chemfig

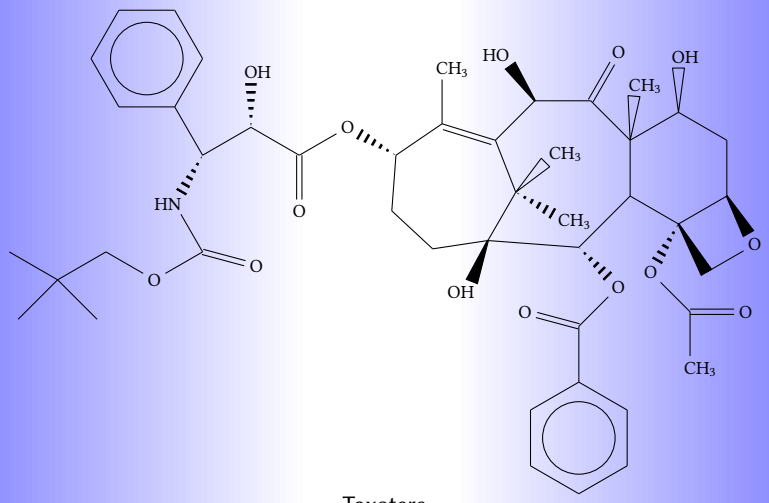
v1.71

30 october 2025

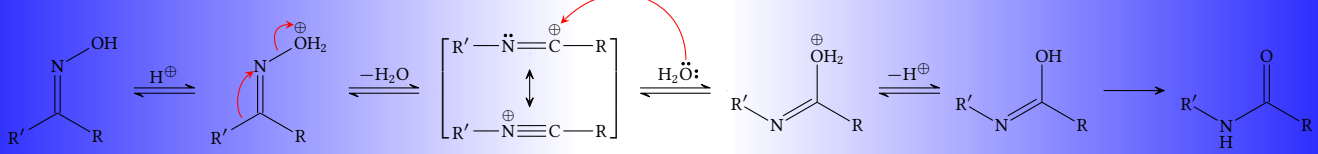
Christian Tellechea

unbonpetit@netc.fr

A TEX package for drawing molecules



Taxotere



The Beckmann rearrangement

Contents

Introduction

1 New in v1.7

The change log is no longer at the end of the chemfig.tex file but at the end of this manual, see page 86. It is not translated into English.

1.1 Horizontal reaction

To draw a horizontal reaction, chem*fig* provides a new environment:

```
\hreac[⟨keys⟩=⟨values⟩]
    ⟨reaction⟩
\endhreac
```

This environment certainly has far fewer possibilities than `\schemestart` ⟨*reaction*⟩ `\schemestop`, but it allows you to draw a horizontal reaction more quickly while retaining certain advanced features. See page 66.

1.2 Internal changes

Some internal changes have been made:

- By default, starting with version 1.7, the strut of the previous atom is no longer added to the argument of `\printatom`. This will slightly change the placement of atoms in some cases. However, you can revert to the previous behavior with `use atom strut` = ⟨*true*⟩. See page 31;
- When `use atom strut` = ⟨*false*⟩ is set, which is the default behaviour, each atom is typeset only once, whereas previously it was sometimes 5 times or more.
  An internal macro was corrected.

Important: for consistency, the next version of chem*fig* will require a change in the syntax of `\schemestart`. The two optional arguments will be removed for a single optional argument containing ⟨*keys*⟩=⟨*values*⟩. The settings that were possible with the two optional arguments will be accessible with new ⟨*keys*⟩.

2 Presenting chem*fig*

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

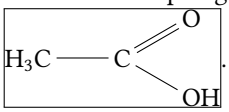
- `\input chemfig.tex` with ε-TEX;
- `\usepackage{chemfig}` with LaTeX;

In all cases, the ti*k*z package, if not loaded before, is loaded by chem*fig*.

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*k*z package, placed inside a `tikzpicture` environment. The choice of ti*k*z implies that:

- the user has a choice of compilation method: pdfLaTeX can be used equally well in `dvi mode` (tex ⟶ dvi ⟶ ps ⟶ pdf) or in `pdf mode` (tex ⟶ pdf). In effect ti*k*z, via the underlying `pgf`, gives identical graphical results in the two modes;
- the `bounding box` is automatically calculated by ti*k*z 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: 

.

3 Acknowledgment

This package has seen the light of day thanks to the assistance of Christophe Casseau, who had the idea. I thank him for his help before writing the code and for the tests he carried out.

I also want to warmly thank Theo Hopman for offering to translate this manual into English.

Operation of chem*fig*

This part is devoted to describing the most common features of chem*fig*. 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 chem*fig*. The "Advanced usage", page 25, will be more practical and will illustrate advanced features for the most demanding uses. It will also highlight methods of building real molecules, page 36. Finally, the last part will give examples of molecules and the code used to draw them.

1 The `\chemfig` macro

The macro `\chemfig` has the following syntax

`\chemfig[list of ⟨keys⟩=⟨values⟩]{⟨molecule code⟩}`

The optional argument in square brackets sets the parameters used for this molecule. It should be noted that the parameters are only modified for the current molecule and will be restored to their previous values after the macro has been executed. To permanently modify parameters, the macro `\setchemfig{⟨key⟩=⟨values⟩}` should be used.

Here is the complete list of parameters for the `\chemfig` macro, their default values, and a brief description.

⟨*keys*⟩	default ⟨*values*⟩	Description
`chemfig style`	⟨*empty*⟩	style given to ti*k*z
`use atom strut`	`false`	put in the the strut of previous atom in the argument of `\printaom`
`atom style`	⟨*empty*⟩	style of ti*k*z nodes containing atoms
`bond join`	`false`	boolean for bond joins
`fixed length`	`false`	boolean for fixed bond widths
`cram rectangle`	`false`	boolean to draw rectangle Cram bond
`cram width`	`1.5ex`	length of the base of the triangles Cram bonds
`cram dash width`	`1pt`	width of dash Cram bonds
`cram dash sep`	`2pt`	space between dash Cram bonds
`atom sep`	`3em`	space between atoms
`bond offset`	`2pt`	space between atom and bond

⟨*keys*⟩	default ⟨*values*⟩	Description
double bond sep	2pt	space between multiple bonds lines
angle increment	45	increment of the angle of bonds
node style	⟨*empty*⟩	style of atoms
bond style	⟨*empty*⟩	style of bonds
baseline	0pt	dimension or name of node to set the vertical position
debug	false	show nodes, anchors and names of nodes
cycle radius coeff	0.75	shrinkage ratio of the circle or arc inside cycles
stack sep	1.5pt	vertical gap between arguments of \chemaboveand \chembelowmacros
show cntcycle	false	show rings numbers
autoreset cntcycle	true	reset ring counter at \chemfigexecution
gchemname	true	if true, makes the depth assignments made by \chemnameinitand \chemnameglobal

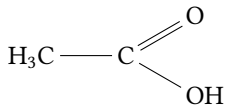
The ⟨*molecule code*⟩ contains instructions for drawing the molecule according to a syntax that will be explained in this document. There are no restrictions on the characters accepted in the code:

- all catcode 11 or 12 characters, i. e. upper and lower-case letters, numbers, mathematical operators (+ – * / =), punctuation marks whether active or not (. , ; : ! ? ’ ‘ ” |), parenthesis and brackets;
- more special characters such as ”~”, ”#”[1] as well as ”^” and ”_” which have their normal mathematical mode properties;
- spaces, but these are ignored by default because the atoms are composed in mathematical mode;
- the ”{” and ”}” braces that have their normal behavior as group markers or macro argument delimiters;
- macros.

2 Groups of atoms

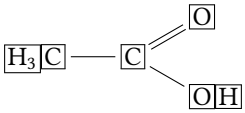
Drawing a molecule consists inherently of connecting groups of atoms with lines. Thus, in the molecule O═O, there are two groups of atoms, each consisting of a single atom “O”.

However, in this molecule



there are four groups of atoms: “H_3C”, “C”, “O” and “OH”. For reasons which we shall see later, chem*fig* splits each group into single atoms. Each atom extends up to the next capital letter or one of these special characters: - = ~ (! * < > @. chem*fig* ignores all characters inside braces when splitting groups into atoms.

Therefore the first group of atoms “H_3C” is split into two atoms: H3 and 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 chem*fig*’s definition. Thus chem*fig* sees the preceding molecule as follows:



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

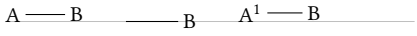
3 First atom’s role

It is important to understand that the placement of the entire molecule depends on the first atom placed, i.e. the first atom of the first group of atoms. For this first atom, its ti*k*z anchor ”base east” is placed on the baseline of the current line (drawn in gray in the examples of this manual).

[1] To avoid that # is doubled when the macro \chemfig is in the argument of a macro, instead of #, the macro \# or the macro \CFhash can be used.

Influence of the first atom

```
\chemfig{A-B}\qquad
\chemfig{-B}\qquad
\chemfig{A^1-B}
```



To set an arbitrary vertical offset or place a group of atoms on the baseline, use the ⟨*baseline*⟩ key (see page 30).

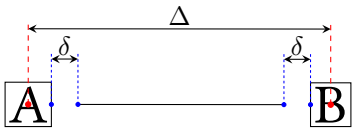
4 Different types of bonds

For chem*fig*, 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 ⁞▶ B	right Cram, dashed
7	`\chemfig{A<:B}`	A ◀⁞ B	left Cram, dashed
8	`\chemfig{A>|B}`	A ▷ B	right Cram, hollow
9	`\chemfig{A<|B}`	A ◁ B	left Cram, hollow

The ⟨*key*⟩ `double bond sep=`⟨*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 centers 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 ⟨*key*⟩ `atom sep` = ⟨*dim*⟩ adjusts the interatomic distance Δ. This setting, like all other settings, affects all the following molecules.

Interatomic distance

```
\chemfig[atom sep=2em]{A-B}\par
\chemfig[atom sep=50pt]{A-B}
```

The ⟨*key*⟩ `bond offset` = ⟨*dim*⟩ sets the spacing δ between the bond line and the atom. Its default value is 2pt.

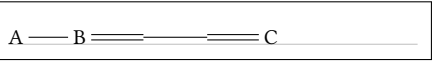
Trimming bonds

```
\chemfig[bond offset=0pt]{A-B}\par
\chemfig[bond offset=5pt]{A-B}
```

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

Empty groups

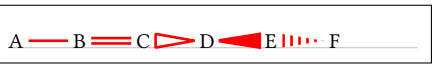
```
\chemfig{A-B=-=C}
```



The ⟨*key*⟩ `bond style` = ⟨*tikz code*⟩ sets the style for all the bonds drawn thereafter. The ⟨*tikz code*⟩ is empty by default. To custom a single bond, see page 11.

Style of bonds

```
\chemfig[bond style={line width=1pt,red}]{A-B=C>|D<E>:F}
```

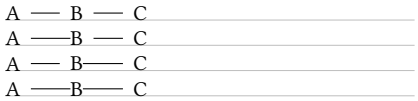


The spacing δ for just one bond can be specified with either the character #, the macro \# or \CFhash. It is important to notice that if the macro \chemfig is in the argument of a macro, # must *not* be used and in that case, \# or \CFhash must be prefered.

The token #, \# or \CFhash 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

```
\setchemfig{bond offset=4pt}
\chemfig{A-B-C}\par
\chemfig{A-#(,0pt)B-C}\par
\chemfig{A-B-#(0pt)C}\par
\chemfig{A-#(,0pt)B-#(0pt)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[2]:

Math mode

```
\chemfig{A_1B^2-C _ 3 ^ 4}
```

There are settings specifically for Cram bonds:

- `cram width` = ⟨*dim*⟩ is the size of the base of the triangle, and is 1.5pt by default;
- `cram dash width` = ⟨*dim*⟩ is the thickness of the dots, and is 1pt by default;
- `cram dash sep` = ⟨*dim*⟩ is the spacing between the dots, and is 2pt by default.

Here is an example where the three dimensions are changed:

Modified Cram bonds

```
\chemfig[cram width=10pt,
         cram dash width=0.4pt,
         cram dash sep=1pt]{A>B>:C>|D}
```

5 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.

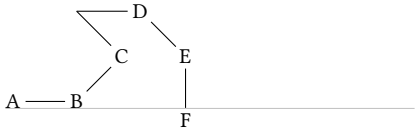
There are several ways of specifying the bond angle.

5.1 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.

Predefined angles

```
\chemfig{A-B-[1]C-[3]-D-[7]E-[6]F}
```

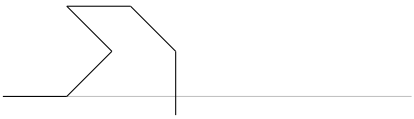


These angles remain valid if the atoms are empty, and this is the case for all the features we will see below:

[2] There is a problem with the placement of groups of atoms containing exponents or subscripts.

Predefined angles with empty groups

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



The ⟨*key*⟩ `angle increment` = ⟨*angle*⟩ sets the default angle used to calculate the angle of a bond:

Set the predefined angle

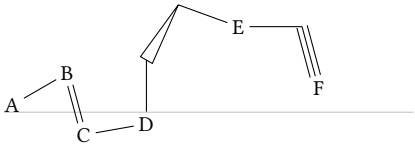
```
Default (45) : \chemfig{-[1]-[-1]-[1]-[-1]}

Angle of 30 : \chemfig[angle increment=30]{-[1]-[-1]-[1]-[-1]}
```

5.2 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}
```



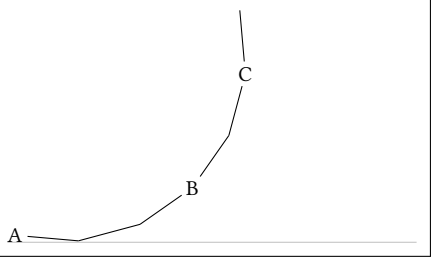
5.3 Relative angles

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

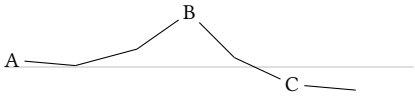
```
\chemfig{A-[:-5]-[::+20]-[::20]B-[::+20]-[::20]C-[::20]}
```



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]}
```



6 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 `atom sep` as we have seen on page 7. 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

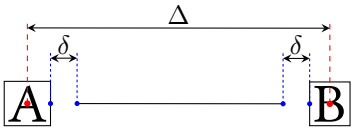
```
\chemfig{A^{++}_{2}-B^{-}_3}
```

$A_2^{++}-B_3^-$

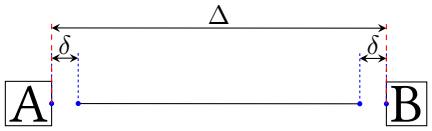
It is important to note that the exponent – is *put inside braces*. If this were not done, chem*fig* would stop the atom on this character, which is a bond character. The atom would then be "B^", which would lead to unexpected results.

It is possible to change the behavior of chem*fig* about the interatomic spacing. Indeed, when the `\chemfig` macro is immediately followed by a star, the ⟨*key*⟩ `atom sep`no longer defines the distance between the centers of atoms, denoted Δ, but *length of the bonds*. Consequently, the bonds have fixed lengths while the distance between the centers of the atoms is variable and depends on their size. Here is the diagram on page 7 and what becomes with the two boolean values of key `fixed length`:

`fixed length` = ⟨*false*⟩



`fixed length` = ⟨*true*⟩



In rings, even when `fixed length` = ⟨*true*⟩, the default behavior is restored for the bonds of the cycle, in order to draw regular polygons.

Fixed length bonds

```
\chemfig{Cl-Cl}\par
\chemfig[fixed length=true]{Cl-Cl}
```

Cl—Cl
Cl—Cl

Especially with the default behavior, to avoid too short bonds, it is sometimes necessary to increase (or perhaps reduce) the interatomic distance. 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

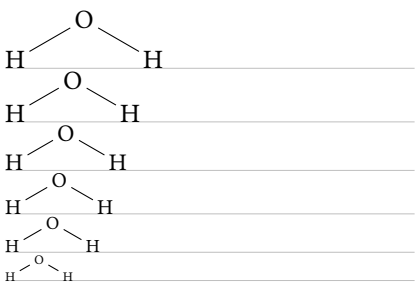
```
\chemfig{A^{++}_{2}-[,2]B^{-}_3}\par
\chemfig{A-B-[,2]C=[,0.5]D}\par
\chemfig{-=[,1.5]-[,0.75]=[:-20,2]}
```

A_2^{++}—B$_3^-$
A—B—C=D

We can change the `size of molecules` by altering the font size or the ⟨*key*⟩ `atom sep`, possibly on both, 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
\setchemfig{atom sep=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}
```



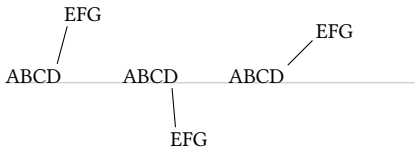
7 Departure and arrival atoms

A group of atoms can contain several atoms. Suppose we want to connect the group “ABCD” to the group “EFG” with a bond. chem*fig* 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

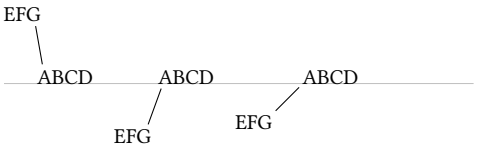
```
\chemfig{ABCD-[:75]EFG}\quad
  \chemfig{ABCD-[:-85]EFG}\quad
  \chemfig{ABCD-[1]EFG}
```



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}
```



In some cases, you may want a bond to start from atoms other than those calculated by chem*fig*. You can force a starting atom or an ending atom with the optional argument of the bond. You must write:

`[,,`⟨*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}
```

8 Customization of bonds

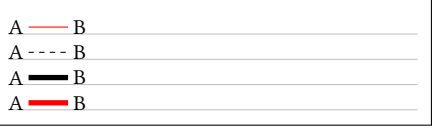
There is a fifth and last optional argument for bonds which is found after the fourth comma:

`[,,,,`⟨*tikz code*⟩`]`

This ⟨*tikz code*⟩ is passed directly to ti*k*z 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*k*z decoration library has been loaded. A bond can be made invisible by writing “`draw=none`”. To set several attributes, the syntax of ti*k*z is used, separating them by a comma:

Passing tikz code

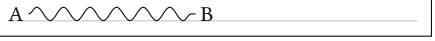
```
\chemfig{A-[,,,,red]B}\par
\chemfig{A-[,,,,dash pattern=on 2pt off 2pt]B}\par
\chemfig{A-[,,,,line width=2pt]B}\par
\chemfig{A-[,,,,red,line width=2pt]B}
```



Numerous ti*k*z 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}
```



Cram bonds ignore thickness and dash settings.

9 Connecting bonds

By default, the lines representing single bonds do not connect, which can be too visible and unsightly when the line widths are large.

For those who find this "ugly[3]", it is now possible connect the single bonds with a slightly increased compilation time. The boolean ⟨*key*⟩ `bond join` = ⟨*boolean*⟩ macro enables this feature when ⟨*true*⟩ and disables it when ⟨*false*⟩, which is the better behavior, set by default.

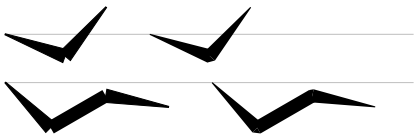
Connecting bonds

```
\setchemfig{bond style={line width=3pt}}
\chemfig{-[1]-[7]} et
\chemfig[bond join=true]{-[1]-[7]}
```

The problem is even more obvious with Cram bonds when they connect to a single bond or another Cram bond. Here again, setting `bond join` to ⟨*true*⟩ makes for more aesthetically pleasing connections.

Connecting Cram bonds

```
\chemfig{<[:-20]>[:50]}\qquad
\chemfig[bond join]{<[:-20]>[:50]}
\medbreak
\setchemfig{cram width=5pt}
\chemfig{<[:-45]-[:30,,,,line width=5pt]>[:-10]}\qquad
\chemfig[bond join]{<[:-45]-[:30,,,,line width=5pt]>[:-10]}
```



It is important to note the tow following points:

- `bond join` is ⟨*true*⟩, the Cram bond is drawn without an outline: it will therefore appear slightly thinner (the importance depending on the ⟨*line width*⟩ parameter);
- `bond join` has no effect between a simple bond and a Cram bond when the 2 vertices of the base of the triangle of the Cram bond are outside the space delimited by the 2 parallel edges of the simple bond. Mathematically, it is the case when $d \cos \alpha > w$, where d is the value of ⟨*cram width*⟩, w is the width of the single bond and α is the angle between the 2 bonds.

Connecting Cram bonds

```
\setchemfig{cram width=5pt, bond join}
\chemfig{-[,,,,line width=3pt]>[:30]}\qquad
\chemfig{-[,,,,line width=3pt]>[:65]}
```

10 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 chem*fig* calculate these based on the bond angle;
- ⟨*empty*⟩ for the parameters passed to ti*kz*.

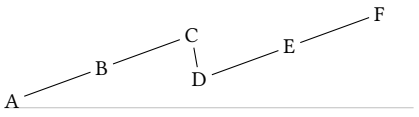
These default values can be changed for the whole molecule by beginning the molecule code with

$$[\langle angle\rangle,\langle coeff\rangle,\langle n1\rangle,\langle n2\rangle,\langle code\ tikz\rangle]$$

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}
```

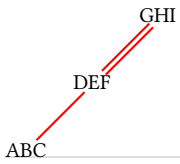


[3] See `http://tex.stackexchange.com/questions/161796/ugly-bond-joints-in-chemfig detokenize`

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}
```



11 Branches

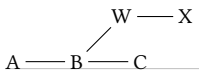
11.1 Principle

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 sub-molecule which will be attached to the atom.

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

A branch

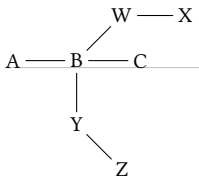
```
\chemfig{A-B(-[1]W-X)-C}
```



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

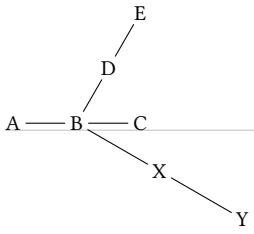
```
\chemfig{A-B(-[1]W-X)(-[6]Y-[7]Z)-C}
```



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 "`[::-30,1.5]-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

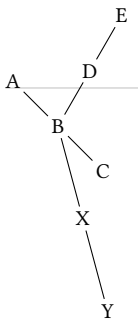
```
\chemfig{A-B([:60]-D-E)([::-30,1.5]-X-Y)-C}
```



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}
```



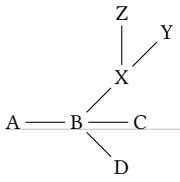
We see that the angle between the bond `B–C` and the bond `B–X` stays at 30° because it is a relative angle for the sub-molecule "`–X–Y`". By contrast, the branch "`–D–E`" stays inclined at 60° to the horizontal, and does not follow the rotation given by the $-45°$ angle at the beginning; this is expected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

11.2 Nesting

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

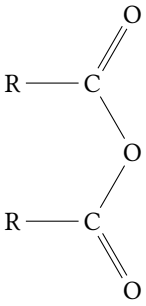
Nested branches

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



11.3 Method

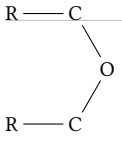
Suppose now that we want to draw an acid anhydride molecule:



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

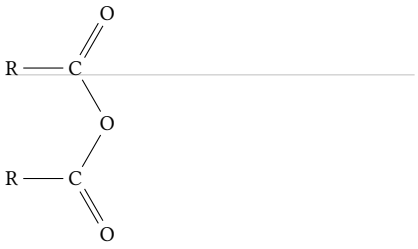
```
\chemfig{R-C-[::-60]O-[::-60]C-[::-60]R}
```



To this structure we just have to add two "`=O`" sub-molecules to each of the carbon atoms:

Acid anhydride

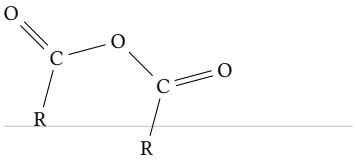
```
\chemfig{R-C(=[::+60]O)-[::-60]O-[::-60]C(=[::+60]O)-[::-60]R}
```



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}
```



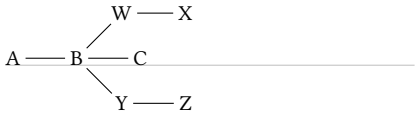
12 Connecting distant atoms

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:

Branched structure

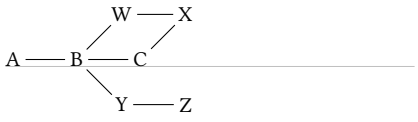
```
\chemfig{A-B(-[1]W-X)(-[7]Y-Z)-C}
```



and suppose that we want to connect the atoms `X` and `C`. In this case, chem*fig* 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`:

Distant bond

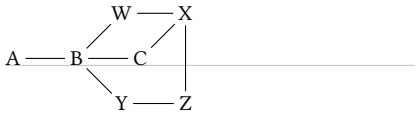
```
\chemfig{A-B(-[1]W-X?)(-[7]Y-Z)-C?}
```



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

Several distant bonds

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



Now imagine if we were to leave the distant bonds `X–C` and `X–Z`while 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:

$$\texttt{?[}\langle \textit{name} \rangle\texttt{,}\langle \textit{bond} \rangle\texttt{,}\langle \textit{tikz} \rangle\texttt{]}$$

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

- The $\langle \textit{name} \rangle$ is the name of the hook: all alphanumeric characters (a…z, A…Z, 0…9) are allowed[4]. The name is a by default. In the first occurrence of the hook with this name, only this field is used.

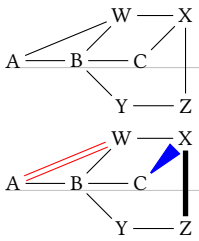
[4] This is not exactly right. Actually all the characters that can be put between `\csname...\endcsname` are allowed.

- ⟨*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 7 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*k*z 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:

Multiple distant bonds

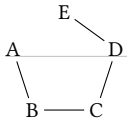
```
\chemfig{A?[a]-B(-[1]W?[a]-X?[b])(-[7]Y-Z?[b])-C?[b]}\par\medskip
  \chemfig{A?[a]-B(-[1]W?[a,2,red]-X?[b])(-[7]Y-
  Z?[b,1,{line width=2pt}])-C?[b,{>},blue]}
```



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`:

An incomplete ring

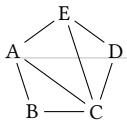
```
\chemfig{A-[:-72]B-C-[:72]D-[:144]E}
```



Then we must do this:

Multiple distant bonds

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



13 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.

13.1 Syntax

chem*fig* can easily draw regular polygons. The idea is to attach a ring to an ⟨*atom*⟩ outside the ring with this syntax:

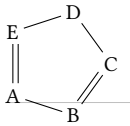
⟨*atom*⟩`*`⟨*n*⟩`(`⟨*code*⟩`)`

⟨*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

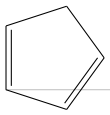
```
\chemfig{A*5(-B=C-D-E=)}
```



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

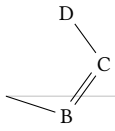
```
\chemfig{*5(-=--=)}
```



A ring can be incomplete:

Incomplete 5-ring

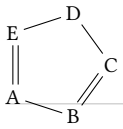
```
\chemfig{*5(-B=C-D)}
```



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:

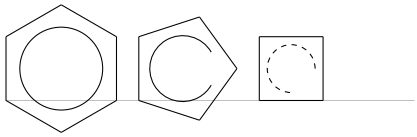
$$\langle atom\rangle\texttt{**[}\langle angle\ 1\rangle\texttt{,}\langle angle\ 2\rangle\texttt{,}\langle tikz\rangle\texttt{]}\langle n\rangle\texttt{(}\langle code\rangle\texttt{)}$$

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

- $\langle angle\ 1\rangle$ and $\langle angle\ 2\rangle$ 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;
- $\langle tikz\rangle$ 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(----)}
```



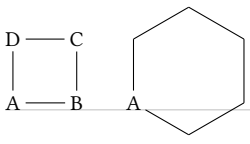
13.2 Angular position

13.2.1 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

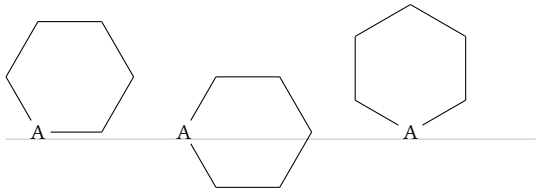
```
\chemfig{A*4(-B-C-D-)}\qquad\chemfig{A*6(------)}
```



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(------)}
```



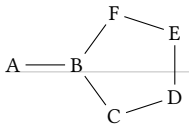
13.2.2 After a bond

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

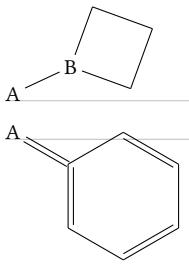
```
\chemfig{A-B*5(-C-D-E-F-)}
```



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(=-=-=-)}
```



13.3 Branches on a ring

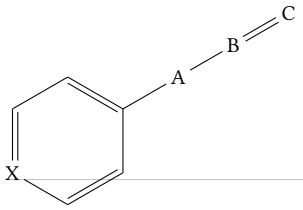
To have branches attached to the vertices of a ring, we use the syntax we have already seen:

$$\langle atom \rangle (\langle code \rangle)$$

where the $\langle code \rangle$ is that of the sub-molecule and the $\langle atom \rangle$ 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:

Branch on a ring

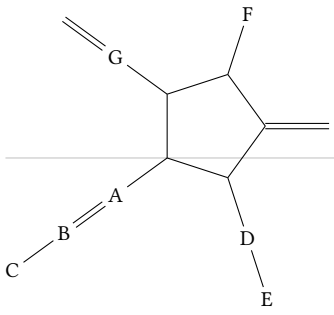
```
\chemfig{X*6(-=-(-A-B=C)=-=-)}
```



A sub-molecule can be attached to the first vertex of a ring, just like the other vertices:

Ring and branches

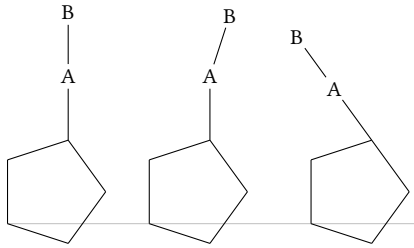
```
\chemfig{*5((-A=B-C)-(-D-E)-(=)-(-F)-(-G=)-)}
```



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:

Branches at specified angles

```
\chemfig{*5(---([:90]-A-B)--)}\qquad
  \chemfig{*5(---(-[:90]A-B)--)}\qquad
  \chemfig{*5(---([::+0]-A-B)--)}
```

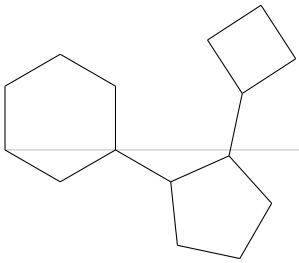


It is worth noting that in the third example, where a relative angle of 0° was given, the bonds of the branch are drawn in line with the preceding bond in the ring. This is the rule on page 9 which specified that the reference angle was that of the bond last drawn.

We can now connect together rings with bonds:

Connected rings

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

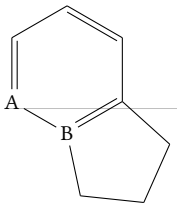


13.4 Nested rings

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:

Nested rings

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

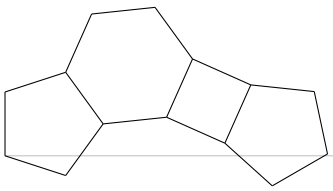


Note that the ring which is going to be attached to the main ring has an angular position such that two of the rings' sides coincide. In addition, the 5-ring has only four bonds "----". In effect, the fifth will be useless because it is the second side of the 6-ring, which has already been drawn.

It is quite possible to glue multiple rings together:

Multiple nested rings

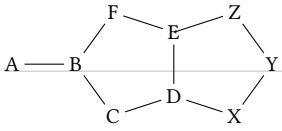
```
\chemfig{*5(--*6(-*4(-*5(----)--)----)---)}
```



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 center of atom "E".

Flawed drawing

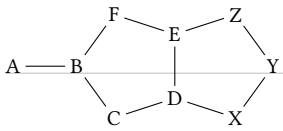
```
\chemfig{A-B*5(-C-D*5(-X-Y-Z-)-E-F-)}
```



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

Distant bond and ring

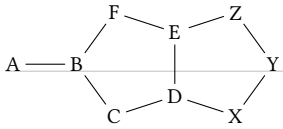
```
\chemfig{A-B*5(-C-D*5(-X-Y-Z?)-E?-F-)}
```



We could also use a `\phantom{E}` at the last vertex of the 5-ring:

Using \phantom

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

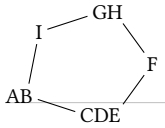


13.5 Rings and groups of atoms

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

Ring and groups of atoms

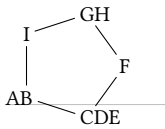
```
\chemfig{AB*5(-CDE-F-GH-I-)}
```



In order for the ring to have a regular shape, it is necessary to override the chem*fig* 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-)}
```



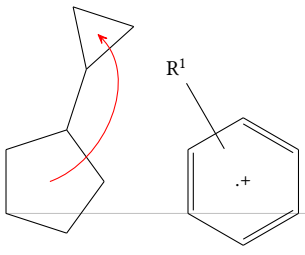
13.6 Center of rings

Each ring has at its center a node of zero dimension whose name is `centrecycle`⟨*n*⟩ where ⟨*n*⟩ is the number of the ring, (the rings are numbered in the order in which they are drawn). It is possible to display the number of each ring by setting the boolean `show cntcycle` to true.

The default true boolean `autoreset cntcycle` resets the ring counter at the beginning of each molecule to 0 (i.e., each time `\chemfig` is executed).

Centre of rings

```
\chemfig{*5(---(-*3(---))--)}
\chemmove{\draw[red](cyclecenter1)to[out=20,in=-45](cyclecenter2);}
\qquad
\chemfig{*6(-=-=-=)}
\chemmove{%
  \node[at=(cyclecenter1)](){.+}
  node [at=(cyclecenter1),shift=(120:1.75cm)](end){\printatom{R^1}};
  \draw[-,shorten <=.5cm](cyclecenter1)--(end);
}
```



14 Representing electron movements

Starting with chem*fig* 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*k*z 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*k*z instructions. These instructions are placed in the argument of the command `\chemmove`[5] 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*k*z package.

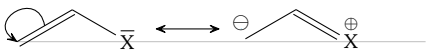
14.1 Mesomeric effects

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.

The macros `\schemestart`, `\schemestop`, `\arrow` and `\+` are explained in the chapter IV, starting on page 49.

Mesomeric effect 1

```
\schemestart
  \chemfig{@{a1}=_[@{db}::30]-[::-60]\charge{90=\|}{X}}
  \arrow{<->}
  \chemfig{\chemabove{\vphantom{X}}{\ominus}-[::30]=_[::-60]
  \chemabove{X}{\scriptstyle\oplus}}
\schemestop
  \chemmove{\draw(db).. controls +(100:5mm) and +(145:5mm).. (a1);}
```



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:

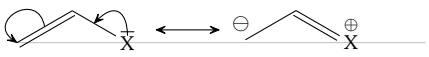
`\chemmove{\draw(db)..controls +(80:8mm) and +(145:8mm)..(a1);}`

For arrows in `\chemmove`, the default tip is "CF".In this example we ask for an arrow (`[->]`) and we use two `control points`[6]. 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").

[5] Actually, the `\chemmove` command puts its argument in a "`tikzpicture`" environment with the options "`remember picture, overlay`".
[6] To find all the ways of connecting two nodes with ti*k*z, read the documentation for that package.

Mesomeric effect 2

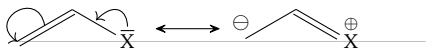
```
\schemestart
  \chemfig{@{a1}=_[@{db}::30]-[@{sb}::-60]@{dnl}\charge{90=\|}{X}}
  \arrow{<->}
  \chemfig{\chemabove{\vphantom{X}}{\ominus}-[::30]=_[::-60]
  \chemabove{X}{\scriptstyle\oplus}}
\schemestop
\chemmove{
    \draw(db)..controls +(100:5mm) and +(145:5mm)..(a1);
    \draw(dnl)..controls +(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

```
\schemestart
  \chemfig{@{a1}=_[@{db}::30]-[@{sb}::-60]@{dnl}\charge{90=\|}{X}}
  \arrow{<->}
  \chemfig{\chemabove{\vphantom{X}}{\ominus}-[::30]=_[::-60]
  \chemabove{X}{\scriptstyle\oplus}}
\schemestop
\chemmove[->]{% change the tip style
    \draw(db).. controls +(100:5mm) and +(145:5mm).. (a1);
    \draw[shorten <=3pt,shorten >=1pt](dnl) .. controls +(90:4mm)
          and +(45:4mm) .. (sb);}
```

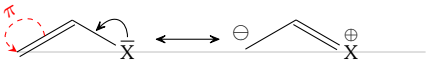


The option “`shorten <=3pt`” indicates that the tail of the arrow is to be shortened by 3 pt just as “`shorten >=2pt`” means that the head of the arrow is shortened by 2 pt.

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 and set it to “`-stealth`”, 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

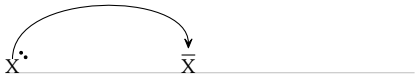
```
\schemestart
  \chemfig{@{a1}=_[@{db}::30]-[@{sb}::-60]@{dnl}\charge{90=\|}{X}}
  \arrow{<->}
  \chemfig{\chemabove{\vphantom{X}}{\ominus}-[::30]=_[::-60]
  \chemabove{X}{\scriptstyle\oplus}}
\schemestop
\chemmove{
    \draw[-stealth,thin,dash pattern= on 2pt off 2pt,red]
        (db).. controls +(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 1

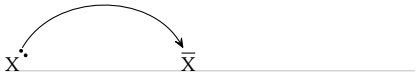
```
\chemfig{@{x1}\charge{45=\:}{X}}
\hspace{2cm}
\chemfig{@{x2}\charge{90=\|}{X}}
\chemmove{
    \draw[shorten >=4pt](x1).. controls +(90:1cm) and +(90:1cm).. (x2);}
```



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*k*z makes the arrow leave from the anchor “x1.90” which corresponds to the intersection of the ray leaving from the center 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

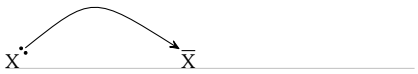
```
\chemfig{@{x1}\charge{45=\:}{X}}
\hspace{2cm}
\chemfig{@{x2}\charge{90=\|}{X}}
\chemmove[shorten <=4pt,shorten >=4pt]{
    \draw(x1.57).. controls +(60:1cm) and +(120:1cm).. (x2.90);}
```



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}\charge{45=\:}{X}}
\hspace{2cm}
\chemfig{@{x2}\charge{90=\|}{X}}
\chemmove[shorten <=4pt,shorten >=4pt]{
    \draw(x1.57).. controls +(1cm,.8cm).. (x2.90);}
```



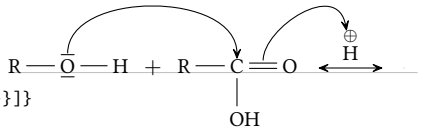
All the graphics drawn by means of the command \chemmove are superimposed and will not be included in the bounding boxes. We can see this in the preceding example.

14.2 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

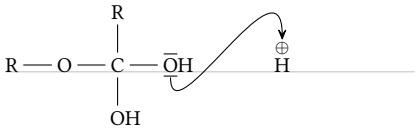
```
\setchemfig{atom sep=7mm}
\schemestart
  \chemfig{R-@{dnl}\charge{90=\|,-90=\|}{O}-H}
  \+
  \chemfig{R-@{atoc}C([6]-OH)=[@{db}]O}
  \arrow(.mid east--){<->[\chemfig{@{atoh}\chemabove{H}{\scriptstyle\oplus}}]}
\schemestop
\chemmove[shorten <=2pt]{
  \draw(dnl)..controls +(90:1cm)and+(north:1cm)..(atoc);
  \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 (⊕ or ⊖). 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

```
\setchemfig{atom sep=7mm}
\chemfig{R-O-C(-[2]R)(-[6]OH)-@{dnl}\charge{90=\|,-90=\|}{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....

15 Writing a name under a molecule

For convenience, chem*fig* can write the name of a molecule underneath it with the command

\chemname[⟨*dim*⟩]{\chemfig{⟨*code of the molecule*⟩}}{⟨*name*⟩}

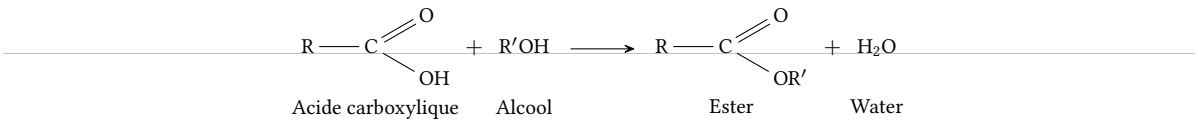
The ⟨*dim*⟩, which is 1.5ex by default, will be inserted between the `baseline` of the molecule and the top of the letters of the ⟨*name*⟩. The ⟨*name*⟩ will be centered relative to the molecule, but the ⟨*name*⟩ may not contain multiple paragraphs. As we see in this example: H—O—H, the ⟨*name*⟩ which is displayed under the molecule is taken into account only for the vertical size of the bounding box. The horizontal size of ⟨*name*⟩ is always zero.

The water molecule: H_2O

Here is a reaction with the names under the molecules:

Displaying names of molecules

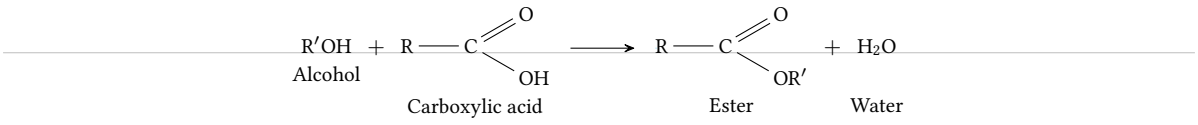
```
\schemestart
  \chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Acide carboxylique}
  \+
  \chemname{\chemfig{R'OH}}{Alcool}
  \arrow(.mid east--.mid west)
  \chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
  \+
  \chemname{\chemfig{H_2O}}{Water}
\schemestop
\chemnameinit{}
```



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

Name alignment 1

```
\schemestart
  \chemname{\chemfig{R'OH}}{Alcohol}
  \+
  \chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic acid}
  \arrow(.mid east--.mid west)
  \chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
  \+
  \chemname{\chemfig{H_2O}}{Water}
\schemestop
\chemnameinit{}
```



In fact, to draw the ⟨*name*⟩ the command \chemname inserts 1.5ex + *the largest of the depths*[7] *of the molecules thus far* below the baseline of each molecule (light gray for the examples in this manual). The command \chenameinit{⟨*stuff*⟩} initializes this largest depth with the ⟨*stuff*⟩. Therefore one should:

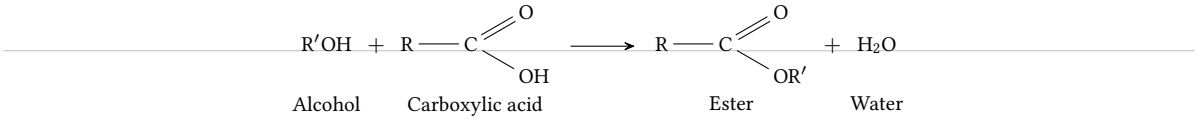
- write \chemnameinit{⟨*deepest molecule*⟩} before using the \chemname command in a reaction, unless the reaction begins with the deepest molecule;
- write \chemnameinit{} after having written all the names in a chemical reaction lest the greatest depth in this reaction interfere with a future reaction.

Note that the assignments for the largest depth are global if gchemname = ⟨*true*⟩, which is the default behavior. They are local otherwise.

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

Name alignment 2

```
\chemnameinit{\chemfig{R-C(-[:-30]OH)=[:30]O}}
\schemestart
  \chemname{\chemfig{R'OH}}{Alcohol}
  \+
  \chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic acid}
  \arrow(.mid east--.mid west)
  \chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
  \+
  \chemname{\chemfig{H_2O}}{Water}
\schemestop
\chemnameinit{}
```

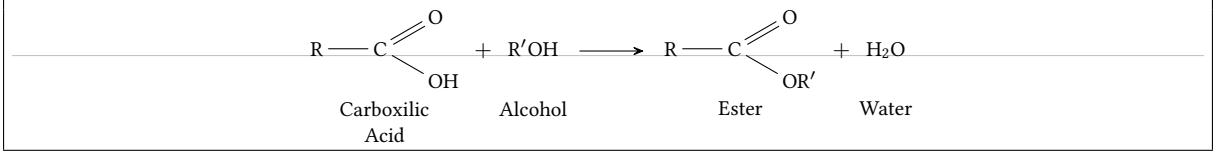


[7] In TEX terms, the depth is the dimension which extends vertically below the baseline.

Finally, to write a name on multiple lines, the command \\ encountered in a ⟨*name*⟩ causes a line break[8]:

Name on 2 lines

```
\schemestart
  \chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxilic\\Acid}
  \+
  \chemname{\chemfig{R'OH}}{Alcohol}
  \arrow(.mid east--.mid west)
  \chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
  \+
  \chemname{\chemfig{H_2O}}{Water}
\schemestop
\chemnameinit{}
```



If \chemname*{⟨*name*⟩} is written, the macro does not take into account the previous names.

Advanced usage

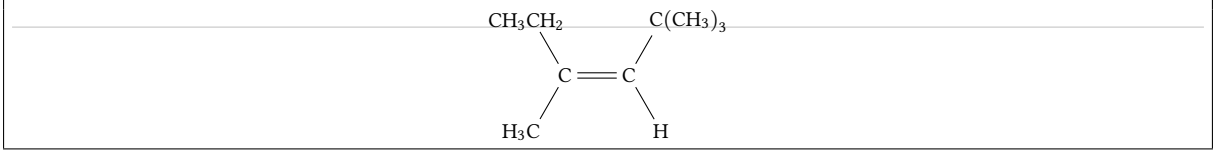
1 Separating atoms

The `separating atom mechanism` described previously extends each atom until the next capital letter or one of the characters - = ~ (! * < > @

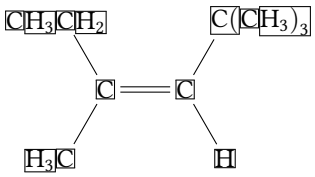
In certain cases this automatic separation produces incorrect atoms which can translate into an imperfect diagram. Consider this example molecule, noting that the "`(`" character is placed between braces so that chem*fig* doesn't incorrectly create a branch:

Alkene

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



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 chem*fig* 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:

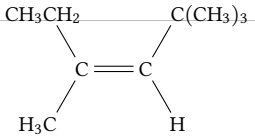


The character "`|`" forces splitting of the atom when it is encountered. Thus we can write `C|{(CH_3)_3}` to ensure that chem*fig* separates just two atoms here: "`C`" and "`{(CH_3)_3}`". The problem of the too-short bond is thus solved:

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

Alkene

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



2 Displaying atoms

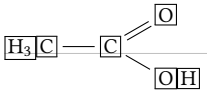
Once a molecule has been split into atoms, the macro `\printatom` is called internally by chem*fig* in order to display each atom. Its only argument consists of tokens read in the molecule code representing the current atom (e.g., "H_3"); when the `use atom strut` key is set to ⟨*true*⟩, a `\vphantom` is added to this argument (see page 31). This macro switches to math mode and displays its argument using the math font "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 ε-TeX ou ConTeXt.

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

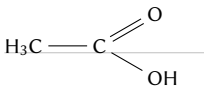
```
\fboxsep=1pt
\renewcommand*\printatom[1]{\fbox{\ensuremath{\mathrm{#1}}}}
\chemfig{H_3C-C(=[:30]O)(-[:-30]OH)}
```



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)}
```



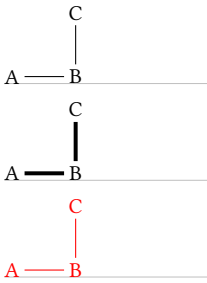
3 Arguments given to tikz

The ⟨*key*⟩ `chemfig style` contains ti*k*z instructions which will be passed to the `tikzpicture` environment in which the molecule is drawn. On the other hand, the The ⟨*key*⟩ `atom style` contains ti*k*z instructions which will be executed when each node; these instructions are added to the end of `every node/.style{<argument>}`, i.e. after the fhe following instructions: "`anchor=base,inner sep=0pt,outer sep=0pt,minimum size=0pt`".

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

Style choice

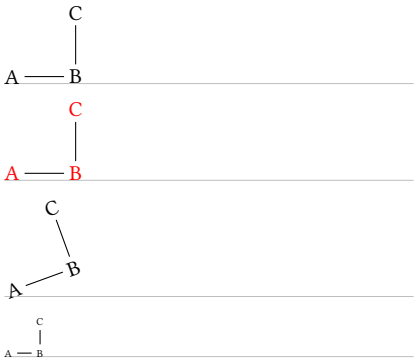
```
\chemfig{A-B-[2]C}\par\medskip
\setchemfig{chemfig style={line width=1.5pt}}\chemfig{A-B-[2]C}\par\medskip
\setchemfig{chemfig style=red}\chemfig{A-B-[2]C}
```



With `node style`, one can choose the colour of nodes drawn by ti*k*z, change the angle of the drawing or its scale:

Style choices

```
\chemfig{A-B-[2]C}\par\medskip
\setchemfig{atom style=red}\chemfig{A-B-[2]C}\par\medskip
\setchemfig{atom style={rotate=20}}\chemfig{A-B-[2]C}\par\medskip
\setchemfig{atom style={scale=0.5}}\chemfig{A-B-[2]C}
```



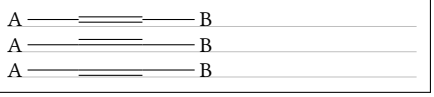
4 Shifted double bonds

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

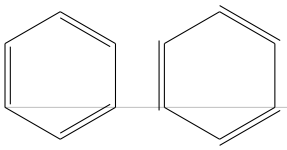
```
\chemfig{A-=-B}\par
\chemfig{A-=^-B}\par
\chemfig{A-=_-B}
```



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

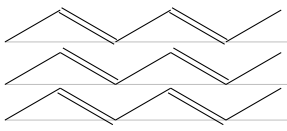
```
\chemfig{*6(-=-=-=)}\qquad
\chemfig{*6(-=_-=_-=_)}
```



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]}
```



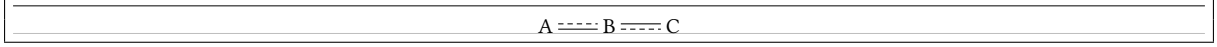
5 Delocalized double bonds

It is sometimes necessary to draw a double bond so that one line would be full and the other dashed. This feature is not hard-coded in chem*fig* since ti*kz*, with its "decorations.markings" library makes it possible.

Delocalized bonds

```
\catcode`\_=11
\tikzset{
  ddbond/.style args={#1}{
    draw=none,
    decoration={%
      markings,
      mark=at position 0 with {
```

```
        \coordinate (CF@startdeloc) at (0,\dimexpr#1\CF_doublesep/2)
        coordinate (CF@startaxis) at (0,\dimexpr-#1\CF_doublesep/2);
        },
      mark=at position 1 with {
        \coordinate (CF@enddeloc) at (0,\dimexpr#1\CF_doublesep/2)
        coordinate (CF@endaxis) at (0,\dimexpr-#1\CF_doublesep/2);
        \draw[dash pattern=on 2pt off 1.5pt] (CF@startdeloc)--(CF@enddeloc);
        \draw (CF@startaxis)--(CF@endaxis);
        }
      },
    postaction={decorate}
  }
}
\catcode`\_=8
\chemfig{A-[,,,,ddbond={+}]B-[,,,,ddbond={-}]C}
```



6 Saving a sub-molecule

chem*fig* 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

\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;
- a control sequence.

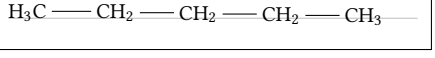
In all cases, if the alias is already defined you should not overwrite it with a new definition using \definesubmol. A warning will be issued to the user that the old alias will be overwritten by the new one. To override the definition of an alias made previously, use:

\redefinesubmol{⟨*name*⟩}{⟨*code*⟩}

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

Pentane

```
\definesubmol{xy}{CH_2}
  \chemfig{H_3C-!{xy}-!{xy}-!{xy}-CH_3}
```

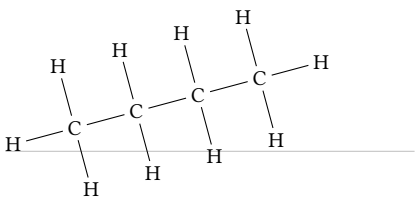


In this case the technique is not very interesting because "`!{xy}`" is just as long to type as the code it replaces.

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:

Butane

```
\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:

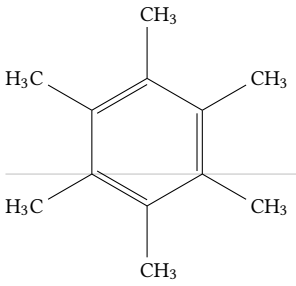
\definesubmol{⟨*name*⟩}[⟨*code1*⟩]{code2}

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:

Dual alias

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



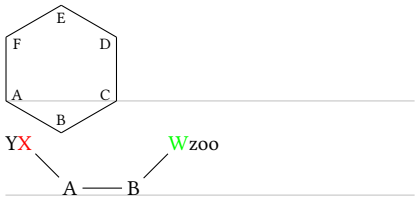
The sub-molecule saved with a ⟨*name*⟩ does not admit an argument when it is called after "`!`|". To define a sub-molecule admitting one or more arguments, place this ⟨*number*⟩ of arguments just after the ⟨*name*⟩. And the full syntax of `\definesubmol` is:

`\definesubmol{`⟨*name*⟩`}`⟨*number*⟩`[`⟨*code1*⟩`]{`⟨*code2*⟩`}`

In the ⟨*codes*⟩, the arguments must appear in their usual form "`#`⟨*n*⟩" where ⟨*n*⟩ is the argument number.

`\definesubmol` with arguments

```
\definesubmol\X1{-[,-0.2,,,draw=none]{\scriptstyle#1}}
\chemfig{*6((!\X A)-(!\X B)-(!\X C)-(!\X D)-(!\X E)-(!\X F)-)}

\definesubmol{foo}3[#3|\textcolor{#1}{#2}]{\textcolor{#1}{#2}|#3}
\chemfig{A(-[:135]!{foo}{red}XY)-B(-[:45]!{foo}{green}{W}{zoo})}
```

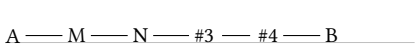


It should be noted that if the ⟨*number*⟩ of arguments is incorrect (negative or greater than 9), an error message will be issued and chem*fig* will consider that the sub molecule does not admit an argument.

Except in cases where the character "`#`" is followed by a number between 1 and ⟨*number*⟩ in which case it represents an argument, "`#`" are allowed in the sub-molecule codes.

Use of #

```
\definesubmol\X2{#1-#2-#3-#(3pt,3pt)#4}
\chemfig{A-!\X{M}{N}-B}
```



In this example, only #1 and #2 are understood as the arguments of the sub molecule `\X`. The other "`#`" are displayed as they are in the molecule (case of #3 and #4) or understood as the character specifying the fine adjustment of the offset of the bonds.

7 Placement of Atoms

7.1 Groups of atoms

In a group of atoms, the atoms are placed one after the other in a well-established order:

- the first one which is placed (which we will call "reference atom") is the one on which the bond arrives; in the case of the beginning of the molecule, the atom on the left is the reference atom;
- the atoms to the right of the reference atom are then placed from left to right;
- atoms to the left of the reference atom are finally placed from right to left.

In the group of atoms thus formed, the baselines of each atom are on *the same horizontal line*, in other words, the atoms are all aligned on the same horizontal line.

In the example below whose code would be "`\chemfig{A[:-60,,,3]BCDEF}`" the reference atom of the 2nd group of atoms is "D" because the bond is requested to arrive on the 3rd atom. Below each atom of this group is the sequence number in which the atom is displayed:

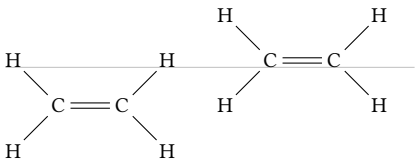
7.2 Vertical alignment

The `baseline` is used to finely control the vertical placement of the molecule in relation to the baseline of the current paragraph.

It is set to ⟨*0pt*⟩ by default, and in this case, the first atom encountered (whether empty or not) is the one placed on the baseline of the current paragraph, shown in gray on the examples in this manual. The choice of this first atom therefore conditions the placement of all the others, and often influences the placement of the entire molecule.

First atom

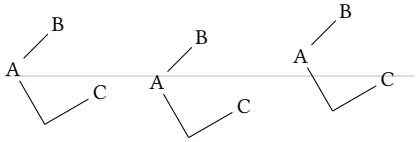
```
\chemfig{H-[7]C(-[5]H)=C(-[1]H)-[7]H}\qquad
\chemfig{C(-[3]H)(-[5]H)=C(-[1]H)-[7]H}
```



An arbitrary dimension can be specified to vertically shift the molecule by this value with the syntax `baseline` = ⟨*dimension*⟩:

Vertical shift

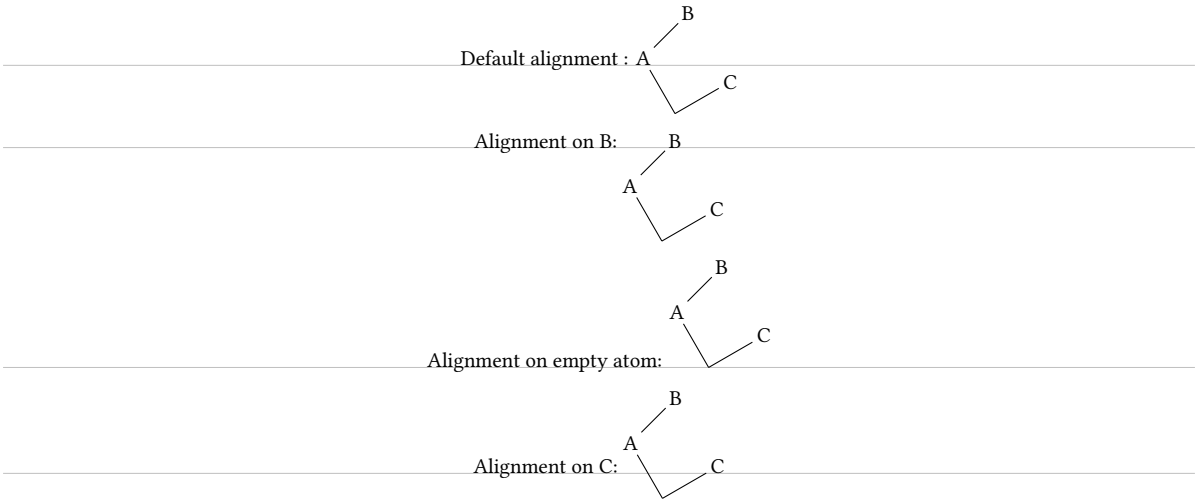
```
\chemfig{A(-[:-60]-[:30]C)-[:45]B}\qquad
\chemfig[baseline=5pt]{A(-[:-60]-[:30]C)-[:45]B}\qquad
\chemfig[baseline=-5pt]{A(-[:-60]-[:30]C)-[:45]B}
```



With the syntax `baseline` = ⟨*(name)*⟩ (the name must be in parentheses), we specify that the baseline of the molecule is at the node named ⟨*name*⟩. The name of the atom can be the one assigned automatically by chem*fig* (of the form n⟨*a*⟩-⟨*b*⟩) or a name given by the user using the syntax `@{`⟨*name* ⟩`}` (see page 21).

Alignment on atoms

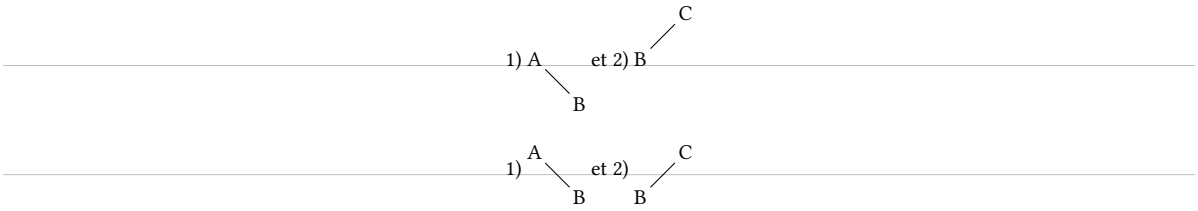
```
Default alignment :      \chemfig{A(-[:-60]-[:30]C)-[:45]B}\medbreak
Alignment on B:          \chemfig[baseline=(b.base)]{A(-[:-60]-[:30]C)-[:45]@{b}B}\medbreak
Alignment on empty atom: \chemfig[baseline=(vide)]{A(-[:-60]@{vide}-[:30]C)-[:45]B}\medbreak
Alignment on C:          \chemfig[baseline=(c.base)]{A(-[:-60]-[:30]@{c}C)-[:45]B}
```



It is possible to name ti*k*z nodes. Thus, if we want to vertically center several molecules on the current baseline, we set `baseline` to ⟨*(current bounding box.center)*⟩.

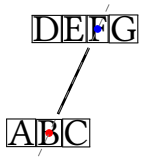
Centered Alignment

```
1) \chemfig{A-[:-45]B} et 2) \chemfig{B-[:45]C}\bigbreak

\setchemfig{baseline=(current bounding box.center)}% vertical centering of the following molecules
1) \chemfig{A-[:-45]B} et 2) \chemfig{B-[:45]C}
\setchemfig{baseline=0pt}% back to default value
```



7.3 Bonds between atoms

A bond starting from an atom would, if extended, pass through the centre of its bounding box. The atom placed at the end of the bond has its center of its bounding box is in the extension of the bond. Therefore, a bond between two atoms extends through the centers of their bounding boxes, as shown in this example:



This mechanism can create misalignments between groups of atoms that are particularly visible when the bonds are horizontal. Everything works well when the atoms have the same vertical dimensions; however, if a departure atom is high (with exponent) or deep (with subscript) and the arrival atom has a different vertical dimension, the alignment is broken.

Horizontal alignment

```
\large\setchemfig{atom sep=2em}
\chemfig{A^1-B-C-D}\par
\chemfig{E_1-F-G-H}
```

$A^1 - B - C - D$
$E_1 - F - G - H$

This phenomenon can be highlighted by making the bounding boxes of the atoms visible, where we can clearly see that the atoms "B" and "F" have bounding boxes whose height takes into account the heights of the previous atoms:

Horizontal placement and bounbding boxes

```
\large\setchemfig{atom sep=2em}
\fboxsep=-0.2pt \fboxrule0.2pt
\renewcommand\printatom[1]{\fbox{\ensuremath{\mathrm#1}}}
\chemfig{A^1-B-C-D}\par
\chemfig{E_1-F-G-H}
```

$A^1 - B - C - D$
$E_1 - F - G - H$

Since the bounding box of the first atom has a different vertical extent than the next one, a vertical offset is created for the subsequent bounding boxes. This offset is clearly visible when the bonds are horizontal.

The behavior prior to version 1.7 of chem*fig* masked this effect on the first atom following the problematic bounding box by adding the `\vphantom` of the previous atom to the argument of each `\printatom`. You can revert to this behavior by setting the `use atom strut` key to ⟨*true*⟩.

Horizontal placement and bounding boxes

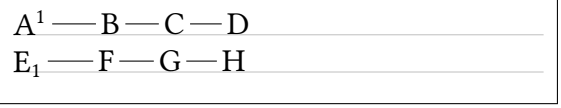
```
\large\setchemfig{atom sep=2em}
\fboxsep=-0.2pt \fboxrule0.2pt
\renewcommand\printatom[1]{\fbox{\ensuremath{\mathrm#1}}}
\chemfig{A^1-B-C-D}\quad
\chemfig[use atom strut=true]{A^1-B-C-D}

\chemfig{E_1-F-G-H}\quad
\chemfig[use atom strut=true]{E_1-F-G-H}
```

$A^1 - B - C - D$ $A^1 - B - C - D$
$E_1 - F - G - H$ $E_1 - F - G - H$

Since no automatic solution is satisfactory, this problem can be manually circumvented by creating an end atom that is a “strut” equal to |vphantomX|: thus, the starting atom has a “normal” height and no offset will affect the following group of atoms. A sub-molecule is used here for greater conciseness.

Bypassing vertical position

```
\large\setchemfig{atom sep=2em}
\definesubmol\I{\vphantom{X}}
\chemfig{A^1|!\I-B-C-D}\par
\chemfig{E_1|!\I-F-G-H}
```



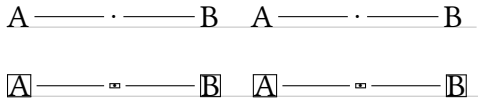
The disadvantage is that the first bond is too long because the departure atom now has a zero horizontal dimension.

7.4 La macro \chemskipalign

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

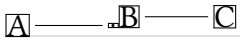
```
\large
\chemfig{A-.-B}\quad
\chemfig{A-\chemskipalign.-B}\par\bigskip
\fboxsep=0pt
\renewcommand\printatom[1]{\fbox{\ensuremath{\mathrm{#1}}}}
\chemfig{A-.-B}\quad
\chemfig{A-\chemskipalign.-B}
```



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 \chemskipaligncommand

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

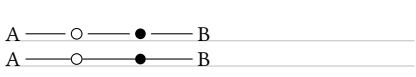


This feature can sometimes be useful. Suppose we want to draw the following molecule

We can define commands which will draw the empty and full disks with ti*k*z. 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 8.

Use of \chemskipalign and #

```
\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}
```



8 The macro \charge

8.1 Overview

The macro load, which requires two mandatory arguments, allows to arrange elements (called ⟨*charges*⟩) around an ⟨*atome*⟩; its syntax is

\charge{[⟨*general parameters*⟩]⟨*position*⟩[⟨*tikz code*⟩]=⟨*charge*⟩}{⟨*atom*⟩}

where:

- the ⟨*atom*⟩ is usually one or two letters, but it can also be empty;
- the ⟨*charge*⟩ is an arbitrary content that will be placed around the atom. Few constraints exist on this content: it can be text (in math mode if needed), or even ti*k*z code or a molecule drawn with `chemfig`;
- the ⟨*general parameters*⟩ (optional) are a list of key/values specifying the options that this execution of the macro must satisfy. These keys/values are described below;
- the ⟨*position*⟩ is "⟨*angle*⟩`:`⟨*shift*⟩", but it is possible to specify only the ⟨*angle*⟩, in which case, the ⟨*shift*⟩ will be equal to `0pt`;
- the optionnal ⟨*tikz code*⟩ sets the options given to the ti*k*z macro `\node`, which places the ⟨*charge*⟩.

8.2 Parameters

The ⟨*keys*⟩ = ⟨*values*⟩ available in the ⟨*general parameters*⟩ are:

⟨*keys*⟩	default ⟨*values*⟩	Description
`debug`	`false`	boolean which, when ⟨*true*⟩, draws the outlines of the nodes receiving the ⟨*atoms*⟩ (in green), the ⟨*loads*⟩ (in blue) and the ⟨*charge*⟩ (in red).
`macro atom`	`\printatom`	macro receiving the ⟨*atom*⟩ as argument.
`circle`	`false`	boolean which, when ⟨*true*⟩, puts the ⟨*atom*⟩ in a circular node; otherwise, the node is rectangular.
`macro charge`	⟨*vide*⟩	macro (e.g., `\printatom` or `\ensuremath`) receiving each charge as an argument.
`extra sep`	`1.5pt`	node size increment of the ⟨*atom*⟩ to put the ⟨*charges*⟩: it is the value passed to the `inner sep` of ti*k*z.
`overlay`	`true`	boolean which, when ⟨*true*⟩, draws the ⟨*charges*⟩ "overlay", i.e. outside the final bounding box.
`shortcuts`	`true`	boolean which, when ⟨*true*⟩, activates the shortcuts"`\.`," "`\:`," "`|` and "`\"`" to draw Lewis formuas.
`lewisautorot`	`true`	boolean which, when ⟨*true*⟩, automatically rotates "`\:`," "`|` and "`\"`".
`.radius`	`0.15ex`	radius of the point used to plot "`\.`" and "`\:`".
`:sep`	`0.3em`	separation between the two dots of "`\:`".
`.style`	`fill=black`	ti*k*z style used to draw the "`\.`" and "`\:`" dots.
`"length`	`1.5ex`	length of the rectangle "`\ `"and the line "`|`".
`"width`	`.3ex`	width of the rectangle `\"`.
`"style`	`black,line width=0.4pt`	ti*k*z style used to draw the rectangle `\"`.
`|style`	`black,line width=0.4pt`	TIKZ style used to draw the line `|`.

It is possible to set some (or all) of these parameters by running the macro

\setcharge{⟨*keys*⟩=⟨*values*⟩}

and reset all parameters to their default values with

\resetcharge

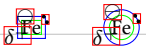
The `\charge` macro places the ⟨*charges*⟩ out of the bounding box (unless otherwise specified in the ⟨*parameters*⟩) while `\Charge` places them into the bounding box.

The ⟨*angle*⟩ is the location on the boundary of the node where the ⟨*charge*⟩ is placed. This ⟨*angle*⟩ can be expressed in degrees or it can be a boundary anchor in the sense of ti*k*z, like "south east." The ⟨*shift*⟩ is a TEX-dimension and represents an additional length between the boundary of the node containing the ⟨*atom*⟩ and the place where the ⟨*charge*⟩ is placed. Unless otherwise specified in the ⟨*tikz code*⟩, the *center* anchor of ⟨*charges*⟩.

In the two following examples, debug will be set to ⟨*true*⟩ in order to better perceive the changes induced by the modification of the parameters. In addition, the macro \Charge will be used so that the bounding boxes take into account the charges. Here we see the influence of the node shape on the placement of the charges:

Generic example

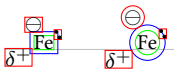
```
\setcharge{debug}
Default then circle:
\Charge{30=\:,120=$\ominus$,210=$\delta^+$}{Fe}\qquad
\Charge{[circle]30=\:,120=$\ominus$,210=$\delta^+$}{Fe}
```

Default then circle: 

To place the loads ⊖ and δ^+ further away, we can play on the ⟨*shift*⟩ or better, on the anchor: the ⟨*angle*⟩ where the load is placed is stored in the macro \chargeangle, so it is wise to choose the anchor 180+\chargeangle. It is also possible to specify a circular node to place the charge.

Fine positioning

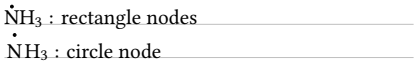
```
\setcharge{debug}
\Charge{30=\:,120:3pt=$\ominus$,210:5pt=$\delta^+$}{Fe}\qquad
\Charge{[circle]30=\:,
        120[circle,anchor=180+\chargeangle]=$\ominus$,
        210[anchor=180+\chargeangle]=$\delta^+$}{Fe}
```



It is important to note that circular nodes have dimensions *sometimes very different* from the "classic" rectangular nodes, especially in terms of horizontal and vertical extent. It is therefore advisable to set ⟨*true*⟩ the boolean key circle knowingly.

Circular nodes

```
\chemfig{\charge{90=\.}{N}H_3} : rectangle nodes\smallbreak
\chemfig{\charge{[circle]90=\.}{N}H_3} : circle node
```



8.3 Lewis formula

When shortcut is ⟨*true*⟩, the shortcuts "\.», "\:», "\| and "\"» are actgive to draw Lewis formulas "•», "⁘», "⁞» et "𝄀». You can deactivate them at any time with the \disableshortcuts macro and reactivate them with \enableshortcuts.

When the boolean shortcut is ⟨*false*⟩ or the shortcuts have been disabled with \disableshortcuts, shortcuts "\:", "\|" and "\"" are no longer programmed to draw Lewis formulas, so the macros \chargedot, \chargeddot, \chargeline and \chargerect must be used instead.

The key lewisautorot, which is ⟨*true*⟩ by default, acts on "⁘", "⁞" and "𝄀" and rotates them.

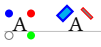
Autorot

```
\Charge{60=\:,150=\"}{A} et
\Charge{[lewisautorot=false]60=\:,150=\"}{A}
```

The customization of Lewis' formulas is done via the macro \setcharge or via the optional argument of \charge by acting on the keys .radius, :sep, .style, |style, "length, "width and "style. It is also possible to modify these keys for each formula with their optional argument which receives a list of keys = ⟨*values*⟩.

Ccustomization

```
\Charge{[.radius=1.5pt,.style={draw=gray}]
   45  =\.[{.style={draw=none,fill=red}}],
   135 =\.[{.style={draw=none,fill=blue}}],
   -45 =\.[{.style={draw=none,fill=green}}],
   -135=\.}{A}\quad
\Charge{
   45 =\"[{"style={draw=red,fill=gray}}],
   135=\"[{"width=3pt,"style={line width=.8pt,draw=blue,fill=cyan}}]}{A}
```

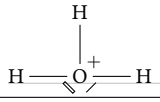


8.4 Integration in chem*fig*

A macro \charge can take the place of an atom.

Charge in chem*fig*

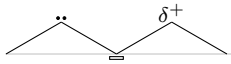
```
\chemfig{H-\charge{45:1.5pt=$\scriptstyle+$,-45=\|,-135=\"}{O}(-[2]H)-H}
```



However, chem*fig* has been modified so that the bonds are *joined* when the dimensions of an atom is zero, that is, if its width, height and depth are all `0pt`. This was previously only the case if the atom was empty. This new feature makes it easy to place charges in carbon chains.

Charge in chain

```
\chemfig{[:30]-\charge{90=\:}{}
-[:-30]\charge{-90=\"}{}-\charge{90:2pt=$\delta^+$}{}-[:-30]}
```



9 Stacking

The macros

\chemabove[⟨*dim*⟩]{⟨*code*⟩}{⟨*stuff*⟩}

and

\chembelow[⟨*dim*⟩]{⟨*code*⟩}{⟨*stuff*⟩}

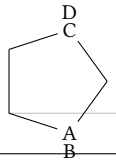
place the ⟨*stuff*⟩ above and below the ⟨*code*⟩ respectively at a vertical distance ⟨*dim*⟩, without changing the bounding box of ⟨*code*⟩. The optional argument allows, if written, to specify this dimension at each call. If the optional argument is not used, a default size will be taken: its value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

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 chem*fig* from starting a new atom on these letters:

Staking in rings

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

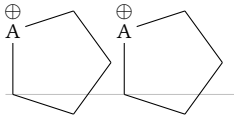


The \Chemabove and \Chembelow commands work in the same way, except that the bounding box takes into account the ⟨*stuff*⟩ placed above or below.

What's the difference between \chemabove and \charge for placing one item above or below another?

\chemabove or \charge

```
\chemfig{*5(----\chemabove{A}{\oplus}-)}
\chemfig{*5(----\charge{90[anchor=-90]=$\oplus$}{A}-)}
```



By default, the two macros give very similar results. However, there are differences in their use:

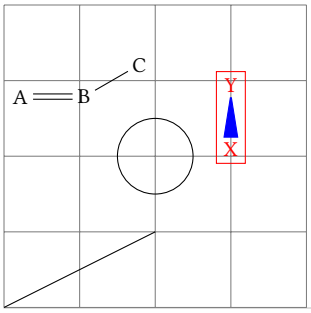
- \chemabove and \chemabelow can only be used in the argument of chemfig, which is not the case for \charge;
- the \charge macro requires ti*k*z, whereas \chemabove and \chemabelow use low-level TEX primitives and are therefore fast and independent of any package.

10 Using \chemfig in the tikzpicture environment

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

\chemfig inside tikzpicture

```
\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}
```

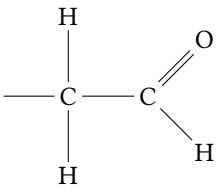


11 Annotated examples

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 chem*fig* 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 chem*fig* allows some flexibility in encoding molecules. One can see how each is put together and adopt the methods with which one is most comfortable.

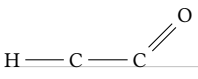
11.1 Ethanal

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

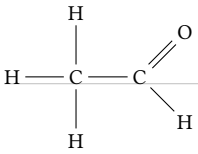
```
\chemfig{H-C-C=[1]O}
```



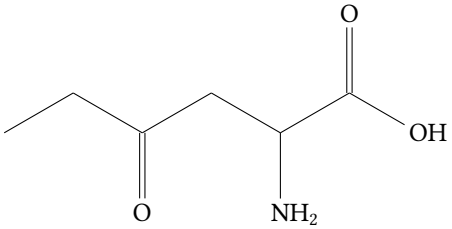
The three hydrogen atoms still have to placed at the correct orientation with the help of predefined angles. The first is at 90° with the branch "`(-[2]H)`", the second at 270° with "`(-[6H])`", and the one on the right at 315° with "`(-[7]H)`":

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: 

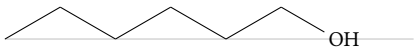
As is often the case for most molecules, there are several methods and for each several different ways of getting the result. Here we will look at four different methods.

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)

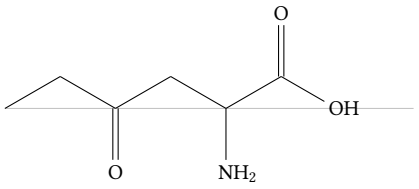
```
\chemfig{[:30]--[:-30]--[:-30]--[:-30]OH}
```



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}
```

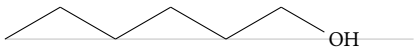


11.2.2 Relative angles

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

Structure (relative angles)

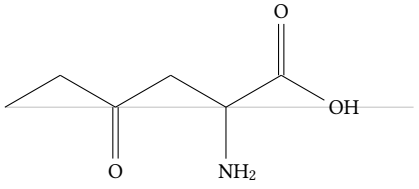
```
\chemfig{[:30]--[::-60]--[::-60]--[::-60]OH}
```



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:

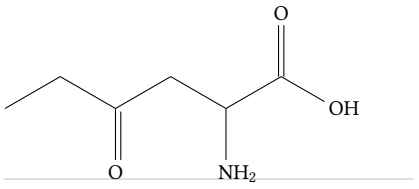
Backbone (ring)

```
\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)}
```

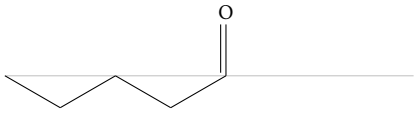


11.2.4 Nested rings

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

Backbone (nested rings)

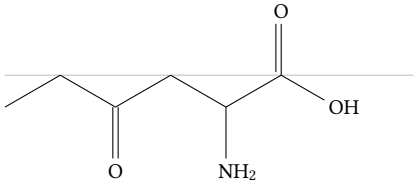
```
\chemfig{*6(--*6(--=O))}
```



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)

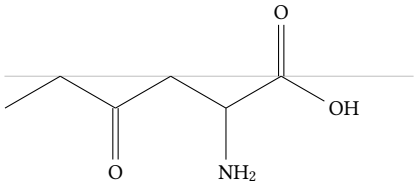
```
\chemfig{*6((-)-(=O)-*6(-(-NH_2)-(-OH)=O))}
```



A close look shows that the second line segment of the double bond to the oxygen atom is *inside* the incomplete 6-ring[9] 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

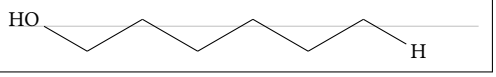
The goal here is to represent the glucose molecule according to several different conventions.

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}
```

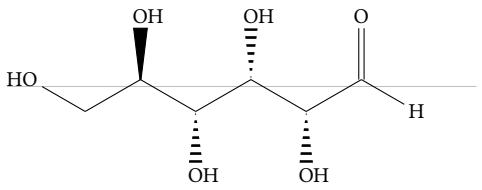


Adding the branches is no problem. We use predefined absolute angles:

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

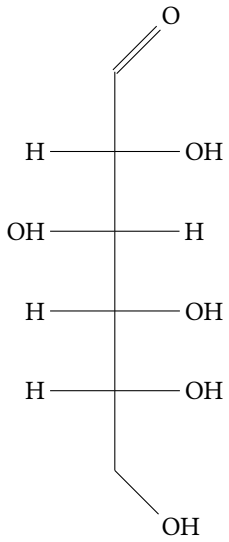
Glucose, skeleton diagram

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



11.3.2 Fisher projection

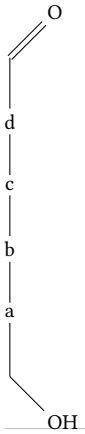
The goal is to get the molecule below:



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

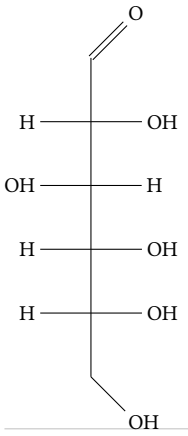
```
\chemfig{[2]OH-[3]-a-b-c-d-=[1]O}
```



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}`":

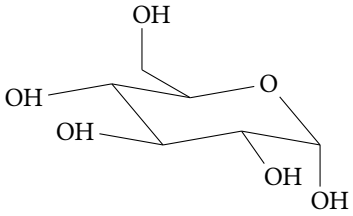
Glucose (Fisher projection)

```
\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: $-50°$, $10°$, $-10°$, $130°$, $190°$.

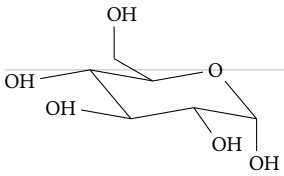
Structure

```
\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:

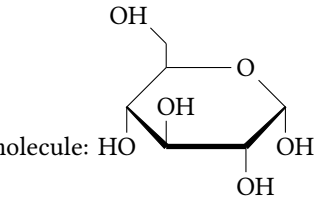
Chair representation

```
\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:

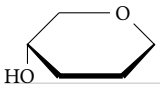


First of all we will choose the longest chain, which starts at the "HO" group on the left and continues through fives sides of the ring. The ring will be closed with a hook. For the vertical bond which leaves from the first "HO" group, we need to specify that it will leave from the second atom using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

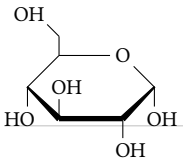
```
\chemfig[cram width=2pt]{HO-[2,0.5,2]?<[7,0.7]-[,,,,
line width=2pt]>[1,0.7]-[3,0.7]O-[4]?}
```



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:

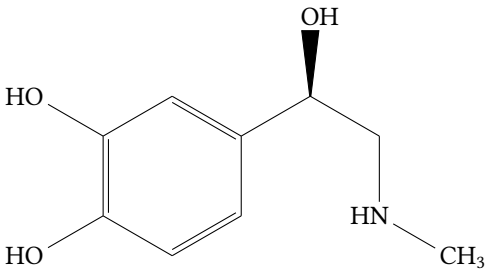
Projection de Haworth

```
\chemfig[cram width=2pt]{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:



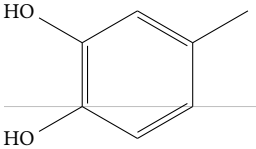
We are going to use two different methods.

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

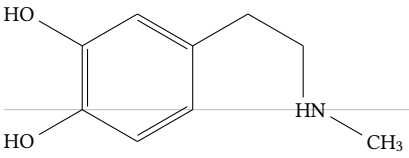
```
\chemfig{*6((-HO)-=-(-)=-(-HO)=)}
```



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

Adrenaline, step two

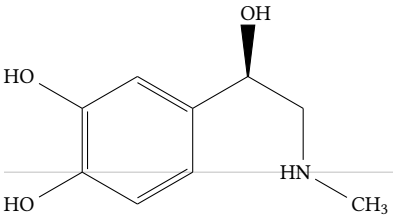
```
\chemfig{*6((-HO)-=-(--[::-60]-[::-60]
HN-[::+60]CH_3)=-(-HO)=)}
```



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)=)}
```



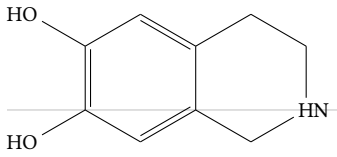
11.4.2 Using two rings

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

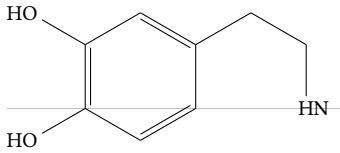
```
\chemfig{*6((-HO)-=*6(--HN---)-=-(-HO)=)}
```



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

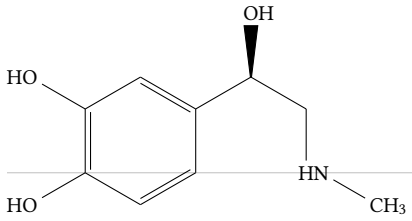
```
\definesubmol{&}{-[,,,,draw=none]}
\chemfig{*6((-HO)-=*6(!&!&HN---)-=-(-HO)=)}
```



The rest becomes easy; just add the branches to the right vertices:

Adrenaline, step three

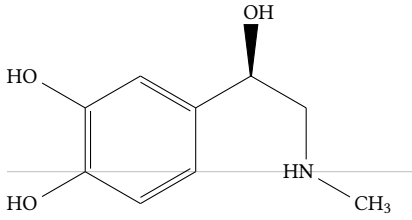
```
\definesubmol{&}{-[,,,,draw=none]}
\chemfig{*6((-HO)-=*6(!&!&HN(-CH_3)--(<OH)-)-=-(-HO)=)}
```



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`":

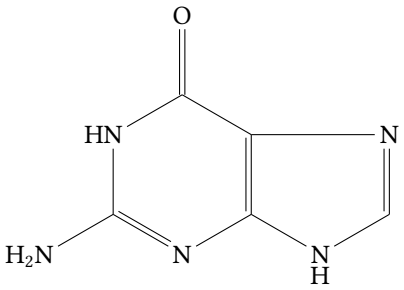
Adrenaline

```
\definesubmol{&}{-[,,,,draw=none]}
\definesubmol{&&}{-[,,,2,draw=none]}
\chemfig{*6((-HO)-=*6(!&!{&&}HN(-CH_3)-[,,2]-(<OH)-)-=-(-HO)=)}
```



11.5 Guanine

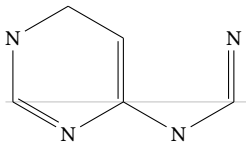
We will draw the guanine molecule:



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

Guanine, skeleton

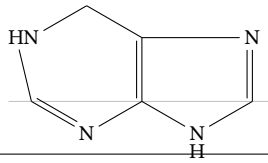
```
\chemfig{*6(=N-*6(-N-=N)=--N-)}
```



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

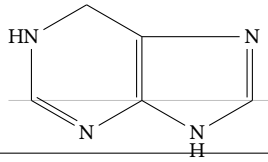
```
\chemfig{*6(=N-*6(-\chembelow{N}{H}-=N?)=?--HN-)}
```



We note that one bond leaves from the wrong atom[10]! 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

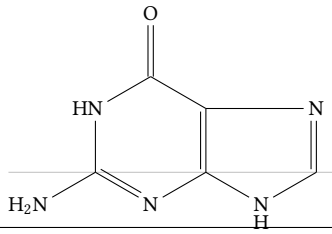
```
\chemfig{*6(=N-*6(-\chembelow{N}{H}-=N?)=?--HN-[,,2])}
```



Simply add the branches to the right vertices. Note especially the branch leaving the first vertex of the first 6-ring "`(-N_2N)`":

Guanine

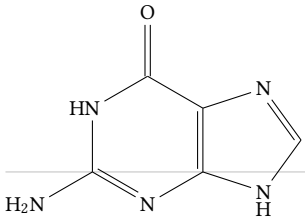
```
\chemfig{*6((-H_2N)=N-*6(-\chembelow{N}{H}-=N?)=?-(=O)-HN-[,,2])}
```



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

Guanine with 5-ring

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



[10] This seems illogical because the angle of the bond from the `HN` group toward the first vertex lies between $-90°$ and $90°$; chem*fig* 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 $-90°$). chem*fig* 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.

12 How to ...

12.1 Write a colored atom

Since the package xcolor is loaded by ti*kz*, itself loaded by chem*fig*, 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.

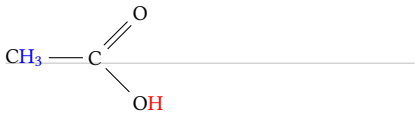
Colors

```
\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 starts 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 chem*fig* to cut the first atom just after "C" with the character "|" and then include \color{blue}H_3 between braces so that chem*fig* does not stop the atom 2 before the uppercase "H" which would leave the color change alone and therefore ineffective in an atom:

Colors

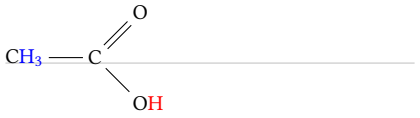
```
\chemfig{C|{\color{blue}H_3}-C(=[1]O)-[7]O|{\color{red}H}}
```



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[11], which often gives good results. We see here that with \rlap, the horizontal alignment of atoms is preserved:

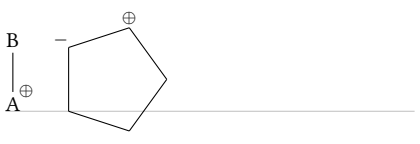
Charge and bond

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

The macro \charge allows this task to be performed simply and accurately.

Placing charges

```
\chemfig{\charge{[extra sep=0pt]45[anchor=180+\chargeangle]=%
$\scriptstyle\oplus$}{A}-[2]B}
\qquad
\chemfig{*5(---\charge{90:2pt=$\scriptstyle\oplus$}{}-%
\charge{135:2pt=$\scriptstyle-$}{}-)}
```



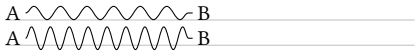
[11] If you have to put the charge at the left of the atom, you must use the command \llap.

12.3 Draw a curve bond

We have already seen that with the ti*kz* library “decorations.pathmorphing”, we can draw a wavy bond:

Wavy bond

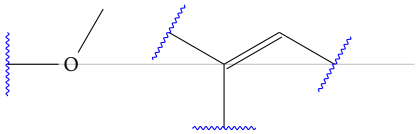
```
\chemfig{A-[,3,,,decorate,decoration=snake]B}

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



It is also possible to define a ”wb” sub-molecule to draw a wavy bond perpendicular to the previous bond:

Wavy bond

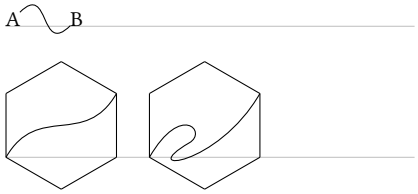
```
\definesubmol{wb}{%
  (-[::90,.5,,,draw=none]%
  -[::180,,,,blue,decorate,decoration={%
    snake,amplitude=0.2mm,segment length=0.75mm
    }%
  ])}
\chemfig{!{wb}-O-[:60]}
\qquad
\chemfig{([:150]-!{wb})([:-90]-!{wb})=_[:30]-[:-30]!{wb}}
```



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 Draw a polymer element

The macro \polymerdelim, until now undocumented and in the test phase, becomes officially released in chem*fig* with version 1.33. Its syntax is as follows:

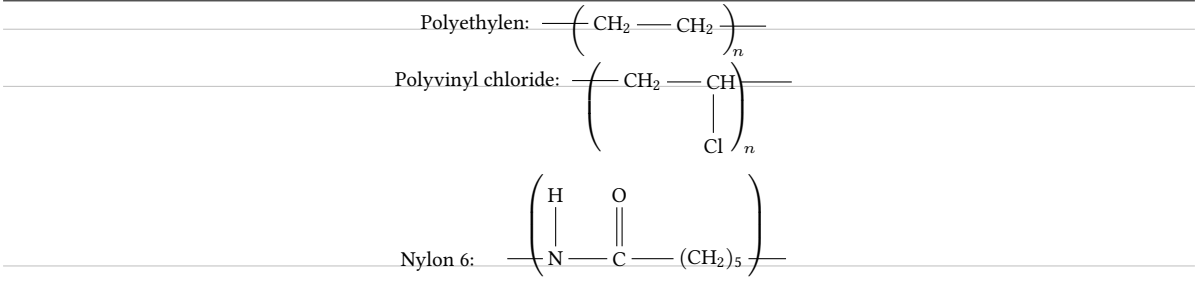
\polymerdelim[⟨*keys*⟩=⟨*values*⟩]{⟨*node1*⟩}⟨*node2*⟩}

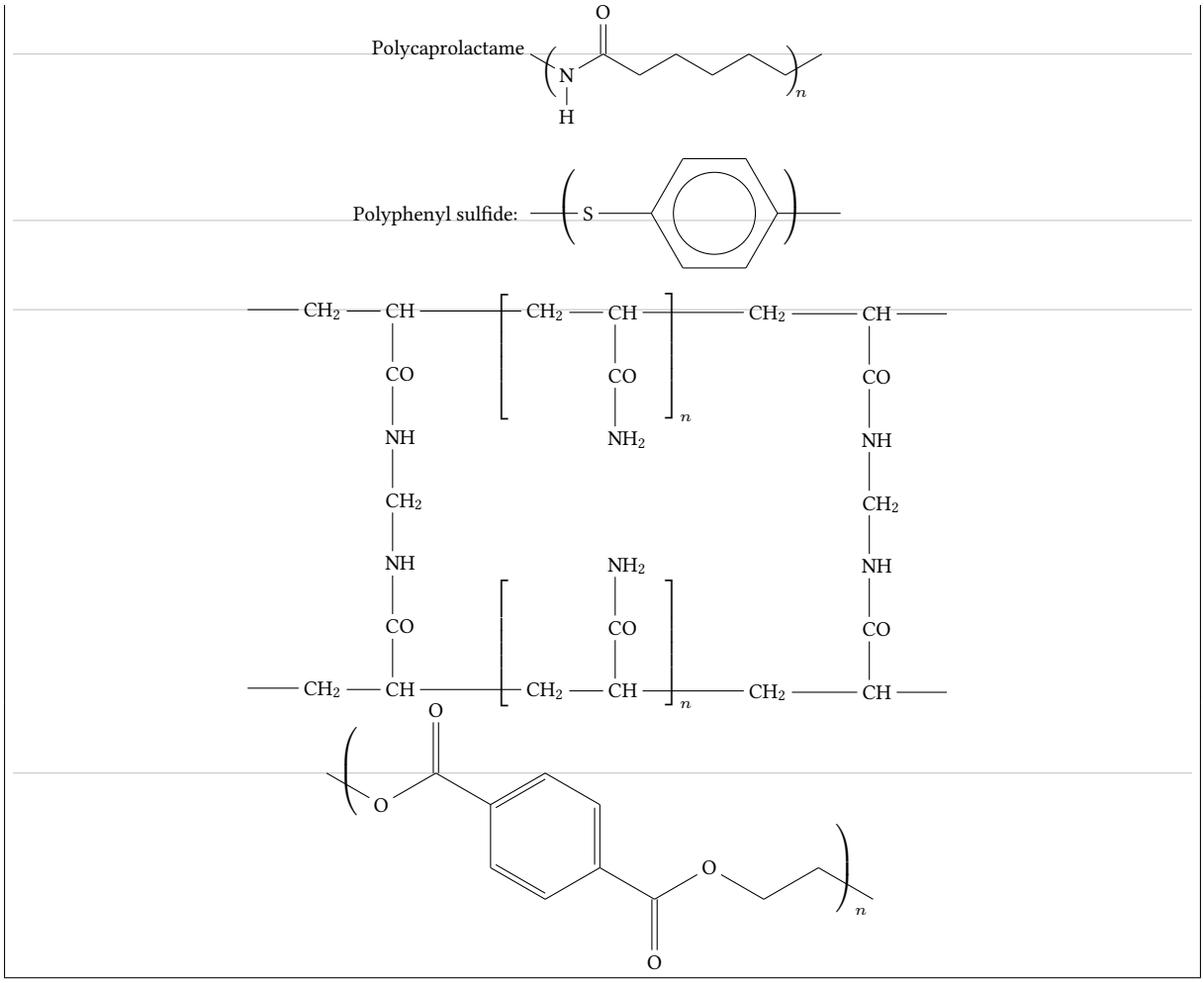
The effect, after possibly *two* compilations, is to place vertical delimiters at the specified nodes. The parameters are specified via the ⟨*keys*⟩ and ⟨*values*⟩, which are listed below, default values and actions.

⟨*keys*⟩	default ⟨*values*⟩	Action
delimiters	()	Defines the delimiters. If these delimiters are brackets, write delimiters={[]}.
height	10pt	Defines the height (above the node) of the delimiters.
depth	⟨*vide*⟩	Defines the depth (below the node) of the delimiters. If the ⟨*value*⟩ is empty, then the depth is equal to the height.
h align	true	Boolean which, when ⟨*false*⟩, places the 2nd delimiter on the 2nd node, at the risk that the delimiters are not on the same horizontal line.
auto rotate	false	Boolean which, when ⟨*true*⟩ and h align = ⟨*false*⟩, automatically turns the delimiters to be perpendicular to the line that connects the two nodes.
rotate	0	When h align = ⟨*false*⟩ and auto rotate = ⟨*false*⟩, sets the rotation angle of the two delimiters.
open xshift	0pt	Defines the horizontal offset of the opening delimiter.
close xshift	⟨*vide*⟩	Defines the horizontal offset of the closing delimiter. If the ⟨*value*⟩ is empty, then this offset becomes opposite to the offset of the opening delimiter.
indice	n	Defines the indices that will be placed at the right bottom of the closing delimiter.

Polymers

```
Polyethylen:
\chemfig{\vphantom{CH_2}-[@{op,.75}]CH_2-CH_2-[@{cl,0.25}]}
\polymerdelim[height = 5pt, indice = \!\!n]{op}{cl}
\bigskip

Polyvinyl chloride:
\chemfig{\vphantom{CH_2}-[@{op,1}]CH_2-CH(-[6]Cl)-[@{cl,0}]}
\polymerdelim[height = 5pt, depth = 25pt, open xshift = -10pt, indice = \!\!n]{op}{cl}
\bigskip

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

Polycaprolactame
\chemfig[atom sep = 2em]{[:-30]-[@{left,.75}]N(-[6]H)-[:30](=[2]O)--[:30]--[:30]--[@{right,0.25}:30]}
\polymerdelim[height = 5pt, indice = \!\!n]{left}{right}
\bigskip

Polyphenyl sulfide:
\chemfig{\vphantom{S}-[@{op,.75}]S-(**6(---(-[@{cl,0.25}])---))}
\polymerdelim[delimiters = (), height = 15pt, indice = {}]{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)-}
\polymerdelim[delimiters ={[]}, height = 5pt, depth = 40pt, indice = n]{upleft}{upright}
\polymerdelim[delimiters ={[]}, height = 40pt, depth = 5pt, indice = n]{downleft}{downright}

\chemfig{-[@{op,.5}:-30]O-[::60](=[::60]O)-[::-60]*6(-=-(-(=[::-60]O)-[::60]O-[::-60]-[::60]-[@{cl,.5}::-60])=-=)}
\polymerdelim[height=6ex, indice=n, h align=false]{op}{cl}
```





12.5 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}
```

12.6 Add text above bonds and arc to angles

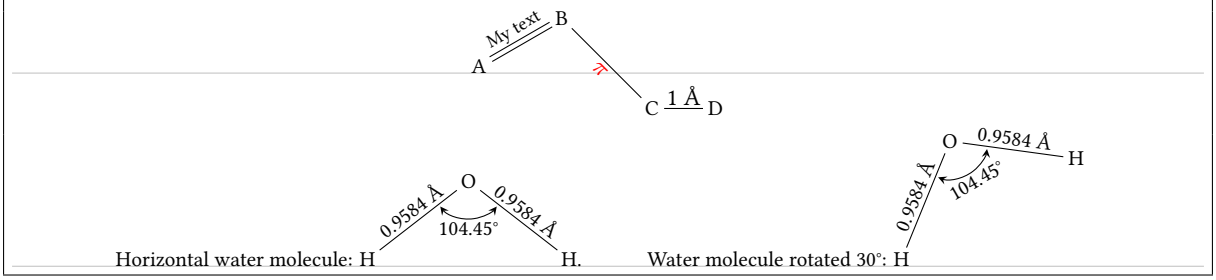
Once we have understood that the character "`@`" can put a "global" ti*k*z node, that is to say a node accessible after the molecule has been drawn, everything that ti*k*z can do with nodes (that is to say a lot of things) becomes possible.

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.

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 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\angstrom{\mbox{\normalfont\AA}}
\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 \angstrom}
\medskip

Horizontal water molecule: \chemfig{@{1}H-[::37.775,2]@{2}O-[::-75.55,2]@{3}H}.
\namebond{1}{2}{\footnotesize0.9584 \angstrom}
\namebond{2}{3}{\footnotesize0.9584 \angstrom}
\arclabel{0.5cm}{1}{2}{3}{\footnotesize104.45\textdegree}
\qquad
Water molecule rotated 30\textdegree: \chemfig{[:30]@1H-[::37.775,2]@2O-[::-75.55,2]@3H}
\namebond12{\footnotesize0.9584 \angstrom}
\namebond23{\footnotesize0.9584 \angstrom}
\arclabel{0.5cm}{1}{2}{3}{\footnotesize104.45\textdegree}
```



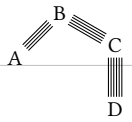
12.7 Dessiner des liaisons multiples

Again, the “decorations.markings” library allows to draw multiple bonds:

Liaisons multiples

```
\catcode`_=11
\tikzset{nbond/.style args={#1}{%
  draw=none,%
  decoration={%
    markings,%
    mark=at position 0 with {\coordinate (CFstart@) at (0,0);},
    mark=at position 1 with {%
    \foreach\CF_i in{0,1,...,\number\numexpr#1-1}{%
      \pgfmathsetmacro\CF_nbondcoeff{\CF_i-0.5*(#1-1)}%
      \draw ([yshift=\CF_nbondcoeff\CF_doublesep]CFstart@)--(0,\CF_nbondcoeff\CF_doublesep);
      }%
    }
    },
  postaction={decorate}
  }
```

```
}
\catcode`\_=8
\chemfig{A-[1,,,,nbond=4]B-[:-30,,,,nbond=5]C-[6,,,,nbond=6]D}
```



Reaction schemes

Following several requests from users, it had become evident that chem*fig* had a weakness regarding the drawing of reaction schemes. The gap is now filled. Therefore, chem*fig* 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 has the following syntax:

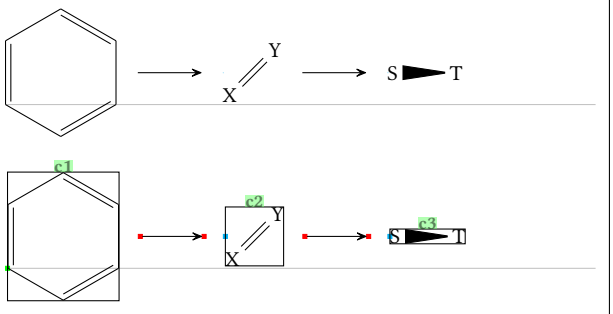
`\schemestart[angle,coeff,style][position]` ⟨*reaction*⟩ `\schemestop`

Important: one of the next versions will change the syntax of optional arguments. The second will be removed, and the first will be used to contain a list of `keys` = ⟨*values*⟩. It is therefore advisable to use the following ⟨*keys*⟩ to make the settings allowed by the optional arguments: `arrow angle`, `arrow coeff`, `arrow style`, and `init anchor`.

As shown in this example, `debug information` is either hidden or displayed with the ⟨*key*⟩ `scheme debug` and the value ⟨*true*⟩ ou ⟨*false*⟩:

Example 1

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

\setchemfig{scheme debug=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 `scheme debug` ⟨*key*⟩. It consists of:

  - the green label above the bounding boxes is the default name assigned to compounds by chem*fig*. 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 ⟨*key*⟩ `compound sep` = ⟨*dim*⟩. By default this ⟨*dim*⟩ is 5em;
- finally, the distance between the edges of the compounds and the beginning and end of the arrows is defined with the ⟨*key*⟩ `arrow offset` = ⟨*dim*⟩. By default, this ⟨*dim*⟩ is 4pt.

Here is the complete list of parameters for the `\schemestart` macro, their default values, and a brief description. These parameters must be set using the `\setchemfig{`⟨*keys*⟩`=`⟨*values*⟩`}` macro.

⟨*keys*⟩	Default ⟨*values*⟩	Description
`schemestart code`	⟨*empty*⟩	ccode executed at the very beginning of a non nested reaction scheme
`schemestop code`	⟨*empty*⟩	code executed at the very end of a non-nested reaction scheme
`scheme debug`	`false`	if ⟨*true*⟩, show nodes, names and anchors
`compound style`	⟨*empty*⟩	style of compounds
`compound sep`	`5em`	space between compounds
`name sep`	`1.5ex`	minimum vertical distance between the compound boxes and its name
`arrow offset`	`1em`	space between compound and arrow
`arrow angle`	`0`	angle of the reaction arrow
`arrow coeff`	`1`	length ratio of arrows
`arrow style`	⟨*empty*⟩	style of arrows
`arrow double sep`	`2pt`	space between double arrows
`arrow double coeff`	`0.6`	shrinkage ratio for the little arrow in "<–»" and "«–>"
`arrow double harpoon`	`true`	boolean for double harpoon arrows
`arrow label sep`	`3pt`	space between arrow and its label
`arrow head`	`-CF`	style of arrow head
`+ sep left`	`0.5em`	space before the + sign
`+ sep right`	`0.5em`	space after the + sign
`+ vshift`	`0pt`	vertical shift of the + sign

Important: for consistency, the next version of chem*fig* will require a change in the syntax of `\schemestart`. The two optional arguments will be removed for a single optional argument containing ⟨*keys*⟩=⟨*values*⟩. The settings that were possible with the two optional arguments will be accessible with new ⟨*keys*⟩.

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
\schemestart A\arrow{-/>}B \schemestop\par
\schemestart A\arrow{<-}B \schemestop\par
\schemestart A\arrow{<->}B \schemestop\par
\schemestart A\arrow{<=>}B \schemestop\par
\schemestart A\arrow{<->>}B \schemestop\par
\schemestart A\arrow{<<->}B \schemestop\par
\schemestart A\arrow{0}B \schemestop\par
\schemestart A\arrow{-U>}B \schemestop
```

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 58. Here is a "-U>" arrow with the arc on top of it: A $\overset{x}{\overset{\frown}{\longrightarrow}}{}^{y}$ B

For the sake of clarity, capital letters will be used throughout the documentation instead of chemical formulas made with the \chemfigcommand 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:

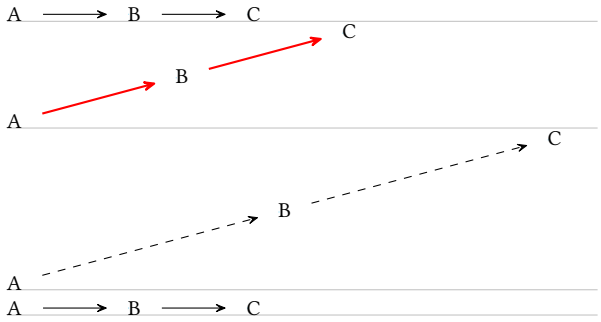
- an angle expressed in degrees;
- a coefficient that specifies the arrow length through the multiplication of the compounds spacing value defined by compound sep;
- a style with tikz instructions to customize the color, the thickness or other graphical attribute of the arrow.

These features are defined with the ⟨*keys*⟩

- arrow angle = ⟨*angle*⟩, which default value is 0;
- arrow coeff = ⟨*decimal*⟩, which default value is 1;
- arrow style = ⟨*code tikz*⟩, empty by default.

Definition of default values

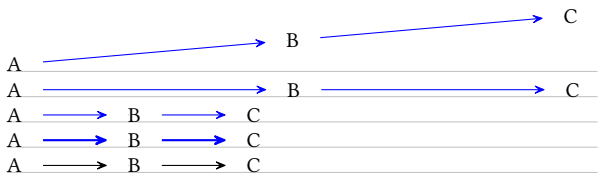
```
\schemestart A\arrow B\arrow C\schemestop

\setchemfig{arrow angle=15,arrow coeff=1.5,
arrow style={red, thick}}
\schemestart A\arrow B\arrow C\schemestop

\setchemfig{arrow coeff=2.5,arrow style=dashed}
\schemestart A\arrow B\arrow C\schemestop

\setchemfig{arrow angle={},arrow coeff={},arrow style={}}
\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

```
\setchemfig{arrow angle=5,arrow coeff=2.5,arrow style=blue}
\schemestart A\arrow B\arrow C\schemestop

\schemestart[0] A\arrow B\arrow C\schemestop

\schemestart[0,1] A\arrow B\arrow C\schemestop

\schemestart[0,1,thick] A\arrow B\arrow C\schemestop

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

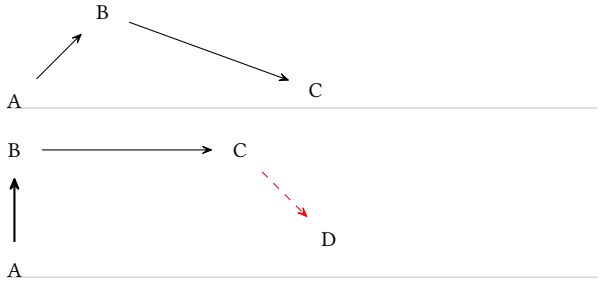


Regarding style, the rule is: the style specified in the argument in brackets applies *after* the default style, without overwriting it! This is why only the "black" color attribute is able to overwrite the " blue" default style.

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.

Arrows features

```
\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
```



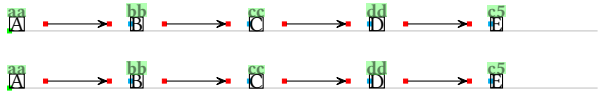
4 Compounds names

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)`.

Compounds names

```
\setchemfig{scheme debug=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:

Overfull naming

```
\setchemfig{scheme debug=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 chem*fig* 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

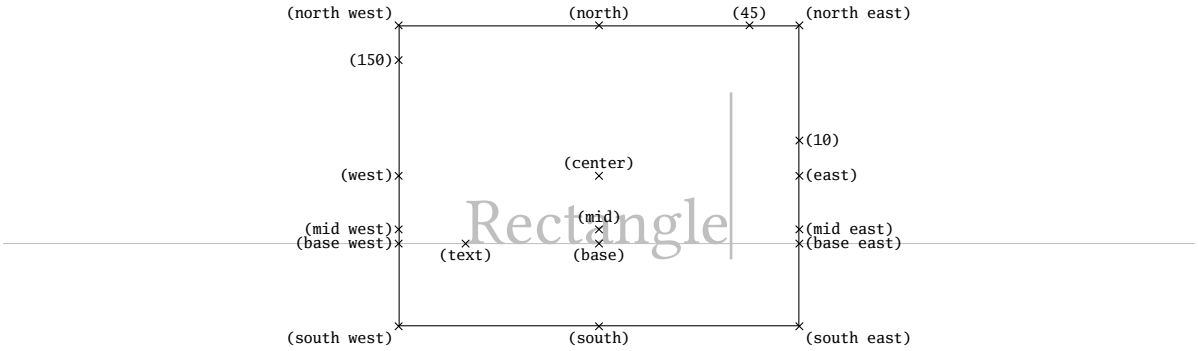
As noted above, arrows lie on the line connecting the center of the compounds’ bounding boxes. Default anchors are called “center” in the sense of ti*k*z. Non-default anchoring points can be user-specified as well with an argument between brackets:

```
(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*k*z 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}
```

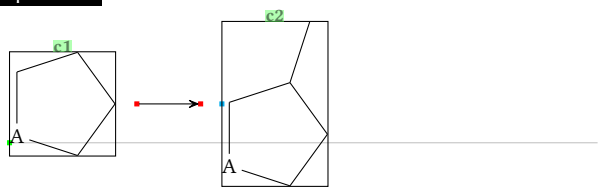


Like for names, arrival and departure anchoring points are independent and optional.

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

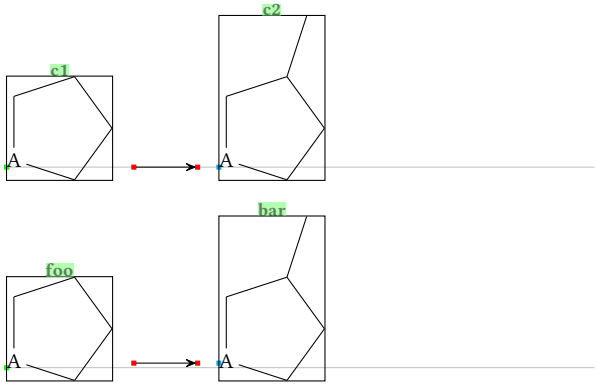
```
\setchemfig{scheme debug=true}
\schemestart
  \chemfig{A*5(-----)}
  \arrow
  \chemfig{A*5(---(-)--)}
\schemestop
```



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

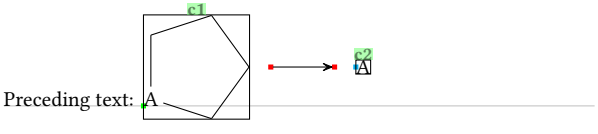
```
\setchemfig{scheme debug=true}
\schemestart
  \chemfig{A*5(-----)}
  \arrow(.base east--.base west)
  \chemfig{A*5(---(-)--)}
\schemestop
\bigskip

\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:

Initial anchoring

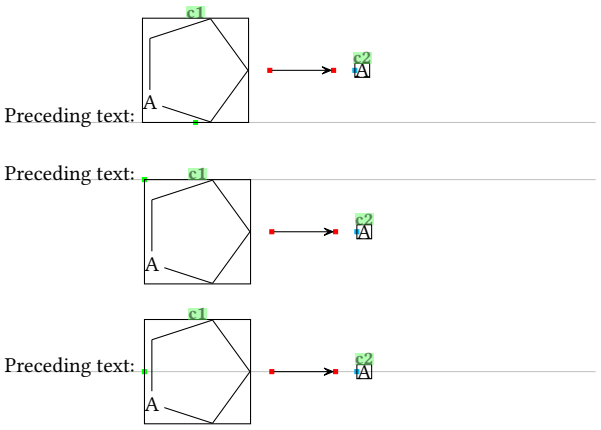
```
\setchemfig{scheme debug=true}
Preceding text:
\schemestart
  \chemfig{A*5(-----)}\arrow A
\schemestop
```



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:

Adjusting the initial anchoring

```
\setchemfig{scheme debug=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*kz* 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*kz* 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.

Compounds style

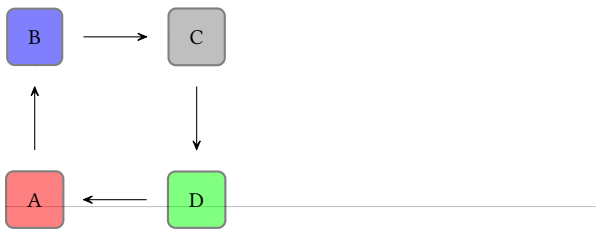
```
\schemestart
  A
  \arrow([red]--[fill=blue,semitransparent,text opacity=1,
  inner sep=10pt,rounded corners=2mm])
  B
\schemestop
\bigskip

\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:

Global styles

```
\setchemfig{compound style={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

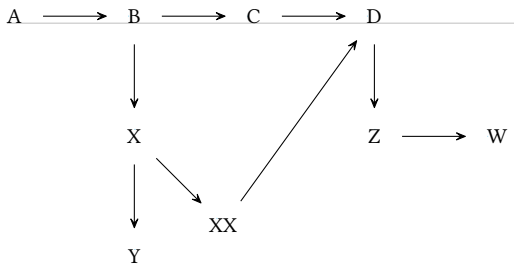
So far, only linear reaction schemes have been treated. Branched schemes are also possible and this is where compound names play a key role. When a name is preceded by "@" in the argument between brackets of the `\arrow` command, it means that the compound already exists. Several scenarios are possible:

- `(@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”:

Branching

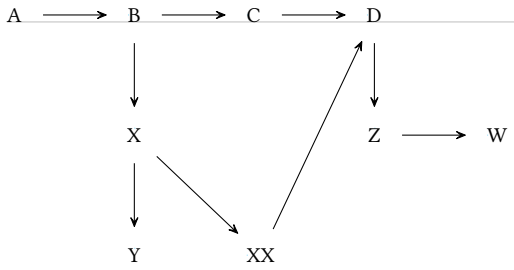
```
\schemestart
  A\arrow(aa--bb)B\arrow(--cc)C\arrow D
  \arrow(@bb--xx1)[-90]X\arrow[-90]Y% 1st branch
  \arrow(@c4--)[-90]Z\arrow W% 2nd branch
  \arrow(@xx1--xx2)[-45]XX% 3rd branch
  \arrow(@xx2--@c4)% XX-to-D arrow
\schemestop
```



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”:

Branching

```
\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
```

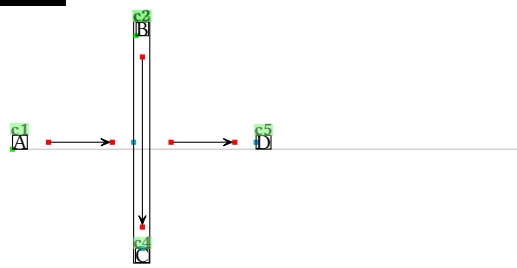


8 Subscheme

A fraction of the reaction scheme can be defined within a single bounding box, so that chem*fig* 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 regarded as a single entity. When `\subscheme` is located after an arrow, the command labels this subscheme as a compound named “c<n+1>”:

Subscheme

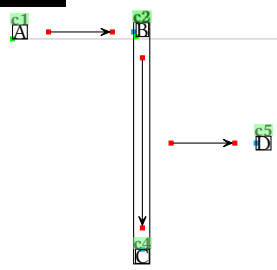
```
\setchemfig{scheme debug=true}
\schemestart
  A\arrow
  \subscheme{B\arrow[-90,2]C}
  \arrow
  D
\schemestop
```



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:

Subscheme

```
\setchemfig{scheme debug=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

Consequently, it can be used with the same optional arguments as \schemestart.

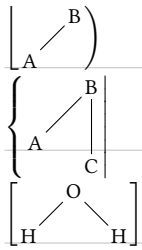
chem*fig* provides the \chemleft and \chemright command pair. These allow to set expandable delimiters on either side of a material. The commands must be followed by delimiters, just like in the case of TEX primitive commands \left and \right:

\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.

The \chemleft and \chemrightmacros

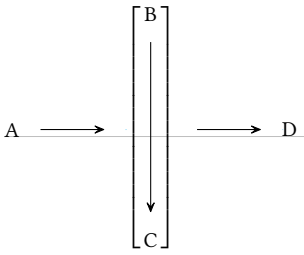
```
\chemleft\lfloor\chemfig{A-[1]B}\chemright)

\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:

Reaction scheme with \chemleft and \chemright

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



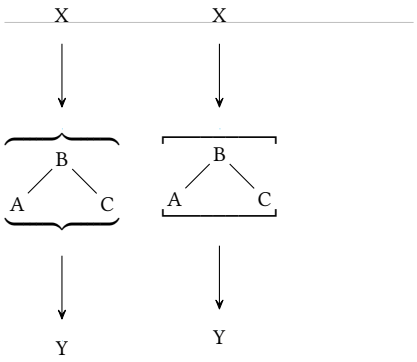
By analogy, the macros \chemup and \chemdown can be used to draw expandable delimiters above and below the material, respectively:

\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:

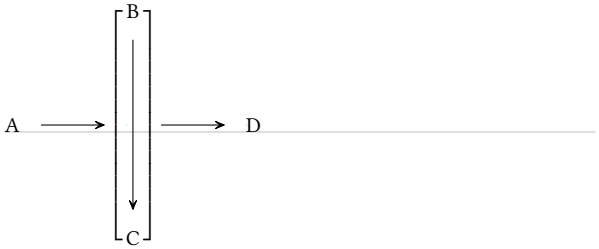
```
\usetikzlibrary{matrix}
```

Since the `\chemleft`, `\chemright`, `\chemup` and `\chemdown` commands are available, the chem*fig* 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

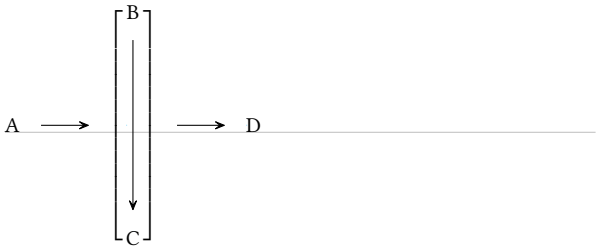
```
\schemestart
  A\arrow(--[left delimiter={[}, right delimiter={]}])
  \subscheme{B\arrow[-90,2]C}
  \arrow
  D
\schemestop
```



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

Subscheme

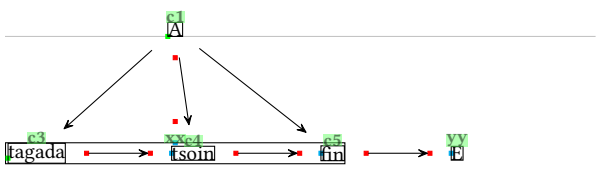
```
\schemestart
  A\arrow(--[left delimiter={[},
  right delimiter={]}])[,,shorten >=6pt]
  \subscheme{B\arrow[-90,2]C}
  \arrow[,,shorten <=6pt]
  D
\schemestop
```



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

```
\setchemfig{scheme debug=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
```

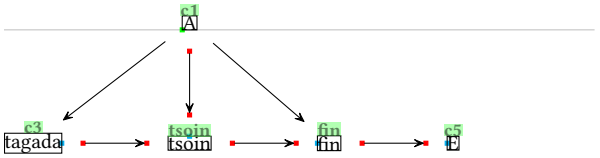


The center of the subscheme is exactly located on the same vertical line as the center of compound ”A”. This is because the two entities are connected by an invisible arrow with a −90 angle. However, the arrow between the two pre-existing compounds “A” and “tsoin” is *not* vertical because “tsoin” is 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.

Subscheme

```
\setchemfig{scheme debug=true}
\schemestart
  A
  \arrow(--tsoin){->}[-90]
  tsoin
  \arrow{<-}[180]
  tagada
  \arrow(@tsoin--fin){}
  fin
  \arrow{}
  E
  \arrow(@c1--@c3){}
  \arrow(@c1--@fin){}
\schemestop
```



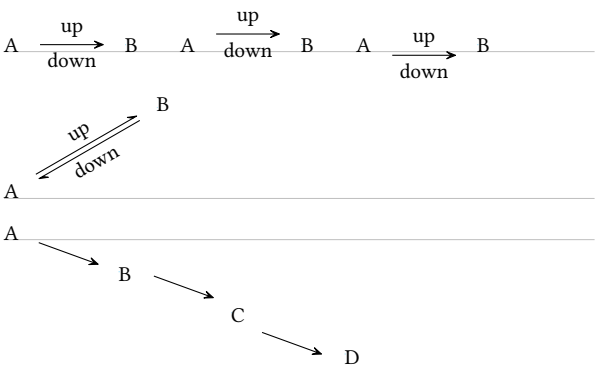
9 Arrows optional arguments

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 chem*fig*:

- 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 ⟨*key*⟩ `arrow label sep` = ⟨*dim*⟩ which value is 3pt by default. 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;

Arrows optional arguments

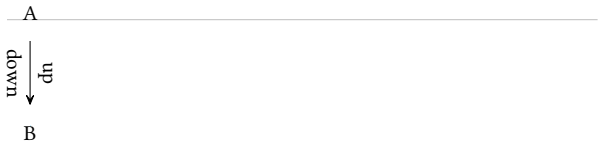
```
\setchemfig{scheme debug=false}
\schemestart A\arrow{->[up][down]}B \schemestop
\qquad
\schemestart A\arrow{->[up][down][4pt]}B \schemestop
\qquad
\schemestart A\arrow{->[up][down][-4pt]}B \schemestop
\medskip

\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

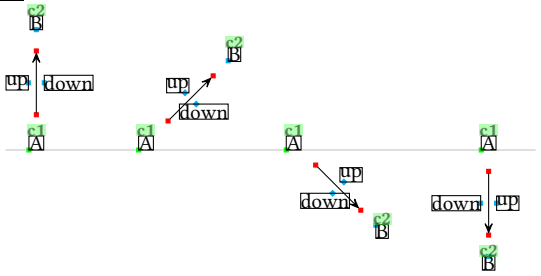
```
\schemestart
  A\arrow{->[up][down]}[-90]B
\schemestop
```



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

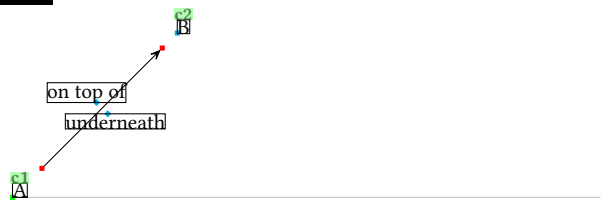
```
\setchemfig{scheme debug=true}
\schemestart A\arrow{->[*{0}up][*{0}down]}[90]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}up][*{0}down]}[45]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}up][*{0}down]}[-45]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}up][*{0}down]}[-90]B\schemestop
```



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

Anchors

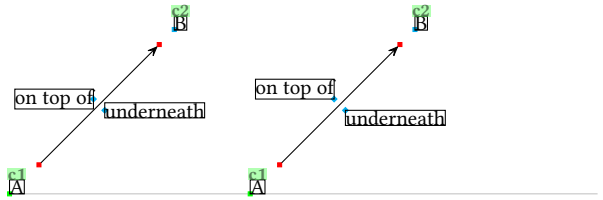
```
\setchemfig{scheme debug=true}
\schemestart
  A\arrow{->[*{0}on top of][*{0}underneath]}[45,2]B
\schemestop
```



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

Anchors

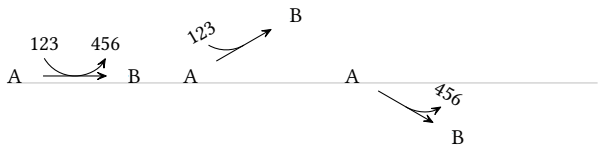
```
\setchemfig{scheme debug=true}
\schemestart
  A\arrow{->[*{0.0}on top of][*{0.180}underneath]}[45,2]B
\schemestop
\qquad
\schemestart
  A\arrow{->[*{0.south east}on top of]%
    [*{0.north west}underneath]}[45,2]B
\schemestop
```



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

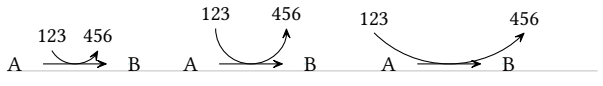
```
\schemestart A\arrow{-U>[123][456]}B\schemestop
\qquad
\schemestart A\arrow{-U>[123]}[30]B\schemestop
\qquad
\schemestart A\arrow{-U>[][456]}[-30]B\schemestop
```



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:

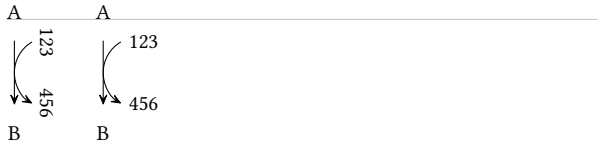
The –U> arrow

```
\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:

The –U> arrow

```
\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:

`\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 between `\catcode‘\_=11` and `\catcode‘\_=8` commands.

- `\CF_arrowstartname` and `\CF_arrowendname` include the names of the compounds (considered as nodes by ti*kz*) between which the arrow is drawn;
- `\CF_arrowstartnode` and `\CF_arrowendnode` 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 `\arrow`command, unless the field is left empty;
- `\CF_arrowcurrentstyle` and `\CF_arrowcurrentangle` contain the style and the angle of the arrow to be drawn;
- `\CF_arrowshiftnodes{`⟨*dim*⟩`}` shifts the nodes "`\CF_arrowstartnode`" and "`\CF_arrowendnode`" perpendicularly relative to the arrow by a dimension specified in the argument;
- `\CF_arrowdisplaylabel{#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 heads are based on "`CF`" for a full arrow and have the "harpoon" option for half arrows.

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

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_arrowshiftnodes`, so the code of the arrow will start with: `\CF_arrowshiftnodes{#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*k*z code is:

```
\edef\pt_radius{\ifx\empty#4\empty 2pt\else #4\fi}% circle radius
\expandafter\draw\expandafter[\CF_arrowcurrentstyle,-CF]
    (\CF_arrowstartnode)--(\CF_arrowendnode)coordinate[midway](mid@point);
\filldraw(mid@point)circle(\pt_radius);%
```

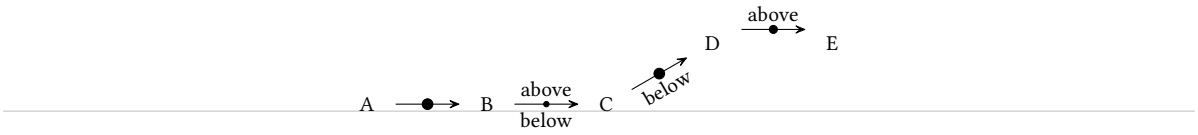
The last step is to enter the labels, if any, with the folwing line:

```
\CF_arrowdisplaylabel{#1}{0.5}{+}{\CF_arrowstartnode}{#2}{0.5}{-}{\CF_arrowendnode}
```

Here is the completed arrow:

Arrow "-.>"

```
\catcode`\_11
\definearrow4{-.>}{%
  \edef\pt_radius{\ifx\empty#4\empty 2pt\else #4\fi}% dot radius
  \CF_arrowshiftnodes{#3}%
  \expandafter\draw\expandafter[\CF_arrowcurrentstyle,-CF](\CF_arrowstartnode)--(\CF_arrowendnode)
    coordinate[midway](mid@point);
  \filldraw(mid@point)circle(\pt_radius);%
  \CF_arrowdisplaylabel{#1}{0.5}{+}{\CF_arrowstartnode}{#2}{0.5}{-}{\CF_arrowendnode}
  }
\catcode`\_8
\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*k*z code that will specify the point(s) of control. If this argument is empty, a "`-CF`" type arrow will be plotted.

If `#1` is not empty, attention should not be drawn to "`\CF_arrowstartnode`" and "`\CF_arrowendnode`" 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_arrowstartname` and `\CF_arrowendname` which contain the names of the compound (which are nodes for ti*k*z), since the arrow must be plotted between them. Here's the ti*k*z code to draw the curved arrow between the two compounds:

```
\draw[shorten <=\CF_arrowoffset,shorten >=\CF_arrowoffset,\CF_arrowcurrentstyle,-CF,
(\CF_arrowstartname).. controls #1 ..(\CF_arrowendname);%
```

One must add a ti*k*z code to shorten the arrow by an amount `\CF_arrowoffset` defined by `\setarrowoffset`. Indeed, the nodes ar not the same as those for straight arrows (`\CF_arrowstartnode` and `\CF_arrowendnode`). So before `\CF_arrowcurrentstyle`, the following code must be added:

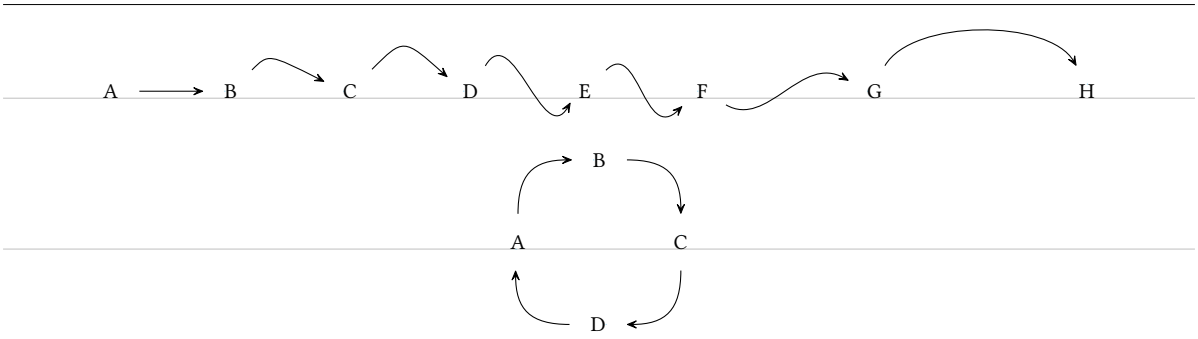
```
shorten <=\CF_arrowoffset, shorten >=\CF_arrowoffset
```

this is the role the two lines after `\else`.

So here is our curved arrow:

Curved arrow

```
\catcode`\_11
\definearrow1{s>}{%
\ifx\empty#1\empty
  \expandafter\draw\expandafter[\CF_arrowcurrentstyle,-CF](\CF_arrowstartnode)--(\CF_arrowendnode);%
\else
  \def\curvedarrow_style{shorten <=\CF_arrowoffset,shorten >=\CF_arrowoffset,}%
  \CF_eaddtomacro\curvedarrow_style\CF_arrowcurrentstyle
  \expandafter\draw\expandafter[\curvedarrow_style,-CF](\CF_arrowstartname)..controls#1..(\CF_arrowendname);
\fi
}
\catcode`\_8
\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”.

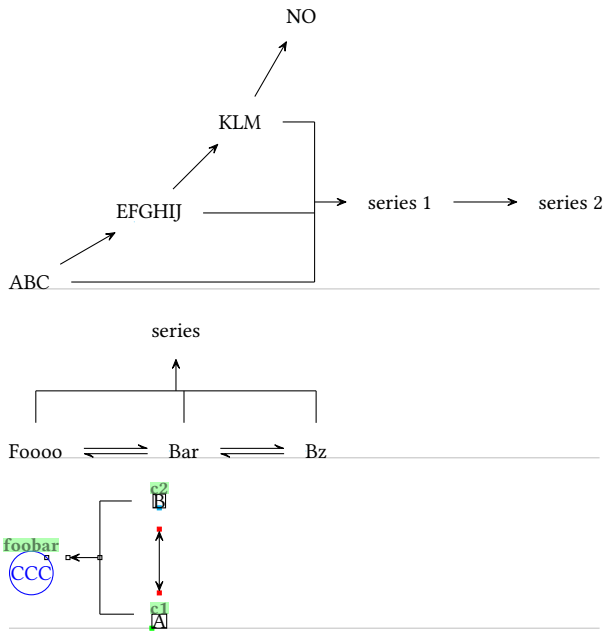
The syntax follows with:

```
\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 out coming 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 \mergecommand

```
\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

\setchemfig{scheme debug=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:

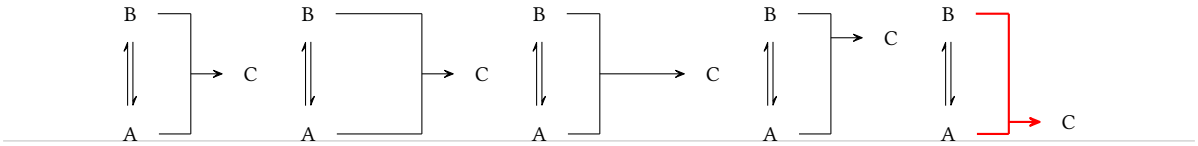
```
\merge{dir}(n1.a1)(n2.a2)(...)(ni.ai)--(n.a[s])[c1,c2,c,style]
```

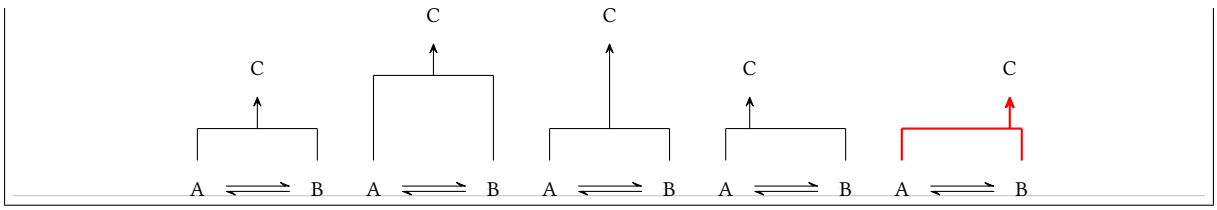
where:

- 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
```





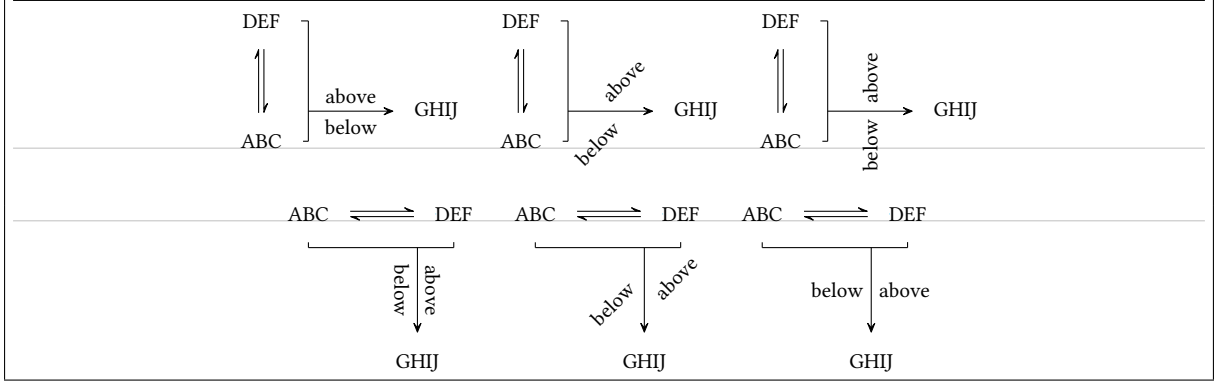
Finally, it is possible to write labels above or below the merged arrow. For this, the direction character accepts two optional arguments in brackets, a first one for the label above the arrow and a second one for the label below it. Therefore, the full syntax of the `merge` command is:

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

All the features introduced before for arrow labeling 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 \mergecommand

```
\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

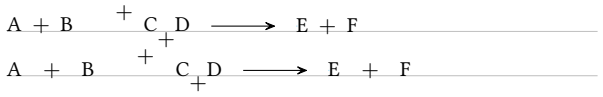
The use of a "\+" macro that displays a + sign is available between the commands `\schemestart` and `\schemestop`. This macro accepts an optional argument in braces with 3 dimensions in the form {⟨*dim1*⟩,⟨*dim2*⟩,⟨*dim3*⟩}, where:

- ⟨*dim1*⟩ and ⟨*dim2*⟩ are the dimensions to be inserted before and after the + sign;
- ⟨*dim3*⟩ is the vertical offset of the sign.

These dimensions can also be set, for all the following + signs with the ⟨*keys*⟩ `+ sep left` = ⟨*dim*⟩, `+ sep right` = ⟨*dim*⟩ et `+ vshift` = ⟨*dim*⟩. The default values are 0.5em for the two first and 0pt for the third.

The \+ command

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

\setchemfig{+ sep left=1em,+ sep right=1em,+ vshift=0pt}
\schemestart
A\+B\+{2em,,5pt}C\+{0pt,0pt,-5pt}D\arrow E\+F
\schemestop
```



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

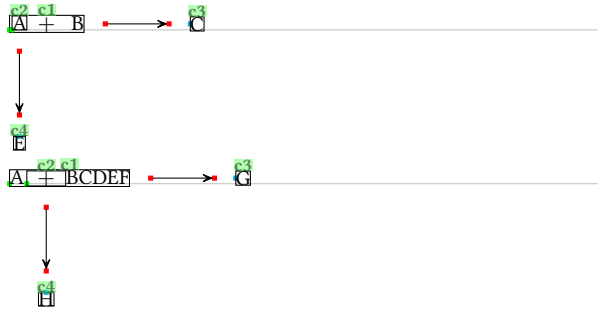
Compounds and \+

```
\setchemfig{scheme debug=true}
\schemestart A\+ B\+{,,5pt}C\arrow D\+ E\schemestop
```

This makes it difficult to draw a vertical arrow exactly below the letter “A” since this letter is not a single compound for chem*fig*. 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 \+

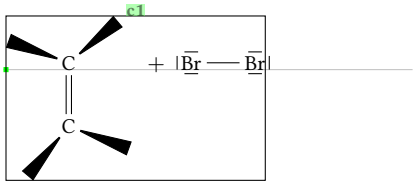
```
\setchemfig{scheme debug=true}
\schemestart
\subscheme{A}\+ B\arrow C
\arrow(@c2--)[-90]E
\schemestop
\medskip

\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

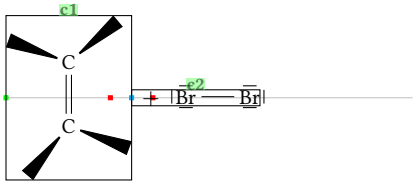
```
\setchemfig{scheme debug=true}
\schemestart
  \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
  \+
  \chemfig{\charge{90=\|,180=\|,270=\|}{Br}-\charge{0=\|,90=\|,-90=\|}{Br}}
\schemestop
```



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 “|Br —— Br|”. In fact, the “+” sign does not prevent chem*fig* 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

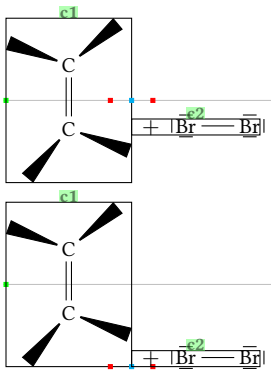
```
\setchemfig{scheme debug=true}
\schemestart[][west]
  \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
  \arrow{0}[,0]\+
  \chemfig{\charge{90=\|,180=\|,270=\|}{Br}-\charge{0=\|,90=\|,-90=\|}{Br}}
\schemestop
```



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

```
\setchemfig{scheme debug=true}
\schemestart[][west]
  \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
  \arrow(--[yshift=-10pt]){0}[,0]\+
  \chemfig{\charge{90=\|,180=\|,270=\|}{Br}-\charge{0=\|,90=\|,-90=\|}{Br}}
\schemestop
\medskip

\schemestart[][west]
  \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
  \arrow(.south east--.south west){0}[,0]\+
  \chemfig{\charge{90=\|,180=\|,270=\|}{Br}-\charge{0=\|,90=\|,-90=\|}{Br}}
\schemestop
```



Horizontal reactions

To typeset a horizontal reaction with a simpler syntax, you can use the environment “hreac”.

\hreac[⟨*keys*⟩=⟨*values*⟩]
⟨*reaction*⟩
\endhreac

As seen previously, the options can be set with \setchemfig{⟨*keys*⟩=⟨*values*⟩} if you want to set them for the rest of the document.

To apply settings to a single reaction only, use the optional argument of the environment.

Here is the complete list of parameters for the \hreac macro, their default values, and a short description.

Key	Default value	Description
harrow minwidth	3em	minimal length of an arrow
label xsep	3pt	when the arrow extends to fit its labels, its length is equal to the length of the longest label plus twice that dimension
label align	c	text alignment in arrow labels (possible values ⟨*c*⟩, ⟨*r*⟩, or ⟨*l*⟩)
name sep	1.5ex	minimum vertical distance between the compound boxes and its name
hreac anchor	text	anchor of the first compound
hreac sep	0.5em	horizontal space between boxes containing compounds
hreac debug	false	if ⟨*true*⟩, draw the boundaries of the compounds, the anchors, the node names, and the horizon line
arrow label sep	3pt	for this key and the following ones, see options of \schemestart page 50
arrow style	⟨*empty*⟩	
arrow double sep	2pt	
arrow double coeff	0.6	
arrow double harpoon	true	
arrow label sep	3pt	
arrow head	-CF	

1 Syntax

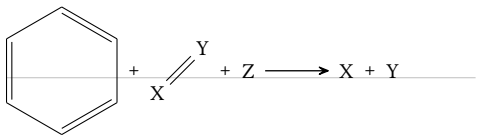
The tokens ⊞,⊵, ⧹> ⧹^ have meaning for the syntax of ⟨*reaction*⟩.

In ⟨*reaction*⟩, compounds, + signs, and arrows may exist according to the following syntax and rules:

- a ⟨*compound*⟩ extends to the next occurrence of "+", ">" or the end of the environment;
- in ⟨*reaction*⟩, any code belonging to a ⟨*compound*⟩ between braces is not examined and is left as it is between braces;
- spaces are ignored (except when they are between braces, see previous point);
- a ⟨*compound*⟩ consists of any code except the reserved tokens seen above, unless they are between braces;
- the sign "+" displays the sign + and is a ⟨*compound*⟩ on its own;
- the sign ">" displays an arrow and is a ⟨*compound*⟩ on its own;

Example 1

```
\hreac
  \chemfig{*6(-=-=-=)}
  +
  \chemfig{X=[1]Y}
  +
  Z > X + Y
\endhreac
```

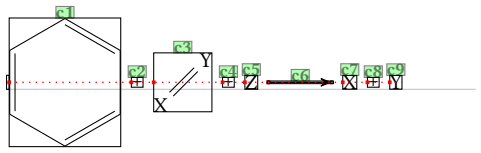


2 Placement of compounds

All compounds are numbered c⟨*n*⟩ (where ⟨*n*⟩ is an integer) and are placed in boxes. The centers of all boxes are placed on the same horizontal line, which is drawn in red dots when the `hreac debug` option is enabled:

Vertical alignement vertical of compounds

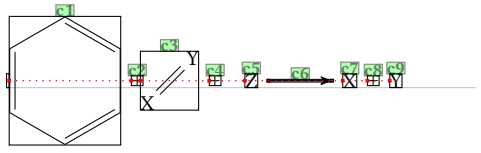
```
\hreac[hreac debug]
  \chemfig{*6(-=-=-=)}
  +
  \chemfig{X=[1]Y}
  +
  Z > X + Y
\endhreac
```



When \>{⟨*dimension*⟩} is encountered at the beginning of ⟨*compound*⟩, the latter is moved horizontally by {⟨*dimension*⟩}.

Horizontal adjustment of compounds

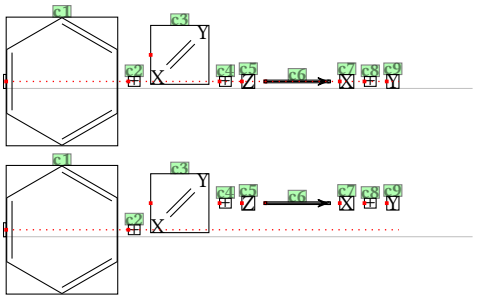
```
\hreac[hreac debug]
  \chemfig{*6(-=-=-=)}
  +
  \>{-5pt}\chemfig{X=[1]Y}
  +
  \>{5pt}Z > X + Y
\endhreac
```



When \^{⟨*dimension*⟩} is encountered at the beginning of ⟨*compound*⟩, the latter (and only the latter) is moved vertically by {⟨*dimension*⟩}. In the starred version \^*{⟨*dimension*⟩}, the compound and all subsequent ones are affected.

Vertical adjustment of compounds

```
\hreac[hreac debug]
  \chemfig{*6(-=-=-=)}
  +
  \^{10pt}\chemfig{X=[1]Y}
  +
  Z > X + Y
\endhreac\medbreak
\hreac[hreac debug]
  \chemfig{*6(-=-=-=)}
  +
  \^*{10pt}\chemfig{X=[1]Y}
  +
  Z > X + Y
\endhreac
```



3 Compound names

To display the ⟨*name*⟩ of a ⟨*compound*⟩, either of the following syntaxes can be used:

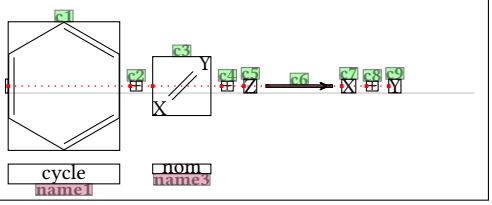
- `\chemname[`⟨*dimension*⟩`]{`⟨*compound*⟩`}{`⟨*name*⟩`}`;
- ⟨*compound*⟩`\name[`⟨*dimension*⟩`]{`⟨*name*⟩`}` or `\name[`⟨*dimension*⟩`]{`⟨*name*⟩`}`⟨*compound*⟩, the order of appearance being irrelevant

The optional ⟨*dimension*⟩ is the minimum distance between the ⟨*compound*⟩ box and the ⟨*name*⟩ box. When omitted, its default value is the value of the `name sep` key.

All names are numbered `name`⟨*n*⟩ where ⟨*n*⟩ is the integer corresponding to ⟨*compound*⟩. They are placed in boxes whose width is that of ⟨*compound*⟩. The upper borders of these boxes on the same horizontal line are positioned vertically so as to be below the compound with the greatest depth.

Names of compounds

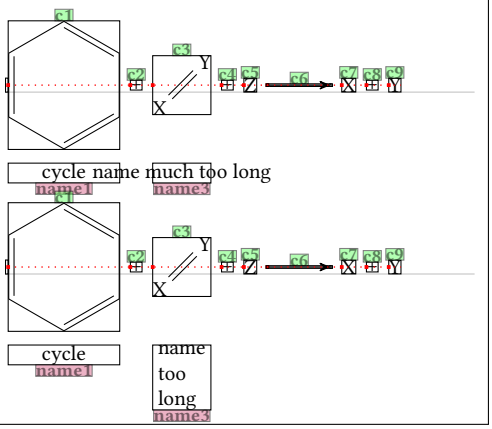
```
\hreac[hreac debug]
  \chemfig{*6(-=-=-=)}\name{cycle}
  +
  \chemfig{X=[1]Y}\name{nom}
  +
  Z > X + Y
\endhreac
```



If the width of a ⟨*name*⟩ exceeds that of its ⟨*compound*⟩, no adjustment is made to the width of the ⟨*compound*⟩ and the ⟨*name*⟩ will extend beyond the boundaries of the box containing it. It is up to the user to use the `\\` macro to display the ⟨*name*⟩ on multiple lines.

Name of compounds

```
\hreac[hreac debug]
  \chemfig{*6(-=-=-=)}\name{cycle}
  +
  \chemfig{X=[1]Y}\name{name much too long}
  +
  Z > X + Y
\endhreac\medbreak
\hreac[hreac debug]
  \chemfig{*6(-=-=-=)}\name{cycle} +
  \chemfig{X=[1]Y}\name{name\\too\\long} +
  Z > X + Y
\endhreac
```

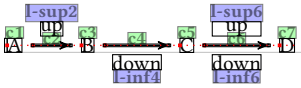


4 Arrow labels

If the token “>” is immediately followed by an optional argument `[`⟨*label*⟩`]`, this ⟨*label*⟩ will be placed above the arrow. If another optional argument `[`⟨*label*⟩`]` is present, the other ⟨*label*⟩ will be placed below the arrow.

Arrow labels

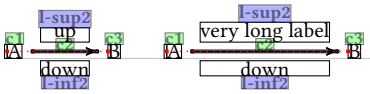
```
\hreac[hreac debug]
  A >[up] B >[][down] C >[up][down] D
\endhreac
```



The default width of an arrow is 3em, but if a label exceeds this dimension, the arrow lengthens.

Arrow lengthening

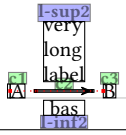
```
\hreac[hreac debug]
  A >[up][down] B
\endhreac
\qquad
\hreac[hreac debug]
  A >[very long label][down] B
\endhreac
```



As with compound names, it is possible to use \\ to compose a ⟨*label*⟩ over several lines.

Label over several lines

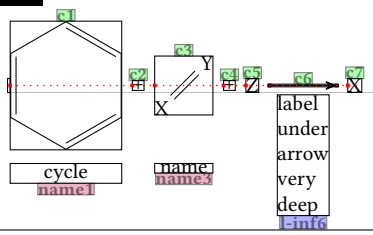
```
\hreac[hreac debug]
  A >[very\\long\\label][bas] B
\endhreac
```



The box containing the bottom label has no influence on the vertical position of the compound names.

Independence of labels and names

```
\hreac[hreac debug]
  \chemfig{*6(-=-=-=)}\name{cycle}
  +
  \chemfig{X=[1]Y}\name{name}
  +
  Z >[][label\\under\\arrow\\very\\deep] X
\endhreac
```



5 Arrow types

If the token > is immediately followed by text between curly braces, this optional text specifies the arrow type. Its default value is the value of the key `arrow head`, which contains ⟨*-CF*⟩ by default.

The complete syntax of the > token is therefore:

>{⟨*arrow type*⟩}[⟨*top label*⟩][⟨*bottom label*⟩]

By using this optional parameter between curly brackets, you can access these other types of arrows, already described in the section “Reaction diagrams” (page 50) : “->”, “<-”, “<->”, “<=>”, ”<< ->,“ and ”<->>.”

The 6 types of arrows

```
\hreac A > B \endhreac\par% identique à >{->}
\hreac A >{<-} B \endhreac\par
\hreac A >{<->} B \endhreac\par
\hreac A >{<=>} B \endhreac\par
\hreac A >{<<->} B \endhreac\par
\hreac A >{<->>} B \endhreac
```

List of commands

The commands created by chem*fig* are:

Commands	Description
\chemfig[⟨*settings*⟩]{⟨*code*⟩}	draws the molecule whose design is described by the ⟨*code*⟩
\setchemfig{⟨*settings*⟩}	sets the parameters with the syntax ⟨*key*⟩ = ⟨*value*⟩. These ⟨*key*⟩ and ⟨*value*⟩ are listed on page 5 for \chemfig, page 50 for the \schemestart macro, and page 66 for the \hreac macro.
\resetchemfig	Reset the parameters to their default values
\printatom	displays the atoms within the molecules. It can be redefined to customize the output. See page 26
\hflipnext	the next molecule will be horizontally flipped
\vflipnext	the next molecule will be vertically flipped
\definesubmol{⟨*name*⟩}⟨*n*⟩[⟨*code1*⟩]{⟨*code2*⟩}	creates an alias !⟨*name*⟩ 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 28
\chemskipalign	tells the vertical alignment mechanism to ignore the current group of atoms. See page 32.
\redefinesubmol{⟨*name*⟩}⟨*n*⟩[⟨*code1*⟩]{⟨*code2*⟩}	replaces a preexisting alias !⟨*name*⟩ with the new ⟨*code*⟩. See page 28
\charge{[⟨*parameters*⟩]⟨*pos*⟩[⟨*tikz*⟩]}{⟨*atom*⟩}	prints ⟨*atome*⟩ and places the charges according to their ⟨*positions*⟩. The charges are places out of the bounding box of the ⟨*atome*⟩. See page 32
\Charge{[⟨*parameters*⟩]⟨*pos*⟩[⟨*tikz*⟩]}{⟨*atom*⟩}	Same behaviour as \charge, but the final bounding box takes the charges into account.
\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 21.
\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 35
\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 35
\Chemabove[⟨*dim*⟩]{⟨*txt1*⟩}{txt2}	writes ⟨*txt1*⟩ and places ⟨*txt2*⟩ above, leaving ⟨*dim*⟩ of vertical space. See page 35
\Chembelow[⟨*dim*⟩]{⟨*txt1*⟩}{txt2}	writes {txt1} and places ⟨*txt2*⟩ below, leaving ⟨*dim*⟩ of vertical space. See page 35
\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 49.
\arrow	draws an arrow in a reaction scheme (this command is only defined inside a reaction scheme). See page 50.
\+	prints a + sign in a reaction scheme (this command is only defined inside a reaction scheme). See page 64.
\subscheme{⟨*code*⟩}	draws a subscheme (this command is only defined inside a reaction scheme). See 55.
\definearrow	defines an arrow. See page 60.
\chemleft⟨*car1*⟩⟨*stuff*⟩\chemright⟨*car1*⟩	draws expandable delimiters defined with ⟨*car1*⟩ and ⟨*car2*⟩ on the left and on the right of the ⟨*stuff*⟩, see page 56.
\chemup⟨*car1*⟩⟨*stuff*⟩\chemdown⟨*car1*⟩	draws expandable delimiters defined with ⟨*car1*⟩ and ⟨*car2*⟩ above and below the ⟨*stuff*⟩, see page 56.
\polymerdelim[⟨*settings*⟩{⟨*node1*⟩}{*node2*⟩}	draws delimiters at specified nodes, see page 45
\hreac...\endhreac	tags between which a horizontal reaction is drawn, see page 66
>{⟨*type*⟩}[⟨*top label*⟩][⟨*bottom label*⟩]	draws a horizontal arrow, see page 67
+	displays the + sign in the horizontal reaction
\>{⟨*dimension*⟩}	moves the current compound horizontally, see page 69
\^{⟨*dimension*⟩} or \^*{⟨*dimension*⟩}	moves the current compound vertically, see page 67
\chemname{⟨*compound*⟩}{⟨*name*⟩}	typeset ⟨*name*⟩ under ⟨*compound*⟩, see page 68
\name{⟨*name*⟩}	typeset ⟨*name*⟩ under the current compound, see page 68

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.

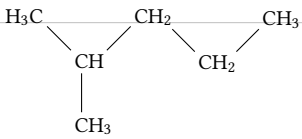
Anyway, I hope that this package will help all LaTeX users wishing to draw molecules. Although chem*fig* has been thoroughly tested and although its version number is now greater than 1.0, I hope that you will be forgiving with bugs you encounter and send me an **email** to let me know of any malfunctions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

Christian Tellechea

⋆
⋆ ⋆

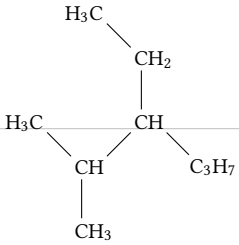
2-methylpentane

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



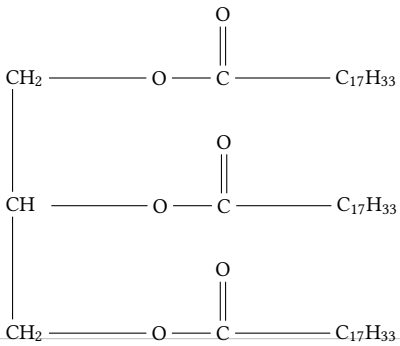
3-ethyl-2-methylhexane

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



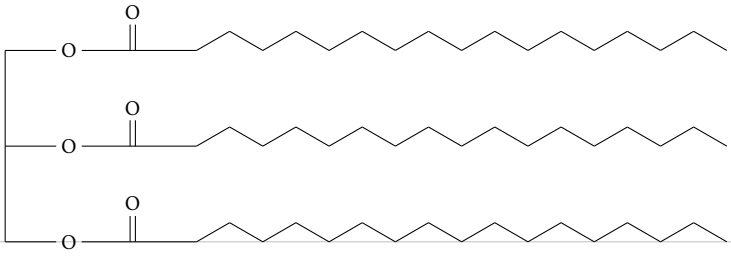
Stearine, condensed structural diagram

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



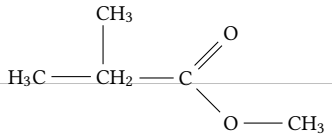
Stearine, skeleton diagram

```
\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)}
```



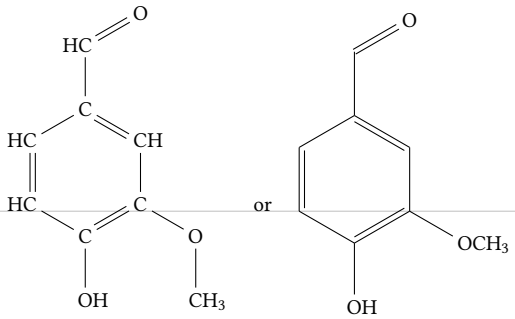
Methyl 2-methylpropanoate

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



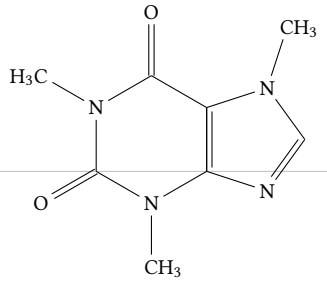
Vanillin

```
\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)-=)}
```



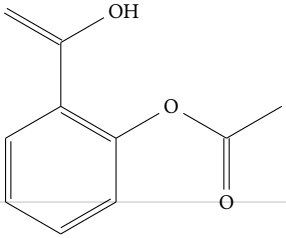
Caffeine

```
\chemfig{*6((=O)-N(-CH_3)-*5(-N=-N(-CH_3)-=)--(=O)-N(-H_3C)-)}
```



Aspirin

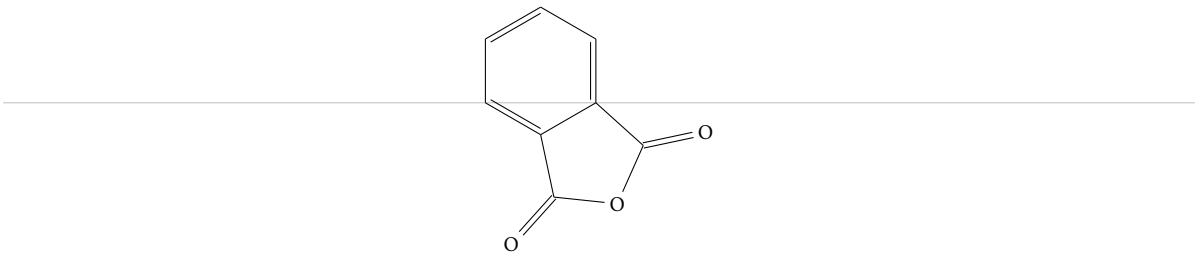
```
\chemfig{*6(-=-(-O-[::-60](=[::-60]O)-[::+60])=(-(=[::+60])-[::-60]OH)-=)}
```



Aspirin is a registered trademark of Bayer in many countries.

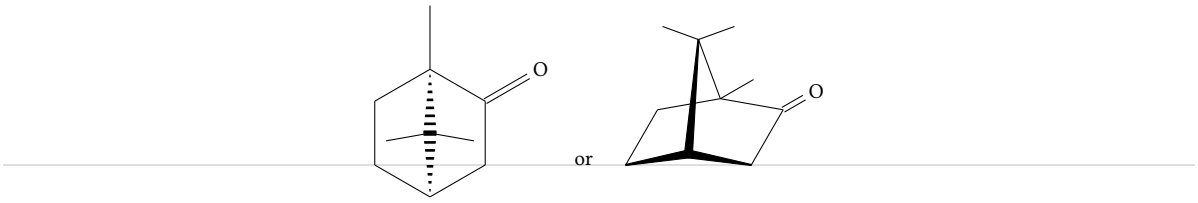
Phthalic anhydride

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



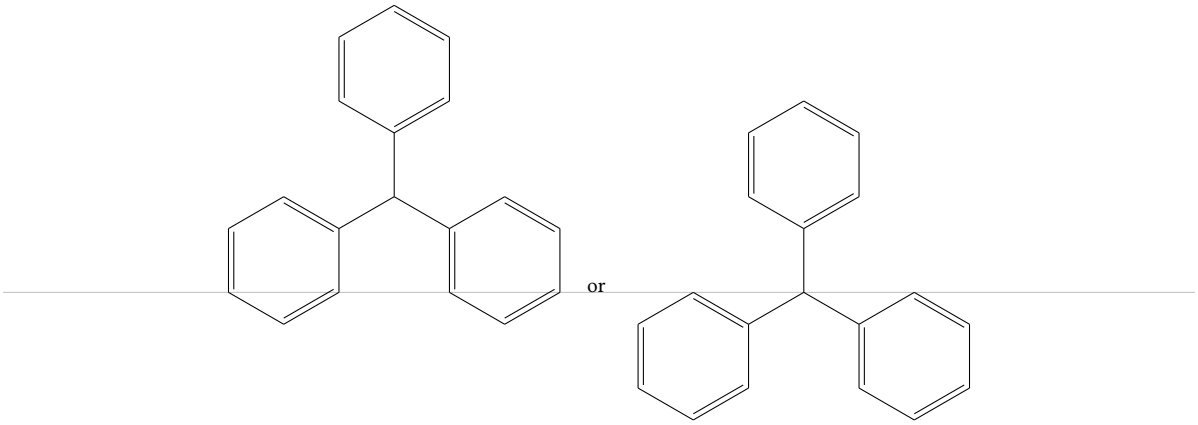
Camphor

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



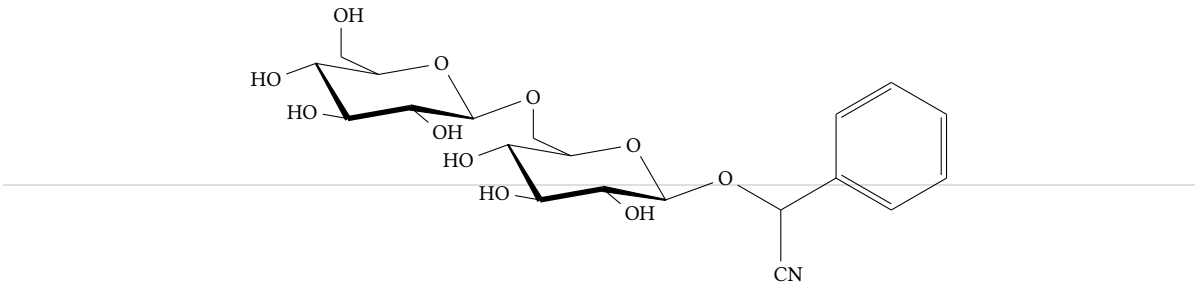
Triphenylmethane

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



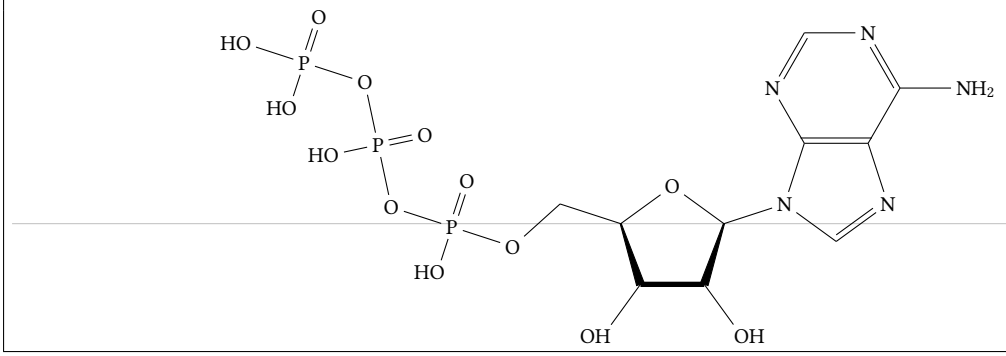
Amygdalin

```
\setchemfig{cram width=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(=-=-=-)}
```



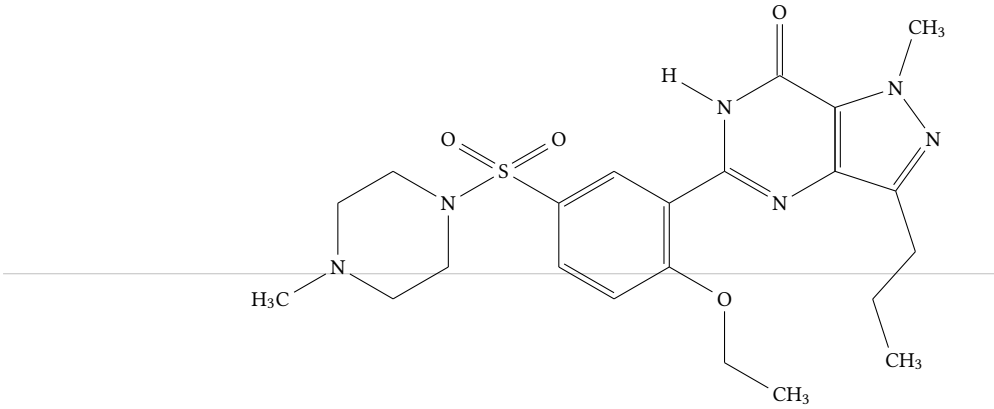
Adenosine triphosphate

```
\setchemfig{cram width=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-)}
```



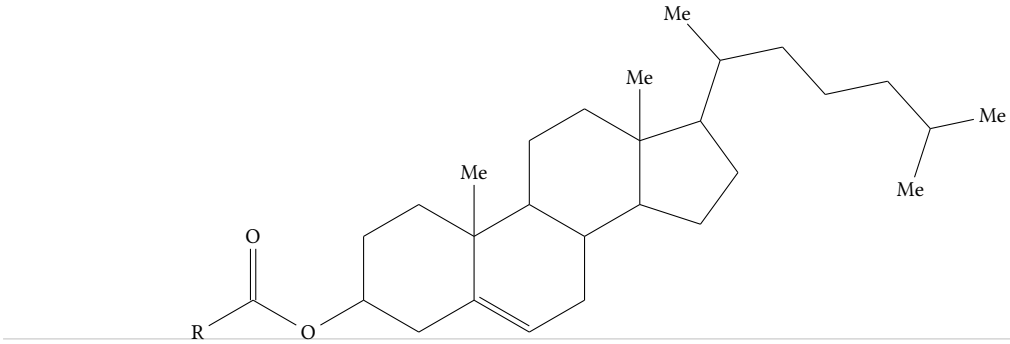
Viagra

```
\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)-))-=))---)}
```



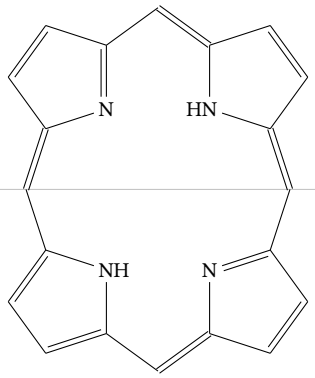
Cholesterol ester

```
\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)---)}
```



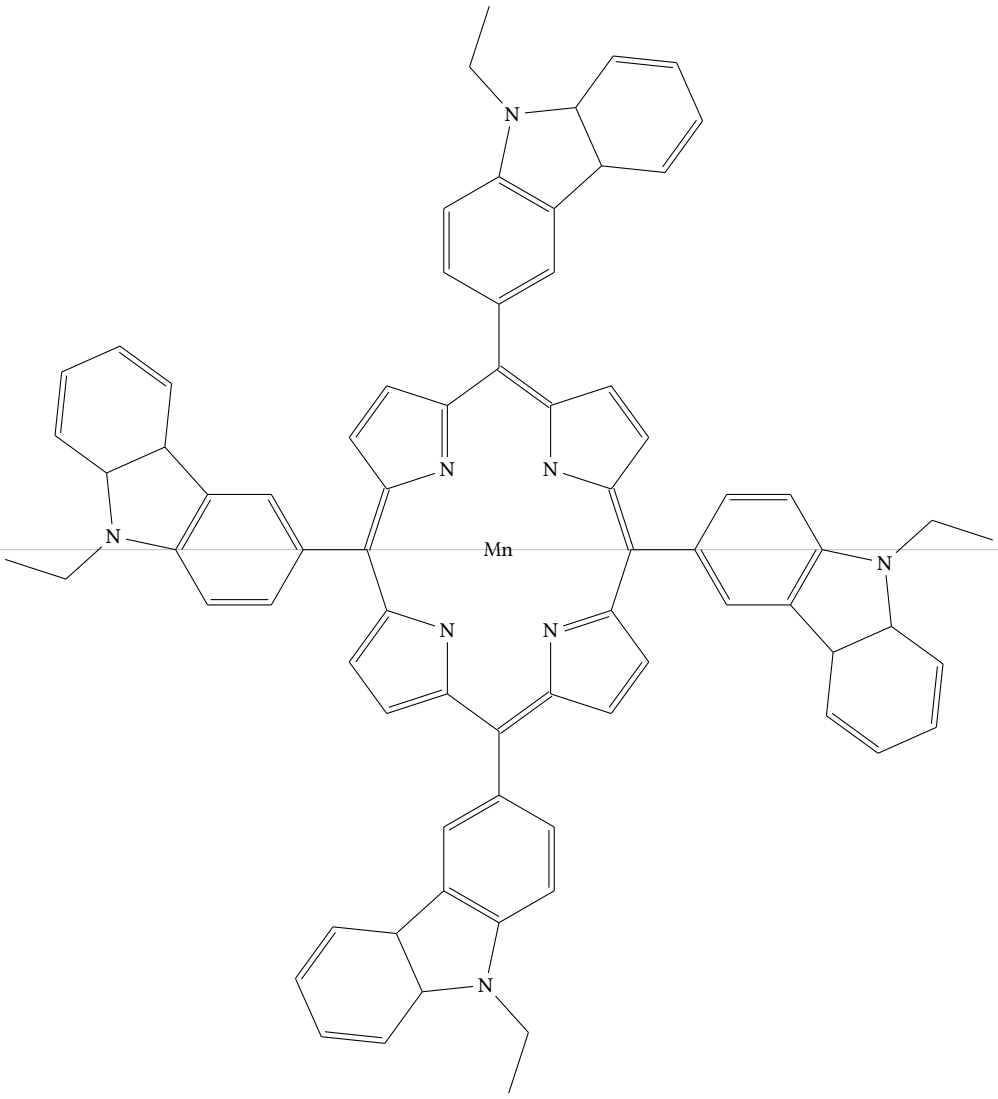
Porphyrin

```
\chemfig{?=[::+72]*5(-N=(-=[::-72]*5(-[,,,2]HN-[,,2](=-[::-36]*5(=N-(=-[::-72]*5(-NH-[,,1]?=-=))
-=-))-=-))-=-)}
```



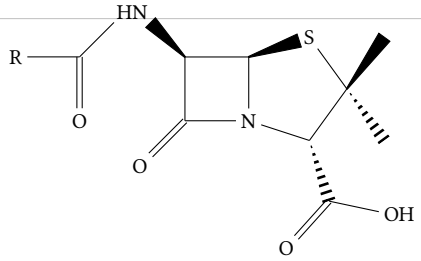
Manganese 5,10,15,20-tetra(N-ethyl-3-carbazolyl) porphyrin

```
\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-(-)=-=))-=-))-=-))-=-)}
```



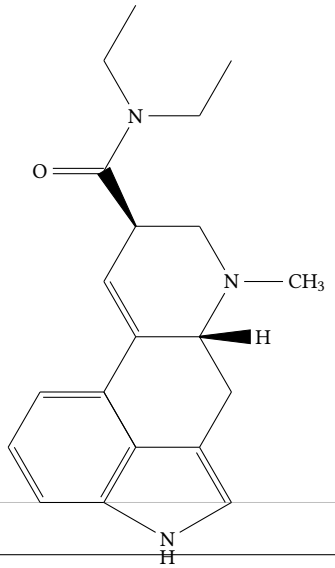
Penicillin

```
\chemfig{[:-90]HN(-[::-45](-[::-45]R)=[::+45]O)>[::+45]*4(-(=O)-N*5(-(<:(=[::-60]O)
-[::+60]OH)-(<[::+0])(<:[::-108])-S>)--)}
```



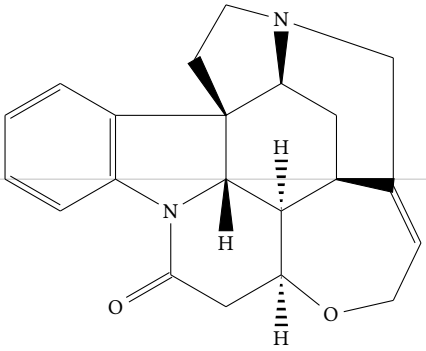
LSD

```
\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}?)=))}
```



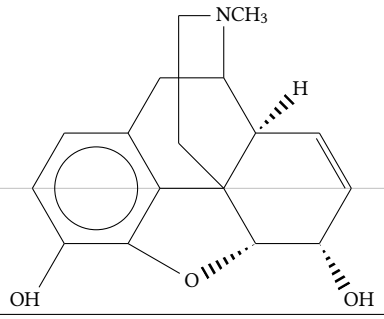
Strychnine

```
\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))--?)=?-=-)}
```



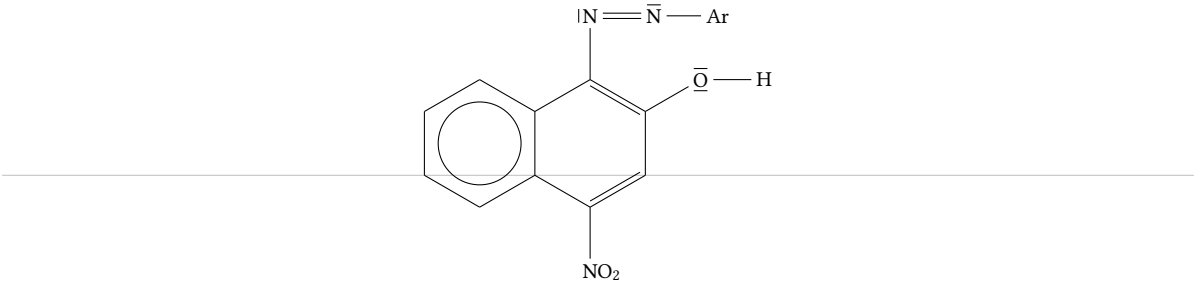
Codeine

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



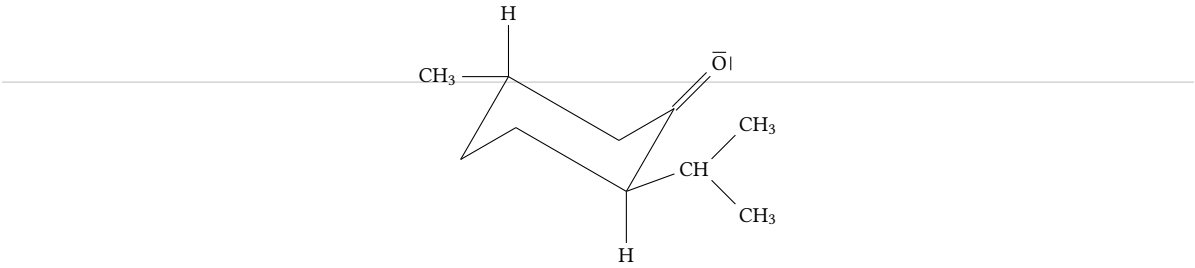
A dye (red)

```
\chemfig{**6(--*6(-(-NO_2)=-(-\charge{90=\|,-90=\|}{O}-[0]H)=(-\charge{180=\|}{N}=[0]\charge{90=\|}{N}-[0]Ar)-)----
)}
```



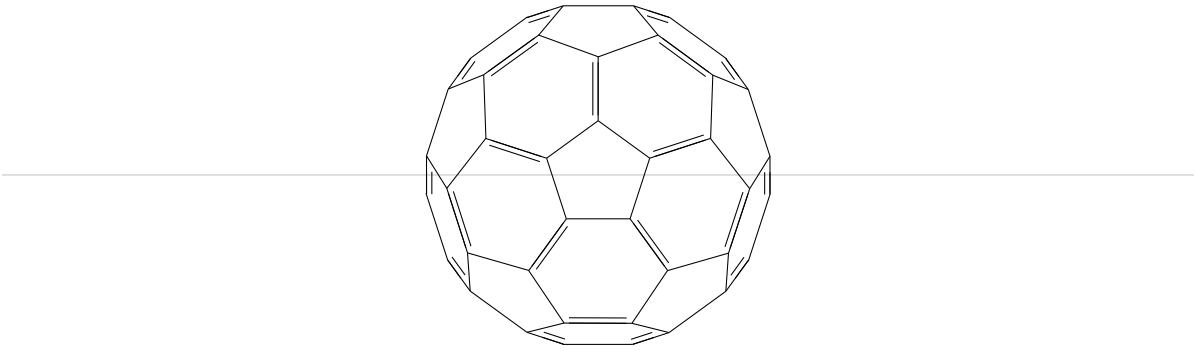
Menthone

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



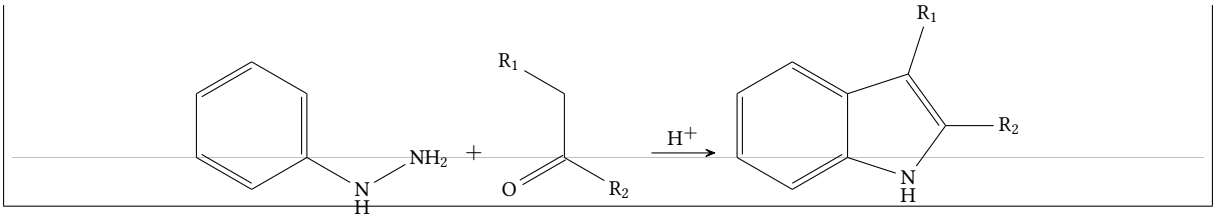
Fullerene

```
\definesubmol\fragment1{
  (-[:#1,0.85,,,draw=none]
  -[::126]-[::-54](=_#(2pt,2pt)[::180])
  -[::-70](-[::-56.2,1.07]=^#(2pt,2pt)[::180,1.07])
  -[::110,0.6](-[::-148,0.60](=^[::180,0.35])-[::-18,1.1])
  -[::50,1.1](-[::18,0.60]=_[::180,0.35])
  -[::50,0.6]
  -[::110])
}
\chemfig{
  !\fragment{18}
  !\fragment{90}
  !\fragment{162}
  !\fragment{234}
  !\fragment{306}
}
```



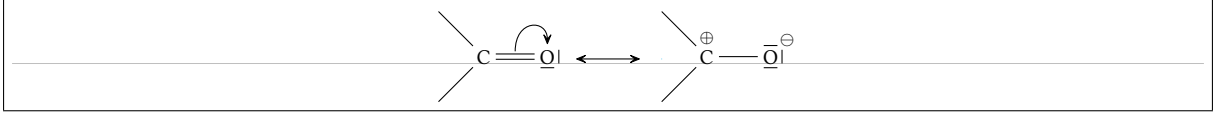
Fischer indole synthesis

```
\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
```



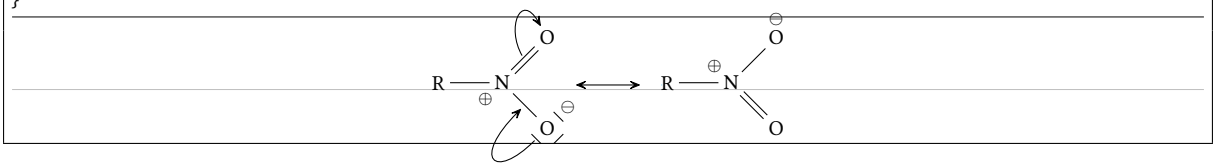
Reaction mechanisms: carbonyl group

```
\schemestart
  \chemfig{C([3]-)([5]-)=[@{db,.5}]@{atoo}\charge{0=\|,-90=\|}{O}}
  \arrow(.mid east--.mid west){<->}
  \chemfig{\charge{90:3pt=$\scriptstyle\oplus$}{C}([3]-)([5]-)-%
    \charge{0=\|,90=\|,-90=\|,45:3pt=$\scriptstyle\ominus$}{O}}
\schemestop
\chemmove{\draw[shorten <=2pt, shorten >=2pt](db) ..controls +(up:5mm) and +(up:5mm)..(atoo);}
```



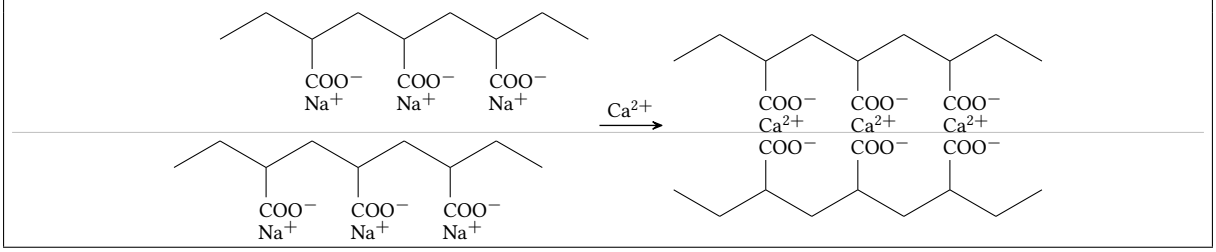
Reaction mechanisms: nitro group

```
\schemestart
  \chemfig{R-\charge{225:3pt=$\scriptstyle\oplus$}{N}([1]=[@{db}]@{atoo1}O)([7]-[@{sb}]@{atoo2}
  \charge{45=\|,-45=\|,-135=\|,45:5pt=$\scriptstyle\ominus$}{O})}
  \arrow(.mid east--.mid west){<->}
  \chemfig{R-\charge{135:3pt=$\scriptstyle\oplus$}{N}([1]-\charge{90:3pt=$\scriptstyle\ominus$}{O})([7]=O)}
\schemestop
\chemmove{
  \draw[shorten <=2pt, shorten >=2pt](db) ..controls +(120:5mm) and +(120:7mm)..(atoo1);
  \draw[shorten <=3pt, shorten >=2pt](atoo2) ..controls +(225:10mm) and +(225:10mm)..(sb);
}
```



Calcium alginate gel

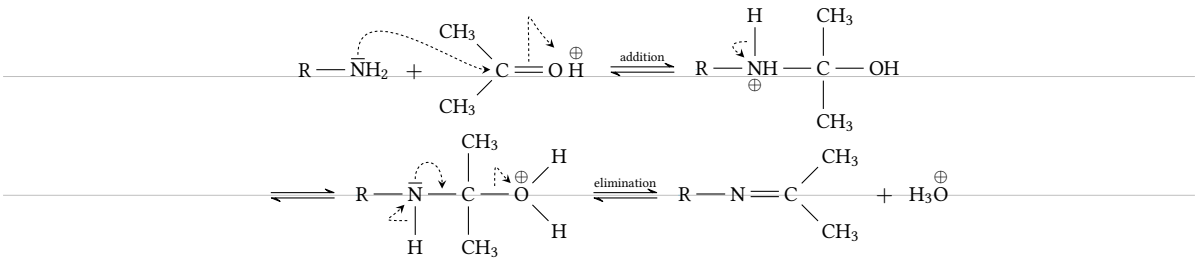
```
\definesubmol{0}{-[-3,.15,,,draw=none]}% invisible bond to stack
\definesubmol{COO/Na+}1{-[7]((-[-3,.66]COO^{-}!0N|a^+#1))}
\definesubmol{chain}{-[5]!{COO/Na+}{}-[5]!{COO/Na+}{}-[5]-[7]}
\definesubmol{COO/Ca2+}1{-[7]((-[-3,.66]COO^{-}!0C|a^{2+}!0COO^{-}-[-3,.66](-[7]-[5])#1))}
\hreac[fixed length,atom sep=2.5em,angle increment=30]
  \chemfig{-[5]!{COO/Na+}{!0\phantom{C}!0(!{COO/Na+}{}!{chain})(-[-1])}!{chain}}
  >[\printatom{Ca^{2+}}]
  \chemfig{-[5]!{COO/Ca2+}{(-[-1]-[1])}-[5]!{COO/Ca2+}{}-[5]!{COO/Ca2+}{}-[5]-[7]}
\endhreac
```



Nucleophilic addition. Primary amines

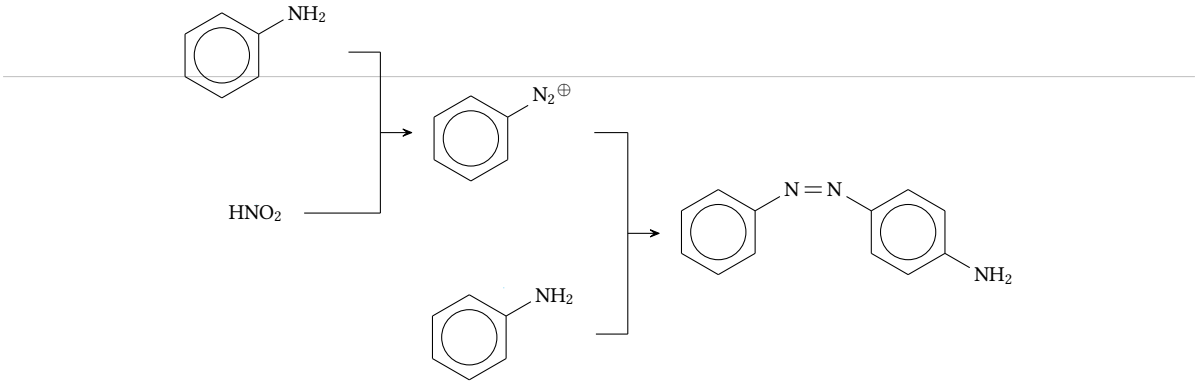
```
\setchemfig{atom sep=2.5em,compound sep=5em}
\schemestart
  \chemfig{R-@{aton}\charge{90=\|}{N}H_2}
  \+
  \chemfig{@{atoc}C([3]-CH_3)([5]-CH_3)=[@{atoo1}]O}
  \chemfig{@{atoo2}\chemabove{H}{\scriptstyle\oplus}}
  \chemmove[-stealth,shorten <=3pt,dash pattern= on 1pt off 1pt,thin]{
    \draw[shorten >=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}\charge{90=\|}{N}([6]-[@{sbh}]H)-[@{sb}]C(-[2]CH_3)(-[6]CH_3)-[@{sbo}]@{atoo}
  \chemabove{O}{\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 elimination]}\chemfig{R-N=C(-[1]CH_3)(-[7]CH_3)}
  \+
  \chemfig{H_3\chemabove{O}{\scriptstyle\oplus}}
\schemestop
```



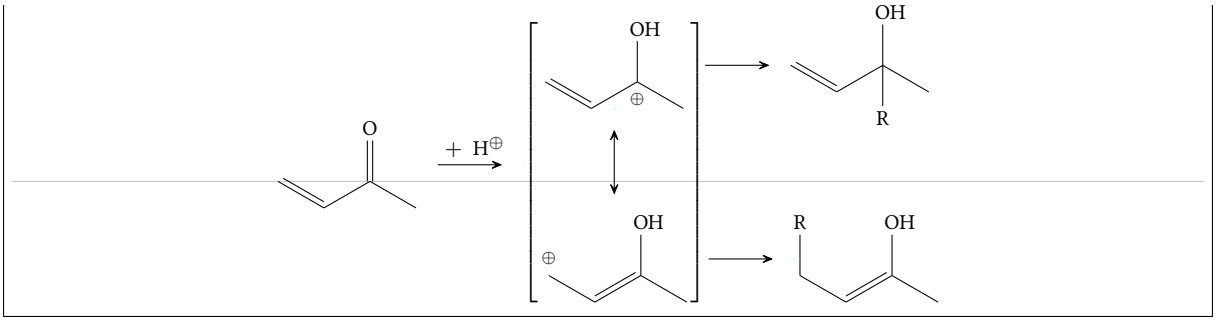
Reaction scheme

```
\setchemfig{atom sep=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
```



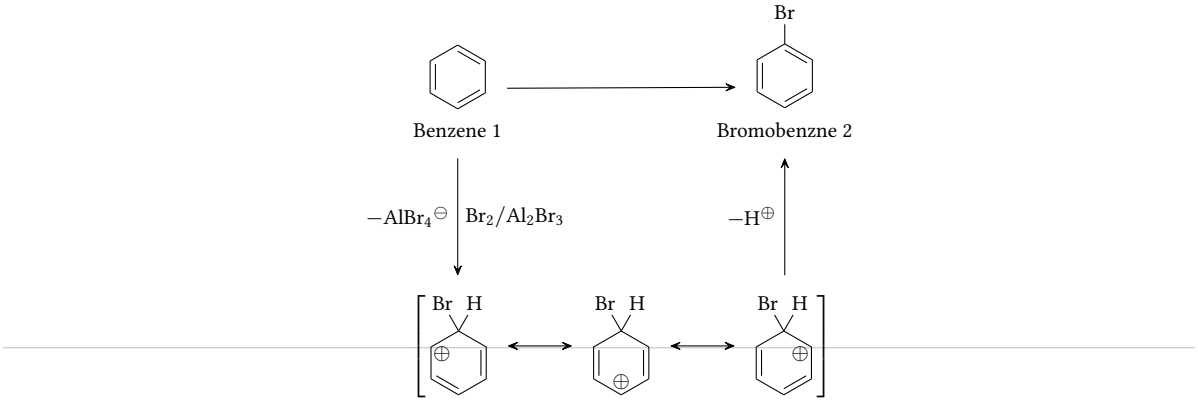
Addition

```
\setchemfig{atom sep=2.5em}
\schemestart
  \chemfig{*6(=-(-)(=[2]O))}
  \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
```



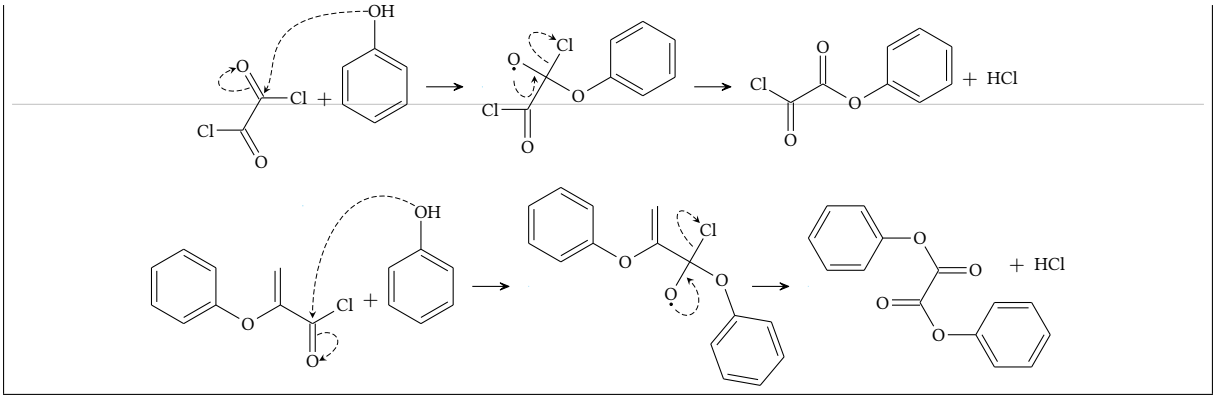
Electrophilic aromatic substitution

```
\setchemfig{atom sep=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(-=-=-=-)}}{Benzene 1}
  \arrow(@c4--){->[*0\chemfig{{-}H^\oplus}]}[90,1.5]
  \chemname{\chemfig{*6(-=-=(-Br)-=-)}}{Bromobenzne 2}
  \arrow(@c5.mid east--@c6.mid west)
\schemestop
\chemnameinit{}
```



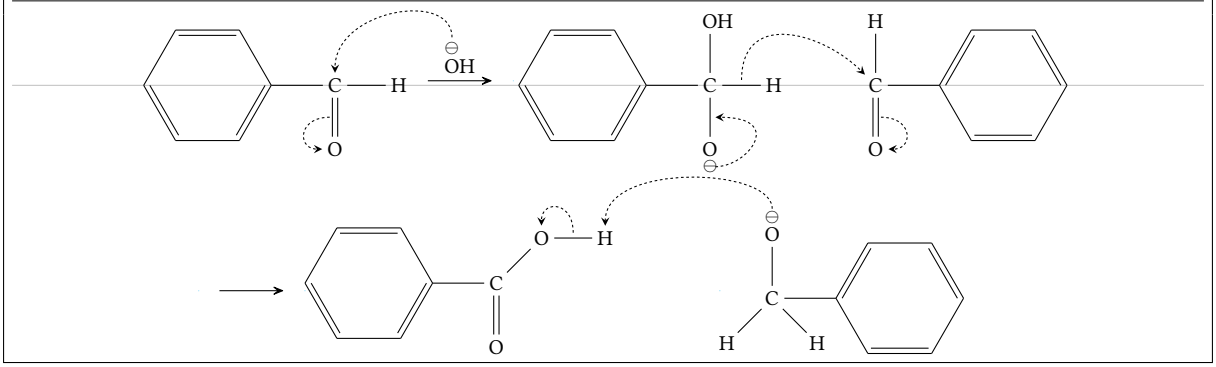
Reaction mechanism of chlorination

```
\scriptsize\setchemfig{bond offset=1pt,atom sep=2em,compound sep=4em}
\schemestart
  \chemfig{Cl-[4]@{a0}(=[@{a1}:120]@{a2}O)-[:-120](=[:-60]O)-[4]Cl}\+\chemfig{*6(-=-=(-@{oh1}OH)-=)}\arrow
  \chemfig{*6((-O-[:150](-[@{o0}:150]@{o1}\charge{-90=\.}{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}\charge{-90=\.}{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)..(o0);
  \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

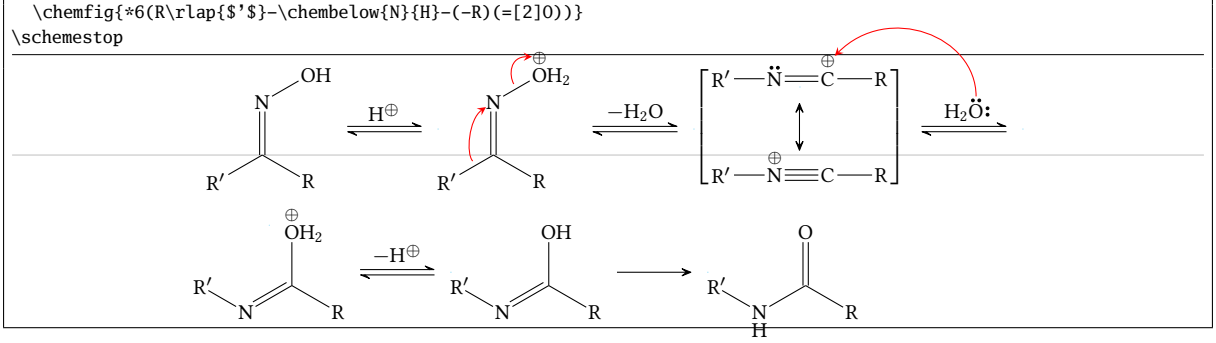
```
\schemestart
  \chemfig{[:-30]*6(=-=(-@{atoc}C([6]=[@{db}]@{atoo1}O)-H)-=-)}
  \arrow(start.mid east--.mid west){->[\chemfig{@{atoo2}\chemabove{O}{\scriptstyle\ominus}}H]}
  \chemmove[-stealth,shorten >=2pt,dash pattern=on 1pt off 1pt,thin]{
      \draw[shorten <=8pt](atoo2) ..controls +(up:10mm) 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);}
  \arrow(@start.base west--){0}[-75,2]
  {}
  \arrow
  \chemfig{[:-30]*6(=-=(-C([1]-@{atoo2}O-[@{sb}0]@{atoh}H)([6]=O))-=-)}
  \arrow{0}
  \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);}
\schemestop
```



Beckmann rearrangement

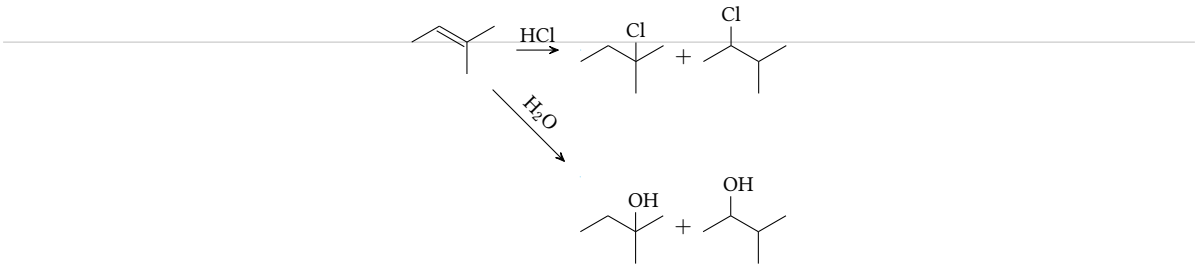
```
\setchemfig{bond offset=1pt,atom sep=2.5em,compound sep=5em,arrow offset=6pt}
\schemestart
  \chemfig{(-[:-150]R')(-[:-30]R)=[2]N-[:30]OH}
  \arrow{<=>[\chemfig{H^\oplus}]}
  \chemfig{(-[@{a0}:-150]R')(-[:-30]R)=[2]@{a1}N-[@{b0}:30]@{b1}\chemabove{O}{\scriptstyle\oplus}H_2}
  \chemmove[red,-stealth,red,shorten <=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'-\charge{90=\:}{N}=@{a1}\chemabove{C}{\scriptstyle\oplus}-R}}\chemright]
  \arrow{<=>[\chemfig{H_2@{a0}\charge{0=\:,90=\:}{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
```



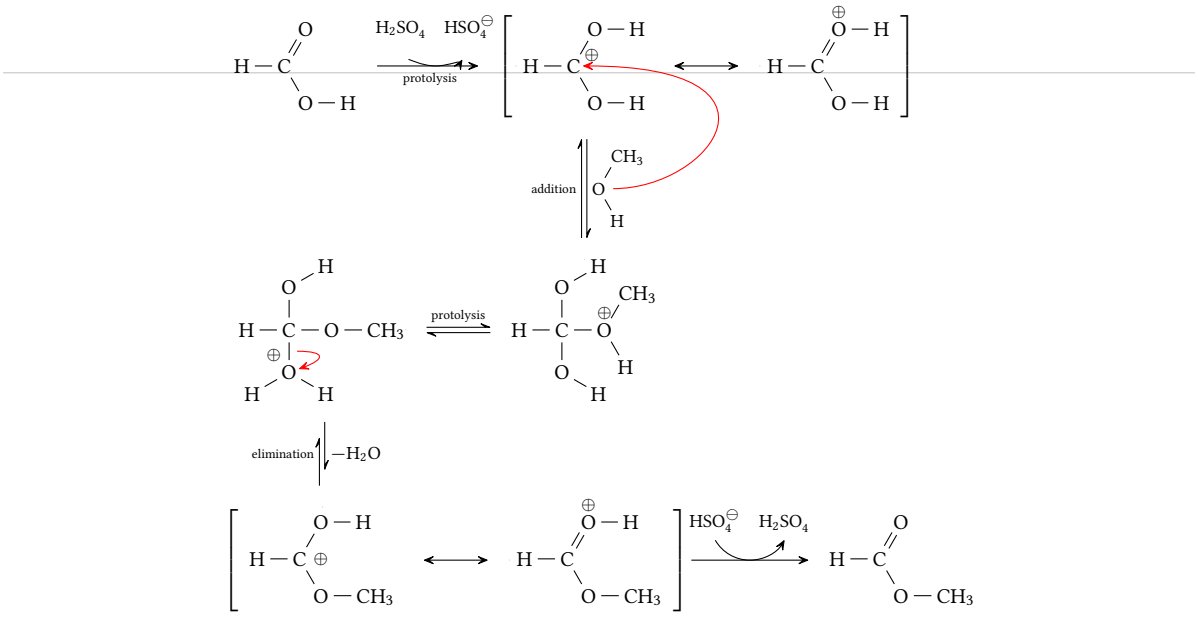
Reaction scheme

```
\setchemfig{atom sep=1.5em,compound sep=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_2O}]}[-45,1.7]
  \chemfig{-[::30]-[::-60](-[::120]OH)(-[::-60])-[::60]}\+\chemfig{-[::30](-[::60]OH)-[::-60](-[::-60])-[::60]}
\schemestop
```



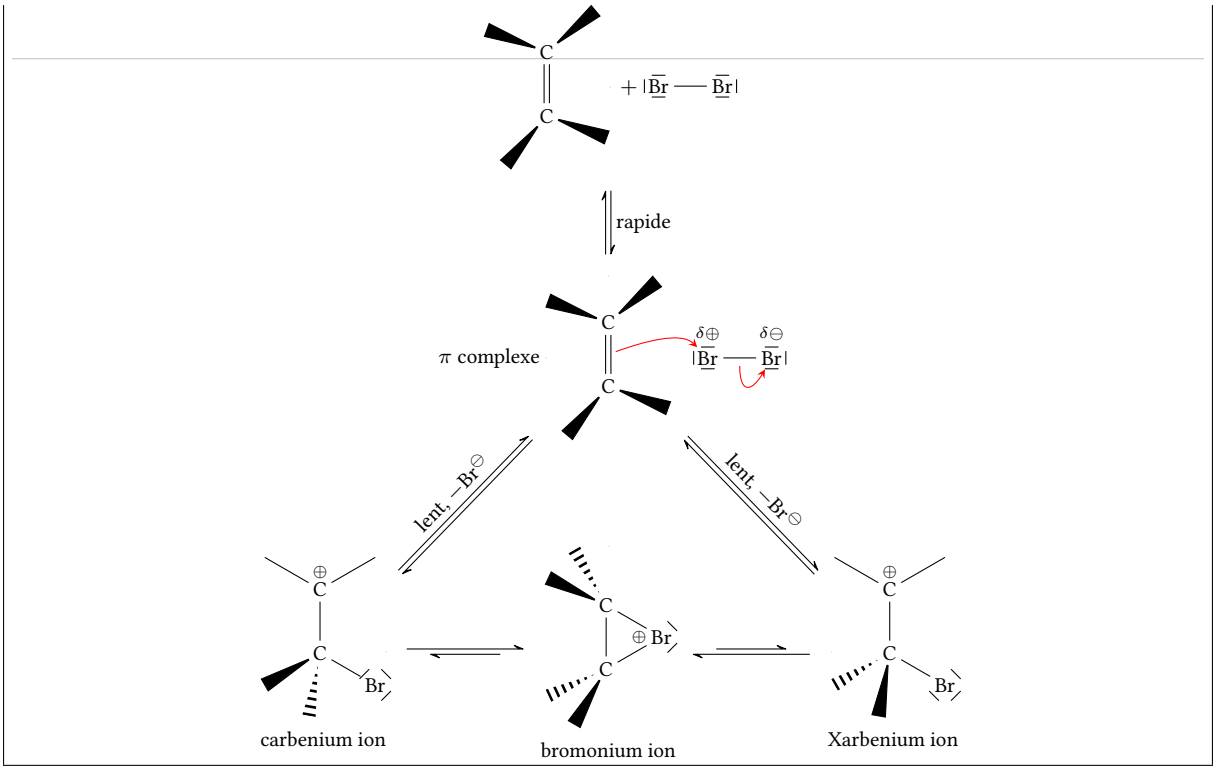
Esterification of formic acid

```
\tikzset{obrace/.style={left delimiter={[},inner sep=3pt},
  cbrace/.style={right delimiter={]},inner sep=3pt},
  braces/.style={left delimiter={[},right delimiter={]},inner sep=3pt}}
\setchemfig{atom sep=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_2O}][*{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
```



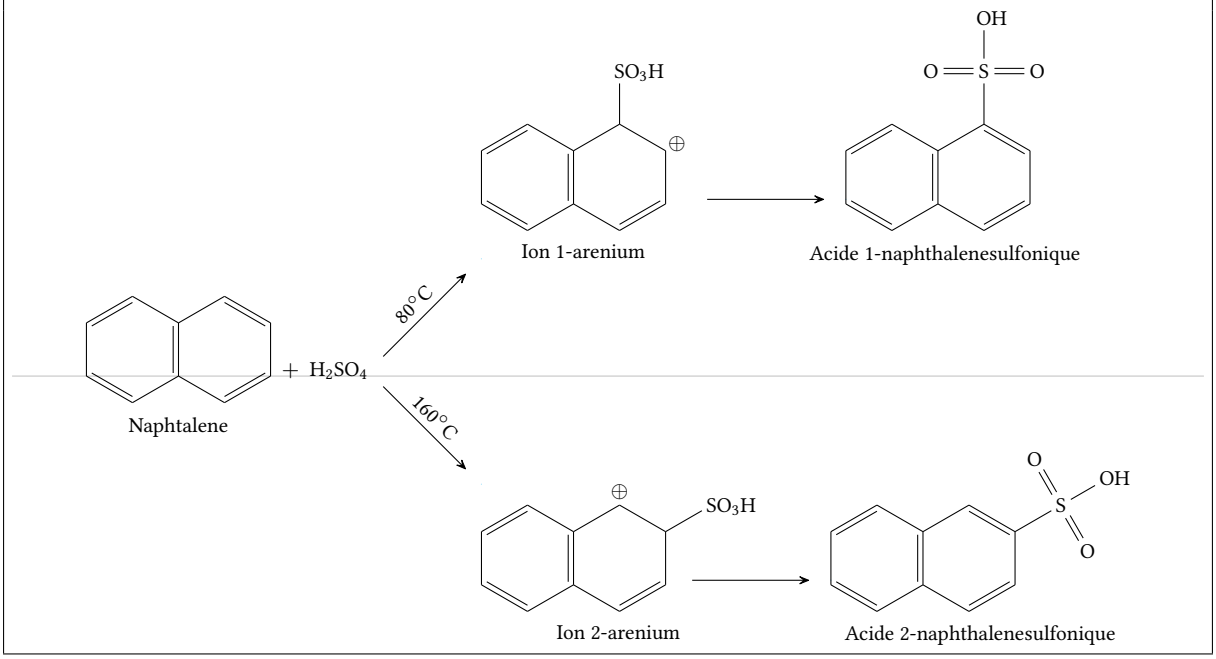
Electrophilic addition of halogen to olefin

```
\schemestart
  \subscheme{%
    \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
    \arrow{0}[,0]\+\chemfig{\charge{90=\|,180=\|,270=\|}{Br}-\charge{90=\|,0=\|,270=\|}{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])-\charge{45=\|,-45=\|,180:3pt=$\scriptstyle\oplus$}{Br}-C(<:)(<[:155])-)}}
    {bromonium ion}
  \arrow(--carbeniumA){<<->}[,1.5]
  \chemname{\chemfig{-[:-30]\chemabove{C}{\scriptstyle\oplus}(-[:30])-[6]C(<:[:-150])(<[:-100])-[:-30]
    \charge{45=\|,-45=\|,225=\|}{Br}}}{Xarbenium ion}
  \arrow(@bromonium--carbeniumB){<<->}[180,1.5]
  \chemname{\chemfig{-[:-30]\chemabove{C}{\scriptstyle\oplus}(-[:30])-[6]C(<[:-150])
    (<:[:-100])-[:-30]\charge{45=\|,-45=\|,135=\|}{Br}}}{carbenium ion}
  \arrow(@olefin--){0}[,.25]
  \chemfig{@{Br1}\charge{90=\|,180=\|,270=\|,90:5pt=$\scriptstyle\delta\oplus$}{Br}-[@{b2}]@{Br2}
    \charge{90=\|,0=\|,270=\|,90:5pt=$\scriptstyle\delta\ominus$}{Br}}
  \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
\chemnameinit{}
```



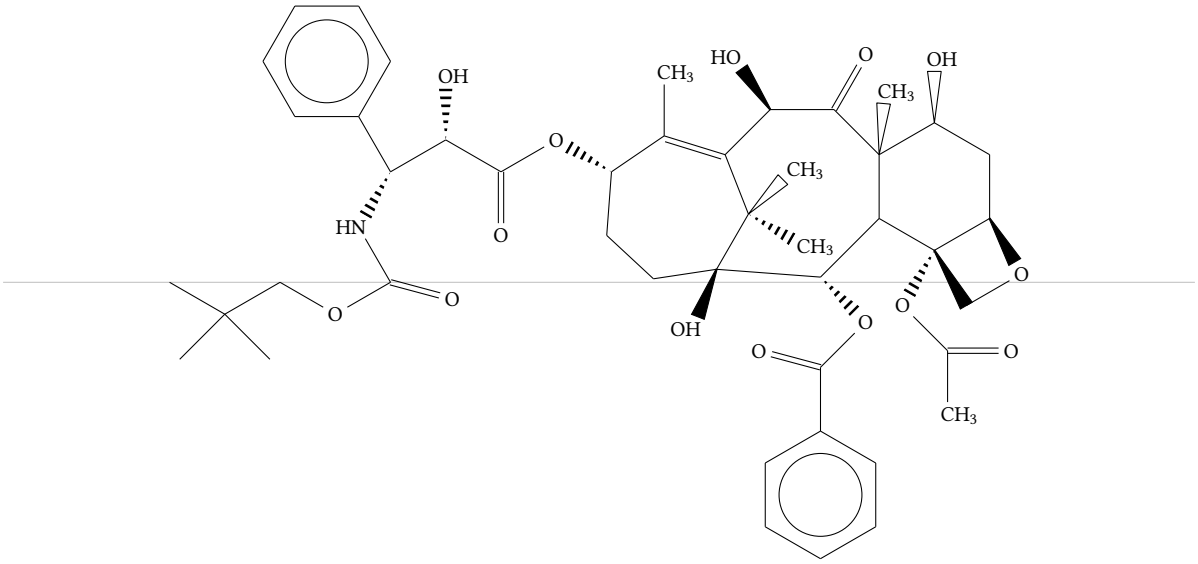
Sulfonation of naphthalene

```
\definesubmol\cycleoplus{-[,0.25,,,draw=none]\oplus}
\definesubmol{so2oh}{S(=[::90]O)(=[::-90]O)-OH}
\setchemfig{atom sep=2.5em}
\schemestart[,1.5]
  \chemname{\chemfig{*6(=-*6(-=-=-)=-=-)}}{Naphtalene}\+\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
\chemnameinit{}
```



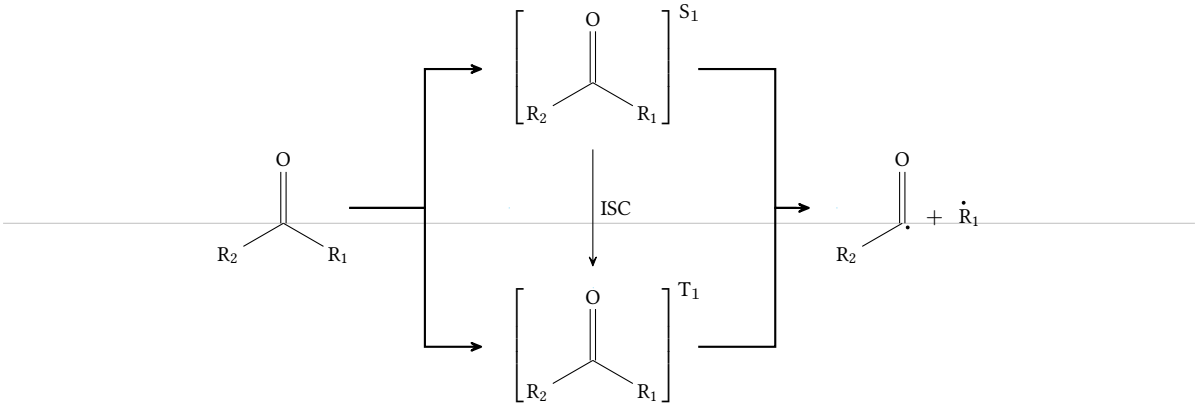
Taxotere

```
\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)-)}
```



Diverging arrows

```
\schemestart
  \chemfig{(-[:210]R_2)(-[:330]R_1)=[2]O}
  \arrow(a--)[,1.5,,,draw=none]
  \subscheme{
    \charge{30:4pt=$\mathrm{S}_1$}{\chemleft{[}\chemfig{(-[:210]R_2)(-[:330]R_1)=[2]O}\chemright{]}}
    \arrow(b--c){->[*{0}ISC]}[-90,1.5]
    \charge{30:4pt=$\mathrm{T}_1$}{\chemleft[\chemfig{(-[:210]R_2)(-[:330]R_1)=[2]O}\chemright{]}}
  }
  \arrow(--d)[,1.5,,,draw=none]
  \chemfig{(-[:210]R_2)(-[:330,0.1,,,draw=none]\charge{330:-1pt=\.\,}{})=[2]O}
  \+
  \chemfig{\charge{90:1pt=\.\,}{R}_1}
  \schemestop
\chemmove{
  \draw[thick,shorten >=10pt] (a.east) -- ++(1,0) |- (b.west);
  \draw[thick,shorten >=10pt] (a.east) -- ++(1,0) |- (c.west);
  \draw[-,thick,shorten >=15pt,shorten <=8pt] (b.east) -- ++(1.3,0) |- (d.west);
  \draw[thick,shorten >=10pt,shorten <=8pt] (c.east) -- ++(1.3,0) |- (d.west);
}
```



Change history

Changes (most recent first) made in chem*fig* since the first version. This section has not been translated into English.

Version 1.71 du 30/10/2025

- bugfix : trop de mauvais raccords de liaisons avec la nouvelle macro `\CF_ifzerodim`. Retour à la précédente définition (la macro `\CF_ifzerodim` n'est utilisée que si `use atom strut` = ⟨*true*⟩).
  Les codes donnent à nouveau les bons raccords de liaisons :
  1. `\chemfig{?-A-[:90]B?}`
  2. `\chemfig{A-@{a}-B}`
  3. `\chemfig{[:30]-\charge{90=a}{}-[:-30]}`

  On obtient bien `3,6` avec le code suivant :

  ```
  \newcount\xx
  \begingroup
    \def\printatom#1{\global\advance\xx1 \ensuremath{\mathrm{#1}}}
    \xx=0 \setbox0\hbox{\chemfig{A-B-C}} \the\xx,
    \xx=0 \setbox0\hbox{\chemfig[use atom strut]{A-B-C}}\the\xx
  \endgroup
  ```

- bugfix : prise en compte de l'argument optionnel de `\chemfig` dans l'environnement `hreac`.
  Le code `\hreac\chemfig[angle increment=+30]{A-[1]B} > C\endhreac` ne plante plus à cause du `+` dans l'argument optionnel
- bugfix : prise en compte de l'argument optionnel de `\chemname` dans l'environnement `hreac`.
- bugfix : dans l'environnement `hreac`, l'argument de `\^{`⟨*dimension*⟩`}` n'est plus évaluée à `0pt` si exprimé en ex ou em.
- ajout : par souci de cohérence, dans l'environnement `hreac`, la macro `\name[`⟨*dim*⟩`]{`⟨*nom*⟩`}` a un argument optionnel pour spécifier l'espacement du ⟨*nom*⟩ avec le ⟨*composé*⟩
- mise en place de warnings si utilisation d'un ou deux arguments optionnels de `\schemestart`. Création de la clé `init anchor` pour accéder au réglage possible avec le deuxième argument optionnel.
- mise en cohérence : la clé `name sep` vaut désormais `1.5`ex par défaut et s'applique à `\chemname` si son argument optionnel est vide

Version 1.7 du 26/10/2025

- réaction horizontales avec l'environnement

  `\hreac...\endhreac`

  qui offre une syntaxe plus simple que `\schemestart...\schemestop`
- par défaut désormais, l'argument de `\printatom` est développé et dépourvu de strut [12].
  Afin d'assurer la compatibilité, le nouveau booléen `use atom strut` peut être mis à ⟨*true*⟩ afin de retrouver le comportement précédent
- lorsque la clé `use atom strut` = ⟨*false*⟩, chaque atome n'est composé typographiquement qu'une seule fois alors que c'était parfois 5 fois ou plus auparavant puisque `\vphantom` compose son argument dans une boite de TEX.
  La macro `\CF_ifzerodim` a été rapidement corrigée pour que ce soit le cas. Il reste encore des investigations à faire pour que le code soit plus propre…
  Il se peut que ce changement provoque des petits bugs d'affichage dans le dessin de certaines molécules. Merci de me les signaler.
- suppression de l'historique des changements dans `chemfig.tex` pour le mettre ici dans le manuel.

[12] dysfonctionnement aimablement signalé sur 2 sites par Joseph Wright avec son amusante habitude d'exprimer publiquement le mal qu'il pense de mes packages.

Version 1.66 du 28/12/2023

- les liaisons de Cram pleines sont jointes entre elles ou aux liaisons simples lorsque `bond join` = ⟨*true*⟩

Version 1.6e du 30/06/2023

- nouvelle clé `baseline` pour régler finement l'alignement vertical d'une molécule

Version 1.6d du 18/02/2023

- les ancres 'b' et 'd' n'étaient pas prises en compte pour le tracé de flèches directes de type `(@a.b-@c.d)` dans les schémas réactionnels
- correction d'un bug : `\CF_currentfromatom` n'était pas initialisé au début d'un cycle et donc le code suivant plantait `\chemfig{AB-[,,2]*3(---)}`

Version 1.6c du 27/09/2022

- nouvelles clés `gchemname`, `schemestart code` et `schemestop code` (suggestion de Balazs Debreceni)

Version 1.6b du 01/08/2021

- encodage UTF-8
- la macro `\#` n'était pas définie pour remplacer `#(...)` lorsque `\chemfig` se trouve dans l'argument d'une macro.

Version 1.6a du 28/02/2021

- le fichier `lewis.tex` a été renommé `chemfig-lewis.tex`

Version 1.6 du 26/02/2021

- les macros des formules de Lewis sont retirées et placées dans le fichier séparé "`lewis.tex`" que l'utilisateur peut charger s'il le souhaite
- ajout d'une clé `debug` pour le trousseau `[chemfig]`
- à l'intérieur d'un schéma, le token '`#`' est permis dans l'argument de `\chemfig`

Version 1.56 du 13/07/2020

- le centre des cycles est désormais accessible via un nœud spécifique pour chacun d'eux.

Version 1.55 du 15/06/2020

- chemfig est incompatible avec conTEXt, vu que ce moteur redéfinit des primitives telles que `\expanded`, `\unexpanded` et peut être d'autres.

Version 1.54 du 21/05/2020

- chemfig ne peut plus fonctionner sans `\expanded`
- bug : un signe "=" laissé par erreur dans le flux

Version 1.53 du 27/04/2020

- mise à jour en fonction des nouvelles fonctionnalités de l'extension simplekv
- bug : `\CF_ifzerodim` interrompt maintenant le tracé dans la `\hbox`

Version 1.52 du 14/04/2020

- bug : définition corrigée de `\CFthesubmol` dans `\CF_defsubmolc` pour qu'elle se développe en 1 coup seulement

Version 1.51 du 06/04/2020

- bug corrigé dans \chargerect_a et \chargeline_a

Version 1.5 du 05/03/2020

- nouvelles macros \charge et \Charge. Les macros \lewis et \Lewis sont obsolètes et amenées à disparaitre à moyen terme (au moins 9 mois), soit fin 2020
- prise en compte de la dimension d'un groupe d'atome pour tracer des liaisons jointives
- bug corrigé dans \CF_searchnode
- ajout d'une section dans le manuel (placement des atomes)

Version 1.41 du 21/05/2019

- utilisation de la nouvelle primitive \expanded
- nouvelle clé h align (⟨*true*⟩ par défaut) pour les délimiteurs de \polymerdelim. Lorsque à ⟨*false*⟩, les délimiteurs ne sont plus alignés horizontalement mais positionnés aux noeuds demandés
- nouvelle clé auto rotate qui n'a de sens que si h align = ⟨*false*⟩ : les délimiteurs sont automatiquement inclinés
- nouvelle clé rotate qui n'a de sens que si halign = ⟨*false*⟩ ET auto rotate = ⟨*false*⟩ : l'inclinaison des délimiteurs peut êtreà false choisie

Version 1.4 du 18/04/2019

- corrections de nombreux bugs
- caractère privé "_" et non plus "@" d'où des modifications à prévoir notamment dans la doc avec les codes spécifiques aux flèches, ça risque de couiner sur tex.stackexchange.com
- anciennes macros abandonnées et désormais indéfinies :
  \setcrambond, \setatomsep, \setbondoffset, \setdoublesep, \setangleincrement, \enablefixedbondlength, \disablefixedbondlength, \setnodestyle, \setbondstyle, \setlewis, \setlewisdist, \setstacksep, \setarrowdefault, \setcompoundstyle, \setandsign, \setarrowoffset, \setcompoundsep, \setarrowlabelsep, \enablebondjoin, \disablebondjoin et \schemedebug
- l'ancienne syntaxe \chemfig[][]{} est abandonnée et n'est plus acceptée, désormais c'est

  \chemfig[⟨*clés*⟩=⟨*valeurs*⟩]{⟨*code molécule*⟩}
- l'ancienne syntaxe \lewis[⟨*coeff*⟩] ou \Lewis[⟨*coeff*⟩] n'est plus acceptée au profit de \lewis[⟨*clés*⟩=⟨*valeurs*⟩]

Version 1.34 du 23/02/2019

- bug dans la flèche "<->" corrigé

Version 1.33 du 31/10/2018

- les macros définies par \definesubmol peuvent désormais avoir un ou plusieurs arguments
- macro \polymerdelim documentée

Version 1.32 du 23/08/2018

- définition de \printatom, \CF_begintikzpicture et \CF_endtikzpicture dans le fichier t-chemfig.tex

Version 1.31 du 05/04/2018

- correction d'un espace indésirable dans \CF_ifnextchar

Version 1.3 du 08/03/2018

- tous les paramètres sont désormais passés via \setchemfig qui fait appel à "simplekv". Par conséquent, *toutes* les macros qui réglaient des paramètres deviennent obsolètes, à savoir :
  \setcrambond, \setatomsep, \setbondoffset, \setdoublesep, \setangleincrement, \enablefixedbondlength, \disablefixedbondlength, \setnodestyle, \setbondstyle, \setlewis, \setlewisdist, \setstacksep, \setarrowdefault, \setcompoundstyle, \setandsign, \setarrowoffset, \setcompoundsep, \setarrowlabelsep, \enablebondjoin, \disablebondjoin et \schemedebug
  Ces macros seront **supprimées** dans une future version.

- la version étoilée `\chemfig*` et les deux arguments optionnels de la macro `\chemfig[][]` sont également optionnels et seront **supprimés** dans une future version afin d'accéder à la syntaxe `\chemfig[clés=valeurs]{code}`
- 6 nouveaux paramètres : `lewis radius`, `arrow double sep`, `arrow double coeff`, `arrow double harpoon`, `cycle radius coeff`, `arrow head`.
- correction d'un bug dans `\CF_parsemergeopt` qui dans certains cas, envoyait vers l'affichage des caractères
- petit toilettage du code
- macro `\polymerdelim` (non documentée) expérimentale et encore en phase de tests
- suppression d'un registre d'écriture de fichier

Version 1.2e du 20/05/2017

- la macro contenant la définition d'une flèche est désormais "`\CF_arrow(`⟨*nom*⟩`)`", ainsi la macro `\0` n'est plus définie par `\definearrow`
- remerciements rajoutés après une suppression indue, pour ne froisser aucune susceptibilité

Version 1.2d du 01/12/2015

- correction d'un bug dans la flèche "`-U`"
- la version étoilée de `\setcrambond` dessine les liaisons de Cram en pointillés sous forme de trait large et non pas sous forme de triangle.

Version 1.2c du 20/11/2015

- Correction d'un bug dans `\CF_setbondangle` : l'angle renvoyé pouvait être négatif
- Correction d'un bug dans `\CF_directarrow` : la macro `\CF_ifempty` n'est pas correctement développée dans l'argument de `\pgfpointanchor`

Version 1.2b du 15/11/2015

- bug dans `\CF_searchsubmol` qui laissait "`*`" dans le flux de lecture de TeX. Un message d'erreur est également ajouté en cas de "`!`" en fin de traitement.
- correction d'un bug dans `\CF_setbondangle` où l'angle `[`⟨`:a`⟩`]` n'était pas évalué par `\pgfmathsetmacro`.

Version 1.2a du 21/10/2015

- erreur de copier-coller dans le code: une adresse url était malencontreusement présente en plein milieu d'une ligne de code

Version 1.2 du 08/10/2015

- correction d'un bug dans le tracé des liaisons de Cram.
- création de `\setangleincrement`.
- chargement de "arrows.meta" et définition de la flèche "`CF`" basée sur "`Stealth`" et définie avec `\pgfdeclarearrow`. Les anciennes flèches "`CF_full`" et "`CF_half`" sont abandonnées puisque définies avec `\pgfarrowsdeclare`.
- flèche "-U>" corrigée : le placement des labels est maintenant correct dans tous les cas. Ainsi :

  $$\texttt{-U>[}\langle a\rangle\texttt{][}\langle b\rangle\texttt{][}\langle d\rangle\texttt{][}\langle r\rangle\texttt{][}\langle a\rangle\texttt{]}$$

  place le label ⟨a⟩ près du début de la flèche, quels que soient les signes du rayon ⟨r⟩ et de l'angle ⟨a⟩.
- `\chemrel`, `\setchemrel` et `\chemsign` sont supprimées.
- compatibilité, avec les limitations d'usage, avec la librairie "`externalize`" : le `\begin{tikzpicture}` voit désormais le `\end{tikzpicture}` correspondant dans la macro `\CF_chemfigb`.

Version 1.1a du 23/02/2015

- correction d'un bug dans `\CF_grabbondoffset`. Si `\chemfig` est dans l'argument d'une macro, les "`#`" sont doublés par l'action de `\scantokens` de la macro `\CF_chemfigb` et il faut un argument délimité avant "`(`" pour absorber tous les "`#`".

Version 1.1 du 13/02/2015

- correction d'un bug dans `\CF_searchsubmol` : la macro `\CF_molecule` est dépouillé de son éventuel espace en première position.
- correction d'un bug dans `\CF_arrowf` : le nom du prochain nœud courant "`end@arrow@i`" était erroné dans le cas où une flèche contenait un sous schéma. Ce nom doit dépendre de `\CF_schemenest`.
- la jonction entre deux liaisons consécutives dans l'axe peut être activé avec `\enablebondjoin` et désactivé avec `\disablebondjoin` (préférable, état par défaut).
- `\chemfig` suivi d'une "`*`" demande à ce que les liaisons aient une longueur invariable : la distance inter-atome devient donc variable. Cette fonctionnalité est désactivé dans les cycles afin que les polygones soient réguliers. `\enablefixedbondlength` permet cette fonctionnalité pour toutes les macros `\chemfig` (même non étoilée) tandis que `\disablefixedbondlength` le désactive.

Version 1.0h du 28/11/2013

- `\chemname` admet maintenant une version étoilé qui ne tient pas compte des profondeurs précédentes.
- `\CF_dpmax` est géré globalement.
- correction d'un bug dans "`-U>`" : le style de la flèche n'était pris en compte pour l'arc.
- correction d'un bug dans `\CF_directarrow` : l'angle de la flèche n'était pas calculé

Version 1.0g du 16/11/2012

- correction d'un bug dans `\CF_directarrow` pour faire prendre en compte le style des flèche par défaut
- correction d'un bug dans `\CF_lewisc` : la boite *doit* être composée en dehors de l'environnement tikzpicture pour éviter nullfont si jamais `\printatom` ne passe pas en mode math.
- correction d'un bug dans `\CF_chemfigc` : si une longueur par défaut est modifiée par `[,`$\langle l \rangle$`]` au début d'une molécule et si des cycles étaient emboités, cette longueur n'était pas appliquée aux sous-cycles.
- ré-écriture des macros `\chemabove` et `\chembemow` pour prendre en compte le bug (désormais corrigé) dans luatex.
- nouvelle macro `\setstacksep` qui définit l'espacement par défaut dans les macros `\chemabove` et `\chembelow`.

Version 1.0f du 24/02/2012

- correction d'un bug avec `\definesubmol`, les catcodes n'étaient pas correctement gérés.

Version 1.0e du 13/01/2012

- la gestion des espaces dans les groupes d'atomes est désormais plus rigoureuse. Plusieurs bugs ont été corrigés

Version 1.0d du 19/12/2011

- les cercles des cycles étaient tracés au mauvais moment. La longueur de la liaison qui les précédait influait sur le rayon du cercle :

  ```
  \chemfig{-[,0.5]**6(------)}
  ```

  donnait un bug à l'affichage.

Version 1.0c du 30/11/2011

- la macro `\+` n'est plus explicitement écrite
- vérifie que eTeX est le moteur utilisé

Version 1.0b du 29/11/2011

- la commande `\merge` est désormais protégée entre `\schemestart` et `\schemestop` contre des définitions par d'autres packages.
- `\box0` est utilisé au lieu du maladroit `\unhbox0`

Version 1.0a du 18/09/2011

- les macros `\Lewis` et `\lewis` admettent un argument optionnel
- la macro `\setlewisdist` règle la distance entre les 2 électrons

Version 1.0 du 15/06/2011

- les schémas réactionnels sont désormais disponibles.
- `\Chemabove` et `\Chembelow` modifient la boite englobante.
- `\Lewis` modifie la boite englobante
- les macros `\chemleft`, `\chemright`, `\chemup` et `\chemdown` affichent des délimiteurs extensibles à gauche, à droite, au dessus et au dessous d'un matériel.

Version 0.4b du 24/04/2011

- l'argument de `\chemfig` est tokénisé avec `\scantokens` ce qui rend caduc tout souci de code de catégorie, à part `#`.
- la commande `\setbondstyle` permet de définir le style des liaisons.
- correction de l'affichage incorrect des doubles liaisons dans les cycles après les commandes `\hflipnext` et `\vflipnext`
- correction d'un bug lorsqu'un alias commence une molécule

Version 0.4a du 10/04/2011

- Correction d'un bug concernant l'argument optionnel en début de molécule.

Version 0.4 du 07/03/2011

- chemfig est désormais écrit en plain-etex et donc utilisable par d'autres formats que LaTeX.
- Un peu plus de rigueur avec les catcodes des caractères spéciaux, notamment lorsque la commande `\chemfig` se trouve dans l'argument de `\chemmove`, `\chemabove`, `\chembelow`, `\chemrel`. TODO : faut-il `\scantoken` l'argument de `\chemfig` pour être définitivement débarrassé de ces histoires de catcode ???
- Correction d'un bug dans le calcul de l'angle des liaisons

Version 0.3a du 08/01/2011

- Correction d'un bug dans l'argument optionnel de `\definesubmol` lorsque celui-ci comporte des crochets.
- Mise à jour du manuel en anglais.
- Ajout de `\vflipnext` et `\hflipnext` pour retourner horizontalement ou verticalement la prochaine molécule.

Version 0.3 du 21/11/2010

- Amélioration de `\definesubmol` qui accepte les séquences de contrôle. On peut aussi choisir un alias dont la substitution est différente selon l'orientation de la liaison qui lui arrive dessus.
- Le caractère "`|`" force la fin d'un atome. Si on écrit "`D|ef`" alors, chemfig verra deux atomes "`D`" et "`ef`".
- Le caractère "`#`" est reconnu lorsqu'il suit un caractère de liaison. Il doit être suivi d'un argument entre parenthèses qui contient l'offset de début et de fin qui s'appliqueront à cette liaison.
- La macro `\chemfig` admet un argument optionnel qui sera passé à l'environnement tikzpicture dans lequel elle est dessinée
- Mise en place de la représentation des mécanismes réactionnels avec la syntaxe "`@{`⟨*nom*⟩`}`" devant un atome où "`@{`⟨*nom*⟩`,`⟨*coeff*⟩`}`" au tout début de l'argument d'une liaison. Cette syntaxe permet de placer un nœud (au sens de tikz) qui deviendra l'extrémité des flèches des mécanismes. Le tracé des flèches est faite par la macro `\chemmove` dont l'argument optionnel devient celui de l'environnement tikzpicture dans lequel sont faites les flèches.
- Pour le mécanisme d'alignement vertical via le `\vphantom`, la commande `\chemskipalign` permet d'ignorer le groupe d'atomes dans lequel elle est écrite.
- La commande `\chemname` permet d'afficher un nom sous une molécule. la commande `\chemnameinit` initialise la plus grande profondeur rencontrée.
- La commande `\lewis` a été modifiée de telle sorte que les dessins des décorations soient proportionnels à la taille de la police.

Version 0.2 du 31/08/2010

- Ajout de la documentation en anglais.
- Correction de bugs.
- `\printatom` est désormais une macro publique.
- Les espaces sont permis dans les molécules. Ils seront ignorés par défaut puisque les atomes sont composés en mode math par `\printatom`.

- Une paire de Lewis peut être représentée "`:`".
- Dans les cycles, une correction de la longueur du trait déporté des liaisons doubles est fait de telle sorte que si l'on écrit `\chemfig{*5(=====)}`, on obtient deux polygones réguliers concentriques.
- La séquence de contrôle `\setnodestyle` permet de spécifier le style des noeuds dessinés par tikz.

Version 0.1 du 23/06/2010

- Première version publique sur le CTAN