The TEX package

chemfig

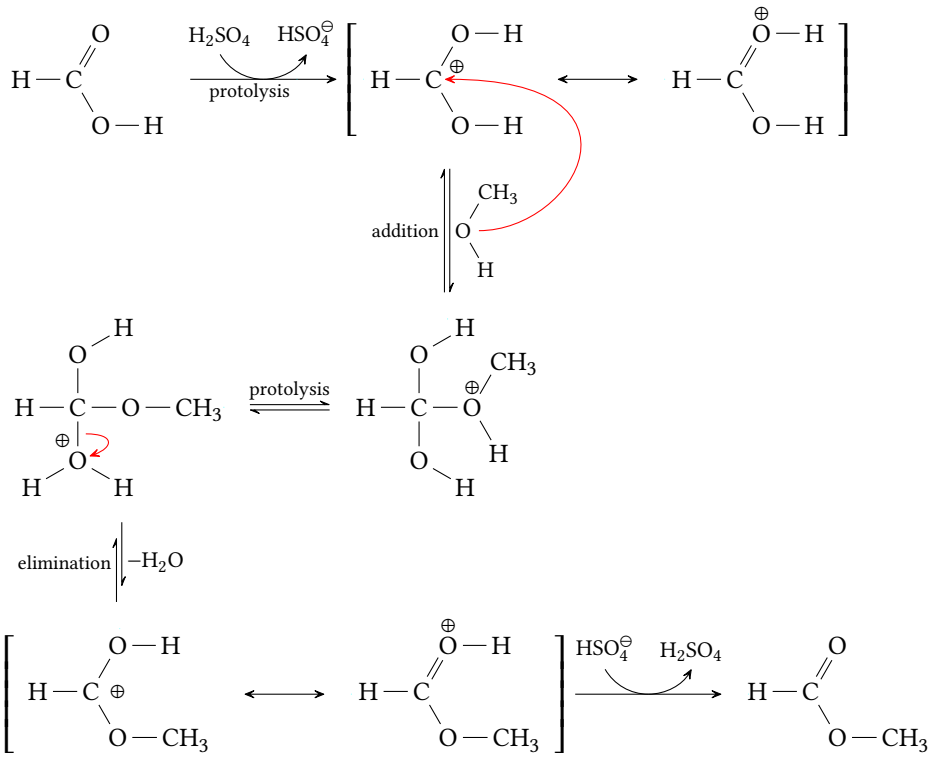
v1.84

November 14, 2026

User's manual

Christian Tellechea

unbonpetit@netc.fr



1 The \chemfig command 4
1.1 New in version 1.84 . . . 4
1.2 Syntax . . . 4
1.3 Parameters of \chemfig . . . 4

2 Atoms and bonds 6
2.1 Atoms and groups of atoms . . . 6
2.2 Role of the first atom . . . 6
2.3 Bonds between groups of atoms . . . 6
2.4 Optional argument of a bond . . . 7
2.4.1 Bond angle: predefined angles . . . 7
2.4.2 Bond angle: absolute angles . . . 8
2.4.3 Bond angle: relative angles . . . 8
2.4.4 Bond length: influence of the size of atoms . . . 8
2.4.5 Fixed bond lengths . . . 8
2.4.6 Bond length: setting with the optional argument . . . 9
2.4.7 Departure and arrival atoms . . . 9
2.4.8 Customizing a bond . . . 9
2.5 Joins between bonds . . . 10
2.6 Default values for a molecule . . . 11
2.7 Branches . . . 11
2.7.1 Principle . . . 11
2.7.2 Example: the acid anhydride molecule . . . 12
2.8 Connecting distant atoms . . . 12

3 Rings 14
3.1 Syntax . . . 14
3.2 Angular position . . . 15
3.3 Branches leaving a ring . . . 15
3.4 Fused rings . . . 16
3.5 Groups of atoms in a ring . . . 17

4 Representing electron movements 18
4.1 Mesomeric effect . . . 18
4.2 Center of a ring . . . 19
4.3 Reaction mechanisms . . . 19
4.4 Writing a name under a molecule . . . 20

5 Advanced usage 22
5.1 Splitting atoms . . . 22
5.2 Formatting atoms . . . 22
5.3 Parameters passed to tikz . . . 23
5.4 Vertical alignment . . . 23
5.4.1 The key baseline . . . 23
5.4.2 Importance of the height or depth of an atom . . . 24
5.4.3 The macro \chemskipalign . . . 25
5.5 Shifted double bonds . . . 26
5.6 Delocalized double bonds . . . 26
5.7 Saving part of a molecule with \definesubmol . . . 26
5.8 The macro \charge . . . 28
5.8.1 Presentation . . . 28
5.8.2 Parameters . . . 28
5.9 Stacking characters . . . 30
5.10 Saving to a file . . . 30
5.11 How to…? . . . 30
5.11.1 Write a colored atom . . . 30
5.11.2 Add a superscript without affecting a bond . . . 31
5.11.3 Draw a curved bond . . . 31
5.11.4 Draw a polymer element . . . 31
5.11.5 Draw the mirror image of a molecule . . . 33
5.11.6 Add text above bonds and mark angles . . . 34
5.11.7 Draw multiple bonds . . . 34
5.12 Annotated examples . . . 35

5.12.1 Ethanal . . . 35
5.12.2 2-amino-4-oxohexanoic acid . . . 35
5.12.3 Glucose . . . 37
5.12.4 Adrenaline . . . 39
5.12.5 Guanine . . . 40

6 Reaction schemes **42**
6.1 Overview . . . 42
6.2 Arrow types . . . 43
6.3 Arrow features . . . 43
6.4 Compound names . . . 44
6.5 Anchors . . . 45
6.6 Compound style . . . 46
6.7 Branches . . . 46
6.8 Subscheme . . . 47
6.9 Optional arguments of arrows . . . 49
6.10 Creating your own arrows . . . 51
6.10.1 A first arrow . . . 51
6.10.2 A curved arrow . . . 52
6.11 The `\merge` command . . . 53
6.12 The + sign . . . 55

7 Horizontal reactions **57**
7.1 Syntax . . . 57
7.2 Placement of compounds . . . 57
7.3 Compound names . . . 58
7.4 Arrow types . . . 58
7.5 Arrow labels . . . 58

8 List of commands **60**

9 Gallery **62**

10 Change history **77**

1 The \chemfig command

To use the "chem*fig*" package, one must execute:

- `\input chemfig.tex` or `\input{chemfig.tex}` when using (pdf)(Xe)(lua)TEX;
- `\usepackage{chemfig}` in the preamble when using (pdf)(Xe)(lua)LATEX.

The packages tikz and simplekv are required and are loaded if this has not already been done.

This package was made possible thanks to the help of Christophe Casseau, who had the idea for it. I thank him for the help he gave me before the code was written, as well as for the tests he carried out.

I would also like to warmly thank Evys Ancede Gallardo, who spontaneously sent me this English translation. Here are a few words from him for the readers of this manual: *Note: this English manual is an unofficial translation of the French manual of version 1.83. The author of* chem*fig no longer provides an English manual; in case of doubt, the French manual is authoritative.*

1.1 New in version 1.84

The changes in version 1.84 are listed below; for earlier changes, see the end of the manual, page 77.

1. The manual is available in English, thanks to Evys Ancede Gallardo
2. A few errors have been corrected in the manual

1.2 Syntax

The macro `\chemfig` is used with this syntax:

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

The optional argument in square brackets specifies the parameter settings that will be used to draw the molecule. These parameters apply *only to the current molecule* and are restored to their previous values once the molecule is drawn. To change parameters for the rest of the document, one must execute

```
\setchemfig{⟨keys⟩=⟨values⟩}
```

The ⟨*molecule code*⟩ contains the instructions to draw the molecule, according to a syntax that will be explained in this document. There is no a priori restriction on the characters accepted in this code:

- all characters of catcode 11 or 12, that is upper- or lowercase letters, digits, mathematical operators (`+ - * / =`), punctuation signs, active or not (`. , ; : ! ? ' ‘ " |`-), parentheses and square brackets;
- more special characters such as "`~`", "`#`"[1] as well as "`^`" and "`_`", which have their usual math mode properties;
- spaces, but these are ignored by default since atoms are typeset in math mode;
- the braces "`{`" and "`}`", which have their normal behavior as group markers or macro argument delimiters;
- macros.

1.3 Parameters of \chemfig

The table below gives all the keys = ⟨*values*⟩ allowed in the optional argument of `\chemfig` or in the mandatory argument of `\setchemfig`. Each of them will be described more precisely later in the manual:

keys	default values	Description
chemfig style	⟨*empty*⟩	style passed to ti*kz*, see p. 23
use atom strut	false	puts the strut of the previous atom in the argument of `\printatom` in order to correct alignment defects between atoms
atom style	⟨*empty*⟩	style of the nodes containing the atoms, see p. 23
bond join	false	if ⟨*true*⟩, joins adjacent bonds in a more aesthetic way, see p. 10

[1] To prevent # from being doubled when the macro `\chemfig` is in the argument of a macro, one can use `\#` or the macro `\CFhash` instead.

keys	default values	Description
`<> join angle`	`100`	if `bond join =` ⟨*true*⟩, angle difference beyond which a chamfer is drawn to join 2 Cram bonds, see p. 10
`-> join angle`	`45`	if `bond join =` ⟨*true*⟩, angle difference below which a single bond is not joined to a Cram bond, see p. 10
`fixed length`	`false`	if ⟨*true*⟩, imposes fixed bond lengths, see p. 8
`cram rectangle`	`false`	if ⟨*true*⟩, draws Cram bonds as rectangles, see p. 6
`cram width`	`1.5ex`	width of the base of the Cram bond triangles, see p. 6
`cram dash width`	`1pt`	width of the dashes of Cram bonds, see p. 6
`cram dash sep`	`2pt`	spacing between the dashes of Cram bonds, see p. 6
`atom sep`	`3em`	spacing between atoms, see p. 7
`bond offset`	`2pt`	spacing between the atom and the bond, see p. 7
`double bond sep`	`2pt`	spacing between the lines of multiple bonds
`angle increment`	`45`	increment of the bond angle, see p. 7
`node style`	⟨*empty*⟩	style of the atoms, see p. 23
`bond style`	⟨*empty*⟩	style of the bonds, see p. 7
`baseline`	`0pt`	dimension or node name to set the vertical position of the molecule, see p. 23
`debug`	`false`	if ⟨*true*⟩, displays the boundaries of the drawn elements and the names of the atoms
`cycle radius coeff`	`0.75`	coefficient of the circle or arc drawn in rings
`stack sep`	`1.5pt`	vertical spacing for the arguments of the macros `\chemabove` and `\chembelow`, see p. 30
`show cntcycle`	`false`	displays the ring numbers
`autoreset cntcycle`	`true`	resets the ring counter to 0 at each execution of `\chemfig`
`gchemname`	`true`	if ⟨*true*⟩, makes the assignment of the greatest depth global for the placement of molecule names
`save`	`false`	if ⟨*true*⟩, saves in a file the ti*k*z instructions generated by the macro `\chemfig`, see p. 30
`save name`	`\jobname -fig*`	specifies the name of the file, see p. 30

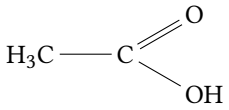
The macro `\resetchemfig` restores all keys to their default values.

2 Atoms and bonds

2.1 Atoms and groups of atoms

Intrinsically, drawing a chemical molecule consists of connecting atoms with lines of different types. Thus, in the molecule O═O, there are 2 groups of atoms, each made up of a single atom “O”.

But in this molecule

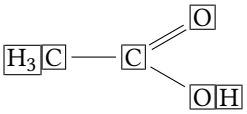


there are 4 groups of atoms: “H_3C”, “C”, “O” and “OH”. For reasons we will see later, chem*fig* examines each group of atoms and splits it into atoms. Each atom extends until an uppercase letter or one of these special characters is encountered:

`-` `=` `~` `(` `!` `*` `<` `>` `@`

For chem*fig*, all characters between braces are ignored when splitting into atoms.

Consequently, the first group of atoms “H_3C” is split into 2 atoms: `H3` and `C`. Chemically, these are sometimes not real atoms since H_3, for example, is made up of 3 hydrogen atoms. By abuse of language, we will nevertheless speak of atoms from now on. Consequently, chem*fig* sees the previous molecule like this:



2.2 Role of the first atom

It is important to understand that the placement of the whole molecule depends on the first atom placed, that is, the first atom of the first group of atoms. For this first atom, the ti*kz* anchor “`base east`” is placed on the baseline of the current line (shown in gray in the examples of this manual).

Influence of the first atom

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

A—B —B A^1—B

To specify an arbitrary vertical shift or to place an atom on the baseline, use the key `baseline` (see page 23).

2.3 Bonds between groups of atoms

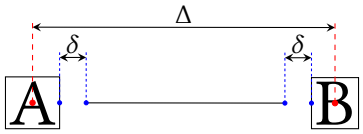
For chem*fig*, there are 9 types of bonds between 2 groups of atoms, corresponding to the characters `-`, `=`, `~`, `>`, `<`, `>:`, `<:`, `>|` and `<|`:

	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 following keys set the appearance of the bonds:

- `double bond sep` = ⟨*dim*⟩ sets the spacing between the lines of double or triple bonds. This spacing is 2pt by default;
- `cram rectangle` = ⟨*boolean*⟩ draws Cram bonds as rectangles when ⟨*true*⟩;
- `cram width` = ⟨*dim*⟩ specifies the width of the base of the Cram bond triangles;
- `cram dash width` = ⟨*dim*⟩ and `cram dash sep` = ⟨*dim*⟩ define the width of the dashes and their spacing in Cram bonds.

When a bond is made between 2 atoms, these atoms are contained in invisible rectangular boxes. The centers of the two rectangles are separated by an adjustable distance Δ, called the "interatomic distance". Moreover, bonds do not exactly connect the boundaries of the rectangles: a distance δ, also adjustable, separates the edge of the rectangles from the start and the end of the bond. To help understanding, the rectangular boxes are made visible in this diagram:



The key `atom sep` = ⟨*dim*⟩ sets the interatomic distance Δ and `bond offset` = ⟨*dim*⟩ sets the spacing δ. If two bonds follow each other, chem*fig* inserts an empty group `{}`, and around this empty group, the spacing δ is zero.

Bond settings

```
Length $\Delta$:
    \chemfig[atom sep=2em]{A-B} and
    \chemfig[atom sep=50pt]{A-B}\par
Length $\delta$:
    \chemfig[bond offset=0pt]{A-B} and
    \chemfig[bond offset=5pt]{A-B}\par
Empty groups: \chemfig{A-B=-=C}
```

Length Δ: A — B and A ———— B
Length δ: A——B and A — B
Empty groups: A —— B ═══——═══ C

The spacing δ can be specified for a single bond with a marker which is, at will, the character `#`, the macro `\#` or `\CFhash`. If the macro `\chemfig` is in the argument of another macro, one *must* use the macro `\#` or `\CFhash` instead of the character `#`.

This marker must be placed *immediately* after the bond sign and has a mandatory argument between parentheses of the form "`(`⟨*dim1*⟩`,`⟨*dim2*⟩`)`", where ⟨*dim1*⟩ is the spacing δ at the start of the bond and ⟨*dim2*⟩ the one at the end. If ⟨*dim2*⟩ is omitted, the spacing at the end of the bond takes the value of δ in force at that moment. In this example one can see how the offset, set to 4pt to be more visible, is made zero for the bond arriving at "B", then for the one leaving "B" and finally for both at once:

Fine tuning of bond offsets

```
\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}
```

A — B — C
A ——B — C
A — B—— C
A ——B—— C

The key `bond style` = ⟨*tikz code*⟩ defines the style of the bonds. The ⟨*tikz code*⟩ is empty by default. To customize bonds one by one, see p. 9.

Bond style

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

A — B ═ C ▷ D ◀ E ⁞ F

By default, all atoms in the groups of atoms are typeset in math mode, so spaces are ignored. They may therefore contain characters specific to this mode, such as subscripts or superscripts.

2.4 Optional argument of a bond

After each bond, an optional argument in square brackets may be written, containing 5 comma-separated fields:

`[`⟨*angle*⟩`,`⟨*length*⟩`,`⟨*departure atom*⟩`,`⟨*arrival atom*⟩`,`⟨*tikz code*⟩`]`

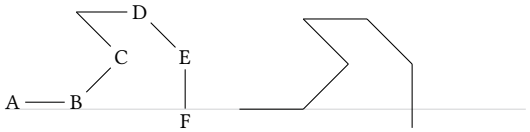
These fields must be written in this order, but may be omitted if, after a field, none of the following ones is specified.

2.4.1 Bond angle: predefined angles

When ⟨*angle*⟩ is an integer, the angle the bond makes with the horizontal is 45° multiplied by this integer:

Predefined angles

```
\chemfig{A-B-[1]C-[3]-[0]D-[7]E-[6]F}
\qquad
\chemfig{--[1]-[3]-[0]-[7]-[6]}
```



The key `angle increment` = ⟨*angle*⟩ allows choosing an angle other than the default 45°:

Changing the predefined angle

```
By default (45): \chemfig{-[1]-[-1]-[1]-[-1]}\par
Angle of 30°: \chemfig[angle increment=30]{-[1]-[-1]-[1]-[-1]}
```

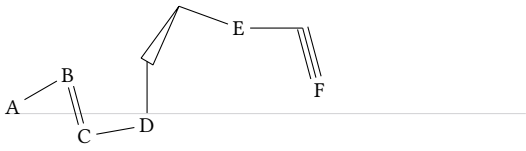


2.4.2 Bond angle: absolute angles

To specify an angle in degrees with respect to the horizontal, the optional argument must take this form: `[:⟨angle⟩]`. If necessary, chem*fig* brings ⟨*angle*⟩ back into the interval [0, 360[:

Absolute angles

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



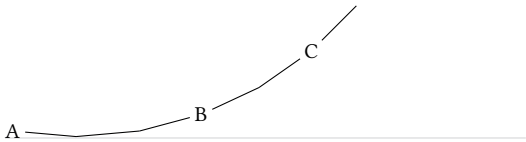
2.4.3 Bond angle: relative angles

To specify the angle a bond makes with the previous one, use the syntax `[::⟨angle⟩]`.

In this example, the bond angles increase by 10° after each one, with a first bond whose absolute angle is −5°:

Sequence of relative angles

```
\chemfig{A-[:-5]-[::+10]-[::10]B-[::+10]-[::10]C-[::10]}
```



A sequence of relative angles can be "broken" whenever desired by giving an absolute or predefined angle. Here, it is at atom "B" that the next bond has an absolute angle of −25°.

Relative angles then an absolute angle

```
\chemfig{
    A-[:-5]-[::10]-[::10]B-[:-25]-[::10]C-[::10]
}
```



2.4.4 Bond length: influence of the size of atoms

The interatomic distance Δ can be set with the key `atom sep`. Once this parameter is fixed, the length of a bond depends on the contents of the atoms and, to a lesser extent, on the angle the bond makes with the horizontal. Two "narrow" atoms will have the edges of their bounding boxes further apart than in the opposite case. The effect on the bond appears in this example where the atoms "I" are narrower than the atoms "M", so that the bond between the "I"s is longer than the one between the "M"s. The effect is even more noticeable — to the point of being undesirable — when an atom has several letters or contains a superscript.

Influence of the width of the atom

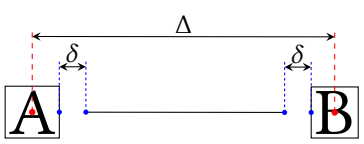
```
\chemfig{I-I}\par
\chemfig{M-M}\par
\chemfig{Mg-Mg}\par
\chemfig{A^{++}-B_2}
```



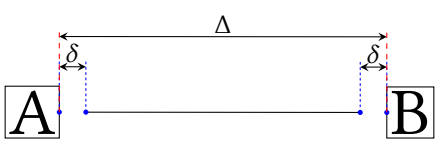
2.4.5 Fixed bond lengths

The behavior of chem*fig* regarding the interatomic space can be changed. Indeed, when the key `fixed length` is set to ⟨*true*⟩, the key `atom sep` no longer specifies the distance between the centers of the atoms, denoted Δ, but *the length of the bonds*. Bonds then have fixed lengths while the distance between the centers of the atoms becomes variable and depends on their size. Here are two diagrams illustrating the boolean values of `fixed length`:

`fixed length` = ⟨*false*⟩

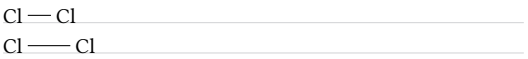


`fixed length` = ⟨*true*⟩



Fixed-length bonds

```
\setchemfig{atom sep=2.5em}
\chemfig{Cl-Cl}\par
\chemfig[fixed length=true]{Cl-Cl}
```



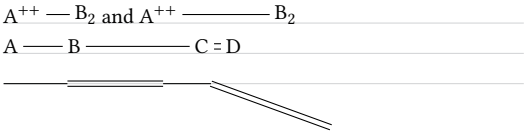
So that rings (see p. 14) are regular polygons, the default behavior is restored for ring bonds, even if `fixed length` = ⟨*true*⟩.

2.4.6 Bond length: setting with the optional argument

The second field of the optional argument of a bond is a decimal coefficient *k* indicating by which value the value ⟨*atom sep*⟩ must be multiplied for this bond.

Length coefficient

```
\chemfig{A^{++}-B_2} and \chemfig{A^{++}-[,2]B_2}\par
\chemfig{A-B-[,2]C=[,0.5]D}\par
\chemfig{-=[,1.5]-[,0.75]=[:-20,2]}
```



2.4.7 Departure and arrival atoms

In a group of atoms — which may contain several —, the atoms are numbered from left to right starting at 1.

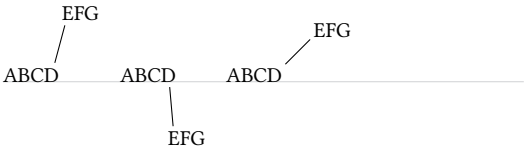
Suppose we want to connect the group "ABCD" to the group "EFG" with a bond. chem*fig* computes which atom of the first group and which atom of the second group must be connected depending on the angle the bond makes with the horizontal:

- if the angle is between −90° and 90° (modulo 360°), these values being *excluded*, 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 by default between the first atom of the first group and the last atom of the second group.

In these cases, the angles are in the interval $]-90°, 90°[$, and the bond is therefore made between D and E:

Atoms connected by default

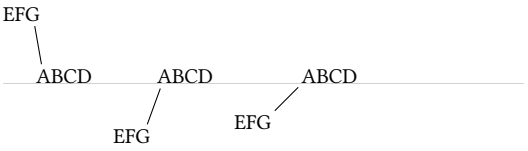
```
\chemfig{ABCD-[:75]EFG}\quad
\chemfig{ABCD-[:-85]EFG}\quad
\chemfig{ABCD-[1]EFG}
```



In these other cases, the angles belong to $[90°, 270°]$ and so the bond is made between A and G:

Atoms connected by default

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



In some cases, one may want a bond to start from atoms other than those computed by chem*fig*. A departure atom or an arrival atom can be forced with the optional argument of the bond. One must write:

`[,,`⟨*n1*⟩`,`⟨*n2*⟩`]`

where ⟨*n1*⟩ and ⟨*n2*⟩ are the numbers of the desired departure and arrival atoms. These atoms must exist, otherwise an error message is issued.

Forced connected atoms

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



2.4.8 Customizing a bond

The 5th field is passed directly to ti*k*z when the bond is drawn. One can put color attributes there such as "`red`", dash attributes such as "`dash pattern=on 2pt off 2pt`", thickness such as "`line width=2pt`", or even decorations if a ti*k*z decoration library has been loaded. A bond can also be made invisible by writing "`draw=none`". To specify several attributes, use the ti*k*z syntax, separating them with commas:

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}
```



The many ti*k*z decoration libraries are available. For example, one can load the library "`pathmorphing`" by writing `\usetikzlibrary{decorations.pathmorphing}` in the preamble to draw wavy bonds:

Wavy bond

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



2.5 Joins between bonds

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

For those who find this "ugly[1]", it is now possible to join single bonds together at the cost of a slightly longer compilation time. The boolean key `bond join` = ⟨*boolean*⟩, when ⟨*true*⟩, enables this feature. The default value is ⟨*false*⟩, which is the preferable behavior.

Join between single bonds

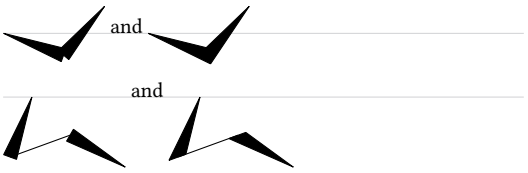
```
\setchemfig{bond style={line width=3pt}}
\chemfig{-[1]-[7]} and
\chemfig[bond join=true]{-[1]-[7]}
```



The problem is even more obvious with plain Cram bonds when they connect to a single bond or to another Cram bond. Here again, setting `bond join` to ⟨*true*⟩ gives more aesthetic joins.

Join with Cram bonds

```
\chemfig{<[:-20]>[:50]} and
\chemfig[bond join]{<[:-20]>[:50]}
\medbreak
\chemfig{<[:-110]-[:20]>[:-30]} and
\chemfig[bond join]{<[:-110]-[:20]>[:-30]}
```



When `bond join` = ⟨*true*⟩, it is important to note that the way two consecutive Cram bonds are joined depends on the angle difference between them. If this angle is too large and becomes greater than the value of the key `<> join angle`, which is 100 by default, a chamfer is drawn instead of a pointed join on the outer side of the join. This avoids too large a spatial extension of the join:

Cram-Cram join

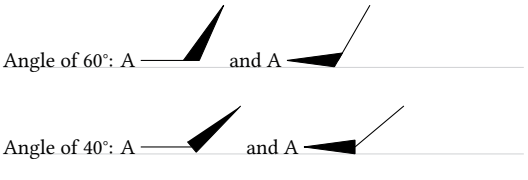
```
Angle of 50°: \chemfig[bond join]{<[:10]>[:60]}
\bigbreak
Angle of 110°: \chemfig[bond join]{<[:20]>[:-90]}
```



Furthermore, the join between a single bond and a Cram bond is only made when the angle difference between these two bonds is greater than or equal to a limit angle contained in the key `-> join angle`, which is 45 by default.

Single-Cram bond join

```
Angle of 60°:
\chemfig[bond join]{A->[:60]} and \chemfig[bond join]{A<-[:60]}
\bigbreak
Angle of 40°:
\chemfig[bond join]{A->[:40]} and \chemfig[bond join]{A<-[:40]}
```



When the angle between a single bond and a plain Cram bond is less than `-> join angle`, it is nevertheless possible to join these 2 bonds. The base of the triangle of the Cram bond must be inside the space delimited by the 2 outer edges of the single bond (assumed thick enough for this to happen). Mathematically, this is the case if $d \cos \alpha > w$ where:

- d is the base of the triangle, which is equal to `cram width` increased by the value of `line width` for the Cram bond;
- w is `line width` for the single bond;
- α is the angle difference between the 2 bonds.

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

In the example below, since $(6 + 0.4)\cos 40° < 5$, the join is made when `bond join` = ⟨*true*⟩:

Single-Cram bond join with a small angle

```
\setchemfig{cram width=6pt,line width=0.4pt}
\chemfig{-[,,,,line width=5pt]>[:40]}\qquad
\chemfig[bond join]{-[,,,,line width=5pt]>[:40]}
\bigbreak
\chemfig{<-[:40,,,,line width=5pt]}\qquad
\chemfig[bond join]{<-[:40,,,,line width=5pt]}
```



2.6 Default values for a molecule

At the beginning of each molecule, the default values of the optional fields of the bonds are initialized. They are:

- 0 for the bond angle;
- 1 for the length multiplication coefficient;
- ⟨*empty*⟩ for the departure and arrival atom numbers, which leaves chem*fig* to compute them according to the bond angle;
- ⟨*empty*⟩ for the parameter passed to ti*k*z.

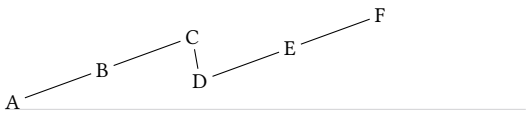
These default values can be changed for the whole molecule by starting the molecule code with

`[`⟨*angle*⟩`,`⟨*coeff*⟩`,`⟨*n1*⟩`,`⟨*n2*⟩`,`⟨*tikz code*⟩`]`

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

Overriding default values

```
\chemfig{[:20,1.5]A-B-C-[:-80,0.7]D-E-F}
```



If one writes an unlikely `[:30,1.5,2,2,red,thick]`, unless otherwise specified, all bonds will have an angle of 30° with the horizontal, the interatomic distances will be 1.5 times the default value, the bonds will start from and arrive at the second atom of each group and will be red and thick:

Default values

```
\chemfig{[:30,1.5,2,2,red,thick]ABC-DEF=GHI}
```



2.7 Branches

2.7.1 Principle

So far, all molecules were linear, which is rare. A molecule can be attached to an atom by following this atom with a ⟨*code*⟩ between parentheses.

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

Branch

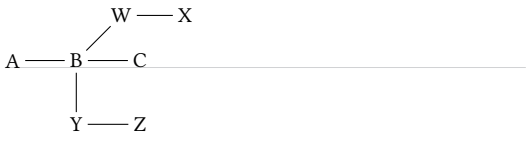
```
\chemfig{A-B(-[1]W-X)-C}
```



Several molecules can be attached to the same atom: simply follow it with several parentheses containing the code of each one:

Multiple branches

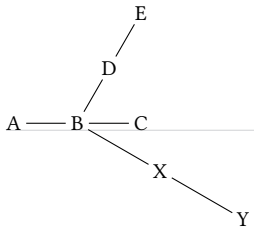
```
\chemfig{A-B(-[1]W-X)(-[6]Y-Z)-C}
```



The code of each attached molecule can define its own default values, which will be valid throughout its scope. Here, we attach to "B" a molecule "`[:60]-D-E`" whose default angle is 60° absolute. We also attach to "B" a submolecule "`[::-30,1.5]-X-<`" whose default angle is 30° less than that of the previous bond (the one going from "A" to "B") and whose interatomic distance is 1.5 times the default one:

Default value in branches

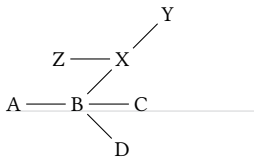
```
\chemfig{A-B([:60]-D-E)([::-30,1.5]-X-Y)-C}
```



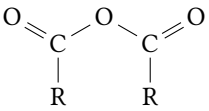
Attached molecules can be nested, and the rules seen in the previous paragraph remain valid:

Nested branches

```
\chemfig{A-B([1]-X([4]-Z)-Y)(-[7]D)-C}
```



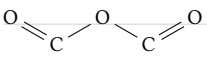
2.7.2 Example: the acid anhydride molecule

Let us try to draw the acid anhydride molecule: 

One method to achieve this is to choose the longest path `O=C-O-C=O`, taking the angles into account and using only relative angles:

Structure of the acid anhydride

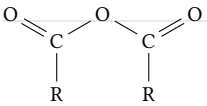
```
\setchemfig{atom sep=2.5em}
\chemfig[atom sep=2.5em]{O=[::-30]C-[::60]O-[::-60]C=[::60]O}
```



Starting from this structure, it suffices to attach two "`-R`" to the carbon atoms, specifying a relative angle of −60°, since the previous bond already has a relative angle of −30°:

Acid anhydride

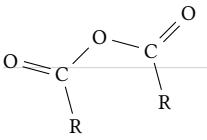
```
\setchemfig{atom sep=2.5em}
\chemfig[atom sep=2.5em]{O=[::-30]C(-[::-60]R)-[::60]O-[::-60]C(-[::-60]R)=[::60]O}
```



Since only relative angles were used, this molecule can be rotated by specifying a default angle of 15°, for example:

Rotating the molecule

```
\setchemfig{atom sep=2.5em}
\chemfig[atom sep=2.5em]{[:15]O=[::-30]C(-[::-60]R)-[::60]O-[::-60]C(-[::-60]R)=[::60]O}
```



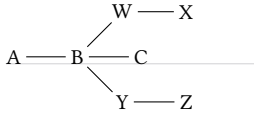
2.8 Connecting distant atoms

We have seen how to connect atoms *that follow each other in the code.* It is sometimes necessary to connect atoms that do not follow each other in the code. Let us call these particular bonds "distant bonds".

In this molecule

Branched structure

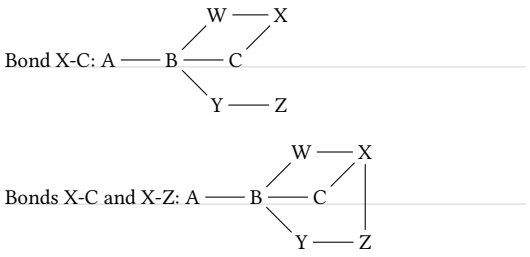
```
\chemfig{A-B(-[1]W-X)(-[7]Y-Z)-C}
```



suppose we want to connect the atoms `X` and `C`. In this case, chem*fig* allows placing a "hook" *immediately* after the atom of interest. The character used to place this hook is "`?`", for its resemblance to a hook. Thus, if one writes `X?`, the atom `X` will carry this hook. Later in the code, all atoms followed by a `?` will be connected to `X`:

Distant bond

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



Now, imagine we have to keep these distant bonds `X-C` and `X->`, while adding another one: `A-W`. We must therefore place two *different* hooks, one on `A` and the other on `X`. In fact, the character "`?`" has an optional argument:

`?[`⟨*name*⟩`,`⟨*bond*⟩`,`⟨*tikz*⟩`]`

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

- the ⟨*name*⟩ is the name of the hook: all alphanumeric characters (`a..z`, `A..>`, `0..9`) are allowed[2]. This name is "`a`" by default. Only this field is taken into account at the first occurrence of the hook bearing this ⟨*name*⟩.
- ⟨*bond*⟩ specifies the bond type.

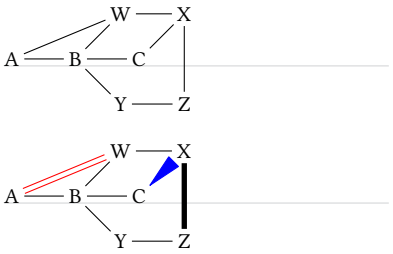
  Two cases are possible; either this field is an integer representing the desired bond type: 1=single bond, 2=double bond, etc. See the table on page 6 for the bond type numbers.

  Or the ⟨*bond*⟩ consists of the characters coding the bond, provided these characters are *between braces*;
- ⟨*tikz*⟩ will be passed directly to ti*k*z as we saw with ordinary bonds.

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

Several distant bonds

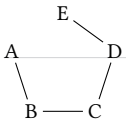
```
\chemfig{A?[a]-B(-[1]W?[a]-X?[b])(-[7]Y-Z?[b])-C?[b]}
\bigbreak
\chemfig{A?[a]-B(-[1]W?[a,2,red]-X?[b])(-[7]Y-
    Z?[b,1,{line width=2pt}])-C?[b,{>},blue]}
```



Several different hooks can be written after an atom. Suppose that in this incomplete pentagon we want to connect `A-E`, `A-C` and `E-C`:

An incomplete ring

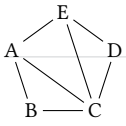
```
\chemfig{A-[:-72]B-C-[:72]D-[:144]E}
```



Then one must proceed this way:

Several distant bonds

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



[2] This is not entirely accurate. In reality, all tokens that can be put between `\csname...\endcsname` are allowed.

3 Rings

The previous example shows how to draw a regular polygon, but the method is tedious since the angles depend on the number of sides of the polygon.

3.1 Syntax

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

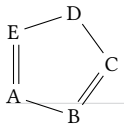
⟨*atom*⟩`*`⟨*n*⟩`(`⟨*code*⟩`)`

⟨*n*⟩ is the number of sides of the polygon and the ⟨*code*⟩ represents the bonds and groups of atoms that make up its vertices and edges. This code *must* begin with a bond since the `atom` is outside the ring.

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

5-ring

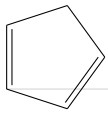
```
\chemfig{A*5(-B=C-D-E=)}
```



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

5-ring with empty groups

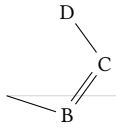
```
\chemfig{*5(-===-)}
```



A ring may be incomplete:

Incomplete 5-ring

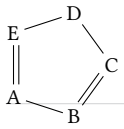
```
\chemfig{*5(-B=C-D)}
```



If the code of a ring contains too many bonds and groups of atoms for the specified number of vertices, all bonds and groups beyond the allowed maximum are ignored:

Truncated 5-ring

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



A circle or an arc of a circle can be drawn inside a ring. For this, use this syntax:

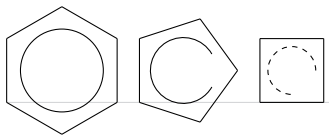
⟨*atom*⟩`**[`⟨*angle 1*⟩`,`⟨*angle 2*⟩`,`⟨*tikz*⟩`]`⟨*n*⟩`(`⟨*code*⟩`)`

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

- ⟨*angle 1*⟩ and ⟨*angle 2*⟩ are the absolute start and end angles of the arc. By default, they are 0° and 360° so that a full circle is drawn by default;
- ⟨*tikz*⟩ is the code that will be passed to ti*k*z to draw the arc.

Rings and arcs

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

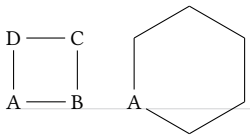


3.2 Angular position

When a ring is at the beginning of a molecule, and as seen in the examples below, the rule is that the attachment atom "A" is always at the south-west of the ring. Moreover, the ring is always drawn counterclockwise and the last bond falls vertically onto the attachment atom:

Angular position of rings

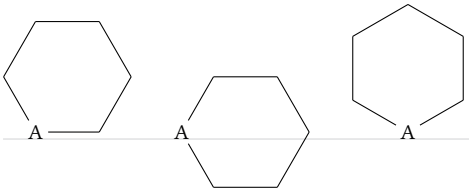
```
\chemfig{A*4(-B-C-D-)}\qquad\chemfig{A*6(------)}
```



If this angular position is not suitable, another default angle can be specified with the optional argument at the beginning of the molecule. Here is a 6-ring rotated by +30°, by −30° and then by +60°:

Rotating rings

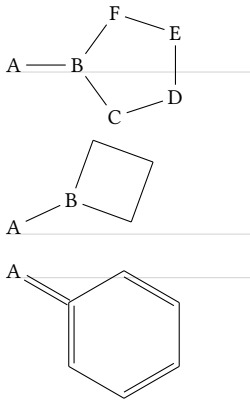
```
\chemfig{[:30]A*6(------)}\qquad
\chemfig{[:-30]A*6(------)}\qquad
\chemfig{[:60]A*6(------)}
```



When a ring does not begin a molecule, the default angular position changes: the ring is drawn so that the bond arriving at the attachment atom is the bisector of the first and last sides of the ring:

Bond arriving at a ring

```
\chemfig{A-B*5(-C-D-E-F-)}\vskip5pt
\chemfig{A-[:25]B*4(----)}\vskip5pt
\chemfig{A=[:-30]*6(=-=-=-)}
```



3.3 Branches leaving a ring

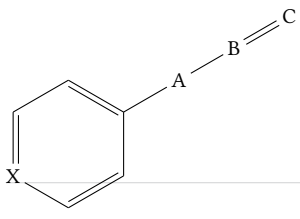
To start a branch from one of the vertices of the ring, use the syntax already seen:

⟨*atom*⟩(⟨*code*⟩)

where the ⟨*code*⟩ is that of the submolecule and the ⟨*atom*⟩ occupies the vertex. A particularity of rings is that the default angle of the submolecule is not 0° but is computed so that it is the bisector of the sides leaving the vertex:

Branch leaving a ring

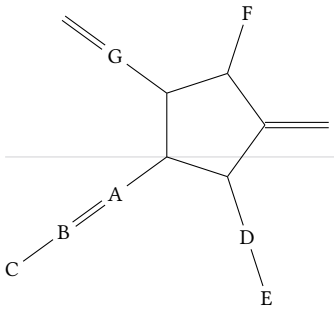
```
\chemfig{X*6(-=-(-A-B=C)=-=-)}
```



A submolecule can start from the first vertex of the ring, as well as from all the other vertices:

Ring and branches

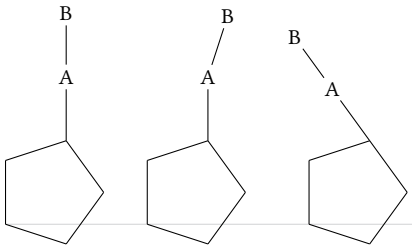
```
\chemfig{*5((-A=B-C)-(-D-E)-(=)-(-F)-(-G=)-)}
```



If the bond leaving a vertex should not be the bisector of its sides, one can use the global optional parameter or the optional parameter of the bond:

Branches and specified angles

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

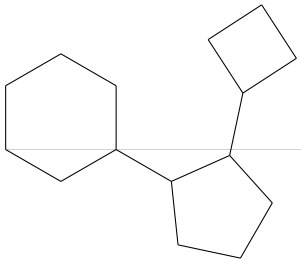


It is interesting to note in the third example that if a relative angle of 0° is specified, the bond is drawn in the continuation of the previous bond of the ring.

Rings can now be connected to each other by bonds:

Connected rings

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

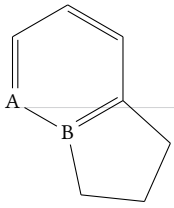


3.4 Fused rings

To “glue” 2 rings together, the syntax is slightly different: one identifies the vertex from which the other ring will start. It suffices to follow this vertex with the usual ring syntax. Here, for example, is a 5-ring starting from the second vertex of a 6-ring:

Nested rings

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

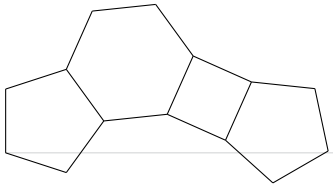


Note that the ring grafted onto the main ring has an angular position such that two of their sides coincide. Moreover, the 5-ring has only 4 bonds. Indeed, the 5th would be useless since it is the second side of the 6-ring, which is already drawn.

It is of course possible to glue several rings together:

Several nested rings

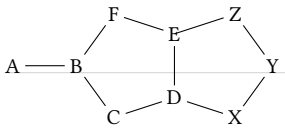
```
\chemfig{*5(--*6(-*4(-*5(----)--)----)---)}
```



Some cases deserve more attention. In this example one can see that the fourth side of the second 5-ring closes on the center of the atom “E”.

Imperfect drawing

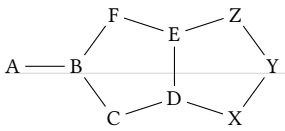
```
\chemfig{A-B*5(-C-D*5(-X-Y-Z-)-E-F-)}
```



This is normal since the second 5-ring (which starts from the atom “D”) is drawn *before* chem*fig* knows about the atom “E”. In this case, two hooks must be used to draw the bond `>-E`:

Distant bond and ring

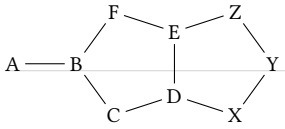
```
\chemfig{A-B*5(-C-D*5(-X-Y-Z?)-E?-F-)}
```



One could also have used 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-)}
```

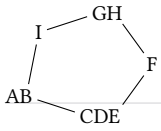


3.5 Groups of atoms in a ring

Care must be taken with rings when one or more vertices consist of several atoms:

Ring and groups of atoms

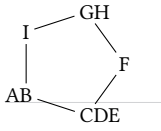
```
\chemfig{AB*5(-CDE-F-GH-I-)}
```



For the ring to have a regular shape, the chem*fig* mechanism that automatically computes the departure and arrival atoms of the bonds must be overridden. Here, `C-F` and `F-G` must be connected by specifying it with the optional argument of these bonds:

Imposed departure and arrival atoms

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



4 Representing electron movements

Electron movements of mesomeric effects or reaction mechanisms can be represented. This is made possible by marking the starting point and the end point of the arrow used to indicate the migration of electrons, with the syntax "`@{⟨argument⟩}`".

This syntax places a node (in the ti*k*z sense) and makes this node accessible outside the argument of the `\chemfig` command thanks to the option "`remember picture`" which is passed to all "`tikzpicture`" environments. This assumes that the viewer supports "picture remembering" and that the compilation is run twice.

There are two possible places to define a node:

- a zero-size node on a bond, using the syntax "`@{⟨name⟩,⟨coeff⟩}`" placed immediately at the beginning of the optional argument of the bond concerned, without being followed by a comma if there is a first optional argument. In this case, the node is named "⟨*name*⟩" and the ⟨*coeff*⟩, between 0 and 1, determines where the node is located on the bond. If the ⟨*coeff*⟩ is omitted, it takes the value 0.5 by default, which means that the node is placed in the middle of the bond;
- a node on an atom, using the syntax "`@{⟨name⟩}`" immediately before the atom concerned. In this case, the node has exactly the same size as the atom; in particular it can be empty and therefore have zero dimensions.

Once the `\chemfig` command has drawn the molecule(s) and placed the nodes with the syntax described above, these nodes can be connected with ti*k*z instructions. These instructions are placed in the argument of the command `\chemmove`[1] and have the following syntax if, for example, a node named "⟨*name1*⟩" must be connected to the node named "⟨*name2*⟩":

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

The optional argument ⟨*opt*⟩ of the `\chemmove` command is added to the argument of the `tikzpicture` environment in which the links between the nodes are drawn. The instructions ⟨*tikz opt*⟩ and ⟨*tikz link*⟩ are exhaustively described in the documentation of the ti*k*z package.

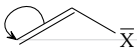
4.1 Mesomeric effect

To fix ideas, let us take the example of a mesomeric effect involving a conjugated double bond and lone pair. Let us start with the possible delocalization of the electrons of 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 macro `\charge` will be described later in the manual, see p. 28.

Mesomeric effect 1

```
\chemfig{@{a1}=_[@{db}::30]-[::-60]\charge{90=\|}{X}}
\chemmove{\draw(db).. controls +(100:5mm) and +(145:5mm).. (a1);}
```



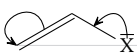
To connect the nodes "db" and "a1", we used the following syntax:

```
\chemmove{\draw(db)..controls +(100:5mm) and +(145:5mm)..(a1);}
```

The default arrow tip style in `\chemmove` is "`CF`". In this example, we use 2 control points[2]. They are located in polar coordinates at 100° and 5 mm from "db" for the first and at 145° and 5 mm from "a1" for the second. This syntax may seem complicated at first reading, but its use reduces in most cases to a simple copy-paste in which only the names of the nodes and the polar coordinates of the control points are modified. This is what we will check right away by adding an arrow from the lone pair (node "`dnl`") to the single bond (node "`sb`").

Mesomeric effect 2

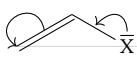
```
\chemfig{@{a1}=_[@{db}::30]-[@{sb}::-60]@{dnl}\charge{90=\|}{X}}
\chemmove{
    \draw(db)..controls +(100:5mm) and +(145:5mm)..(a1);
    \draw(dnl)..controls +(90:4mm) and +(45:4mm)..(sb);}
```



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

Mesomeric effect 3

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



[1] In fact, the command `\chemmove` puts its argument in a "`tikzpicture`" environment with the options "`remember picture,overlay`".

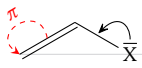
[2] To learn all the ways of connecting two nodes with ti*k*z, read the documentation of this package.

The option “shorten <=3pt” indicates that the starting point of the arrow must be shortened by 3 pt and likewise “shorten >=2pt” indicates that the end point must be shortened by 2 pt.

The full power of ti*k*z instructions can be used to modify the drawing of the arrow. Here, we change the tip of the arrow leaving the double bond to “-stealth”, and draw it dashed, thin and red. We also add the letter π above the arrow at its middle:

Mesomeric effect 4

```
\chemfig{@{a1}=_[@{db}::30]-[@{sb}::-60]@{dnl}\charge{90=\|}{X}}
\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 will see how to indicate the position of the starting or ending anchor. If we write

Starting and ending anchors 1

```
\chemfig{@{x1}\charge{45=\:}{X}}
\hskip2cm\relax
\chemfig{@{x2}\charge{90=\|}{X}}
\chemmove{
    \draw[shorten >=4pt](x1).. controls +(90:1cm) and +(90:1cm).. (x2);}
```



We see that the starting anchor of our arrow does not point correctly at our electrons. The arrow starts from the middle of the upper edge of the node. Indeed, we chose a starting angle of 90° and so ti*k*z starts the arrow from the anchor “x1.90”, which corresponds to the intersection of the half-line starting from the center of the node “x1” and making an angle of 90° with the horizontal, with the edge of the node, which is a rectangle. To have the arrow start where we want, we must specify its position. After some trial and error, it is “x1.57”:

Starting and ending anchors 2

```
\chemfig{@{x1}\charge{45=\:}{X}}
\hskip2cm\relax
\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 coordinates for the control points. Here, we use only one control point located 1 cm horizontally and 1.5 cm vertically from “x1”:

A single control point

```
\chemfig{@{x1}\charge{45=\:}{X}}
\hskip2cm\relax
\chemfig{@{x2}\charge{90=\|}{X}}
\chemmove[shorten <=4pt,shorten >=4pt]{
    \draw(x1.57).. controls +(1cm,.8cm).. (x2.90);}
```



In this case we place a control point 1 cm horizontally and 2 cm vertically from “x1”. All graphic objects drawn through the \chemmove command are overprinted and are not counted in the bounding boxes.

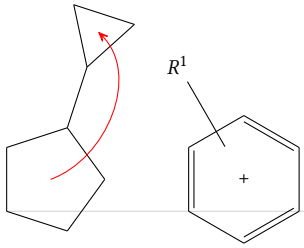
4.2 Center of a ring

The center of a ring becomes addressable via \chemmove because each ring carries at its center a zero-size node whose name is cyclecenter⟨*n*⟩ where ⟨*n*⟩ is the number of the ring, it being understood that rings are numbered from 1 in the order in which they are drawn. The number of each ring can be displayed by setting the boolean key show cntcycle to ⟨*true*⟩.

The boolean key autoreset cntcycle, ⟨*true*⟩ by default, resets the ring counter to 0 at the beginning of each molecule drawn (that is, at each call of \chemfig).

Ring center

```
\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){$R^1$};
    \draw[-,shorten <=.5cm](cyclecenter1)--(end);
}
```



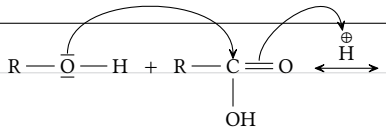
4.3 Reaction mechanisms

Thanks to the option remember picture which is passed to all “tikzpicture” environments, we can easily draw the arrows indicating reaction mechanisms. Let us take as an example the first step of the esterification reaction.

The macros \schemestart, \schemestop, \arrow and \+ will be described later in the manual.

Esterification: step 1

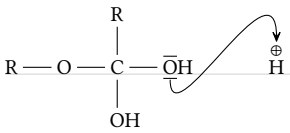
```
\setchemfig{atom sep=7mm}
\schemestart
    \chemfig{R-@{dnl}\charge{90=\|,-90=\|}{O}-H}
    \+
    \chemfig{R-@{atoc}C([6]-OH)=[@{db}]O}
    \arrow(.mid east--){<->[\chemfig{@{atoh}\chemabove{H}{\scriptstyle\oplus}}]}
\schemestop
\chemmove[shorten <=2pt]{
    \draw(dnl)..controls +(90:1cm)and+(north:1cm)..(atoc);
    \draw[shorten >=6pt](db)..controls +(north:5mm)and+(100:1cm)..(atoh);}
```



Using the command `\chemabove{`⟨*code*⟩`}{`⟨*material*⟩`}` does not change the dimensions of the bounding box of the ⟨*code*⟩. For this reason one may have some difficulty pointing at the material indicating the charge (⊕ 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, one can see that the arrow can take more complicated shapes without necessarily overloading the code.

Esterification: step 2

```
\setchemfig{atom sep=7mm}
\chemfig{R-O-C(-[2]R)(-[6]OH)-@{dnl}\charge{90=\|,-90=\|}{O}H}\hskip1cm
\chemfig{@{atoh}\chemabove{H}{\scriptstyle\oplus}}
\chemmove{
    \draw[shorten <=2pt, shorten >=7pt]
        (dnl).. controls +(south:1cm) and +(north:1.5cm).. (atoh);}
```



We leave it to the reader to write the rest.

4.4 Writing a name under a molecule

For convenience, chem*fig* allows writing the name of a molecule under it with the command

`\chemname[`⟨*dimension*⟩`]{\chemfig{`⟨*molecule code*⟩`}}{`⟨*name*⟩`}`

The ⟨*dimension*⟩, which is `1.5ex` by default, is inserted vertically between the baseline of the molecule and the top of the letters of the ⟨*name*⟩. The ⟨*name*⟩ is centered with respect to the molecule, but this ⟨*name*⟩ cannot contain several paragraphs. As seen in this example: H —— O —— H, the ⟨*name*⟩ displayed under the molecule is taken into account in the bounding box,

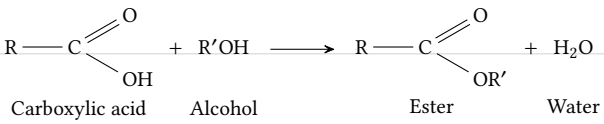
The water molecule: H_2O

but only as far as its *vertical* dimension is concerned. The horizontal size of the ⟨*name*⟩ is always zero.

Here is a reaction with the names under the molecules:

Displaying the names of molecules

```
\schemestart
    \chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic 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{}
```

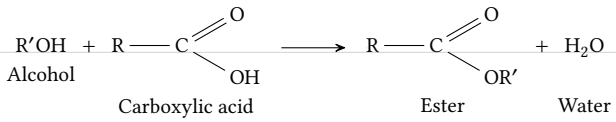


There are some constraints with this command. To become aware of them, suppose we have swapped the acid and the alcohol in the left-hand side:

Alignment of names 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, below the baseline of each molecule (in light gray in the examples of this manual), to write the ⟨*name*⟩, the command `\chemname` inserts 1.5ex + *the greatest of the depths*[3] *of the molecules encountered.*

The macro `\chemnameinit{`⟨*argument*⟩`}` initializes this greatest depth with the contents of its ⟨*argument*⟩. One should therefore:

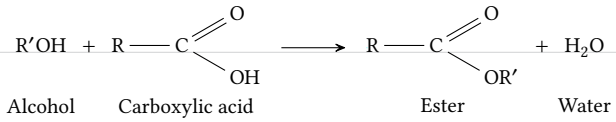
- write `\chemnameinit{`⟨*deepest molecule*⟩`}` before involving this command in a reaction, unless this reaction begins with the deepest molecule;
- write `\chemnameinit{}` after having written all the names of a chemical reaction, to prevent the greatest depth found in this reaction from interfering with a future reaction.

Note that the assignments concerning the greatest depth are global if `gchemname = `⟨*true*⟩, which is the default behavior. Otherwise they are local.

The correct code is therefore to use `\chemnameinit` before and after the reaction:

Alignment of names 2

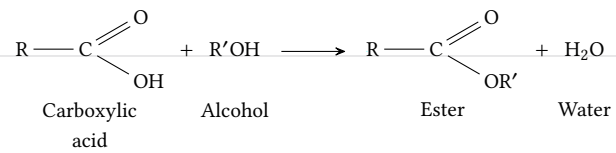
```
\chemnameinit{\chemfig{R-C(-[:-30]OH)=[:30]O}}
\schemestart
    \chemname{\chemfig{R'OH}}{Alcohol}
    \+
    \chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic acid}
    \arrow(.mid east--.mid west)
    \chemname{\chemfig{R-C(-[:-30]OR')=[:30]O}}{Ester}
    \+
    \chemname{\chemfig{H_2O}}{Water}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



Finally, to write a name on several lines, the command `\\` encountered in a ⟨*name*⟩ performs a line break[4]:

Name on 2 lines

```
\schemestart
    \chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Carboxylic\\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 one writes `\chemname*{`⟨*name*⟩`}`, then the macro does not take into account the names encountered previously.

[3] In TEX language, the depth is the dimension extending vertically below the baseline.
[4] On the other hand, the command `par` is forbidden and will cause an error at compilation.

5 Advanced usage

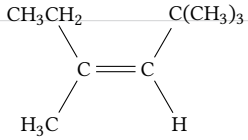
5.1 Splitting atoms

The automatic mechanism for splitting into atoms, already described, makes each atom extend up to the next uppercase letter or one of the characters `-` `=` `~` `(` `!` `*` `<` `>` `@`.

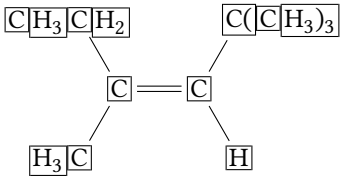
In some cases, this automatic splitting produces incorrect atoms, which can result in an imperfect display. For this molecule, note that the character "`(`" is put between braces so that chem*fig* does not wrongly understand that a branch is being created:

Alkene

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



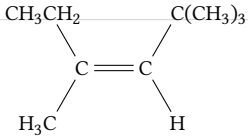
The bond arriving at the carbon atom at the top right is too short. This happens because, if the chem*fig* splitting rules are applied to the group of atoms at the top right, the atoms are split this way: "`C{(}`", "`C`", "`H_3{)}_3`". One then understands that in math mode, the first atom, which contains a parenthesis, has too large a depth, as can be seen by making visible the boxes containing the atoms:



The character "`|`" causes the current atom to be cut where it is encountered. Thus, one can write `C|{(CH_3)_3}` so that chem*fig* splits only 2 atoms here: "`C`" and "`{(CH_3)_3}`". The problem of the too-short bond is thus solved:

Alkene

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



5.2 Formatting atoms

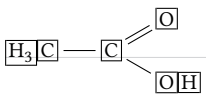
Once the splitting into atoms is done, the macro `\printatom` is called internally by chem*fig* to display each atom. Its only argument consists of the tokens read in the molecule code representing the current atom (for example "`H_3`"). This macro switches to math mode and displays its argument with the math font "`rm`". It is defined by the following code:

- `\def\printatom#1{\ifmmode\rm#1\else$\rm#1$\fi}` when compiling with (pdf)(xe)(lua)TEX;
- `\newcommand*\printatom[1]{\ensuremath{\mathrm{#1}}}` when compiling with (pdf)(xe)(lua)LATEX.

The code of this macro can be modified to customize the display of atoms. In the example below, `\printatom` is redefined so that each atom is contained in a rectangular box, using the OpTEX macro `\frame`:

Redefining \printatom

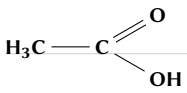
```
\vvkern=1pt \hhkern=1pt % h and v spaces between rules and content
\def\printatom#1{\frame{\ifmmode\rm#1\else$\rm#1$\fi}}
\chemfig{H_3C-C(=[:30]O)(-[:-30]OH)}
```



Here is how to redefine it to use the bold sans serif font of math mode:

Atoms displayed in bold sans serif

```
\def\printatom#1{\ifmmode\bf#1\else$\bf#1$\fi}
\chemfig{H_3C-C(=[:30]O)(-[:-30]OH)}
```



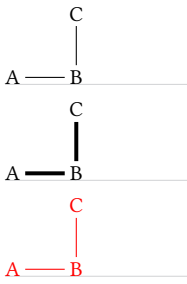
5.3 Parameters passed to ti*k*z

The key `chemfig style` contains ti*k*z instructions that are passed to the `tikzpicture` environment in which the molecule is drawn. Furthermore, the key `atom style` contains ti*k*z instructions that are executed when drawing each node; these instructions are added at the end of `every node/.style{`⟨*argument*⟩`}`, that is, after the following instructions: "`anchor=base,inner sep=0pt,outer sep=0pt,minimum size=0pt`".

Through `chemfig style`, one can choose, for example, the general color or the line width:

Choosing the style

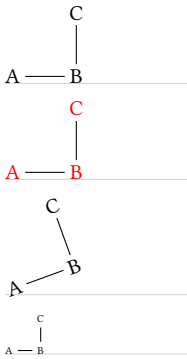
```
\chemfig{A-B-[2]C}\par\medskip
\chemfig[chemfig style={line width=1.5pt}]{A-B-[2]C}\par\medskip
\chemfig[chemfig style=red]{A-B-[2]C}
```



With the key `node style`, one can choose the color of the nodes drawn by ti*k*z, change the slant of the drawing or change the scale:

Choosing styles

```
\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}
```



5.4 Vertical alignment

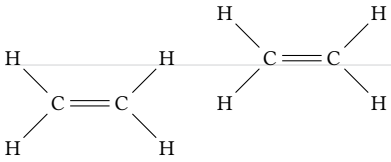
5.4.1 The key `baseline`

The key `baseline` gives fine control of the vertical placement of the molecule with respect to the baseline of the current paragraph.

Its default value is ⟨*0pt*⟩, 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 in the examples of this manual. The choice of this first atom therefore determines the placement of all the others relative to it, and often affects the placement of the whole molecule.

First atom

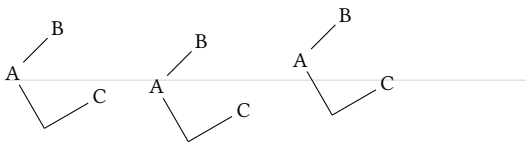
```
\chemfig{H-[7]C(-[5]H)=C(-[1]H)-[7]H}\qquad
\chemfig{C(-[3]H)(-[5]H)=C(-[1]H)-[7]H}
```



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

Vertical shift

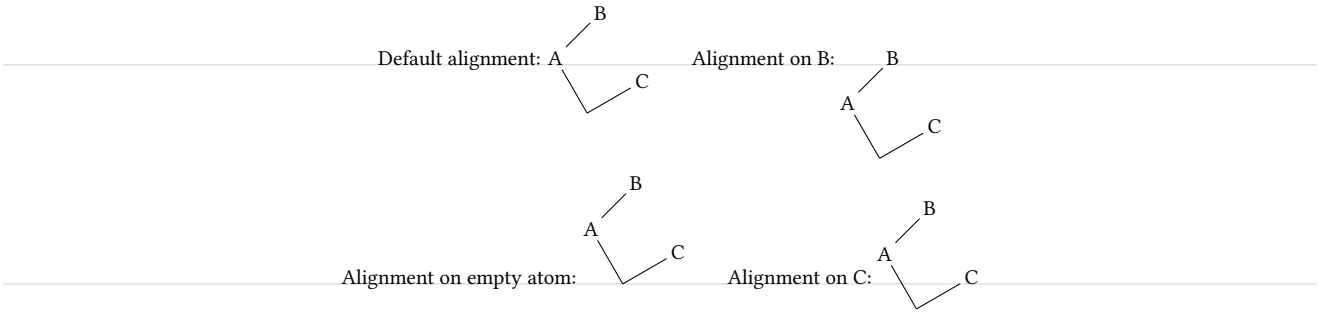
```
\chemfig{A(-[:-60]-[:30]C)-[:45]B}\qquad
\chemfig[baseline=5pt]{A(-[:-60]-[:30]C)-[:45]B}\qquad
\chemfig[baseline=-5pt]{A(-[:-60]-[:30]C)-[:45]B}
```



With the syntax `baseline = ⟨(<name>)⟩` (the name must be between parentheses), one specifies that the baseline of the molecule is on the node named `c`⟨*name*⟩. The name of the atom can be the one automatically assigned by chem*fig* (of the form `n`⟨*a*⟩`-`⟨*b*⟩) or a name given by the user with the syntax `@{`⟨*name*⟩`}` (see page 19).

Alignment on atoms

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

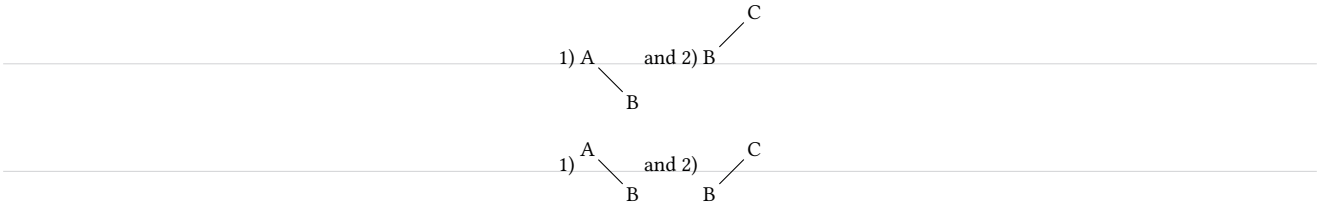


ti*k*z node names can also be used. Thus, to center several molecules vertically on the current baseline, put the value `⟨(current bounding box.center)⟩` in the key `baseline`.

Centered alignment

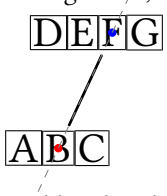
```
1) \chemfig{A-[:-45]B} and 2) \chemfig{B-[:45]C}\bigbreak

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



5.4.2 Importance of the height or depth of an atom

A bond leaving an atom would, if extended, pass through the center of its bounding box. The arrival atom is placed at the end of the bond so that the center of its bounding box is in the continuation of the bond. Consequently, a bond between two atoms passes, when extended, through the centers of their bounding boxes, as illustrated by this example:



This mechanism can create a height shift, particularly visible when bonds are horizontal. Everything goes well when atoms have the same vertical dimensions; on the other hand, it is enough that a departure atom be tall (with a superscript) or deep (with a subscript) and that the arrival atom have a different vertical dimension for a height shift to occur. In this example one can see that the bonds and atoms following `A^1` are slightly shifted upward and those following `E_1` downward.

Horizontal placement

```
\typosize[13/]
\setchemfig{atom sep=2em}
\chemfig{A^1-B-C-D}\qquad \chemfig{E_1-F-G-H}
```



This phenomenon can be highlighted by making the bounding boxes of the atoms visible:

Horizontal placement and bounding boxes

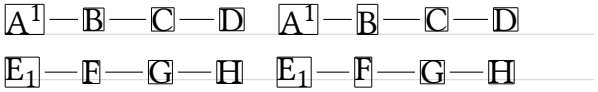
```
\typosize[13/]
\setchemfig{atom sep=2em}
\vvkern=0pt \hhkern=0pt \rulewidth=0.1pt
\def\printatom#1{\frame{\ifmmode\rm#1\else$\rm#1$\fi}}
\chemfig{A^1-B-C-D}\qquad
\chemfig{E_1-F-G-H}
```



The behavior prior to version 1.7 of chem*fig* masked this effect on the first atom following the problematic bounding box by adding to the argument of each `\printatom` the `\vphantom` of the previous atom (without taking the added strut into account). This behavior can be restored by setting the key `use atom strut` to ⟨*true*⟩, but this correction is only effective on a single atom.

Horizontal placement and bounding boxes

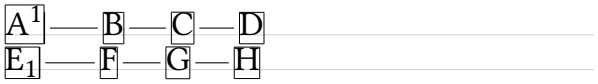
```
\typosize[13/]
\setchemfig{atom sep=2em}
\vvkern=0pt \hhkern=0pt \rulewidth=0.1pt
\def\printatom#1{\frame{\ifmmode\rm#1\else$\rm#1$\fi}}
\chemfig{A^1-B-C-D}\quad
\chemfig[use atom strut=true]{A^1-B-C-D}
\medbreak
\chemfig{E_1-F-G-H}\quad
\chemfig[use atom strut=true]{E_1-F-G-H}
```



Since no automatic solution is satisfactory, this problem can be worked around "with an axe" by redefining `\printatom` to add to each typeset atom the strut defined by `\strut`:

Working around vertical placement

```
\typosize[13/]
\setchemfig{atom sep=2em}
\vvkern=0pt \hhkern=0pt \rulewidth=0.1pt
\def\printatom#1{\frame{\strut\ifmmode\rm#1\else$\rm#1$\fi}}
\chemfig{A^1|\vphantom{X}-B-C-D}\par
\chemfig{E_1|\vphantom{X}-F-G-H}
```

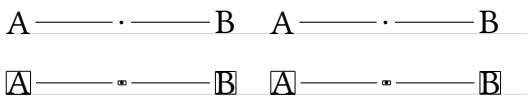


5.4.3 The macro `\chemskipalign`

For any group of atoms, the alignment adjustment mechanism can be temporarily disabled and the `\vphantom` neutralized. To do this, simply place the command `\chemskipalign` in the group of atoms: alignment resumes at the next group of atoms as if the group of atoms containing `\chemskipalign` had not existed. The following example shows the effect of this instruction, which places the reference point of the box containing the first atom at the level of the bond arriving from the left. The boxes enclosing the atoms have been drawn on the second line:

Disabling the alignment mechanism

```
\typosize[13/]
\chemfig{A-.-B}\quad
\chemfig{A-\chemskipalign.-B}\par\bigbreak
\vvkern=0pt \hhkern=0pt \rulewidth=0.1pt
\def\printatom#1{\frame{\ifmmode\rm#1\else$\rm#1$\fi}}
\chemfig{A-.-B}\quad
\chemfig{A-\chemskipalign.-B}
```



This command should be used with care because the alignment of the atoms in the following group of atoms may be disturbed. As a general rule, everything will go well if the group of atoms in which `\chemskipalign` appears contains *a single atom* whose height and depth are *smaller* than those of the preceding and following atoms, and if the preceding and following atoms have equal depth and height. Here, for example, is the mishap that occurs when the group of atoms contains 2 atoms, here "`\chemskipalign.`" and "`B`":

Consequence of the command \chemskipalign

```
\typosize[13/]
\vvkern=0pt \hhkern=0pt \rulewidth=0.1pt
\def\printatom#1{\frame{\ifmmode\rm#1\else$\rm#1$\fi}}
\chemfig{A-\chemskipalign.B-C}
```



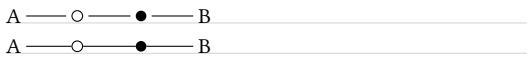
This feature can sometimes prove useful. Suppose we want to draw the molecule



The commands drawing the empty and filled disks can be defined with ti*kz*. So that these disks are at the right height, that is, at the height of the bond arriving at them, we use the command `\chemskipalign`. So that, on the second line of the example below, the bonds "stick" to the disks, we use the possibility of changing the offset of a bond with the character "`#`":

Using \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}
```



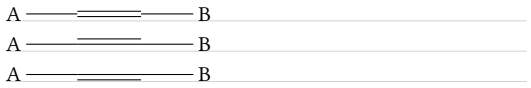
5.5 Shifted double bonds

All double bonds are made of 2 lines drawn on either side of the theoretical line that the single bond would follow. This double bond can be shifted so that one of the two lines is on this theoretical line, the other line then being above or below the bond. In fact, it is more rigorous to say "to the left" or "to the right" of the theoretical line when traveling along the bond in the drawing direction.

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

Shifted double bonds

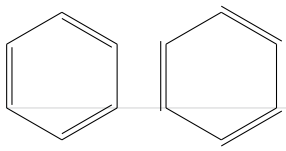
```
\chemfig{A-=-B}\par
\chemfig{A-=^-B}\par
\chemfig{A-=_-B}
```



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

Shifted double bonds and rings

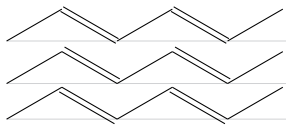
```
\chemfig{*6(-=-=-=)}\qquad
\chemfig{*6(-=_-=_-=_)}
```



Shifted bonds are particularly useful for drawing skeletal formulas of molecules containing carbon chains with double bonds. They give a continuous broken line, whereas this broken line would be discontinuous with normal double bonds:

Shifted bonds and skeletal formulas

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

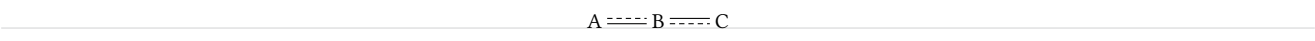


5.6 Delocalized double bonds

It is sometimes necessary to draw a double bond with one line solid and the other dashed. This feature is not hard-coded in chem*fig* since ti*k*z, with its library "`decorations.markings`", 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}
```



5.7 Saving part of a molecule with `\definesubmol`

chem*fig* can save a ⟨*code*⟩ as an alias to reuse it in compact form in the code of a molecule. This is particularly useful when the ⟨*code*⟩ appears several times.

For this, the following command is available:

`\definesubmol{`⟨*name*⟩`}{`⟨*code*⟩`}`

which saves the ⟨*code*⟩ so that it can be called in the code of the molecule with the shortcut "`!{`⟨*name*⟩`}`". This ⟨*name*⟩ can be:

- a sequence of characters: all alphanumeric characters that can be between `\csname` and `\endcsname` are accepted;
- a control sequence.

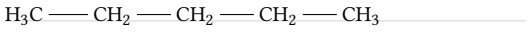
In all cases, if the alias is already defined, overwriting it with a new definition made with `\definesubmol` is not recommended. A warning is issued, telling the user that this alias will be overwritten by the new one. To overwrite the definition of a previously defined alias, use:

`\redefinesubmol{`⟨*name*⟩`}{`⟨*code*⟩`}`

Here is a code that draws the pentane molecule. An alias "`xy`" has been defined beforehand for the code `CH_2`:

Pentane

```
\def\printatom#1{\strut\ifmmode\rm#1\else$\rm#1$\fi}
\definesubmol{xy}{CH_2}
\chemfig{H_3|C-!{xy}-!{xy}-!{xy}-CH_3}
```

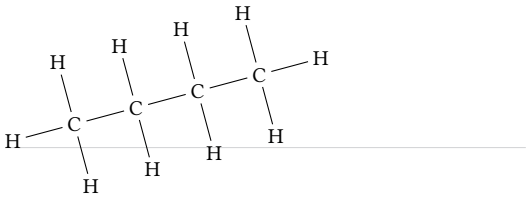


Here, the maneuver is not very interesting since "`!{xy}`" is as long to type as the code it replaces.

But in some cases, this feature saves a lot of space in the molecule code and improves its readability. In the following example, the expanded formula of butane is drawn. For this, an alias with the control sequence "`xx`" is defined for `CH_2`. Since only relative angles are used, the whole molecule can be rotated by any angle through the angle of the global optional parameter, which specifies the default bond angle of the main molecule. Here we use 15°:

Butane

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



The command `\definesubmol` accepts an optional argument and its syntax is as follows:

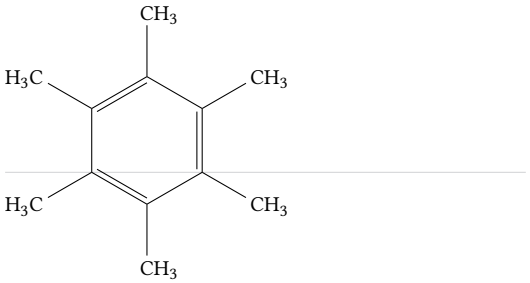
`\definesubmol{`⟨*name*⟩`}[`⟨*code1*⟩`]{`⟨*code2*⟩`}`

When the optional argument is present, the alias "`!`⟨*name*⟩" is replaced by ⟨*code1*⟩ if the bond arriving at the alias comes from the right, that is, if the angle of the incoming bond is between −90° and 90°, these values being excluded. In all other cases, where the bond arrives from the left or vertically, the alias is replaced by ⟨*code2*⟩.

We will define a control sequence `\Me` for "methyl" so that the alias "`!\Me`" is replaced by "`H_3C`" when the bond arrives from the right and by "`CH_3`" when it arrives from the left. With this alias, the example shows that we no longer need to worry about the angle (which in some more complex molecules is not even known):

Dual alias

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



The submolecule saved under a ⟨*name*⟩ takes no argument when called with "`!`". To define a submolecule taking one or more arguments, write this ⟨*number*⟩ of arguments just after the ⟨*name*⟩. The complete syntax of `\definesubmol` is therefore:

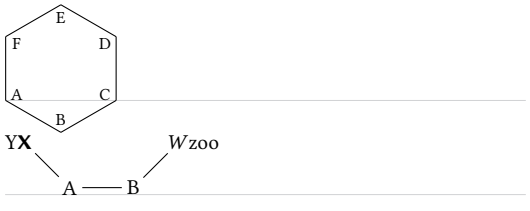
`\definesubmol{`⟨*name*⟩`}`⟨*number*⟩`[`⟨*code1*⟩`]{`⟨*code2*⟩`}`

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

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



Note that if the ⟨*number*⟩ of arguments is incorrect (negative or greater than 9), an error message is issued and chem*fig* considers that the submolecule takes no argument.

Except when the character "`#`" is followed by a digit between 1 and ⟨*number*⟩, in which case it represents an argument, "`#`" characters are allowed in the codes of submolecules.

Using #

```
\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 submolecule `\X`. The other "`#`" are displayed as is in the molecule (case of `#3` and `#4`) or understood as the character specifying the fine tuning of bond offsets.

5.8 The macro `\charge`

5.8.1 Presentation

The macro `\charge`, which requires two mandatory arguments, allows elements — which we will call charges — to be placed around an atom; its syntax is as follows

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

where:

- the ⟨*atom*⟩ a priori consists of one or two letters, but may also be empty;
- the ⟨*charge*⟩ is *arbitrary* content that will be placed around the ⟨*atom*⟩. There are few constraints on this ⟨*charge*⟩: it can be text (in math mode if needed), or even ti*k*z code or a molecule drawn with `\chemfig`;
- the (optional) ⟨*general parameters*⟩ are a list of ⟨*keys*⟩ = ⟨*values*⟩ specifying the options this execution of the macro `\charge` must satisfy. These ⟨*keys*⟩ and ⟨*values*⟩ are described below;
- the ⟨*position*⟩ has the form ⟨*angle*⟩`:`⟨*shift*⟩, but it is possible to specify only the ⟨*angle*⟩, in which case the ⟨*shift*⟩ is taken equal to `0pt`;
- the optional ⟨*tikz code*⟩ contains the options passed to the ti*k*z macro `\node`, which places the ⟨*charge*⟩.

5.8.2 Parameters

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

keys	default values	Description
`debug`	`false`	Boolean which, when equal to ⟨*true*⟩, draws the outlines of the node receiving the ⟨*atom*⟩ (in green), of the one on which the ⟨*charges*⟩ are placed (in blue) and of those receiving the ⟨*charges*⟩ (in red).
`macro atom`	`\printatom`	Macro taking the ⟨*atom*⟩ as argument.
`circle`	`false`	Boolean which, when equal to ⟨*true*⟩, puts the ⟨*atom*⟩ in a circular node; otherwise, the node is rectangular.
`macro charge`	⟨*empty*⟩	Macro (`\printatom` or `\ensuremath` for example) taking each charge as argument.
`extra sep`	`1.5pt`	Increase in the size of the node (circle or rectangle) for the position of the charges: this is the value passed to the ti*k*z key `inner sep`.
`overlay`	`true`	Boolean which, when ⟨*true*⟩, draws the charges "overprinted", that is, outside the final bounding box.
`shortcuts`	`true`	Boolean which, when ⟨*true*⟩, enables the shortcuts "`\.`", "`\:`", "`|`" and "`\"`" to draw Lewis formulas.
`lewisautorot`	`true`	Boolean which, when ⟨*true*⟩, automatically rotates "`\:`" and "`\"`".
`.radius`	`0.15ex`	Radius of the dot used to draw "`\.`" and "`\:`"
`:sep`	`0.3em`	Separation between the two dots of "`\:`"
`.style`	`fill=black`	ti*k*z style used to draw the dots "`\.`" and "`\:`"

keys	default values	Description
"length	1.5ex	Length of the rectangle \" and of the line |
"width	.3ex	Width of the rectangle \"
"style	black,line width=0.4pt	tikz style used to draw the rectangle \"
|style	black,line width=0.4pt	tikz style used to draw the line |

Some (or all) of these parameters can be changed by executing the macro

`\setcharge{`⟨*keys*⟩`=`⟨*values*⟩`}`

and all parameters can be reset to their default values with

`\resetcharge`

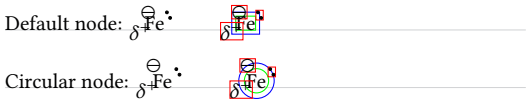
The macro `\charge` places the ⟨*charges*⟩ outside the bounding box (unless otherwise specified in the ⟨*parameters*⟩), whereas `\Charge` places them *inside* the bounding box.

The ⟨*angle*⟩ is the place on the boundary of the node where the ⟨*charge*⟩ is placed. This ⟨*angle*⟩ can be expressed in degrees or be a boundary anchor in the tikz sense, such as "`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 indicated in the ⟨*tikz code*⟩, the placement concerns the *center* of the node containing the charge.

In the following two examples, `debug` is set to ⟨*true*⟩ to better perceive the changes induced by modifying the parameters. Moreover, the macro `\Charge` is used so that the bounding boxes take the charges into account. Here one can see the influence of the shape of the node on the placement of the charges:

Generic example

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



To move the charges ⊖ and δ^+ further away, the ⟨*shift*⟩ can be used

Shifting charges

```
Without shift:
  \Charge{30=\:,120=$\ominus$,210=$\delta^+$}{Fe}
\medbreak
With shift:
  \Charge{[circle]30:3pt=\:,120:5pt=$\ominus$,210:5pt=$\delta^+$}{Fe}
```



The anchor can also be adjusted: the ⟨*angle*⟩ where the ⟨*charge*⟩ is placed is stored in the macro `\chargeangle`; it is therefore wise to choose the anchor equal to `180+\chargeangle`. It is also possible to specify a circular node in which to place the ⟨*charge*⟩.

Fine positioning

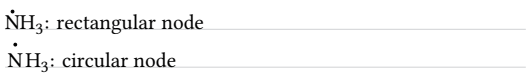
```
\Charge{30=\:,120:3pt=$\ominus$,210:5pt=$\delta^+$}{Fe}\qquad
\setcharge{debug}
\Charge{30=\:,120:3pt=$\ominus$,210:5pt=$\delta^+$}{Fe}
\setcharge{debug=false}
\medbreak
\Charge{[circle]30=\:,
        120[circle,anchor=180+\chargeangle]=$\ominus$,
        210[anchor=180+\chargeangle]=$\delta^+$}{Fe}%
\setcharge{debug}\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 sizes that are *sometimes very different* from "classic" rectangular nodes, especially regarding horizontal and vertical extent. The boolean key `circle` should therefore be set to ⟨*true*⟩ knowingly.

Circular nodes

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



5.9 Stacking characters

The macros

`\chemabove[⟨dim⟩]{⟨code⟩}{⟨material⟩}`

and

`\chembelow[⟨dim⟩]{⟨code⟩}{⟨material⟩}`

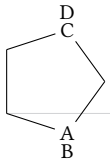
place the ⟨*material*⟩ respectively above and below the ⟨*code*⟩ at a vertical distance ⟨*dim*⟩, without changing the bounding box of the ⟨*code*⟩. The optional argument allows specifying this dimension at each call if desired. If the optional argument is not used, a default dimension is taken: it is ⟨*1.5pt*⟩ but can be changed with the key `stack sep` = ⟨*dim*⟩.

These commands are independent of the macro `\chemfig` and can be called inside or outside its argument.

They can be used in particular in rings, taking care to put braces around the letters A, B, C and D to prevent chem*fig* from stopping the reading of the atom at these letters:

Stacking in rings

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

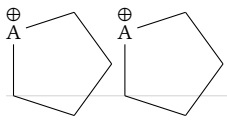


The commands `\Chemabove` and `\Chembelow` work the same way except that the bounding box *takes into account* the ⟨*material*⟩ placed above or below.

What is the difference between `\chemabove` and `\charge` when it comes to placing content above or below another?

\chemabove or \charge

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



By default, both macros give very similar results. There are however differences in their use: the macro `\charge` requires the ti*k*z package, whereas `\chemabove` and `\chembelow` are coded with low-level TEX primitives and are therefore *fast* and independent of any package.

5.10 Saving to a file

The boolean key `save`, which is ⟨*false*⟩ by default, allows, when it is ⟨*true*⟩, saving to a file the ti*k*z instructions[1] generated by `\chemfig{`⟨*code*⟩`}`.

The key `save name` specifies the name of the file (the extension will always be `tex`). Its default value is

`\jobname-fig*`

where the character "`*`", if used, is replaced by a number that starts at 1 and is incremented each time the macro `\chemfig` is called with `save` = ⟨*true*⟩.

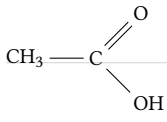
5.11 How to…?

5.11.1 Write a colored atom

Since the package `xcolor` is loaded by ti*k*z, itself loaded by chem*fig*, color instructions such as `\Red`, `\Blue`, etc. can be put in the code of a molecule[2]. Since atoms are displayed in ti*k*z nodes, which behave like TEX boxes, everything happens as if these atoms were confined in a group, so the color change remains local to the atom.

Colors

```
\chemfig{C\Blue H_3-C(=[1]O)-[7]O\Red H}
```



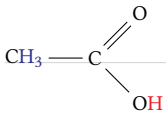
One can see that this code does not work, again because of the rule used by chem*fig*: here, the first atom of the molecule starts at "`C`" and extends to the next uppercase letter. This atom is therefore "`C\Blue`" and so the color change happens at the end of the atom and has no effect. One must force chem*fig* to cut the first atom just after "`C`" with the character "`|`" and then enclose `\Blue H_3` between braces so that chem*fig* does not stop the second atom before the "`H`":

[1] The macros `\chemabove`, `\chembelow` and `\charge` are not translated into ti*k*z instructions and are left as they are.

[2] For those using LATEX, these will be the macros `\color` and `\textcolor`.

Colors

```
\chemfig{C|{\Blue H_3}-C(=[1]O)-[7]O|{\Red H}}
```



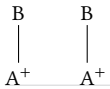
The main drawback is that the operation must be repeated for each atom to be colored, even if they are contiguous.

5.11.2 Add a superscript without affecting a bond

Adding a ⟨*charge*⟩ as a mathematical superscript to an atom means that the box (and therefore the ti*k*z node) containing it will have its dimensions changed. This hardly matters if the atom is at the end of a chain, but the aesthetics can be compromised if a bond must leave from it. The first reflex is to put the superscript in a zero-size TEX box and thus to use the macro `\rlap`[3], which often gives satisfactory results. Here one can see that thanks to `\rlap`, the horizontal alignment of the atoms is preserved:

Charge and bond

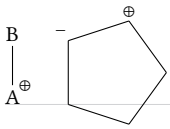
```
\chemfig{A^+-[2]B}
\qquad
\chemfig{A\rlap{${}^+$}-[2]B}
```



The macro `\charge` performs this task simply and precisely.

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=$\scriptstyle-$}{}-)}
```



5.11.3 Draw a curved bond

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

Wavy bond

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

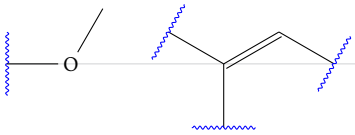
\chemfig{A-[,3,,,decorate,decoration={snake,amplitude=1.5mm,
    segment length=2.5mm}]B}
```



It is also possible to define a submolecule "wb" to draw a wavy line 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}}
```

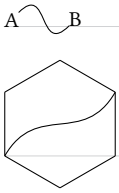


For more flexibility, one can also place nodes with the marker "`@`" and reuse these nodes after the molecule has been drawn to connect them with a curved line using `\chemmove`:

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);}
```



5.11.4 Draw a polymer element

The macro `\polymerdelim` has the following syntax:

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

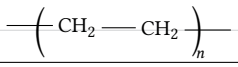
The effect, possibly after *two* compilations, is to place vertical delimiters at the specified nodes: by default, the first delimiter is placed on the first node and the second on the same horizontal line as the first, but at the abscissa of the second. The parameters are specified via keys and values, whose list, default values and actions are given here.

[3] If the charge must be put to the left of the atom, use the command `\llap`.

keys	default values	Description
delimiters	()	Defines the delimiters. If these delimiters are square brackets, one must write delimiters={[]}.
height	10pt	Defines the height (above the node) of the delimiters.
depth	⟨*empty*⟩	Defines the depth (below the node) of the delimiters. If the value is empty, 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 equal to ⟨*true*⟩ and when h align = ⟨*false*⟩, automatically rotates the delimiters so that they are perpendicular to the line connecting the two nodes.
rotate	0	When h align = ⟨*false*⟩ and auto rotate = ⟨*false*⟩, specifies the rotation angle of both delimiters.
open xshift	0pt	Defines the horizontal shift of the opening delimiter.
close xshift	⟨*empty*⟩	Defines the horizontal shift of the closing delimiter. If the value is empty, this shift is the opposite of the shift of the opening delimiter.
indice	n	Defines the subscript placed at the bottom right of the closing delimiter.

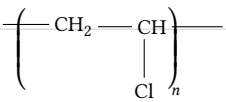
Polyethylene

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



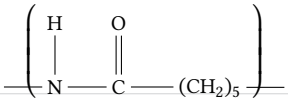
Polyvinyl chloride

```
\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}
```



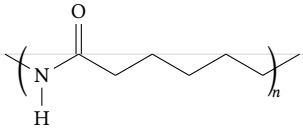
Nylon 6

```
\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}
```



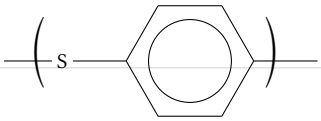
Polycaprolactam

```
\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}
```



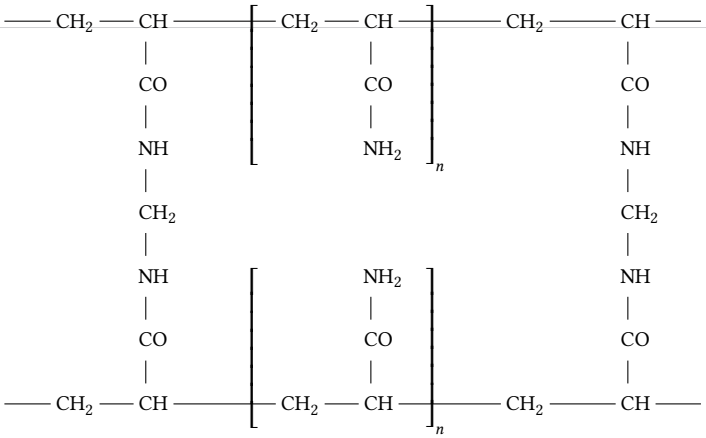
Polyphenylene sulfide

```
\chemfig{\vphantom{S}-[@{op,.75}]S-(**6(---(-[@{cl,0.25}])---))}
\polymerdelim[delimiters = (), height = 15pt, indice = {}]{op}{cl}
```



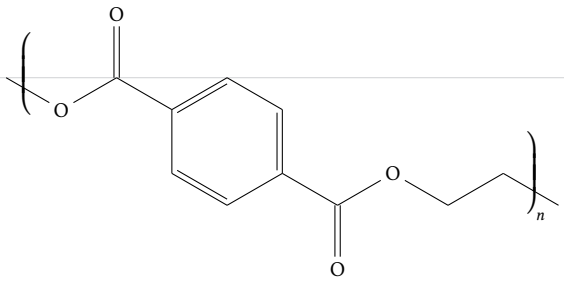
Polymer

```
\def\printatom#1{\strut\ifmmode\rm#1\else$\rm#1$\fi}
\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 = 55pt, indice = n]{upleft}  {upright}
\polymerdelim[delimiters ={[]}, height = 50pt, depth = 10pt, indice = n]{downleft}{downright}
```



Polymer

```
\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}
```

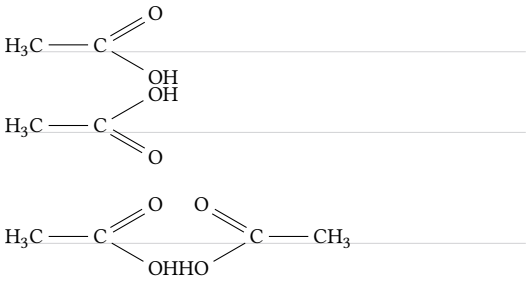


5.11.5 Draw the mirror image of a molecule

The two commands `\hflipnext` and `\vflipnext` draw the mirror image of the next molecule with respect to a horizontal or vertical axis. To build the mirror image of several molecules, the commands must be written before each molecule concerned.

Mirror images

```
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}\par% original
\vflipnext% draws the vertical mirror image of:
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}
\bigbreak
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}% original
\hflipnext% draws the horizontal mirror image of:
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}
```



5.11.6 Add text above bonds and mark angles

Once one has understood that the character "@" places a "global" ti*k*z node, that is, accessible after the molecule has been drawn, everything that is possible with ti*k*z and nodes (that is, many things) becomes conceivable.

To write something above or below a bond, one can therefore place "global" nodes on the atoms at the ends of this bond and write, halfway between these two nodes, a text that is raised or lowered so that it is just above or below the bond. This is what the command `\namebond` does in the code below.

For angles between two bonds leaving an atom, 3 atoms are involved and a global node must be placed on each of them. The angle between two lines leaving the same node can easily be computed; this is what the command `\arcbetweennodes` does. Then the macro `\arclabel` draws an arc between two bonds and writes a text next to this arc: the optional argument of this macro is the code passed to ti*k*z. Argument 2 is the radius of the arc, while the next three arguments are the names of the global nodes between which the arc must be drawn, the middle name being that of the vertex of the angle. Finally, the last argument is the text to write.

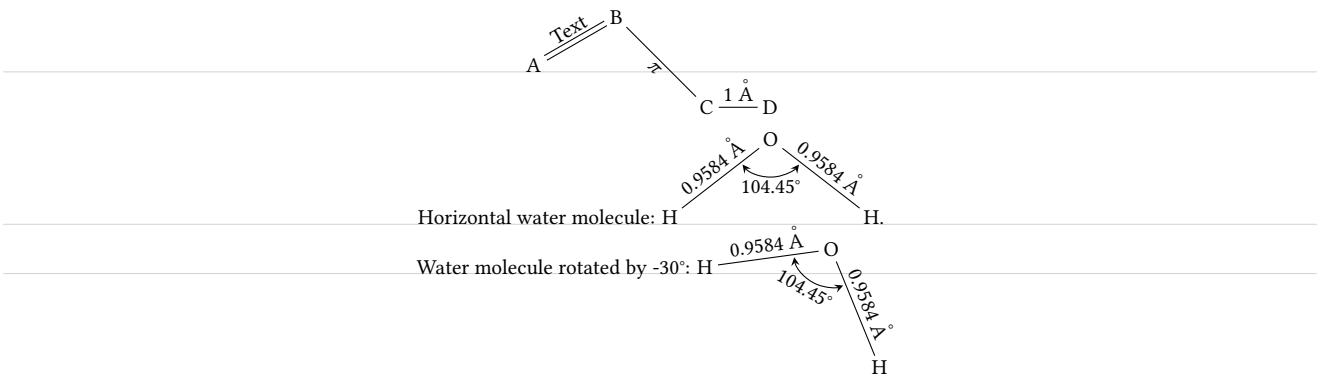
Angles and text on bonds

```
\def\angstrom{\charge{90:-1pt=°}{A}}
\def\namebond#1#2#3#4{\chemmove{\path(#2)--(#3)node[midway,sloped,yshift=#1]{#4};}}

\def\arcbetweennodes#1#2#3{%
  \pgfmathanglebetweenpoints{\pgfpointanchor{#1}{center}}{\pgfpointanchor{#2}{center}}%
  \let#3\pgfmathresult}

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

\chemfig{@{a}A=[:30,1.5]@{b}B-[7,2]@{c}C-@{d}D}
\namebond{5pt}{a}{b}{ Text}
\namebond{-3.5pt}{b}{c}{$\pi$}
\namebond{5pt}{c}{d}{1 \angstrom}
\medbreak
Horizontal water molecule: \chemfig{@{1}H-[::37.775,2]@{2}O-[::-75.55,2]@{3}H}.
\namebond{5pt}{1}{2}{0.9584 \angstrom}
\namebond{5pt}{2}{3}{0.9584 \angstrom}
\arclabel{0.5cm}{1}{2}{3}{104.45°}
\medbreak
Water molecule rotated by -30°: \chemfig{[:-30]@1H-[::37.775,2]@2O-[::-75.55,2]@3H}
\namebond{5pt}12{0.9584 \angstrom}
\namebond{5pt}23{0.9584 \angstrom}
\arclabel{0.5cm}{1}{2}{3}{104.45°}
```

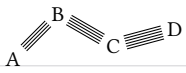


5.11.7 Draw multiple bonds

Here again, the library "`decorations.markings`" makes it possible to draw multiple bonds:

Multiple bonds

```
\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\ii in{0,1,...,\number\numexpr#1-1}{%
            \pgfmathsetmacro\CF_nbondcoeff{\ii-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-[:15,,,,nbond=6]D}
```

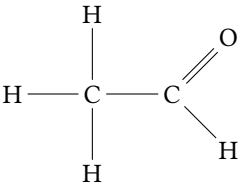


5.12 Annotated examples

In this chapter, several molecules will be drawn using the methods previously presented. The aim here is to show in what logical order a molecule can be built, so that the user unfamiliar with chem*fig* acquires a method for building complex molecules. To help, the construction steps will be shown.

Moreover, it will be shown that several possibilities are available to the user for the same graphic result, some intuitive and others much less so, the point being to show that chem*fig* allows a certain flexibility in coding molecules. It is then up to each user to build and adopt the methods they are most comfortable with.

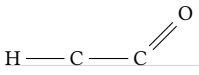
5.12.1 Ethanal

Here we will draw the ethanal molecule: 

The best method for non-cyclic molecules is to choose the longest chain. Here one can take "`H-C-C=O`", for example. The bond `C=O` must be tilted by 45° using the predefined angle "`[1]`". This gives the "structure" of the molecule, to which it will suffice to add the branches:

Structure of ethanal

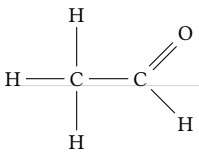
```
\chemfig{H-C-C=[1]O}
```



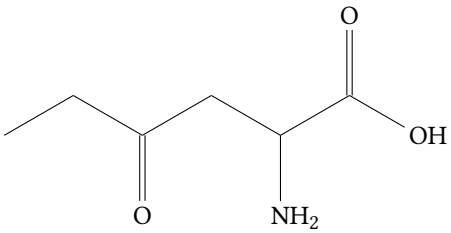
It remains to place 3 hydrogen atoms with the right tilts using predefined angles. The first at 90° with the branch "`(-[2]H)`", the second at 270° with "`(-[6]H)`" and the one on the right at 315° with "`(-[7]H)`":

Ethanal

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



5.12.2 2-amino-4-oxohexanoic acid

Here is the molecule to draw: 

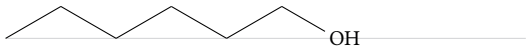
There are — as is often the case for most molecules — several methods, and for each, several different ways of reaching the result. Here, we will examine 4 different methods.

Method with absolute angles

We first draw the middle chain with absolute angles. The default angle is set to +30° with the optional argument, so only the "descending" bonds need their absolute angle of −30° to be specified:

Structure (absolute angles)

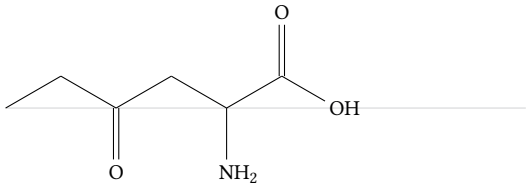
```
\chemfig{[:30]--[:-30]--[:-30]--[:-30]OH}
```



It remains to add the branches "`(=[6]O)`", "`(-[6]NH_2)`" and "`(=[2]O)`" on the right vertices:

Molecule (absolute angles)

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

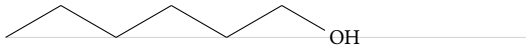


Method with relative angles

A more general approach would have been to use only relative angles. One would have proceeded this way:

Structure (relative angles)

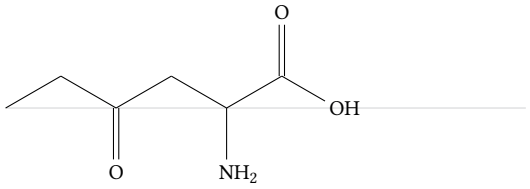
```
\chemfig{[:30]--[::-60]--[::-60]--[::-60]OH}
```



then

Molecule (relative angles)

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



Method with a ring

Since the angles between the bonds are 120°, one may think of using a 6-ring, although this method is less natural. Here one must take advantage of the fact that a ring can be incomplete. It is also necessary to rotate the ring by 120° so that the first vertex is at the south-east of the ring:

Structure

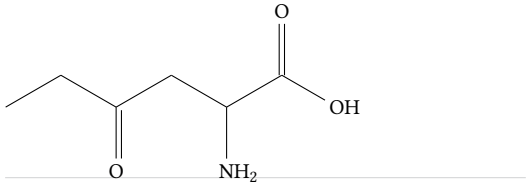
```
\chemfig{[:120]NH_2*6(---=O)}
```



The branches must now start from the appropriate vertices:

Molecule (ring)

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

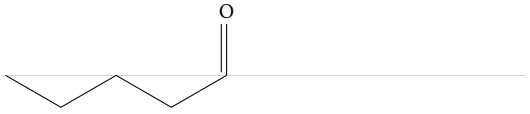


Method with fused rings

Going further with the ring method, one can also think of nesting incomplete 6-rings. One could start from this structure:

Structure

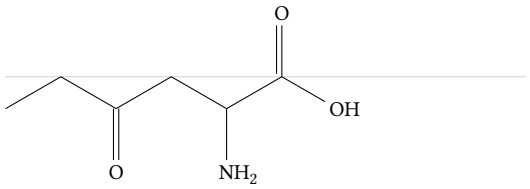
```
\chemfig{*6(--*6(--=O))}
```



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

Molecule (nested rings)

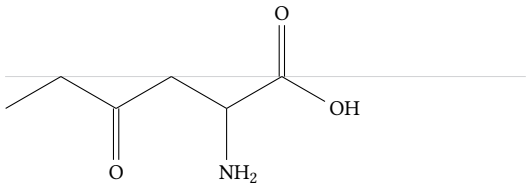
```
\chemfig{*6((-)-(=O)-*6(-(-NH_2)-(-OH)=O))}
```



A careful examination reveals, however, that the second double bond to the oxygen atom is *inside* the incomplete 6-ring[4]. Despite its conciseness, this code therefore does not lead to a perfect drawing. This defect can of course be corrected by lengthening the code a little:

Molecule (corrected nested rings)

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



5.12.3 Glucose

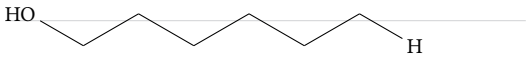
The aim here is to represent the glucose molecule according to different conventions.

Skeletal formula

The code here resembles that of 2-amino-4-oxohexanoic acid. Almost the same structure is obtained with absolute angles, except that here the default angle is −30°:

Structure

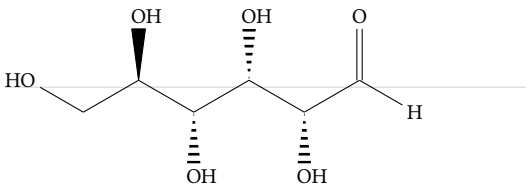
```
\chemfig{[:-30]HO--[:30]--[:30]--[:30]-H}
```



And adding the branches poses no particular problem. Absolute angles are used:

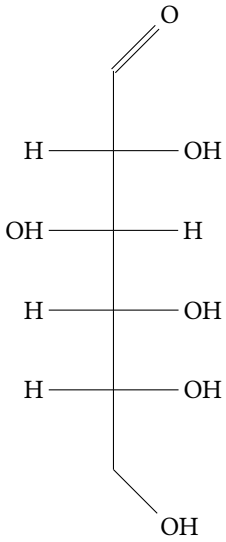
Fructose, skeletal formula

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



Fischer projection

The aim is to obtain the molecule below:

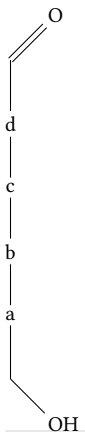


The idea is to start by drawing the longest vertical chain, specifying a default angle "`[2]`". Here is the structure, where lowercase letters are deliberately put at the end of each vertical bond:

[4] This was also the case for the previous method with a single ring.

Structure

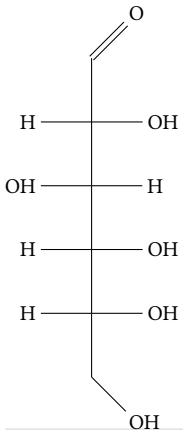
```
\chemfig{[2]OH-[3]-a-b-c-d-=[1]O}
```



Then two aliases are defined for the horizontal bonds and the atoms that will end them. Let us take “x”, which we will put in place of the lowercase letters a, b and d, and “y”, which will take the place of the letter c. Since the aliases consist of a single character, braces can be omitted and one can write “!x” instead of “!{x}”:

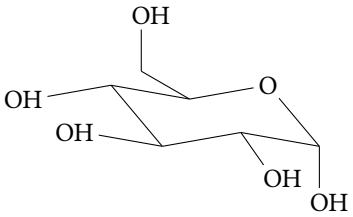
Glucose (Fischer projection)

```
\definesubmol{x}{(-[4]H)(-[0]OH)}
\definesubmol{y}{(-[0]H)(-[4]OH)}
\chemfig{[2]OH-[3]-!x-!x-!y-!x-=[1]O}
```



“Chair” representation

We will represent this molecule of α-D-glucose:



For this, we first draw 5 sides of the chair and connect the first vertex to the last with a hook “?”. The following absolute angles are adopted, given in counterclockwise order: −50°, 10°, −10°, 130°, 190°.

Structure

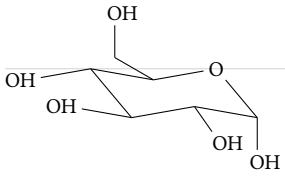
```
\chemfig{?-[:-50]-[:10]-[:-10]-[:130]O-[:190]?}
```



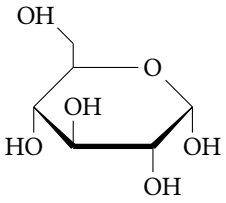
Now it suffices to add the branches between parentheses. The angles are chosen as well as possible to give an impression of perspective, and some bonds are shortened by a coefficient of 0.7:

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)}
```



Haworth projection

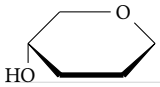
The aim is to represent this molecule of D-glucopyranose: 

First, we choose the longest chain, which starts from the "HO" group on the left and continues over 5 sides of the ring. The ring will be closed with a hook. For the vertical bond leaving the first "HO" group, it must be specified with the optional argument that it starts from the second atom. Moreover, it will be shortened by a coefficient of 0.5. Its optional argument will therefore be "`[2,0.5,2]`".

Then, to give an impression of perspective to the ring, the tilted bonds are reduced by a coefficient of 0.7. For the tilted bold lines, Cram bonds are used, with the width of the base of the triangles redefined to 2pt. The horizontal bold bond must be drawn with a width of 2pt and its optional argument will therefore be "`[0,,,,line width=2pt]`". Here is the structure of the molecule:

Structure

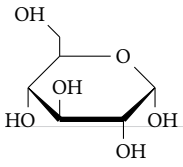
```
\chemfig[cram width=2pt]{HO-[2,0.5,2]?<[7,0.7]-[,,,,
line width=2pt]>[1,0.7]-[3,0.7]O-[4]?}
```



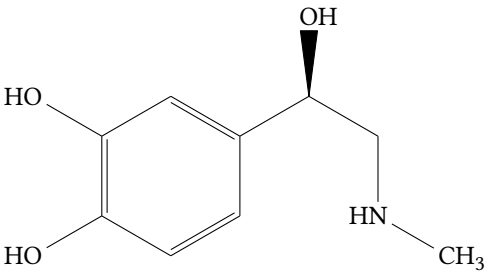
All that remains is to add the branches, in the right places, specifying well-chosen absolute angles and sometimes reduced lengths to better give the illusion of perspective:

Haworth projection

```
\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)}
```



5.12.4 Adrenaline

We want to draw the adrenaline molecule: 

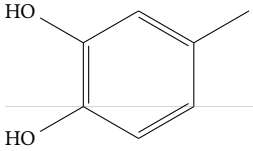
We will use two different methods.

Using one ring

First, we start from the 6-ring and draw the beginning of the branches leaving it:

Structure of adrenaline

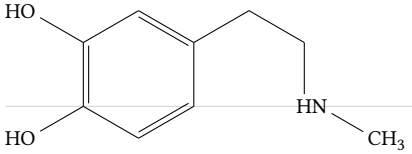
```
\chemfig{*6((-HO)-=-(-)=-(-HO)=)}
```



The branch on the right must now be completed, using, for example, relative angles:

Adrenaline, step 2

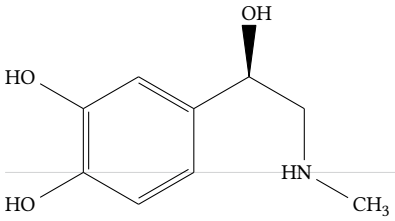
```
\chemfig{*6((-HO)-=-(--[::-60]-[::-60]
HN-[::+60]CH_3)=-(-HO)=)}
```



Then the Cram bond to `OH` must be added, and it must be specified that the bond arriving at "`HN`" does so on the second atom "N". We use the 4th optional argument of the bond:

Adrenaline

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



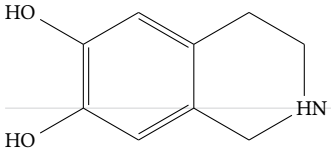
Using 2 rings

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

This code could be improved by considering that the drawing of the adrenaline molecule consists of two 6-rings fused together:

Adrenaline, 2-ring structure

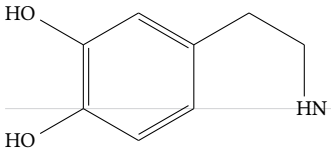
```
\chemfig{*6((-HO)-=*6(--HN---)-=-(-HO)=)}
```



The first two bonds of the right ring must therefore be made invisible. For this, we use the argument passed to ti*kz*, specifying "`draw=none`". These bonds therefore have this code: "`-[,,,,draw=none]`". To keep the code readable, an alias named "`&`" is defined for these bonds:

Adrenaline, step 2

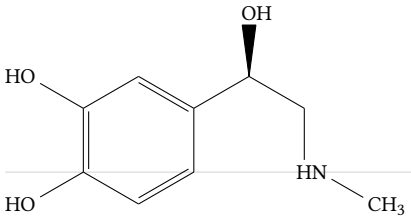
```
\definesubmol{&}{-[,,,,draw=none]}
\chemfig{*6((-HO)-=*6(!&!&HN---)-=-(-HO)=)}
```



The rest becomes easy: it suffices to add the branches at the right vertices:

Adrenaline, step 3

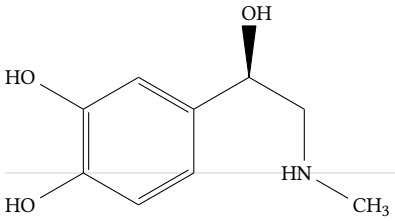
```
\definesubmol{&}{-[,,,,draw=none]}
\chemfig{*6((-HO)-=*6(!&!&HN(-CH_3)--(<OH)-)-=-(-HO)=)}
```



Finally, we specify that the bonds *arriving at and leaving* "`HN`" must do so at the second atom. We therefore define another alias for the invisible bond arriving at "`HN`":

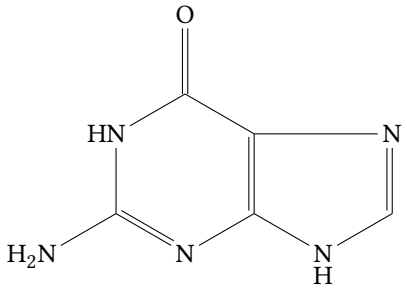
Adrenaline

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



5.12.5 Guanine

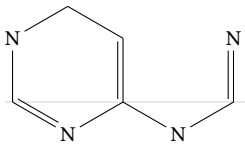
We will draw the guanine molecule:



First, let us draw the nested rings, putting only the nitrogen atoms at the vertices:

Guanine, structure

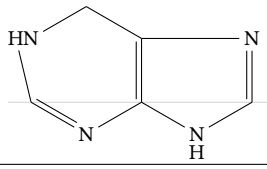
```
\chemfig{*6(=N-*6(-N-=N)=--N-)}
```



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

Guanine, step 2

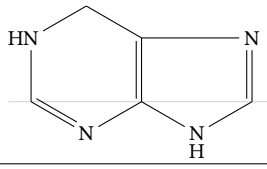
```
\chemfig{*6(=N-*6(-\chembelow{N}{H}-=N?)=?--HN-)}
```



One can see that a bond leaves from the wrong atom[5]! The automatic computation mechanism must be corrected so that the bond starts from the 2nd atom "`N`" instead of the first. For this, an optional argument "`[,,2]`" is specified for the last bond of the first 6-ring:

Guanine, step 3

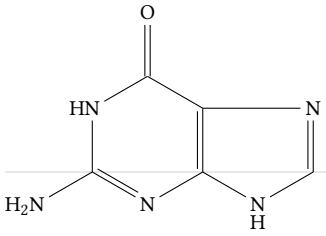
```
\chemfig{*6(=N-*6(-\chembelow{N}{H}-=N?)=?--HN-[,,2])}
```



It suffices to add the branches at the right vertices. Note in particular the branch leaving the first vertex of the first 6-ring, "`(-H_2N)`":

Guanine

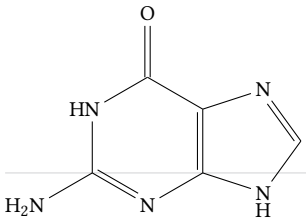
```
\chemfig{*6((-H_2N)=N-*6(-\chembelow{N}{H}-=N?)=?-(=O)-HN-[,,2])}
```



This same molecule could also have been drawn 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])}
```



[5] This seems illogical since the angle of the bond from the group `HN` to the first vertex of the 6-ring is between −90° and 90°; chem*fig* should therefore start from the 2nd atom. To explain this contradiction, one must know that in rings, the last bond always connects the last vertex to the first vertex, ignoring the *computed theoretical* angle of this bond (which is −90° here). chem*fig* uses this theoretical angle to determine the departure and arrival atoms, but does not use it to draw the bond since both ends are already defined. The departure atom of the last bond is therefore the first one.

6 Reaction schemes

I thank Clemens Niederberger for the help he gave me and the tests he carried out on the new features presented in this part.

6.1 Overview

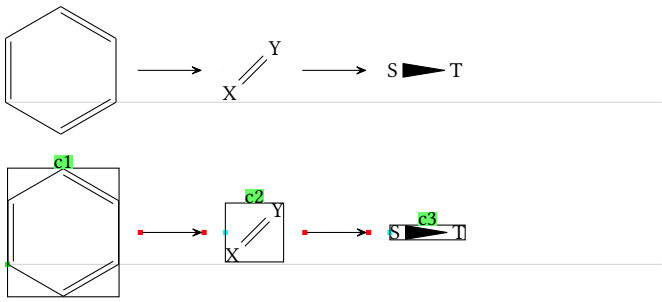
A reaction scheme is placed in the environment delimited by the macros `\schemestart` and `\schemestop`. The syntax of the macro `\schemestart` is

`\schemestart[⟨keys⟩=⟨values⟩] ⟨reaction⟩ \schemestop`

As can be seen in this example, debugging information has been hidden and then made visible to help understanding, with the key `scheme debug` and the value ⟨*true*⟩ or ⟨*false*⟩:

Example 1

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



Some remarks:

- the `\arrow` commands draw the arrows;
- everything between two `\arrow` commands is considered a compound. The choice was made that all possible settings, whether they concern arrows or compounds, are made through the arguments of the `\arrow` command, whose syntax becomes rather complex as a result;
- arrows are drawn horizontally, which can of course be changed;
- arrows lie on the imaginary line connecting the centers of the bounding boxes of the compounds (the red and blue dots are the attachment points of the arrows). This behavior can also be changed;
- the debugging information, made visible or not with the key `scheme debug`, is:
  - what appears in green above the bounding boxes is the name assigned by default by chem*fig* to the compounds, following the progression `c1`, `c2`, etc. Numbering restarts at 1 for a new reaction scheme;
  - the bounding boxes of the compounds;
  - the ends of the arrows are shown by red dots and the anchors by blue dots;
- the edge-to-edge distance between two compounds is defined with the key `compound sep` = ⟨*dim*⟩. The default value of ⟨*dim*⟩ is 5em;
- finally, the distance between the edges of the compounds and the start and end of the arrows is defined with the key `arrow offset` = ⟨*dim*⟩. By default, this ⟨*dim*⟩ is 4pt.

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

keys	default values	Description
`schemestart code`	⟨*empty*⟩	arbitrary code executed at the very beginning of the non-nested environment `\schemestart` (just after the `\begingroup`)
`schemestop code`	⟨*empty*⟩	arbitrary code executed at the very end of the non-nested environment `\schemestart` (just before the `\endgroup`)
`init anchor`	`text`	anchor of the first compound
`scheme debug`	`false`	if ⟨*true*⟩, displays the boundaries of the compounds, their names and the anchors

keys	default values	Description
compound style	⟨*empty*⟩	style of the compounds
compound sep	5em	spacing between 2 compounds
name sep	1.5ex	minimum vertical distance between the boxes of the compound and of its name
arrow offset	1em	spacing between a compound and the arrow
arrow angle	0	default angle of reaction arrows
arrow coeff	1	length coefficient of the arrows
arrow style	⟨*empty*⟩	style of the arrows
arrow double sep	2pt	spacing between double arrows
arrow double coeff	0.6	reduction coefficient of the small arrow for double arrows
arrow double harpoon	true	boolean for drawing double arrows as “harpoons”
arrow label sep	3pt	spacing between the arrow and its text
arrow head	-CF	style of the arrow tips
+ sep left	0.5em	spacing before the + sign
+ sep right	0.5em	spacing after the + sign
+ vshift	0pt	vertical shift of the + sign

6.2 Arrow types

When the command `\arrow` is followed by an optional argument between braces (which is therefore not mandatory), this argument contains the arrow type:

Arrow types

```
\schemestart A\arrow{->}B\schemestop\par % 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
```

A → B
A ⇾ B
A ← B
A ↔ B
A ⇌ B
A ↼⇀ B
A ⇁↽ B
A B
A → B

The arrow “`-U>`” is not drawn completely; an arc tangent to it at its middle can be added with the optional arguments passed to this arrow, see page 49. Here is a “`-U>`” arrow with its arc: A  B

For the sake of brevity, from now on, except for special examples, uppercase letters will be used instead of chemical formulas created with the `\chemfig` command. Obviously, reaction schemes work identically with letters or molecule drawings. Several examples involving reaction schemes can be seen in the gallery.

6.3 Arrow features

Each arrow is characterized by:

- an angle in degrees, set with `arrow angle` = ⟨*angle*⟩, which is 0 by default;
- a coefficient specified by `arrow coeff` = ⟨*decimal*⟩, which is 1 by default and gives the length of the arrow by multiplying the value of the spacing between compounds defined by `compound sep`;
- a style, defined by `arrow style` = ⟨*tikz code*⟩, empty by default, which contains ti*k*z instructions allowing the color, the thickness of the arrow or another graphic attribute to be customized.

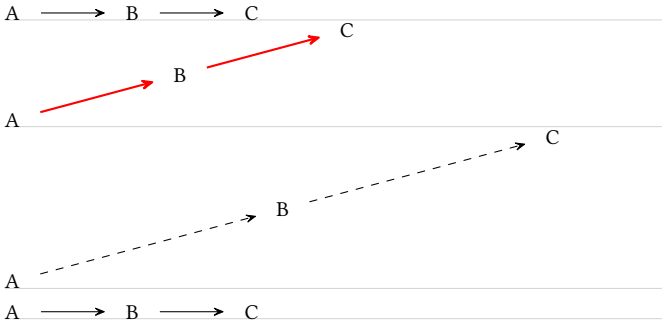
Setting default values

```
\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
```



To change one or all of these default values, one can use `\setchemfig` or the optional argument of `\schemestart`, so that the changes are limited to the current scheme.

Optional argument

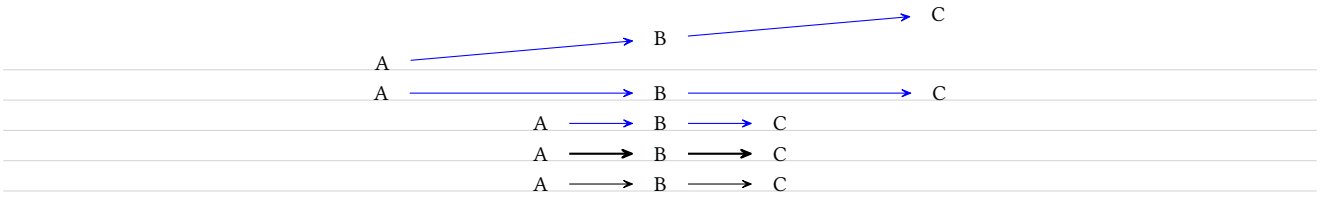
```
\setchemfig{arrow angle=5,arrow coeff=2.5,arrow style=blue}
\schemestart A\arrow B\arrow C\schemestop

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

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

\schemestart[arrow angle=0,arrow coeff=1,arrow style=thick] A\arrow B\arrow C\schemestop

\schemestart[arrow angle=0,arrow coeff=1,arrow style=black] A\arrow B\arrow C\schemestop
```



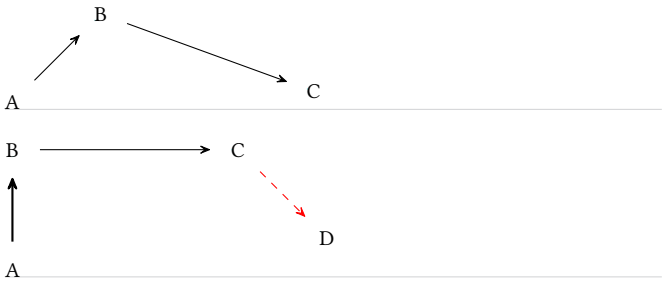
For the style, the rule is as follows: the style specified in the argument between square brackets is applied *after* the default style, without overwriting it! This is why only the color attribute “black” manages to override the default one, which is “blue”.

Finally, the command `\arrow` accepts an optional argument between square brackets of the form `[`⟨*angle*⟩`,`⟨*coeff*⟩`,`⟨*style*⟩`]` to change the features of the arrow concerned. For the style, as before, it is applied *after* the default style and any style specified in the optional argument of the command `\schemestart`, all without overwriting.

Arrow 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
```



6.4 Compound names

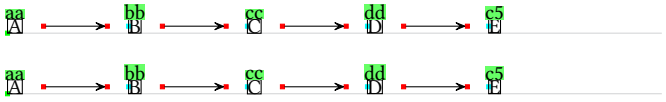
The automatic mechanism that names the compounds “c1”, “c2”, etc. can be overridden. For this, the command `\arrow` accepts an argument between parentheses which must immediately follow the command. This argument has the form `(n1--n2)`. It names “`n1`” the compound at the origin of the arrow and “`n2`” the one at its end. Any string of alphanumeric characters can be used. The numbering of the names “`c`⟨*n*⟩” continues internally, which means that if a compound has a name other than the default one, this does not change the default name of the following compounds.

The names are optional and the argument can just as well be `(n1--)` or `(--n2)`.

Compound 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
```



As can be seen, both methods are equivalent, and a compound can therefore be named with the arrow preceding it or with the arrow following it. On the other hand, when a compound is surrounded by two arrows specifying its name, the first name is ignored and a warning is issued:

Redundant names

```
\setchemfig{scheme debug=true}
\schemestart
  A\arrow(--foo)B\arrow(bar--)C
\schemestop
```



Here, the compound “B” is named “foo” by the arrow arriving at it and “bar” by the one leaving it. chem*fig* therefore issues the following warning, announcing that the name “foo” will be ignored:

```
Package chemfig Warning: two names for the same node, first name "foo" ignored
```

6.5 Anchors

As we have seen, arrows lie on the line connecting the centers of the bounding boxes of the compounds. An anchor other than the default one (which, in the ti*k*z sense, is called “`center`”) can be specified. For this, the argument between parentheses is used:

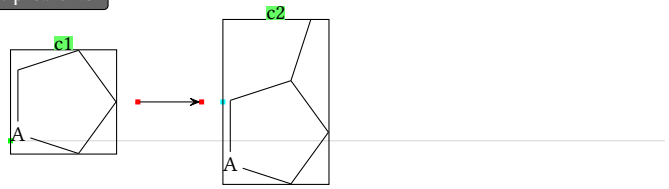
```
(n1.a1--n2.a2)
```

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 where the default alignment is not good because the two “A” are not vertically aligned. The debugging information has been made visible to show that the default “`center`” anchors are not suitable:

Alignment problems

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

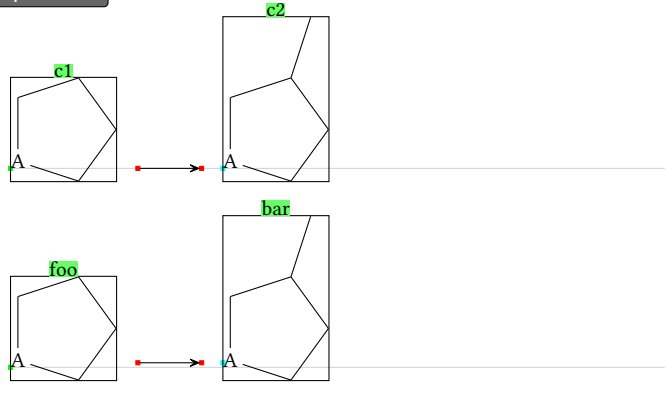


To obtain a correct alignment, the arrow will start/end either at the anchors “`base east`”/“`base west`”, or at the anchors “`mid east`”/“`mid west`”:

Alignment problems

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

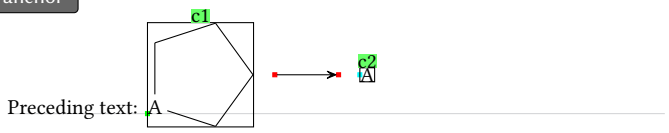
\schemestart
  \chemfig{A*5(-----)}
  \arrow(foo.mid east--bar.mid west)
  \chemfig{A*5(---(-)--)}
\schemestop
```



One last anchor remains to be specified: that of the first compound with respect to the baseline of the text preceding it. It is the green dot on the left in the following scheme:

Initial anchor

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



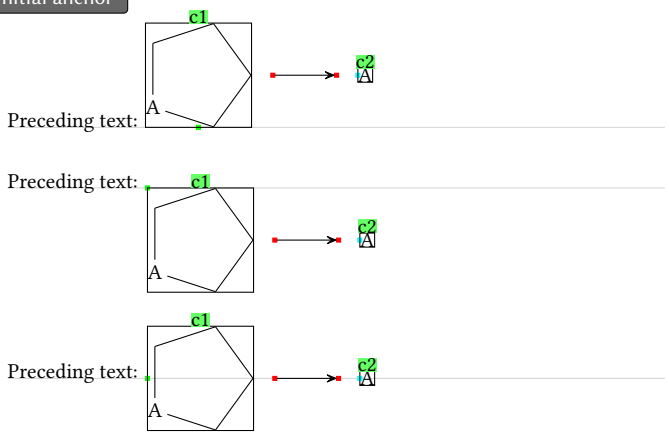
The location of this anchor on the box containing the first compound is "`text`" by default. It can be set with the key `init anchor`:

Setting the initial anchor

```
\setchemfig{scheme debug=true}
Preceding text:
\schemestart[init anchor=south]
  \chemfig{A*5(-----)}\arrow A
\schemestop
\bigskip

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

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



6.6 Compound style

Again through the argument between parentheses of the `\arrow` command, the style "⟨*s1*⟩" or "⟨*s2*⟩" of the bounding box of the departure or arrival compound can be specified with ti*k*z instructions. The complete syntax of the `\arrow` command is therefore as follows, where each specification is optional and can be omitted:

`\arrow(n1.a1[`⟨*s1*⟩`]--n2.a2[`⟨*s2*⟩`]){`⟨*arrow type*⟩`}[`⟨*angle*⟩`,`⟨*coeff*⟩`,`⟨*arrow style*⟩`]`

Just as for names, if a style is specified for a compound by the arrow arriving at it and by the one leaving it, the first style is ignored with a warning.

Compound 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
```

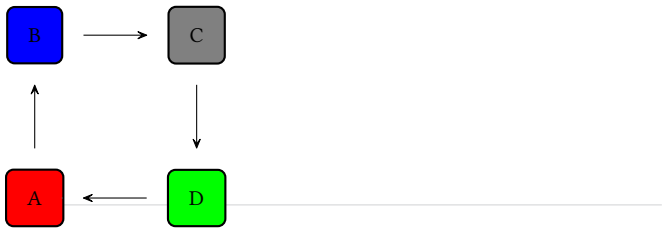


The key `compound style` = ⟨*tikz code*⟩ defines in general the style of the compounds displayed afterwards. Obviously an empty argument must be given for no style to be applied, which is the default.

Here, we will define a style in the form of boxes with rounded corners, whose background is semi-transparent:

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
```



6.7 Branches

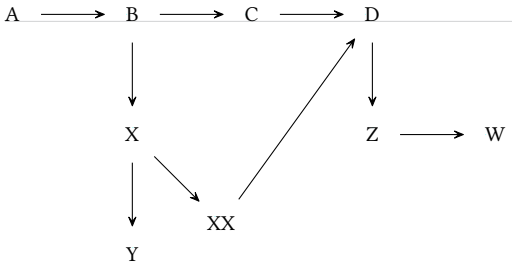
So far, only linear reaction schemes could be drawn. Branches can also be made, and this is where compound names become really important. In the argument between parentheses of the `\arrow` command, if a name is preceded by "`@`", this means that the compound already exists. Several cases can arise:

- `(@n1--n2)`: the arrow is drawn from the existing compound "`n1`" and the scheme continues after this arrow, thus creating a branch;
- `(@n1--@n2)`: the arrow is drawn between two existing compounds, whether they are already defined or defined later in the reaction scheme: this syntax can therefore be placed *anywhere* in the code of the reaction scheme;
- `(n1--@n2)`: this syntax is not allowed.

Here is an example where 3 branches are created, one starting from “B”, another from “D” and the last from “X”. At the end, an arrow is drawn between 2 existing compounds, “XX” and “D”:

Branches

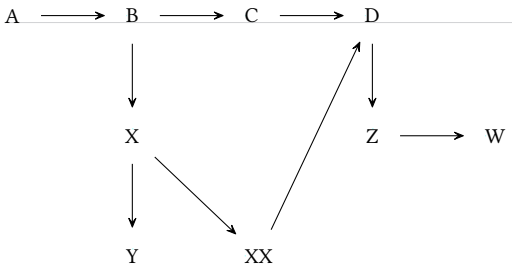
```
\schemestart
  A\arrow(aa--bb)B\arrow(--cc)C\arrow D
  \arrow(@bb--xx1)[-90]X\arrow[-90]Y% 1st branch
  \arrow(@c4--)[-90]Z\arrow W% 2nd branch
  \arrow(@xx1--xx2)[-45]XX% 3rd branch
  \arrow(@xx2--@c4)% arrow from XX to D
\schemestop
```



One might want “Y” and “XX” to be on the same horizontal line. For this, we draw an invisible horizontal arrow from “Y” to “XX” and end the scheme with an arrow between the two existing compounds “XX” and “D”:

Branches

```
\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)% arrow from X to XX
  \arrow(@xx2--@c4)% arrow from XX to D
\schemestop
```

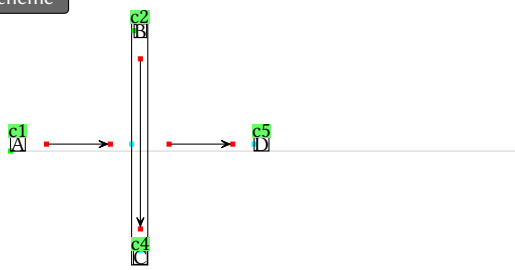


6.8 Subscheme

Part of a reaction scheme can be placed in a single bounding box, considered by chem*fig* as a compound. The command `\subscheme`, whose mandatory argument contains the subscheme between braces, groups a subscheme as a whole, and when `\subscheme` comes after an arrow, the command wraps this subscheme in a compound named “`c`⟨*n+1*⟩”:

Subscheme

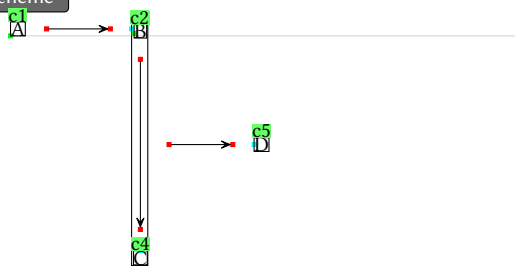
```
\setchemfig{scheme debug=true}
\schemestart
  A\arrow
  \subscheme{B\arrow[-90,2]C}
  \arrow
  D
\schemestop
```



Although it is not clear because the names overlap, the box containing the subscheme is named “`c2`” and numbering continues inside the subscheme: “`c3`” for B, “`c4`” for C. Since the first compound of the subscheme is “B”, the baseline of the subscheme is that of “B”. This can be highlighted 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 the command `\subscheme{`⟨*scheme*⟩`}` is nothing but a convenient shortcut for

`\schemestart`⟨*scheme*⟩`\schemestop`

Consequently, `\subscheme` accepts the same optional arguments as `\schemestart`.

chem*fig* provides the pair of commands `\chemleft` and `\chemright`, which place extensible delimiters on either side of some material. These commands must be followed by delimiters, just as one would put after the TEX primitives `\left` and `\right`:

`\chemleft`⟨*char1*⟩⟨*material*⟩`\chemright`⟨*char2*⟩

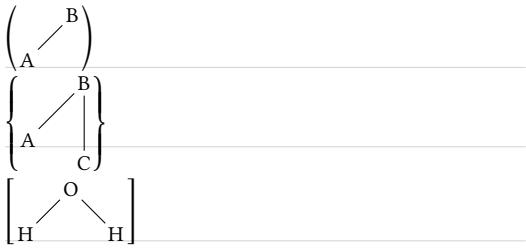
where ⟨*char1*⟩ and ⟨*char2*⟩ are "(" and ")" or "[" and "]" or any other extensible delimiter accepted by the commands `\left` and `\right`.

The macros \chemleft and \chemright

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

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

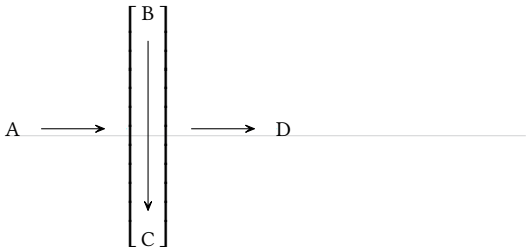
\chemleft[\chemfig{H-[1]O-[7]H}\chemright]
```



Here is the code of the reaction scheme seen previously, using `\chemleft` and `\chemright`:

Reaction scheme with \chemleft and \chemright

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



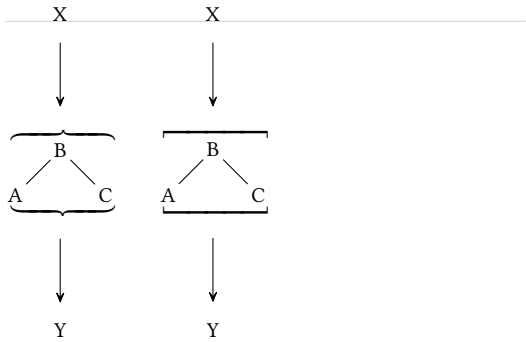
In the same vein, the macros `\chemup` and `\chemdown` work the same way except that the extensible delimiters are positioned above and below the material:

`\chemup`⟨*char1*⟩⟨*material*⟩`\chemdown`⟨*char2*⟩

Here is an example:

The macros \chemup and \chemdown

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



With compound styles, it is also possible to delimit any compound (and therefore a subscheme) with extensible delimiters (parentheses, square brackets, braces), after loading the ti*k*z library "`matrix`" by putting in the preamble of the document:

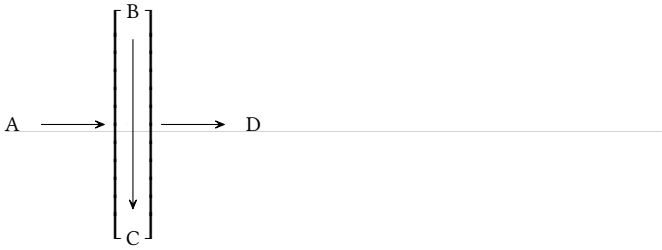
```
\usetikzlibrary{matrix}
```

Since the commands `\chemleft`, `\chemright`, `\chemup` and `\chemdown` are available, the chem*fig* package does *not* perform this loading; it is up to the user to do so if they want access to these delimiters.

Here is the previous subscheme delimited by square brackets:

The delimiters of the "matrix" library

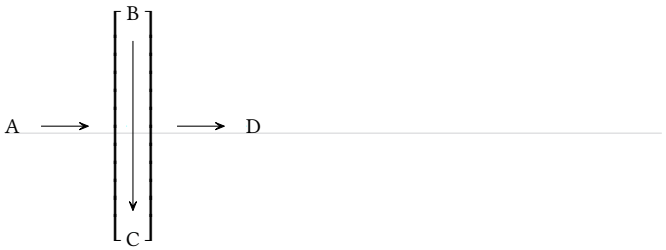
```
%\usetikzlibrary{matrix}
\schemestart
  A\arrow(--[left delimiter={[}, right delimiter={]}])
  \subscheme{B\arrow[-90,2]C}
  \arrow
  D
\schemestop
```



Since the delimiters are drawn outside the bounding box, it is advisable here to shorten the arrows arriving at and leaving these delimiters a little:

Subscheme

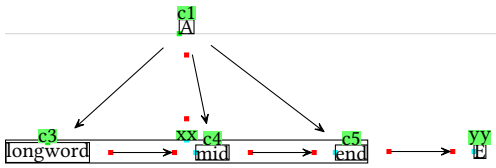
```
%\usetikzlibrary{matrix}
\schemestart
  A\arrow(--[left delimiter={[},
  right delimiter={]}])[,,shorten >=6pt]
  \subscheme{B\arrow[-90,2]C}
  \arrow[,,shorten <=6pt]
  D
\schemestop
```



Subschemes must be used wisely and can sometimes give undesired results. Here is an example where a subscheme is used to align 3 compounds horizontally:

Subscheme

```
\setchemfig{scheme debug=true}
\schemestart
  A
  \arrow{0}[-90]
  \subscheme{%
    longword\arrow{}
    mid\arrow{}
    end}
  \arrow(xx--yy){}E
  \arrow(@c1--@c3){}
  \arrow(@c1--@c5){}
  \arrow(@c1--@c4){}
\schemestop
```

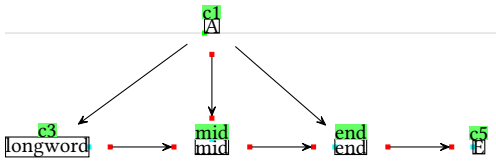


One can see that the center of the subscheme is on the same vertical line as the center of the compound “A”, since an invisible arrow with an angle of −90° was drawn. On the other hand, the arrow between the two existing compounds “A” and “mid” is *not* vertical, because “mid” is not under the center of the subscheme, since “longword” is wider than “end”. To make this arrow vertical, a subscheme cannot help, except by specifying an angle found by trial and error for the arrival anchor of the invisible arrow.

It is much simpler to proceed without a subscheme and to use a branch: draw a *visible* arrow between “A” and “mid”, then start from “mid” to draw horizontal arrows on either side, creating a branch for the arrows to the right.

Subscheme

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



6.9 Optional arguments of arrows

Inside the argument between braces of the `\arrow` command, the name of the arrow can be followed by optional arguments placed between square brackets. Here are these optional arguments and their meaning for the arrows defined by chem*fig*:

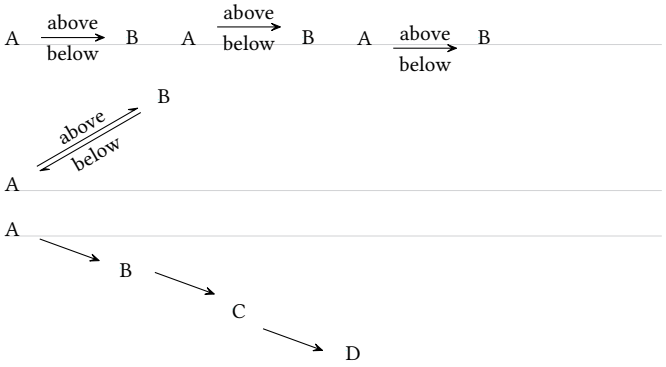
- the arrows “`->`”, “`<-`”, “`<->`”, “`<=>`”, “`<<->`”, “`<->>`”, “`-/>`” have three optional arguments:
  - the first contains the “label” placed above the arrow;
  - the second contains the “label” placed below the arrow. These two labels are oriented at the same angle as the arrow. The perpendicular shift between the arrow and the anchor of the labels can be set with the key `arrow label sep` = ⟨*dim*⟩, which is 3pt by default. The labels contained in the two optional arguments *are not* typeset in math mode;
  - the third is a dimension corresponding to the shift to be applied to the arrow in the direction perpendicular to it: this dimension is positive to shift the arrow (and its labels, if any) upward and negative to shift it downward.
- the arrow “`-U>`” has 5 optional arguments:
  - the first three are identical to those of the other arrow types;
  - the fourth is the coefficient (0.333 by default) multiplying the length of the arrow to obtain the radius of the arc;
  - the fifth is the half-angle from the center of the drawn arc; it is 60 degrees by default.
- the invisible arrow “`0`” accepts 2 optional arguments, of the same type as the first two seen above.

Optional arguments of arrows

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

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

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



A problem arises when arrows are vertical:

Vertical arrows

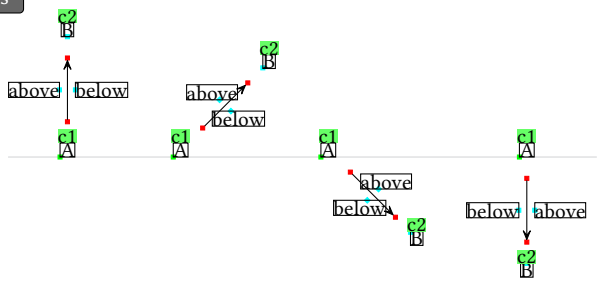
```
\schemestart
  A\arrow{->[above][below]}[-90]B
\schemestop
```



For readability, it would be preferable for the labels placed above and below to be horizontal. The angle at which the labels are displayed can be chosen, the default value being the angle of the arrow. To specify another angle, simply put `*{`⟨*angle*⟩`}` at the beginning of the optional arguments:

Choosing angles

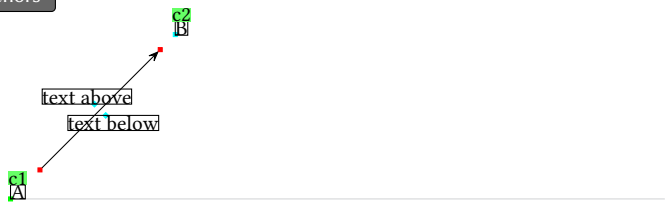
```
\setchemfig{scheme debug=true}
\schemestart A\arrow{->[*{0}above][*{0}below]}[90]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}above][*{0}below]}[45]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}above][*{0}below]}[-45]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}above][*{0}below]}[-90]B\schemestop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



The default anchor to which each label is attached sometimes produces undesirable displays:

Anchors

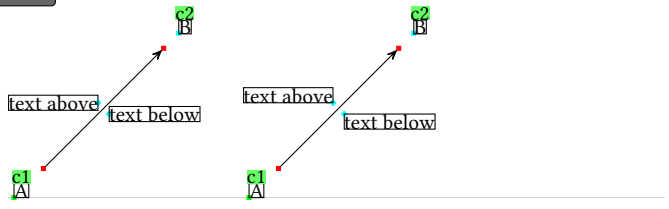
```
\setchemfig{scheme debug=true}
\schemestart
  A\arrow{->[*{0}text above][*{0}text below]}[45,2]B
\schemestop
```



The attachment anchor can also be specified to override the one chosen by default by chem*fig*. Simply use the following syntax: `*{`⟨*angle*⟩`.`⟨*anchor*⟩`}`.

Anchors

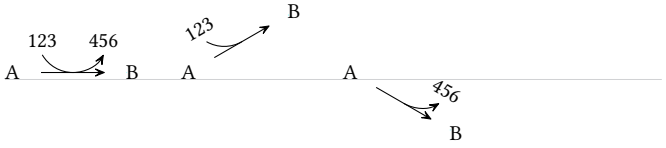
```
\setchemfig{scheme debug=true}
\schemestart
  A\arrow{->[*{0.0}text above][*{0.180}text below]}[45,2]B
\schemestop
\qquad
\schemestart
  A\arrow{->[*{0.south east}text above]%
    [*{0.north west}text below]}[45,2]B
\schemestop
```



The special case to consider is the arrow "`-U>`". If either of the two labels contained in the first two optional arguments is present, the corresponding arc is drawn:

The arrow -U>

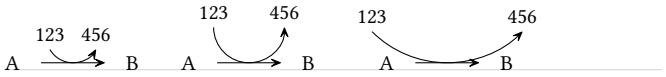
```
\schemestart A\arrow{-U>[123][456]}B\schemestop
\qquad
\schemestart A\arrow{-U>[123]}[30]B\schemestop
\qquad
\schemestart A\arrow{-U>[][456]}[-30]B\schemestop
```



The fourth and fifth optional arguments change the dimensions of the arc. They represent respectively the coefficient multiplying the length of the arrow to obtain the radius, and the angle corresponding to the half arc (only the absolute value of the angle is taken into account):

The arrow -U>

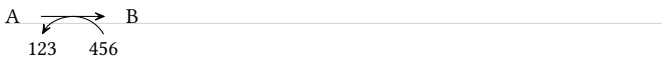
```
\schemestart A\arrow{-U>[123][456][][0.25]}B\schemestop
\qquad
\schemestart A\arrow{-U>[123][456][][][90]}B\schemestop
\qquad
\schemestart A\arrow{-U>[123][456][][1][45]}B\schemestop
```



With a negative radius and angle, the arc is located below the arrow:

The arrow -U>

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



The first two arguments have the same features regarding display angles and configurable anchors as those of the other arrows:

The arrow -U>

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



6.10 Creating your own arrows

This section, rather technical and requiring some knowledge of ti*k*z, will only interest advanced users who need to define their own arrows.

The command `\definearrow` allows custom arrows to be built. The syntax of this command is as follows:

`\definearrow{`⟨*number*⟩`}{`⟨*arrow name*⟩`}{`⟨*code*⟩`}`

where ⟨*number*⟩ is the number of optional arguments, referred to in the ⟨*code*⟩ with the usual syntax `#1`, `#2`, etc. Default values cannot be defined for these optional arguments, and if they are absent when the user uses the macro `\arrow`, they will be empty.

Before going further, let us examine the macros available internally when arrows are drawn. Since these macros have the character "_" in their name, they can only be accessed between the commands `\catcode‘\_=11` and `\catcode‘\_=8`.

- `\CF_arrowstartname` and `\CF_arrowendname` contain the names of the compounds (which are nodes for ti*k*z) between which the arrow must be drawn;
- `\CF_arrowstartnode` and `\CF_arrowendnode` contain the names of the nodes where the ends of the arrows will be placed. These names are followed, if not empty, by the anchors specified by the user in the argument between parentheses of the `\arrow` command;
- `\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`" by the dimension contained in its argument, in the direction perpendicular to the arrow;
- `\CF_arrowdisplaylabel{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}` is the most complex and places the labels with the following arguments:
  - `#1` and `#5` are the labels to write;
  - `#2` and `#6` are real numbers between 0 and 1 specifying where on the arrow these labels must be written, 0 being the beginning and 1 the end, all this assuming that the arrow is *straight*;
  - `#3` and `#7` are the characters "+" or "-", "+" displaying the label above and "-" displaying it below;
  - `#4` and `#8` are the names of the nodes corresponding to the beginning and the end of the arrow.
- the arrow tips used are based on the model "`CF`" for a full arrow, and with the option "harpoon" for the upper half arrow.

6.10.1 A first arrow

As an example, let us build an arrow with a disk in its middle, which we will call "`-.>`". Let us decide that it will accept 4 optional arguments. As for the arrows already defined, the 1st and 2nd will be the labels to put above and below; the 3th will be the shift of the arrow perpendicular to its direction. Finally, the 4th argument will be the size of the dot, which will be 2pt if the 4th argument is absent.

We will first write `\definearrow{4}{-.>}` to declare that the arrow will have 4 optional arguments and will be called "`-.>`". First of all, the positions of the nodes between which the arrow must be drawn must be modified to take into account the shift contained in the 3th argument. This is done by the macro `\CF_arrowshiftnodes`, so it suffices to start the code of the arrow with: `\CF_arrowshiftnodes{#3}`. Then the arrow itself must be drawn, placing a node named "`mid@point`" in the middle of the segment along the way, and then a circle centered on this last node is drawn. All this is done by the following ti*k*z code:

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

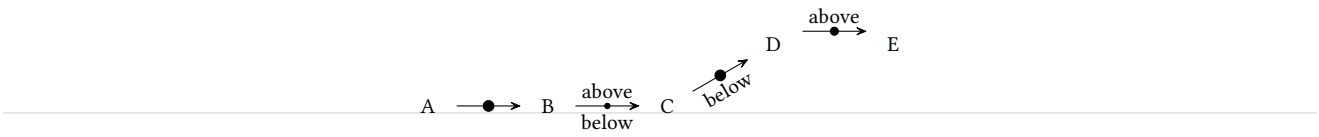
All that remains is to place the labels, if they exist, with the following line:

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

Here is our finished arrow:

Arrow "`-.>`"

```
\catcode`\_11
\definearrow4{-.>}{%
  \edef\pt_radius{\ifx\empty#4\empty 2pt\else #4\fi}% radius of the dot
  \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
```



6.10.2 A curved arrow

Why not now create a curved arrow? To keep things simple, let us say that it will have only one optional argument containing the ti*k*z code defining the control point(s), and if this argument is empty, an arrow of type "`-CF`" will be drawn.

If the argument `#1` is not empty, what interests us is not `\CF_arrowstartnode` and `\CF_arrowendnode`, which contain the names of the nodes where the ends of the arrows will be placed, because the location of these nodes is already determined by anchors computed for *straight* arrows! Instead we will take `\CF_arrowstartname` and `\CF_arrowendname`, which contain the names of the compounds (which are nodes for ti*k*z) between which the arrow must be drawn. Here is the ti*k*z code that will draw the curved line between these two compounds:

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

ti*k*z code must be added to shorten the arrow by the value `\CF_arrowoffset` defined by the key `arrow offset`, since we do not start from the nodes corresponding to straight arrows `\CF_arrowstartnode` and `\CF_arrowendnode`. The following code must therefore be added at the beginning of `\CF_arrowcurrentstyle`:

```
shorten ⟨=\CF_arrowoffset, shorten ⟩=\CF_arrowoffset
```

this is what the two lines following the `\else` do.

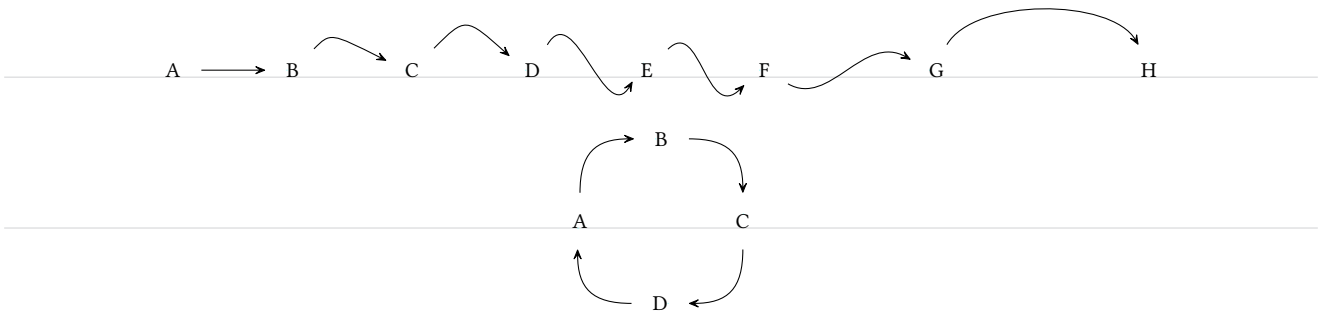
Here is our arrow:

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
```



6.11 The \merge command

The command `\merge` draws arrows that start from several existing compounds and then join to continue the reaction scheme. Immediately after the command `\merge`, the direction of progression must be specified. For this, one of the 4 direction characters is used: "`>`" (the default if the character is absent), "`<`", "`^`" and "`v`".

The syntax then becomes:

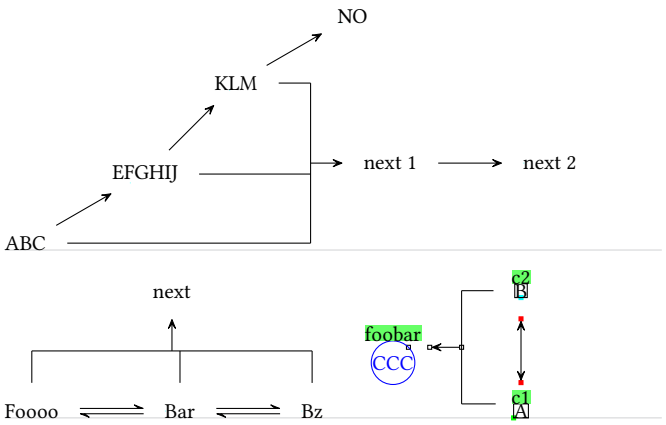
```
\merge{dir}(n1.a1)(n2.a2)(...)(ni.ai)--(n.a[s])
```

where the names "`ni`" before the double dash are those of already existing compounds to be joined. The anchor "`ai`" from which the arrow must start can also be specified if the default anchor is not suitable. As for the `\arrow` command, "`n.a[s]`" contains the name, the anchor and the style of the next compound.

The \merge command

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

\schemestart
Foooo\arrow(foo--bar){<=>}Bar\arrow(--baz){<=>}Bz
\merge^(foo)(bar)(baz)--()next
\schemestop
\qquad
\setchemfig{scheme debug=true}
\schemestart
A\arrow{<->}[90]B
\merge<(c1.120)(c2)--(foobar.45[circle,blue])CCC
\schemestop
```



As for the geometry of the `\merge` arrow, it consists of *n* lines leaving each compound and going up to the junction line perpendicular to them: the default length of the shortest of these lines is half the spacing between compounds defined with `\setcompoundsep`. The arrow drawn from the junction line to the next compound also has this default length, and it starts from the middle of the junction line. These 3 geometric features can be changed with the optional argument that comes after the name of the next compound:

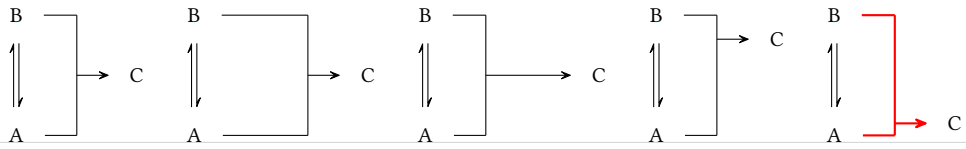
```
\merge{dir}(n1.a1)(n2.a2)(...)(ni.ai)--(n.a[s])[c1,c2,c,⟨style⟩]
```

where:

- the length of the shortest line between the compounds to be joined and the junction line is obtained by multiplying the spacing between compounds defined with `\setcompoundsep` by the coefficient `c1`, which is 0.5 by default;
- the length of the arrow between the junction line and the next compound is obtained by multiplying the spacing between compounds by the coefficient `c2`, which is 0.5 by default;
- the place where the arrow leaves the junction line is determined by the coefficient `c`, knowing that if it is 0, the arrow starts from the left of the junction line (or from the top if the direction is `v` or `^`);
- the style of the arrow drawn with `\merge` is defined by the last argument ⟨*style*⟩.

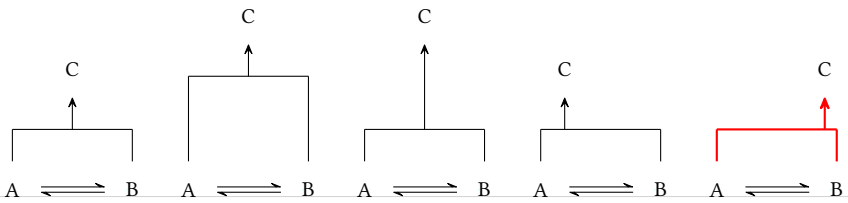
Geometric parameters of \merge

```
\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
```



Geometric parameters of \merge

```
\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, labels can be written above or below the arrow created by the `\merge` command. For this, the character defining the direction accepts two optional arguments between square brackets, the first containing the label put above and the second the one put below the arrow. The complete syntax of the `\merge` command is therefore:

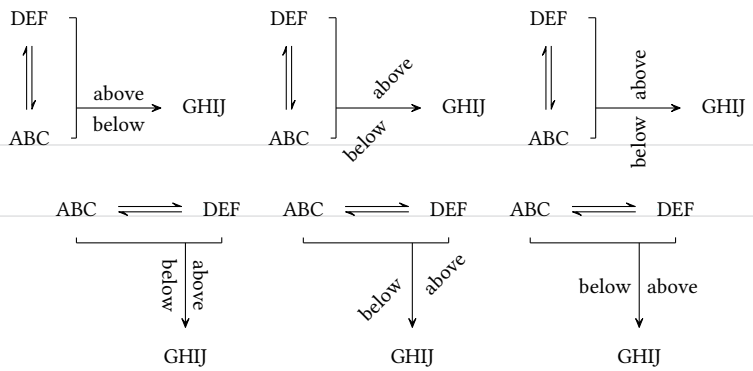
```
\merge{dir}[labelup][labeldown](n1.a1)(n2.a2)(...)(ni.ai)--(n.a[s])[c1,c2,c,style]
```

All the features already presented for arrow labels are possible here, namely choosing the rotation angle and the anchor with the syntax `*{angle.anchor}` placed just before the content of the label.

Labels of the \merge command

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

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



6.12 The + sign

Between the commands `\schemestart` and `\schemestop`, the macro "`\+`" is available, which displays a + sign. It accepts an optional argument *between braces* containing 3 dimensions in this form {⟨*dim1*⟩,⟨*dim2*⟩,⟨*dim3*⟩} where:

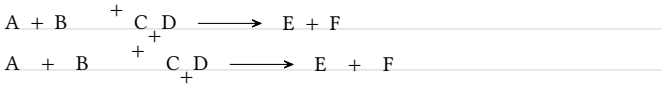
- ⟨*dim1*⟩ and ⟨*dim2*⟩ are the dimensions inserted before and after the + sign;
- ⟨*dim3*⟩ is the vertical shift to be applied to the sign.

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

The \+ command

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

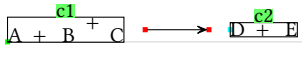
\setchemfig{+ sep left=1em,+ sep right=1em,+ vshift=0pt}
\schemestart
A\+B\+{2em,,5pt}C\+{0pt,0pt,-5pt}D\arrow E\+F
\schemestop
```



As can be seen in the example below, it must be understood 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
```

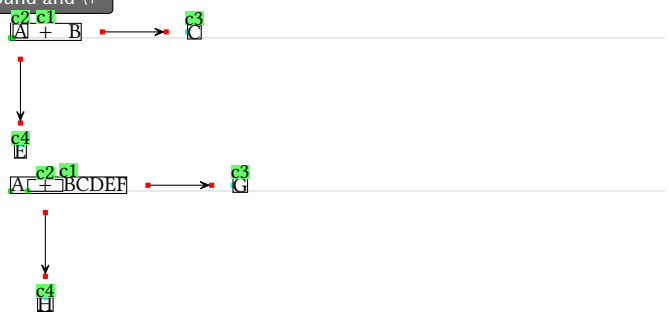


It therefore becomes difficult to start an arrow exactly under the letter "A", since this letter is not a single compound in the eyes of chem*fig*. In this case, the `\subscheme` command can be used to wrap the letter "A" in a single compound, or even the sign itself, so that it can be referred to later with the name assigned to it:

Subcompound and \+

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

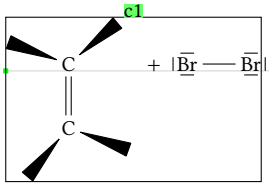
\schemestart
A\subscheme{\+}BCDEF \arrow G
\arrow(@c2--)[-90]H
\schemestop
```



The alignment of the "+" sign with the preceding or following molecules may sometimes be a problem. Here is an example:

Alignment of the + sign

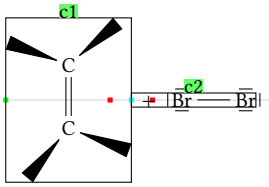
```
\setchemfig{scheme debug=true}
\schemestart
  \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
  \+
  \chemfig{\charge{90=\|,180=\|,270=\|}{Br}-\charge{0=\|,90=\|,-90=\|}{Br}}
\schemestop
```



What happens is that the "+" sign is on the same baseline as the preceding compound, and this baseline is that of the upper atom "C". One could of course shift the "+" sign, but this would not change the vertical position of "Br — Br". In fact, the "+" sign does not stop the reading of a compound for chem*fig*, as can be seen in the example above where everything is included in the compound "c1". We will therefore have to stop the compound after the first molecule with an `\arrow{0}[,0]`, which produces an invisible arrow of zero length. To center everything vertically, we will also specify that the anchor of the first compound must be "`west`" (or "`180`", which is a synonym) with the optional argument of the `\schemestart` command:

Alignment of the + sign

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

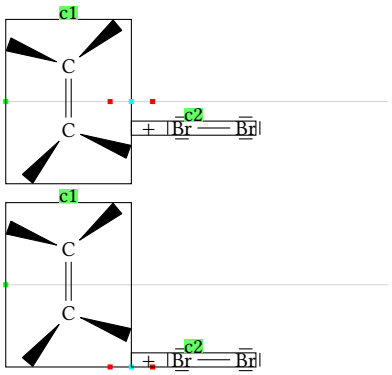


In this way, the first compound "c1" is the first molecule and the second compound is the rest, that is, the "+" sign and the second molecule. One could have used the anchors or styles via the `\arrow` command to place the second compound elsewhere. Here, for example, the second compound is shifted 10pt downward in the first case, and in the second case the anchor "south east" of the first compound is made to coincide with the anchor "south west" of the second:

Alignment of the + sign

```
\setchemfig{scheme debug=true}
\schemestart[init anchor=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[init anchor=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
```



7 Horizontal reactions

7.1 Syntax

A horizontal reaction is placed in an environment with the syntax:

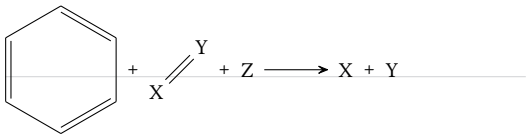
```
\hreac ⟨reaction⟩\endhreac
```

The tokens `+` `>` `\>` `\^` are reserved and have a meaning in the ⟨*reaction*⟩:

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

Example 1

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

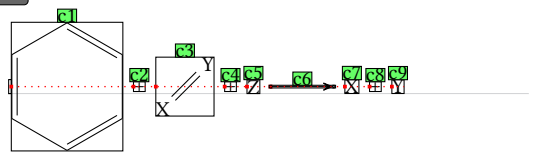


7.2 Placement of compounds

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 as red dots when the option `hreac debug` is enabled:

Vertical alignment of compounds

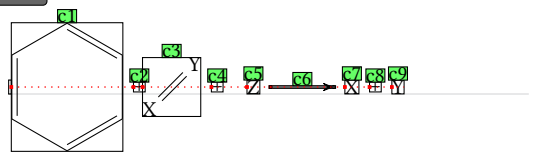
```
\hreac[hreac debug]
    \chemfig{*6(-=-=-=)}
    +
    \chemfig{X=[1]Y}
    +
    Z > X + Y
\endhreac
```



When `\>{`⟨*dimension*⟩`}` is encountered within a ⟨*compound*⟩ (but preferably at its beginning), the compound is shifted horizontally by ⟨*dimension*⟩.

Horizontal adjustment of compounds

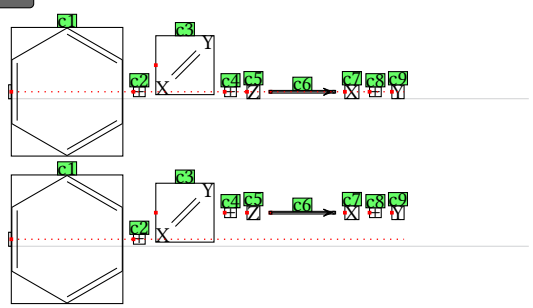
```
\hreac[hreac debug]
    \chemfig{*6(-=-=-=)}
    +
    \>{-5pt}\chemfig{X=[1]Y}
    +
    \>{5pt}Z > X + Y
\endhreac
```



When `\^{`⟨*dimension*⟩`}` is encountered at the beginning of a ⟨*compound*⟩, this compound (and only this one) is shifted vertically by `{`⟨*dimension*⟩`}`. In the starred version `\^*{`⟨*dimension*⟩`}`, the compound and all following 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
```



7.3 Compound names

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

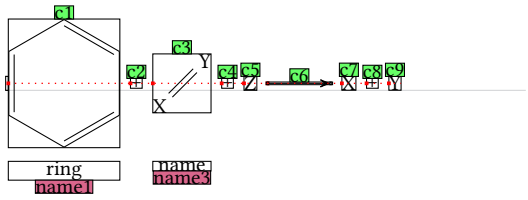
- `\chemname[`⟨*dimension*⟩`]{`⟨*compound*⟩`}{`⟨*name*⟩`}`;
- ⟨*compound*⟩`\name[`⟨*dimension*⟩`]{`⟨*name*⟩`}` or `\name[`⟨*dimension*⟩`]{`⟨*name*⟩`}`⟨*compound*⟩, the order of appearance being irrelevant.

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

All names are numbered `name`⟨*n*⟩ where ⟨*n*⟩ is the integer corresponding to the ⟨*compound*⟩. They are placed in boxes whose width is that of the ⟨*compound*⟩. The upper boundaries of these boxes are on the same horizontal line, whose vertical position is computed to be below the compound with the greatest depth.

Compound names

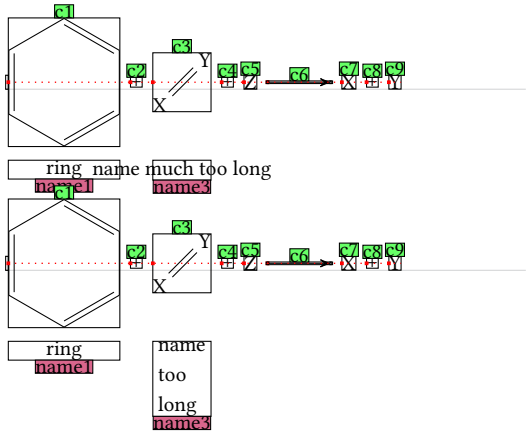
```
\hreac[hreac debug]
    \chemfig{*6(-=-=-=)}\name{ring}
    +
    \chemfig{X=[1]Y}\name{name}
    +
    Z > X + Y
\endhreac
```



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

Compound names

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



7.4 Arrow types

If the token “>” is immediately followed by a text between 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⟩}[⟨upper label⟩][⟨lower label⟩]
```

Using this optional parameter between braces gives access to these other arrow types, already described in the part “Reaction schemes”: “`->`”, “`<-`”, “`<->`”, “`<=>`”, “`<<->`” and “`<->>`”.

The 6 arrow types

```
\hreac A > B \endhreac\par% same as >{->}
\hreac A >{<-} B \endhreac\par
\hreac A >{<->} B \endhreac\par
\hreac A >{<=>} B \endhreac\par
\hreac A >{«->} B \endhreac\par
\hreac A >{<-»} B \endhreac
```

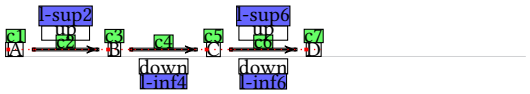


7.5 Arrow labels

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

Arrow labels

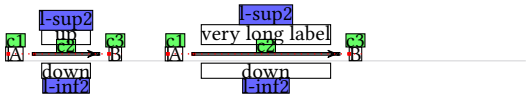
```
\hreac[hreac debug]
    A >[up] B >[][down] C >[up][down] D
\endhreac
```



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

Arrow extension

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



As for compound names, `\\` can be used to typeset a ⟨*label*⟩ on several lines.

Label on several lines

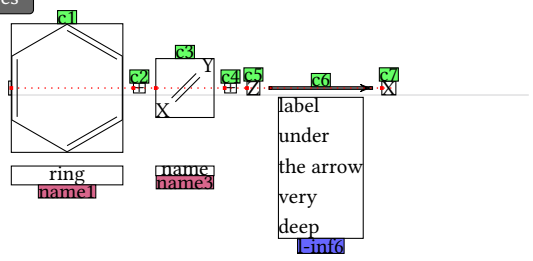
```
\hreac[hreac debug]
    A >[very\\long\\label][down] B
\endhreac
```



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

Independence of labels and names

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



8 List of commands

The macros created by chem*fig* are:

Macros	Description
`\chemfig[⟨parameters⟩]{⟨code⟩}`	draws the molecule whose drawing is described by the ⟨*code*⟩
`\setchemfig{⟨parameters⟩}`	sets the parameters with the syntax `key = ⟨value⟩`, see p. 4
`\resetchemfig`	Restores the parameters to their default values
`\printatom`	this macro displays the atoms in molecules, see p. 22
`\hflipnext`	the next molecule will be flipped horizontally, see p. 33
`\vflipnext`	the next molecule will be flipped vertically, see p. 33
`\definesubmol {⟨name⟩}⟨n⟩[⟨code1⟩]{⟨code2⟩}`	creates an alias ⟨*name*⟩ that can be placed in the code of molecules to be drawn and is replaced by ⟨*code1*⟩ or ⟨*code2*⟩ depending on the slope of the last bond, see p. 26
`\chemskipalign`	Ignores the current group of atoms for the vertical alignment mechanism, see p. 25
`\redefinesubmol {⟨name⟩}⟨n⟩[⟨code1⟩]{⟨code2⟩}`	replaces the already existing alias ⟨*name*⟩ with the new ⟨*code*⟩, see p. 27
`\charge [⟨parameters⟩]⟨pos⟩ [⟨tikz⟩]{⟨atom⟩}`	displays the ⟨*atom*⟩ and places the charges at their ⟨*positions*⟩ without taking them into account in the bounding box, see p. 28
`\Charge [⟨parameters⟩]⟨pos⟩ [⟨tikz⟩]{⟨atom⟩}`	same as `\charge`, but the charges are taken into account in the bounding box.
`\chemmove[⟨tikz options⟩]⟨tikz code⟩`	opens a `\tikzpicture` environment, adding the ⟨*tikz options*⟩ to those already existing, and connects with the ⟨*tikz code*⟩ the nodes placed in the molecules with the character "`@`", see p. 18
`\chemabove[⟨dim⟩]{⟨txt1⟩}{⟨txt2⟩}`	writes ⟨*txt1*⟩ and places ⟨*txt2*⟩ above it, leaving ⟨*dim*⟩ of vertical space. This command does not change the bounding box of ⟨*txt1*⟩, see p. 30
`\chembelow[⟨dim⟩]{⟨txt1⟩}{⟨txt2⟩}`	writes ⟨*txt1*⟩ and places ⟨*txt2*⟩ below it, leaving ⟨*dim*⟩ of vertical space. This command does not change the bounding box of ⟨*txt1*⟩, see p. 30
`\Chemabove[⟨dim⟩]{⟨txt1⟩}{⟨txt2⟩}`	writes ⟨*txt1*⟩ and places ⟨*txt2*⟩ above it, leaving ⟨*dim*⟩ of vertical space, see p. 30
`\Chembelow[⟨dim⟩]{⟨txt1⟩}{⟨txt2⟩}`	writes ⟨*txt1*⟩ and places ⟨*txt2*⟩ below it, leaving ⟨*dim*⟩ of vertical space, see p. 30
`\chemname[⟨dim⟩]{⟨molecule⟩}{⟨name⟩}`	displays the ⟨*name*⟩ under the ⟨*molecule*⟩, see p. 20
`\chemnameinit`	initializes the greatest depth of the molecules encountered, to obtain a correct alignment of their names
`\schemestart...\schemestop`	markers between which a reaction scheme is drawn, see p. 42
`\arrow`	draws an arrow in a reaction scheme (the command is only defined inside a reaction scheme)
`\+`	displays a + sign in a reaction scheme (the command is only defined inside a reaction scheme), see p. 43
`\subscheme{⟨code⟩}`	draws a reaction subscheme (the command is only defined inside a reaction scheme), see p. 47
`\definearrow`	defines an arrow for a reaction scheme, see p. 51
`\chemleft⟨char1⟩⟨material⟩ ...\chemright⟨char2⟩`	draws extensible delimiters defined by ⟨*char1*⟩ and ⟨*char2*⟩ to the left and right of the ⟨*material*⟩, see p. 47

`\chemup⟨char1⟩⟨material⟩ ...\chemdown⟨char2⟩`	draws extensible delimiters defined by ⟨*char1*⟩ and ⟨*char2*⟩ above and below the ⟨*material*⟩, see p. 48
`\polymerdelim [⟨parameters⟩]{⟨node1⟩}{⟨node2⟩}`	draws delimiters at the specified nodes, see p. 31
`\hreac...\endhreac`	markers between which a horizontal reaction is drawn, see p. 57
`>{⟨type⟩}[⟨upper label⟩][⟨lower label⟩]`	draws a horizontal arrow, see p. 58
`+`	displays the + sign in the horizontal reaction
`\>{⟨dimension⟩}`	shifts the current compound horizontally, see p. 57
`\^{⟨dimension⟩}` or `\^*{⟨dimension⟩}`	shifts the current compound vertically, see p. 57
`\name{⟨name⟩}`	puts the ⟨*name*⟩ under the current compound, see p. 58

9 Gallery

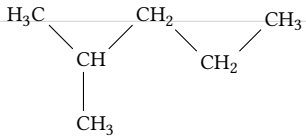
This manual ends with drawings of molecules of varying complexity.

The curious user may take an interest in the ⟨*code*⟩ of each molecule, although it sometimes becomes off-putting when the molecules become complex. Indeed, beyond a certain level of complexity, although it is fairly easy to write a ⟨*code*⟩ to draw a molecule, it is rather hard to reread it in order to analyze it afterwards. The limits of the immediate readability of the code of a complex drawing are quickly reached.

In any case, I hope this package will help all users of TEX, and even those of LATEX, who wish to draw chemical molecules. Although chem*fig* has been thoroughly tested, I hope you will be lenient regarding any bugs you encounter. An `email` to report a malfunction or to suggest any improvement will be welcome.

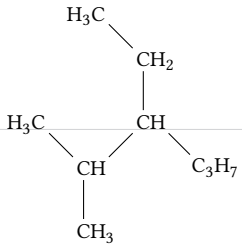
2-methylpentane

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



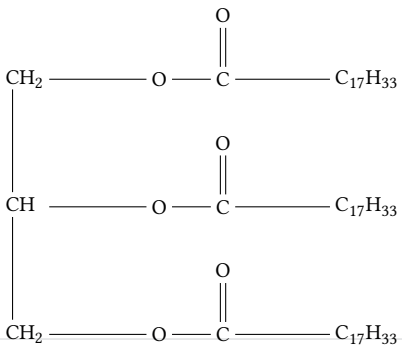
3-ethyl-2-methylhexane

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



Stearin, condensed structural formula

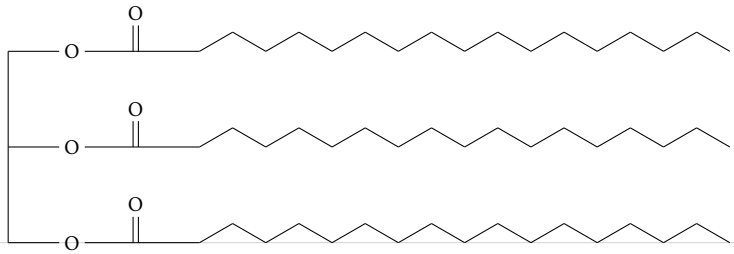
```
\definesubmol{@}{([0,2]-O-[0,1]C(=[2,1]O)-C_{17}H_{33})}
\chemfig{[2,2]CH_2!@-CH_{\phantom 2}!@-CH_2!@}
```



Stearin, skeletal formula

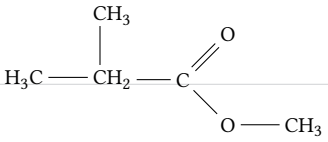
```
\definesubmol{x}{-[:+30,.6]-[:-30,.6]}
\definesubmol{y}{-O-(=[2,.6]O)-!x!x!x!x!x!x!x}
```

```
\chemfig{[2]([0]!y)-[,1.5]([0]!y)-[,1.5]([0]!y)}
```



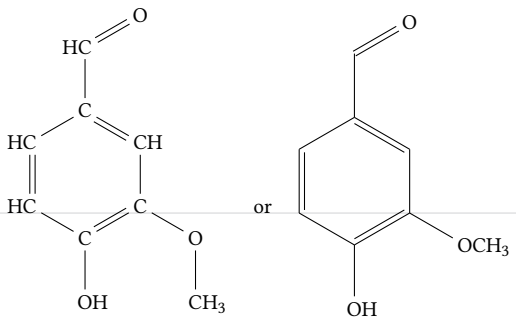
Methyl 2-methylpropanoate

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



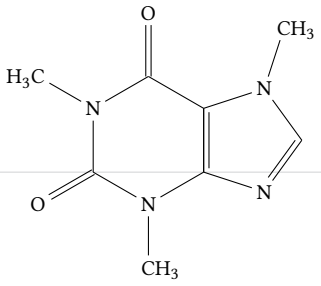
Vanillin

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



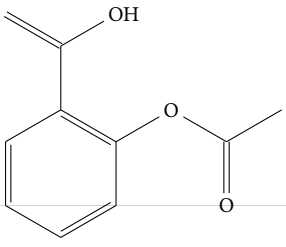
Caffeine

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



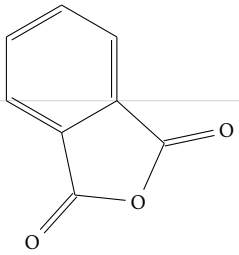
Aspirin

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



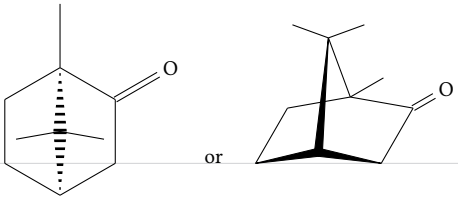
Phthalic anhydride

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



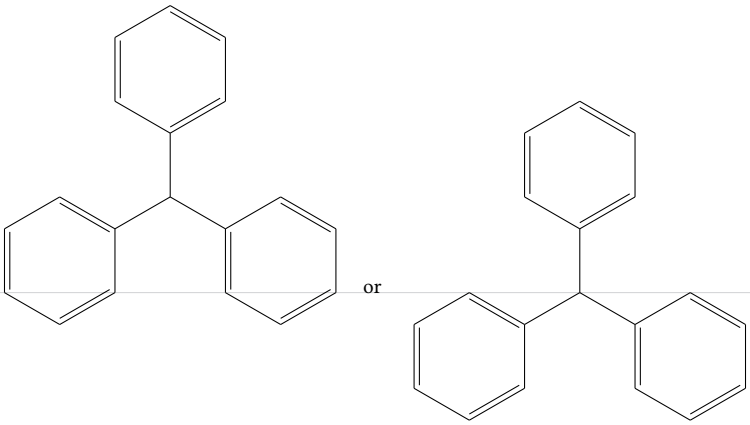
Camphor

```
\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]}
```



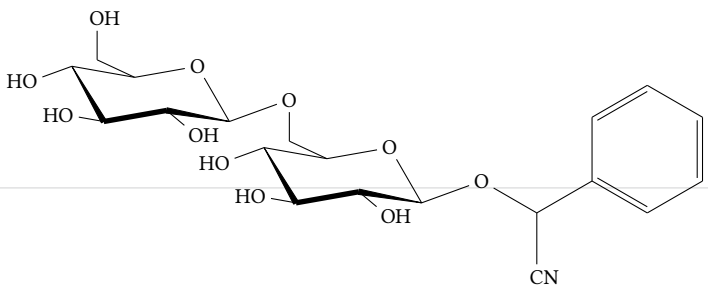
Triphenylmethane

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



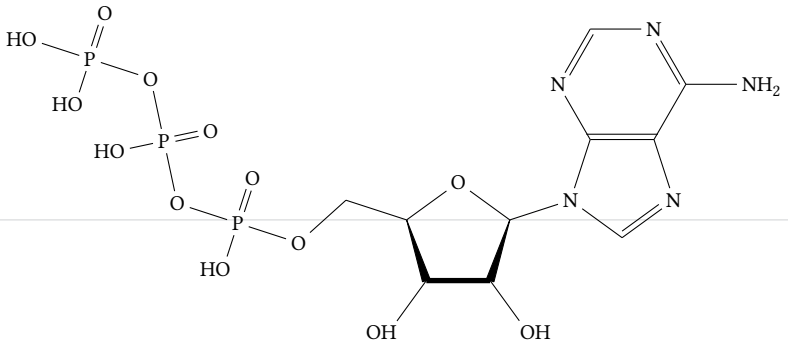
Amygdalin

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



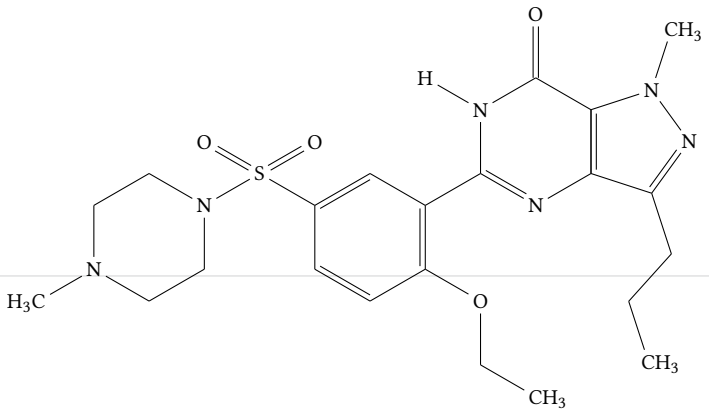
Adenosine triphosphate

```
\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-)}
```



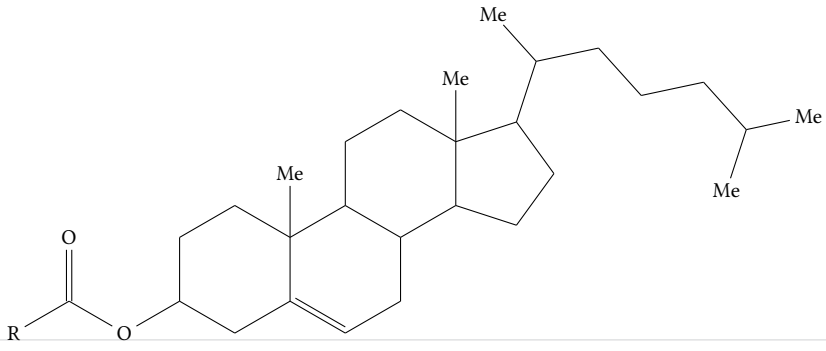
Viagra

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



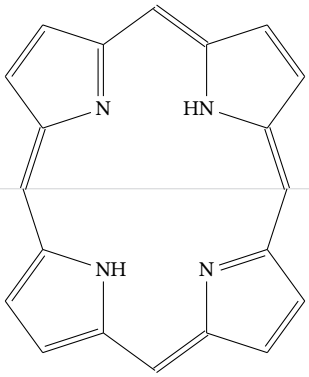
Cholesterol ester

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



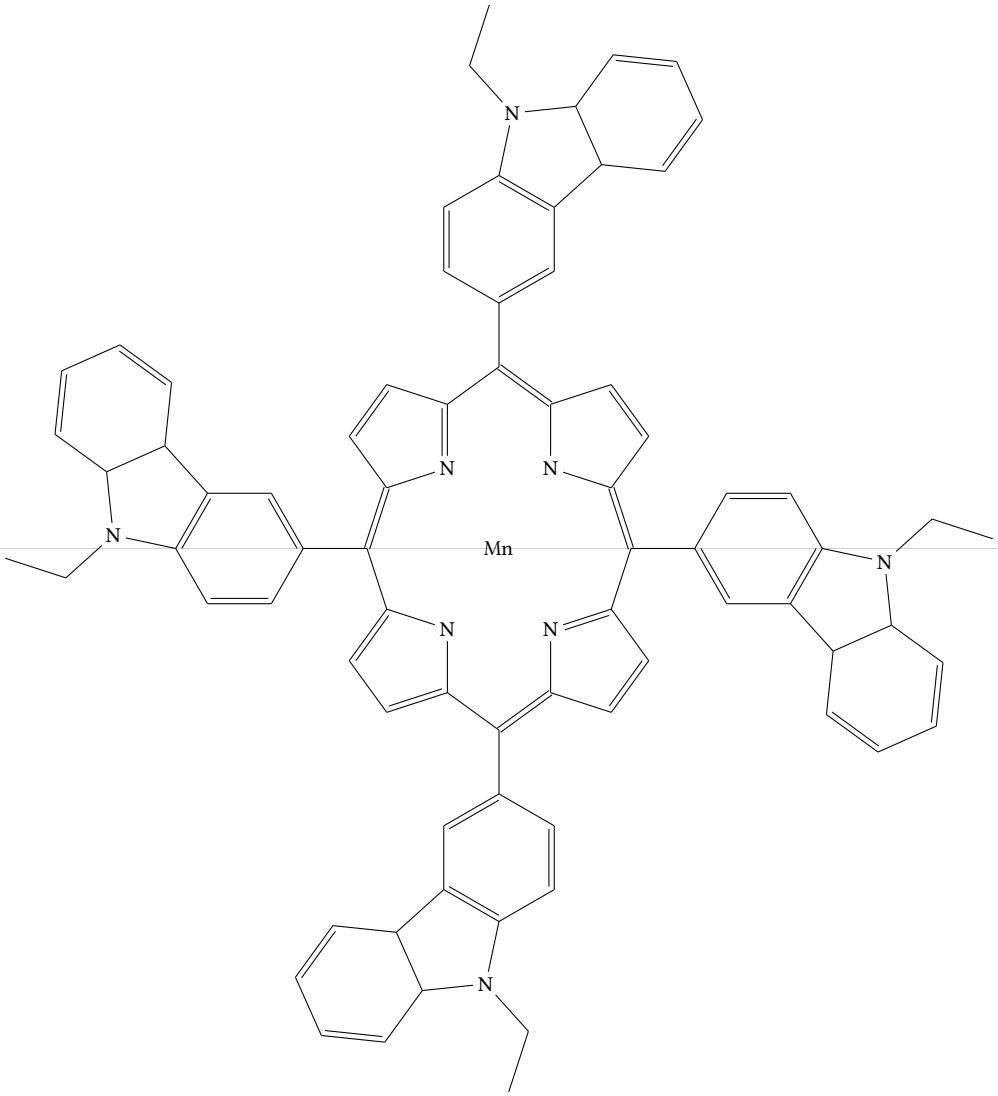
Porphyrin

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



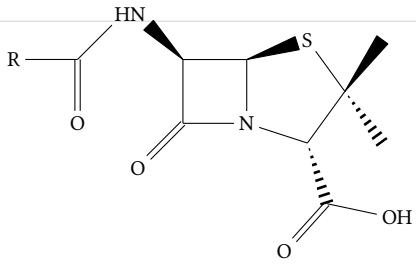
Manganese 5,10,15,20-tetra(N-ethyl-3-carbazolyl) porphyrin

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



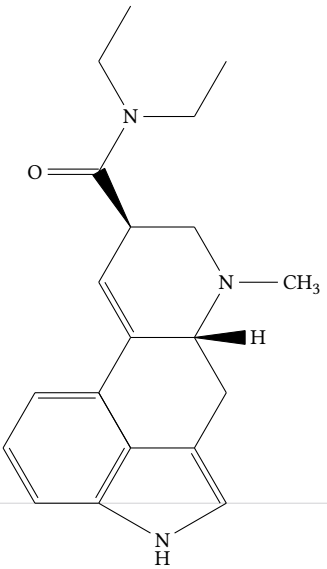
Penicillin

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



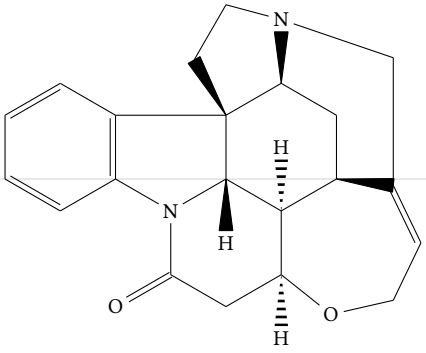
LSD

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



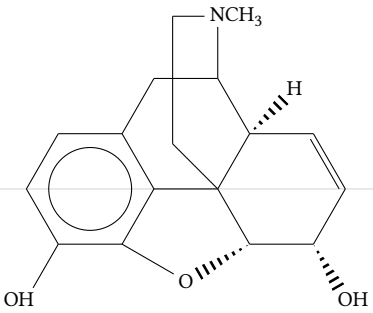
Strychnine

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



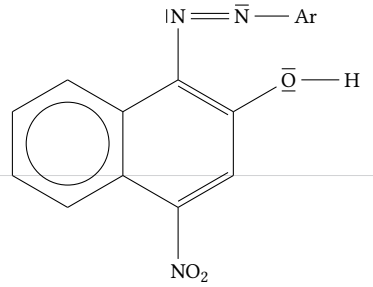
Codeine

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



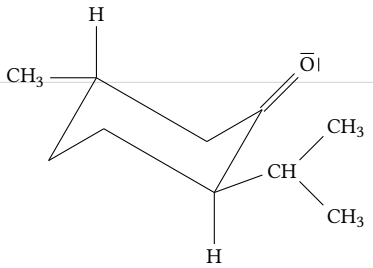
Dye (red)

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



Menthone

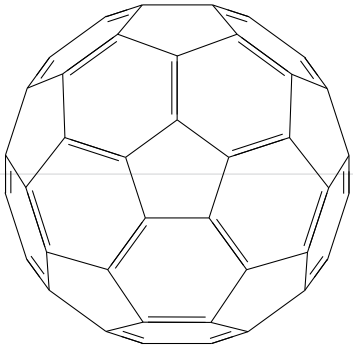
```
\chemfig{CH_3-?(-[2]H)(-[::-30,2]-[::+60](=[1]\charge{0=\|,90=\|}{O})
-[::-150,1.5](-[:20]CH(-[1]CH_3)(-[7]CH_3))(-[6]H)-[::-90,2]-[::+60]?)}
```



Fullerene

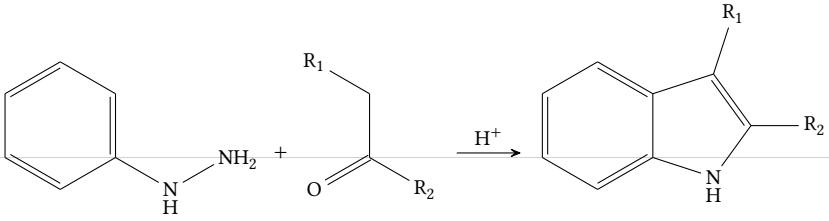
```
\definesubmol\fragment1{
    (-[:#1,0.85,,,draw=none]
    -[::126]-[::-54](=_#(2pt,2pt)[::180])
    -[::-70](-[::-56.2,1.07]=^#(2pt,2pt)[::180,1.07])
    -[::110,0.6](-[::-148,0.60](=^[::180,0.35])-[::-18,1.1])
    -[::50,1.1](-[::18,0.60]=_[::180,0.35])
    -[::50,0.6]
    -[::110])
}
\chemfig{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

```
}
```



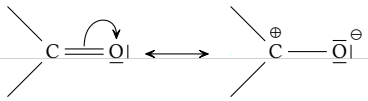
Fischer indole synthesis

```
\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
```



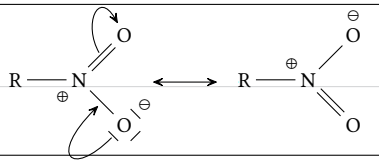
Reaction mechanism: carbonyl group

```
\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);}
```



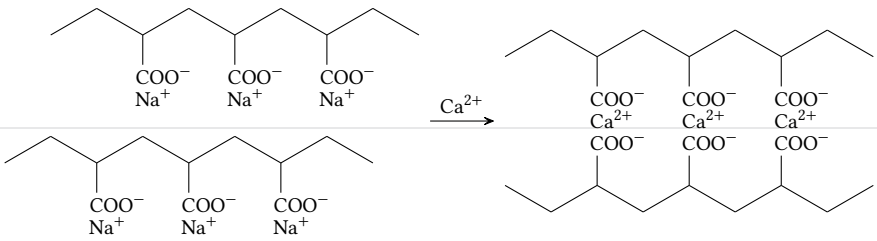
Reaction mechanisms: nitro compounds

```
\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);
}
```



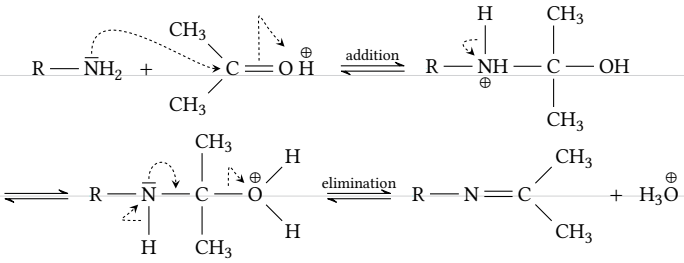
Calcium alginate gel

```
\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

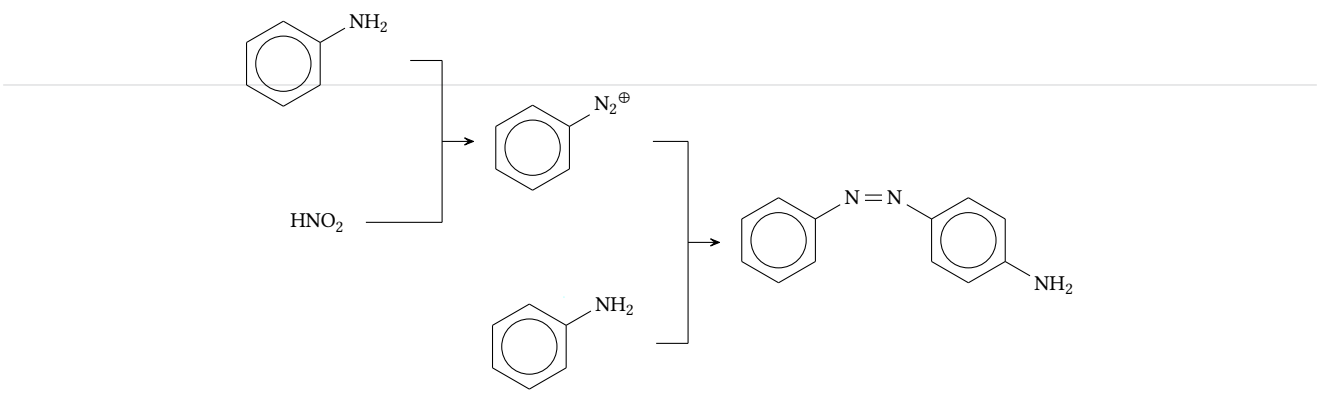
```
\setchemfig{atom sep=2.5em,compound sep=5em}
\schemestart
    \chemfig{R-@{aton}\charge{90=\|}{N}H_2}
    \+
    \chemfig{@{atoc}C([3]-CH_3)([5]-CH_3)=[@{atoo1}]O}
    \chemfig{@{atoo2}\chemabove{H}{\scriptstyle\oplus}}
    \chemmove[-stealth,shorten <=3pt,dash pattern= on 1pt off 1pt,thin]{
   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

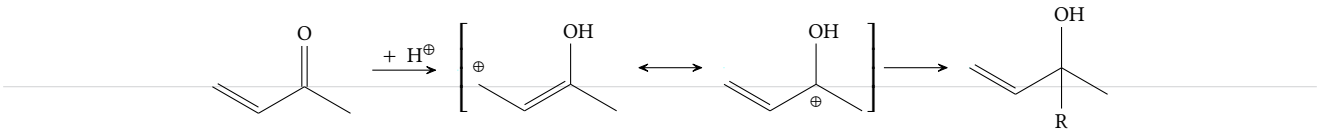
```
\setchemfig{atom sep=2em}
\schemestart[arrow angle=-90]
    \chemfig{**6(---(-NH _2)---)}\arrow{0}\chemfig{HNO_2}
    \merge(c1)(c2)--()
    \chemfig{**6(---(-N_2|{}^\oplus)---)}\arrow{0}\chemfig{**6(---(-NH _2)---)}
    \merge(c3)(c4)--()
    \chemfig{**6(---(-N=[::-30]N-[::-30]**6(---(-NH_2)---))---)}
```

```
\schemestop
```



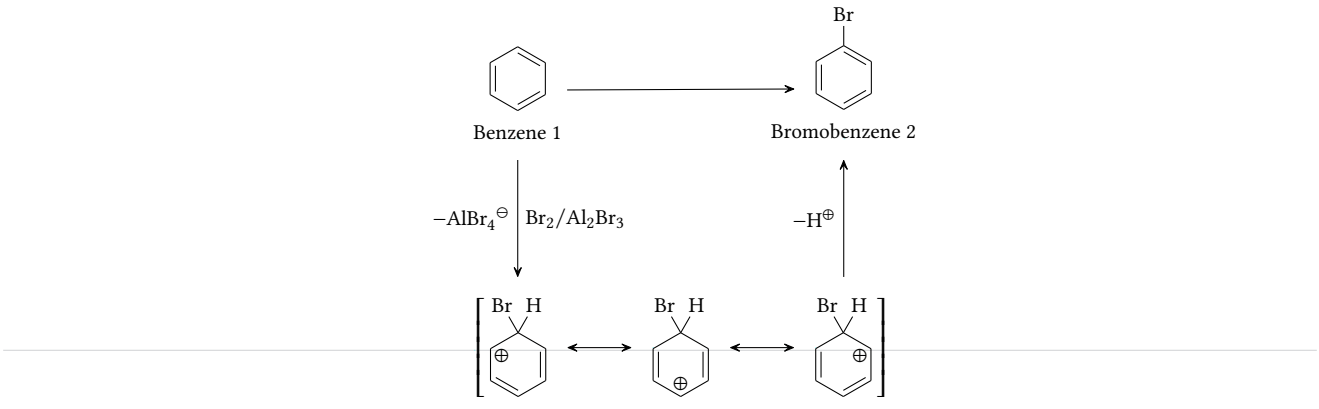
Addition reaction

```
\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(@c4--)\chemfig{*6(=-(-)(-[6]R)-OH)}
\schemestop
```



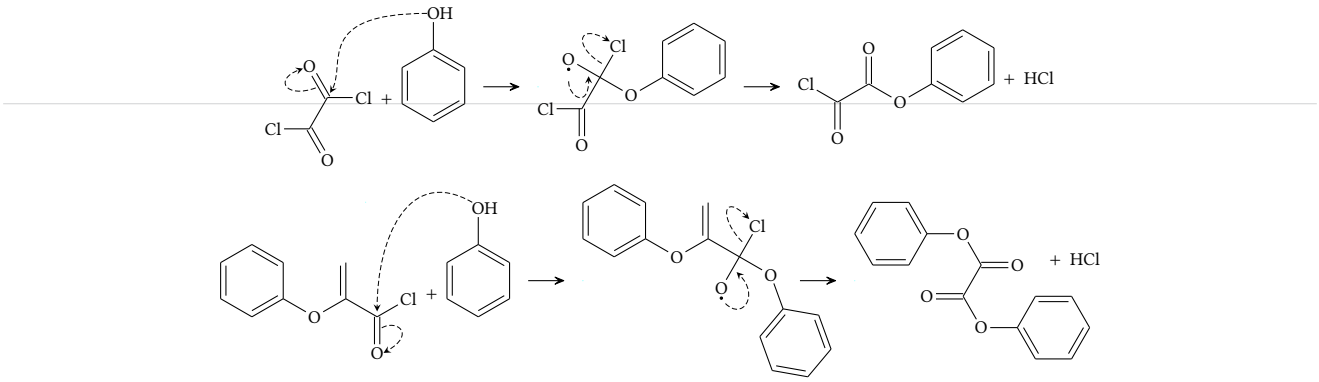
Electrophilic aromatic substitution

```
\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)-=-)}}{Bromobenzene 2}
    \arrow(@c5.mid east--@c6.mid west)
\schemestop
\chemnameinit{}
```



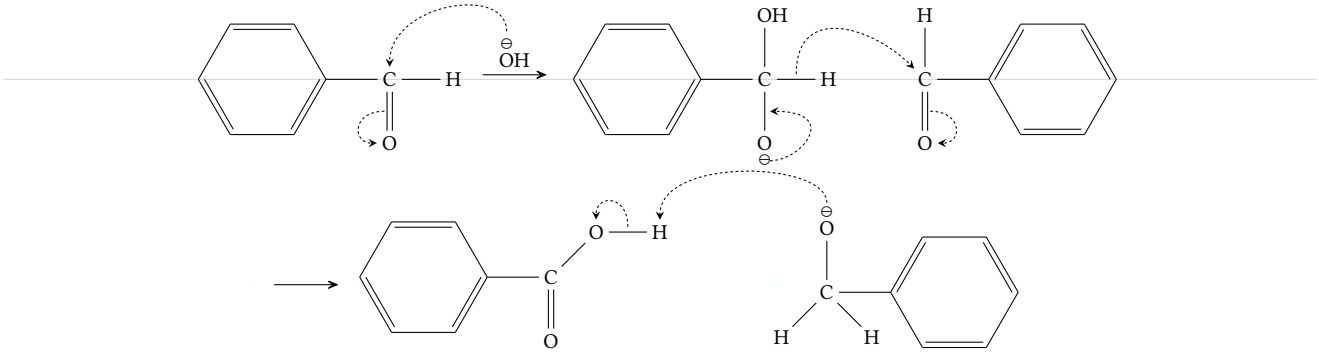
Reaction mechanisms of chlorination

```
\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);}
```



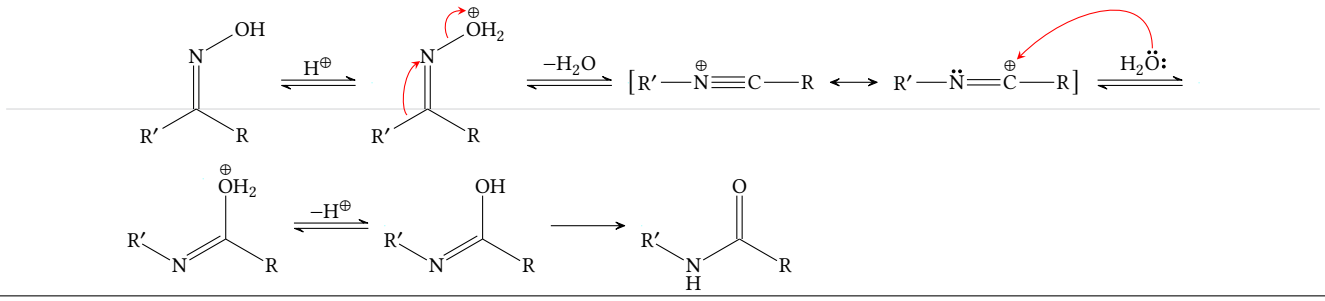
Cannizzaro

```
\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)-=-)}
    \hskip1cm \relax
    \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
```



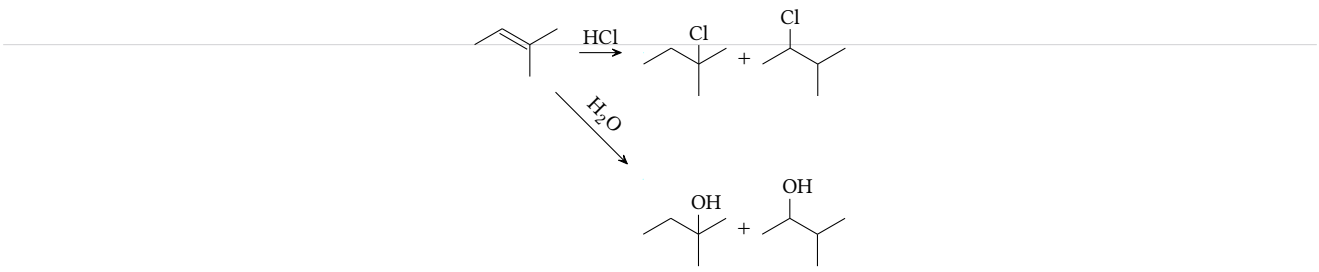
Beckmann rearrangement

```
\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

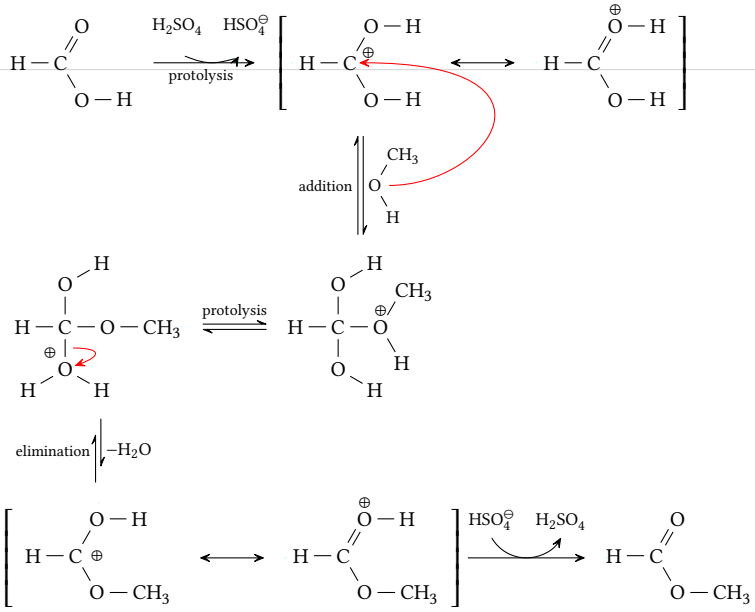
```
\setchemfig{atom sep=1.5em,compound sep=4em}
\schemestart
    \chemfig{-[::30]=_[::-60](-[:: -60])-[::60]}
    \arrow{->[\chemfig{HCl}]}
    \chemfig{-[::30]-[::-60](-[::120]Cl)(-[::-60])-[::60]}\+\chemfig{-[::30](-[::60]Cl)-[::-60](-[::-60])-[::60]}
    \arrow(@c1--.north west){->[\chemfig{H_2O}]}[-45,1.7]
    \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}
```

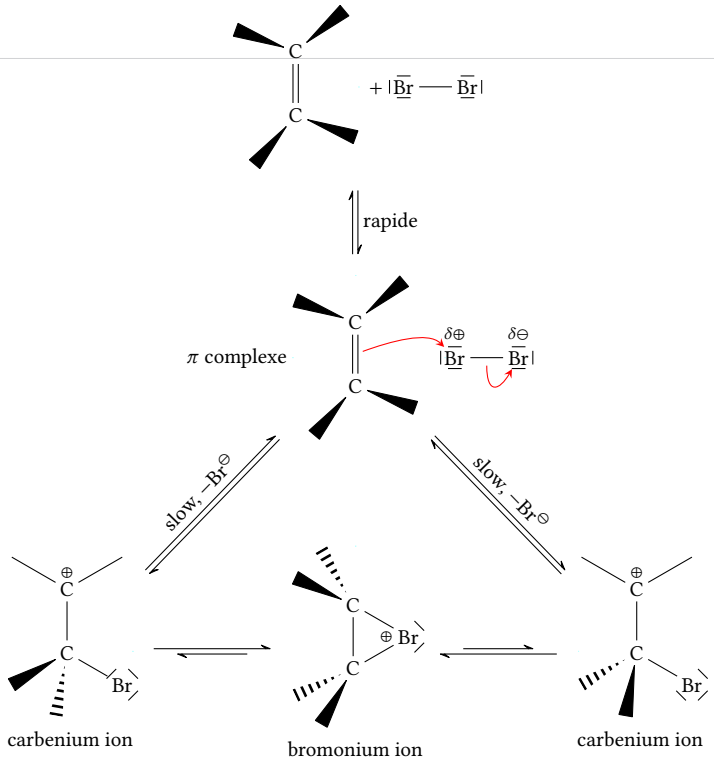
```
    \chemmove[red,shorten <=3pt,shorten >=1pt]{
        \draw(a1)..controls +(0:1.5cm)and+(0:3cm).. (a2);
        \draw(b1)..controls +(0:5mm)and+(20:5mm)..(a3);}
\schemestop
```



Electrophilic addition of a halogen to an olefin

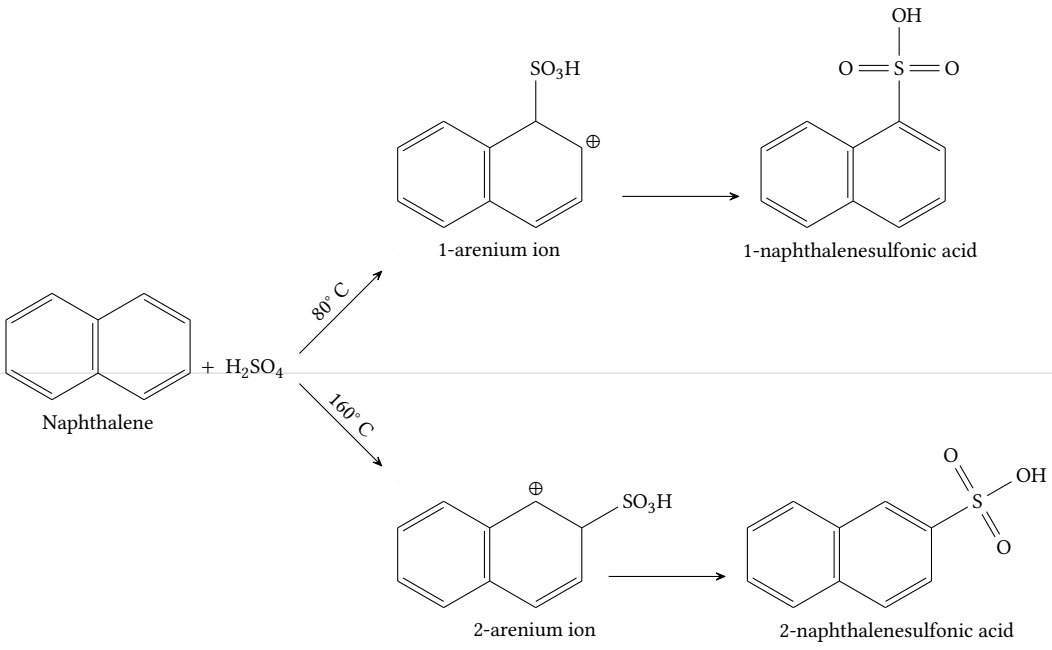
```
\schemestart
    \subscheme{%
        \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
        \arrow{0}[,0]\+\chemfig{\charge{90=\|,180=\|,270=\|}{Br}-\charge{90=\|,0=\|,270=\|}{Br}}}
    \arrow(@c1--olefin){<=>[*{0}rapide]}[-90]
    \chemfig{>[:-20]C(<[:40])=[@{db}6]C(<[:-130])<[:-20]}
    \arrow(--bromonium){0}[-90]
    \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}}}{carbenium 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){<=>[slow, \chemfig{{-}Br^\ominus}]}
    \arrow(@carbeniumB--@olefin){<=>[slow, \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{}
```



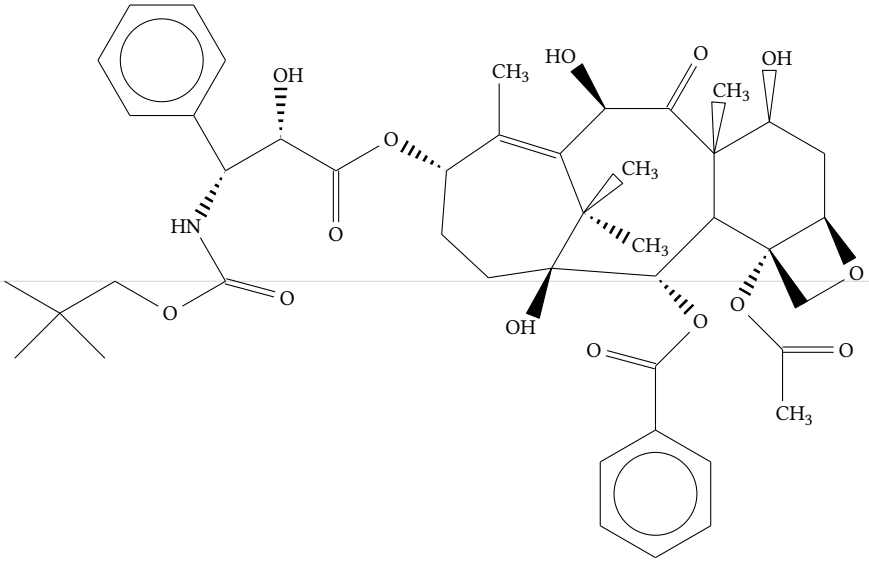
Sulfonation of naphthalene

```
\definesubmol\cycleoplus{-[,0.25,,,draw=none]\oplus}
\definesubmol{so2oh}{S(=[::90]O)(=[::-90]O)-OH}
\setchemfig{atom sep=2.5em}
\schemestart[arrow coeff=1.5]
    \chemname{\chemfig{*6(=-*6(-==-)=-=-)}}{Naphthalene}\+\chemfig{H_2SO_4}
    \arrow(nph.mid east--.south west){->[80° C]}[45]
    \chemname{\chemfig{*6(=-*6(-=-(!\cycleoplus)-(-SO_3H)-)=-=-)}}{1-arenium ion}
    \arrow(.mid east--.mid west)
    \chemname{\chemfig{*6(=-*6(-=-=(-!{so2oh})-)=-=-)}}{1-naphthalenesulfonic acid}
    \arrow(@nph.mid east--.north west){->[160° C]}[-45]
    \chemname{\chemfig{*6(=-*6(-=-(-SO_3H)-(!\cycleoplus)-)=-=-)}}{2-arenium ion}\kern-4em
    \arrow(.mid east--.mid west)
    \chemname{\chemfig{*6(=-*6(-=-(-!{so2oh})=-)=-=-)}}{2-naphthalenesulfonic acid}
\schemestop
\chemnameinit{}
```



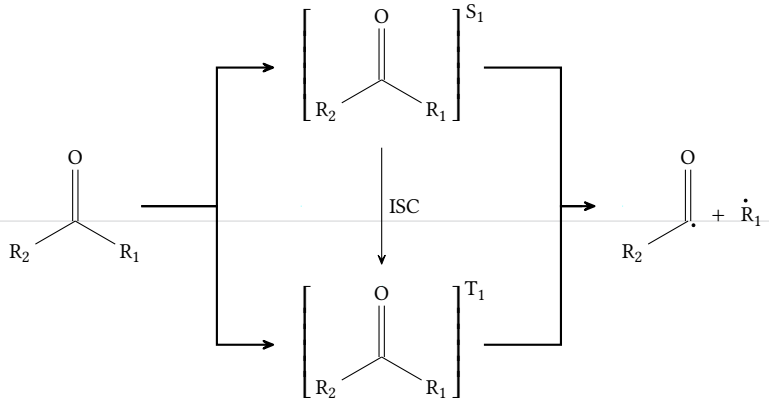
Taxotere

```
\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=$\rm S_1$}{\chemleft{[}\chemfig{(-[:210]R_2)(-[:330]R_1)=[2]O}\chemright{]}}
        \arrow(b--c){->[*{0}ISC]}[-90,1.5]
        \charge{30:4pt=$\rm T_1$}{\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);
}
```



10 Change history

Changes (most recent first) made to chem*fig* since the first version. *The change history below is reproduced from the French manual and has not been translated.*

Version 1.84 du 14/11/2026

- le manuel en anglais refait son apparition, merci à Evys Ancede Gallardo qui me l'a spontanément envoyé sans quoi il aurait sans doute fallu attendre plusieurs mois
- quelques erreurs ont été corrigées dans le manuel

Version 1.83 du 07/11/2026

- lorsque `bond join` = ⟨*true*⟩, les jointures entre liaisons Cram-Cram et Simple-Cram sont améliorés
- la macro `\polymerdelim` est prise en charge dans l'environnement `\hreac...\endhreac` pour que les espaces dans son argument optionnel soient préservés
- le manuel est composé avec OpTEX afin de se tenir éloigné de la lourdeur et la complexité toujours croissantes de LaTeX
- le manuel en anglais est abandonné (au mieux provisoirement)

Version 1.82 du 19/09/2026

- 2 erreurs dans le manuel corrigées
- bug : avec `\chemfig{*6((-A)(-B)--(-C)(-D)----)}`, les liaisons vers A et B ne se superposent pas comme le font celles vers C et D

Version 1.81 du 01/09/2026

- la macro `\?`, let-égale à `\relax` par babel french sous pdflatex terminait prématurément la macro `\CF_setcatcodes_a`

Version 1.8 du 30/08/2026

- la clé booléenne `save` permet d'exporter dans un fichier les instructions tikz ayant permis de tracer une molécule. La clé `savet name` donne la possibilité de choisir le nom donné au fichier
- la clé `arrow coeff` prise en compte dans l'environnement `hreac`
- les clés `arrow head` et `arrow style` sont prises en compte dans l'environnement `hreac`
- tout type de flèche défini par `\definarrow` est admis dans l'environnement `hreac`
- comportement incohérent du ⟨*style*⟩ lorsque une flèche ayant le type «`@`⟨*nom*⟩`[`⟨*style*⟩`]--`» est tracée dans l'environnement `\schemestart`

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} > Cendhreac` 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 `1.5ex` 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[1].
  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.

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

[1] dysfonctionnement non pas aimablement signalé par email mais sur 2 sites (tex.stackexchange et reddit) par Joseph Wright avec sa désormais amusante habitude d'exprimer publiquement le mal qu'il pense de mes packages.

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 `\setcrambon` 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 :
  `-U>[`⟨*a*⟩`][`⟨*b*⟩`][`⟨*d*⟩`][`⟨*r*⟩`][`⟨*a*⟩`]` 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 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* `[,`⟨*l*⟩`]` *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 `\chemskipalig` 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