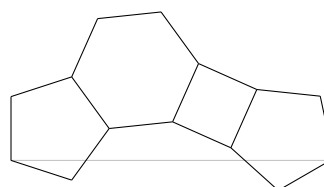
There is a case where a trick must be used. It can be seen in this example that the fourth side of the second 5-ring just passes through the centre of atom "E".
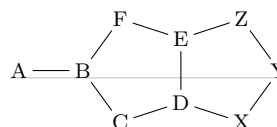

```
\chemfig{A-B*5(-C-D*5(-X-Y-Z-)-E-F-)}
```

This is normal because the second 5-ring (which is attached to atom "D") is drawn *before* ChemFig knows about atom "E". In this case, it is necessary to use two hooks to draw the bond Z–E:


```
\chemfig{A-B*5(-C-D*5(-X-Y-Z?)-E?-F-)}
```

We could also use a at the last vertex of the 5-ring:


```
\chemfig{A-B*5(-C-D*5(-X-Y-Z-\phantom{E})-E-F-)}
```

10.5   Rings and groups of atoms

Some care must be taken with rings when one or more vertices are made up of groups of atoms:

Ring and groups of atoms

`\chemfig{AB*5(-CDE-F-GH-I-)}`

In order for the ring to have a regular shape, it is necessary to override the ChemFig mechanism which automatically calculates the departure and arrival atoms of bonds. Here, C-F and F-G must be connected by using the optional argument of these bonds:

Forced departure and arrival atoms

`\chemfig{AB*5(-CDE-[,,1]F-[,,1]GH-I-)}`

11   Representing electron movements

Starting with ChemFig version 0.3, we can represent the movement of electrons in mesomeric effects or reaction mechanisms. This is done by marking the departure and arrival points of the electron movement arrow using the syntax "@{<argument>}". This syntax allows a ti*kz* node to be placed and makes this node accessible outside the argument of the \chemfig command thanks to the "remember picture" option which is passed to all the "tikzpicture" environments. It is assumed that the viewer supports "picture remembering" and that the compilation is done twice.

Two types of diagrams can arise, so we can ask for:

- a zero size node on a bond using the syntax "@{<name>,<coeff>}" placed at the beginning of the optional argument of the relevant bond, without being followed by a comma if there is a first optional argument. In this case, the node takes the name "<name>" and the <coeff>, which must be between 0 and 1, determines where the node is located on the bond. If "@{<name>}" is used, the <coeff> is set to 0.5 by default, which means that the node is placed halfway along the bond;
- a node on an atom using the syntax "@{<name>}" immediately before the relevant atom. In this case, the node has exactly the same footprint as the atom, but may be empty and therefore have zero dimensions.

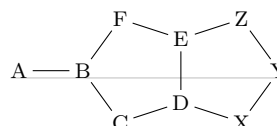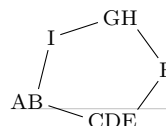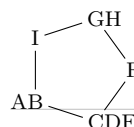Once the \chemfig command has drawn the molecule(s) and has placed the nodes with the syntax described above, we can connect these nodes to each other with ti*kz* instructions. These instructions are placed in the argument of the command \chemmove[4] and has the following syntax if (for example) we need to connect a node named "<name1>" to the node named "<name2>":

`\chemmove[<opt>]{\draw[<tikz opt>](<name1>)<tikz link>(<name2>);}`

The optional argument <opt> of the \chemmove command will be added to the argument of the tikzpicture environment in which the links between the nodes will be drawn. The <tikz opt> and <tikz link> instructions are describe in detail in the documentation of the ti*kz* package.

11.1   Mesomeric effects

To make these concepts concrete, let's take the example of a mesomeric effect involving a double bond and non-bonding lone pair conjugate. Let's begin with the possible delocalization of electrons from the double bond. We will place a node named "db" (double bond) in the middle of the double bond and a node named "a1" on the end of the double bond.

Mesomeric effect I

```
\chemfig{@{a1}=_[@{db}::30]-[::-60]\lewis{2,X}}
\chemrel{<->}
\chemfig{\chemabove{\vphantom{X}}{\ominus}-[::30]=_[::-60]
\chemabove{X}{\scriptstyle\oplus}}
\chemmove{\draw[->](db)..  controls +(80:8mm) and +(145:8mm)..  (a1);}
```


---

[4]Actually, the \chemmove command puts its argument in a "tikzpicture" environment with the options "remember picture, overlay".

As noted above, there is no comma after the node placed in the optional arguments of a bond; we write "=_[@{db}::30]" and not "=_[@{db},::30]" as one might be tempted to do.

To link the nodes "db" and "a1" we have used the following syntax:

```
\chemmove{\draw[->](db)..controls +(80:8mm) and +(145:8mm)..(a1);}
```

In this example we ask for an arrow ([->]) and we use two control points[5]. These will be located using polar coordinates at 80° and 8 mm from "db" for the first and at 145° and 8 mm from "a1" for the second. Though this syntax may seem complicated at first reading, one need not be alarmed because its use will usually be a matter of copying and pasting. Only the names and coordinates of the control points need be changed, as can be verified from the example below, where an arrow has been added from the lone pair (node "dnl" to the single bond (node "sb").

Mesomeric effect 2

```
\chemfig{@{a1}=_[@{db}::30]-[@{sb}::-60]@{dnl}\lewis{2,X}}
\chemrel{<->}
\chemfig{\chemabove{\vphantom{X}}{\ominus}-[::30]=_[::-60]
\chemabove{X}{\scriptstyle\oplus}}
\chemmove{
    \draw[->](db)..controls +(100:5mm) and +(145:5mm)..(a1);
    \draw[->](dnl)..controls +(90:4mm) and +(45:4mm)..(sb);}
```

For our new arrow we have set the control points as follows: 4 mm at an angle of 90° from "dnl" and 4 mm at an angle of 45° from "sb". But we are not completely satisfied, since we would like the arrow not to touch the line segment representing the lone pair. To do this we will add some options to our arrow.

Mesomeric effect 3

```
\chemfig{@{a1}=_[@{db}::30]-[@{sb}::-60]@{dnl}\lewis{2,X}}
\chemrel{<->}
\chemfig{\chemabove{\vphantom{X}}{\ominus}-[::30]=_[::-60]
\chemabove{X}{\scriptstyle\oplus}}
\chemmove[->]{
    \draw(db)..  controls +(100:5mm) and +(145:5mm)..  (a1);
    \draw[shorten <=3pt,shorten >=1pt](dnl) ..  controls +(90:4mm)
         and +(45:4mm) ..  (sb);}
```

The option "shorten <=3pt" indicates that the tail of the arrow is to be shortened by 3 pt just as "shorten >=2pt" means that the head of the arrow is shortened by 2 pt.

We can use all the power of ti*k*z instructions to modify the style of the arrow. Here we change the head of the arrow leaving the double bound, writing "-stealth" instead of "->", and we draw the arrow with a fine dashed red line. We also add the letter π above the middle of the arrow:

Mesomeric effect 4
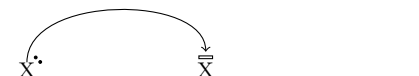
```
\chemfig{@{a1}=_[@{db}::30]-[@{sb}::-60]@{dnl}\lewis{2,X}}
\chemrel{<->}
\chemfig{\chemabove{\vphantom{X}}{\ominus}-[::30]=_[::-60]
\chemabove{X}{\scriptstyle\oplus}}
\chemmove{
    \draw[-stealth,thin,dash pattern= on 2pt off 2pt,red]
        (db)..  controls +(100:5mm) and +(145:5mm)..
        node[sloped,above] {$\pi$} (a1);
    \draw[->, shorten <=3pt, shorten >= 1pt]
        (dnl)..  controls +(90:4mm) and +(45:4mm)..  (sb);}
```

In the following example, we'll see how to indicate the position of the departure or arrival anchor points of the arrow. If we write

---

Departure or arrival anchor point I

```
\chemfig{@{x1}\lewis{1:,X}}
\hspace{2cm}
\chemfig{@{x2}\lewis{2|,X}}
\chemmove{\draw[->,shorten >=4pt]
    (x1)..  controls +(90:1cm) and +(90:1cm)..  (x2);}
```


---

[5]To find all the ways of connecting two nodes with ti*k*z, read the documentation for that package.

Note that the tail of the arrow does not leave correctly from our electrons; it leaves from the middle of the upper edge of the node. Indeed, we chose a departure angle of 90 ° and so ti*kz* makes the arrow leave from the anchor "x1.90" which corresponds to the intersection of the ray leaving from the centre of node "x1" at a 90° angle relative to the horizontal and of the edge of the rectangular node. To get the arrow departure angle that we want, we must specify its position. After some trial and error, it is "x1.57":

Departure or arrival anchor point 2

```
\chemfig{@{x1}\lewis{1:,X}}
\hspace{2cm}
\chemfig{@{x2}\lewis{2|,X}}
\chemmove{\draw[->,shorten <=4pt,shorten >=4pt]
    (x1.57).. controls +(60:1cm) and +(120:1cm).. (x2);}
```

In some cases it will be easier to use Cartesian coordinated for the control points. Here we use just one control point placed 1 cm to the right of and 1.5 cm above "x1":

A single control point

```
\chemfig{@{x1}\lewis{1:,X}}
\hspace{2cm}
\chemfig{@{x2}\lewis{2|,X}}
\chemmove{\draw[->,shorten <=4pt,shorten >=4pt]
    (x1.57).. controls +(1cm,1.5cm).. (x2);}
```
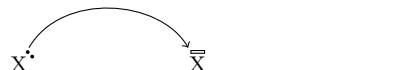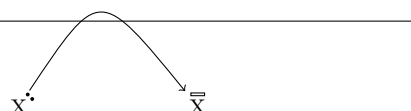
All the graphics drawn by means of the command \chemmove are superimposed and will not be included in the bounding boxes. We can see this in the preceding example.

11.2   Reaction mechanisms

Thanks to the option remenber picture which is passed to all the "tikzpicture" environments we can easily draw arrows indicating reaction mechanisms. Let's take for example the first step of the esterification reaction.

Esterification: step 1

```
\setatomsep{7mm}
\setchemrel{}{}{5mm}
\chemfig{R-@{dnl}\lewis{26,O}-H}
\chemsign{+}
\chemfig{R-@{atoc}C([6]-OH)=[@{db}]O}
\chemrel[\chemfig{@{atoh}\chemabove{H}{\scriptstyle\oplus}}]{<>}
\chemmove[->,shorten <=2pt]{
    \draw[shorten >=2pt](dnl)..controls +(90:1cm)and+(north:1cm)..(atoc);
    \draw[shorten >=6pt](db)..controls +(north:5mm)and+(100:1cm)..(atoh);}
```
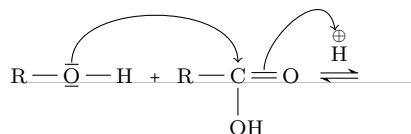
The use of the \chemabove{<code>}{<materiel>} command does not change the dimensions of the bounding box of <code>. For this reason we can run into some difficulty in pointing to the symbol representing the charge carried ($\oplus$ or $\ominus$). In the example above the solution is to create a control point with an angle of 110° at 1 cm from "atoh" and to shorten the arrow by 6pt. In the following example, the second step of the esterification reaction, we can see that the arrow can take more complicated forms without complicating the code.

Esterification: step 2

```
\setatomsep{7mm}
\setchemrel{}{}{5mm}
\chemfig{R-O-C(-[2]R)(-[6]OH)-@{dnl}\lewis{26,O}H}\hspace{1cm}
\chemfig{@{atoh}\chemabove{H}{\scriptstyle\oplus}}
\chemmove{
    \draw[->,shorten <=2pt, shorten >=7pt]
        (dnl).. controls +(south:1cm) and +(north:1.5cm).. (atoh);}
```
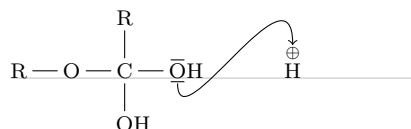
The rest is left as an exercise to the reader. . . .

12   Writing a name under a molecule

For convenience, ChemFig can write the name of a molecule underneath it with the command

```
\chemname[<dim>]{\chemfig{<code of the molecule>}}{<name>}
```

The `<dim>`, which is 1.5ex by default, will be inserted between the baseline of the molecule and the top of the letters of the `<name>`. The `<name>` will be centred relative to the molecule, but the `<name>` may not contain multiple paragraphs. As we see in this example: H —— O —— H, the `<name>` which is displayed under the molecule is

<div align="center">**The water molecule: H$_2$O**</div>

taken into account only for the vertical size of the bounding box. The horizontal size of `<name>` is always zero.

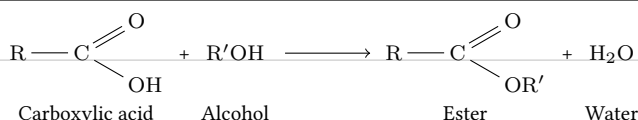Here is a reaction with the names under the molecules:


```
\chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic acid}
\chemsign{+}
\chemname{\chemfig{R'OH}}{Alcohol}
\chemrel{->}
\chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
\chemsign{+}
\chemname{\chemfig{H_20}}{Water}
```

There are some limitations to this command. Suppose we switch the acid and the alcohol on the left side:


```
\chemname{\chemfig{R'OH}}{Alcohol}
\chemsign{+}
\chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic acid}
\chemrel{->}
\chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
\chemsign{+}
\chemname{\chemfig{H_20}}{Water}
```

In fact, to draw the `<name>` the command `\chemname` inserts 1.5ex + *the largest of the depths*[6] *of the molecules thus far* below the baseline of each molecule (light grey for the examples in this manual). The command `\chenameinit{<stuff>}` initializes this largest depth with the `<stuff>`. Therefore one should:

- write `\chemnameinit{<deepest molecule>}` before using the `\chemname` command in a reaction, unless the reaction begins with the deepest molecule;
- write `\chemnameinit{}` after having written all the names in a chemical reaction lest the greatest depth in this reaction interfere with a future reaction.

Thus the correct code uses `\chemnameinit` before and after the reaction:


```
\chemnameinit{\chemfig{R-C(-[:-30]OH)=[:30]O}}
\chemname{\chemfig{R'OH}}{Alcohol}
\chemsign{+}
\chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic acid}
\chemrel{->}
\chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
\chemsign{+}
\chemname{\chemfig{H_20}}{Water}
\chemnameinit{}
```

Finally, to write a name on multiple lines, the command `\\` encountered in a `<name>` causes a line break[7]:

---

[6]In TeX terms, the depth is the dimension which extends vertically below the baseline.

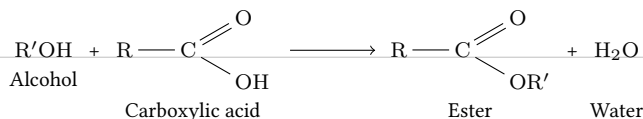[7]Conversely, the command `\par` is forbidden and causes a compilation error.

```
Name on 2 lines
\chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic\\acid}
\chemsign{+}
\chemname{\chemfig{R'OH}}{Alcohol}
\chemrel{->}
\chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
\chemsign{+}
\chemname{\chemfig{H_2O}}{Water}
\chemnameinit{}
```

$$R \!-\! C \overset{\displaystyle O}{\underset{\displaystyle OH}{\Big\langle}} \quad + \quad R'OH \quad \longrightarrow \quad R \!-\! C \overset{\displaystyle O}{\underset{\displaystyle OR'}{\Big\langle}} \quad + \quad H_2O$$

| Carboxylic   Alcohol   Ester   Water |
| acid |
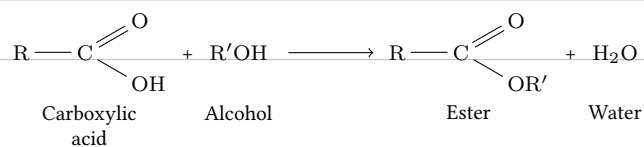
PART III

Advanced usage

1   Separating atoms

The separating atom mechanism described previously extends each atom until the next capital letter or one of the characters ⊟ ⊟ ⊡ 〔 ! ⊞ ⊲ ⊳ @

In certain cases this automatic separation produces incorrect atoms which can translate into an imperfect diagram. Consider this example molecule, noting that the "(" character is placed between braces so that ChemFig doesn't incorrectly create a branch:

```
\chemfig{CH_3CH_2-[:-60,,3]C(-[:-120]H_3C)=C(-[:-60]H)-[:60]C{(}CH_3{)}_3}
```

CH₃CH₂ and C(CH₃)₃ etc.
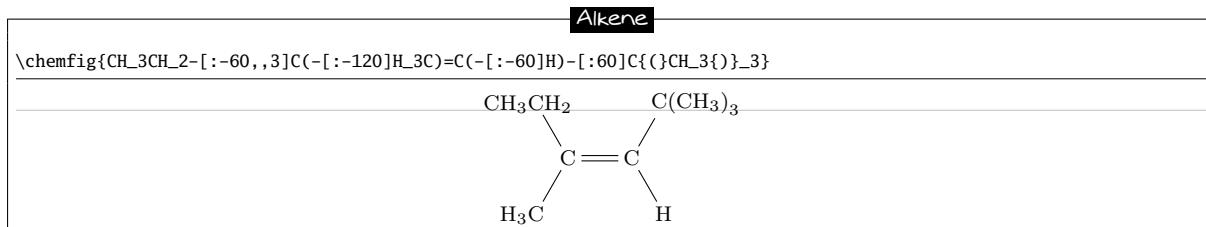
We find that the bond which arrives at the carbon atom in the upper right is too short. This happens because, if we apply the ChemFig rules for separating atoms to the upper right group, the atoms are split in this way: "C{(}", "C", "H_3{)}_3". We now realize that the first atom contains a parenthesis and thus has too great a depth in math mode; we can see this by making the bounding boxes visible:
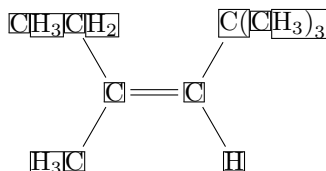
The character "|" forces splitting of the atom when it is encountered. Thus we can write C|{(CH_3)_3} to ensure that ChemFig separates just two atoms here: "C" and "{(CH_3)_3}". The problem of the too-short bond is thus solved:

```
\chemfig{CH_3CH_2-[:-60,,3]C(-[:-120]H_3C)=C(-[:-60]H)-[:60]C|{(CH_3)_3}}
```
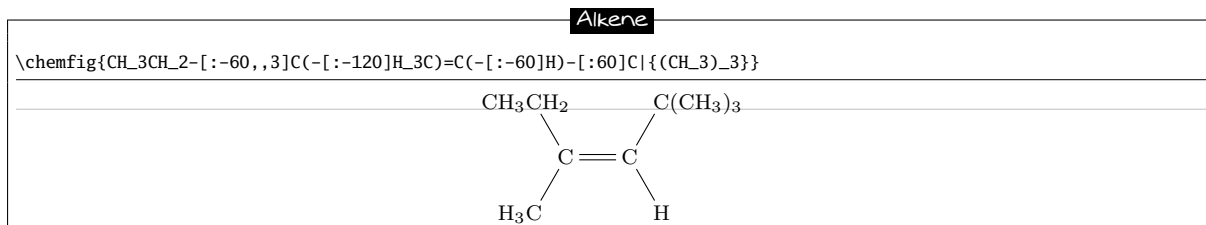
2   Displaying atoms

Once a molecule has been split into atoms, the macro \printatom is called internally by ChemFig in order to display each atom. Its sole argument is the code of the atom to be displayed (e.g. "H_3"). By default, this macro enters math mode and displays its argument with the math font family "rm". It is defined by the following code:

- \newcommand*\printatom[1]{\ensuremath{\mathrm{#1}}}   when compiling with LaTeX

- \def\printatom#1{\ifmmode\rm#1\else$\rm#1$\fi}   when compiling with εTEX ou ConTEXtX.

One can modify the code of this macro to customize how atoms are displayed. In the following example, we redefine \printatom so that each atom will be enclosed in a rectangular box:

Redefinition of \printatom

```
\fboxsep=1pt
\renewcommand*\printatom[1]{\fbox{\ensuremath{\mathrm{#1}}}}
\chemfig{H_3C-C(=[:30]O)(-[:-30]OH)}
```

Here is how to redefine it to use the "sf" font family of math mode:
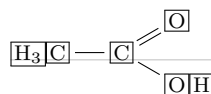
Atoms displayed with "sf" font family

```
\renewcommand*\printatom[1]{\ensuremath{\mathsf{#1}}}
\chemfig{H_3C-C(=[:30]O)(-[:-30]OH)}
```
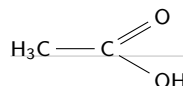
3   Optional arguments

The \chemfig command takes two optional arguments; their syntax is as follows:

$$\text{\chemfig[<opt1>][<opt2>]\{<molecule code>\}}$$

The first optional argument <opt1> contains ti*kz* instructions which will be passed to the tikzpicture environment in which the molecule is drawn. The second optional argument <opt2> contains ti*kz* instructions which will be executed when each node[8] is drawn.

With the use of the first optional argument one can, for example, choose the global colour or thickness of lines:

Style choice

```
\chemfig{A-B-[2]C}\par\medskip
\chemfig[line width=1.5pt]{A-B-[2]C}\par\medskip
\chemfig[red]{A-B-[2]C}
```

With the second optional argument, one can choose the colour of nodes drawn by ti*kz*, change the angle of the drawing or its scale:

Style choices

```
\chemfig{A-B-[2]C}\par\medskip
\chemfig[][red]{A-B-[2]C}\par\medskip
\chemfig[dash pattern=on 1pt off 2pt][red]{A-B-[2]C}\par\medskip
\chemfig[][rotate=20]{A-B-[2]C}\par\medskip
\chemfig[][scale=0.5]{A-B-[2]C}
```

---

[8]These instructions are added to the end of the argument of every node/.style{<argument>}. This argument contains by default the following instructions: "anchor=base,inner sep=0pt,outer sep=0pt,minimum size=0pt".

4 Vertical alignment

In some cases with condensed structural diagram of molecules having horizontal bonds, the placement of groups of atoms is incorrect.

Careful study of the following example shows that the groups of atoms are not correctly aligned on the baseline:


```
\Huge\setatomsep{2em}
\chemfig{A^1-B-C-D}\qquad
\chemfig{E_1-F-G-H}
```

Surprisingly, the second atom is correctly aligned while the last two undergo a vertical shift which seems to be the results of the different height of the bounding box of the atoms "A^1" and "E_1''".

In order to understand this phenomenon, we need to consider how ChemFig places groups of atoms relative to each other. Let us limit ourselves to the case of horizontal bonds in order to simplify terminology, although the algorithm is the same for other bonds. A horizontal bond leaves from the middle of the right side of the bounding box of the departure atom of this bond. The arrival atom is positioned in such a way that the middle of the left side of its bounding box is at the end of the bond. It follows that the vertical placement of the arrival atom depends on the height of the departure atom. To limit this phenomenon, ChemFig adds to each arrival atom the \vphantom of the departure atom, but does not include it in the contents of the arrival atom; this \vphantom is not intended to affect the following atoms. The atoms remaining in each group are aligned so that their baseline coincides with the baseline of the preceding atom.

The defective alignment can thus be explained. The atoms "B" and "F" are aligned correctly as they reflect the height of the atoms before them because of their \vphantom. For the atoms "C" and "F", the heights of the immediately preceding atoms are taken into account, but those of the atoms "A^1" and "E_1" are ignored! It follows that these atoms are a little too high or too low, depending on the height of these bonds.

We can show this by making visible the bounding boxes of the atoms; one sees clearly that the atoms "B" and "F" have bounding boxes that reflect the heights of the immediately preceding atoms:
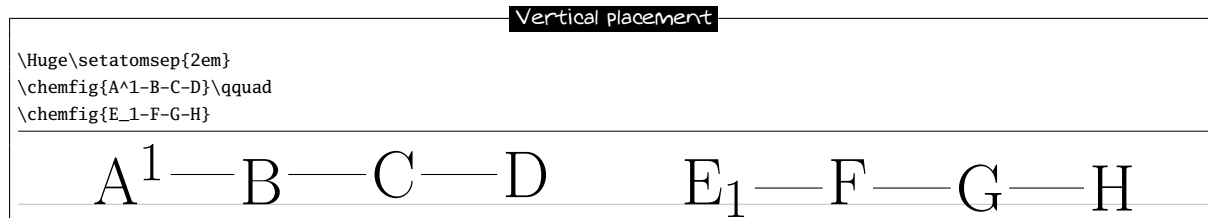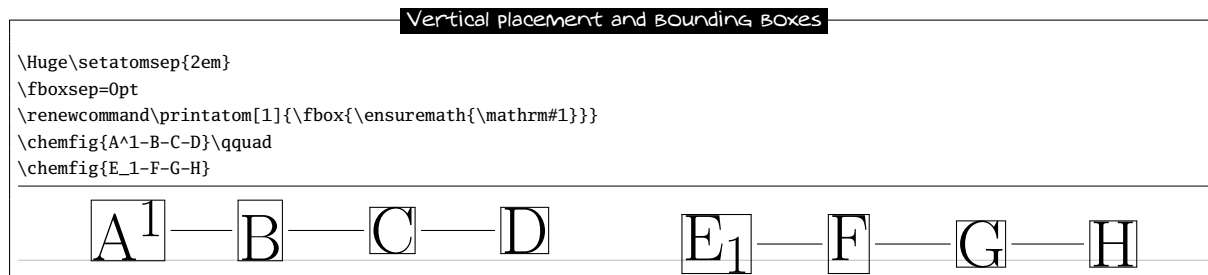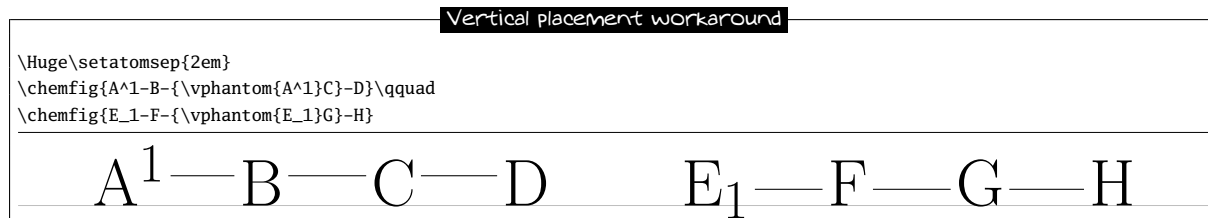


```
\Huge\setatomsep{2em}
\fboxsep=0pt
\renewcommand\printatom[1]{\fbox{\ensuremath{\mathrm#1}}}
\chemfig{A^1-B-C-D}\qquad
\chemfig{E_1-F-G-H}
```

Since there is no satisfactory manual solution, this problem can be worked around manually by putting *inside* the third atom a \vphantom having the same height as the first, so that the height affects the following atoms:


```
\Huge\setatomsep{2em}
\chemfig{A^1-B-{\vphantom{A^1}C}-D}\qquad
\chemfig{E_1-F-{\vphantom{E_1}G}-H}
```

For any group of atoms it is possible to temporarily deactivate the alignment adjustment mechanism and thus neutralize the \vphantom. Simply place the \chemskipalign command in the group of atoms; the alignment will resume in the following group of atoms as if the group of atoms containing \chemskipalign had never existed. The following example shows the effects of this instruction: the reference point of the box containing the first atom is placed at the level of the bond which arrives from the left. The bounding boxes of the atoms are drawn in the second line.

Deactivation of the alignment mechanism

```
\large
\chemfig{A-.-B}\quad
\chemfig{A\chemskipalign.-B}\par\bigskip
\fboxsep=0pt
\renewcommand\printatom[1]{\fbox{\ensuremath{\mathrm{#1}}}}
\chemfig{A-.-B}\quad
\chemfig{A\chemskipalign.-B}
```

A ——— . ——— B   A ——— · ——— B

A ——— ⎕ ——— B   A ——— ▫ ——— B
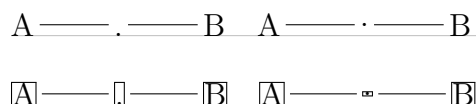
This command is to be used with caution lest the alignment of atoms in the next group be disrupted. In general, all will be well if the group of atoms featuring \chemskipalign contains *a single atom* whose height and depth are *less* than those of the preceding and following atoms, and if the preceding and following atoms have identical heights and depths. Here is an example of the mess that results when the group of atoms contains two atoms, here "\chemskipalign." and "B":

Consequence of the \chemskipalign command

```
\large
\fboxsep=0pt
\renewcommand\printatom[1]{\fbox{\ensuremath{\mathrm{#1}}}}
\chemfig{A-\chemskipalign.B-C}
```
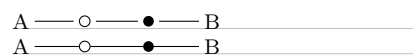
A ——— ▫B ——— C

This feature can sometimes be useful. Suppose we want to draw the following molecule

A ——— ○ ——— ● ——— B

We can define commands which will draw the empty and full disks with ti*kz*. To ensure that these disks are at the right height, namely the height of the bond arriving at them, we will use the command \chemskipalign. In the second line below the bonds are "stuck" to the disks by using the ability to change the bond shortening with the "#" character, a feature seen on page 6.

Use of \chemskipalign and #

```
\begingroup
\def\emptydisk{\chemskipalign\tikz\draw(0,0)circle(2pt);}
\def\fulldisk{\chemskipalign\tikz\fill(0,0)circle(2pt);}
\chemfig{A-\emptydisk-\fulldisk-B}\par
\chemfig{A-#(,0pt)\emptydisk-#(0pt,0pt)\fulldisk-#(0pt)B}
\endgroup
```

A ——— ○ ——— ● ——— B
A ——— ○ ——— ● ——— B

---
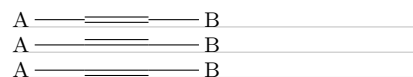
5   Shifted double bonds

All double bonds are made up of two line segments, and these segments are drawn on either side of the imaginary line along which a single bond would be drawn. It is possible to shift a double bond so that one of the line segments lies on the imaginary line. The other segment is then shifted above or below the bond. Actually, it is more correct to say "left" or "right" of the imaginary line, as the bond is traversed in the direction of drawing.

To shift the bond to the left, write "=^" and to shift it to the right, write "=_":

Shifted double bonds

```
\chemfig{A-=-B}\par
\chemfig{A-=^-B}\par
\chemfig{A-=_-B}
```
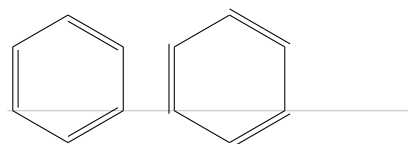
A ════ B
A ──── B
A ──── B

In rings, double bonds are automatically shifted to the left. However, they can be shifted to the right by specifying it with "=_":

Shifted double bonds and rings

```
\chemfig{*6(-=-=-=)}\qquad
\chemfig{*6(-=_-=_-=_)}
```

Shifted bonds are particularly useful in drawing skeleton diagrams of molecules consisting of carbon chains with double bonds. They give a continuous zig-zag path, whereas the path will be broken with regular double bonds:

Shifted bonds and skeleton diagrams

```
\chemfig{-[:30]=[:-30]-[:30]=[:-30]-[:30]}\par
\chemfig{-[:30]=^[:-30]-[:30]=^[:-30]-[:30]}\par
\chemfig{-[:30]=_[:-30]-[:30]=_[:-30]-[:30]}
```

---

6   Delocalized double bonds

It is sometimes necessary to draw a double bond so that one line segment is dashed while the other is solid[9]. This feature is not hard-coded into ChemFig; instead ti*kz*, its "decorations" library and its programmable styles make it possible.

First of all, after having loaded the "decorations" library by putting \usetikzlibrary{decorations} in the preamble, a decoration named "ddbond" (for Dashed Double Bond) is defined (lines 1 to 14) and then two ti*kz* styles called "lddbond" and "rddbond"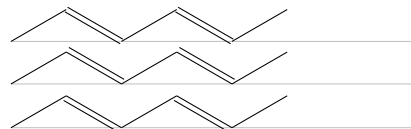 (lines 15 and 16). For the former, the dashed line segment is on the left of the solid middle segment; for the latter it is on the right. These styles can be called with the fifth optional argument of bonds:

Custom decorations

```
\pgfdeclaredecoration{ddbond}{initial}
{
  \state{initial}[width=4pt]
  {
    \pgfpathlineto{\pgfpoint{4pt}{0pt}}
    \pgfpathmoveto{\pgfpoint{2pt}{2pt}}
    \pgfpathlineto{\pgfpoint{4pt}{2pt}}
    \pgfpathmoveto{\pgfpoint{4pt}{0pt}}
  }
  \state{final}
  {
    \pgfpathlineto{\pgfpointdecoratedpathlast}
  }
}
\tikzset{lddbond/.style={decorate,decoration=ddbond}}
\tikzset{rddbond/.style={decorate,decoration={ddbond,mirror}}}
\setatomsep{4em}
\chemfig{[:-30]R-C-[::60]C(-[::60,,,,rddbond]O)-[,,,,lddbond]N(-[::-60]H)-[::60]C-R}
```


---
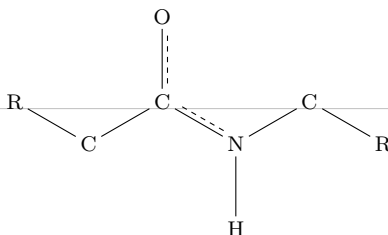
7   Saving a sub-molecule

ChemFig is capable of saving a <code> as an alias for reuse in a more compact form in the code of a molecule. This is particularly useful when the <code> appears several times.

To do this, one gives the command

$$\text{\textbackslash definesubmol\{<name>\}\{<code>\}}$$

which saves the <code> for recall in the code of the molecule via the shortcut "!{name}". This <name> can be:

- a sequence of characters: all the alphanumeric characters able to be between \csname and \endcsname are accepted;

---
[9]Thanks to Andreas Bröermann who suggested this feature and who gave me the solution to this problem.

- a control sequence.

In all cases, if the alias is already defined you should not overwrite it with a new definition using \definesubmol. A warning will be issued to the user that the old alias will be overwritten by the new one. To override the definition of an alias made previously, use:

$$\redefinesubmol\{<name>\}\{<code>\}$$

Here is a code which draws the pentane molecule. An alias "xy" was defined beforehand for the code CH_2:


```
\definesubmol{xy}{CH_2}
\chemfig{H_3C-!{xy}-!{xy}-!{xy}-CH_3}
```
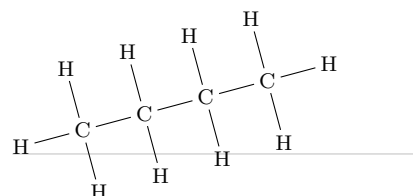
In this case the technique is not very interesting because "!{xy}" is just as long to type as the code it replaces.

But in certain cases, this feature saves a lot of space in the code of the molecule and increases readability. In the following example, we draw the complete structural diagram of butane. We will define an alias with the control sequence "\xx" for the sub-molecule CH_2. If we use only relative angles, it is possible to rotate the entire molecule to any given angle by using the optional global angle parameter which specifies the default bond angle of the main molecule. It is set to 15° here:


```
\definesubmol\xx{C(-[::+90]H)(-[::-90]H)}
\chemfig{[:15]H-!\xx-!\xx-!\xx-!\xx-H}
```

The \definesubmol command takes an optional argument; its syntax is as follows:
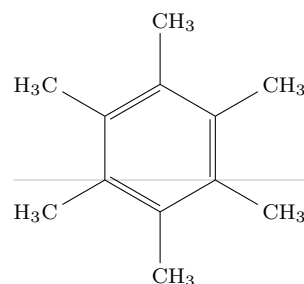
$$\definesubmol\{<name>\}[<code1>]\{code2\}$$

When the optional argument is present, the alias "!<name>" will be replaced by <code1> if the bond which arrives at the alias comes from the right, i.e., if the angle which the arriving bond makes is between but is not equal to 90° and 270°. For all the other cases where the bond arrives from the left of vertically, the alias will be replaced by <code2>.

We will define a control sequence \Me pour "methyl" so that the alias "!\Me" will be replaced by "H_3C" when the bond arrives from the right and by "CH_3" when it arrives from the left. We can observe in the example that with this alias we need no longer worry about the angle:


```
\definesubmol\Me[H_3C]{CH_3}
\chemfig{*6((-!\Me)=(-!\Me)-(-!\Me)=(-!\Me)-(-!\Me)=(-!\Me)-)}
```

8   Decorations

8.1   Lewis diagrams

The macro \lewis allows placement of pairs of electrons, of single electrons, or of empty slots. This syntax is used:

$$\lewis\{<n1><n2>...<ni>,<atom>\}$$

where the <n1>…<ni> represent the desired positions (in multiples of 45°) around the <atom>. These whole numbers must be between 0 and 7.

This command can also be used inside the argument of \chemfig:
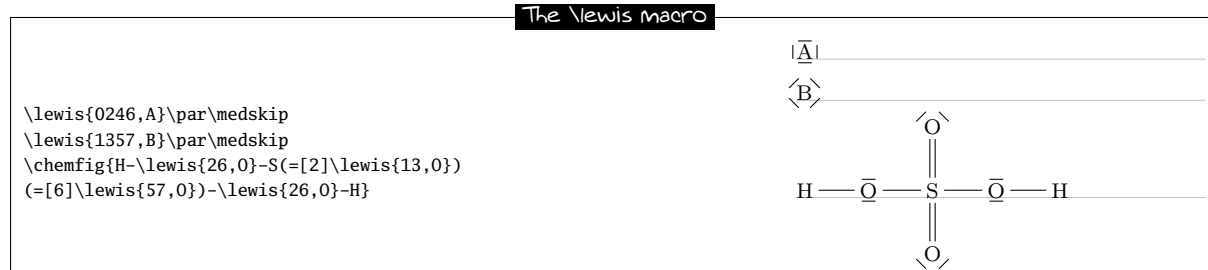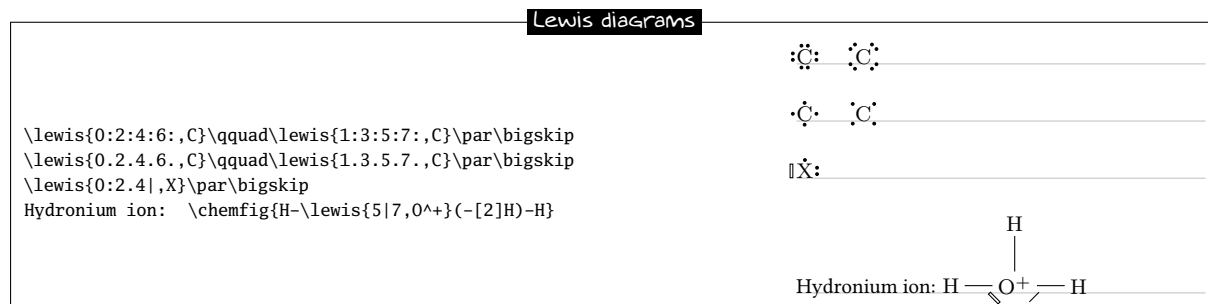
```
\lewis{0246,A}\par\medskip
\lewis{1357,B}\par\medskip
\chemfig{H-\lewis{26,O}-S(=[2]\lewis{13,O})
(=[6]\lewis{57,O})-\lewis{26,O}-H}
```
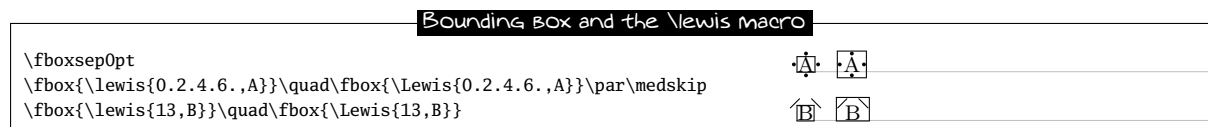
If one wishes to draw two electrons instead of a line, follow the integer with a ":". If one wishes to draw a single electron, follow it with a ".". To draw a lacuna, follow it with a "|":

```
\lewis{0:2:4:6:,C}\qquad\lewis{1:3:5:7:,C}\par\bigskip
\lewis{0.2.4.6.,C}\qquad\lewis{1.3.5.7.,C}\par\bigskip
\lewis{0:2:4|,X}\par\bigskip
Hydronium ion:  \chemfig{H-\lewis{5|7,O^+}(-[2]H)-H}
```

All the decorations drawn by \lewis are not included in the bounding box of the atom; they are drawn afterwards. A consequence of this is seen in the two examples above, where the frame does not appear to be properly fitted to the drawing of the molecule, which extends downward slightly. This will be seen more often in this the "Decorations" chapter, which presents commands which do not change the bounding box.

The \Lewis macro works the same way as \lewis but decorations are taken into account in the bounding box.

This can be seen more clearly by drawing an \fbox around decorated atoms:

```
\fboxsep0pt
\fbox{\lewis{0.2.4.6.,A}}\quad\fbox{\Lewis{0.2.4.6.,A}}\par\medskip
\fbox{\lewis{13,B}}\quad\fbox{\Lewis{13,B}}
```
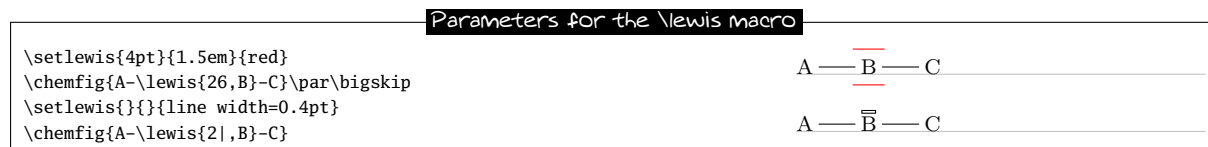
Several parameters can be set with the help of the macro

$$\setlewis[dim1]\{<dim2>\}\{<dim3>\}\{<tikz code>\}$$

If an argument is empty, it takes its default value.

- <dim1> is the width of the rectangle which represents the empty slot obtained whith the character "|";
- <dim2> is the distance between the bounding box and the decoration. It is 0.2ex by default;
- <dim3> is the length of the line segment representing a pair of electrons. It is 1.5ex by default;
- <code tikz> is code which is passed directly to ti*kz*. This code is empty by default.

```
\setlewis{4pt}{1.5em}{red}
\chemfig{A-\lewis{26,B}-C}\par\bigskip
\setlewis{}{}{line width=0.4pt}
\chemfig{A-\lewis{2|,B}-C}
```

A problem sometimes occurs with the decorations of Lewis in the odd directions. In the example below with the atom "O", the decoration in position 1 seems farther from the atom than the decoration in position 4:

```
\huge
\Lewis{1|4|,O}
```

However, it is not the case as shown below by drawing the bounding box of the atom:

```
                          Optical illusion
\huge
\fboxsep0pt
\def\printatom#1{\fbox{$\mathrm{#1}$}}
\Lewis{1|4|,O}
```
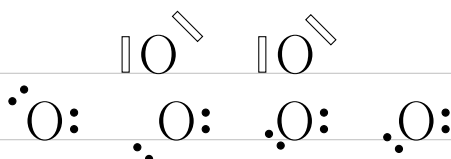
The impression of greater distance is due to the shape of the letter "O" which is farther from the one of the bounding box in the corners, that is to say, in odd directions.

To move nearer (or farther) the Lewis drawings in odd directions, the \lewis and \Lewis accept an optional argument that contains a factor which multiplies the gap between the bounding box and decoration Lewis set with the \setlewis command. For the letter "O", it semms that 0.5 is the appropriated value:

```
                      Optional argument of \lewis
\huge
\Lewis{1|4|,O}\quad \Lewis[0.5]{1|4|,O}

\Lewis{0:3:,O}\quad  \Lewis[2]{0:5:,O}\quad  \Lewis[0]{0:5:,O}\quad  \Lewis[0.5]{0:5:,O}
```
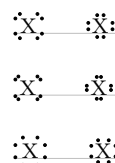
Finally, the \setlewisdist macro sets the distance between the two disks representing a pair of electrons. The argument of this macro must be valid distance for TeX or, if it is empty sets the default value is 0.3em:

```
                      Distance between electrons
\Lewis{1:3:5:7:,X}\qquad\Lewis{0:2:4:6:,X}\bigskip

\setlewisdist{0.2em}\Lewis{1:3:5:7:,X}\qquad\Lewis{0:2:4:6:,X}\bigskip

\setlewisdist{4pt}\Lewis{1:3:5:7:,X}\qquad\Lewis{0:2:4:6:,X}
```

8.2   Stacking characters

The macros

$$\text{\chemabove[<dim>]\{<code>\}\{<stuff>\}}$$

and

$$\text{\chembelow[<dim>]\{<code>\}\{<stuff>\}}$$

place the <stuff> above and below the <code> respectively at a vertical distance <dim>, without changing the bounding box of <code>. The optional argument allows, if written, to specify this dimension at each call. If the optional argument is not used, a default size will be taken: its value is 1.5pt but it can be modified with the macro

$$\text{\setstacksep\{<dimension>\}}$$
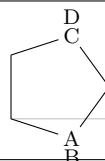
These commands are independent of the macro \chemfig and can be used either inside or outside its argument.

They are especially useful in rings, if care is taken to put braces around the letters A, B, C and D in order to prevent ChemFig from starting a new atom on these letters:

```
                         Staking in rings
\chemfig{*5(-\chembelow{A}{B}--\chemabove{C}{D}--)}
```

They are sometimes useful for placing pseudo-exponents which do not change the bounding box of the atoms, so that the bonds do not end up being too short:

Hydronium ion

`\chemfig{H-\chemabove{\lewis{5|7,0}}{\quad\scriptstyle+}(-[2]H)-H}`

Les commandes \Chemabove et \Chembelow fonctionnent de la même façon sauf que la boîte englobante *tient compte* du <matériel> placé au dessus ou au dessous.

8.3   Chemical reactions

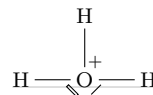To write chemical reactions, ChemFig provides a command for the signs and a command for the arrows.

The command \chemsign[<dim>]<sign> typesets the <sign>, which is surrounded on both sides by an unbreakable horizontal space <dim> defaulting to 0.5em.

The command \chemrel[<arg1>][<arg2>]{<arrow code>} draws an arrow where the optional arguments <arg1> and <arg2> are placed above and below the arrow respectively, without modifying its bounding box. The <arrow code> is passed directly to ti*kz* except when it involves "<>", which allows two arrows to be drawn one above the other.

Types of arrows

```
A\chemrel{->}B\par
A\chemrel{<-}B\par
A\chemrel{<->}B\par
A\chemrel{<>}B\par
A\chemrel{->,red,thick}B
```
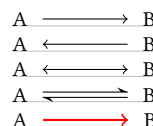
The command \setchemrel{<dim1>}{<dim2>}{<dim3>} allows settings of the dimensions used in drawing the arrow:

- <dim1> is the vertical spacing between the arrow and the optional text above and/or below it. If the argument is empty, it defaults to 2pt;
- <dim2> is the unbreakable horizontal space inserted before and after the arrow. If the argument is empty, it defaults to 0.7em;
- <dim3> is the length of the arrow. If the argument is empty, if defaults to 4em.

Text above arrows

```
\setchemrel{1pt}{}{6em}
A\chemrel[\footnotesize above][\tiny below]{<->}B
```
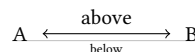
Here is an example of a chemical reaction:

Chemical reaction

```
\setchemrel{0pt}{1.2em}{6em}
\chemfig{**6(------)}\chemsign+\chemfig{H_3C-Cl}
\chemrel[\itshape\footnotesize Catalyst]{->}
\chemfig{**6(---(-)---)}\chemsign+\chemfig{H-Cl}
```
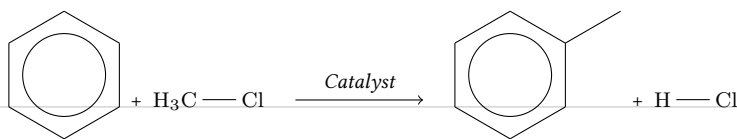
And another:

Synthesis of phenylacetylene

```
\chemfig{*6(-=*6(-(-Br)-(-Br))-=-=)}
\chemrel[\chemfig{NaNH_2}][\chemfig{NH_3}]{->}
\chemfig{*6(-=(-~)-=-=)}
```
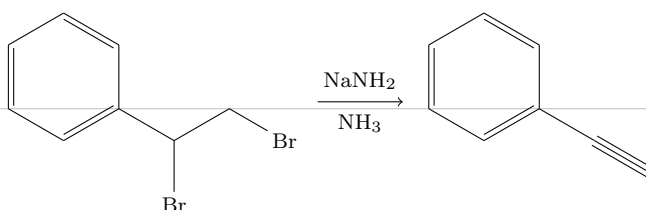
9   Using \chemfig in the tikzpicture environment

It is possible to call the \chemfig inside a tikzpicture environment:

```
\begin{tikzpicture}[help lines/.style={thin,draw=black!50}]
\draw[help lines] (0,0) grid (4,4);
\draw(0,0) -- (2,1);
\draw(2,2) circle (0.5);
\node at (1,3) {\chemfig{A=B-[:30]C}};
\node[draw,red,anchor=base] at(3,2){\chemfig{X>[2,,,,blue]Y}};
\end{tikzpicture}
```

10   Beyond chemistry
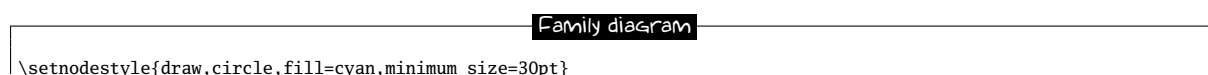
At heart ChemFig is a tool for drawing graphs, and this tool has been programmed to adapt it for chemistry. In some ways it is possible to return ChemFig to its roots to draw organization charts or other diagrams represented by graphs.
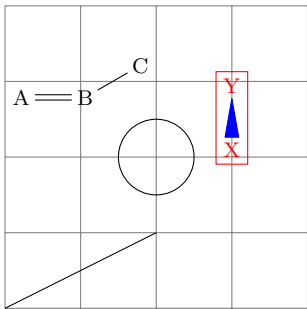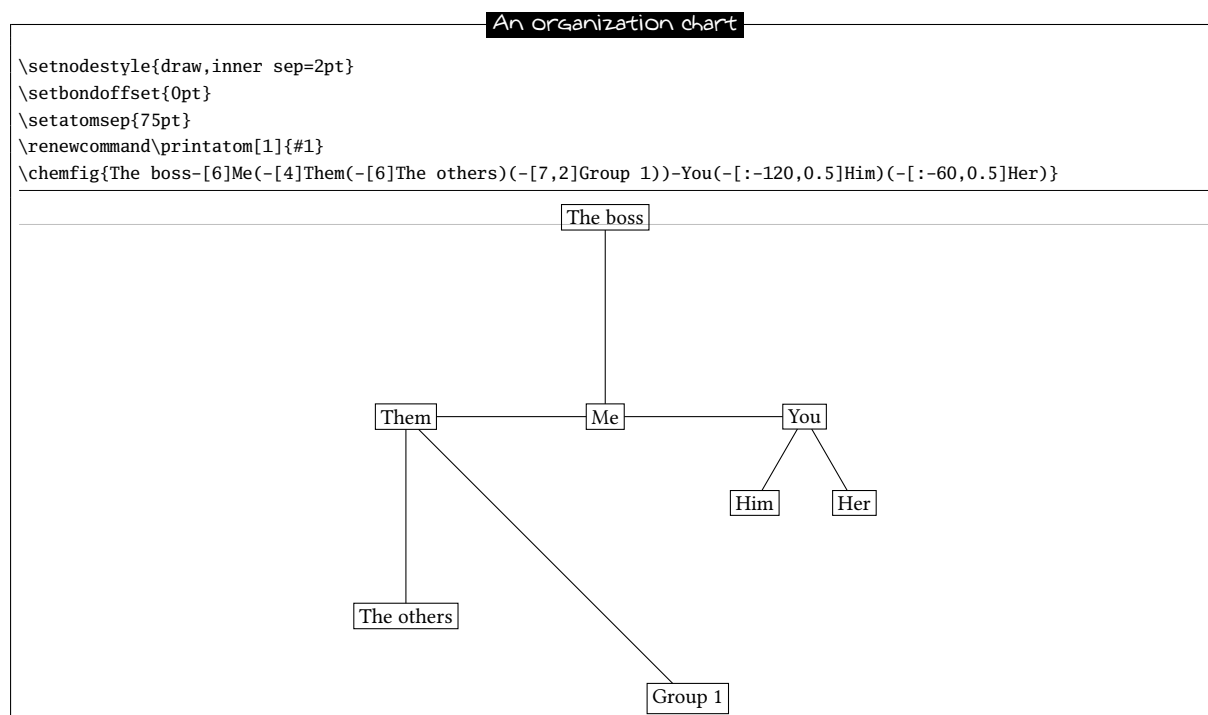
Each atom is contained in a ti*kz* node. By default these nodes have an "inner sep" and an "outer sep" equal to 0pt. They are rectangular as seen on page 6. These defaults can be overwritten with the macro \setnodestyle, the argument of which is passed to ti*kz* and specifies the style of the nodes containing the atoms.

In this example we specify only "draw,inner sep=2pt", which has the effect of drawing the outline of the nodes and separating the outline and node contents by 2pt. We also specify \setbondoffset{0pt}\setbondoffset so that the bonds touch the edges of the nodes. The interatomic spacing is increased to 75pt. Finally, the command \printatom is made as simple as possible so that math mode is no longer used and spaces are thus preserved.

```
\setnodestyle{draw,inner sep=2pt}
\setbondoffset{0pt}
\setatomsep{75pt}
\renewcommand\printatom[1]{#1}
\chemfig{The boss-[6]Me(-[4]Them(-[6]The others)(-[7,2]Group 1))-You(-[:-120,0.5]Him)(-[:-60,0.5]Her)}
```
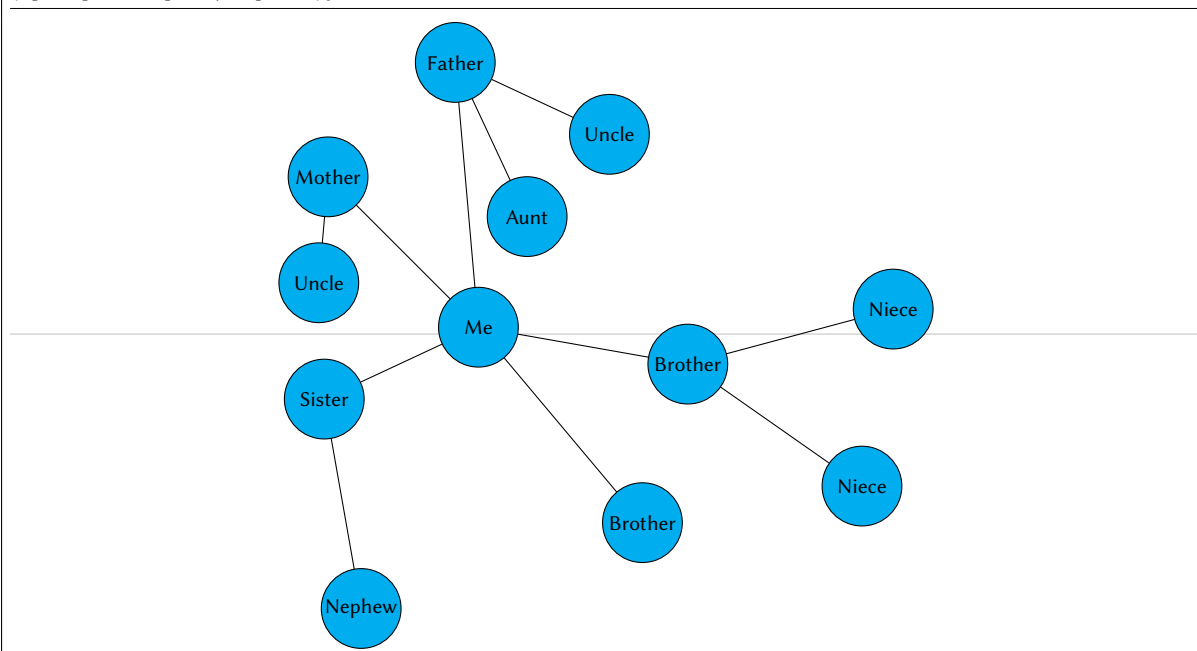
Here is another organization chart where the nodes are circular and coloured cyan:

```
\setnodestyle{draw,circle,fill=cyan,minimum size=30pt}
```

```
\setbondoffset{0pt}
\setatomsep{80pt}
\renewcommand\printatom[1]{\textsf{#1}}
\chemfig{Me(-[:-50,1.2]Brother)(-[:-10]Brother(-[:15]Niece)(-[:-35]Niece))
(-[:-155,0.8]Sister-[:-80]Nephew)(-[:95,1.25]Father(-[:-25,0.8]Uncle)(-[:-65,0.8]Aunt))
(-[:135]Mother-[:-95,0.5]Uncle)}
```


11   Annotated examples

In this chapter, several molecules will be drawn, putting into use the methods previously described. The aim here is to show a logical order for putting together a molecule so that the user unfamiliar with ChemFig will learn how to construct complex molecules. The construction steps will be shown to help with this learning process.

In addition, several possibilities — some intuitive and others less so — will be shown which give the same graphical results, with the objective being to show that ChemFig allows some flexibility in encoding molecules. One can see how each is put together and adopt the methods with which one is most comfortable.

11.1   Ethanal

Here we will draw the ethanal (or acetaldehyde) molecule:
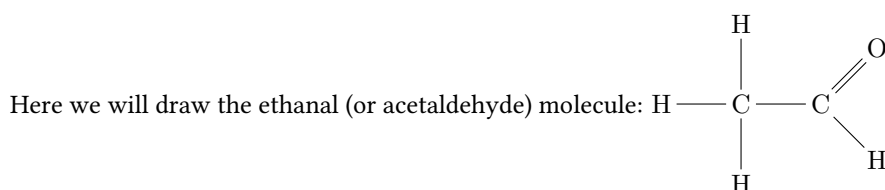
The best method for non-cyclic molecules is to select the longest chain. Here one could take "H–C–C=O" for example. The bond C=O is tilted to 45° by using the predefined angle "[1]". This give a "backbone" of the molecule to which the branches merely have to be added:

Backbone of ethanal

```
\chemfig{H-C-C=[1]O}
```
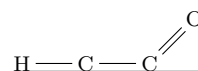
The three hydrogen atoms still have to placed at the correct orientation with the help of predefined angles. The first is at 90° with the branch "(-^{2}H)", the second at 270° with "(-[6H])", and the one on the right at 315° with "(-[7]H)":

Ethanal

`\chemfig{H-C(-[2]H)(-[6]H)-C(-[7]H)=[1]O}`

11.2 2-amino-4-oxohexanoic acid

Here is the molecule to be drawn:
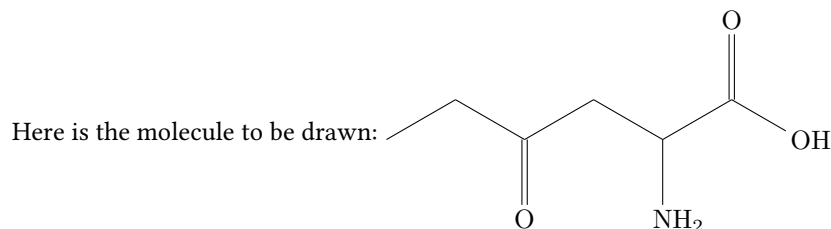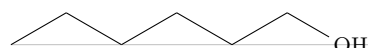
As is often the case for most molecules, there are several methods and for each several different ways of getting the result. Here we will look at four different methods.

11.2.1 Absolute angles

We will first of all draw the central chain with absolute angles. We set the default angle to $+30°$ with the optional argument, and so only the descending bonds need to have their absolute angle set to $-30°$:

Backbone (absolute angles)
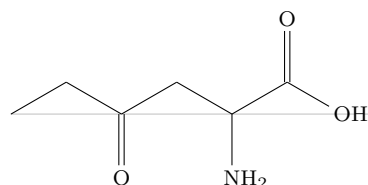
`\chemfig{[:30]--[:-30]--[:-30]--[:-30]OH}`

The branches "(=[6]O)", "(-[6]NH_2)" and "(=[2]O)" still have to be added to the correct vertices:
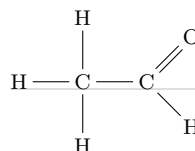
Molecule (absolute angles)

`\chemfig{[:30]--[:-30](=[6]O)--[:-30](-[6]NH_2)-(=[2]O)-[:-30]OH}`

11.2.2 Relative angles

A more general approach uses only relative angles, in this way:

Structure (relative angles)
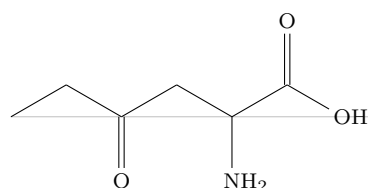
`\chemfig{[:30]--[::-60]--[::-60]--[::-60]OH}`

then

Molecule (relative angles)

`\chemfig{[:30]--[::-60](=[::-60]O)--[::-60](-[::-60]NH_2)`
`-(=[::60]O)-[::-60]OH}`

11.2.3   Ring

Since the angles between the bonds are 120°, it is possible to use a 6-ring, although this method is less natural. Here we take advantage of the fact that a ring can be left unfinished. The ring must be rotated 120° so that the first vertex is to the south-east of the ring:

Backbone (ring)
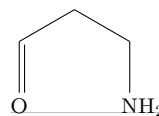
`\chemfig{[:120]NH_2*6(---=O)}`

Now the branches must be added to the right vertices:

Molecule (ring)

`\chemfig{[:120]NH_2*6(-(=[::60]O)-[::-60]OH)--(--[::60])=O)}`
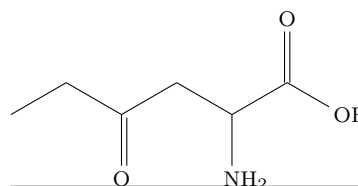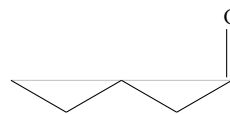
11.2.4   Nested rings

Delving deeper into the ring method, we can also consider nesting incomplete 6-rings. We could start with this backbone:

Backbone (nested rings)

`\chemfig{*6(--*6(--=O))}`

And then add the bonds which leave the vertices of these rings. There are no angles to worry about because the bonds leaving the rings are the bisectors of the sides of the ring, exactly what we want here:

Molecule (nested rings)

`\chemfig{*6((-)-(=O)-*6(-(-NH_2)-(-OH)=O))}`

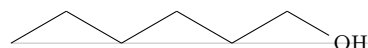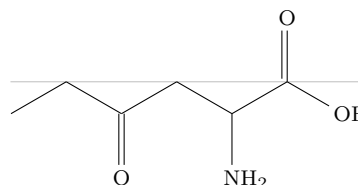A close look shows that the second line segment of the double bond to the oxygen atom is *inside* the incomplete 6-ring[10] Despite its brevity, this code does not give a perfect drawing. This can of course be corrected by adding a little to the code:

Molecule (corrected nested rings)

`\chemfig{*6((-)-(=O)-*6(-(-NH_2)-(-OH)(=[::60]O)))}`
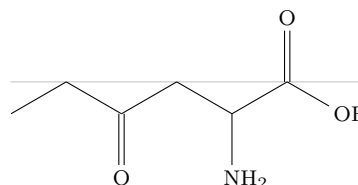
11.3   Glucose

The goal here is to represent the glucose molecule according to several different conventions.

---

[10]This was also true for the preceding method with one ring.

11.3.1 Skeleton diagram

The code here looks like that of 2-amino-4-oxohexanoic acid. This gives almost the same structure with absolute angles, except here the default angle is $-30°$:

Backbone

`\chemfig{[:-30]HO--[:30]--[:30]--[:30]-H}`

Adding the branches is no problem. We use predefined absolute angles:

Glucose, skeleton diagram

`\chemfig{[:-30]HO--[:30](<[2]OH)-(<:[6]OH)`
`-[:30](<:[2]OH)-(<:[6]OH)-[:30](=[2]O)-H}`
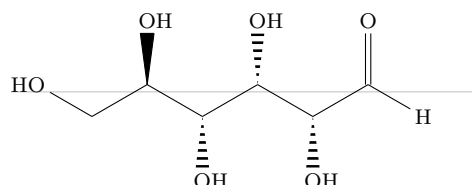
11.3.2 Fisher projection
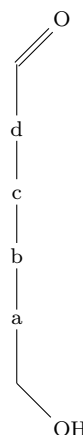
The goal is to get the molecule below:
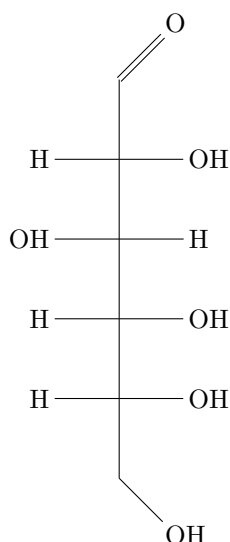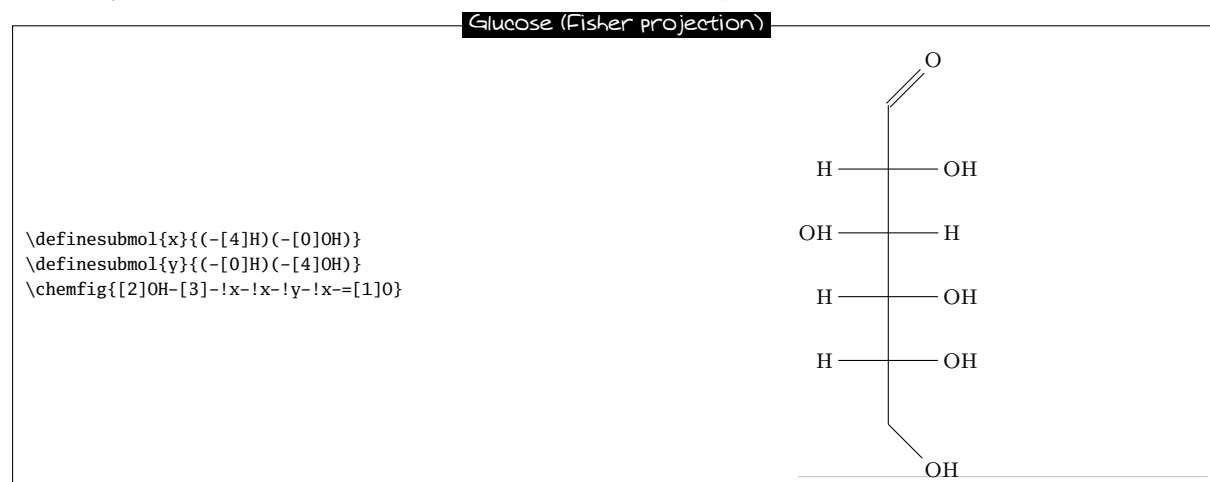
The idea is to begin to draw the longest chain vertically by giving a default angle of "[2]". Here is the skeleton, where we have added lower case letters at the end of each vertical bond:

Skeleton

`\chemfig{[2]OH-[3]-a-b-c-d-=[1]O}`
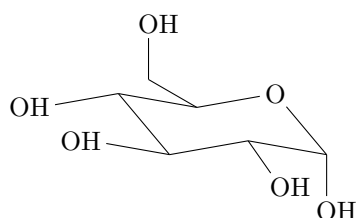
Next we define two aliases for the horizontal bonds and the atoms at their ends. Lets choose "x" which we will put in place of the lower case a, c and d, and "y" which will replace the letter c. Since these alias are just one character, we do not need braces and can write "!x" instead of "!{x}":

```
\definesubmol{x}{(-[4]H)(-[0]OH)}
\definesubmol{y}{(-[0]H)(-[4]OH)}
\chemfig{[2]OH-[3]-!x-!x-!y-!x-=[1]O}
```
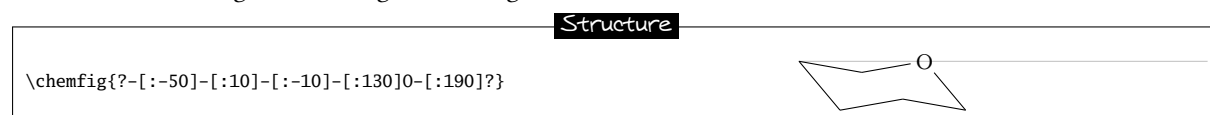
11.3.3 "Chair" representation
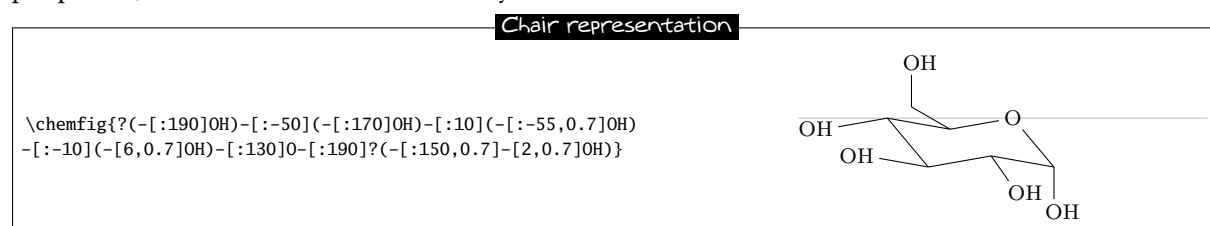
We will depict the α-D-glucose molecule:

To do this, we will first of all draw five sides of the chair and link the first vertex to the last with a hook "?". We will use the following absolute angles, running counterclockwise: $-50°$, $10°$, $-10°$, $130°$, $190°$.

```
\chemfig{?-[:-50]-[:10]-[:-10]-[:130]O-[:190]?}
```

Now we simply add the branches inside parentheses. The angles are chosen to give the best impression of perspective, and some bonds are shortened by a factor of 0.7:

```
\chemfig{?(-[:190]OH)-[:-50](-[:170]OH)-[:10](-[:-55,0.7]OH)
-[:-10](-[6,0.7]OH)-[:130]O-[:190]?(-[:150,0.7]-[2,0.7]OH)}
```

11.3.4 Haworth projection
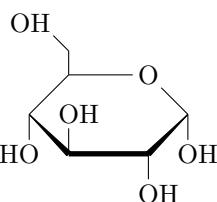
The goal is to depict this D-glucopyranose molecule:

First of all we will choose the longest chain, which starts at the "HO" group on the left and continues through fives sides of the ring. The ring will be closed with a hook. For the vertical bond which leaves from the first "HO"

group, we need to specify that it will leave from the second atom using the optional argument. Furthermore, it will be shortened with a coefficient of 0.5. Its optional argument will thus be "[2,0.5,2]".

Next, to give the impression of perspective to the ring, the diagonal bonds will be shortened by a coefficient of 0.7. For the bold diagonal lines we will use Cram bonds, having redefined the base of the triangles to be 2pt. The bold horizontal bond needs to be drawn with a thickness of 2pt, and so its optional argument will be "[0,,,,line width=2pt]". Here is the skeleton of the molecule:

Structure

```
\setcrambond{2pt}{}{}
\chemfig{HO-[2,0.5,2]?<[7,0.7]-[,,,,
line width=2pt]>[1,0.7]-[3,0.7]O-[4]?}
```
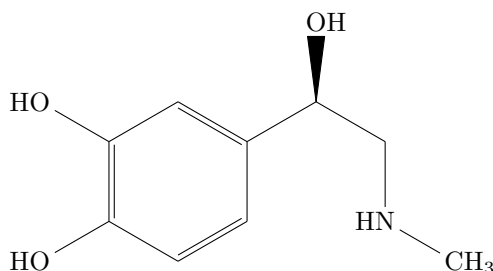
All that needs to be done now is to add the branches at the correct places, giving the right absolute angles and sometimes reducing the length to better give the illusion of perspective:

Haworth projection

```
\setcrambond{2pt}{}{}
\chemfig{HO-[2,0.5,2]?<[7,0.7](-[2,0.5]OH)-[,,,,
line width=2pt](-[6,0.5]OH)>[1,0.7](-[6,0.5]OH)-[3,0.7]
O-[4]?(-[2,0.3]-[3,0.5]OH)}
```
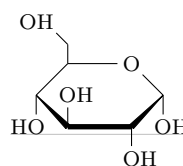
11.4   Adrenaline

We want to draw the adrenaline molecule:
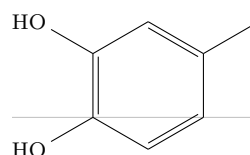
We are going to use two different methods.

11.4.1   Using one ring

First of all, we start with a 6-ring and we draw the start of the branches which leave it:

Skeleton of adrenaline
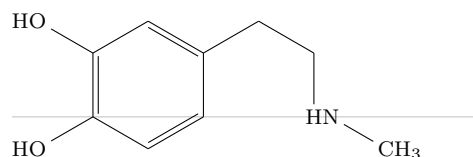
```
\chemfig{*6((-HO)-=-(-)=-(-HO)=)}
```

The branch on the right still needs to be completed using, for example, relative angles:

Adrenaline, step two

```
\chemfig{*6((-HO)-=-(--[::-60]-[::-60]
HN-[::+60]CH_3)=-(-HO)=)}
```

Then we need to add a Cram-bonded OH and indicate that the bond which arrives at "HN" does so on the second atom, i.e., "N". We use the fourth optional argument of the bond:

Adrenaline

```
\chemfig{*6((-HO)-=-(-(<[::60]OH)-[::-60]-[::-60,,,2]
HN-[::+60]CH_3)=-(-HO)=)}
```
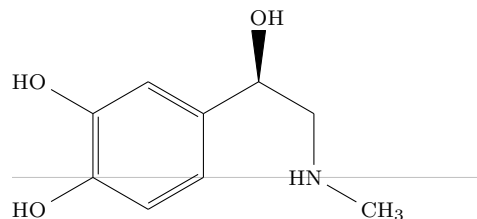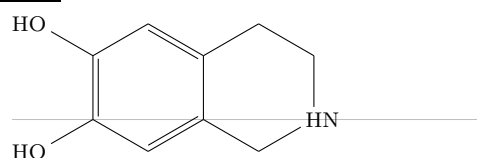
11.4.2   Using two rings

This method is less natural, but the goal is to show here how to make a bond invisible.

We could improve this code by considering that the drawing of the adrenaline molecule is made of two 6-rings adjacent to each other:

Adrenaline, two-ring skeleton

```
\chemfig{*6((-HO)-=*6(--HN---)-=-(-HO)=)}
```

Now the first two bonds of the ring on the right need to be made invisible. To do this we use the argument that is passed to ti*k*z, specifying "draw=none". These bonds will thus have this code: "-[,,,,,draw=none]". To keep the code readable, we define an alias named "&" for these bonds:

Adrenaline, step two
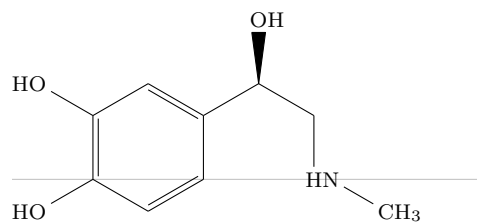
```
\definesubmol{&}{-[,,,,draw=none]}
\chemfig{*6((-HO)-=*6(!&!&HN---)-=-(-HO)=)}
```
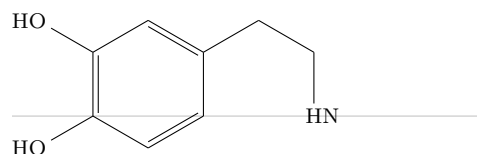
The rest becomes easy; just add the branches to the right vertices:

Adrenaline, step three

```
\definesubmol{&}{-[,,,,draw=none]}
\chemfig{*6((-HO)-=*6(!&!&HN(-CH_3)--(<OH)-)-=-(-HO)=)}
```

To finish, we specify that the bonds that *arrive at and leave from* "HN" must do so at the second atom. We therefore define another alias for the invisible bond which arrives at "HN":
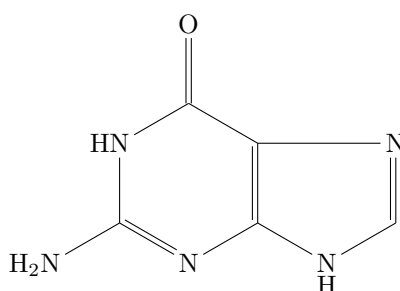
Adrenaline

```
\definesubmol{&}{-[,,,,draw=none]}
\definesubmol{&&}{-[,,,2,draw=none]}
\chemfig{*6((-HO)-=*6(!&!{&&}HN(-CH_3)-[,,2]-(<OH)-)-=-(-HO)=)}
```
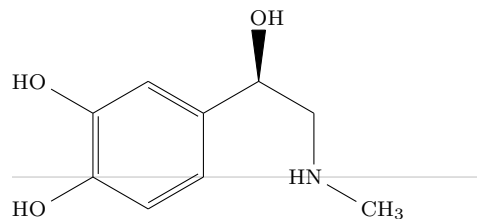
11.5   Guanine
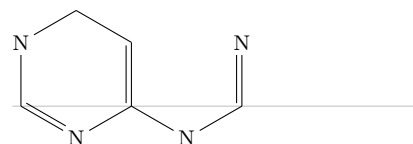
We will draw the guanine molecule:

First of all, let's begin by drawing the nested rings, putting just the nitrogen atoms at the vertices:

Guanine, skeleton

`\chemfig{*6(=N-*6(-N-=N)=--N-)}`

Then we can draw the horizontal bond in the right ring with a hook. We will also place a hydrogen atom under the nitrogen atom of the 5-ring with the command `\chembelow{N}{H}`. We also need to write "HN" instead of "N" at the vertex at the upper left of the molecule:

Guanine, step two

`\chemfig{*6(=N-*6(-\chembelow{N}{H}-=N?)=?--HN-)}`
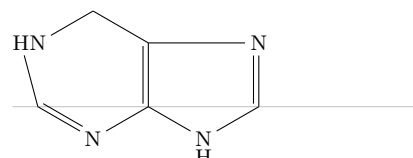
We note that one bond leaves from the wrong atom[11]! The automatic calculation mechanism must be corrected so that the bond leaves from the second atom "N" instead of the first. To do this we give an optional argument for the last bond of the first 6-ring "[,,2]":

Guanine, step three
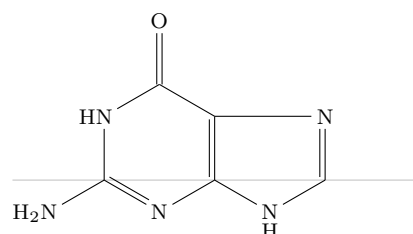
`\chemfig{*6(=N-*6(-\chembelow{N}{H}-=N?)=?--HN-[,,2])}`
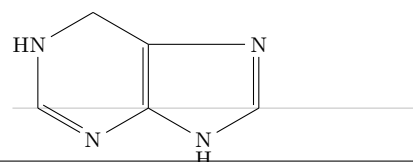
Simply add the branches to the right vertices. Note especially the branch leaving the first vertex of the first 6-ring "(-N_2N)":

Guanine

`\chemfig{*6((-H_2N)=N-*6(-\chembelow{N}{H}-=N?)=?-(=O)-HN-[,,2])}`

We could also draw the same molecule with a regular 5-ring, as is sometimes done:

---

[11]This seems illogical because the angle of the bond from the HN group toward the first vertex lies between $-90°$ and $90°$; ChemFig should therefore leave from the second atom. To explain this contradiction, one must know that in rings, the last bond always links the last vertex to the first, ignoring the *calculated theoretical* angle of this bond (which here is $-90°$). ChemFig uses this theoretical angle to determine the departure and arrival atoms, but does not use it to draw the bond because the two ends are already defined. The departure atom for the last bond is thus the first atom.

Guanine with 5-ring

`\chemfig{*6((-H_2N)=N-*5(-\chembelow{N}{H}-=N-)=-(=O)-HN-[,,2])}`
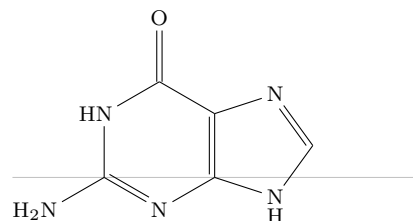
12   How to …

12.1   Write a colored atom

Since the package xcolor is loaded by ti*kz*, itself loaded by ChemFig, we can write color commands in the code of a molecule, mainly `\color` and `\textcolor`. The atoms are displayed in ti*kz* nodes which behaves like boxes of TeX and it is as if these atoms were put in a group. Therefore, the color change remains local to the atom.

Colors

`\chemfig{C\color{blue}H_3-C(=[1]O)-[7]O\color{red}H}`
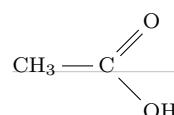
This code does not work, because of the rule used to separate atoms: here, the first atom sarts at "`C`" and spreads to the next uppercase letter. Therefore, this atom is "`C\color{blue}`" and the color change occurs at the end of atom and has no effect. We need to force ChemFig to cut the first atom just after "`C`" with the character "`|`" and then include `\color{blue}H_3` between braces so that ChemFig does not stop the atom 2 before the uppercase "`H`" which would leave the color change alone and therefore ineffective in an atom:

Colors

`\chemfig{C|{\color{blue}H_3}-C(=[1]O)-[7]O|{\color{red}H}}`
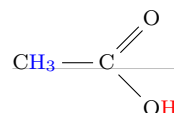
The same effect can be obtained with `\textcolor`:

Colors

`\chemfig{C|\textcolor{blue}{H_3}-C(=[1]O)-[7]O|\textcolor{red}{H}}`
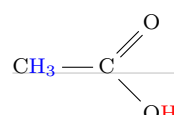
The main disadvantage is that we have to do the same for every atom that need to be colored, even if they are contiguous.

12.2   Add a superscript without modifying a bond

Adding a charge to an atom with a mathematical exponent implies that the box (and therefore the ti*kz* node) containing the atom has its dimensions modified. It has no importance when the atom is trailing but the alignment may be compromised if a bond is attached to the atom. The first reaction is to put the charge in a box with no width and therefore use the command `\rlap`[12], which often gives good results. We see here that with `\rlap`, the horizontal alignment of atoms is preserved:

Charge and bond

`\chemfig{A^+-[2]B}`
`\qquad`
`\chemfig{A\rlap{${}^+$}-[2]B}`

---
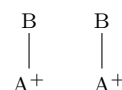
[12]If you have to put the charge at the left of the atom, you must use the command `\llap`.

If you want to use the command \oplus which displays "⊕", some could find that the charge is too low: A⊕. In that case, why not use \chemabove to put as precisely as you will, both vertically and horizontally the charge:

Charge et \chemabove

```
\chemfig{\chemabove[0.5pt]{A}{\scriptstyle\hspace{3.5mm}\oplus}-[2]B}
\qquad
\chemfig{{\chemabove[-0.5pt]{A}{\scriptstyle\hspace{3.5mm}\oplus}}-[2]B}
```
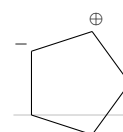
We notice an additional level of braces for the second molecule. Indeed, as we specify "-0.5pt" for the optional argument of \chemabove to lower the charge, it is necessary to prevent ChemFig to understand the sign "-" as a single bond.

To add a load near the vertex of a cycle , the best method is to attach an invisible bond to this vertex, which is done here with \definesubmol with a bond with a length coefficient equal to 0.2:

Charges and cycles

```
\definesubmol\nobond{-[,0.2,,,draw=none]}
\chemfig{*5(---(!\nobond\scriptstyle\oplus)-(!\nobond\scriptstyle{-})-)}
```

12.3   Draw a curve bond

We have already seen that with the ti*k*z library "decorations.pathmorphing", we can draw a wavy bond:

Wavy bond
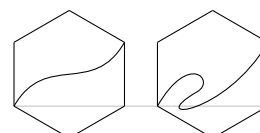
```
\chemfig{A-[,3,,,decorate,decoration=snake]B}

\chemfig{A-[,3,,,decorate,decoration={snake,amplitude=1.5mm,
    segment length=2.5mm}]B}
```

For more flexibility, you can also define nodes using the character "@" and reuse these nodes after the molecule has been drawn to connect them with a curved line using \chemmove:

Curved bonds

```
\chemfig{@{a}A-[,,,,draw=none]@{b}B}
\chemmove{\draw[-](a)..controls +(45:7mm) and +(225:7mm)..(b);}
\bigskip

\chemfig{*6(@{a}---@{b}---)}
\chemmove{\draw[-](a)..controls +(60:3em) and +(240:3em)..(b);}
\quad
\chemfig{*6(@{a}---@{b}---)}
\chemmove{\draw[-](a)..controls +(60:3em) and +(30:1em)..
    ++(20:2em) ..controls +(210:3em) and +(-120:4em) ..(b);}
```
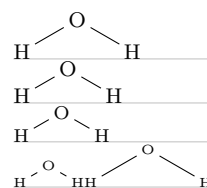
12.4   Modify the size of a molecule

Two parameters determine the default size of a molecule: the font size and the parameter of \setatomsep which is 3em the default. We can modify independently these two parameters to change the size of a molecule:

Change the size

```
\definesubmol\hho{H-[:30]O-[:-30]H}
\chemfig{!\hho}

\setatomsep{2.5em}\chemfig{!\hho}

\scriptsize\chemfig{!\hho}

\tiny\chemfig{!\hho}\setatomsep{5em}\chemfig{!\hho}
```

You can use the optional second argument of \chemfig to *globally* enlarge or reduce a molecule, i.e. the text and links will be reduced by the same ratio:

```
                          ┌────────── Global change of the size ──────────┐
\definesubmol\hho{H-[:30]O-[:-30]H}
\chemfig{!\hho}

\chemfig[][scale=0.75]{!\hho}

\chemfig[][scale=0.5]{!\hho}

\chemfig[][scale=0.33]{!\hho}
```

12.5   Draw a ploymer element

The difficulty lies in the display of delimiters (parentheses or brackets) on bond. For this, we will again use the character "@" to define global nodes that will be used later as anchors for delimiters.

We will write a simple macro \setpolymerbracket, followed by two characters which define the opening delimiter and closing delimiter.

Then, the macro \makebraces has an argument of the form "[<dim up>,<dim down>]". These two dimensions are the height and depth of the delimiters from the baseline. The following arguments are the subscript located at the right bottom of the closing delimiter and the names of the 2 nodes at which the opening and closing delimiters will be drawn.

```
                                    ┌────── Polymers ──────┐
\newcommand\setpolymerdelim[2]{\def\delimleft{#1}\def\delimright{#2}}
\def\makebraces[#1,#2]#3#4#5{%
\edef\delimhalfdim{\the\dimexpr(#1+#2)/2}%
\edef\delimvshift{\the\dimexpr(#1-#2)/2}%
\chemmove{%
\node[at=(#4),yshift=(\delimvshift)]
{$\left\delimleft\vrule height\delimhalfdim depth\delimhalfdim
width0pt\right.$};%
\node[at=(#5),yshift=(\delimvshift)]
{$\left.\vrule height\delimhalfdim depth\delimhalfdim
width0pt\right\delimright_{\rlap{$\scriptstyle#3$}}$};}}
\setpolymerdelim()
Polyéthylène:
\chemfig{\vphantom{CH_2}-[@{op,.75}]CH_2-CH_2-[@{cl,0.25}]}
\makebraces[5pt,5pt]{\!\!n}{op}{cl}
\bigskip


Polyvinyl chloride:
\chemfig{\vphantom{CH_2}-[@{op,.75}]CH_2-CH(-[6]Cl)-[@{cl,0.25}]}
\makebraces[5pt,25pt]{\!\!\!n}{op}{cl}
\bigskip


Nylon 6:
\chemfig{\phantom{N}-[@{op,.75}]{N}(-[2]H)-C(=[2]O)-{(}CH_2{)}_5-[@{cl,0.25}]}
\makebraces[30pt,5pt]{}{op}{cl}
\bigskip


Polycaprolactame:\setatomsep{2em}
\chemfig{[:-30]-[@{left,.75}]N(-[6]H)-[:30](=[2]O)--[:30]--[:30]-[@{right,0.25}:30]}
\makebraces[5pt,25pt]{\!\!\!n}{left}{right}
\bigskip


\setpolymerdelim[]
Polyphénylène sulfide:
\chemfig{\vphantom{S}-[@{op,.75}]S-(**6(---(-[@{cl,0.25}])---))}
\makebraces[15pt,15pt]{}{op}{cl}
\bigskip


\chemfig{-CH_2-CH([6]-CO-NH-CH_2-NH-CO-CH([4]-CH_2-)([0]-[@{downleft,0.8},2]CH_2
-CH([2]-CO-NH_2)-[@{downright,0.3},2]CH_2-[,1.5]C?H-))-[@{upleft,0.8},2]CH_2
-CH([6]-CO-NH_2)-[@{upright,0.3},2]CH_2-[,1.5]CH([6]-CO-NH-CH_2-NH-C?O)-}
\makebraces[5pt,40pt]{n}{upleft}{upright}
\makebraces[38pt,7pt]{n}{downleft}{downright}
```
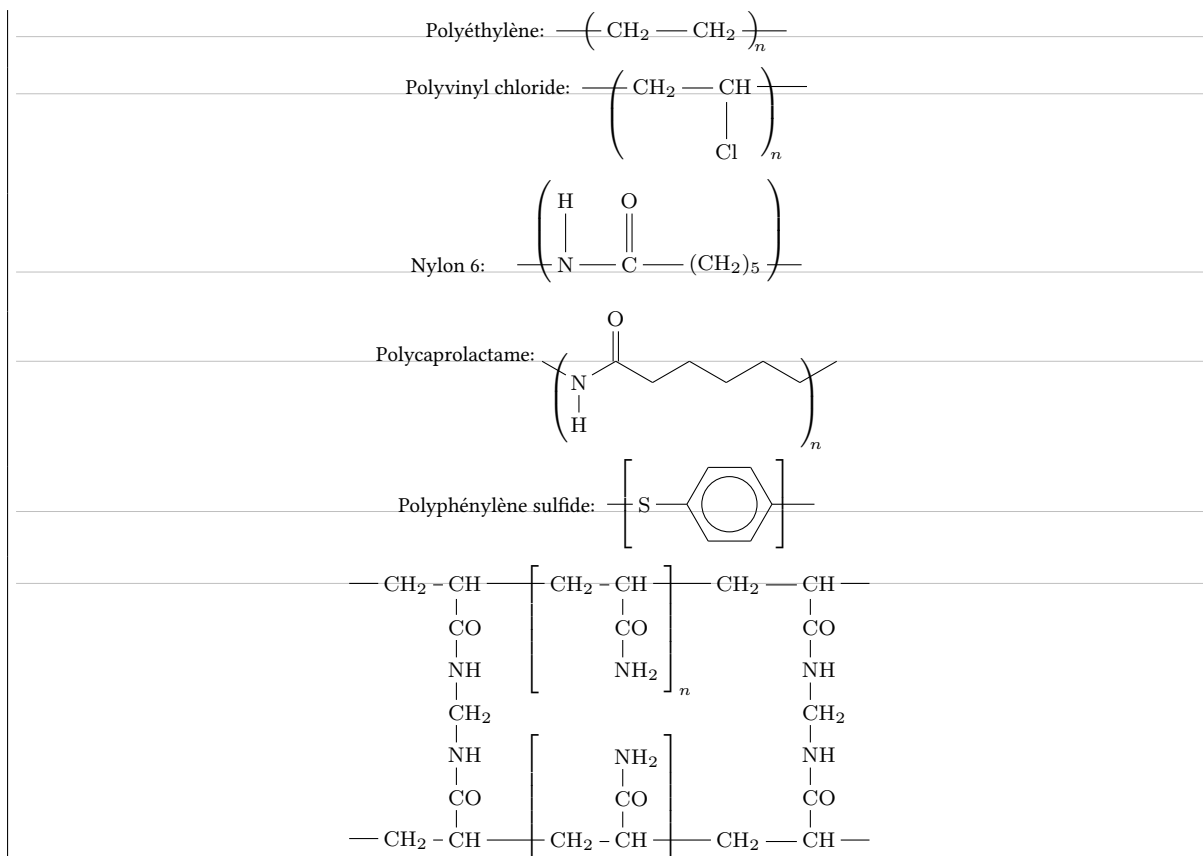
Polyéthylène: $\left(\!-\!CH_2\!-\!CH_2\!-\!\right)_n$

Polyvinyl chloride:
$$\left(\!-\!CH_2\!-\!\underset{\underset{Cl}{|}}{CH}\!-\!\right)_n$$

Nylon 6:
$$\left(\!-\!\underset{\underset{N}{|}}{\overset{\overset{H}{|}}{N}}\!-\!\underset{}{\overset{\overset{O}{\|}}{C}}\!-\!(CH_2)_5\!-\!\right)$$

Polycaprolactame:
$$\left(\!\underset{\underset{H}{|}}{N}\!\overset{\overset{O}{\|}}{C}\cdots\!\right)_n$$

Polyphénylène sulfide:
$$\left[\!-\!S\!-\!\bigcirc\!-\!\right]$$

$$-CH_2\!-\!\underset{\underset{\underset{\underset{\underset{\underset{-CH_2-CH-}{|}}{CO}}{|}}{NH}}{|}}{CH}\cdots$$
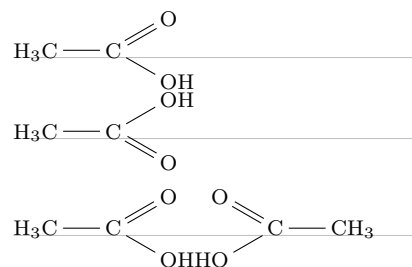
12.6   Draw the symmetrical of a molecule

The two commands \hflipnext and \vflipnext allow to draw the symmetrical of the next molecule about a horizontal or vertical axis. If we want to draw more symmetrical molecules, we need to write these commands before each molecule involved.

Symmetry

```
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}% original

\vflipnext
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}\medskip

\chemfig{H_3C-C(=[:30]O)-[:-30]OH}% original
\hflipnext
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}
```

$$H_3C\!-\!C\!\overset{O}{\underset{OH}{\big\langle}}$$

$$H_3C\!-\!C\!\overset{OH}{\underset{O}{\big\langle}}$$

$$H_3C\!-\!C\!\overset{O}{\underset{OHHO}{\big\langle}}\!\overset{O}{\big\rangle}\!C\!-\!CH_3$$

12.7   Add text above bonds and arc to angles

Once we have understood that the character "@" can put a "global" ti*kz* node, that is to say a node accessible after the molecule has been drawn, everything that ti*kz* can do with nodes (that is to say a lot of things) becomes possible.

To write something above or below a bond, we can put two "global" nodes on the atoms at the ends of this bond and write midway of them a text, raised or lowered so that it falls to just above or below the bond. This is done by the macro \bondname in the code below [13].

To draw an arc between two bonds, three atoms are involved on which we have to put three "global" nodes. The macro \arcbetweennodes calculates the angle between two lines drawn from a node. Then \arclabel draws an

---

[13]The \si command is provided by the **siunitx** package.

L*A*T*E*Xed by Christian T*ELLECHEA*, the November 16, 2012.

44

arc between two bonds and writes a text next to the arc: the optional argument of this macro is the ti*k*z code used to custom the arc. The second argument is the radius of the arc and the following three arguments are the names of global nodes between which the arc must be drawn, the middle name needs to be the vertex of the angle. The last argument is the text to write.

Arcs and text on bonds

```
\newcommand\namebond[4][5pt]{\chemmove{\path(#2)--(#3)node[midway,sloped,yshift=#1]{#4};}}

\newcommand\arcbetweennodes[3]{%
  \pgfmathanglebetweenpoints{\pgfpointanchor{#1}{center}}{\pgfpointanchor{#2}{center}}%
  \let#3\pgfmathresult}

\newcommand\arclabel[6][stealth-stealth,shorten <=1pt,shorten >=1pt]{%
  \chemmove{%
    \arcbetweennodes{#4}{#3}\anglestart \arcbetweennodes{#4}{#5}\angleend
    \draw[#1]([shift=(\anglestart:#2)]#4)arc(\anglestart:\angleend:#2);
    \pgfmathparse{(\anglestart+\angleend)/2}\let\anglestart\pgfmathresult
    \node[shift=(\anglestart:#2+1pt)#4,anchor=\anglestart+180,rotate=\anglestart+90,inner sep=0pt,
        outer sep=0pt]at(#4){#6};}}

\chemfig{@{a}A=[:30,1.5]@{b}B-[7,2]@{c}C-@{d}D}
\namebond{a}{b}{\scriptsize My text}
\namebond[-3.5pt]{b}{c}{\small\color{red}$\pi$}
\namebond{c}{d}{\small1\si\angstrom}
\medskip

Horizontal water molecule:  \chemfig{@{1}H-[::37.775,2]@{2}O-[::-75.55,2]@{3}H}.
\namebond{1}{2}{\footnotesize0.9584\si\angstrom}
\namebond{2}{3}{\footnotesize0.9584\si\angstrom}
\arclabel{0.5cm}{1}{2}{3}{\footnotesize104.45\si\degree}
\qquad
Water molecule rorated 30\si\degree:  \chemfig{[:30]@1H-[::37.775,2]@2O-[::-75.55,2]@3H}
\namebond12{\footnotesize0.9584\si\angstrom}
\namebond23{\footnotesize0.9584\si\angstrom}
\arclabel{0.5cm}{1}{2}{3}{\footnotesize104.45\si\degree}
```
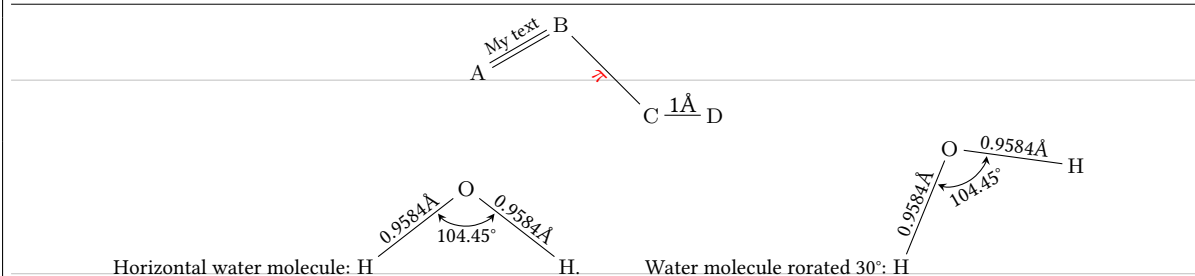
12.8   Schéma de Lewis à l'angle près

Pour quelques cas très particuliers, il est parfois nécessaire de positionner des schémas de Lewis au degrés près et non plus à des multiples de 45 degrés.

Pour cela, il est relativement facile d'écrire une macro `\mylewis`, admettant un argument optionnel (la longueur du doublet) qui vaut ici 2ex et deux arguments obligatoires; le premier étant le nom de l'atome et le second étant la liste des angles séparés par des virgules.

L'astuce consiste à créer un nœud invisible circulaire qui contient l'atome et tracer les doublets aux angles désirés de façon à ce qu'ils soient tangents à ce cercle :

Doublets au degrés prés

```
\makeatletter
\newcommand\mylewis[3][2ex]{% #1=longueur, #2=atome #3=liste des angles
\tikzpicture[baseline,anchor=base]%
\node[inner sep=0pt,outer sep=1pt,circle,overlay](atom@@)at(0,0){\phantom{#2}};%
\node[inner sep=0pt,outer sep=0pt]at(0,0){#2};%
\def\list@angle{#3,}%
\loop
```

```
\expandafter\grab@angle\list@angle\@nil
\pgfextractx\CF@dim{\pgfpointanchor{atom@@}\current@angle}\edef\CF@dimax{\the\CF@dim}%
\pgfextracty\CF@dim{\pgfpointanchor{atom@@}\current@angle}\edef\CF@dimay{\the\CF@dim}%
\pgfmathparse{#1*sin(\current@angle)/2}\let\offset@xx\pgfmathresult
\pgfmathparse{#1*cos(\current@angle)/2}\let\offset@yy\pgfmathresult
\draw[line width=.4pt,overlay]
(\CF@dimax-\offset@xx,\CF@dimay+\offset@yy)--(\CF@dimax+\offset@xx,\CF@dimay-\offset@yy);%
\unless\ifx\@empty\list@angle
\repeat
\endtikzpicture
}
\def\grab@angle#1,#2\@nil{\def\current@angle{#1}\def\list@angle{#2}}
\makeatother
Essai :  \mylewis{Cl}{15}\qquad \mylewis[3ex]{Cl}{-60}\qquad \mylewis[1.5ex]{Cl}{60,120,240,300}
\bigskip

\chemfig{P(-[:72]\mylewis{Cl}{-18,72,162})(-[:-72]Cl)
(-[:144]Cl)(-[:-144]Cl)-\mylewis{Cl}{0,90,-90}}
```
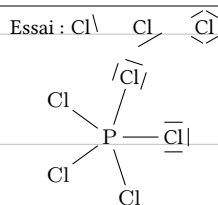
PART IV

Reaction schemes

Following several requests from users, it had become evident that ChemFig had a weakness regarding the drawing of reaction schemes. The gap is now filled. Therefore, ChemFig has now reached version 1.0 since I consider that the main features sought are now available.

I thank Clemens NIEDERBERGER for his help and the tests he carried out on the new features presented in this part.

1   Overview

A reaction scheme must be contained between the commands "\schemestart" and "\schemestop". As shown in this example, debug information is either hidden or displayed with the command "\schemedebug" and the argument "false" or "true":

Example 1

```
\schemedebug{false}
\schemestart
  \chemfig{*6(-=-=-=)}\arrow
  \chemfig{X=[1]Y}\arrow
  \chemfig{S>T}
\schemestop
\bigskip

\schemedebug{true}
\schemestart
  \chemfig{*6(-=-=-=)}\arrow
  \chemfig{X=[1]Y}\arrow
  \chemfig{S>T}
\schemestop
```
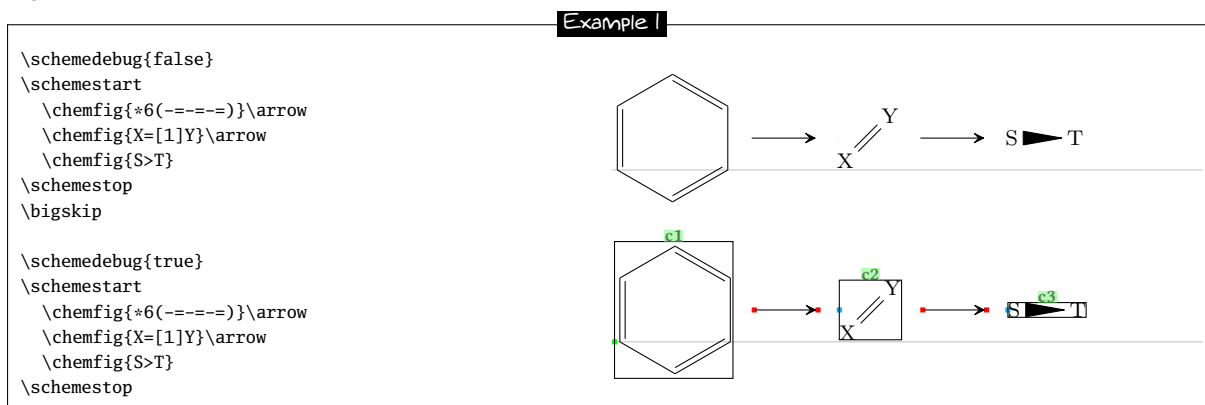
Some comments:
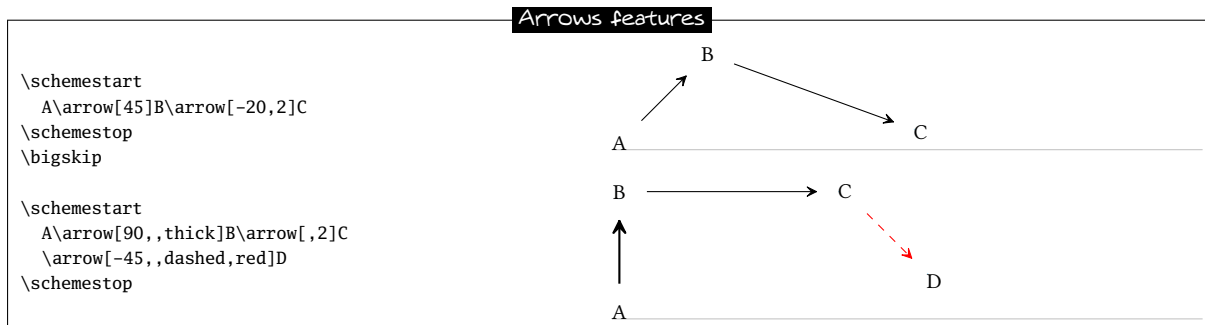
- the \arrow commands draw the arrows;
- everything lying between two \arrow commands is considered a compound. It was decided that all possible settings, whether for arrows or compounds, are controlled by the arguments of the \arrow command, whose syntax may become quite complex;
- arrows are plotted horizontally, this can obviously be modified;
- arrows are plotted on the imaginary line connecting the center of the compounds' bounding boxes (the red and blue squares are the anchoring points of arrows). This behavior can also be modified;
- debug information is displayed with the \schemedebug command. It consists of:
  - the green label above the bounding boxes is the default name assigned to compounds by ChemFig. It follows the series "c1", "c2", etc. Numbering is reset to 1 for every reaction scheme.
  - display of the compounds bounding boxes;
  - the arrows start and end points represented by red points and anchors by blue points;
- the distance from edge to edge between two compounds is defined with the \setcompoundsep{<dim>} command, where the argument is a valid dimension in the sense of TeX. By default, and if the argument is empty, this dimension is 5em;
- finally, the distance between the edges of the compounds and the beginning and end of the arrows is defined with the \setarrowoffset{<dim>} command where the argument is valid under a dimension of TeX. By default, and if the argument is empty, this dimension is 4pt.

2   Arrow types

When the `\arrow` command is followed by an optional argument in braces (which is not mandatory), the argument defines the type of arrow:

```
                                        Arrow types
\schemestart A\arrow{->}B\schemestop\par % by default    A ————→ B
\schemestart A\arrow{-/>}B \schemestop\par               A —⫽→ B
\schemestart A\arrow{<-}B \schemestop\par                A ←———— B
\schemestart A\arrow{<->}B \schemestop\par               A ←———→ B
\schemestart A\arrow{<=>}B \schemestop\par               A ⇌ B
\schemestart A\arrow{<->>}B \schemestop\par              A ⇀ B
\schemestart A\arrow{<<->}B \schemestop\par              A ↽ B
\schemestart A\arrow{0}B \schemestop\par                 A      B
\schemestart A\arrow{-U>}B \schemestop                   A ————→ B
```

The arrow "`-U>`" is not fully drawn, an arc can be added tangent to the arrow center using optional arguments on the command, see page 55. Here is a "`-U>`" arrow with the arc on top of it: A $\overset{x\quad y}{\longrightarrow}$ B

For the sake of clarity, capital letters will be used throughout the documentation instead of chemical formulas made with the `\chemfig`command except for specific examples. Reaction schemes obviously work identically with letters and drawn molecules. Several examples are shown in the Gallery with proper reaction schemes.

3   Arrows features

Each arrow is characterized by:

- an angle expressed in degrees;
- a coefficient that specifies the arrow length through the multiplication of the compounds spacing value defined by `\setcompoundsep`;
- a style with ti*kz* instructions to customize the color, the thickness or other graphical attribute of the arrow.

These features are defined with the command `\setarrowdefault{angle,coeff,style}`. And this is made *outside* the reaction scheme. If the angle is empty, it takes the value 0, and if the coefficient is empty, it is equal to 1. The as-defined features will be kept throughout the document, from the point where the instruction is found. To reset the default settings, just enter an empty argument.

```
                                Definition of default values
\schemestart A\arrow B\arrow C\schemestop

\setarrowdefault{15,1.5,red,thick}                  A ————→ B ————→ C
\schemestart A\arrow B\arrow C\schemestop

\setarrowdefault{,2.5,dashed}
\schemestart A\arrow B\arrow C\schemestop           A - - - - - → B - - - - - → C
\setarrowdefault{}                                  A ————→ B ————→ C
\schemestart A\arrow B\arrow C\schemestop
```

In order to locally modify one or all of these default values, the `\schemestart` command accepts an optional argument in the form `[angle,coeff,style]` which changes the default arrow features within the sole reaction scheme:

```
                                    Optional argument
\setarrowdefault{5,2.5,blue}
\schemestart A\arrow B\arrow C\schemestop

\schemestart[0] A\arrow B\arrow C\schemestop        A ————→ B ————→ C
\schemestart[0,1] A\arrow B\arrow C\schemestop      A ——→ B ——→ C
\schemestart[0,1,thick] A\arrow B\arrow C\schemestop   A ——→ B ——→ C
\schemestart[0,1,black] A\arrow B\arrow C\schemestop   A ——→ B ——→ C
```

Regarding style, the rule is: the style specified in the argument in brackets applies *after* the default style, without overwrting it! This is why only the "black" color attribute is able to overwrite the " blue" default style.
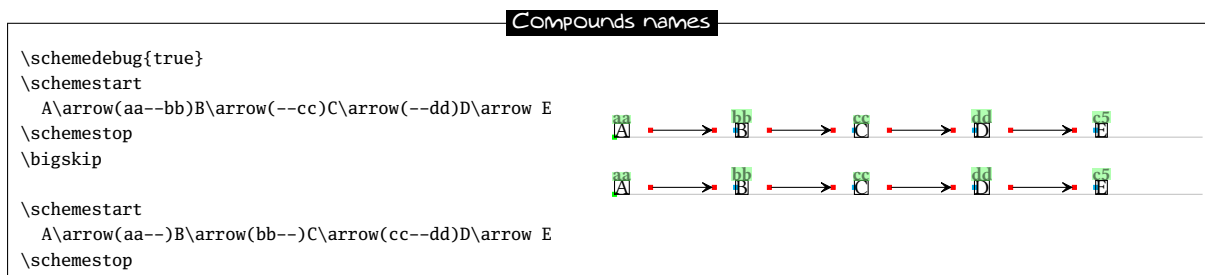
Finally, the \arrow command accepts an optional argument in brackets in the form [angle,coeff,style] to change the feature of that given arrow. As above, style applies *after* the default style and *after* the style possibly-specified in the optional argument of the \schemestart command, again without overwrting them.


```
\schemestart
  A\arrow[45]B\arrow[-20,2]C
\schemestop
\bigskip

\schemestart
  A\arrow[90,,thick]B\arrow[,2]C
  \arrow[-45,,dashed,red]D
\schemestop
```
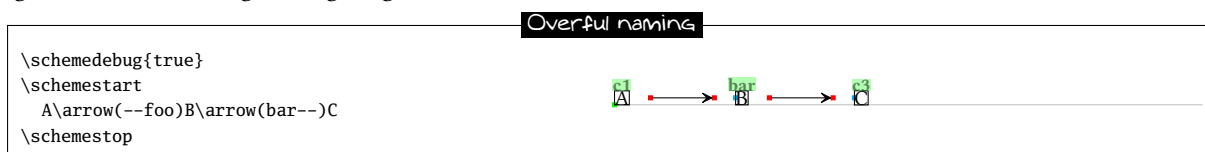
4   Compounds names

Automatic naming of compounds ("c1", "c2", etc.) can be overridden. For this, the \arrow command must be immediately followed by an argument in parentheses. The argument is of the form: (n1--n2). The compounds located at the beginning and at the end of the arrow are named "n1" and "n2", respectively. Any alphanumeric string can be used. The numbering of the names "c<n>" continues internally, so if a compound has a different name than the default one, it does not affect the default name of the subsequent compounds.

Names are optional, and the argument can be either (n1--) and (--n2).


```
\schemedebug{true}
\schemestart
  A\arrow(aa--bb)B\arrow(--cc)C\arrow(--dd)D\arrow E
\schemestop
\bigskip

\schemestart
  A\arrow(aa--)B\arrow(bb--)C\arrow(cc--dd)D\arrow E
\schemestop
```

Note that both methods are equivalent. Therefore, compounds can either be named by arrows preceding or following them. However, when a compound is surrounded by two arrows specifying its name, the first name is ignored and a warning message is generated:


```
\schemedebug{true}
\schemestart
  A\arrow(--foo)B\arrow(bar--)C
\schemestop
```

Here compound "B" is called "foo" by the arrow pointing at it, and "bar" by the arrowing leaving from it. Thus ChemFig generates a warning mentioning that the name "foo" will be ignored:
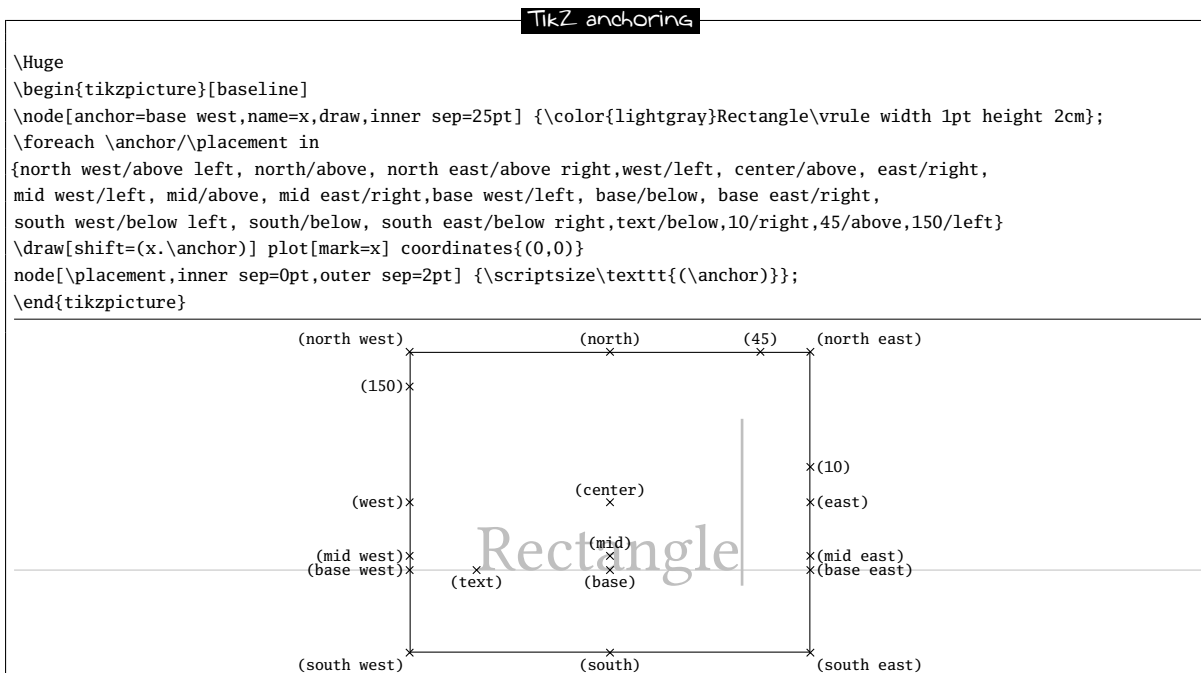
```
Package chemfig Warning: two names for the same node, first name "foo" ignored
```

5   Anchoring

As noted above, arrows lie on the line connecting the center of the compounds' bounding boxes. Default anchors are called "center" in the sense of ti*kz*. Non-default anchoring points can be user-specified as well with an argument between brackets:

$$(n1.a1--n2.a2)$$

where the anchor "a1" or "a2" can be: north west, north, north east, west, center, east, mid west, mid, mid east, base west, base, base east, south west, south , south east, text, or any angle. Here is an example from the ti*kz* manual where the anchors are located on the bounding box:
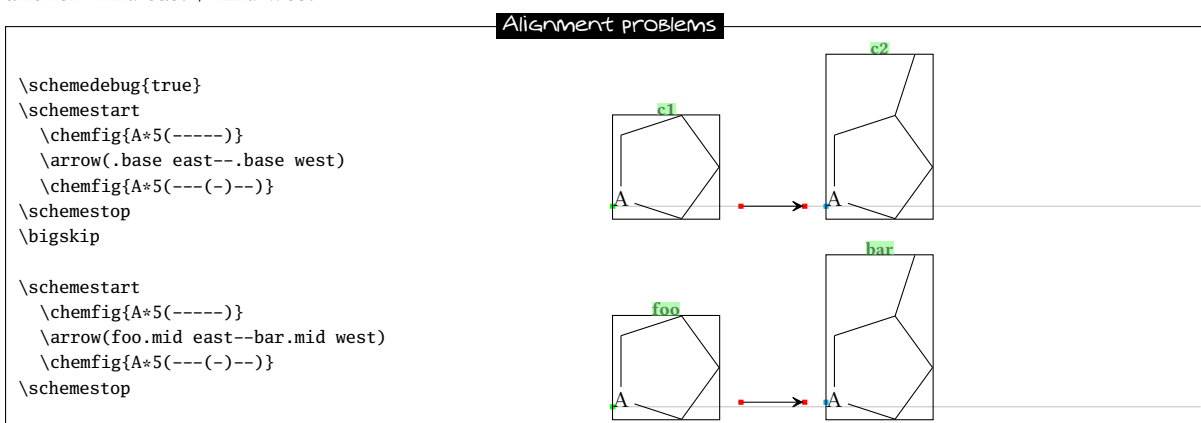

TikZ anchoring

```
\Huge
\begin{tikzpicture}[baseline]
\node[anchor=base west,name=x,draw,inner sep=25pt] {\color{lightgray}Rectangle\vrule width 1pt height 2cm};
\foreach \anchor/\placement in
{north west/above left, north/above, north east/above right,west/left, center/above, east/right,
mid west/left, mid/above, mid east/right,base west/left, base/below, base east/right,
south west/below left, south/below, south east/below right,text/below,10/right,45/above,150/left}
\draw[shift=(x.\anchor)] plot[mark=x] coordinates{(0,0)}
node[\placement,inner sep=0pt,outer sep=2pt] {\scriptsize\texttt{(\anchor)}};
\end{tikzpicture}
```

Like for names, arrival and departure anchoring points are independent and optional.

In this example, the default alignment is not good because the two "A" are not aligned vertically. Debug information show that the default "center" anchors are not suitable:

Alignment problems

```
\schemedebug{true}
\schemestart
  \chemfig{A*5(-----)}
  \arrow
  \chemfig{A*5(---(-)--)}
\schemestop
```
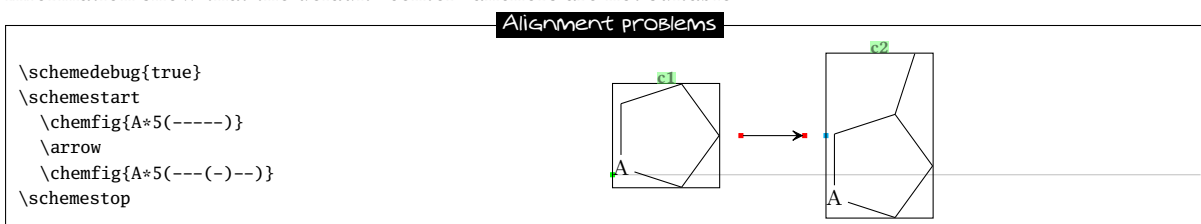
For the alignment to be correct, arrows will leave/arrive either from the anchor "base east"/"base west", or from anchor "mid east"/"mid west":

Alignment problems

```
\schemedebug{true}
\schemestart
  \chemfig{A*5(-----)}
  \arrow(.base east--.base west)
  \chemfig{A*5(---(-)--)}
\schemestop
\bigskip

\schemestart
  \chemfig{A*5(-----)}
  \arrow(foo.mid east--bar.mid west)
  \chemfig{A*5(---(-)--)}
\schemestop
```

One last anchor need be specified: the anchor of the first compound with respect to the baseline of the text just before it. This is illustrated by the green point on the left-hand side of the scheme below:
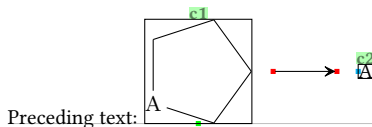
```
\schemedebug{true}
Preceding text:
\schemestart
  \chemfig{A*5(-----)}\arrow A
\schemestop
```
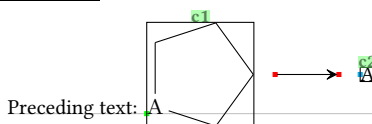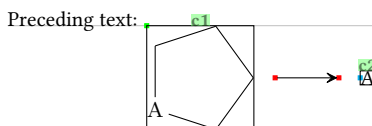
The default position of this anchor on the first compound's bounding box is that given by "text". This position can be controlled with the second optional argument of the \schemestart command:
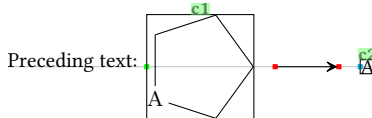
```
\schemedebug{true}
Preceding text:
\schemestart[][south]
  \chemfig{A*5(-----)}\arrow A
\schemestop
\bigskip

Preceding text:
\schemestart[][north west]
  \chemfig{A*5(-----)}\arrow A
\schemestop
\bigskip

Preceding text:
\schemestart[][west]
  \chemfig{A*5(-----)}\arrow A
\schemestop
```

6   Compounds style

The \arrow command can also include ti*k*z instructions to define the bounding box style "s" of the reactant and the product of the reaction. This is done with the argument between parentheses. Always style through the argument in brackets of the \arrow, we can specify with ti*k*z instructions the style "s" to bounding box of the compound of departure or of arrival. Therefore the complete syntax of the \arrow command, with each specification being optional, is as follows:
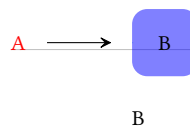
\arrow(n1.a1[s1]--n2.a2[s2]){arrow type}[angle,coeff,arrow style]

Like forn names, if specific styles are given to one compound by arrows arriving on it and leaving from it, the first style will be ignored with a warning.

```
\schemestart
  A
  \arrow([red]--[fill=blue,semitransparent,text opacity=1,
  inner sep=10pt,rounded corners=2mm])
  B
\schemestop
\bigskip

\schemestart
  A\arrow(--foo[yshift=5mm])B
\schemestop
```
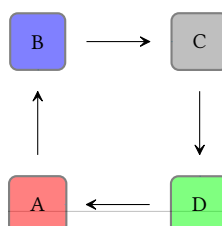
The macro \setcompoundstyle{<code tikz>} allows to globally define the style of compounds displayed thereafter. Entering an empty argument results in the absence of style, which corresponds to the default case.

Here a style is defined with round corner-shaped boxes and semitransparent background:

```
\setcompoundstyle{draw,line width=0.8pt,semitransparent,
text opacity=1,inner sep=8pt,rounded corners=1mm}
\schemestart
  A\arrow([fill=red]--[fill=blue])[90]
  B\arrow(--[fill=gray])
  C\arrow(--[fill=green])[-90]
  D\arrow(--[draw=none])[-180]
\schemestop
```
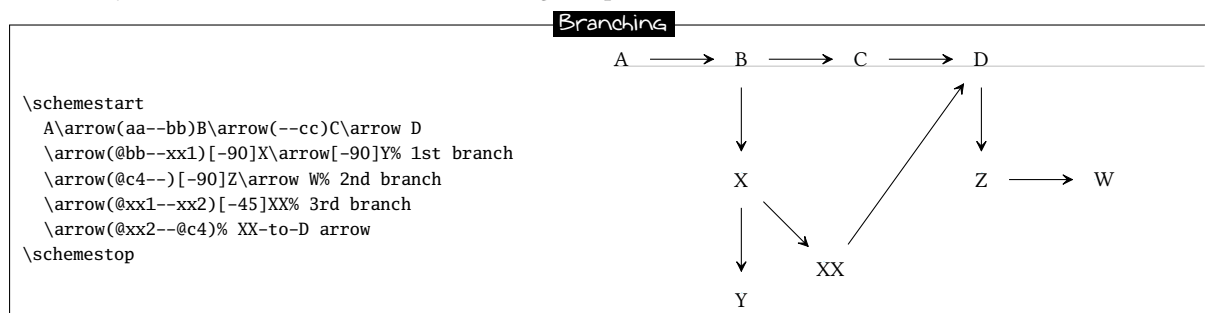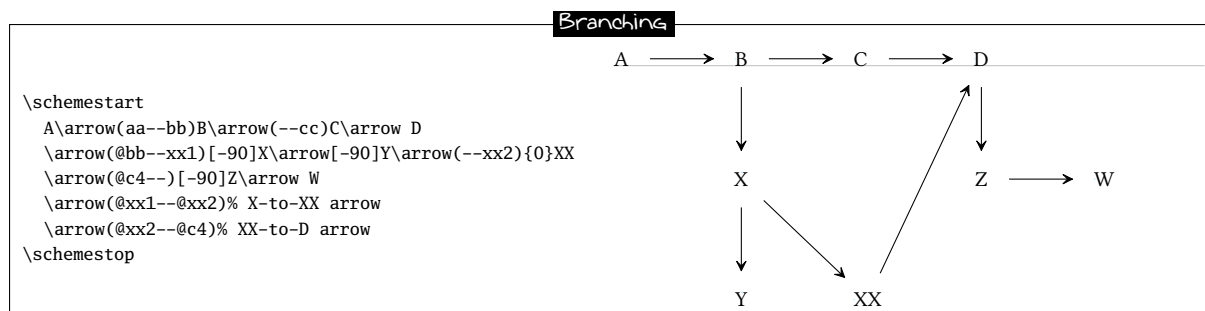
7 Branching

So far, only linear reaction schemes have been treated. Branched schemes are also possible and this is where compound names play a key role. When a name is preceded by "@" in the argument between brackets of the \arrow command, it means that the compound already exists. Several scenarios are possible:

- (@n1--n2): the arrow will leave from the existing compound "n1" and the scheme will continue following the arrow, thus creating a branch;
- (@n1--@n2): the arrow is drawn between two existing compounds, no matter whether they are already defined or whether they will later in the reaction scheme: therefore this syntax can be placed *anywhere* in the code of the reaction scheme;
- (n1--@n2): this syntax is not permitted;

In the following example, 3 branches are made, a first one from "B", a second one from "D" and a last one from "X". Finally one more arrow connects two existing compounds: "XX" and "D":


```
\schemestart
  A\arrow(aa--bb)B\arrow(--cc)C\arrow D
  \arrow(@bb--xx1)[-90]X\arrow[-90]Y% 1st branch
  \arrow(@c4--)[-90]Z\arrow W% 2nd branch
  \arrow(@xx1--xx2)[-45]XX% 3rd branch
  \arrow(@xx2--@c4)% XX-to-D arrow
\schemestop
```
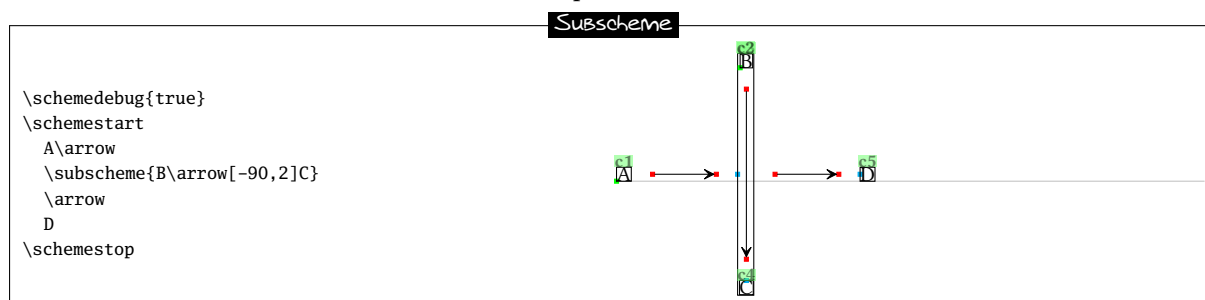
One may wish to have "Y" and "XX" on the same horizontal line. To achieve this, a horizontal invisible bond is drawn between "Y" and "XX"; the scheme is completed with a final arrow between the two existing compounds "XX" and "D":


```
\schemestart
  A\arrow(aa--bb)B\arrow(--cc)C\arrow D
  \arrow(@bb--xx1)[-90]X\arrow[-90]Y\arrow(--xx2){0}XX
  \arrow(@c4--)[-90]Z\arrow W
  \arrow(@xx1--@xx2)% X-to-XX arrow
  \arrow(@xx2--@c4)% XX-to-D arrow
\schemestop
```
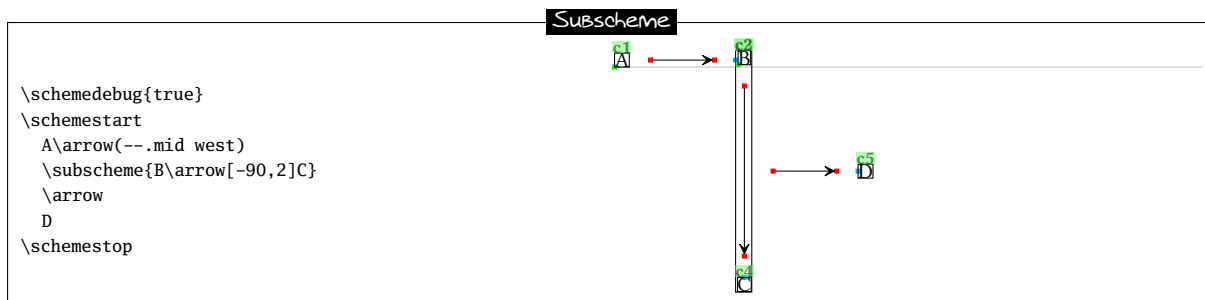
8 Subscheme

A fraction of the reaction scheme can be defined within a single bounding box, so that ChemFig treats it as a compound. The reaction scheme fraction is defined inside the compulsory argument between braces of the \subscheme command so it is subsequently reagarded as a single entity. When \subscheme is located after an arrow, the command labels this subscheme as a compound named "c<n+1>":


```
\schemedebug{true}
\schemestart
  A\arrow
  \subscheme{B\arrow[-90,2]C}
  \arrow
  D
\schemestop
```

Although this is not clearly seen because of labels overlap, the box around the subscheme is called "c2", and name numbering continues inside the subscheme with B called "c3" and C called "c4". Since the first compound in the subscheme is "B", the subscheme's baseline is that of "B". This can be pointed out by specifying the anchors:

```
\schemedebug{true}
\schemestart
  A\arrow(--.mid west)
  \subscheme{B\arrow[-90,2]C}
  \arrow
  D
\schemestop
```

Note that since "\subscheme{<scheme>}" is only a convenient shortcut for \schemestart<scheme>\schemestop, it can be used with the same optional arguments as \schemestart.

ChemFig provides the \chemleft and \chemright command pair. These allow to set expandable delimiters on either side of a material. The commands must be followed by delimiters, just like in the case of TeX primitive commands \left and \right:

$$\text{\chemleft<car1><material>\chemright<car2>}$$

where <car1> and <car2> can be "(" et ")" or "[" and "]", or any other expandable delimiter consistent with the \left et \right commands.

```
\chemleft\lfloor\chemfig{A-[1]B}\chemright)

\chemleft\{\chemfig{A-[1,1.25]B-[6,1.25]C}\chemright|

\chemleft[\chemfig{H-[1]O-[7]H}\chemright]
```
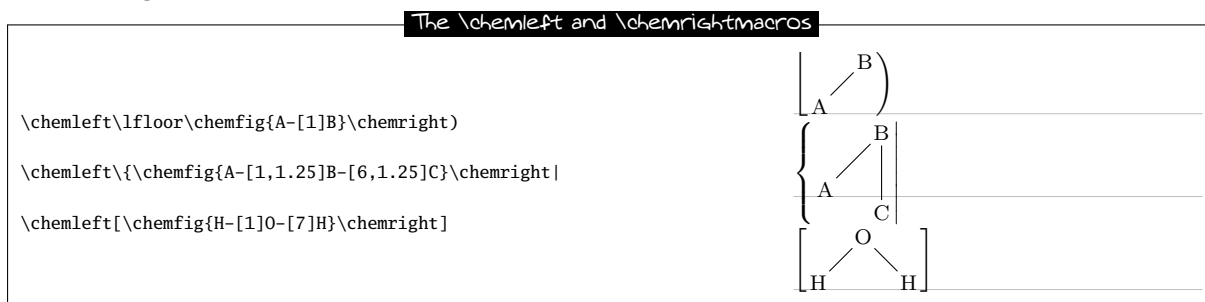
The code of the reaction scheme discussed above including \chemleft and \chemright is written:

```
\schemestart
  A\arrow
  \chemleft[\subscheme{B\arrow[-90,2]C}\chemright]
  \arrow
  D
\schemestop
```
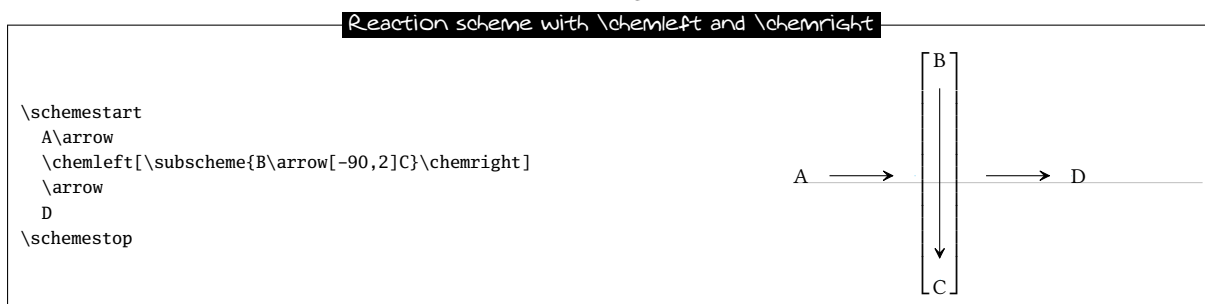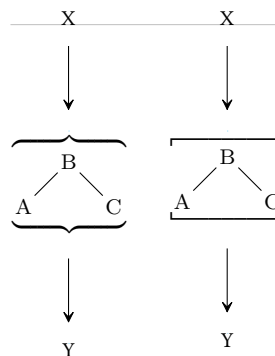
By analogy, the macros \chemup and \chemdown can be used to draw expandable delimiters above and below the material, respectively:

$$\text{\chemup<car1><material>\chemdown<car2>}$$

For example:

The \chemup and \chemdownmacros

```
\schemestart[-90]
X\arrow
\chemup\{\chemfig{A-[1]B-[7]C}\chemdown\}
\arrow Y
\schemestop
\qquad
\schemestart[-90]
X\arrow
\chemup[\chemfig{A-[1]B-[7]C}\chemdown]
\arrow Y
\schemestop
```

Delimiters can can also be drawn through compounds' style and apply them to a random compound (and hereby to a subscheme). These expandable delimiters (parentheses, brackets, braces) can be used upon loading the "matrix" ti*k*z library in the document preamble:
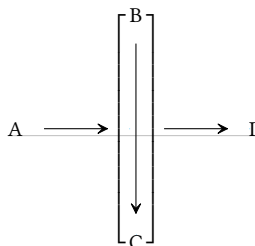
$$\usetikzlibrary{matrix}$$

Since the \chemleft, \chemright, \chemup and \chemdown commands are available, the ChemFig package will *not* automatically load the library. As long as the user want to access this special set of delimiters, the library must be explicitly loaded.

The same brackets-delimited subscheme as above is presented again:

The "matrix" library delimiters
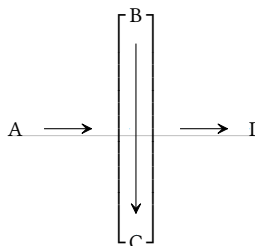
```
\schemestart
  A\arrow(--[left delimiter={[}, right delimiter={]}])
  \subscheme{B\arrow[-90,2]C}
  \arrow
  D
\schemestop
```

Since the delimiters are drawn outside the bounding box, it is advisable to slightly shorten the incoming and outgoing arrows:

Subscheme
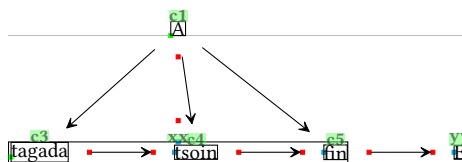
```
\schemestart
  A\arrow(--[left delimiter={[},
  right delimiter={]}])[,,shorten >=6pt]
  \subscheme{B\arrow[-90,2]C}
  \arrow[,,shorten <=6pt]
  D
\schemestop
```

Subschemes should be used with care, undesired results are sometimes observed. In this example, a subscheme is used to horizontally align 3 different compounds:

Subscheme

```
\schemedebug{true}
\schemestart
  A
  \arrow{0}[-90]
  \subscheme{%
    tagada\arrow{}
    tsoin\arrow{}
    fin}
  \arrow(xx--yy){}E
  \arrow(@c1--@c3){}
  \arrow(@c1--@c5){}
  \arrow(@c1--@c4){}
\schemestop
```

The center of the subscheme is exactly located on the same vertical lign as the center of compound "A". This is because the two entities are connected by an invisible arrow with a -90 angle. However, the arrow between the two pre-existing compounds "A" and "tsoin" is *not* vertical because "tsoin" is not on the center of the subscheme since "tagada" is wider than "end". If this arrow is to be vertical within the use of the `\subscheme` command, one must find a correct angle for the arrival anchor of the invisible arrow by try-and-error.
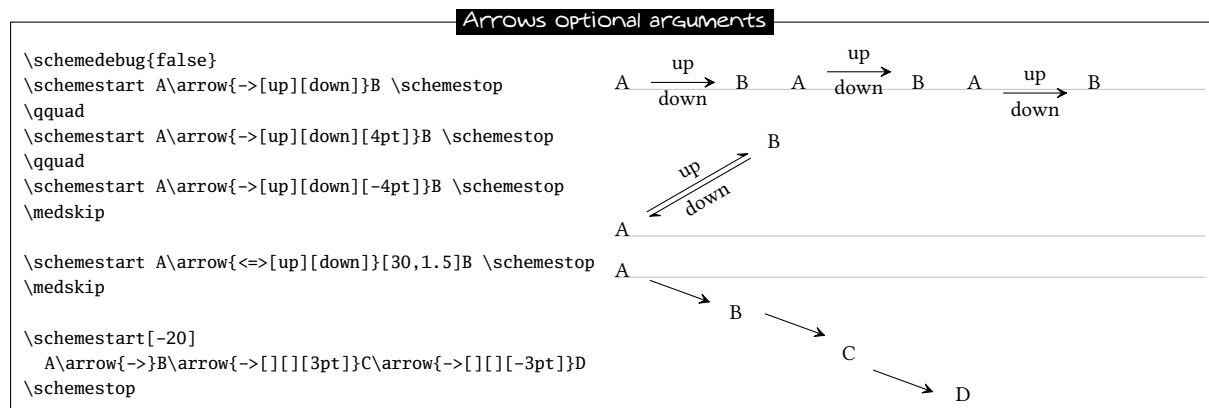
A much simpler method is to use a branch instead of a subscheme: draw a *visible* arrow between "A" and "tsoin", and then draw horizontal arrows on both sides of "tsoin", with a branch for the right-hand side arrows.


```
\schemedebug{true}
\schemestart
  A
  \arrow(--tsoin){->}[-90]
  tsoin
  \arrow{<-}[180]
  tagada
  \arrow(@tsoin--fin){}
  fin
  \arrow{}
  E
  \arrow(@c1--@c3){}
  \arrow(@c1--@fin){}
\schemestop
```
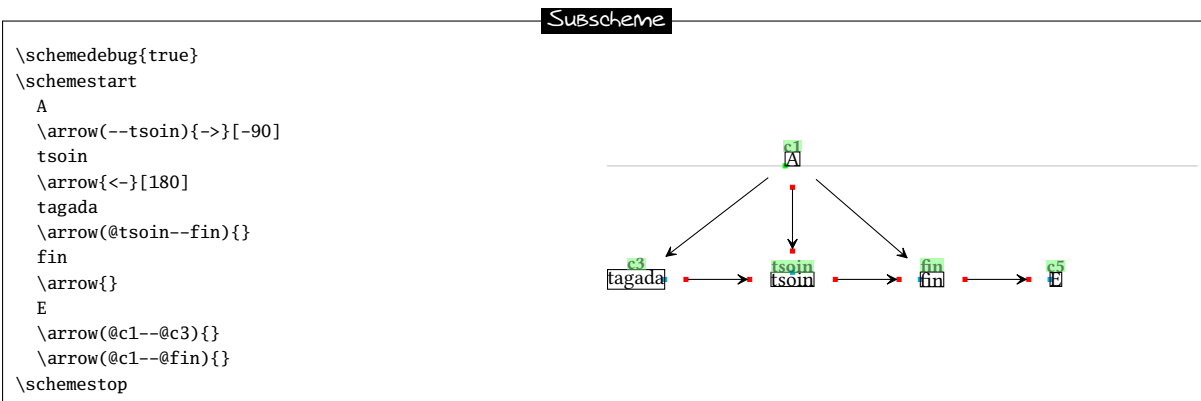
9   Arrows optional arguments

Within the argument in braces of the `\arrow` command, the arrow name can be followed by optional arguments written between brackets. Here are the possible values for these optional arguments and their meaning, as defined by ChemFig:

- the arrows "->", "<-", "<->", "<=>", "<<->","<->>", "-/>" have three optional arguments:
  - the first one contains the "label" to be placed above the arrow;
  - the second one contains the "label" to be placed below the arrow. The orientation of these two labels is given by the same angle as the arrow. The perpendicular shift between the arrow and the label anchor can be adjusted with the `\setarrowlabelsep{<dim>}` command, its argument is a valid dimension in the sense of TEX. By default or if the argument is empty, this distance is 3pt. Labels contained in the two optional arguments are *not* typed in math mode.
  - the third one is a dimension corresponding to a shift perpendicular to the arrow that can be applied to it: the dimension is positive for an upward shift of the arrow (and of its labels, if any), and negative for a downward shift.
- the "-U>" arrow has 5 optional arguments:
  - the first three are identical to those found in the other arrow types;
  - the fourth one is a coefficient (which is 0.333 by default) which multiplies the length of the arrow to get the radius of the arc;
  - the fifth one is the half-angle from the center of the arc path, it is 60 degrees by default.
- the invisible arrow "0" accepts two optional arguments of the same type as the first two listed above;


```
\schemedebug{false}
\schemestart A\arrow{->[up][down]}B \schemestop
\qquad
\schemestart A\arrow{->[up][down][4pt]}B \schemestop
\qquad
\schemestart A\arrow{->[up][down][-4pt]}B \schemestop
\medskip

\schemestart A\arrow{<=>[up][down]}[30,1.5]B \schemestop
\medskip

\schemestart[-20]
  A\arrow{->}B\arrow{->[][][3pt]}C\arrow{->[][][-3pt]}D
\schemestop
```

A problem arises for vertical arrows:

Vertical arrows

```
\schemestart
  A\arrow{->[up][down]}[-90]B
\schemestop
```

For the sake of clarity, one may prefer to have the "above" and "below" labels written horizontally. Label angles can be specified, while default is the same angle as that of the arrow. To choose a specific angle, *{<angle>} can be written at the beginning of the optional arguments:

Choice of angles

```
\schemedebug{true}
\schemestart A\arrow{->[*{0}up][*{0}down]}[90]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}up][*{0}down]}[45]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}up][*{0}down]}[-45]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}up][*{0}down]}[-90]B\schemestop
```

The default position of the label anchor can lead to undesired results:

Anchors

```
\schemedebug{true}
\schemestart
  A\arrow{->[*{0}on top of][*{0}underneath]}[45,2]B
\schemestop
```

To counter this, the anchoring position can be specified as well to override the one selected by **ChemFig** by default. The syntax for this is: *{<angle>.<ancre>}.

Anchors

```
\schemedebug{true}
\schemestart
  A\arrow{->[*{0.0}on top of][*{0.180}underneath]}[45,2]B
\schemestop
\qquad
\schemestart
  A\arrow{->[*{0.south east}on top of]%
    [*{0.north west}underneath]}[45,2]B
\schemestop
```

The "-U>" arrow remains a particular case. If one of the two labels from the first two optional arguments is present, the corresponding arc is plotted:

The -U> arrow
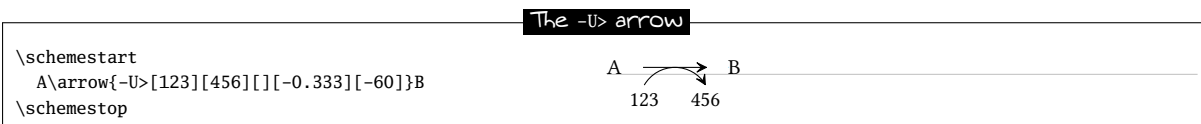
```
\schemestart A\arrow{-U>[123][456]}B\schemestop
\qquad
\schemestart A\arrow{-U>[123]}[30]B\schemestop
\qquad
\schemestart A\arrow{-U>[][456]}[-30]B\schemestop
```

The fourth and fifth optional arguments modify the appearance of the arc: respectively the arrow length coefficient which sets the arc radius, and the angle that defines the half arc:
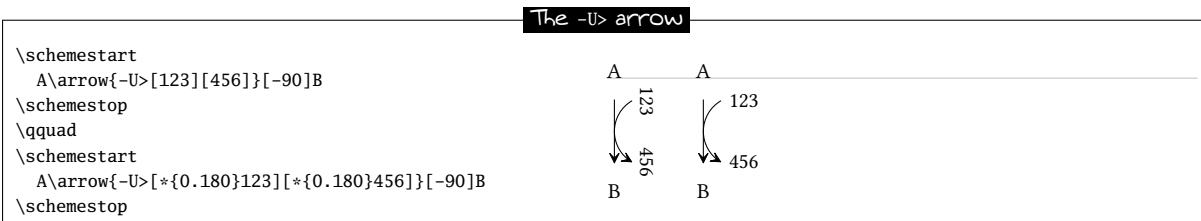
The -U> arrow

```
\schemestart A\arrow{-U>[123][456][][0.25]}B\schemestop
\qquad
\schemestart A\arrow{-U>[123][456][][][90]}B\schemestop
\qquad
\schemestart A\arrow{-U>[123][456][][1][45]}B\schemestop
```

With negative values for the radius and the angle, the arc is drawn below the arrow:

```
\schemestart
  A\arrow{-U>[123][456][][-0.333][-60]}B
\schemestop
```

Label angles and anchoring customization is controlled with the first two arguments, just like for other arrows:

```
\schemestart
  A\arrow{-U>[123][456]}[-90]B
\schemestop
\qquad
\schemestart
  A\arrow{-U>[*{0.180}123][*{0.180}456]}[-90]B
\schemestop
```

10   Arrows customization

This section is quite technical and requires some knowledge of ti*kz*. It is targeted at advanced users only who need to define their own arrows.

The \definearrow command allows to build custom arrows. Its syntax is:

$$\text{\textbackslash definearrow\{<number>\}\{<arrow name>\}\{<code>\}}$$

where <number> is the number of optional arguments that will be used in the <code>, with the usual syntax #1, #2, etc. These optional arguments cannot accept default values; if no value is specified upon using the macro \arrow, the arguments will remain empty.

Before going further, let's examine the available internal macros when drawing arrows. Since these macros include the @ character in their name, they can only be accessed through the \makeatletter and \makeatother commands.

- \CF@arrow@start@name and \CF@arrow@end@name include the names of the compounds (considered as nodes by ti*kz*) between which the arrow is drawn;
- \CF@arrow@start@node and \CF@arrow@end@node include the node names where arrow ends will be located. After these names, user-defined anchors can be specified in the argument between brackets of the \arrowcommand, unless the field is left empty;
- \CF@arrow@current@style and \CF@arrow@current@angle contain the style and the angle of the arrow to be drawn;
- \CF@arrow@shift@nodes{<dim>} shifts the nodes "\CF@arrow@start@node" and "\CF@arrow@end@node" perpendicularly relative to the arrow by a dimension specified in the argument;
- \CF@arrow@display@label{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8} is the most complex one. It gives the labels position with the following arguments:
  - #1 and #5 are the labels to be written;
  - #2 and #6 are real numbers between 0 and 1. They specify the location of the labels on the arrow. 0 is the beginning of the arrow and 1 is its end, assuming a *straight* arrow;
  - #3 and #7 are the "+" or "-" characters. "+" displays the label above the arrow, while "-" does it below it;
  - #4 and #8 are the names of the nodes corresponding to the beginning and the end of the arrow.
- arrow ends are either "CF@full"[14] for a full arrow or "CF@half" for an upper-half arrow.

10.1   First arrow

As an example, assume we want to make an arrow with a circle on its center. Let's call it "-.>". This arrow will accept four optional arguments. Like for previously-defined arrows, the first and second arguments will be the labels to be located above and below the arrow. The third one will define the perpendicular shift relative to the

---

[14]this one is similar to the ti*kz*-defined arrow : stealth.

arrow direction. Finally, the 4th argument will define the circle size. If this last argument is absent the default circle size will be equal to 2pt.

Let's start with \definearrow{4}{-.>} to declare that the arrow will have 4 optional arguments and that it will be called -.>. First, the position of the nodes between which the arrow is to be drawn must be modified in order to take the third-argument shift into account. This is made with the macro \CF@arrow@shift@nodes, so the code of the arrow will start with: \CF@arrow@shift@nodes{#3}%. Then, one must plot the arrow itself, while taking the opportunity to set a node on the center of the segment, which will be called "mid@point". Finally, the circle is defined with its center on that node. The whole ti*kz* code is:

```
\edef\pt@radius{\ifx\@empty#4\@empty 2pt\else #4\fi}% circle radius
\expandafter\draw\expandafter[\CF@arrow@current@style,-CF@full]
    (\CF@arrow@start@node)--(\CF@arrow@end@node)coordinate[midway](mid@point);
\filldraw(mid@point)circle(\pt@radius);%
```

The last step is to enter the labels, if any, with the folwing line:

```
\CF@arrow@display@label{#1}{0.5}{+}{\CF@arrow@start@node}{#2}{0.5}{-}{\CF@arrow@end@node}
```

Here is the completed arrow:

```
\makeatletter
\definearrow4{-.>}{%
  \edef\pt@radius{\ifx\@empty#4\@empty 2pt\else #4\fi}% dot radius
  \CF@arrow@shift@nodes{#3}%
  \expandafter\draw\expandafter[\CF@arrow@current@style,-CF@full](\CF@arrow@start@node)--(\CF@arrow@end@node)
    coordinate[midway](mid@point);
  \filldraw(mid@point)circle(\pt@radius);%
  \CF@arrow@display@label{#1}{0.5}{+}{\CF@arrow@start@node}{#2}{0.5}{-}{\CF@arrow@end@node}
  }
\makeatother
\schemestart
A \arrow{-.>} B \arrow{-.>[above][below][][1pt]} C \arrow{-.>[][below]}[30] D \arrow{-.>[above][][5pt][1.5pt]} E
\schemestop
```


10.2   Curved arrow

How about a curved arrow? To make things as simple as possible, assume it will have one single optional argument with the ti*kz* code that will specify the point(s) of control. If this argument is empty, a "->" type arrow will be plotted.

If #1 is not empty, attention should not be drawn to "\CF@arrow@start@node" and "\CF@arrow@end@node" which contain the node names of arrow ends positions, because the location of these nodes is already determined by the anchors calculated for *straight* arrows! Instead we will use \CF@arrow@start@name and \CF@arrow@end@name which contain the names of the compound (which are nodes for ti*kz*), since the arrow must be plotted between them. Here's the ti*kz* code to draw the curved arrow between the two compounds:

\draw[shorten <=\CF@arrow@offset,shorten >=\CF@arrow@offset,\CF@arrow@current@style,-CF@full, (\CF@arrow@start@name).. controls #1 ..(\CF@arrow@end@name);%

One must add a ti*kz* code to shorten the arrowby an amount \CF@arrow@offset defined by \setarrowoffset. Indeed, the nodes ar not the same as those for straight arrows (\CF@arrow@start@node and \CF@arrow@end@node). So before \CF@arrow@current@style, the follwing code must be added:

shorten <=\CF@arrow@offset, shorten >=\CF@arrow@offset

this is the role the two lines after \else.

So here is our curved arrow:

Curved arrow

```
\makeatletter
\definearrow1{s>}{%
\ifx\@empty#1\@empty
  \expandafter\draw\expandafter[\CF@arrow@current@style,-CF@full](\CF@arrow@start@node)--(\CF@arrow@end@node);%
\else
  \def\curvedarrow@style{shorten <=\CF@arrow@offset,shorten >=\CF@arrow@offset,}%
  \CF@expadd@tocs\curvedarrow@style\CF@arrow@current@style
  \expandafter\draw\expandafter[\curvedarrow@style,-CF@full](\CF@arrow@start@name)..controls#1..(\CF@arrow@end@name);
\fi
}
\makeatother
\schemestart
A\arrow{s>}
B\arrow{s>[+(0.5cm,0.5cm)]}
C\arrow{s>[+(45:1cm)]}
D\arrow(.60--.120){s>[+(60:1cm) and +(-120:1cm)]}
E\arrow{s>[+(45:1) and +(-135:1)]}
F\arrow{s>[+(-30:1) and +(150:1)]}[,1.5]
G\arrow(.90--.90){s>[+(60:1)and+(120:1)]}[,2]
H
\schemestop

\schemestart
A\arrow(.90--.180){s>[+(90:0.8) and +(180:0.8)]}[45]B
\arrow(.0--.90){s>[+(0:0.8) and +(90:0.8)]}[-45]C
\arrow(.-90--.0){s>[+(-90:0.8) and +(0:0.8)]}[-135]D
\arrow(.180--.-90){s>[+(180:0.8) and +(-90:0.8)]}[135]
\schemestop
```
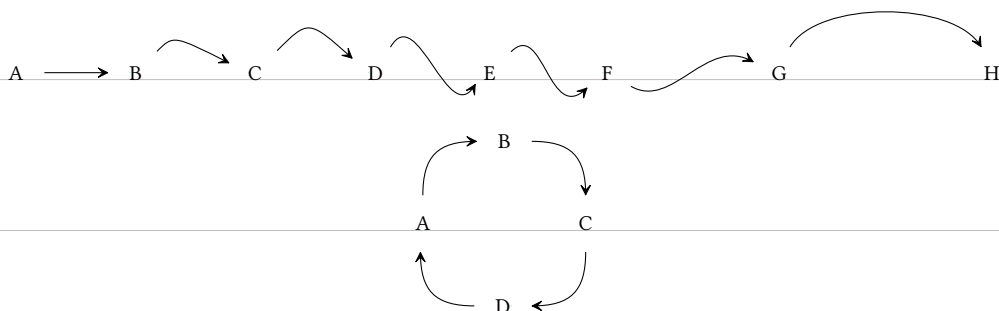

11   The \merge command

The \merge command allows to draw arrows coming from several existing compounds that merge into one single arrow that arrive to one single compounds.

Just after the \merge command, the direction that follows up must be specified. For this, 4 different direction characters can be used: ">" (the default direction if no character is entered), "<", "^" and "v".

The syntax follows with:

$$\merge\{dir\}(n1.a1)(n2.a2)(...)(ni.ai)--(n.a[s])$$

where the "ni" names before the double dash are those already-defined compounds from which outcoming arrows will merge into a single one. One can also specify the "ai" anchor, when the default one is not convenient. Like for the \arrowcommand, the command "n.a[s]" includes the name, the anchor and the style of the target compound.

The \merge command

```
\schemestart
ABC\arrow[30]EFGHIJ\arrow[45]KLM\arrow[60]NO
\merge>(c1)(c2)(c3)--()series 1
\arrow series 2
\schemestop
\bigskip

\schemestart
Foooo\arrow(foo--bar){<=>}Bar\arrow(--baz){<=>}Bz
\merge^(foo)(bar)(baz)--()series
\schemestop
\bigskip

\schemedebug{true}
\schemestart
A\arrow{<->}[90]B
\merge<(c1.120)(c2)--(foobar.45[circle,blue])CCC
\schemestop
```
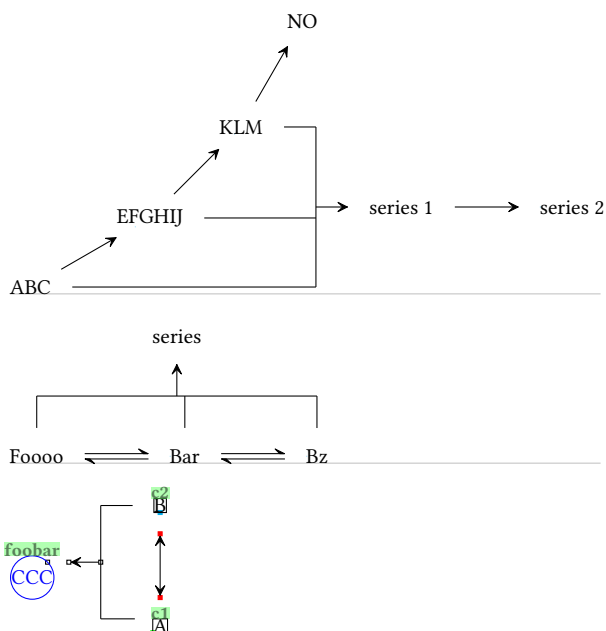
Regarding the geometry of the \merge arrow, it consists of n segments leaving from n compounds up to the perpendicular line that connects them: the default length of the shortest of these segments is worth half of the compound-spacing distance defined by \setcompoundsep. The arrow drawn from the connecting line to the target compound has the same default length, its origin is on the middle of the connecting line. These three geometric features can be customized with the optional argument immediately after the target compound:

$$\merge\{dir\}(n1.a1)(n2.a2)(...)(ni.ai)--(n.a[s])[c1,c2,c,style]$$

where:

- the shortest segment distance between reactants and the connecting line is controlled through the multiplication of the \setcompoundsep distance by a coefficient $c1$, whose default value is 0.5;
- the length of the arrow between the connecting line and the product compound is controlled through the multiplication of the \setcompoundsep distance by a coefficient $c2$, whose default value is 0.5;
- the origin of the arrow between the connecting line and the product compound is determined by the coefficient c, a value of 0 involves a departure from the the left of the connect line (or from its top if the direction is v or ^);
- the style of the \merge arrow is defined with the last argument: style.
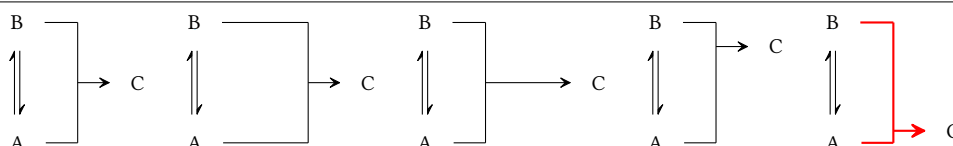
Geometrical parameters of \merge
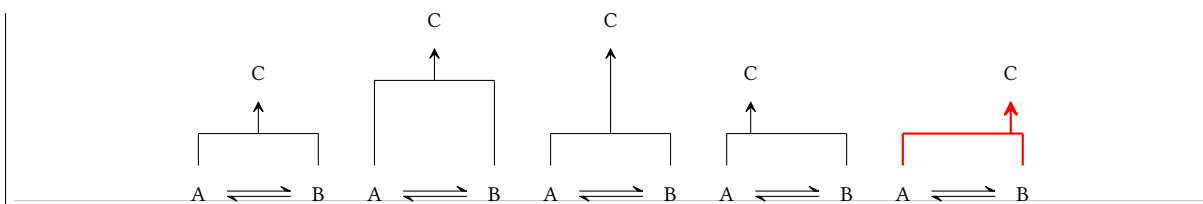
```
\schemestart A\arrow{<=>}[90]B\merge(c1)(c2)--()C\schemestop\qquad
\schemestart A\arrow{<=>}[90]B\merge(c1)(c2)--()[1]C\schemestop\qquad
\schemestart A\arrow{<=>}[90]B\merge(c1)(c2)--()[,1]C\schemestop\qquad
\schemestart A\arrow{<=>}[90]B\merge(c1)(c2)--()[,,0.2]C\schemestop\qquad
\schemestart A\arrow{<=>}[90]B\merge(c1)(c2)--()[,,0.9,red,thick]C\schemestop
\bigskip

\schemestart A\arrow{<=>}B\merge^(c1)(c2)--()C\schemestop\qquad
\schemestart A\arrow{<=>}B\merge^(c1)(c2)--()[1]C\schemestop\qquad
\schemestart A\arrow{<=>}B\merge^(c1)(c2)--()[,1]C\schemestop\qquad
\schemestart A\arrow{<=>}B\merge^(c1)(c2)--()[,,0.2]C\schemestop\qquad
\schemestart A\arrow{<=>}B\merge^(c1)(c2)--()[,,0.9,red,thick]C\schemestop
```
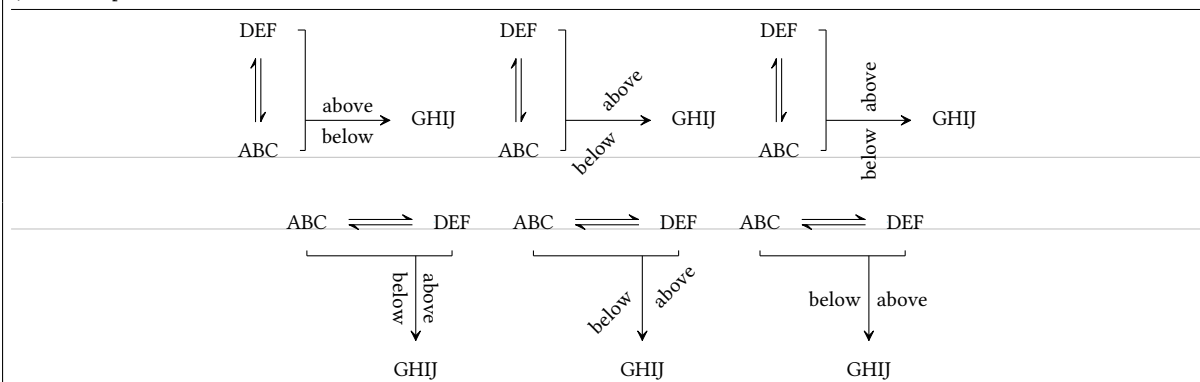
Finally, it is possible to write labels above or below the merged arrow. For this, the direction character accepts two optional arguments in brackets, a first one for the label above the arrow and a second one for the label below it. Therefore, the full syntax of the merge command is:

\merge{dir}[labelup][labeldow](n1.a1)(n2.a2)(...)(ni.ai)--(n.a[s])[c1,c2,c,style]

All the features introduced before for arrow labelling can be implemented here as well, i.e. rotation angle and anchoring with the syntax *{angle.anchor} entered just before the content of the label.

Labels of the \merge command

```
\schemestart
ABC\arrow{<=>}[90]DEF\merge>[above][below](c1)(c2)--()[0.25,1,0.75]GHIJ
\schemestop\qquad
\schemestart
ABC\arrow{<=>}[90]DEF\merge>[*{45.south west}above][*{45.north east}below](c1)(c2)--()[0.25,1,0.75]GHIJ
\schemestop\qquad
\schemestart
ABC\arrow{<=>}[90]DEF\merge>[*{90}above][*{90}below](c1)(c2)--()[0.25,1,0.75]GHIJ
\schemestop
\bigskip

\schemestart
ABC\arrow{<=>}DEF\merge v[above][below](c1)(c2)--()[0.25,1,0.75]GHIJ
\schemestop\qquad
\schemestart
ABC\arrow{<=>}DEF\merge v[*{45.north west}above][*{45.south east}below](c1)(c2)--()[0.25,1,0.75]GHIJ
\schemestop\qquad
\schemestart
ABC\arrow{<=>}DEF\merge v[*{0}above][*{0}below](c1)(c2)--()[0.25,1,0.75]GHIJ
\schemestop
```


---

12   The + sign

The use of a "\+" macro that displays a + sign is available between the commands \schemestart and \schemestop. This macro accepts an optional argument in braces with 3 dimensions in the form {<dim1>,<dim2>,<dim3>}, where:
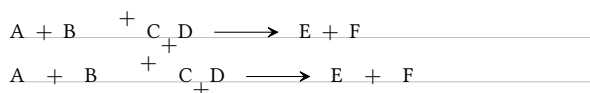
- <dim1> and <dim2> are the dimensions to be inserted before and after the + sign;
- <dim3> is the vertical offset of the sign.

If one argument is left empty or if the argument in braces is missing, the dimensions take their default values, which can be specified with the \setandsign{<dim1>,<dim2>,<dim3>} command. An empty field leads <dim1> and <dim2> to be worth 0.5em, while <dim3> becomes 0pt empty. The default values {0.5em,0.5em,0pt} are used when the user does not specify different ones.

```
                              ┌─────── The \+ command ───────┐
\schemestart
A\+B\+{2em,,5pt}C\+{0pt,0pt,-5pt}D\arrow E\+F
\schemestop

\setandsign{1em,1em,0pt}
\schemestart
A\+B\+{2em,,5pt}C\+{0pt,0pt,-5pt}D\arrow E\+F
\schemestop
```

$$A \;+\; B \quad \overset{+}{} \quad C \underset{+}{\,} D \longrightarrow E \;+\; F$$

$$A \;+\; B \quad \overset{+}{} \quad C \underset{+}{\,} D \longrightarrow E \;+\; F$$

As shown in the example below, it should be kept in mind that the $+$ sign inserted by the \+ command is part of the compound:

```
                              ┌─────── Compounds and \+ ───────┐
\schemedebug{true}
\schemestart A\+ B\+{,,5pt}C\arrow D\+ E\schemestop
```
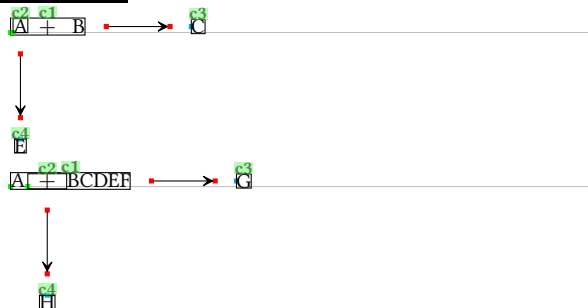
$$\overset{c1}{\boxed{A \;+\; B \;\overset{+}{}\; C}} \longrightarrow \overset{c2}{\boxed{D \;+\; E}}$$

This makes it difficult to draw a vertical arrow exactly below the letter "A" since this letter is not a single compound for ChemFig. This issue can be solved with the use of the \subscheme command to uniquely define the letter "A" as a single compound (the same procedure can be applied to the $+$ sign itself) so that it can be referred to later on with its own name:

```
                              ┌─────── Subcompound and \+ ───────┐
\schemedebug{true}
\schemestart
\subscheme{A}\+ B\arrow C
\arrow(@c2--)[-90]E
\schemestop
\medskip

\schemestart
A\subscheme{\+}BCDEF \arrow G
\arrow(@c2--)[-90]H
\schemestop
```

A common problem can be the misalignment of the "+" sign with the molecules before or after it. For example:

```
                              ┌─────── + sign alignment ───────┐
\schemedebug{true}
\schemestart
  \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
  \+
  \chemfig{\lewis{246,Br}-\lewis{026,Br}}
\schemestop
```
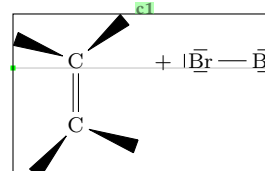
Here, the "+" sign sits on the same baseline as the compound before it, and this baseline is that of the top carbon atom. One may shift the "+" sign, but this would not change the vertical position of "|$\overline{\underline{Br}}$ —— $\overline{\underline{Br}}$|". In fact, the "+" sign does not prevent ChemFig from reading a compound, as shown in the example above where everything is included in the compound " c1". Therefore, one must stop the compound right after the first molecule with a \arrow{0}[,0] that will draw an invisible, zero-length arrow. In order to vertically center the whole scheme, one must also set the the anchor of the first compound as "west" (or "180", which is a synonym) with the second optional argument of the \schemestart command:

```
                              ┌─────── + sign alignment ───────┐
\schemedebug{true}
\schemestart[][west]
  \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
  \arrow{0}[,0]\+
  \chemfig{\lewis{246,Br}-\lewis{026,Br}}
\schemestop
```
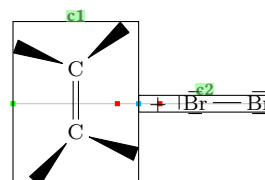
Thus, the first compound " c1" consists of the first molecule and the second compound consists of everything else, i.e. the "+" sign and the second molecule. Alternatively, one can play with anchors or styles via the \arrow command to move the second compound to another location. Here, for example, the second compound is shifted downwards by 10pt in the first case. In the second case, the "south east" anchor of the first compound matches the "south west" anchor of the second one:

┌─ + sign alignment ──

```
\schemedebug{true}
\schemestart[][west]
  \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
  \arrow(--[yshift=-10pt]){0}[,0]\+
  \chemfig{\lewis{246,Br}-\lewis{026,Br}}
\schemestop
\medskip

\schemestart[][west]
  \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
  \arrow(.south east--.south west){0}[,0]\+
  \chemfig{\lewis{246,Br}-\lewis{026,Br}}
\schemestop
```
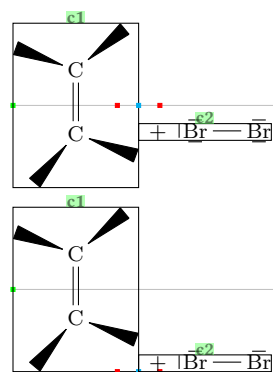
PART V

List of commands

The commands created by ChemFig are:

Commands	Description
\chemfig<code>	draws the molecule whose design is described by the <code>
\printatom	displays the atoms within the molecules. It can be redefined to customize the output. See page 23
\setnodestyle{<style tikz>}	using ti*kz* syntax, this macro defines the style of nodes containing the atoms. See page 32
\setbondestyle{<style tikz>}	with the ti*kz* syntax, this macro defines the style of the bonds. See page 6
\hflipnext	the next molecule will be horizontally flipped
\vflipnext	the next molecule will be vertically flipped
\definesubmol{<nom>}[code1]{<code2>}	creates an alias !<nom> which can be put in the code of molecules to be drawn, and which will be replaced with <code1> or <code2> depending on the angle of the last bond. See page 27
\chemskipalign	tells the vertical alignment mechanism to ignore the current group of atoms. See page 25.
\redefinesubmol{<nom>}{<code>}	replaces a preexisting alias !<name> with the new <code>. See page 28
\setcrambond{<dim1>}{<dim2>}{<dim3>}	sets the dimensions of the triangles representing Cram bonds: <dim1> is the size of the base, <dim2> is the spacing between the dashes and <dim3> is the side of the dashes. See page 7
\setatomsep{<dim>}>	sets the interatomic distance. See page 6
\setbondoffset{<dim>}	sets the space between bonded atoms and the bond. See page 6
\setdoublesep{<dim>}	sets the spacing between the two lines of a double bond. See page 6
\lewis[coeff]{<codes>,<atome>}	displays the <atom> and places Lewis dot decorations as specified in the <code>. The dots drawn do not change the bounding box. See page 28
\Lewis[coeff]{<codes>,<atome>}	displays the <atom> and places Lewis dot decorations as specified in the <code>. See page 28
\setlewis{<dim1>}{<dim2>}{<code tikz>}	sets the Lewis dot parameters; <dim1> is the distance between the atoms and the decoration, <dim2> is the length of the line representing the pair of electrons and <tikz code> is code which is passed directly to ti*kz*. See page 29
\setlewisdist{dim}	sets the distace between the 2 disks representing a pair of electrons. See page 30
\chemmove[<tikz options>]<tikz code>	Makes a tikzpicture environment, adding to it the <tikz options>. Uses the <tikz code> to join the nodes specified in the molecules with the help pf the "@" character. See page 18.
\chemsign[<dim>]<sign>	draws the <sign>, placing before and after it an unbreakable space of length <dim>. See page 31
\chemrel[<txt1>][<txt2>]{<arrow code>}	draws an arrow described by its <arrow code> and places the optional text <txt1> and <txt2> above and below the arrow respectively. See page 31
\setchemrel{<dim1>}{<dim2>}{<dim3>}	sets the dimensions of the arrow drawn with the command \chemrel: <dim1> is the vertical space between the arrow and the optional text, <dim2> is the unbreakable horizontal space inserted before and after the arrow and <dim3> is the length of the arrow. See page 31
\chemabove[<dim>]{<txt1>}{txt2}	writes <txt1> and places <txt2> above, leaving <dim> of vertical space. This command does not change the bounding box of <txt1>. See page 30

\chembelow[<dim>]{<txt1>}{txt2}	writes {txt1} and places <txt2> below, leaving <dim> of vertical space. This command does not change the bounding box of <txt1>. See page 30
\Chemabove[<dim>]{<txt1>}{txt2}	writes <txt1> and places <txt2> above, leaving <dim> of vertical space. See page 30
\Chembelow[<dim>]{<txt1>}{txt2}	writes {txt1} and places <txt2> below, leaving <dim> of vertical space. See page 30
\chemname[<dim>]{<molecule>}{<name>}	Places <name> under the <molecule>
\chemnameinit	Initializes the greatest molecule depth to ensure correct alignment of the names of the following molecules.

\schemestart...\schemestop	commands between which a reaction scheme is drawn. See page 47.
\arrow	draws an arrow in a reaction scheme (this command is only defined inside a reaction scheme). See page 48.
\+	prints a + sign in a reaction scheme (this command is only defined inside a reaction scheme). See page 61.
\subscheme{<code>}	draws a subscheme (this command is only defined inside a reaction scheme). Voir 52.
\definearrow	defines an arrow. See page 57.
\chemleft<car1><stuff>\chemright<car1>	draws expandable delimiters defined with <car1> and <car2> on the left and on the right of the <stuffl>, see page 53.
\chemup<car1><matériel>\chemdown<car1>	draws expandable delimiters defined with <car1> and <car2> above and below the <stuff>, voir page 53.
\setcompoundsep{<dim>}	sets the space between the edges of the compounds in a reaction scheme. See page 47.
\setarrowoffset{<dim>}	sets the space between the edges of the compound and the arrow. See page 47.
\setarrowdefault{angle,coeff,style}	sets the default settings of the arrow. See page 48.
\setcompoundstyle{<code tikz>}	set the default style of the compounds. See page 51.
\setarrowlabelsep{<dim>}	sets the space between the arrow and the anchor of its lables. See page 55.
\setandsign{<dim1>,<dim2>,<dim3>}	sets the defaiult setting of the + sign where <dim1> and <dim2> are the dimensions before and after the sign while <dim3> is the vertical shift. See page 61.

PART VI

Gallery

This manual concludes with drawings of molecules of varying complexity.

The curious user can look at the <code> of each molecule, though it does become less attractive the more complex the molecule gets. Indeed, beyond a certain level of complexity, though it it is fairly easy to write <code>, it becomes much harder to read the <code> to analyze it afterwards. We quickly reached the limits of immediate readability of the code of a complex drawing.

Anyway, I hope that this package will help all LaTeX users wishing to draw molecules. Although ChemFig has been thoroughly tested and although its version number is now greater than 1.0, I hope that you will be forgiving with bugs you encounter and send me an **email** to let me know of any malfunctions or suggestions for improvement.

Christian TELLECHEA

⋆
⋆ ⋆
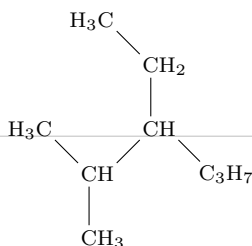
2-methylpentane

```
\chemfig{[7]H_3C-CH(-[6]CH_3)-[1]CH_2-CH_2-[1]CH_3}
```
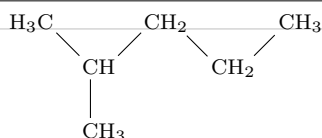

3-ethyl-2-methylhexane

```
\chemfig{H_3C-[7]CH(-[6]CH_3)-[1]CH(-[7]C_3H_7)-[2]CH_2-[3]H_3C}
```


Stearine, condensed structural diagram

```
\definesubmol{@}{([0,2]-O-[0,1]C(=[2,1]O)-C_{17}H_{33})}
\chemfig{[2,2]CH_2!@-CH_{\phantom 2}!@-CH_2!@}
```

$$\text{CH}_2 \text{—} \text{O} \text{—} \overset{\displaystyle \text{O}}{\underset{\displaystyle |}{\text{C}}} \text{—} \text{C}_{17}\text{H}_{33}$$

$$\text{CH} \text{—} \text{O} \text{—} \overset{\displaystyle \text{O}}{\underset{\displaystyle |}{\text{C}}} \text{—} \text{C}_{17}\text{H}_{33}$$

$$\text{CH}_2 \text{—} \text{O} \text{—} \overset{\displaystyle \text{O}}{\underset{\displaystyle |}{\text{C}}} \text{—} \text{C}_{17}\text{H}_{33}$$
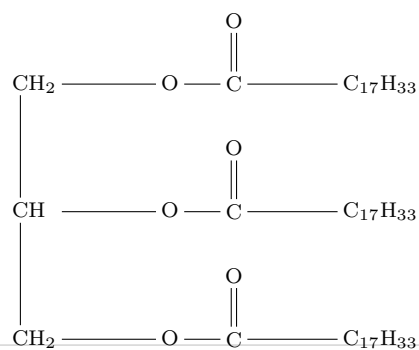
Stearine, skeleton diagram

```
\definesubmol{x}{-[:+30,.6]-[:-30,.6]}
\definesubmol{y}{-O-(=[2,.6]O)-!x!x!x!x!x!x!x!x}
\chemfig{[2]([0]!y)-[,1.5]([0]!y)-[,1.5]([0]!y)}
```
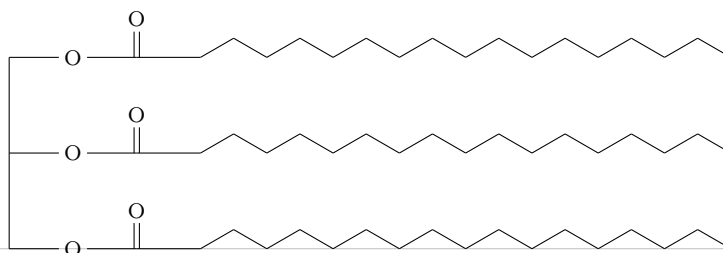
Methyl 2-methylpropanoate

```
\chemfig{H_3C-CH_2(-[2]CH_3)-C(=[1]O)-[7]O-CH_3}
```
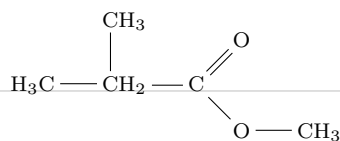
Vanillin

```
\chemfig{HC*6(-C(-OH)=C(-O-[::-60]CH_3)-CH=C(-[,,,2]HC=[::-60]O)-HC=[,,2])} \quad or \quad
\chemfig{*6(-(-OH)=(-OCH_3)-=(-=[::-60]O)-=)}
```
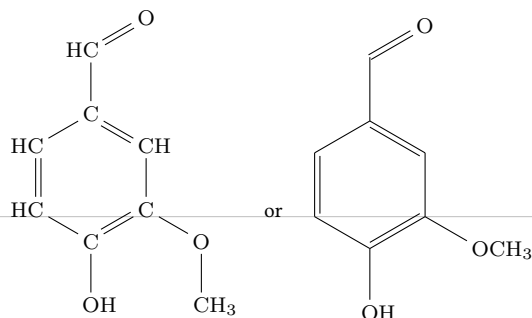
Caffeine
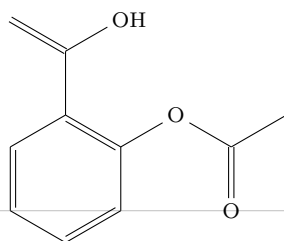
```
\chemfig{*6((=O)-N(-CH_3)-*5(-N=-N(-CH_3)-=)--(=O)-N(-H_3C)-)}
```
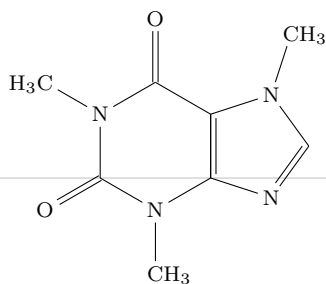
Aspirin
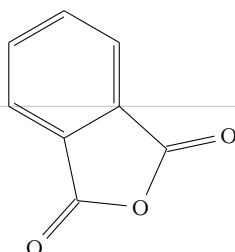
```
\chemfig{*6(-=-(-O-[::-60](=[::-60]O)-[::+60])=(-(=[::+60])-[::-60]OH)-=)}
```

Aspirin is a registered trademark of Bayer in many countries.
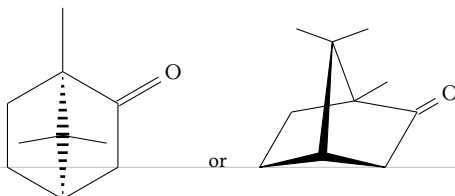
Phthalic anhydride

```
\chemfig{*6(=*5(-(=O)-O-(=O)-)-=-=-)}
```
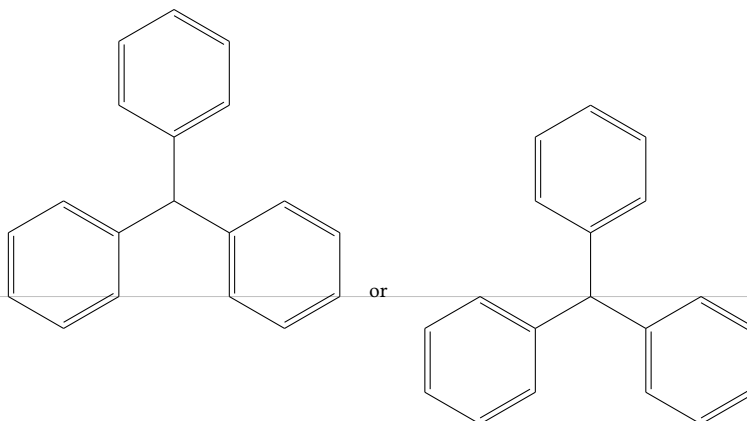
Camphor

```
\chemfig{*6(-(<:[::120](-[::-100,0.7])(-[::100,0.7]))--(=O)-(-)(<:[::120])--)}
\quad or \quad
\setcrambond{3pt}{}{}
\chemfig{<[:10](>[:85,1.8]?(-[:160,0.6])-[:20,0.6])
>[:-10]-[:60](=[:30,0.6]O)-[:170]?(-[:30,0.6])-[:190]-[:240]}
```

Triphenylmethane

```
\chemfig{*6(-=-*6(-(-*6(=-=-=-))-*6(=-=-=-))=-=)}
\quad or \quad
\definesubmol{@}{*6(=-=-=-)}
\chemfig{(-[:-30]!@)(-[:90]!@)(-[:210]!@)}
```

Amygdalin
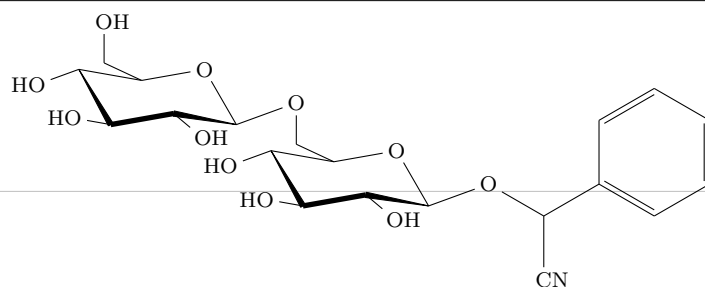
```
\setcrambond{2pt}{}{}
```

```
\definesubmol{c1}{-[:200]-[:120]O-[:190]}
\definesubmol{c2}{-[:170](-[:200,0.7]HO)<[:300](-[:170,0.6]HO)
-[:10,,,,line width=2pt](-[:-40,0.6]OH)>[:-10]}
\definesubmol{csub}{-[:155,0.65]-[:90,0.65]}
\chemfig{O(!{c1}(!{csub}O(!{c1}(!{csub}OH)!{c2}))!{c2})-[:-30](-[:-90]CN)-[:30]*6(=-=-=-)}
```
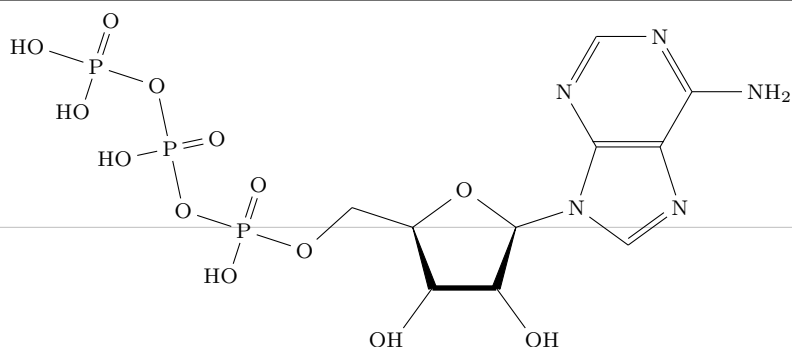
```
\setcrambond{3pt}{}{}
\definesubmol{a}{-P(=[::-90,0.75]O)(-[::90,0.75]HO)-}
\chemfig{[:-54]*5((--[::60]O([::-60]!aO([::-60]!aO([::60]!aHO))))<(-OH)
-[,,,,line width=2pt](-OH)>(-N*5(-=N-*6(-(-NH_2)=N-=N-)=_-))-O-)}
```
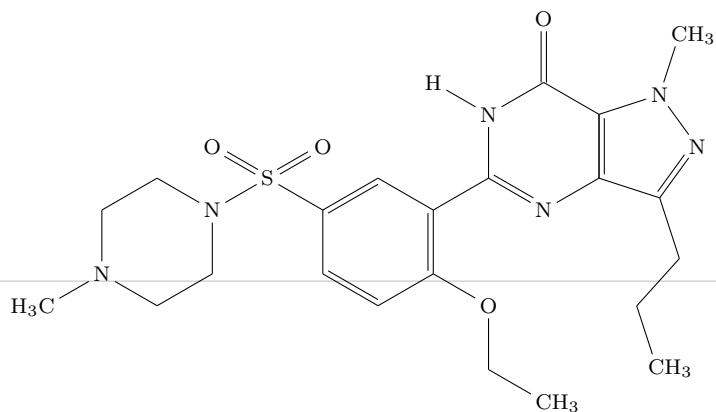
```
\chemfig{N*6((-H_3C)---N(-S(=[::+120]O)(=[::+0]O)-[::-60]*6(-=-(-O-[::-60]-[::+60]CH_3)
=(-*6(=N-*5(-(--[::-60]-[::+60]CH_3)=N-N(-CH_3)-=)--(=O)-N(-H)-)-=)))---)}
```
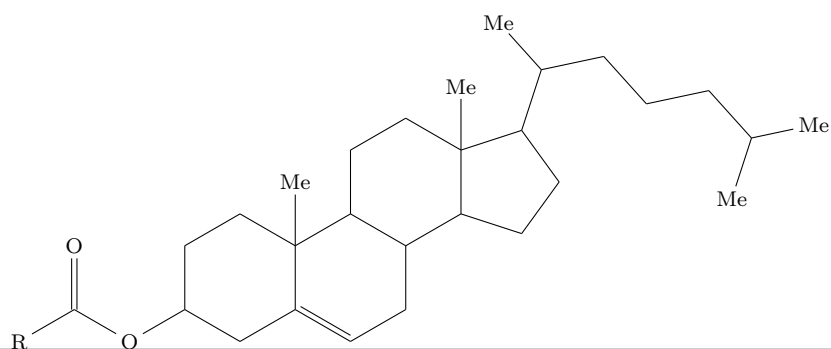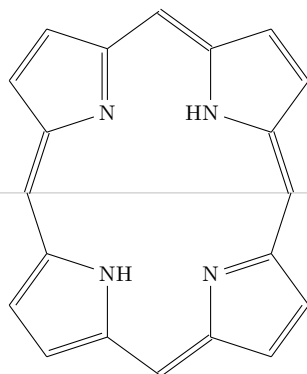
```
\chemfig{[:30]R-(=[::+60]O)-[::-60]O-*6(--*6(=--*6(-*5(---(-(-[::+60]Me)
-[::-60]-[::-60]-[::+60]-[::-60](-[::-60]Me)-[::+60]Me)-)-(-[::+0]Me)---)--)-(-[::+0]Me)---)}
```

\chemfig{?=[::+72]*5(-N=(-=[::-72]*5(-[,,,2]HN-[,,2](=-[::-36]*5(=N-(=-[::-72]*5(-NH-[,,1]?=-=))
-=-))-=-))-=-)}
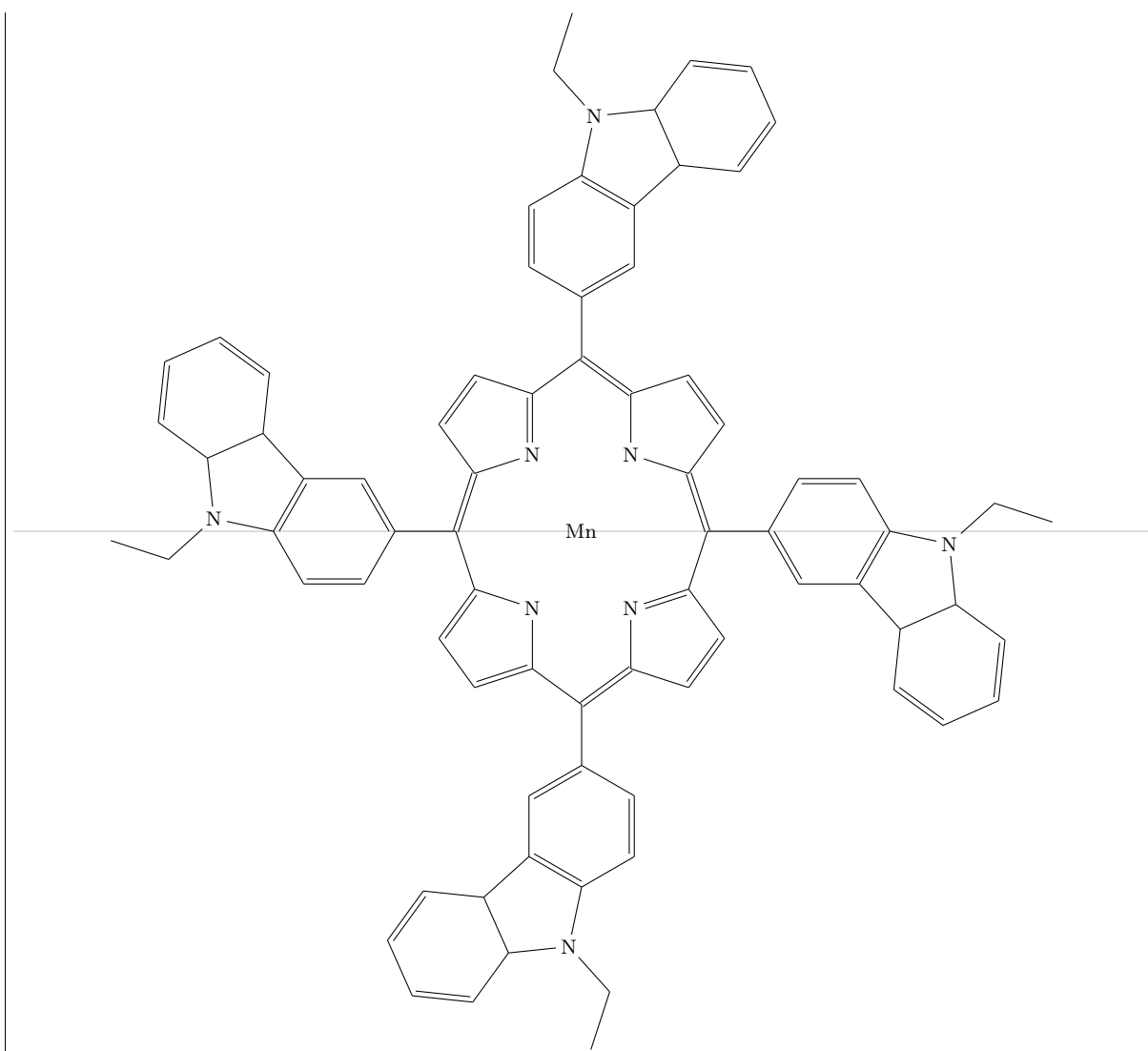
\definesubmol{A}{*6(=-*5(-*6(-=-=-)--N(--[::-60])-)=-=-)}
\chemfig{([::+180]-!A)=[::+72]*5(-N=(-(-[::+54]!A)=[::-72]*5(-N(-[::-33,1.5,,,draw=none]Mn)
-(=(-[::+72]!A)-[::-36]*5(=N-(=(-[::+54]!A)-[::-72]*5(-N-(-)=-=))-=-))-=-))-=-)}

```
\chemfig{[:-90]HN(-[::-45](-[::-45]R)=[::+45]O)>[::+45]*4(-(=O)-N*5(-(<:(=[::-60]O)
-[::+60]OH)-(<[::+0])(<:[::-108])-S>)--)}
```
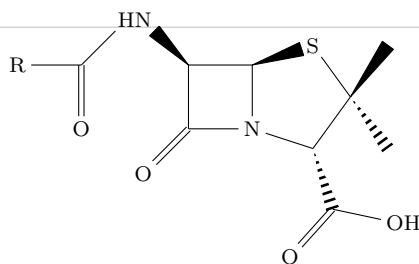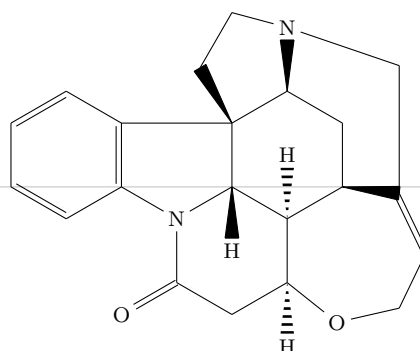
```
\chemfig{[:150]?*6(=*6(--*6(-N(-CH_3)--(<(=[::+60]O)-[::-60]N(-[::+60]-[::-60])
-[::-60]-[::+60])-=)([::-120]<H)---)-*6(-=-=-(-[::-30,1.155]\chembelow{N}{H}?)=)))}
```
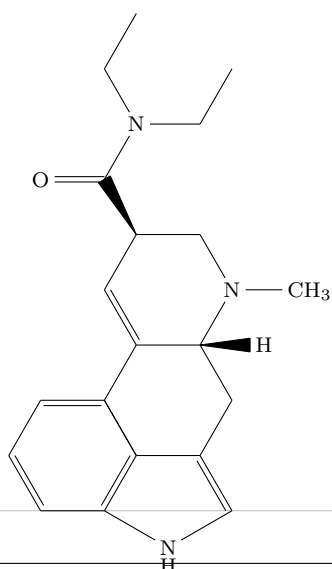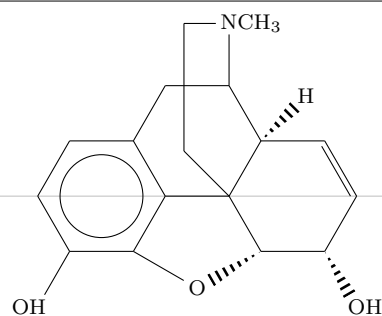
Strychnine

```
\chemfig{*6(=-*6(-N*6(-(=O)--([::-120]<:H)*7(-O--=?[0]([::-25.714]-[,2]?[1]))
-*6(-?[0,{>}]--(<N?[1]?[2])-(<[::-90]-[::-60]?[2]))(<:[::+0]H)-([::+120]<H))--?)=?-=-)}
```
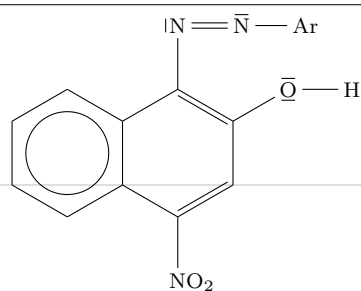

Codeine

```
\chemfig{[:-30]**6(-(-OH)-?-*6(-(-[3]-[2,2]-[0,.5])*6(-(<:[:-150,1.155]O?)
-(<:OH)-=-)-(<:[1]H)-(-[2]NCH_3)--)---)}
```


A dye (red)

```
\chemfig{**6(--*6(-(-NO_2)=-(-\lewis{26,O}-[0]H)=(-\lewis{4,N}=[0]\lewis{2,N}-[0]Ar)-)----)}
```

Menthone

```
\chemfig{CH_3-?(-[2]H)(-[::-30,2]-[::+60](=[1]\lewis{20,0})
-[::-150,1.5](-[:20]CH(-[1]CH_3)(-[7]CH_3))(-[6]H)-[::-90,2]-[::+60]?)}
```
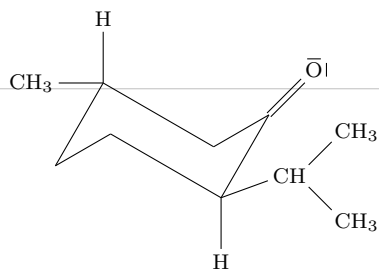
Fischer indole synthesis

```
\schemestart
  \chemfig{*6(=-*6(-\chembelow{N}{H}-NH_2)=-=-)}
  \+
  \chemfig{(=[:-150]O)(-[:-30]R_2)-[2]-[:150]R_1}
  \arrow(.mid east--.mid west){->[\chemfig{H^+}]}
  \chemfig{*6(-=*5(-\chembelow{N}{H}-(-R_2)=(-R_1)-)-=-=)}
\schemestop
```
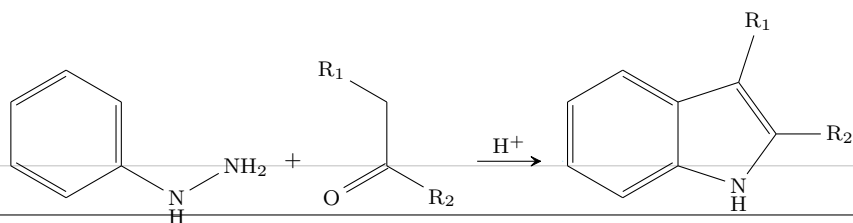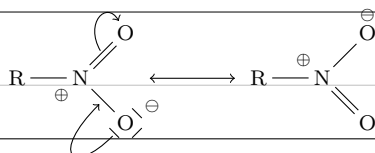
Reaction mechanisms: carbonyl group

```
\chemfig{C([3]-)([5]-)=[@{db,.5}]@{atoo}\lewis{06,0}}
\chemrel{<>}
\chemfig{\chemabove{C}{\scriptstyle\oplus}([3]-)([5]-)-\chemabove
    {\lewis{026,0}}{\hspace{5mm}\scriptstyle\ominus}}
\chemmove{\draw[->,shorten <=2pt, shorten >=2pt](db) ..controls +(up:5mm) and +(up:5mm)..(atoo);}
```

Reaction mechanisms: nitro group

```
\chemfig{R-\chembelow{N}{\hspace{-5mm}\scriptstyle\oplus}([1]=[@{db}]@{atoo1}O)([7]-[@{sb}]@{atoo2}
    \chemabove{\lewis{157,0}}{\hspace{7mm}\scriptstyle\ominus})}
\chemrel{<->}
\chemfig{R-\chemabove{N}{\hspace{-5mm}\scriptstyle\oplus}([1]-\chemabove{O}{\scriptstyle\ominus})([7]=O)}
\chemmove{
    \draw[->,shorten <=2pt, shorten >=2pt](db) ..controls +(120:5mm) and +(120:5mm)..(atoo1);
    \draw[->,shorten <=3pt, shorten >=2pt](atoo2) ..controls +(225:10mm) and +(225:10mm)..(sb);
}
```

Nucleophilic addition. Primary amines

```
\setatomsep{2.5em}
\setcompoundsep{5em}
\schemestart
\chemfig{R-@{aton}\lewis{2,N}H_2}
\+
\chemfig{@{atoc}C([3]-CH_3)([5]-CH_3)=[@{atoo1}]O}
\chemfig{@{atoo2}\chemabove{H}{\scriptstyle\oplus}}
\chemmove[-stealth,shorten <=3pt,dash pattern= on 1pt off 1pt,thin]{
```

```
    \draw[shorten >=2pt](aton) ..controls +(up:10mm) and +(left:5mm)..(atoc);
    \draw[shorten >=8pt](atoo1) ..controls +(up:10mm) and +(north west:10mm)..(atoo2);}
\arrow{<=>[\tiny addition]]
\chemfig{R-@{aton}\chembelow{N}{\scriptstyle\oplus}H([2]-[@{sb}]H)-C(-[2]CH_3)(-[6]CH_3)-OH}
\schemestop
\chemmove{
    \draw[-stealth,dash pattern= on 1pt off 1pt,shorten <=3pt, shorten >=2pt]
    (sb)..controls +(left:5mm) and +(135:2mm)..(aton);}
\par
\schemestart
\arrow{<=>}
\chemfig{R-@{aton}\lewis{2,N}([6]-[@{sbh}]H)-[@{sb}]C(-[2]CH_3)(-[6]CH_3)-[@{sbo}]@{atoo}
\chemabove{0}{\scriptstyle\oplus}(-[1]H)(-[7]H)}
\chemmove[-stealth,shorten <=3pt,shorten >=2pt,dash pattern= on 1pt off 1pt,thin]{
    \draw(aton) ..controls +(up:5mm) and +(up:5mm)..(sb);
    \draw(sbh) ..controls +(left:5mm) and +(south west:5mm)..(aton);
    \draw(sbo) ..controls +(up:5mm) and +(north west:5mm)..(atoo);}
\arrow{<=>[\tiny élimination]]\chemfig{R-N=C(-[1]CH_3)(-[7]CH_3)}
\+
\chemfig{H_3\chemabove{0}{\scriptstyle\oplus}}
\schemestop
```
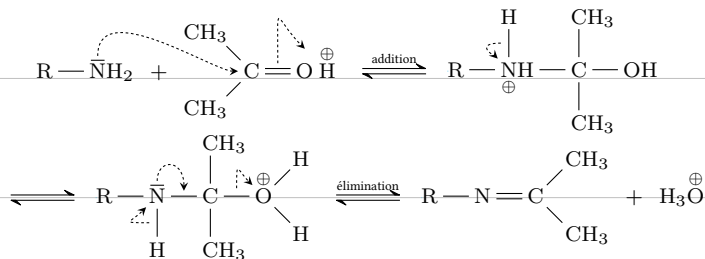
```
\setatomsep{2em}
\schemestart[-90]
  \chemfig{**6(---(-NH _2)---)}\arrow{0}\chemfig{HNO_2}
  \merge(c1)(c2)--()
  \chemfig{**6(---(-N_2|{}^\oplus)---)}\arrow{0}\chemfig{**6(---(-NH _2)---)}
  \merge(c3)(c4)--()
  \chemfig{**6(---(-N=[::-30]N-[::-30]**6(---(-NH_2)---))---)}
\schemestop
```
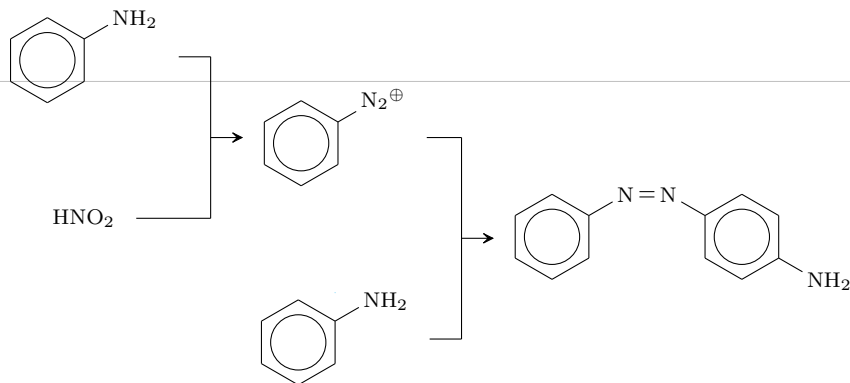
```
\setatomsep{2.5em}
\schemestart
  \chemfig{*6(=-(-)(=[2]0))}
  \arrow{->[\+\chemfig{H^\oplus}]]
  \chemleft[\subscheme[90]{%
    \chemfig{*6((-[2,0.33,,,draw=none]\scriptstyle\oplus)-=(-)-OH)}
    \arrow{<->}
    \chemfig{*6(=-(-)(-[6,0.33,,,draw=none]\scriptstyle\oplus)-OH)}}\chemright]
  \arrow(@c3--)\chemfig{*6((-[2]R)-=(-)-OH)}
  \arrow(@c4--)\chemfig{*6(=-(-)(-[6]R)-OH)}
```

`\schemestop`
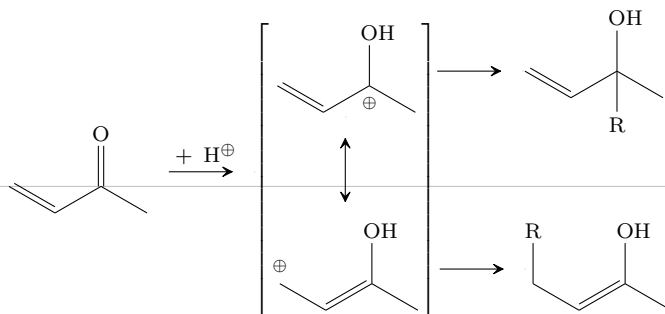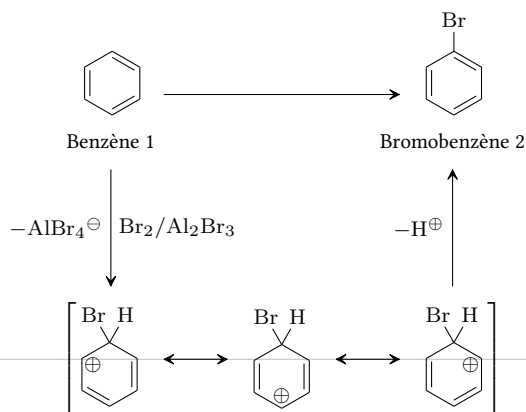
```
\setatomsep{1.5em}%
\definesubmol{+}{-[,-0.4,,,draw=none]\oplus}%
\schemestart
  \arrow{0}[,0]
  \chemleft[\subscheme{\chemfig{*6(=-=-(-[:120]Br)(-[:60]H)-(!+)-)}}
    \arrow{<->}
    \chemfig{*6(-(!+)-=-(-[:120]Br)(-[:60]H)-=)}
    \arrow{<->}
    \chemfig{*6(-=-(!+)-(-[:120]Br)(-[:60]H)-=)}}\chemright]
  \arrow(@c2--){<-[*0\chemfig{{-}AlBr_4|^\ominus}][*0\chemfig{Br_2/Al_2Br_3}]}[90,1.5]
  \chemname{\chemfig{*6(-=-=-=-)}}{Benzène 1}
  \arrow(@c4--){->[*0\chemfig{{-}H^\oplus}]}[90,1.5]
  \chemname{\chemfig{*6(-=-=(-Br)-=-)}}{Bromobenzène 2}
  \arrow(@c5.mid east--@c6.mid west)
\schemestop
```

```
\scriptsize\setbondoffset{1pt}\setatomsep{2em}\setcompoundsep{4em}
\schemestart
\chemfig{Cl-[4]@{a0}(=[@{a1}:120]@{a2}O)-[:-120](=[:-60]O)-[4]Cl}\+\chemfig{*6(-=-=(-@{oh1}OH)-=)}\arrow
\chemfig{*6((-O-[:150](-[@{oO}:150]@{o1}\lewis{6.,O})(-[@{cl0}:60]@{cl1}Cl)-[:240](-[4]Cl)=[6]O)=-=-=-)}
\arrow\chemfig{*6((-O-[:150](=[2]O)-[:-150](=[6]O)-[:150]Cl)=-=-=-)}\+\chemfig{HCl}
\arrow(@c1--){0}[-90,0.5]
\chemfig{*6(-=*6(-O-*6(-@{o2}(=[@{o3}]@{o4}O)-Cl)=)-=-)}\+\chemfig{*6(-=-=(-@{oh2}OH)-=)}\arrow
\chemfig{*6(-=*6(-O-(-(-[@{cl2}:60]@{cl3}Cl)(-[@{o5}:-120]@{o6}\lewis{6.,O})-O-[::-40]*6(=-=-=-))=)-=-)}
\kern-3em \arrow\chemfig{[:30]*6(=-(-O-[:-60](=O)-[:-120](=[4]O)-[:-60]O-*6(=-=-=-))=-=-)}
\kern-3em \+\chemfig{HCl}
\schemestop
\chemmove[line width=0.2pt,-stealth,dash pattern = on 2pt off 1pt]{
    \draw[shorten <=2pt](a1)..controls+(200:5mm)and+(200:5mm)..(a2);
    \draw[shorten >=2pt](oh1.west)..controls+(180:15mm)and+(60:5mm)..(a0);
    \draw[shorten <=6pt,shorten >=2pt](o1)..controls+(270:5mm)and+(270:5mm)..(oO);
    \draw[shorten <=2pt](cl0)..controls+(150:5mm)and+(150:5mm)..(cl1.150);
    \draw[shorten <=2pt](o3)..controls +(30:3mm) and +(30:5mm)..(o4.east);
    \draw[shorten >=2pt](oh2.135)..  controls +(150:10mm) and +(90:10mm)..  (o2);
    \draw[shorten >=2pt,shorten <=5pt]([xshift=-1.5mm]o6.315)..controls +(315:5mm) and +(315:5mm)..(o5);
```

```
\draw[shorten <=2pt](cl2)..controls +(135:5mm) and +(135:5mm)..(cl3.north west);}
```
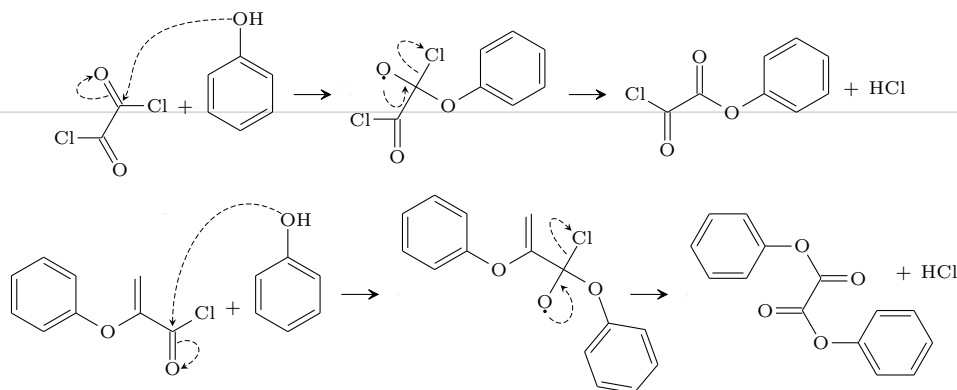

Cannizzaro reaction

```
\chemfig{[:-30]*6(=-=(-@{atoc}C([6]=[@{db}]@{atoo1}O)-H)-=-)}
\chemrel[\chemfig{@{atoo2}\chemabove{O}{\scriptstyle\ominus}}H]{-stealth}
\chemmove[-stealth,shorten >=2pt,dash pattern=on 1pt off 1pt,thin]{
        \draw[shorten <=8pt](atoo2) ..controls +(up:10mm) and +(up:10mm)..(atoc);
        \draw[shorten <=2pt](db) ..controls +(left:5mm) and +(west:5mm)..(atoo1);}
\chemfig{[:-30]*6(=-=(-C([6]-[@{sb1}]@{atoo1}\chembelow{O}{\scriptstyle\ominus})([2]-OH)-@{sb2}]H)-=-)}
\hspace{1cm}
\chemfig{[:-30]*6((-@{atoc}C([6]=[@{db}]@{atoo2}O)-[2]H)-=-=-)}
\chemmove[-stealth,shorten <=2pt,shorten >=2pt,dash pattern=on 1pt off 1pt,thin]{
        \draw([yshift=-4pt]atoo1.270) ..controls +(0:5mm) and +(right:10mm)..(sb1);
        \draw(sb2) ..controls +(up:10mm) and +(north west:10mm)..(atoc);
        \draw(db) ..controls +(right:5mm) and +(east:5mm)..(atoo2);}
\vspace{1cm}
\par
\chemrel{-stealth}
\chemfig{[:-30]*6(=-=(-C([1]-@{atoo2}O-[@{sb}]O]@{atoh}H)([6]=O))-=-)} \hspace{1cm}
\chemfig{[:-30]*6((-C(-[-5]H)(-[-7]H)-[2]@{atoo1}\chemabove{O}{\scriptstyle\ominus})-=-=-=)}
\chemmove[-stealth,shorten >=2pt,dash pattern=on 1pt off 1pt,thin]{
        \draw[shorten <=7pt](atoo1.90) ..controls +(+90:8mm) and +(up:10mm)..(atoh);
        \draw[shorten <=2pt](sb) ..controls +(up:5mm) and +(up:5mm)..(atoo2);}
```
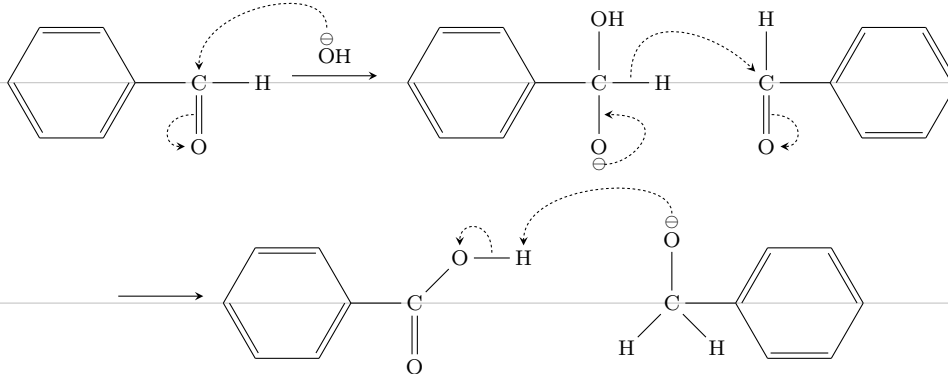

Beckmann rearrangement

```
\setbondoffset{1pt}
\setatomsep{2.5em}
\setcompoundsep{5em}
\setarrowoffset{6pt}
\schemestart
  \chemfig{(-[:-150]R')(-[:-30]R)=[2]N-[:30]OH}
  \arrow{<=>[\chemfig{H^\oplus}]}
  \chemfig{(-[@{a0}:-150]R')(-[:-30]R)=[2]@{a1}N-[@{b0}:30]@{b1}\chemabove{O}{\scriptstyle\oplus}H_2}
  \chemmove[red,-stealth,red,shorten <=2pt]{
     \draw(a0)..controls +(135:2mm) and +(215:4mm).. (a1);
     \draw(b0)..controls +(120:2mm) and +(180:3mm).. ([yshift=7pt]b1.180);}
  \arrow{<=>[\chemfig{{-}H_2O]}][,1.1]
  \chemleft[\subscheme[90]{%
    \chemfig{R'-\chemabove{N}{\scriptstyle\oplus}~C-R}
```

```
  \arrow{<->}[,0.75]
    \chemfig{R'-\lewis{2:,N}=@{a1}\chemabove{C}{\scriptstyle\oplus}-R}}\chemright]
  \arrow{<=>}[\chemfig{H_2@{a0}\lewis{0:2:,O}}]}[,1.1]
\chemmove[red,-stealth,red,shorten <=3pt]{
    \draw(a0)..controls+(90:10mm)and+(45:10mm)..([yshift=6pt]a1.45);}
\arrow(@c1--){0}[-90,0.333]
\chemfig{*6(R\rlap{$'$}-N=(-R)-\chemabove{O}{\scriptstyle\oplus} H_2)}
\arrow{<=>}[\chemfig{{-}H^\oplus}]}
\chemfig{*6(R\rlap{$'$}-N=(-R)-OH)}
\arrow
\chemfig{*6(R\rlap{$'$}-\chembelow{N}{H}-(-R)(=[2]O))}
\schemestop
```
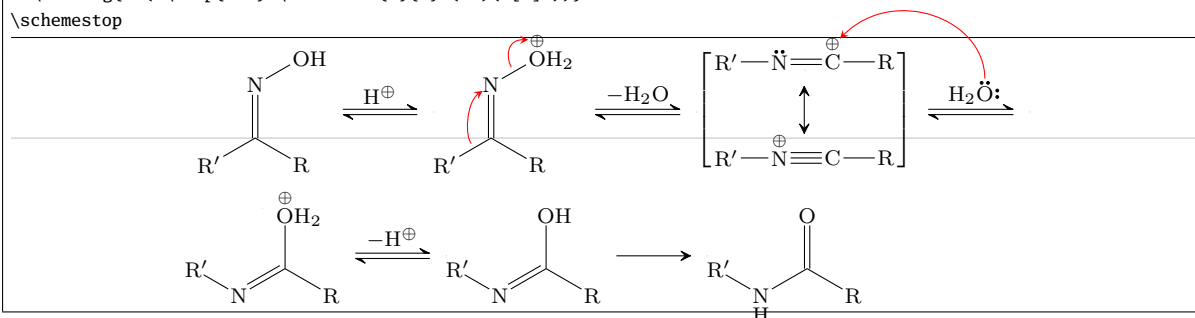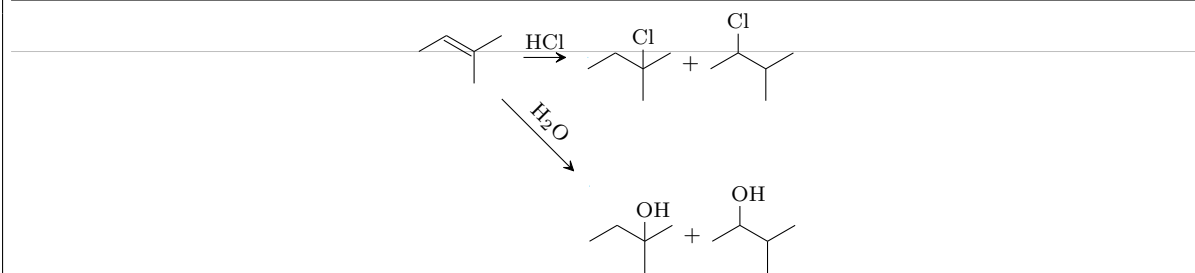
```
\setatomsep{1.5em}
\setcompoundsep{4em}
\schemestart
  \chemfig{-[::30]=_[::-60](-[::  -60])-[::60]}
  \arrow{->}[\chemfig{HCl}]}
\chemfig{-[::30]-[::-60](-[::120]Cl)(-[::-60])-[::60]}\+\chemfig{-[::30](-[::60]Cl)-[::-60](-[::-60])-[::60]}
  \arrow(@c1--.north west){->}[\chemfig{H_20}]}[-45,1.7]
\chemfig{-[::30]-[::-60](-[::120]OH)(-[::-60])-[::60]}\+\chemfig{-[::30](-[::60]OH)-[::-60](-[::-60])-[::60]}
\schemestop
```

```
\tikzset{obrace/.style={left delimiter={[},inner sep=3pt},
         cbrace/.style={right delimiter={]},inner sep=3pt},
         braces/.style={left delimiter={[},right delimiter={]},inner sep=3pt}}
\setatomsep{2em}
\schemestart
  \chemfig{H-C(=[:60]O)-[:-60]O-H}
  \arrow(--M1[obrace]){-U>[\scriptsize\chemfig{H_2SO_4^{}}]][\scriptsize\chemfig{HSO_4^\ominus}][][.25]}%
    [,1.5,shorten >=6pt]
  \chemfig{H-@{a2}C(-[:60]O-H)(-[:30,.5,,,draw=none]{\scriptstyle\oplus})-[:-60]O-H}
  \arrow(--[cbrace]){<->}
  \chemfig{H-C(=[:60]\chemabove{O}{\scriptstyle\oplus}-H)-[:-60]O-H}
  \arrow(@M1--){<=>[*{0}\scriptsize\chemfig{H-[:120]@{a1}O-[:60]CH_3}][*{0}\tiny addition]}[-90,1.33]
  \chemfig{H-C(-[2]O-[:30]H)(-\chemabove{O}{\scriptstyle\oplus}(-[:60]CH_3)-[:-60]H)-6]O-[:-30]H}
  \arrow{<=>[\tiny protolysis]}[180]
  \chemfig{H-C(-[2]O-[:30]H)(-O-CH_3)-[@{b1}6]@{a3}\chemabove{O}{\kern-4mm\scriptstyle\oplus}(-[:-150]H)-[:-30]H}
  \arrow(--[obrace]){<=>[*{0}\scriptsize\chemfig{{-}H_20}][*{0}\tiny elimination]}[-90,,shorten >=6pt]
  \chemfig{H-C(-[:60]O-H)(-[,.5,,,draw=none]{\scriptstyle\oplus})-[:-60]O-CH_3}
  \arrow(--[cbrace]){<->}
  \chemfig{H-C(=[:60]\chemabove{O}{\scriptstyle\oplus}-H)-[:-60]O-CH_3}
  \arrow{-U>[\scriptsize\chemfig{HSO_4^\ominus}][\scriptsize\chemfig{H_2SO_4^{}}][][.25]}[,1.5]
  \chemfig{H-C(=[:60]O)-[:-60]O-CH_3}
  \arrow(@M1--[yshift=-5pt]){0}[180,.5]{\tiny protolysis}
  \chemmove[->,red,shorten <=3pt,shorten >=1pt]{
```

```
    \draw(a1)..controls +(0:1.5cm)and+(0:3cm)..  (a2);
    \draw(b1)..controls +(0:5mm)and+(20:5mm)..(a3);}
\schemestop
```
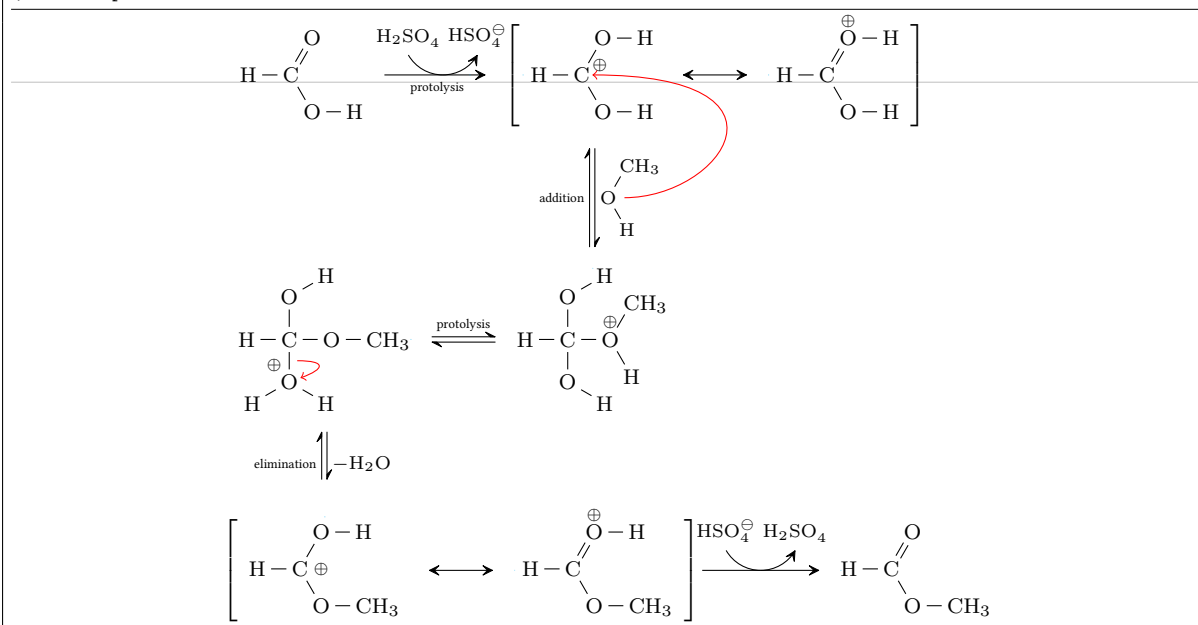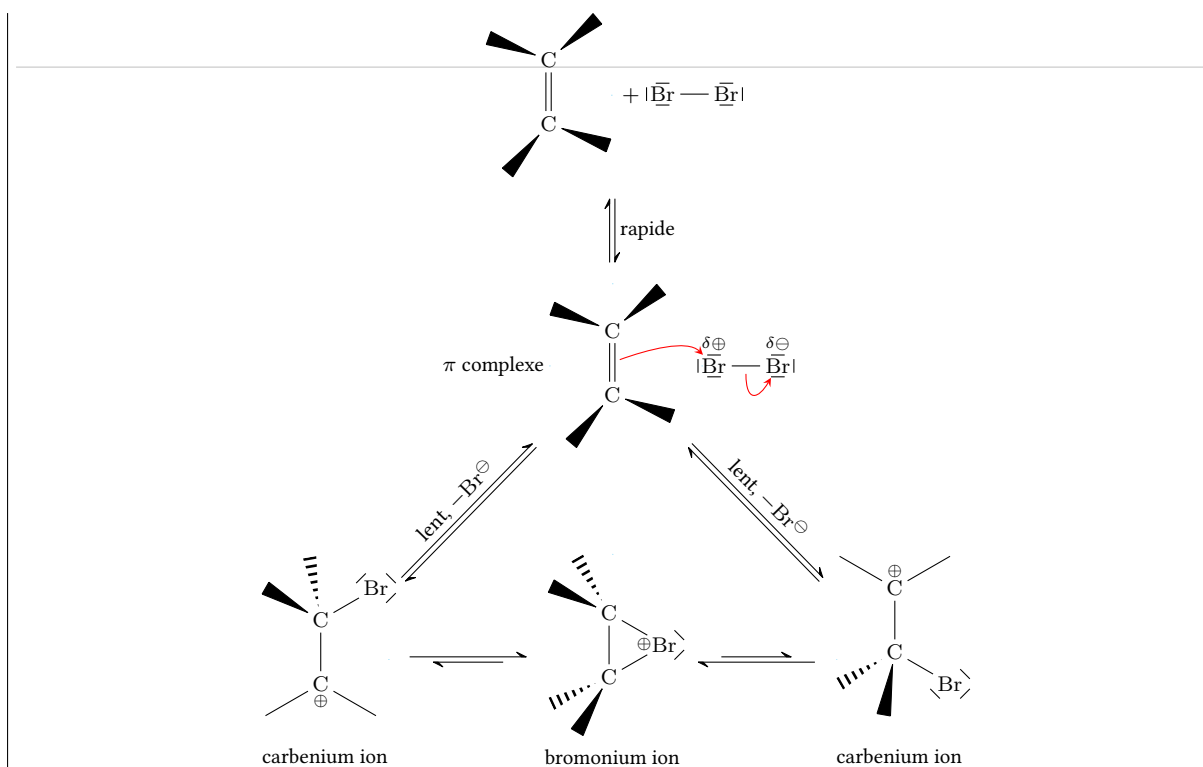
```
\schemestart
  \subscheme{%
    \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
    \arrow{0}[,0]\+\chemfig{\lewis{246,Br}-\lewis{026,Br}}}
  \arrow(@c1--olefin){<=>[*{0}rapide]}[-90]
  \chemfig{>[:-20]C(<[:40])=[@{db}6]C(<[:-130])<[:-20]}
  \arrow(--bromonium){0}[-90]
  \chemname{\chemfig{C*3((<)(<:[:-155])-\raise1.5pt\llap{$\scriptstyle\oplus$}\lewis{17,Br}-C(<:)(<[:155])-)}}
    {bromonium ion}
  \arrow(--carbeniumA){<<->}[,1.5]
  \chemname[12pt]{\chemfig{-[:-30]\chemabove{C}{\scriptstyle\oplus}(-[:30])-[6]C(<:[:-150])(<[:-100])-[:-30]
    \lewis{157,Br}}}{carbenium ion}
  \arrow(@bromonium--carbeniumB){<<->}[180,1.5]
  \chemname[12pt]{\vflipnext\chemfig{-[:-30]\chembelow{C}{\scriptstyle\oplus}(-[:30])-[6]C(<[:-150])
    (<:[:-100])-[:-30]\lewis{137,Br}}}{carbenium ion}
  \arrow(@olefin--){0}[,.25]
  \chemfig{{@{Br1}\chemabove[3pt]{\lewis{246,Br}}{\scriptstyle\delta\oplus}-[@{b2}]@{Br2}
    \chemabove[3pt]{\lewis{026,Br}}{\scriptstyle\delta\ominus}}}
  \arrow(@olefin--[left]){0}[180,0]
  $\pi$ complexe
  \arrow(@carbeniumA--@olefin){<=>[lent, \chemfig{{-}Br^\ominus}]}
  \arrow(@carbeniumB--@olefin){<=>[lent, \chemfig{{-}Br^\ominus}]}
  \chemmove[-stealth,red,shorten <=3pt,shorten >=2pt]{
    \draw(db) ..  controls +(20:5mm) and +(135:5mm) ..  (Br1);
    \draw(b2) ..  controls +(-90:5mm) and +(-120:5mm) ..  (Br2);}
\schemestop
```

$+ |\overline{Br} \, — \, \overline{Br}|$

rapide

π complexe

$\delta\oplus$ $\delta\ominus$
$|\overline{Br} \, — \, \overline{Br}|$

lent, $-Br^{\ominus}$

lent, $-Br^{\ominus}$

carbenium ion

bromonium ion

carbenium ion
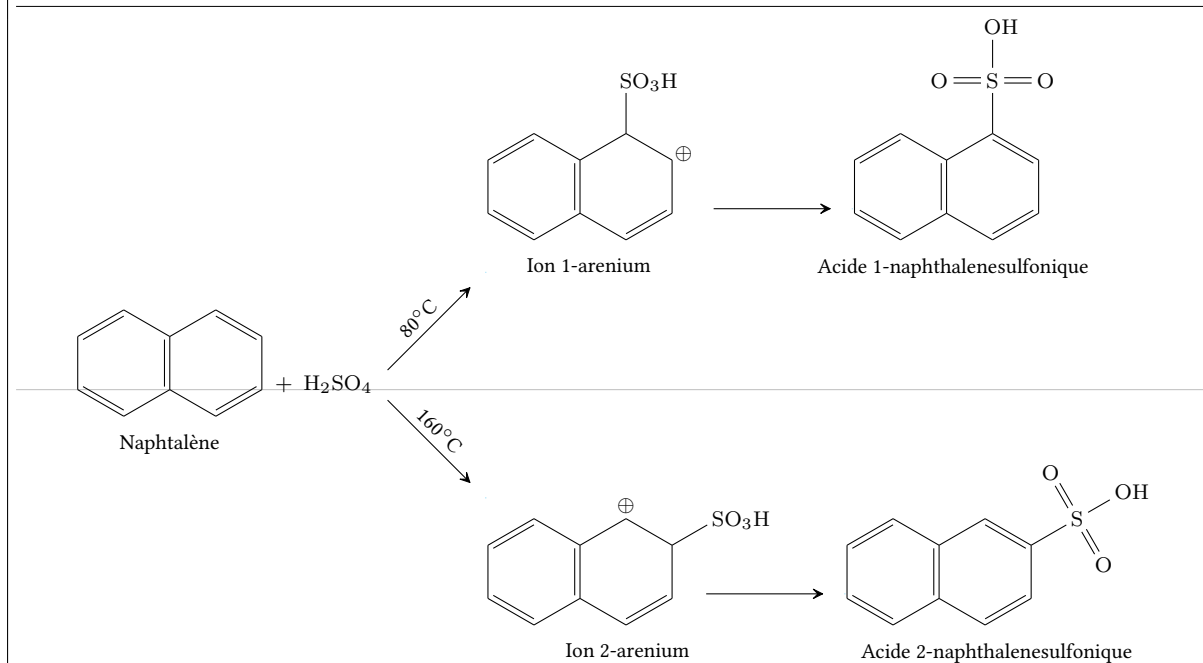
Sulfonation of naphthalene

```
\definesubmol\cycleoplus{-[,0.25,,,draw=none]\oplus}
\definesubmol{so2oh}{S(=[::90]O)(=[::-90]O)-OH}
\setatomsep{2.5em}
\schemestart[,1.5]
  \chemname{\chemfig{*6(=-*6(-=-=-)=-=-)}}{Naphtalène}\+\chemfig{H_2SO_4}
  \arrow(nph.mid east--.south west){->[80\degres C]}[45]
  \chemname{\chemfig{*6(=-*6(-=-(!\cycleoplus)-(-SO_3H)-)=-=-)}}{Ion 1-arenium}
  \arrow(.mid east--.mid west)
  \chemname{\chemfig{*6(=-*6(-=-(-!{so2oh})-)=-=-)}}{Acide 1-naphthalenesulfonique}
  \arrow(@nph.mid east--.north west){->[160\degres C]}[-45]
  \chemname{\chemfig{*6(=-*6(-=-(-SO_3H)-(!\cycleoplus)-)=-=-)}}{Ion 2-arenium}\kern-4em
  \arrow(.mid east--.mid west)
  \chemname{\chemfig{*6(=-*6(-=-(-!{so2oh})=-)=-=-)}}{Acide 2-naphthalenesulfonique}
\schemestop
```


SO$_3$H

Ion 1-arenium

OH
O $=$ S $=$ O

Acide 1-naphthalenesulfonique

Naphtalène $+$ H$_2$SO$_4$

80°C

160°C

SO$_3$H

Ion 2-arenium

O
S
O

OH

Acide 2-naphthalenesulfonique

```
\parbox{0.8\linewidth}{
\hspace{10em}
\tikz[remember picture]\node(n0){\chemname{}{Attacks\\nucleophiles}};\par
\vspace{2ex}\hspace{15em}
\chemfig{R^2-(-[:-60]@{a0}H)-[:60]@{a1}(-[:120]R^1)(-[1,0.25,,,draw=none]\scriptstyle\color{red}\delta+)
    =[@{a2}]@{a3}\lewis{1:7:,O}-[1,0.5,,,draw=none]\scriptstyle\color{red}\delta{-}}
\hspace{5em}
\chemname[-15ex]{}{\tikz[remember picture]\node(n1){};Addition reactions}\kern1em
\chemname{}{\tikz[remember picture]\node(n2){};Basic properties}\par
\vspace{6ex}\hspace{8em}
\chemname{}{Acidic properties of hydrogen\tikz[remember picture]\node(n3){};
    \\atom in $\alpha$ position}
\chemmove[-stealth,line width=0.8pt,green!60!black!70]{
    \draw[shorten >=2pt](n0)..controls+(270:4em)and+(180:2em)..(a1);
    \draw[shorten >=8pt](n1)..controls+(180:2em)and+(60:2em)..(a2);
    \draw[shorten >=5pt](n2)..controls+(180:2em)and+(270:2em)..([xshift=3pt]a3.315);
    \draw[shorten >=2pt](n3)..controls+(0:2em)and+(270:2em)..(a0);
}}
```
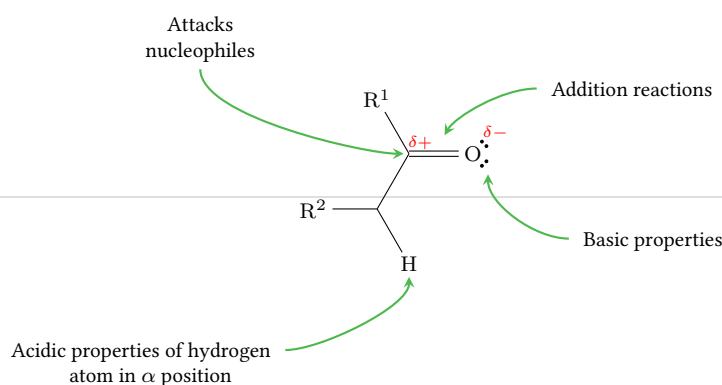
```
\newcommand\disk{\tikz\draw[fill=black,overlay](0,0)circle(2pt);}
\setatomsep{20pt}
\renewcommand\printatom[1]{#1}
\setbondoffset{2pt}
\definesubmol{hat}{-[:40,1.5]\disk-[::-30,2]\disk-[::-30,2]\disk-[::-120,2]\disk-[::-30,2]\disk}
\definesubmol{motif}{-[:40,1.5]\disk(-[2,3])-[::-30,2]\disk
(-[2,3])-[::-30,2]\disk(-[2,3])-[::-120,2]\disk(-[2,3])-[::-30,2]\disk}
\chemfig{\disk?[a](-[2,3]\disk?[b](-[2,3]\disk?[c]!{hat}?[c])!{motif}?[b](-[2,3]))!{motif}?[a](-[2,3])}
\qquad
\redefinesubmol{motif}{\disk(-[2,3])(-[:42,3.6,,,draw=none]\disk)-[:30,2.6]\disk
(-[2,3])-[0,3]\disk(-[2,3])-[:-150,2.6]\disk(-[4,3])-[2,3]-[4,3]}
\redefinesubmol{hat}{\disk-[:30,2.6]\disk-[0,3]\disk-[:-150,2.6]\disk-[4,3]}
\chemfig{!{motif}!{motif}!{hat}}
\qquad
\redefinesubmol{motif}{(-[2,3])(-[:25,2.75,,,white]-[2,1.5,,,white]\disk)-[:50,3]\disk
(-[2,3])-[::-50,3]\disk(-[2,3])-[::-130,3]\disk-[2,3]-[4,3]\disk}
\redefinesubmol{hat}{-[:50,3]\disk-[::-50,3]\disk-[::-130,3]\disk}
\chemfig{-[4,3]\disk!{motif}(-[:25,2.75,,,draw=none]\disk?[uat]?[dat](-[::0,2.75]?[uat1]?[dat1]
-[::-90,1.3]\disk?[uat2]?[dat2])-[::-90,1.3]-[:25,2.75])!{hat}?[uat3]?[dat3]-[:50,1.5]
(-[6,1.5,,,draw=none]\disk?[dat,,,blue]?[dat1,,,blue]?[dat2,,,blue]?[dat3,,,blue])
-[2,1.5,,,draw=none]\disk?[uat,,,red]?[uat1,,,red]?[uat2,,,red]?[uat3,,,red]}
```
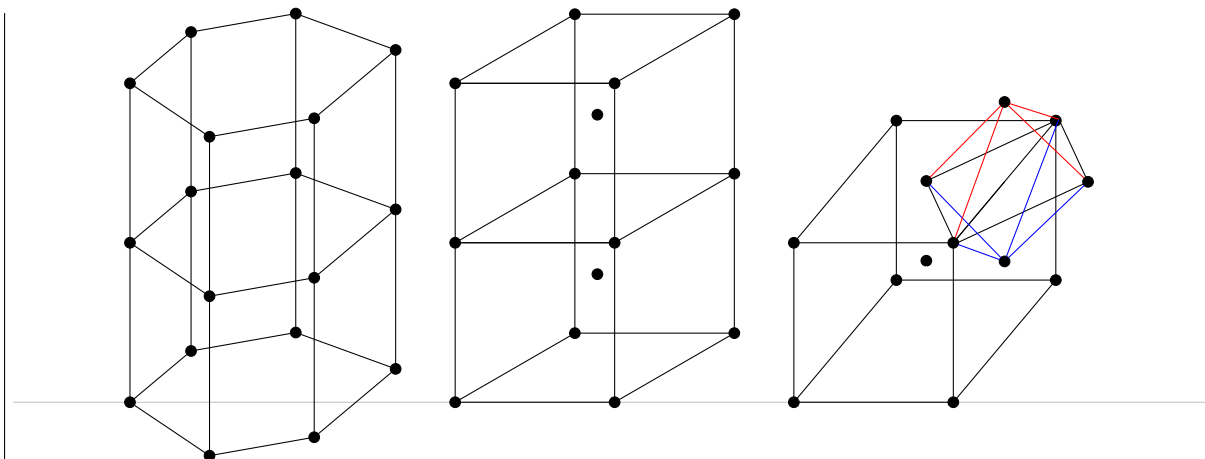
Taxotere

```
\chemfig{-[::-30](-[5])(-[7])-[::+60]-[::-60]O-[::+60](=[::-45]O)-[::+90]HN>[::-60](-[::+60]**6(------))
-[::-30](<:[2]OH)-[::-60](=[6]O)-[::+60]O>[::-60]*7(---?(<[::-120]OH)-(<|[1]CH_3)(<:[::-90]CH_3)
-(-[1](<[::+80]HO)-[0](=[::+60]O)-[7](<|[::+130]CH_3)(-[::+75](<|[2]OH)-[::-60]-[::-60](<[::+30]O-[::-90])
-[::-60](<[::+90])(<:[::+30]O-[7](-[6]CH_3)=[0]O)-[::-60])-[6]-[5,1.3]?(<:[7]O-[5](=[::-60]O)
-[6]**6(------)))=(-[2]CH_3)-)}
```

Index