L'extension pour TEX

chemfig

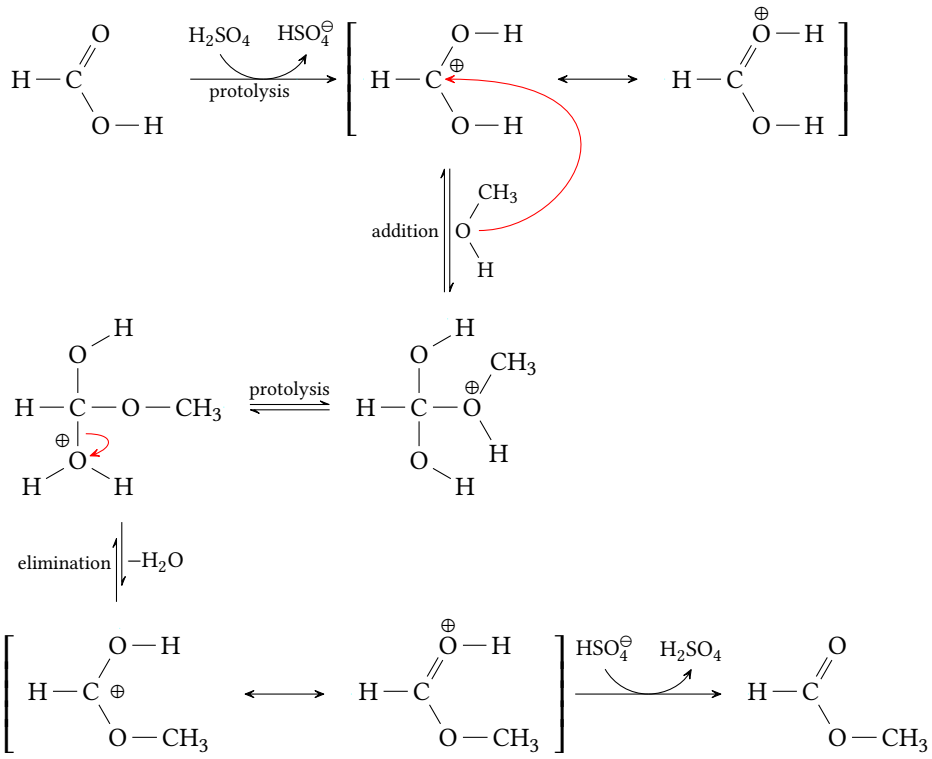
v1.84

14 novembre 2026

Manuel de l'utilisateur

Christian Tellechea

unbonpetit@netc.fr



1 La commande `\chemfig` 4
1.1 Nouveau dans la version 1.84 . . . 4
1.2 Syntaxe . . . 4
1.3 Paramètres de `\chemfig` . . . 4

2 Atomes et liaisons 6
2.1 Atomes et groupes d'atomes . . . 6
2.2 Rôle du premier atome . . . 6
2.3 Liaisons entre groupes d'atomes . . . 6
2.4 Argument optionnel d'une liaison . . . 7
2.4.1 Angle d'une liaison : angles prédéfinis . . . 7
2.4.2 Angle d'une liaison : angles absolus . . . 8
2.4.3 Angle d'une liaison : angles relatifs . . . 8
2.4.4 Longueur des liaison : rôle de l'encombrement des atomes . . . 8
2.4.5 Longueurs des liaisons fixes . . . 8
2.4.6 Longueur d'une liaison : réglage par argument optionnel . . . 9
2.4.7 Atomes de départ et d'arrivée . . . 9
2.4.8 Personnalisation d'une liaison . . . 10
2.5 Raccords entre liaisons . . . 10
2.6 Valeurs par défaut pour une molécule . . . 11
2.7 Ramifications . . . 11
2.7.1 Principe . . . 11
2.7.2 Exemple : la molécule d'anhydride d'acide . . . 12
2.8 Lier des atomes éloignés . . . 12

3 Cycles 14
3.1 Syntaxe . . . 14
3.2 Position angulaire . . . 15
3.3 Ramification partant d'un cycle . . . 15
3.4 Cycles accolés . . . 16
3.5 Groupes d'atomes dans un cycle . . . 17

4 Représentation des déplacements d'électrons 18
4.1 Effet mésomère . . . 18
4.2 Centre d'un cycle . . . 19
4.3 Mécanismes réactionnels . . . 20
4.4 Écrire un nom sous une molécule . . . 20

5 Utilisation avancée 23
5.1 Découpage des atomes . . . 23
5.2 Mise en forme des atomes . . . 23
5.3 Paramètres passés à ti*k*z . . . 24
5.4 Alignement vertical . . . 24
5.4.1 La clé `baseline` . . . 24
5.4.2 Importance de la hauteur ou profondeur d'un atome . . . 25
5.4.3 La macro `\chemskipalign` . . . 26
5.5 Liaisons doubles déportées . . . 27
5.6 Liaisons doubles délocalisées . . . 27
5.7 Sauvegarde d'une partie de molécule avec `\definesubmol` . . . 28
5.8 La macro `\charge` . . . 29
5.8.1 Présentation . . . 29
5.8.2 Paramètres . . . 29
5.9 Empilement de caractères . . . 31
5.10 Sauvegarde dans un fichier . . . 31
5.11 Comment faire ? . . . 32
5.11.1 Écrire un atome en couleur . . . 32
5.11.2 Ajouter un exposant sans influer sur une liaison . . . 32
5.11.3 Dessiner une liaison courbe . . . 32
5.11.4 Dessiner un élément de polymère . . . 33
5.11.5 Dessiner le symétrique d'une molécule . . . 35
5.11.6 Ajouter du texte au dessus des liaisons et coder les angles . . . 35
5.11.7 Dessiner des liaisons multiples . . . 36
5.12 Exemples commentés . . . 36

5.12.1 L'éthanal . . . 36
5.12.2 L'acide 2-amino-4-oxohexanoïque . . . 37
5.12.3 Glucose . . . 38
5.12.4 Adrénaline . . . 40
5.12.5 Guanine . . . 42

6 Schémas réactionnels **44**
6.1 Généralités . . . 44
6.2 Types de flèches . . . 45
6.3 Caractéristiques des flèches . . . 45
6.4 Nom des composés . . . 46
6.5 Ancres . . . 47
6.6 Style des composés . . . 48
6.7 Embranchements . . . 48
6.8 Sous-schéma . . . 49
6.9 Arguments optionnels des flèches . . . 51
6.10 Créer ses propres flèches . . . 53
6.10.1 Une première flèche . . . 54
6.10.2 Une flèche courbe . . . 54
6.11 La commande `\merge` . . . 55
6.12 Le signe + . . . 57

7 Réactions horizontales **59**
7.1 Syntaxe . . . 59
7.2 Placement des composés . . . 59
7.3 Nom des composés . . . 60
7.4 Types de flèches . . . 60
7.5 Label des flèches . . . 60

8 Liste des commandes **62**

9 Exemples **64**

10 Historique des changements **79**

1 La commande \chemfig

Pour l'utiliser l'extension « chem*fig* », il faut exécuter :

— `\input chemfig.tex` ou `\input{chemfig.tex}` lorsqu'on utilise (pdf)(Xe)(lua)TEX ;

— `\usepackage{chemfig}` dans le préambule lorsqu'on utilise (pdf)(Xe)(lua)LATEX.

Les extensions `tikz` et `simplekv` sont requises et chargées si cela n'a pas été le cas.

Cette extension a pu voir le jour grâce à l'aide de Christophe Casseau qui en a eu l'idée. Je le remercie pour l'aide qu'il m'a apportée en amont de l'écriture du code ainsi que pour les tests qu'il a effectués.

Je souhaite également remercier chaleureusement Evys Ancede Gallardo qui m'a spontanément envoyé la traduction de ce manuel en anglais.

1.1 Nouveau dans la version 1.84

Ci-dessous figurent les changement dans la version 1.84 ; pour consulter les changements antérieurs, voir en fin de manuel page 79.

1. Le manuel en anglais est de nouveau disponible, merci à Evys Ancede Gallardo.
2. Quelques erreurs corrigées dans le manuel.

1.2 Syntaxe

La macro `\chemfig` s'emploie selon cette syntaxe :

```
\chemfig[⟨clés⟩=⟨valeurs⟩]{⟨code de la molécule⟩}
```

L'argument optionnel entre crochets spécifie les réglages des paramètres qui seront utilisés pour le tracé de la molécule. Ces paramètres ne s'appliquent *que pour la molécule en cours* et seront restaurés à leurs valeurs précédentes une fois la molécule tracée. Pour modifier des paramètres pour le restant du document, il faut exécuter

```
\setchemfig{⟨clés⟩=⟨valeurs⟩}
```

Le ⟨*code de la molécule*⟩ contient les instructions pour tracer la molécule selon une syntaxe qui sera expliquée dans ce document. Il n'y a pas de restriction a priori sur les caractères acceptés dans ce code :

— tous les caractères de catcode 11 ou 12, c'est-à-dire les lettres majuscules ou minuscules, les chiffres, les opérateurs mathématiques (`+ - * / =`), les signes de ponctuation qu'ils soient actifs ou non (`. , ; : ! ? ’ ‘ ” |`-), les parenthèses et les crochets ;

— les caractères plus spéciaux tels que « ~ », « # »[1] ainsi que « ^ » et « _ » qui ont leur propriétés normales du mode mathématique ;

— les espaces, mais ceux-ci sont ignorés par défaut car les atomes sont composés en mode mathématique ;

— les accolades « { » et « } » qui ont leur comportement normal de marqueurs de groupe ou délimiteurs d'argument de macro ;

— des macros.

1.3 Paramètres de \chemfig

Le tableau ci-dessous donne toutes les `clés` = ⟨*valeurs*⟩ permises dans l'argument optionnel de `\chemfig` ou dans l'argument obligatoire de `\setchemfig`. Chacune d'entre elles sera plus précisément décrite plus loin dans le manuel :

clés	valeurs par défaut	Description
`chemfig style`	⟨*vide*⟩	style passé à ti*k*z, cf. p 24
`use atom strut`	`false`	met dans l'argument de `\printatom` le strut de l'atome précédent afin de corriger des défauts d'alignement entre atomes
`atom style`	⟨*vide*⟩	style des nœuds contenant les atomes, cf. p 24
`bond join`	`false`	si ⟨*true*⟩, raccorde de façon plus esthétique des liaisons jointives, cf. p 10

[1] Pour éviter que # ne soit doublé lorsque la macro `\chemfig` se trouve dans l'argument d'une macro, on peut utiliser à la place `\#` ou la macro `\CFhash`.

clés	valeurs par défaut	Description
`<> join angle`	`100`	si `bond join` = ⟨*true*⟩, différence d'angle au delà de laquelle un chanfrein est tracé pour raccorder 2 liaisons de Cram, cf. p 10
`-> join angle`	`45`	si `bond join` = ⟨*true*⟩, différence d'angle en deçà de laquelle une liaison simple n'est pas jointe à une liaison de Cram, cf. p 10
`fixed length`	`false`	si ⟨*true*⟩, impose des longueurs de liaisons fixes, cf. p 8
`cram rectangle`	`false`	si ⟨*true*⟩, trace les Liaisons de Cram sous forme de rectangle, cf. p 6
`cram width`	`1.5ex`	dimension de la base des triangles des liaisons de Cram, cf. p 6
`cram dash width`	`1pt`	largeur des pointillés des liaisons de Cram, cf. p 6
`cram dash sep`	`2pt`	espacement des pointillés des liaisons de Cram, cf. p 6
`atom sep`	`3em`	espacement entre atomes, cf. p 7
`bond offset`	`2pt`	espacement entre l'atome et la liaison, cf. p 7
`double bond sep`	`2pt`	espacement entre les traits des liaisons multiples
`angle increment`	`45`	incrément de l'angle des liaisons, cf. p 7
`node style`	⟨*vide*⟩	style des atomes, cf. p 24
`bond style`	⟨*vide*⟩	style des liaisons, cf. p 7
`baseline`	`0pt`	dimension ou nom du nœud pour régler la position verticale de la molécule, cf. p 24
`debug`	`false`	si ⟨*true*⟩, affichage des frontières des éléments dessinés et des noms des atomes
`cycle radius coeff`	`0.75`	coefficient du cercle ou de l'arc de cercle tracé dans les cycles
`stack sep`	`1.5pt`	espacement vertical pour les arguments des macros `\chemabove` et `\chembelow`, cf. p 31
`show cntcycle`	`false`	affichage des numéros des cycles
`autoreset cntcycle`	`true`	remise à 0 du compteur de cycles à chaque exécution de `\chemfig`
`gchemname`	`true`	si ⟨*true*⟩, rend l'assignation de la plus grande profondeur globale pour le placement des noms de molécules
`save`	`false`	si ⟨*true*⟩, sauvegarde dans un fichier les instructions ti*k*z générées par la macro `\chemfig`, cf. p 31
`save name`	`\jobname -fig*`	spécifie le nom du fichier, cf. p 31

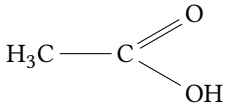
La macro `\resetchemfig` restaure toutes les les clés à leur valeur par défaut.

2 Atomes et liaisons

2.1 Atomes et groupes d'atomes

Intrinsèquement, le dessin d'une molécule chimique consiste à relier des atomes par des traits de différents types. Ainsi, dans la molécule O═O, il y a 2 groupes d'atomes, chacun constitué d'un seul atome « O ».

Mais dans cette molécule

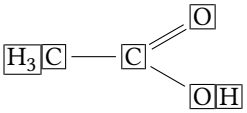


on compte 4 groupes d'atomes : « H_3C », « C », « O » et « OH ». Pour des raisons que nous verrons plus tard, chem*fig* examine chaque groupe d'atomes et le découpe en atomes. Chaque atome s'étend jusqu'à rencontrer une lettre majuscule ou un de ces caractères spéciaux :

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

Pour chem*fig*, tous les caractères entre accolades sont ignorés pour le découpage en atomes.

Par conséquent, le premier groupe d'atomes « H_3C » est découpé en 2 atomes : `H3` et `C`. Chimiquement, il arrive que ce ne soient pas de vrais atomes puisque H_3 par exemple, est constitué de 3 atomes d'hydrogène. Par abus de langage, on parlera d'atomes par la suite. Par conséquent, chem*fig* voit la molécule précédente ainsi :

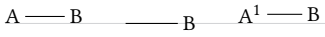


2.2 Rôle du premier atome

Il est important de comprendre que le placement de la molécule entière dépend du premier atome placé, c'est-à-dire le premier atome du premier groupement d'atomes. Pour ce premier atome, l'ancre d'attache de ti*k*z « `base east` » est placé sur la ligne de base de la ligne en cours (représentée en gris dans les exemples de ce manuel).

Influence du premier atome

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



Pour spécifier un décalage vertical arbitraire ou placer un atome sur la ligne de base, il faut utiliser la clé `baseline` (voir page 24).

2.3 Liaisons entre groupes d'atomes

Pour chem*fig*, les liaisons entre 2 groupes atomes sont de 9 types, correspondant aux caractères `-`, `=`, `~`, `>`, `<`, `>:`, `<:`, `>|` et `<|` :

	Code	Résultat	Type de liaison
1	`\chemfig{A-B}`	A—B	Simple
2	`\chemfig{A=B}`	A═B	Double
3	`\chemfig{A~B}`	A≡B	Triple
4	`\chemfig{A>B}`	A▶B	Cram pleine droite
5	`\chemfig{A<B}`	A◀B	Cram pleine gauche
6	`\chemfig{A>:B}`	A⁞B	Cram pointillée droite
7	`\chemfig{A<:B}`	A⁞B	Cram pointillée gauche
8	`\chemfig{A>|B}`	A▷B	Cram évidée droite
9	`\chemfig{A<|B}`	A◁B	Cram évidée gauche

Les clés suivantes permettent de régler l'apparence des liaisons :

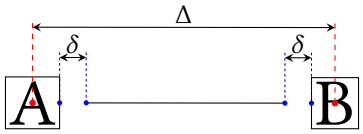
— `double bond sep` = ⟨*dim*⟩ règle l'espacement entre les traits des liaisons doubles ou triples. Cet espacement vaut 2pt par défaut

— `cram rectangle` = ⟨*booléen*⟩, trace les liaisons de Cram sous forme de rectangle lorsque ⟨*true*⟩

— `cram width` = ⟨*dim*⟩ spécifie la largeur de la base des triangles des liaisons de Cram

— `cram dash width` = ⟨*dim*⟩ et `cram dash sep` = ⟨*dim*⟩ définissent la largeur des pointillés et leur espacement dans les liaisons de Cram

Lorsqu'une liaison est faite entre 2 atomes, ces derniers sont contenus dans des boîtes invisibles rectangulaires. Les centres des deux rectangles sont séparés par une distance réglable Δ, appelée « distance interatome ». De plus, les liaisons ne relient pas exactement les frontières des rectangles : une distance δ, réglable elle aussi, sépare le bord des rectangles du début et de la fin de la liaison. Pour aider à la compréhension, les boîtes rectangulaires sont rendues visibles dans ce schéma :



La clé `atom sep` = ⟨*dim*⟩ règle la longueur Δ et `bond offset` = ⟨*dim*⟩ permet de régler l'espacement δ. Si deux liaisons se suivent, alors chem*fig* insère un groupe vide `{}`, et autour de ce groupe vide, l'espacement δ est nul.

Réglages des liaisons

```
Longueur $\Delta$ :
    \chemfig[atom sep=2em]{A-B} et
    \chemfig[atom sep=50pt]{A-B}\par
Longueur $\delta$ :
    \chemfig[bond offset=0pt]{A-B} et
    \chemfig[bond offset=5pt]{A-B}\par
Groupes vides : \chemfig{A-B=-=C}
```

Longueur Δ : A — B et A ——————— B
Longueur δ : A———B et A — B
Groupes vides : A —— B ═══———═══ C

On peut spécifier l'espacement δ pour une seule liaison avec un marqueur qui est, au choix, le caractère `#`, la macro `\#` ou `\CFhash`. Si la macro `\chemfig` se trouve dans l'argument d'une autre macro, il *faut* utiliser la macro `\#` ou `\CFhash` au lieu du caractère `#`.

Ce marqueur doit se situer *immédiatement* après le signe de liaison et dispose d'un argument obligatoire entre parenthèses de la forme « `(`⟨*dim1*⟩`,`⟨*dim2*⟩`)-` », où ⟨*dim1*⟩ représente l'espacement δ au début de la liaison et ⟨*dim2*⟩ celui de la fin. Si ⟨*dim2*⟩ est omis, l'espacement en fin de liaison prend la valeur de δ en vigueur à ce moment. On peut observer sur cet exemple comment le retrait, réglé à 4pt pour être plus visible, est rendu nul pour la liaison arrivant sur « B », puis partant de « B » et enfin pour les deux à la fois :

Réglage fin du retrait des liaisons

```
\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

La clé `bond style` = ⟨*code tikz*⟩ définit le style des liaisons. Le ⟨*code tikz*⟩ est vide par défaut. Pour personnaliser les liaisons une par une, cf. p 10.

Style des liaisons

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

A — B ═ C ▷ D ◁ E ⁞ F

Par défaut, tous les atomes se trouvant dans les groupes d'atomes sont composés en mode mathématique et donc, les espaces sont ignorés. Ils peuvent donc contenir des caractères propres à ce mode comme la mise en indice ou en exposant.

2.4 Argument optionnel d'une liaison

Après chaque liaison, il est possible d'écrire un argument optionnel entre crochet qui contient 5 champs séparés par des virgules :

`[`⟨*angle*⟩`,`⟨*longueur*⟩`,`⟨*atome départ*⟩`,`⟨*atome arrivée*⟩`,`⟨*code tikz*⟩`]`

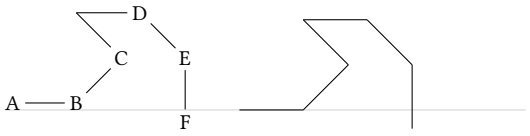
Ces champs doivent être écrits dans cet ordre, mais peuvent être omis si après un champ, on ne spécifie pas les suivants.

2.4.1 Angle d'une liaison : angles prédéfinis

Lorsque ⟨*angle*⟩ est un entier, l'angle que fait la liaison avec l'horizontale sera 45° multiplié par cet entier :

Angles prédéfinis

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



La clé `angle increment` = ⟨*angle*⟩ permet de choisir un autre angle que celui de 45° sélectionné par défaut :

Modification de l'angle prédéfini

```
Par défaut (45) : \chemfig{-[1]-[-1]-[1]-[-1]}\par
Angle de 30° : \chemfig[angle increment=30]{-[1]-[-1]-[1]-[-1]}
```

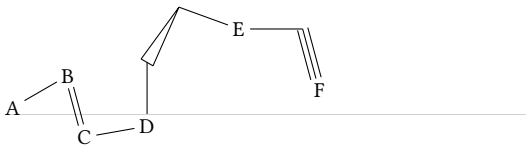


2.4.2 Angle d'une liaison : angles absolus

Si on veut spécifier un angle en degrés avec l'horizontale, alors l'argument optionnel doit prendre cette forme : `[:`⟨*angle*⟩`]`. S'il le faut, chem*fig* ramène ⟨*angle*⟩ dans l'intervalle [0 ; 360[:

Angles absolus

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



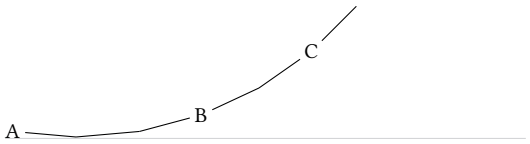
2.4.3 Angle d'une liaison : angles relatifs

Pour spécifier l'angle que fait une liaison avec la précédente, il faut utiliser la syntaxe `[::`⟨*angle*⟩`]`.

Dans cet exemple, les angles des liaisons augmentent de 10° après chacune, avec une première première liaison dont l'angle est absolu de −5° :

Suite d'angles relatifs

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



On peut « casser » un enchaînement d'angles relatifs en mettant un angle absolu ou prédéfini lorsqu'on le souhaite. Ici, c'est à l'atome « B » dont la liaison suivante un angle absolu de −25°.

Suite d'angles relatif puis angle absolu

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



2.4.4 Longueur des liaison : rôle de l'encombrement des atomes

La distance interatome Δ est réglable via la clé `atom sep`. Une fois ce paramètre fixé, la longueur d'une liaison dépend du contenu des atomes et, dans une moindre mesure, de l'angle que fait la liaison avec l'horizontale. Deux atomes peu « encombrants » auront les bords de leur boite englobante plus lointains que dans le cas contraire. L'effet sur la liaison apparait dans exemple où les atomes « I » sont plus étroits que les atomes « M » ce qui entraîne que la liaison entre les « I » est plus longue que celle entre les « M ». L'effet est encore plus sensible – au point d'être indésirable — lorsqu'un atome comporte plusieurs lettres ou contient un exposant.

Influence de la largeur de l'atome

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



2.4.5 Longueurs des liaisons fixes

On peut changer le comportement de chem*fig* quant à l'espace interatome. En effet, lorsque la clé `fixed length` est mise à ⟨*true*⟩, la clé `atom sep` ne spécifie plus la distance entre les centres des atomes, notée Δ, mais *la longueur des liaisons*. Dès lors, les liaisons ont des longueurs fixes tandis que la distance entre les centres des atomes devient variable et dépend de leur encombrement. Voici deux schémas qui illustrent les valeurs booléennes de `fixed length` :



Liaisons de longueur fixe

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

Cl — Cl
Cl —— Cl

Afin que les cycles (cf. p 14) soient des polygones réguliers, le comportement par défaut est rétabli pour les liaisons des cycles, même si `fixed length` = ⟨*true*⟩.

2.4.6 Longueur d'une liaison : réglage par argument optionnel

Le deuxième champ de l'argument optionnel d'une liaison est un coefficient décimal *k* qui indique par quelle valeur il faut multiplier la valeur ⟨*atom sep*⟩ pour cette liaison.

Coefficient de longueur

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

A^{++} — B_2 et A^{++} —— B_2
A — B —— C = D

2.4.7 Atomes de départ et d'arrivée

Dans un groupe d'atomes — qui est suscpetible d'en contenir plusieurs —, les atomes sont numérotés de gauche à droite en commençant à 1.

Supposons que l'on veuille relier le groupe « ABCD » au groupe « EFG » avec une liaison. chem*fig* calcule quel atome du premier groupe et quel atome de second groupe il faut relier en fonction de l'angle que fait la liaison avec l'horizontale :

- si l'angle est compris entre −90° et 90° (modulo 360°) ces valeurs étant *non comprises*, alors la liaison se fait entre le dernier atome du premier groupe et de premier atome du second groupe ;
- dans tous les autres cas, la liaison se fait par défaut entre le premier atome du premier groupe et le dernier atome du second groupe.

Dans ces cas, les angles sont dans l'intervalle] − 90° ; 90°[, et où la liaison se fait donc entre D et E :

Atomes liés par défaut

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

EFG — ABCD ABCD — EFG EFG — ABCD

Dans ces autres cas, les angles appartiennent à [90° ; 270°] et donc la liaison se fait entre A et G :

Atomes liés par défaut

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

EFG — ABCD ABCD — EFG EFG — ABCD

Dans certains cas, on peut désirer qu'une liaison parte d'autres atomes que ceux calculés par chem*fig*. On peut forcer un atome de départ ou un atome d'arrivée avec l'argument optionnel de la liaison. Il faut écrire :

`[,,`⟨*n1*⟩`,`⟨*n2*⟩`]`

où ⟨*n1*⟩ et ⟨*n2*⟩ sont les numéros des atomes de départ et d'arrivée souhaités. Il faut que ces atomes existent, sinon, un message d'erreur sera émis.

Atomes liés forcés

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

EFG — ABCD EFG — ABCD EFG — ABCD

2.4.8 Personnalisation d'une liaison

Le 5e champ est passé directement à ti*k*z lorsque la liaison est tracée. On peut y mettre des attributs de couleur comme « `red` », de pointillés comme « `dash pattern=on 2pt off 2pt` », d'épaisseur comme « `line width=2pt` » ou même de décoration si on a chargé une librairie de décoration de ti*k*z. On peut aussi rendre une liaison invisible en écrivant « `draw=none` ». Si on veut spécifier plusieurs attributs, on utilise la syntaxe de ti*k*z en les séparant par une virgule :

Passage de code tikz

```
\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}
```



On a accès aux nombreuses librairies de décoration de ti*k*z. On peut par exemple charger la librairie « `pathmorphing` » en écrivant `\usetikzlibrary{decorations.pathmorphing}` dans le préambule pour tracer des liaisons ondulées :

Liaison ondulée

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



2.5 Raccords entre liaisons

Par défaut, les lignes représentant les liaisons simples ne se raccordent pas, ce qui peut être trop visible et inesthétique lorsque les épaisseurs des lignes sont importantes.

Pour ceux qui trouvent cela « affreux[1] », il est désormais possible de raccorder les liaisons simples entres-elles moyennant un temps de compilation légèrement augmenté. La clé booléenne `bond join` = ⟨*booléen*⟩ permet, lorsqu'elle est ⟨*true*⟩ d'activer cette fonctionnalité. La valeur par défaut est ⟨*false*⟩ qui est le comportement préférable.

Raccord entre liaisons simples

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



Le problème est encore plus évident avec les liaisons de Cram pleines lorsqu'elles se connectent à une liaison simple ou à une autre liaison de Cram. Là encore, mettre `bond join` à ⟨*true*⟩ permet des raccords plus esthétiques.

Raccord avec liaisons de Cram

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



Lorsque `bond join` = ⟨*true*⟩, il est important de noter que la façon de raccorder deux liaisons de Cram consécutives dépend de la différence d'angle entre elles. Si cet angle est trop grand et devient supérieur à la valeur de la clé `<> join angle` qui vaut 100 par défaut, un chanfrein est dessiné au lieu d'une jointure saillante du côté extérieur du raccord. On évite ainsi une trop grande extension spatiale du raccord :

Raccord Cram-Cram

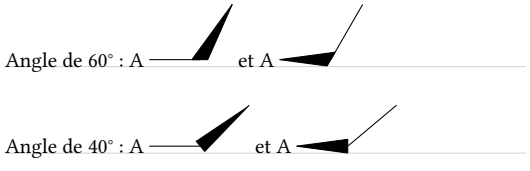
```
Angle de 50° : \chemfig[bond join]{<[:10]>[:60]}
\bigbreak
Angle de 110° : \chemfig[bond join]{<[:20]>[:-90]}
```



Par ailleurs, la jointure entre une liaison simple et une liaison de Cram ne se fait que lorsque la différence d'angle entre ces deux liaisons est supérieure ou égale à un angle limite contenu dans la clé `-> join angle` qui vaut 45 par défaut.

Raccord liaison simple-Cram

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



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

Lorsque l'angle entre une liaison simple et une liaison de Cram pleine est inférieur à `-> join angle`, il néanmoins est possible d'effectuer le raccord entre ces 2 liaisons. Il faut que la base du triangle de la liaison de Cram soit à l'intérieur de l'espace délimité par les 2 bords externes de la liaison simple (supposée suffisamment épaisse pour que cela se produise). Mathématiquement, c'est la cas si $d\cos\alpha > w$ où :

— d est la base du triangle, qui est égale à `cram width` augemntée de la valeur de `line width` pour la liaison de Cram ;

— w est `line width` pour la liaison simple ;

— α est la différence d'angle entre les 2 liaisons.

Dans l'exemple ci-dessous, comme $(6+0.4)\cos 40° < 5$, le raccord se fait lorsque `bond join` = ⟨*true*⟩ :

Raccord liaison simple-Cram avec angle faible

```
\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 Valeurs par défaut pour une molécule

Au début de chaque molécule, les valeurs par défaut des champs optionnels des liaisons sont initialisées. Elles valent :

— 0 pour l'angle des liaisons ;

— 1 pour le coefficient multiplication des longueurs ;

— ⟨*vide*⟩ pour les numéros de départ et d'arrivée des atomes, ce qui laisse à chem*fig* le soin de calculer ceux-ci en fonction de l'angle de la liaison ;

— ⟨*vide*⟩ pour le paramètre passé à ti*kz*.

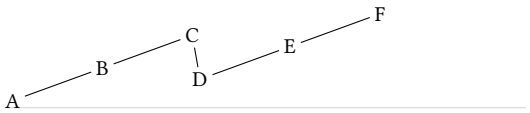
On peut changer ces valeurs par défaut pour toute la molécule en débutant le code de la molécule par

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

Ainsi, si le code de la molécule commence par `[:20,1.5]`, alors toutes les liaisons auront un angle de 20° par défaut et les distances interatomes une longueur qui vaut 1,5 fois la valeur par défaut. On peut à tout moment ignorer ces valeurs par défaut en spécifiant un argument optionnel, comme pour la liaison qui suit l'atome « C » de cet exemple :

Écrasement des valeurs par défaut

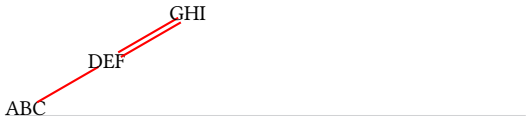
```
\chemfig{[:20,1.5]A-B-C-[:-80,0.7]D-E-F}
```



Si on écrit un improbable `[:30,1.5,2,2,red,thick]`, sauf indication contraire, toutes les liaisons auront un angle de 30° avec l'horizontale, les distances interatomes vaudront 1,5 fois la valeur par défaut, les liaisons partiront et arriveront sur le deuxième atome de chaque groupe et seront rouges et épaisses :

Valeurs par défaut

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



2.7 Ramifications

2.7.1 Principe

Jusqu'à présent, toutes les molécules étaient linéaires, ce qui est rare. On peut attacher une molécule à un atome faisant suivre cet atome d'un ⟨*code*⟩ entre parenthèses.

Dans cet exemple, la molécule « `-[1]W-X` » vient s'attacher sur l'atome « `B` » :

Ramification

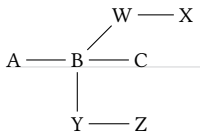
```
\chemfig{A-B(-[1]W-X)-C}
```



On peut avoir plusieurs molécules attachées à un même atome : il suffit de le faire suivre de plusieurs parenthèses contenant le code de chacune :

Ramifications multiples

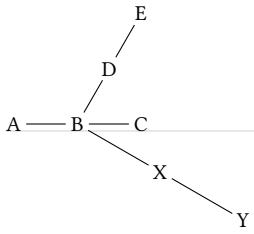
```
\chemfig{A-B(-[1]W-X)(-[6]Y-Z)-C}
```



Le code de chaque molécule attachée peut définir ses propres valeurs par défaut qui seront valables dans toute son étendue. Ici, on attache sur « B » une molécule « `[:60]-D-E` » dont l'angle par défaut vaut 60° absolus. On attache également sur « B » une sous molécule « `[::-30,1.5]-X-<` » dont l'angle par défaut vaut 30° de moins que celui de la liaison précédente (qui était celle qui allait de « A » à « B ») et dont la distance interatome est 1,5 fois celle par défaut :

Valeur par défaut dans les ramifications

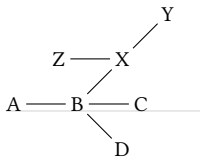
```
\chemfig{A-B([:60]-D-E)([::-30,1.5]-X-Y)-C}
```



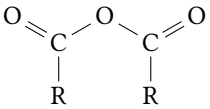
Les molécules attachées peuvent être imbriquées, et les règles vues au paragraphe précédent restent valables :

Ramifications imbriquées

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



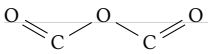
2.7.2 Exemple : la molécule d'anhydride d'acide

Essayons de dessiner la molécule d'anhydride d'acide : 

Une méthode pour y arriver est de choisir le plus long chemin `O=C-O-C=O` en tenant compte des angles et en n'utilisant que les angles relatifs :

Structure de l'anhydride d'acide

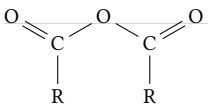
```
\setchemfig{atom sep=2.5em}
\chemfig[atom sep=2.5em]{O=[::-30]C-[::60]O-[::-60]C=[::60]O}
```



Partant de cette structure, il suffit d'attacher deux « `-R` » sur les atomes de carbone, en spécifiant un angle relatif de −60°, car la liaison précédente a déjà un angle relatif de −30° :

Anhydride d'acide

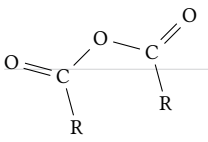
```
\setchemfig{atom sep=2.5em}
\chemfig[atom sep=2.5em]{O=[::-30]C(-[::-60]R)-[::60]O-[::-60]C(-[::-60]R)=[::60]O}
```



N'ayant utilisé que des angles relatifs, on peut faire pivoter cette molécule, en spécifiant un angle par défaut de 15° par exemple :

Rotation de la molécule

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



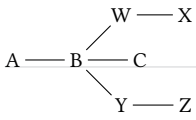
2.8 Lier des atomes éloignés

Nous avons vu comment relier des atomes *qui se suivent dans le code.* Il est parfois indispensable de relier entre-eux des atomes ne se suivant pas dans le code. Appelons « liaisons distantes » ces liaisons particulières.

Dans cette molécule

Structure ramifiée

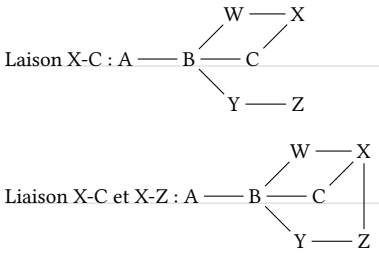
```
\chemfig{A-B(-[1]W-X)(-[7]Y-Z)-C}
```



supposons que l'on veuille relier les atomes `X` et `C`. Dans ce cas, chem*fig* permet de poser un « crochet » *immédiatement* après l'atome qui nous intéresse. Le caractère utilisé pour poser ce crochet est « `?` », pour sa ressemblance avec un crochet. Ainsi, si l'on écrit `X?`, alors, l'atome `X` portera ce crochet. Par la suite dans le code, tous les atomes suivis d'un `?` seront reliés à `X` :

Liaison éloignée

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



Maintenant, imaginons que nous devions laisser ces liaisons distantes `X-C` et `X->`, tout en ajoutant une autre : `A-W`. Il faut donc poser deux crochets *différents*, l'un sur `A` et l'autre sur `X`. En fait, le caractère « `?` » a un argument optionnel :

`?[`⟨*nom*⟩`,`⟨*liaison*⟩`,`⟨*tikz*⟩`]`

où chaque champ prend sa valeur par défaut s'il est vide :

- le ⟨*nom*⟩ est le nom du crochet : tous les caractères alphanumériques (`a..z`, `A..>`, `0..9`) sont admis[2]. Ce nom vaut « `a` » par défaut. Seul ce champ est pris en compte lors de la première occurrence du crochet portant ce ⟨*nom*⟩.
- ⟨*liaison*⟩ spécifie le type de liaison.

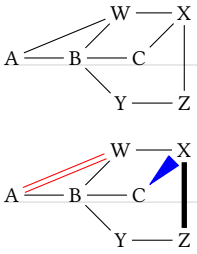
  Deux cas de figure sont possibles ; soit ce champ est un entier représentant le type de liaison voulu : 1=liaison simple, 2=liaison double, etc. Voir le tableau page 6 pour les numéros des types de liaisons.

  Soit la ⟨*liaison*⟩ est constituée des caractères codant la liaison, à condition que ces caractères soient *entre accolades* ;
- ⟨*tikz*⟩ sera passé directement à ti*k*z comme on l'a vu avec les liaisons classiques.

Voici notre molécule avec les liaisons distantes requises, puis avec les liaisons `A-W` et `X-C` personnalisées :

Plusieurs liaisons éloignées

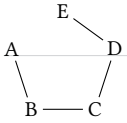
```
\chemfig{A?[a]-B(-[1]W?[a]-X?[b])(-[7]Y-Z?[b])-C?[b]}
\bigbreak
\chemfig{A?[a]-B(-[1]W?[a,2,red]-X?[b])(-[7]Y-
    Z?[b,1,{line width=2pt}])-C?[b,{>},blue]}
```



On peut écrire plusieurs crochets différents après un atome. Supposons que dans ce pentagone incomplet, on veuille relier `A-E`, `A-C` et `E-C` :

Un cycle incomplet

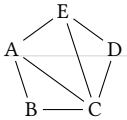
```
\chemfig{A-[:-72]B-C-[:72]D-[:144]E}
```



Alors, il faut procéder de cette façon :

Plusieurs liaisons éloignées

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



[2] Ce n'est pas totalement exact. En réalité, tous les tokens pouvant être mis entre `\csname...\endcsname` sont autorisés.

3 Cycles

L'exemple précédent montre comment tracer un polygone régulier, mais la méthode est fastidieuse puisque les angles dépendent du nombre de côté du polygone.

3.1 Syntaxe

chem*fig* peut tracer des polygones réguliers facilement. L'idée est d'attacher un cycle à un ⟨*atome*⟩ extérieur à ce cycle avec cette syntaxe :

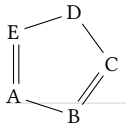
⟨*atome*⟩`*`⟨*n*⟩`(`⟨*code*⟩`)`

⟨*n*⟩ est le nombre de côtés du polygone et le ⟨*code*⟩ représente les liaisons et les groupes d'atomes qui constituent ses sommets et arêtes. Ce code *doit* commencer par une liaison puisque l'`atome` se trouve à l'extérieur du cycle.

Voici un 5-cycle, attaché a l'atome « `A` » :

5-cycle

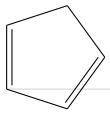
```
\chemfig{A*5(-B=C-D-E=)}
```



On peut également dessiner un cycle avec un, plusieurs, ou tous les groupes d'atomes vides, comme c'est le cas pour les dessins hors des cycles :

5-cycle avec groupes vides

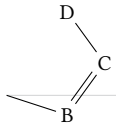
```
\chemfig{*5(-===)}
```



Un cycle peut être incomplet :

5-cycle incomplet

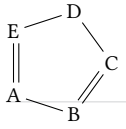
```
\chemfig{*5(-B=C-D)}
```



Si un cycle a un code qui contient trop de liaisons et de groupes d'atomes pour le nombre de sommets spécifiés, toutes les liaisons et les groupes au delà du maximum admissible sont ignorés :

5-cycle tronqué

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



Il est possible de dessiner un cercle ou un arc de cercle à l'intérieur d'un cycle. Pour cela, on utilise cette syntaxe :

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

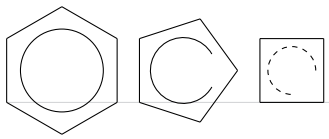
où chaque champ de l'argument optionnel prend sa valeur par défaut s'il est vide :

— ⟨*angle 1*⟩ et ⟨*angle 2*⟩ sont les angles absolus de départ et de fin de l'arc de cercle. Par défaut, ils valent 0° et 360° de telle sorte qu'un cercle complet est tracé par défaut ;

— ⟨*tikz*⟩ est le code qui sera passé à ti*k*z pour le dessin de l'arc de cercle.

Cycles et arcs de cercle

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

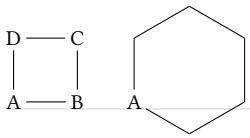


3.2 Position angulaire

Lorsqu'un cycle est au début d'un molécule et comme on le voit dans les exemples ci-dessous, la règle est que l'atome d'attache « A » se trouve toujours au sud-ouest du cycle. De plus, le cycle est toujours construit dans le sens trigonométrique et la dernière liaison tombe verticalement sur l'atome de rattachement :

Position angulaire des cycles

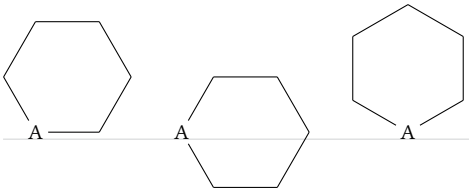
```
\chemfig{A*4(-B-C-D-)}\qquad\chemfig{A*6(------)}
```



Si cette position angulaire ne convient pas, il est possible de spécifier via l'argument optionnel du début de la molécule un autre angle par défaut. Voici un 6-cycle qui a été pivoté de +30°, de −30° puis de +60° :

Rotation des cycles

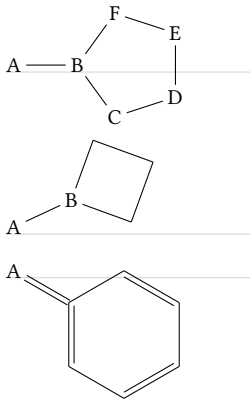
```
\chemfig{[:30]A*6(------)}\qquad
\chemfig{[:-30]A*6(------)}\qquad
\chemfig{[:60]A*6(------)}
```



Lorsqu'un cycle ne commence pas une molécule, la position angulaire par défaut change : le cycle est tracé de telle sorte que la liaison qui arrive sur l'atome de rattachement soit la bissectrice du premier et du dernier côté du cycle. :

Liaison arrivant sur un cycle

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



3.3 Ramification partant d'un cycle

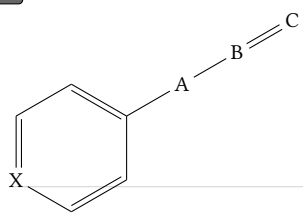
Pour faire partir une ramification d'un des sommets du cycle, on emploie la syntaxe déjà vue :

⟨*atome*⟩(⟨*code*⟩)

où le ⟨*code*⟩ est celui de la sous molécule et l'⟨*atome*⟩ occupe le sommet. Une chose particulière aux cycles est que l'angle par défaut de la sous molécule n'est pas 0° mais est calculé de telle sorte qu'il soit la bissectrice des côtés partant du sommet :

Ramification partant d'un cycle

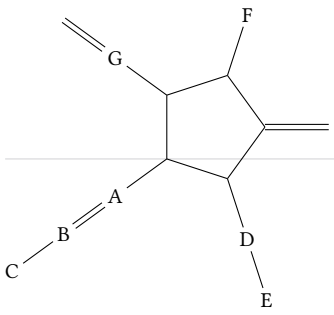
```
\chemfig{X*6(-=-(-A-B=C)=-=-)}
```



On peut faire partir une sous molécule du premier sommet du cycle, ainsi que de tous les autres sommets :

Cycle et ramifications

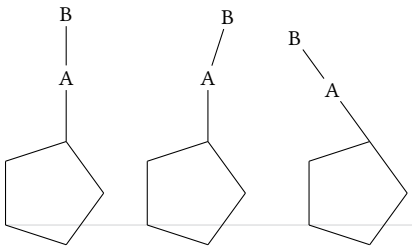
```
\chemfig{*5((-A=B-C)-(-D-E)-(=)-(-F)-(-G=)-)}
```



Si on souhaite que la liaison partant d'un sommet ne soit pas la bissectrice de ses côtés, on peut jouer sur le paramètre optionnel global ou le paramètre optionnel de la liaison :

Ramifications et angles spécifiés

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

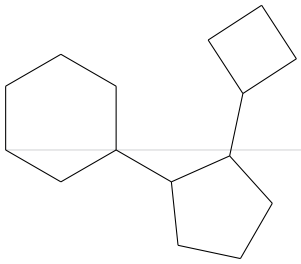


Il est intéressant de constater au troisième exemple que si on spécifie un angle relatif de 0°, la liaison se fait dans le prolongement de la liaison qui précédait dans le cycle.

On peut désormais lier des cycles entre eux par des liaisons :

Cycles liés

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

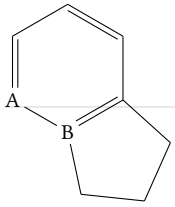


3.4 Cycles accolés

Pour « coller » 2 cycles entre eux, la syntaxe est légèrement différente : on identifie le sommet d'où va commencer l'autre cycle. Il suffit de faire suivre ce sommet par la syntaxe habituelle d'un cycle. Voici par exemple un 5-cycle qui part du deuxième sommet d'un 6-cycle :

Cycles imbriqués

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

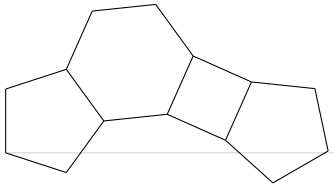


On remarque que le cycle qui vient se greffer sur le cycle principal a une position angulaire telle que deux de leurs côté coïncident. De plus, le 5-cycle n'a que 4 liaisons. En effet, la 5^{e} serait inutile puisqu'il s'agit du deuxième côté du 6-cycle qui est déjà tracé.

Il est bien entendu possible de coller plusieurs cycles entre eux :

Plusieurs cycles imbriqués

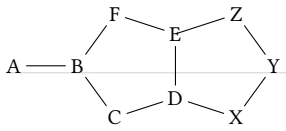
```
\chemfig{*5(--*6(-*4(-*5(----)--)----)---)}
```



Certains cas méritent davantage d'attention. On voit sur cet exemple que le quatrième côté du deuxième 5-cycle vient boucler sur le centre de l'atome « `E` ».

Dessin non parfait

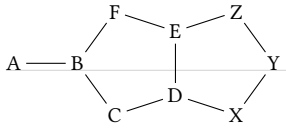
```
\chemfig{A-B*5(-C-D*5(-X-Y-Z-)-E-F-)}
```



Ceci est normal puisque le deuxième 5-cycle (qui part de l'atome « `D` ») est dessiné *avant* que chem*fig* n'ait pris connaissance de l'atome « `E` ». Dans ce cas, il faut se servir de deux crochets pour tracer la liaison `>-E` :

Liaison distante et cycle

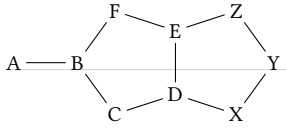
```
\chemfig{A-B*5(-C-D*5(-X-Y-Z?)-E?-F-)}
```



On aurait aussi pu utiliser un `\phantom{E}` au dernier sommet du 5-cycle :

Utilisation de \phantom

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

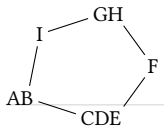


3.5 Groupes d'atomes dans un cycle

Il faut prendre des précautions avec les cycles lorsque un ou plusieurs sommets sont constitués de plusieurs atomes :

Cycle et groupes d'atomes

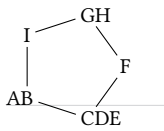
```
\chemfig{AB*5(-CDE-F-GH-I-)}
```



Pour que le cycle ait une forme régulière, il faut passer outre le mécanisme de chem*fig* qui calcule automatiquement les atomes de départ et d'arrivée des liaisons. Ici, il faut relier `C-F` et `F-G` en le spécifiant avec l'argument optionnel de ces liaisons :

Atomes de départ et d'arrivée imposés

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



4 Représentation des déplacements d'électrons

On peut représenter les déplacements d'électrons des effets mésomères ou des mécanismes réactionnels. Cela est rendu possible en marquant le point de départ et le point d'arrivée de la flèche que l'on utilise pour indiquer la migration des électrons à l'aide de la syntaxe « `@{`⟨*argument*⟩`}` ».

Cette syntaxe permet de poser un nœud (au sens de ti*k*z) en rendant ce nœud accessible en dehors de l'argument de la commande `\chemfig` grâce à l'option « `remember picture` » qui est passée à tous les environnements « `tikzpicture` ». Cela suppose que le visualiseur prenne en charge le « picture remembering » et que la compilation soit lancée deux fois.

Pour définir un nœud, deux endroits possibles :

— un nœud de dimension nulle sur une liaison en utilisant la syntaxe « `@{`⟨*nom*⟩`,`⟨*coeff*⟩`}` » placée immédiatement au début de l'argument optionnel de la liaison concernée sans être suivi d'une virgule s'il y a un premier argument optionnel. Dans ce cas, le nœud portera le nom « ⟨*nom*⟩ » et le ⟨*coeff*⟩, compris entre 0 et 1, déterminera où le nœud se situera sur la liaison. Si le ⟨*coeff*⟩ est omis, il prend la valeur de 0.5 par défaut ce qui signifie que le nœud est posé au milieu de la liaison ;

— un nœud sur un atome en utilisant la syntaxe « `@{`⟨*nom*⟩`}` » immédiatement avant l'atome concerné. Dans ce cas, le nœud a exactement le même encombrement que l'atome, il peut notamment être vide et donc avoir des dimensions nulles.

Une fois que la commande `\chemfig` a dessiné la (ou les) molécule(s) et a posé les nœuds avec la syntaxe décrite ci-dessus, on peut relier ces nœuds entre eux avec les instructions de ti*k*z. Ces instructions seront placées dans l'argument de la commande `\chemmove`[1] et auront la syntaxe suivante si par exemple, on doit relier un nœud portant le nom « ⟨*nom1*⟩ » au nœud portant le nom « ⟨*nom2*⟩ » :

```
\chemmove[⟨opt⟩]{\draw[⟨opt tikz⟩](⟨nom1⟩)⟨link tikz⟩(⟨nom2⟩);}
```

L'argument optionnel ⟨*opt*⟩ de la commande `\chemmove` sera ajouté à l'argument de l'environnement `tikzpicture` dans lequel seront tracées les liaisons entre les nœuds. Les instructions ⟨*opt tikz*⟩ et ⟨*link tikz*⟩ sont exhaustivement décrites dans la documentation de l'extension ti*k*z.

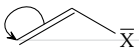
4.1 Effet mésomère

Pour fixer les idées, prenons l'exemple d'un effet mésomère impliquant une double liaison et un doublet non liant conjugués. Commençons par la possible délocalisation des électrons de la double liaison. On va poser un nœud nommé « `db` » (double bond) au milieu de la double liaison et un nœud nommé « `a1` » sur l'extrémité de la double liaison.

La macro `\charge` sera décrite plus loin dans le manuel, cf. p 29.

Effet mésomère 1

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



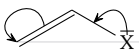
Pour relier les nœuds « db » et « a1 », nous avons utilisé la syntaxe suivante :

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

Le style des pointes de flèche par défaut dans `\chemmove` est « `CF` ». Dans cet exemple, nous utilisons 2 points de contrôle[2]. Ceux-ci seront situés en coordonnées polaires à 100° et 5 mm de « db » pour le premier et à 145° et 5 mm de « a1 » pour le second. Cette syntaxe peut paraître compliquée en première lecture, mais son utilisation se réduit dans la plupart des cas à un simple copier-coller dans lequel on modifie juste le nom des nœuds et les coordonnées polaires des points de contrôle. Ce que nous allons vérifier tout de suite avec l'ajout d'une flèche partant du doublet non liant (nœud « `dnl` ») vers la liaison simple (nœud « `sb` »).

Effet mésomère 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);}
```



Pour notre nouvelle flèche nous avons fixé les points de contrôle comme suit : un angle de 90° à 4 mm de « dnl » et un angle de 45° à 4 mm de « sb ». Mais nous ne sommes pas entièrement satisfaits car nous aimerions que la flèche ne touche pas le trait représentant le doublet non liant. Pour cela nous allons ajouter quelques options à notre flèche.

[1] En réalité, la commande `\chemmove` place son argument dans un environnement « `tikzpicture` » avec les options « `remember picture,overlay` ».
[2] Pour connaître toutes les façons de relier deux nœuds avec ti*k*z, lire la documentation de ce package.

Effet mésomère 3

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



L'option « `shorten <=3pt` » indique que le point de départ de la flèche doit être raccourci de 3 pt et de même, « `shorten >=2pt` » indique que le point d'arrivé doit être raccourci de 2 pt.

On peut utiliser toute la puissance des instructions de ti*k*z pour modifier le dessin de la flèche. Ici, nous changeons l'extrémité de la flèche partant de la double liaison en « `-stealth` », nous la dessinons en pointillés d'épaisseur fine et rouge. Nous rajoutons également la lettre π au dessus de la flèche en son milieu :

Effet mésomère 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);}
```



Dans l'exemple suivant nous allons voir comment indiquer la position de l'ancre de départ ou d'arrivée. Si nous écrivons

Ancre de départ et d'arrivée 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);}
```



Nous constatons que l'ancre de départ de notre flèche ne pointe pas correctement sur nos électrons. La flèche part du milieu du bord supérieur du nœud. En effet, nous avons choisi un angle de départ de 90° et ti*k*z fait donc partir la flèche de l'ancre « x1.90 » qui correspond à l'intersection de la demi droite partant du centre du nœud « x1 » et faisant un angle avec l'horizontale de 90° avec le bord du nœud qui est un rectangle. Pour obtenir le départ de la flèche d'où nous voulons, nous devons spécifier sa position. Après quelques tâtonnements, c'est « x1.57 » :

Ancres de départ et d'arrivée 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);}
```



Dans certains cas il sera plus facile d'utiliser les coordonnées cartésiennes pour les points de contrôle. Ici, nous n'utilisons qu'un seul point de contrôle situé à 1 cm horizontalement de « `x1` » et 1,5 cm verticalement :

Un seul point de contrôle

```
\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);}
```



Dans ce cas nous plaçons un point de contrôle à 1 cm horizontalement et 2 cm verticalement de « x1 ». Tous les objets graphiques dessinées par l'intermédiaire de la commande `\chemmove` sont faits en surimpression et ne seront pas comptés dans les boîtes englobantes.

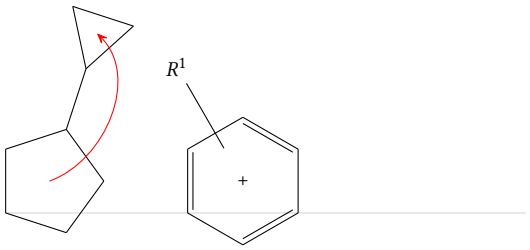
4.2 Centre d'un cycle

Le centre d'un cycle devient adressable via `\chemmove` car chaque cycle porte en son centre un nœud de dimension nulle dont le nom est `cyclecenter`⟨*n*⟩ où ⟨*n*⟩ est le numéro du cycle, entendu que les cycles sont numérotés à partir de 1 dans l'ordre où ils sont tracés. Il est possible d'afficher le numéro de chaque cycle en mettant à ⟨*true*⟩ la clé booléenne `show cntcycle`.

La clé booléenne `autoreset cntcycle`, ⟨*true*⟩ par défaut, remet à 0 le compteur de cycle au début de chaque molécule tracée (c'est-à-dire à chaque appel de `\chemfig`).

Centre de cycle

```
\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);
}
```



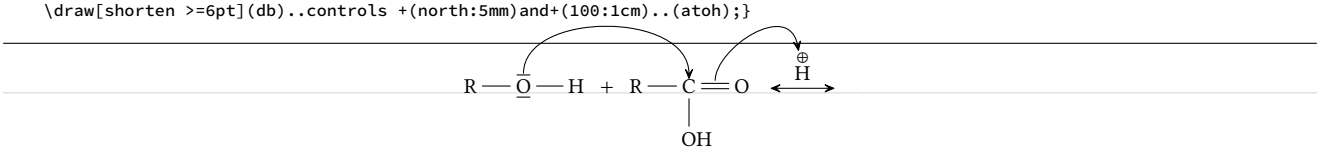
4.3 Mécanismes réactionnels

Grâce à l'option `remember picture` qui est passée à tous les environnements « `tikzpicture` » nous pouvons facilement dessiner les flèches indiquant les mécanismes réactionnels. Prenons comme exemple la première étape de la réaction d'estérification.

Les macros `\schemestart`, `\schemestop`, `\arrow` et `\+` seront décrites plus loin dans le manuel.

Estérification: étape 1

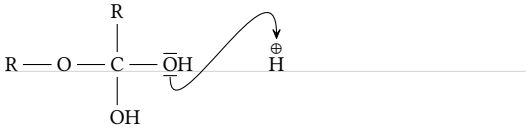
```
\setchemfig{atom sep=7mm}
\schemestart
    \chemfig{R-@{dnl}\charge{90=\|,-90=\|}{O}-H}
    \+
    \chemfig{R-@{atoc}C([6]-OH)=[@{db}]O}
    \arrow(.mid east--){<->[\chemfig{@{atoh}\chemabove{H}{\scriptstyle\oplus}}]}
\schemestop
\chemmove[shorten <=2pt]{
    \draw(dnl)..controls +(90:1cm)and+(north:1cm)..(atoc);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



L'utilisation de la commande `\chemabove{`⟨*code*⟩`}{`⟨*matériel*⟩`}` ne change pas les dimensions de la boîte englobante du ⟨*code*⟩. Pour cette raison on peut rencontrer certaines difficultés pour pointer sur le matériel indiquant la charge portée (⊕ ou ⊖). Dans l'exemple ci-dessus la solution est de créer un point de contrôle avec un angle de 110° à 1 cm de « atoh » et de raccourcir la flèche de 6pt. Dans l'exemple suivant, seconde étape de la réaction d'estérification, on peut voir que la flèche peut prendre des formes plus compliquées sans forcément surcharger le code.

Estérification: étape 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);}
```



Nous laissons le soin au lecteur d'écrire la suite.

4.4 Écrire un nom sous une molécule

Pour plus de commodité, chem*fig* permet d'écrire le nom d'une molécule sous celle-ci avec la commande

`\chemname[`⟨*dimension*⟩`]{\chemfig{`⟨*code de la molécule*⟩`}}{`⟨*nom*⟩`}`

La ⟨*dimension*⟩, qui vaut `1.5ex` par défaut, sera insérée verticalement entre la ligne de base de la molécule et le haut des lettres du ⟨*nom*⟩. Le ⟨*nom*⟩ sera centré par rapport à la molécule mais ce ⟨*nom*⟩ ne peut pas contenir plusieurs paragraphes. Comme on le voit sur cet exemple : H —— O —— H, le ⟨*nom*⟩ qui est affiché sous la molécule est pris en compte dans la boîte englobante

La molécule d'eau : H_2O

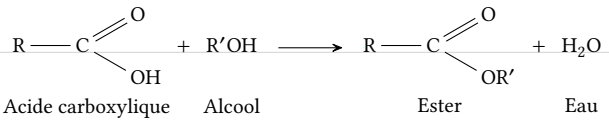
mais uniquement en ce qui concerne sa dimension *verticale*. L'encombrement horizontal du ⟨*nom*⟩ est toujours nul.

Voici une réaction avec les noms sous les molécules :

Affichage du nom des molécules

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

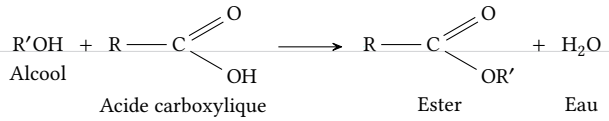
```
    \chemname{\chemfig{H_2O}}{Eau}
\schemestop
\chemnameinit{}
```



Il y a quelques contraintes avec cette commande. Pour en pendre conscience, supposons que l'on ait inversé l'acide et l'alcool dans le membre de gauche :

Alignement des noms 1

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



En fait, au dessous de la ligne de base de chaque molécule (en gris clair sur les exemples de ce manuel), pour écrire le ⟨*nom*⟩, la commande `\chemname` insère 1.5ex + *la plus grande des profondeurs*[3] *des molécules rencontrées.*

La macro `\chemnameinit{`⟨*argument*⟩`}` initialise cette plus grande profondeur avec le contenu de son ⟨*argument*⟩. Il convient donc :

— d'écrire `\chemnameinit{`⟨*plus profonde molécule*⟩`}` avant d'impliquer cette commande dans une réaction, sauf si cette réaction commence par la plus profonde molécule ;

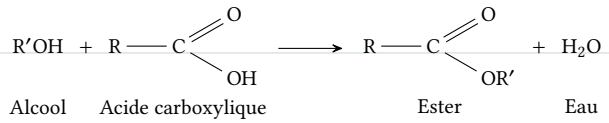
— d'écrire `\chemnameinit{}` après avoir écrit tous les noms d'une réaction chimique afin d'éviter que la plus grande profondeur trouvée dans cette réaction n'interfère dans une réaction future.

Il faut noter que les assignations concernant la plus grande profondeur sont globales si `gchemname` = ⟨*true*⟩, ce qui est le comportement par défaut. Elles sont locales sinon.

Le code correct est donc d'utiliser `\chemnameinit` avant et après la réaction :

Alignement des noms 2

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



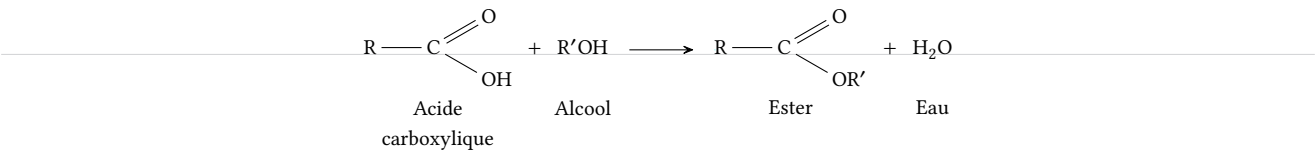
Enfin, pour écrire un nom sur plusieurs lignes, la commande `\\` rencontrée dans un ⟨*nom*⟩ effectue un retour à la ligne[4] :

Nom sur 2 lignes

```
\schemestart
    \chemname{\chemfig{R-C(-[:-30]OH)=[:30]O}}{Acide\\carboxylique}
    \+
    \chemname{\chemfig{R'OH}}{Alcool}
```

[3] En langage TEX, la profondeur est la dimension qui s'étend verticalement sous la ligne de base.
[4] En revanche, la commande `par` est interdite et provoquera une erreur à la compilation.

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



Si l'on écrit `\chemname*{`⟨*nom*⟩`}`, alors la macro ne tient pas compte des noms rencontrés précédemment.

5 Utilisation avancée

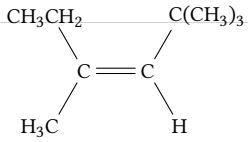
5.1 Découpage des atomes

Le mécanisme automatique de découpage en atomes, déjà décrit auparavant, fait s'étendre chaque atome jusqu'à la prochaine lettre majuscule ou l'un des caractères `-` `=` `~` `(` `!` `*` `<` `>` `@`.

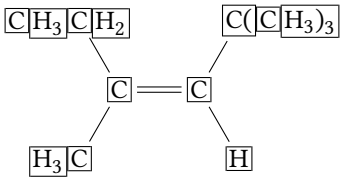
Dans certains cas, ce découpage automatique produit des atomes incorrects ce qui peut se traduire par un affichage imparfait. Pour cette molécule, on remarque que le caractère « `(` » est mis entre accolade de façon à éviter que chem*fig* ne comprenne à tort qu'une ramification est créée :

Alcène

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



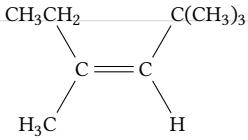
La liaison qui arrive sur l'atome de carbone en haut à droite est trop courte. Cela se produit car, si l'on applique les règles de découpage de chem*fig* dans le groupe d'atomes en haut à droite, les atomes sont découpés de cette façon : « `C{(}` », « `C` », « `H_3{)}_3` ». On comprend alors qu'en mode math, le premier atome qui contient une parenthèse ait une profondeur trop importante comme on le voit en rendant visibles les boîtes contenant les atomes :



Le caractère « `|` » provoque la coupure de l'atome en cours à l'endroit où il est rencontré. Ainsi, on peut écrire `C|{(CH_3)_3}` pour faire en sorte que chem*fig* ne découpe que 2 atomes ici : « `C` » et « `{(CH_3)_3}`. Le problème de la liaison trop courte est ainsi résolu :

Alcène

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



5.2 Mise en forme des atomes

Une fois le découpage en atomes effectué, la macro `\printatom` est appelée de façon interne par chem*fig* pour afficher chaque atome. Son unique argument est constitué des tokens lus dans le code de la molécule représentant l'atome courant (par exemple « `H_3` »). Cette macro se place en mode mathématique et affiche son argument avec la police mathématique « `rm` ». Elle est définie par le code suivant :

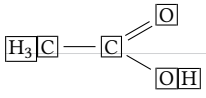
— `\def\printatom#1{\ifmmode\rm#1\else$\rm#1$\fi}` en compilant avec (pdf)(xe)(lua)TEX ;

— `\newcommand*\printatom[1]{\ensuremath{\mathrm{#1}}}` en compilant avec (pdf)(xe)(lua)LATEX.

On peut modifier le code de cette macro pour personnaliser l'affichage de atomes. Dans l'exemple ci-dessous, on redéfinit `\printatom` de façon à ce que chaque atome soit contenu dans une boîte rectangulaire, à l'aide de la macro `\frame` de OpTEX :

Redéfinition de \printatom

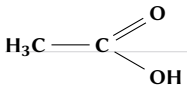
```
\vvkern=1pt \hhkern=1pt % espaces h et v entre lignes et contenu
\def\printatom#1{\frame{\ifmmode\rm#1\else$\rm#1$\fi}}
\chemfig{H_3C-C(=[:30]O)(-[:-30]OH)}
```



Voici comment la redéfinir pour utiliser de la police sans sérif grasse du mode math :

Atomes affichés en police sans sérif grasse

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



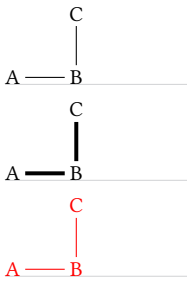
5.3 Paramètres passés à ti*k*z

La clé `chemfig style` contient des instructions ti*k*z qui seront passées à l'environnement `tikzpicture` dans lequel est dessinée la molécule. Par ailleurs, la clé `atom style` contient des instructions ti*k*z qui seront exécutées lors du dessin de chaque nœud ; ces instructions sont ajoutées à la fin de `every node/.style{`⟨*argument*⟩`}`, c'est-à-dire après les instructions suivantes : « `anchor=base,inner sep=0pt,outer sep=0pt,minimum size=0pt` ».

Par l'intermédiaire de `chemfig style`, on peut choisir par exemple la couleur générale ou l'épaisseur des lignes :

Choix du style

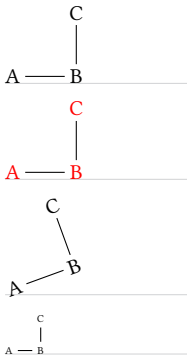
```
\chemfig{A-B-[2]C}\par\medskip
\chemfig[chemfig style={line width=1.5pt}]{A-B-[2]C}\par\medskip
\chemfig[chemfig style=red]{A-B-[2]C}
```



Avec la clé `node style`, on peut choisir la couleur des nœud dessinés par ti*k*z, modifier l'inclinaison du dessin ou changer l'échelle :

Choix des 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 Alignement vertical

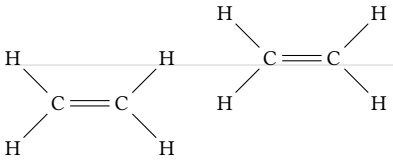
5.4.1 La clé `baseline`

La clé `baseline` permet de contrôler finement le placement vertical de la molécule par rapport à la ligne de base du paragraphe en cours.

Elle vaut ⟨*0pt*⟩ par défaut et dans ce cas, le premier atome rencontré (qu'il soit vide ou pas) est celui qui est placé sur la ligne de base du paragraphe en cours, représentée en gris sur les exemples de ce manuel. Le choix de ce premier atome conditionne donc le placement de tous les autres relativement à lui et influe souvent sur le placement de la molécule toute entière.

Premier atome

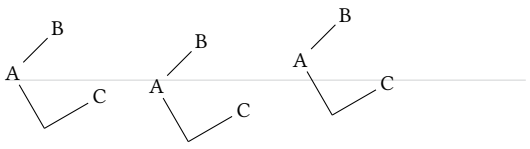
```
\chemfig{H-[7]C(-[5]H)=C(-[1]H)-[7]H}\qquad
\chemfig{C(-[3]H)(-[5]H)=C(-[1]H)-[7]H}
```



On peut spécifier une dimension arbitraire pour décaler verticalement la molécule de cette valeur avec la syntaxe `baseline =` ⟨*dimension*⟩ :

Décalage vertical

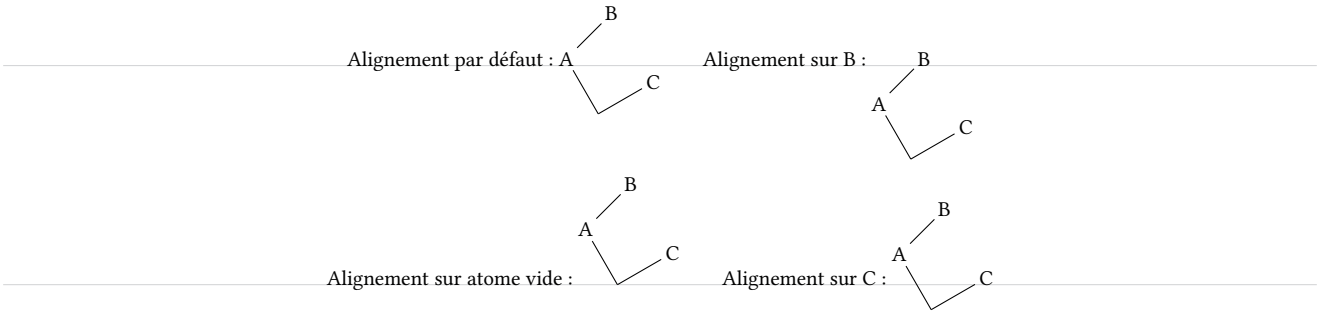
```
\chemfig{A(-[:-60]-[:30]C)-[:45]B}\qquad
\chemfig[baseline=5pt]{A(-[:-60]-[:30]C)-[:45]B}\qquad
\chemfig[baseline=-5pt]{A(-[:-60]-[:30]C)-[:45]B}
```



Avec la syntaxe `baseline = ⟨(<nom>)⟩` (le nom doit se trouver entre parenthèses), on spécifie que la ligne de base de la molécule est sur le nœud nommé c⟨*nom*⟩. Le nom de l'atome peut être celui attribué automatiquement par chem*fig* (de la forme n⟨*a*⟩−⟨*b*⟩) ou bien un nom donné par l'utilisateur par la syntaxe `@{`⟨*nom*⟩`}` (voir page 20).

Alignement sur des atomes

```
Alignement par défaut :     \chemfig{A(-[:-60]-[:30]C)-[:45]B}\qquad
Alignement sur B :          \chemfig[baseline=(b.base)]{A(-[:-60]-[:30]C)-[:45]@{b}B}\medbreak
Alignement sur atome vide : \chemfig[baseline=(vide)]{A(-[:-60]@{vide}-[:30]C)-[:45]B}\qquad
Alignement sur C :          \chemfig[baseline=(c.base)]{A(-[:-60]-[:30]@{c}C)-[:45]B}
```

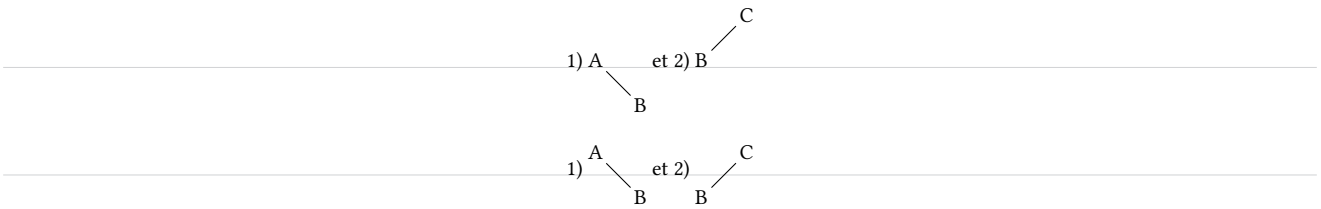


Il est possible de nommer de nœuds de ti*kz*. Ainsi, si l'on veut centrer verticalement plusieurs molécules sur la ligne de base courante, met la valeur `⟨(current bounding box.center)⟩` dans la clé `baseline`.

Alignement centré

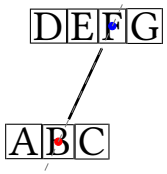
```
1) \chemfig{A-[:-45]B} et 2) \chemfig{B-[:45]C}\bigbreak

\setchemfig{baseline=(current bounding box.center)}% centrage vertical de toutes les molécules
1) \chemfig{A-[:-45]B} et 2) \chemfig{B-[:45]C}
\setchemfig{baseline=0pt}% retour à la valeur par défaut
```



5.4.2 Importance de la hauteur ou profondeur d'un atome

Une liaison partant d'un atome passerait, si on la prolongeait, par le centre de sa boite englobante. L'atome d'arrivée est placé au bout de la liaison de telle sorte que le centre de sa boite englobante soit dans le prolongement de la liaison. Par conséquent, une liaison entre deux atomes passe, par prolongement, par les centres de leurs boites englobantes, comme l'illustre cet exemple :



Ce mécanisme peut créer un décalage de hauteur, particulièrement visibles lorsque les liaisons sont horizontales. Tout se passe bien lorsque les atomes ont les mêmes dimensions verticales ; en revanche, il suffit qu'un atome de départ soit haut (avec exposant) ou profond (avec indice) et que l'atome d'arrivée ait une dimension verticale différente pour qu'un décalage de hauteur se produise. On peut observer dans cet exemple que les liaisons et les atomes qui suivent `A^1` sont légèrement décalés vers le haut et que ceux qui suivent `E_1` le sont vers le bas.

Placement horizontal

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



On peut mettre en évidence ce phénomène en rendant visible les boîtes englobantes des atomes :

Placement horizontal et boîtes englobantes

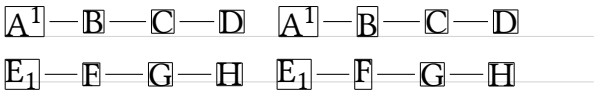
```
\typosize[13/]
\setchemfig{atom sep=2em}
\vvkern=0pt \hhkern=0pt \rulewidth=0.1pt
\def\printatom#1{\frame{\ifmmode\rm#1\else$\rm#1$\fi}}
\chemfig{A^1-B-C-D}\qquad
\chemfig{E_1-F-G-H}
```



Le fonctionnement antérieur à la version 1.7 de chem*fig* masquait cet effet sur le premier atome qui suit la boite englobante problématique en ajoutant à l'argument de chaque `\printatom` le `\vphantom` de l'atome précédent (sans prendre en compte le strut rajouté). On peut revenir à ce comportement en mettant la clé `use atom strut` à ⟨*true*⟩, mais cette correction n'est effective que sur un seul atome.

Placement horizontal et boîtes englobantes

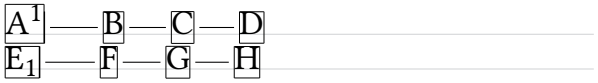
```
\typosize[13/]
\setchemfig{atom sep=2em}
\vvkern=0pt \hhkern=0pt \rulewidth=0.1pt
\def\printatom#1{\frame{\ifmmode\rm#1\else$\rm#1$\fi}}
\chemfig{A^1-B-C-D}\quad
\chemfig[use atom strut=true]{A^1-B-C-D}
\medbreak
\chemfig{E_1-F-G-H}\quad
\chemfig[use atom strut=true]{E_1-F-G-H}
```



Aucune solution automatique n'étant satisfaisante, on peut contourner « à la hache » ce problème en redéfinissant `\printatom` pour ajouter à chaque atome composé la poutre définie par `\strut` :

Contournement du placement vertical

```
\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}
```

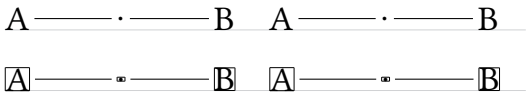


5.4.3 La macro `\chemskipalign`

Il est possible pour n'importe quel groupe d'atomes de désactiver momentanément le mécanisme d'ajustement d'alignement et neutraliser le `\vphantom`. Il suffit pour cela de placer dans le groupe d'atomes la commande `\chemskipalign` : l'alignement reprendra au groupe d'atomes suivant comme si le groupe d'atome contenant `\chemskipalign` n'avait pas existé. On peut se rendre compte sur l'exemple suivant de l'effet de cette instruction qui a pour effet de placer le point de référence de la boîte contenant le premier atome au niveau de la liaison qui arrive de gauche. Les boîtes englobant les atomes ont été dessinées à la seconde ligne :

Désactivation du mécanisme d'alignement

```
\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}
```



Cette commande est à utiliser avec précaution car l'alignement des atomes dans le groupe d'atomes à venir peut être perturbé. En règle générale, tout se passera bien si le groupe d'atomes dans lequel figure `\chemskipalign` contient *un seul atome* dont la hauteur et la profondeur sont *inférieures* à celles de l'atome qui précède et qui suit, et si les atomes qui précèdent et suivent ont leur profondeur et hauteur égales. Voici par exemple la mésaventure qui arrive lorsque le groupe d'atomes contient 2 atomes, ici « `\chemskipalign.` » et « `B` » :

Conséquence de la commande \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}
```



Cette fonctionnalité peut parfois s'avérer utile. Supposons que l'on veuille dessiner la molécule



On peut définir les commandes qui vont dessiner les disques vides et pleins avec ti*kz*. Afin que ces disques soit à la bonne hauteur, c'est-à-dire à la hauteur de la liaison qui leur arrive dessus, on se servira de la commande `\chemskipalign`. Pour que, à la deuxième ligne de l'exemple ci-dessous, les liaisons « collent » aux disques, nous utiliserons la possibilité de modifier le retrait d'une liaison avec le caractère « `#` :

Utilisation de \chemskipalign et #

```
\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}
```



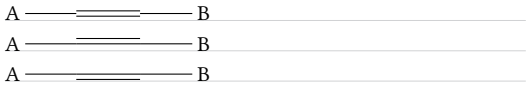
5.5 Liaisons doubles déportées

Toutes les liaisons doubles sont composées de 2 traits tracés de part et d'autre de la ligne théorique que prendrait la liaison simple. Il est possible de déporter cette liaison double de telle sorte qu'un des deux traits soit sur cette ligne théorique. L'autre trait étant alors au-dessus ou au-dessous de la liaison. En fait, il est plus rigoureux de dire « à gauche » ou « à droite » de la ligne théorique lorsqu'on parcourt la liaison dans le sens du tracé.

Pour déporter la liaison à gauche, il suffit d'écrire « `=^` » et pour la déporter à droite « `=_` » :

Liaisons doubles déportées

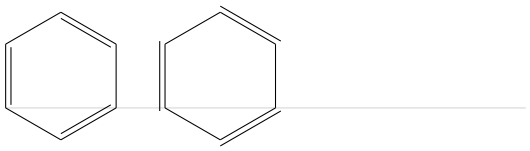
```
\chemfig{A-=-B}\par
\chemfig{A-=^-B}\par
\chemfig{A-=_-B}
```



Dans les cycles, les liaisons doubles sont automatiquement déportées vers la gauche. On peut cependant les déporter vers la droite en le spécifiant avec « `=_` » :

Liaisons doubles déportées et cycles

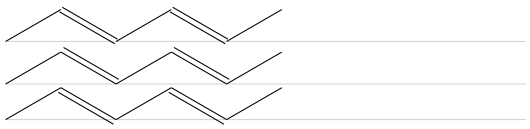
```
\chemfig{*6(-=-=-=)}\qquad
\chemfig{*6(-=_-=_-=_)}
```



Les liaisons déportées sont particulièrement utiles dans le tracé de formules topologiques de molécules comprenant chaînes carbonées avec des liaisons doubles. Elles permettent d'avoir une ligne brisée continue, alors que cette ligne brisée serait discontinue avec les liaisons doubles normales :

Liaisons déportées et formules topologiques

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



5.6 Liaisons doubles délocalisées

Il est parfois nécessaire de tracer une liaison double dont un trait serait plein et l'autre en pointillé. Cette fonctionnalité n'est pas codée en dur dans chem*fig* puisque ti*k*z, avec sa librairie « `decorations.markings` » le rend possible.

Liaisons délocalisées

```
\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 Sauvegarde d'une partie de molécule avec `\definesubmol`

chem*fig* est capable de sauvegarder un ⟨*code*⟩ sous forme d'un alias pour le réutiliser sous forme compacte dans le code d'une molécule. Ceci est particulièrement utile lorsque le ⟨*code*⟩ apparaît plusieurs fois.

Pour cela, on dispose de la commande

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

qui sauvegarde le ⟨*code*⟩ de façon à l'appeler dans le code de la molécule par le raccourci « `!{`⟨*nom*⟩`}` ». Ce ⟨*nom*⟩ peut être :

— une suite de caractères : tous les caractères alphanumériques pouvant se trouver entre `\csname` et `\endcsname` sont acceptés ;

— une séquence de contrôle.

Dans tous les cas, si l'alias est déjà défini il est déconseillé de l'écraser avec une nouvelle définition faite avec `\definesubmol`. Un « warning » sera émis avertissant l'utilisateur que cet alias sera écrasé par le nouveau. Pour écraser la définition d'un alias faite au préalable, il faut utiliser :

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

Voici un code qui dessine la molécule de pentane. On a pris soin auparavant de définir un alias « `xy` » pour le 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}
```

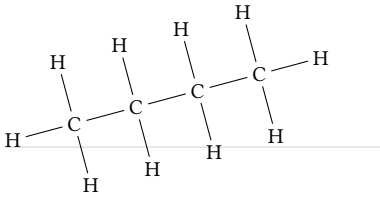


Ici, la manœuvre n'est pas très intéressante puisque « `!{xy}` » est aussi long à taper que le code qu'il remplace.

Mais dans certains cas, cette fonctionnalité fait gagner beaucoup de place dans le code de la molécule et en améliore la lisibilité. Dans l'exemple suivant, on dessine la molécule développée de butane. Pour cela, on va définir un alias avec la séquence de contrôle « `xx` » pour `CH_2`. Comme on n'emploie que des angles relatifs, il est possible de faire pivoter la molécule entière de l'angle que l'on veut via l'angle du paramètre optionnel global qui spécifie l'angle par défaut des liaisons de la molécule principale. On met ici 15° :

Butane

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



La commande `\definesubmol` admet un argument optionnel et sa syntaxe est la suivante :

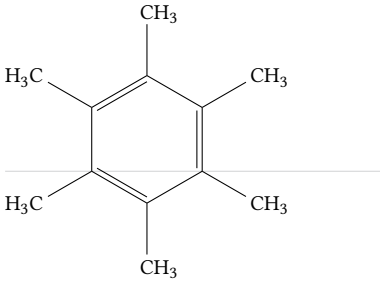
`\definesubmol{`⟨*nom*⟩`}[`⟨*code1*⟩`]{`⟨*code2*⟩`}`

Dans le cas où l'argument optionnel est présent, l'alias « `!`⟨*nom*⟩ » sera remplacé par ⟨*code1*⟩ si la liaison qui arrive sur l'alias vient de droite, c'est-à-dire si l'angle que fait la liaison entrante est compris entre −90° et 90°, ces valeurs étant non comprises. Pour tous les autres cas où la liaison arrive de gauche ou verticalement, l'alias sera remplacé par ⟨*code2*⟩.

On va définir une séquence de contrôle `\Me` pour « méthyl » de telle sorte que l'alias « `!\Me` » soit remplacé par « `H_3C` » lorsque la liaison arrive de droite et par « `CH_3` » lorsqu'elle arrive de gauche. Avec cet alias, on peut observer sur l'exemple que l'on n'a plus à s'occuper de l'angle (que dans certaines molécules plus complexes on ne connaît même pas) :

Alias dual

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



La sous-molécule sauvegardée via un ⟨*nom*⟩ n'admet pas d'argument lorsqu'elle est appelée avec « `!` ». Pour définir une sous-molécule admettant un ou plusieurs argument, il faut écrire ce ⟨*nombre*⟩ d'arguments juste après le ⟨*nom*⟩. La syntaxe complète de `\definesubmol` est donc :

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

Dans les ⟨*codes*⟩, les arguments doivent figurer sous leur forme habituelle « `#`⟨*n*⟩ » où ⟨*n*⟩ est le numéro de l'argument.

\definesubmol avec 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})}
```



Il est à noter que si le ⟨*nombre*⟩ d'arguments est incorrect (négatif ou supérieur à 9), un message d'erreur sera émis et chem*fig* considèrera que la sous molécule n'admet pas d'argument.

En dehors des cas où le caractère « # » est suivi d'un chiffre compris entre 1 et ⟨*nombre*⟩ auquel cas il représente un argument, les « # » sont autorisés dans les codes des sous molécules.

Utilisation de #

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


Dans cet exemple, seuls `#1` et `#2` sont compris comme les arguments de la sous molécule `\X`. Les autres « # » sont affichés tels quels dans la molécule (cas de `#3` et `#4`) ou compris comme le caractère spécifiant le réglage fin du retrait des liaisons.

5.8 La macro \charge

5.8.1 Présentation

La macro `\charge`, qui requiert deux argument obligatoires, permet de disposer des éléments — que l'on appellera charges — autour d'un atome ; sa syntaxe est la suivante

`\charge{[`⟨*paramètres généraux*⟩`]`⟨*position*⟩`[`⟨*code tikz*⟩`]=`⟨*charge*⟩`}{`⟨*atome*⟩`}`

où :

- l'⟨*atome*⟩ est à priori constitué d'une ou deux lettres, mais peut également être vide ;
- la ⟨*charge*⟩ est un contenu *arbitraire* qui sera placé autour de l'⟨*atome*⟩. Peu de contraintes existent sur cette ⟨*charge*⟩, il peut donc être du texte (en mode math si besoin), voire même du code ti*k*z ou une molécule dessinée avec `\chemfig` ;
- les ⟨*paramètres généraux*⟩ (optionnels) sont une liste de ⟨*clés*⟩ = ⟨*valeurs*⟩ spécifiant les options que doit satisfaire cette exécution de la macro `\charge`. Ces ⟨*clés*⟩ et ⟨*valeurs*⟩ sont décrites plus bas ;
- la ⟨*position*⟩ est de la forme ⟨*angle*⟩`:`⟨*décalage*⟩, mais il est possible de ne spécifier que l'⟨*angle*⟩, auquel cas, le ⟨*décalage*⟩ sera pris égal à `0pt` ;
- le ⟨*code tikz*⟩, optionnel, contient les options passées à la macro `\node` de ti*k*z, chargée de placer la ⟨*charge*⟩.

5.8.2 Paramètres

Les ⟨*clés*⟩ = ⟨*valeurs*⟩ disponibles dans les ⟨*paramètres généraux*⟩ sont :

clés	valeurs par défaut	Description
debug	`false`	Booléen qui lorsque égal à ⟨*true*⟩, dessine les contours du nœud recevant l'⟨*atome*⟩ (en vert), de celui où sont placées les ⟨*charges*⟩ (en bleu) et ceux recevant les ⟨*charges*⟩ (en rouge).
macro atom	`\printatom`	Macro qui prend comme argument l'⟨*atome*⟩.
circle	`false`	Booléen qui lorsque égal à ⟨*true*⟩, met l'⟨*atome*⟩ dans un nœud circulaire ; dans le cas contraire, le nœud est rectangulaire.
macro charge	⟨*vide*⟩	Macro (`\printatom` ou `\ensuremath` par exemple) qui prend comme argument chaque charge.

clés	valeurs par défaut	Description
extra sep	1.5pt	Augmentation de la taille du nœud (cercle ou rectangle) pour la position des charges : c'est la valeur passée à la clé inner sep de tikz.
overlay	true	Booléen qui lorsque ⟨*true*⟩, dessine les charges en « surimpression », c'est-à-dire hors de la boite englobante finale.
shortcuts	true	Booléen qui lorsque ⟨*true*⟩, active les raccourcis « \. », « \: », « | » et « \" » pour tracer des formules de Lewis.
lewisautorot	true	Booléen qui lorsque ⟨*true*⟩, effectue une rotation automatique de « \: » et « \" ».
.radius	0.15ex	Rayon du point utilisé pour tracer « \. » et « \: »
:sep	0.3em	Séparation entre les deux points de « \: »
.style	fill=black	Style tikz utilisé pour tracer les points « \. » et « \: »
"length	1.5ex	Longueur du rectangle \" et de la ligne |
"width	.3ex	Largeur du rectangle \"
"style	black,line width=0.4pt	Style tikz utilisé pour tracer le rectangle \"
|style	black,line width=0.4pt	Style tikz utilisé pour tracer la ligne \"

Il est possible de modifier certains de ces paramètres (ou tous) par l'exécution de la macro

```
\setcharge{⟨clés⟩=⟨valeurs⟩}
```

et réinitialiser tous les paramètres à leurs valeurs par défaut avec

```
\resetcharge
```

La macro \charge place les ⟨*charges*⟩ hors de la boite englobante (sauf spécification contraire dans les ⟨*paramètres*⟩) alors que \Charge les place *dans* la boite englobante.

L'⟨*angle*⟩ est l'endroit sur la frontière du nœud où sera placée la ⟨*charge*⟩. Cet ⟨*angle*⟩ peut-être exprimé en degrés ou bien être un ancre de frontière au sens de tikz, comme « south east ». Le ⟨*décalage*⟩ est une dimension au sens de TEX et représente une longueur additionnelle entre la frontière du nœud contenant l'⟨*atome*⟩ et l'endroit où est placé la ⟨*charge*⟩. Sauf indication contraire dans le ⟨*code tikz*⟩, le placement concerne le *centre* du nœud contenant la charge.

Dans les deux exemples qui suivent, debug sera mise à ⟨*true*⟩ afin de mieux percevoir les changements induits par la modification des paramètres. De plus, la macro \Charge sera utilisée afin que les boites englobantes tiennent compte des charges. On voit ici l'influence de la forme du nœud sur le placement des charges :

Exemple générique

```
Nœud par défaut :
    \Charge{30=\:,120=$\ominus$,210=$\delta^+$}{Fe}\qquad
    \setcharge{debug=true}
    \Charge{30=\:,120=$\ominus$,210=$\delta^+$}{Fe}
    \setcharge{debug=false}
\medbreak
Nœud circulaire :
    \Charge{[circle]30=\:,120=$\ominus$,210=$\delta^+$}{Fe}\qquad
    \setcharge{debug=true}
    \Charge{[circle]30=\:,120=$\ominus$,210=$\delta^+$}{Fe}
```

Nœud par défaut : δ^+Fe $\ominus$ δ^+Fe $\ominus$

Nœud circulaire : δ^+Fe $\ominus$ δ^+Fe $\ominus$

Pour éloigner les charges $\ominus$ et δ^+, on peut jouer sur le ⟨*décalage*⟩

Décalage des charges

```
Sans décalage :
  \Charge{30=\:,120=$\ominus$,210=$\delta^+$}{Fe}
\medbreak
Avec décalage :
  \Charge{[circle]30:3pt=\:,120:5pt=$\ominus$,210:5pt=$\delta^+$}{Fe}
```

Sans décalage : δ^+Fe $\ominus$

Avec décalage : δ^+Fe $\ominus$

On peut aussi jouer sur l'ancre d'attache : l'⟨*angle*⟩ où est placé la ⟨*charge*⟩ est stocké dans la macro \chargeangle ; il est donc judicieux de choisir l'ancre d'attache égal à 180+\chargeangle. Il est également possible de spécifier un nœud circulaire pour y placer la ⟨*charge*⟩.

Position fine

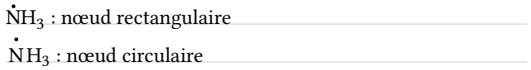
```
\Charge{30=\:,120:3pt=$\ominus$,210:5pt=$\delta^+$}{Fe}\qquad
\setcharge{debug}
\Charge{30=\:,120:3pt=$\ominus$,210:5pt=$\delta^+$}{Fe}
\setcharge{debug=false}
\medbreak
\Charge{[circle]30=\:,
        120[circle,anchor=180+\chargeangle]=$\ominus$,
        210[anchor=180+\chargeangle]=$\delta^+$}{Fe}%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



Il est important de noter que les nœuds circulaires ont des encombrements *parfois très différents* des nœuds « classiques » rectangulaires, notamment en ce qui concerne l'étendue horizontale et verticale. Il convient donc de rendre ⟨*true*⟩ la clé booléenne `circle` en connaissance de cause.

Nœuds circulaires

```
\chemfig{\charge{90=\.}{N}H_3} : nœud rectangulaire\smallbreak
\chemfig{\charge{[circle]90=\.}{N}H_3} : nœud circulaire
```



5.9 Empilement de caractères

Les macros

`\chemabove[`⟨*dim*⟩`]{`⟨*code*⟩`}{`⟨*matériel*⟩`}`

et

`\chembelow[`⟨*dim*⟩`]{`⟨*code*⟩`}{`⟨*matériel*⟩`}`

placent le ⟨*matériel*⟩ respectivement au-dessus et au-dessous du ⟨*code*⟩ à une distance verticale ⟨*dim*⟩, et cela sans changer la boîte englobante du ⟨*code*⟩. L'argument optionnel permet si on le souhaite, de spécifier cette dimension à chaque appel. Si l'argument optionnel n'est pas utilisé, une dimension par défaut sera prise : elle vaut ⟨*1.5pt*⟩ mais peut être modifiée avec clé `stack sep` = ⟨*dim*⟩.

Ces commandes sont indépendantes de la macro `\chemfig` et peuvent aussi bien être appelées à l'intérieur ou à l'extérieur de son argument.

On peut les utiliser notamment dans les cycles en prenant soin de mettre des accolades autour des lettres A, B, C et D pour éviter que chem*fig* ne stoppe la lecture de l'atome sur ces lettres :

Superposition dans les cycles

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

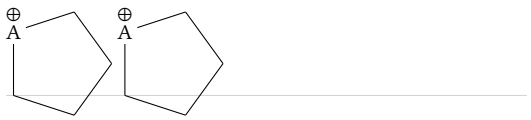


Les commandes `\Chemabove` et `\Chembelow` fonctionnent de la même façon sauf que la boîte englobante *tient compte* du ⟨*matériel*⟩ placé au dessus ou au dessous.

Quelle différence y a-t-il entre `\chemabove` et `\charge` lorsqu'il s'agit de placer un contenu au-dessus ou au-dessous d'un autre ?

\chemabove ou \charge

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



Par défaut, les deux macros donnent des résultats très proches. Des différences quant à leur utilisation existent cependant : la macro `\charge` requiert l'extension ti*k*z alors que `\chemabove` et `\chemabelow` sont codées avec des primitives de bas niveau de TEX et sont donc *rapides* et indépendantes de toute extension.

5.10 Sauvegarde dans un fichier

La clé booléenne `save` qui vaut ⟨*false*⟩ par défaut permet, lorsqu'elle est ⟨*true*⟩, de sauvegarder dans un fichier les instructions[1] ti*k*z générées par `\chemfig{`⟨*code*⟩`}`.

La clé `save name` permet de spécifier le nom du fichier (l'extension sera toujours `tex`). Sa valeur par défaut est

`\jobname-fig*`

[1] Les macros `\chemabove`, `\chembelow` et `\charge` ne sont pas traduites en instructions ti*k*z et sont laissées en l'état.

où le caractère « * », s'il est employé, remplace un nombre qui commence à 1 et est incrémenté à chaque fois que la macro `\chemfig` est appelée avec `save` = ⟨*true*⟩.

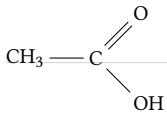
5.11 Comment faire ?

5.11.1 Écrire un atome en couleur

Comme le package `xcolor` est chargé par ti*k*z, lui même chargé par chem*fig*, on peut donc mettre dans le code d'une molécule des instructions de couleur telles que `\Red`, `\Blue`, etc[2]. Les atomes étant affichés dans des nœuds de ti*k*z qui se comportent comme des boîtes de TEX, tout se passe comme si ces atomes étaient confinés dans un groupe ce qui fait que le changement de couleur reste local à l'atome.

Couleurs

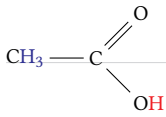
```
\chemfig{C\Blue H_3-C(=[1]O)-[7]O\Red H}
```



On constate que ce code ne fonctionne pas, toujours à cause de la règle qu'utilise chem*fig* : ici, le premier atome de la molécule commence à « `C` » et s'étend jusqu'à la prochaine lettre majuscule. Cet atome est donc « `C\Blue` » et donc, le changement de couleur se fait en fin d'atome et reste sans effet. Il faut forcer chem*fig* à couper le premier atome juste après « `C` » avec le caractère « `|` » et ensuite, englober `\color{blue}H_3` entre accolades de façons à ce que chem*fig* ne stoppe pas le deuxième atome avant le « `H` » :

Couleurs

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



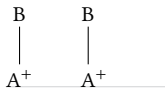
Le principal inconvénient est qu'il faut répéter l'opération pour chaque atome que l'on veut colorer, même si ceux-ci sont contigus.

5.11.2 Ajouter un exposant sans influer sur une liaison

Ajouter une ⟨*charge*⟩ sous forme d'exposant mathématique à un atome implique que la boîte (et donc le nœud de ti*k*z) le contenant aura ses dimensions modifiées. Cela n'a guerre d'importance si l'atome est un fin de chaîne mais l'esthétique peut être compromise si une liaison doit en partir. Le premier réflexe est de mettre l'exposant dans une boîte de TEX de dimensions nulles et donc d'utiliser la macro `\rlap`[3], ce qui donne des résultats souvent satisfaisants. On voit ici que grâce à `\rlap`, l'alignement horizontal des atomes est préservé :

Charge et liaison

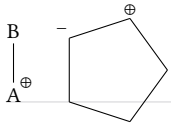
```
\chemfig{A^+-[2]B}
\qquad
\chemfig{A\rlap{${}^+$}-[2]B}
```



La macro `\charge` permet d'effectuer cette tâche de façon simple et précise.

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



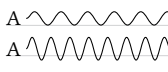
5.11.3 Dessiner une liaison courbe

Nous avons déjà vu qu'avec la librairie « `decorations.pathmorphing` » de ti*k*z, on peut dessiner une liaison ondulée :

Liaison ondulée

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

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

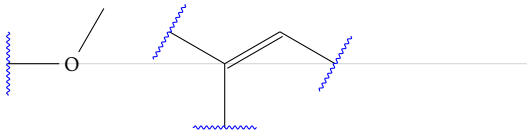


Il est également possible de définir une sous-molécule « wb » pour tracer une ondulation perpendiculaire à la liaison précédente :

[2] Pour ceux qui utilisent LATEX, il s'agira des macros `\color` et `\textcolor`
[3] Si l'on doit mettre la charge a gauche de l'atome, il faut employer la commande `\llap`.

Liaison ondulée

```
\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}}
```

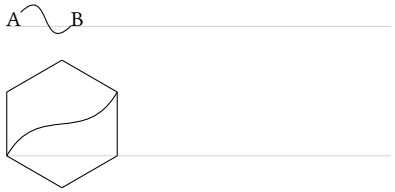


Pour plus de souplesse, on peut aussi poser des nœuds avec le marqueur « `@` » et réutiliser ces nœuds après que la molécule ait été tracée pour les relier avec une ligne courbe à l'aide de `\chemmove` :

Liaisons courbes

```
\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 Dessiner un élément de polymère

La macro `\polymerdelim` a la suivante :

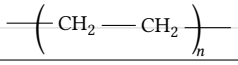
`\polymerdelim[`⟨*clés*⟩`=`⟨*valeurs*⟩`]{`⟨*nœud1*⟩`}{`⟨*nœud2*⟩`}`

L'effet, après éventuellement *deux* compilations, est de positionner des délimiteurs verticaux aux nœuds spécifiés : par défaut, le premier délimiteur est placé sur le premier nœud et le second sur la même horizontale que le premier, mais à l'abscisse du second. Les paramètres sont spécifiés via les clés et les valeurs dont voici la liste, les valeurs par défaut et les actions.

clés	valeurs par défaut	Description
`delimiters`	`()`	Définit les délimiteurs. Si ces délimiteurs sont des crochets, il faut écrire `delimiters={[]}`.
`height`	`10pt`	Définit la hauteur (au-dessus du nœud) des délimiteurs.
`depth`	⟨*vide*⟩	Définit la profondeur (au-dessous du nœud) des délimiteurs. Si la valeur est vide, alors la profondeur est égale à la hauteur.
`h align`	`true`	Booléen qui lorsqu'il est `false`, place le 2^{e} délimiteur sur le 2^{e} nœud, au risque que les délimiteurs ne soient pas sur une même ligne horizontale.
`auto rotate`	`false`	Booléen qui, lorsqu'il est égak à ⟨*true*⟩ et lorsque `h align` = ⟨*false*⟩, tourne automatiquement les délimiteurs pour qu'ils soient perpendiculaires à la ligne qui relie les deux nœuds.
`rotate`	`0`	Lorsque `h align` = ⟨*false*⟩ et `auto rotate` = ⟨*false*⟩, spécifie l'angle de rotation des deux délimiteurs.
`open xshift`	`0pt`	Définit le décalage horizontal du délimiteur ouvrant.
`close xshift`	⟨*vide*⟩	Définit le décalage horizontal du délimiteur fermant. Si la valeur est vide, alors ce décalage devient opposé au décalage du délimiteur ouvrant.
`indice`	`n`	Définit l'indice qui sera placé en bas à droite du délimiteur fermant.

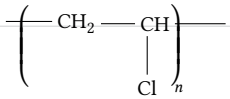
Polyéthylène

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



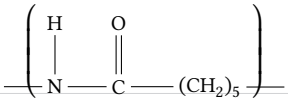
Polyvinyl chloride

```
\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}
```



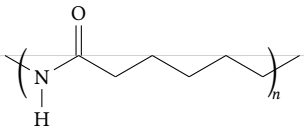
Nylon 6

```
\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}
```



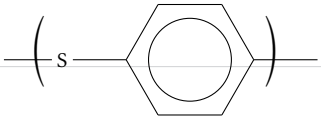
Polycaprolactame

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



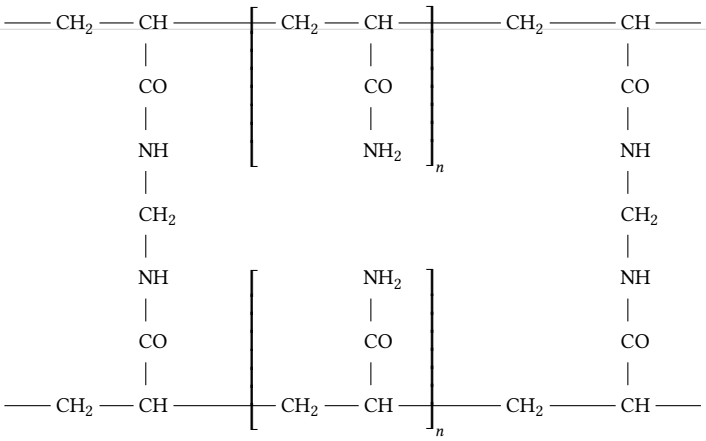
Polyphénylène sulfide

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



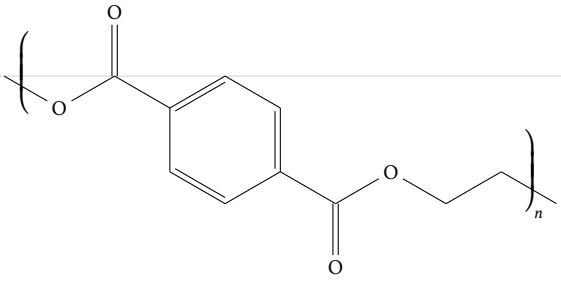
Polymère

```
\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}
```



Polymère

```
\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}
```

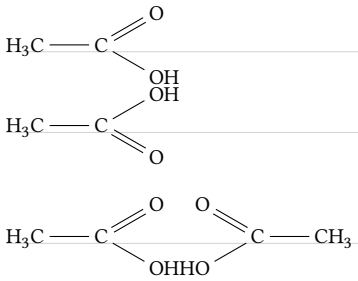


5.11.5 Dessiner le symétrique d'une molécule

Les deux commandes `\hflipnext` et `\vflipnext` qui permettent de tracer le symétrique de la prochaine molécule par rapport à un axe horizontal ou vertical. Si on souhaite construire le symétrique de plusieurs molécules, il faut écrire les commandes avant chaque molécule concernée.

Symétriques

```
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}\par% original
\vflipnext% dessine le symétrique vertical de :
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}
\bigbreak
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}% original
\hflipnext% dessine le symétrique horizontal de :
\chemfig{H_3C-C(=[:30]O)-[:-30]OH}
```



5.11.6 Ajouter du texte au dessus des liaisons et coder les angles

Une fois que l'on a compris que le caractère « `@` » permet de poser un nœud « global » pour ti*k*z, c'est-à-dire accessible après que la molécule ait été dessinée, tout ce qui est possible avec ti*k*z et les nœuds (c'est-à-dire beaucoup de choses) devient envisageable.

Pour écrire quelque chose au-dessus ou au-dessous d'une liaison, on peut donc poser des nœuds « globaux » sur les atomes situés aux extrémités de cette liaison et écrire à mi-chemin de ces deux nœuds un texte que l'on aura soin d'élever ou d'abaisser pour qu'il se situe juste au dessus ou au dessous de la liaison. C'est ce que fait la commande `\bondname` dans le code ci-dessous.

Pour les angles entre deux liaisons partant d'un atome, 3 atomes sont impliqués et il faudra poser un nœud global sur chacun d'eux. On peut aisément calculer l'angle entre deux lignes issues d'un même nœud, c'est ce que fait la commande `\arcbetweennodes`. Ensuite, la macro `\arclabel` trace un arc de cercle entre deux liaisons et écrit un texte à côté de cet arc : l'argument optionnel de cette macro est le code passé à ti*k*z. L'argument 2 est le rayon de l'arc tandis que les trois arguments suivants sont les noms des nœuds globaux entre lesquels doit être tracé l'arc, le nom du milieu devant être celui du sommet de l'angle. Enfin, le dernier argument est le texte à écrire.

Angles et texte sur les liaisons

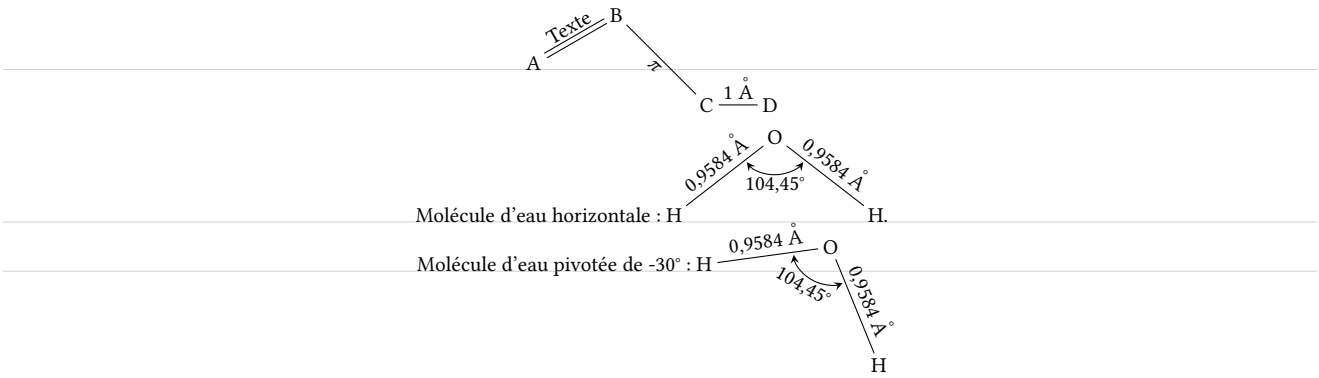
```
\def\angstrom{\charge{90:-1pt=°}{A}}
\def\namebond#1#2#3#4{\chemmove{\path(#2)--(#3)node[midway,sloped,yshift=#1]{#4};}}

\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}{ Texte}
\namebond{-3.5pt}{b}{c}{$\pi$}
\namebond{5pt}{c}{d}{1 \angstrom}
\medbreak
Molécule d'eau horizontale : \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
Molécule d'eau pivotée de -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°}
```

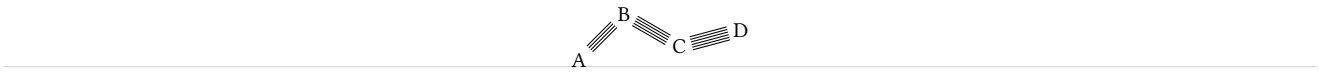


5.11.7 Dessiner des liaisons multiples

Là encore, la librairie « `decorations.markings` » permet de tracer des liaisons multiples :

Liaisons multiples

```
\catcode`\_=11
\tikzset{nbond/.style args={#1}{%
    draw=none,%
    decoration={%
        markings,%
        mark=at position 0 with {\coordinate (CFstart@) at (0,0);},
        mark=at position 1 with {%
        \foreach\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}
```

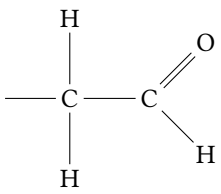


5.12 Exemples commentés

Dans ce chapitre, plusieurs molécules seront dessinées en mettant en œuvre les méthodes précédemment exposées. Le but recherché ici est de montrer dans quel ordre logique peut se construire une molécule de façon à ce que l'utilisateur peu familier avec ch*emfig* acquière une méthode pour construire des molécules complexes. Pour l'y aider, les étapes de la construction seront montrées.

De plus, on mettra en évidence que plusieurs possibilités s'offrent à l'utilisateur pour un même résultat graphique, certaines intuitives et d'autres beaucoup moins, l'essentiel étant de montrer que ch*emfig* permet une certaine souplesse dans le codage des molécules. À chacun ensuite de se construire et s'approprier les méthodes avec lesquelles il est le plus à l'aise.

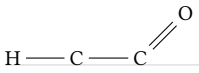
5.12.1 L'éthanal

On va ici dessiner la molécule d'éthanal : 

La meilleure méthode pour les molécules non cycliques est de choisir la plus longue chaîne. Ici on peut prendre « `H-C-C=O` » par exemple. Il faut incliner la liaison `C=O` de 45° en utilisant l'angle prédéfini « `[1]` ». On obtient la « structure » de la molécule sur laquelle il suffira de rajouter les ramifications :

Structure de l'éthanal

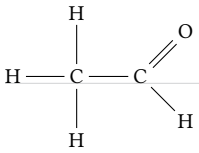
```
\chemfig{H-C-C=[1]O}
```



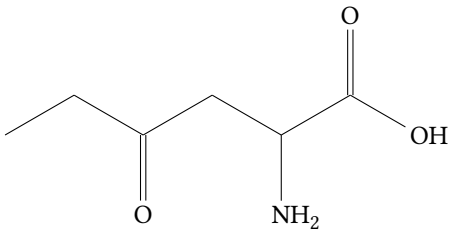
Il reste à placer 3 atomes d'hydrogène avec les bonnes inclinaisons à l'aide des angles prédéfinis. Le premier à 90° avec la ramification « `(-[1]H)` », le second à 270° avec « `(-[6]H)` » et celui de droite à 315° avec « `(-[7]H)` » :

Ethanal

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



5.12.2 L'acide 2-amino-4-oxohexanoïque

Voici la molécule à dessiner : 

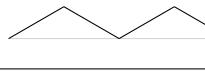
Il existe — comme c'est le souvent cas pour la plupart des molécules — plusieurs méthodes, et pour chacune, plusieurs façons différentes d'arriver au résultat. Ici, nous allons examiner 4 méthodes différentes.

Méthode avec des angles absolus

On va tout d'abord tracer la chaîne médiane avec des angles absolus. On règle l'angle par défaut à +30° avec l'argument optionnel, ainsi seules les liaisons « descendantes » auront besoin que l'on spécifie leur angle absolu de −30° :

Structure (angles absolus)

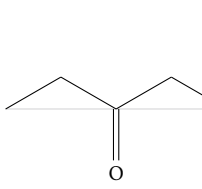
```
\chemfig{[:30]--[:-30]--[:-30]--[:-30]OH}
```



Il reste à ajouter les ramifications « `(=[6]O)` », « `(-[6]NH_2)` » et « `(=[2]O)` » sur les bons sommets :

Molécule (angles absolus)

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

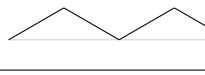


Méthode avec des angles relatifs

Une approche plus générale aurait été de n'utiliser que des angles relatifs. Il aurait fallu procéder de cette façon :

Structure (angles relatifs)

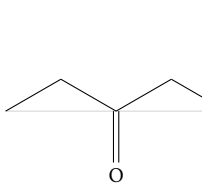
```
\chemfig{[:30]--[::-60]--[::-60]--[::-60]OH}
```



puis

Molécule (angles relatifs)

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

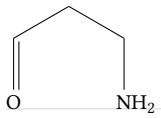


Méthode avec un cycle

Les angles entre les liaisons étant de 120°, on peut penser à utiliser un 6-cycle, quoique cette méthode soit moins naturelle. Il faut ici profiter du fait qu'un cycle peut être incomplet. Il est également nécessaire de faire pivoter le cycle de 120° pour que le premier sommet soit au sud-est du cycle :

Structure

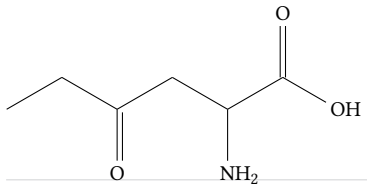
```
\chemfig{[:120]NH_2*6(---=O)}
```



Il faut maintenant faire partir les ramifications des sommets adéquats :

Molécule (cycle)

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

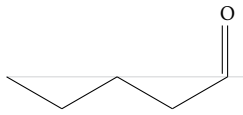


Méthode avec des cycles accolés

En approfondissant la méthode avec les cycles, on peut aussi penser à imbriquer des 6-cycles incomplets. On pourrait partir de cette structure :

Structure

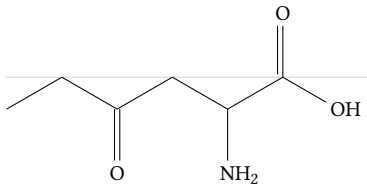
```
\chemfig{*6(--*6(--=O))}
```



Et ensuite ajouter les liaisons qui partent des sommets des ces cycles. Il n'y a pas à se préoccuper des angles puisque les liaisons qui partent des cycles sont les bissectrices des côtés du cycle, ce que justement, on cherche ici :

Molécule (cycles imbriqués)

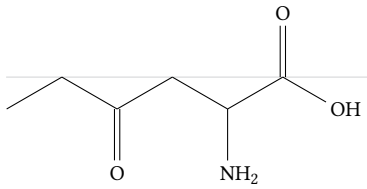
```
\chemfig{*6((-)-(=O)-*6(-(-NH_2)-(-OH)=O))}
```



Un examen attentif révèle cependant que la seconde double liaison vers l'atome d'oxygène se trouve *à l'intérieur* du 6-cycle incomplet[4]. Malgré sa concision, ce code ne conduit donc pas à un dessin parfait. On peut bien sûr corriger ce défaut en allongeant un peu le code :

Molécule (cycles imbriqués corrigés)

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



5.12.3 Glucose

Le but ici est de représenter selon différentes conventions la molécule de glucose.

Formule topologique

Le code ressemble ici à celui de l'acide 2-amino-4-oxohexanoïque. On obtient quasiment la même structure avec des angles absolus sauf qu'ici, l'angle par défaut est de −30° :

Structure

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

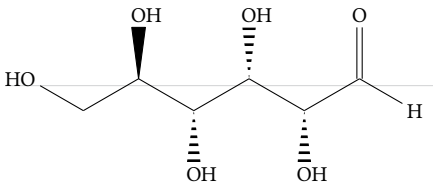


Et l'ajout des ramifications ne pose pas de problème particulier. On utilise des angles absolus :

[4] C'était aussi le cas pour la méthode précédente avec un seul cycle.

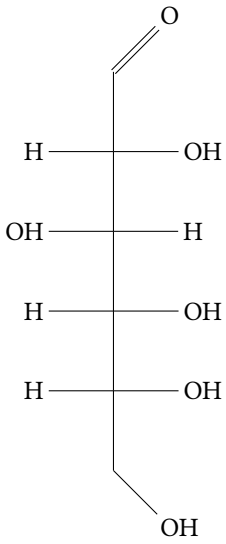
Fructose, formule topologique

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



Projection de Fisher

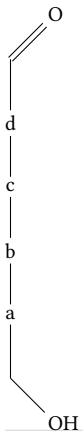
Le but est d'obtenir la molécule ci dessous :



L'idée est de commencer à dessiner la plus longue chaîne verticale en indiquant un angle « `[2]` » par défaut. Voici la structure où l'on met volontairement des lettres minuscules au bout de chaque liaison verticale :

Structure

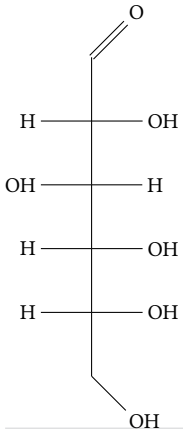
```
\chemfig{[2]OH-[3]-a-b-c-d-=[1]O}
```



Ensuite, on définit deux alias pour les liaisons horizontales et les atomes qui les termineront. Prenons « `x` » que nous mettrons à la place des minuscules a, c et d et « `y` » qui prendra la place de la lettre c. Comme les alias ne comportent qu'un seul caractère, on peut se passer d'accolade et écrire « `!x` » au lieu de « `!{x}` » :

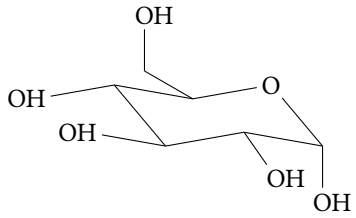
Glucose (projection de Fisher)

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



Représentation « chaise »

On va représenter cette molécule d'α-D-glucose :



Pour cela, on va tout d'abord tracer 5 côtés de la chaise et relier le premier sommet au dernier avec un crochet « `?` ». On adopte les angles absolus suivants, donnés dans l'ordre de parcourt trigonométrique : $-50°$, $10°$, $-10°$, $130°$, $190°$.

Structure

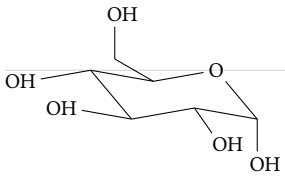
```
\chemfig{?-[:-50]-[:10]-[:-10]-[:130]O-[:190]?}
```



Maintenant, il suffit de rajouter les ramifications entre parenthèses. Les angles sont choisis au mieux pour rendre une impression de perspective, et certaines liaisons sont raccourcis d'un coefficient de 0,7 :

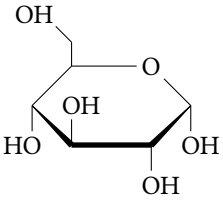
Représentation chaise

```
\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)}
```



Projection de Haworth

Le but est de représenter cette molécule de D-glucopyranose :

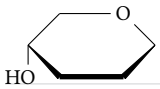


Tout d'abord, on va choisir la plus longue chaîne qui part du groupe « HO » de gauche et continue sur 5 côtés du cycle. Le cycle sera fermé avec un crochet. Pour la liaison verticale qui part du premier groupe « HO », il faut spécifier qu'elle doit partir du second atome avec l'argument optionnel. De plus, elle sera raccourcie avec un coefficient de 0,5. Son argument optionnel sera donc « `[2,0.5,2]` ».

Ensuite, pour donner une impression de perspective au cycle, les liaisons inclinées seront réduites d'un coefficient de 0,7. Pour les traits gras inclinés, on va se servir des liaisons de Cram en ayant redéfini la largeur de la base des triangles à 2pt. Pour la liaison horizontale en trait gras, il faudra la tracer avec une épaisseur de 2pt et son argument optionnel sera donc « `[0,,,,line width=2pt]` ». Voici la structure de la molécule :

Structure

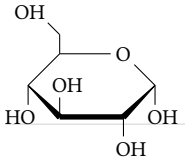
```
\chemfig[cram width=2pt]{HO-[2,0.5,2]?<[7,0.7]-[,,,,
line width=2pt]>[1,0.7]-[3,0.7]O-[4]?}
```



Il ne reste plus qu'à rajouter les ramifications, mises au bon endroit, en spécifiant des angles absolus correctement choisis et des longueurs parfois réduites pour mieux donner l'illusion de la perspective :

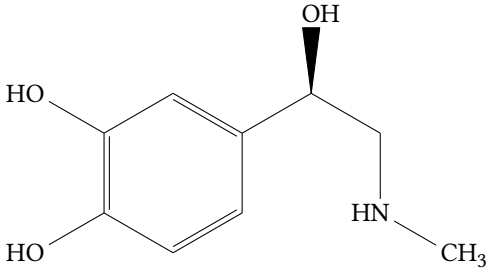
Projection de Haworth

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



5.12.4 Adrénaline

On cherche à dessiner la molécule d'adrénaline :



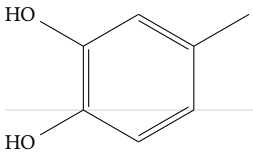
Nous allons utiliser deux méthodes différentes.

Utilisation d'un cycle

Tout d'abord, on part du 6-cycle et nous dessinons le début des ramifications qui en partent :

Structure de l'adrénaline

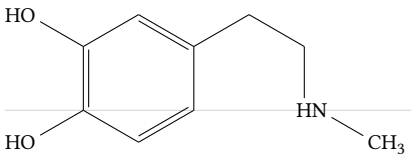
```
\chemfig{*6((-HO)-=-(-)=-(-HO)=)}
```



Il faut maintenant compléter la ramification de droite en utilisant, par exemple, des angles relatifs :

Adrénaline, étape 2

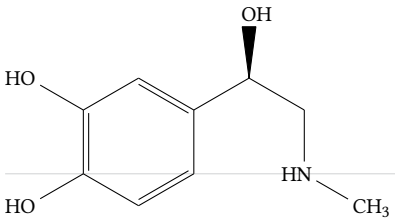
```
\chemfig{*6((-HO)-=-(--[::-60]-[::-60]
HN-[::+60]CH_3)=-(-HO)=)}
```



Puis, il faut rajouter la liaison de Cram vers `OH` et spécifier que la liaison qui arrive sur « `NH` » le fait sur le second atome « N ». Nous utilisons le 4[e] argument optionnel de la liaison :

Adrénaline

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



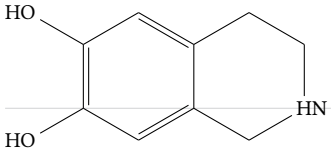
Utilisation de 2 cycles

Cette méthode est moins naturelle, mais le but est ici d'expliquer comment rendre une liaison invisible.

On pourrait améliorer ce code en considérant que le dessin de la molécule d'adrénaline est constitué de deux 6-cycles accolés l'un à l'autre :

Adrénaline, structure 2 cycles

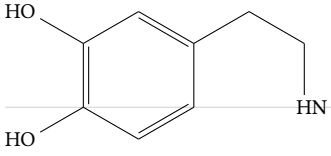
```
\chemfig{*6((-HO)-=*6(--HN---)-=-(-HO)=)}
```



Il faut donc rendre invisible les deux premières liaisons du cycle de droite. Pour cela, on se sert de l'argument qui est passé à ti*k*z en spécifiant « `draw=none` ». Ces liaisons ont donc ce code : « `-[,,,,,draw=none]` ». Pour que le code reste lisible, on définit un alias nommé « `&` » pour ces liaisons :

Adrénaline, étape 2

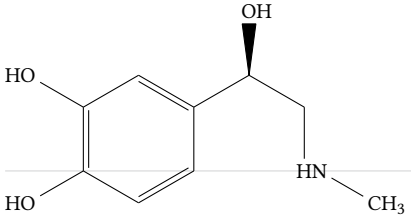
```
\definesubmol{&}{-[,,,,draw=none]}
\chemfig{*6((-HO)-=*6(!&!&HN---)-=-(-HO)=)}
```



Le reste devient facile, il suffit d'ajouter les ramification aux bons sommets :

Adrénaline, étape 3

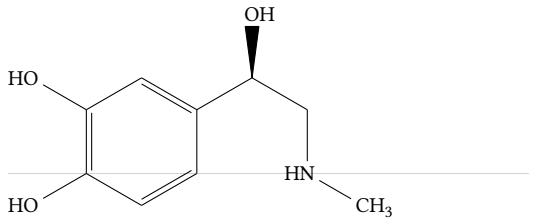
```
\definesubmol{&}{-[,,,,draw=none]}
\chemfig{*6((-HO)-=*6(!&!&HN(-CH_3)--(<OH)-)-=-(-HO)=)}
```



Pour finir, on spécifie que les liaisons qui *arrivent et partent* de « `HN` » doivent le faire sur deuxième atome. On définit donc un autre alias pour la liaison invisible qui arrive sur « `HN` » :

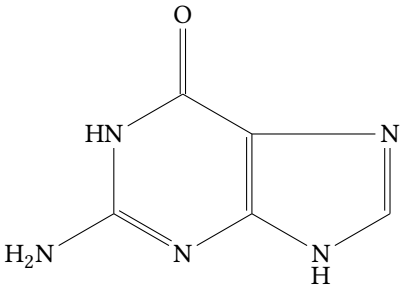
Adrénaline

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



5.12.5 Guanine

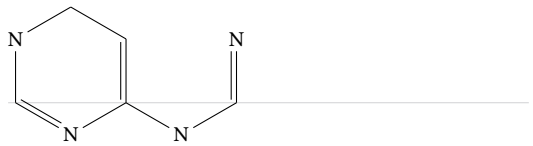
Nous allons dessiner la molécule de guanine :



Tout d'abord, commençons par dessiner les cycles imbriqués en mettant seulement les atomes d'azote aux sommets :

Guanine, structure

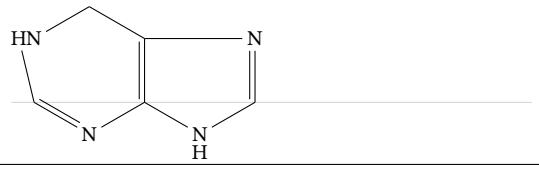
```
\chemfig{*6(=N-*6(-N-=N)=--N-)}
```



Puis nous allons tracer la liaison horizontale dans le cycle de droite avec un crochet. Nous allons aussi positionner un atome d'hydrogène sous l'atome d'azote du 5-cycle avec la commande `\chembelow{N}{H}`. Il faut aussi écrire « `HN` » au lieu de « `N` » au sommet en haut à gauche de la molécule :

Guanine, étape 2

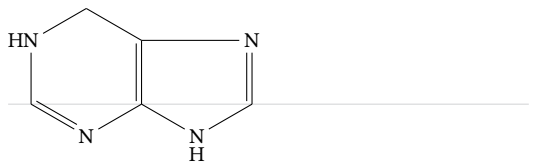
```
\chemfig{*6(=N-*6(-\chembelow{N}{H}-=N?)=?--HN-)}
```



On constate qu'une liaison part du mauvais atome[5] ! Il faut corriger le mécanisme de calcul automatique pour que la liaison parte du 2e atome « `N` » au lieu du premier. Pour cela, on spécifie un argument optionnel pour la dernière liaison du premier 6-cycle « `[,,2]` » :

Guanine, étape 3

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

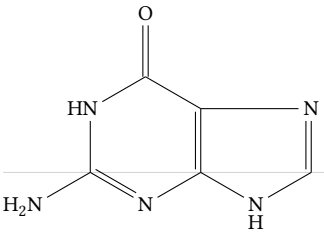


Il suffit de rajouter les ramifications aux bons sommets. On peut notamment remarquer la ramification qui part du premier sommet du premier 6-cycle « `(-H_2N)` » :

[5] Ceci semble illogique puisque l'angle de la liaison du groupe `HN` vers le premier sommet du 6-cycle est comprise entre −90° et 90° ; chem*fig* devrait donc partir du 2e atome. Pour expliquer cette contradiction, il faut savoir que dans les cycles, la dernière liaison relie toujours le dernier sommet au premier sommet en ignorant l'angle *calculé théorique* de cette liaison (qui est ici de −90°). chem*fig* utilise cet angle théorique pour déterminer les atomes de départ et d'arrivée, mais ne s'en sert pas pour tracer la liaison puisque les deux extrémités sont déjà définies. L'atome de départ de la dernière liaison est donc le premier.

Guanine

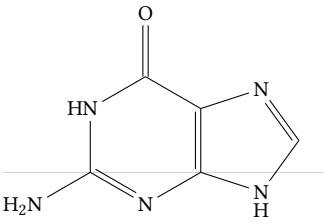
```
\chemfig{*6((-H_2N)=N-*6(-\chembelow{N}{H}-=N?)=?-(=O)-HN-[,,2])}
```



On aurait aussi pu dessiner cette même molécule avec un 5-cycle *régulier*, comme cela se fait parfois :

Guanine avec 5-cycle

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



6 Schémas réactionnels

Je remercie Clemens Niederberger pour l'aide qu'il m'a apportée et les tests qu'il a effectués sur les nouvelles fonctionnalités présentées dans cette partie.

6.1 Généralités

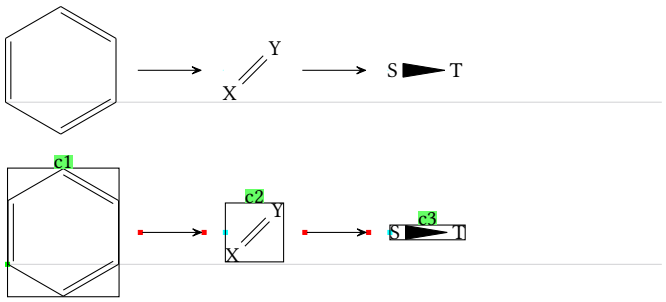
Un schéma réactionnel se trouve dans l'environnement délimité par les macros `\schemestart` et `\schemestop`. La syntaxe de la macro `\schemestart` est

`\schemestart[`⟨*clés*⟩`=`⟨*valeurs*⟩`]` ⟨*réaction*⟩ `\schemestop`

Comme on le constate sur cet exemple, des informations de débogage ont été masquées puis rendues visibles pour aider à la compréhension avec la clé `scheme debug` et la valeur ⟨*true*⟩ ou ⟨*false*⟩ :

Exemple 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
```



Quelques remarques :

— les commandes `\arrow` tracent les flèches ;

— tout ce qui se trouve entre deux commandes `\arrow` est considéré comme étant un composé. Le parti a été pris que tous les réglages possibles, qu'ils concernent les flèches ou les composés, le seront grâce aux arguments de la commande `\arrow`, dont la syntaxe en devient assez complexe ;

— les flèches sont tracées horizontalement, ce qui est bien évidemment modifiable ;

— les flèches sont sur la ligne imaginaire reliant les centre des boîtes englobantes les composés (les points rouges et bleus sont les points d'attache des flèches). Ce comportement est également modifiable ;

— Les informations de débogage, qui sont rendues visibles ou pas avec la clé `scheme debug` sont :

- ce qui se trouve en vert au dessus des boîtes englobantes est le nom attribué par défaut par chem*fig* aux composés et suit la progression `c1`, `c2`, etc. La numérotation recommence à 1 pour un nouveau schéma réactionnel.
- les boîtes englobantes des composés ;
- les extrémités des flèches sont matérialisés par des points rouge et les ancres par des points bleu ;

— la distance de bord à bord entre deux composés est définie avec la clé `compound sep` = ⟨*dim*⟩. La valeur par défaut de la ⟨*dim*⟩ est 5em ;

— enfin, la distance entre les bords des composés et le début et la fin des flèches est défini avec la clé `arrow offset` = ⟨*dim*⟩. Par défaut, cette ⟨*dim*⟩ vaut `1em`.

Voici la liste exhaustive des paramètres concernant la macro `\schemestart`, leurs valeurs par défaut et une brève description. Ces paramètres doivent être réglés par la macro `\setchemfig{`⟨*clés*⟩`=`⟨*valeurs*⟩`}`.

clés	valeurs par défaut	Description
`schemestart code`	⟨*vide*⟩	code arbitraire exécuté au tout début de l'environnement non imbriqué `\schemestart` (juste après le `\begingroup`)
`schemestop code`	⟨*vide*⟩	code arbitraire exécuté à la toute fin de l'environnement non imbriqué `\schemestart` (juste avant le `\endgroup`)

clés	valeurs par défaut	Description
init anchor	text	ancre du premier composant
scheme debug	false	si ⟨*true*⟩, affiche les frontières des composés, leur nom et les ancres
compound style	⟨*vide*⟩	style des composés
compound sep	5em	espacement entre 2 composés
name sep	1.5ex	distance verticale minimale entre les boites du composé et de son nom
arrow offset	1em	espacement entre un composé et la flèche
arrow angle	0	angle par défaut des flèches de réaction
arrow coeff	1	coefficient de longueur des flèches
arrow style	⟨*vide*⟩	style des flèches
arrow double sep	2pt	espacement entre les flèches doubles
arrow double coeff	0.6	coefficient de réduction de la petite flèche pour les flèches doubles
arrow double harpoon	true	booléen pour le tracé des doubles flèches sous forme de « harpon »
arrow label sep	3pt	espacement entre la flèche et son texte
arrow head	-CF	style des pointes de flèches
+ sep left	0.5em	espacement avant le signe +
+ sep right	0.5em	espacement après le signe +
+ vshift	0pt	décalage vertical du signe +

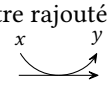
6.2 Types de flèches

Lorsque la commande `\arrow` est suivie d'un argument optionnel entre accolades (qui n'est donc pas obligatoire), cet argument contient le type de flèche :

Types de flèches

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



La flèche « `-U>` » n'est pas dessinée de façon complète, un arc de cercle qui lui est tangent en son milieu peut être rajouté avec les arguments optionnels passés à cette flèche, voir page 51. Voici une flèche « `-U>` » avec son arc de cercle : A 

B

Par souci de concision, par la suite, sauf exemple spécial, des lettres majuscules seront mises au lieu de formules chimiques crées via la commande `\chemfig`. Il est bien évident que les schémas réactionnels fonctionnent de façon identique avec des lettres ou des dessins de molécules. On peut voir dans la galerie plusieurs exemples où les schémas réactionnels entrent en jeu.

6.3 Caractéristiques des flèches

Chaque flèche est caractérisée par :

— un angle exprimé en degrés que l'on règle avec `arrow angle` = ⟨*angle*⟩ et qui vaut 0 par défaut ;
— un coefficient spécifié par `arrow coeff` = ⟨*decimal*⟩, qui vaut 1 par défaut et qui indique la longueur de la flèche par multiplication de la valeur de l'espacement entre composés défini par `compound sep` ;
— un style, défini par `arrow style` = ⟨*code tikz*⟩, vide par défaut et qui contient des instructions ti*k*z qui vont permettre de personnaliser la couleur, l'épaisseur de la flèche ou un autre attribut graphique.

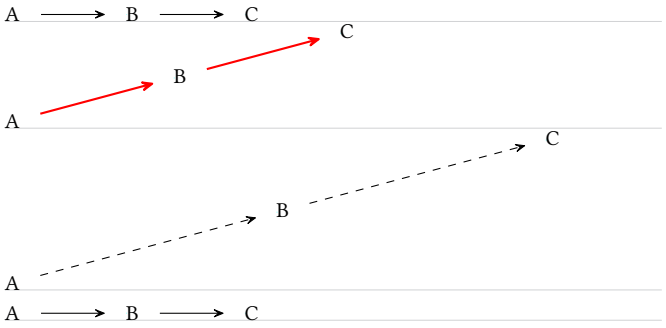
Définition de valeurs par défaut

```
\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
```



Pour modifier une ou toutes ces valeurs par défaut, on peut agir via `\setchemfig` ou par l'argument optionnel de `\schemestart` afin de confier les modifications au schéma en cours.

Argument optionnel

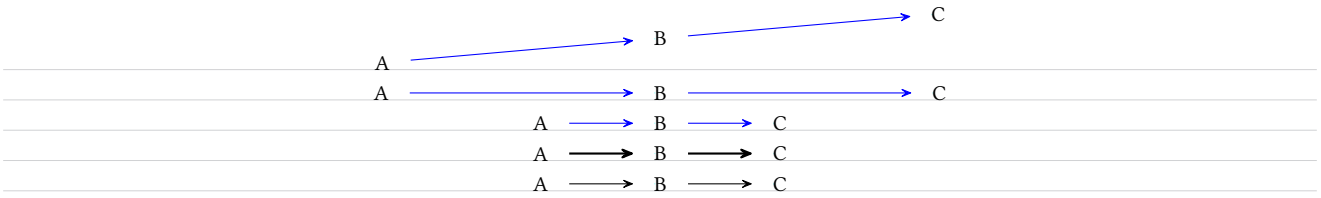
```
\setchemfig{arrow angle=5,arrow coeff=2.5,arrow style=blue}
\schemestart A\arrow B\arrow C\schemestop

\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
```



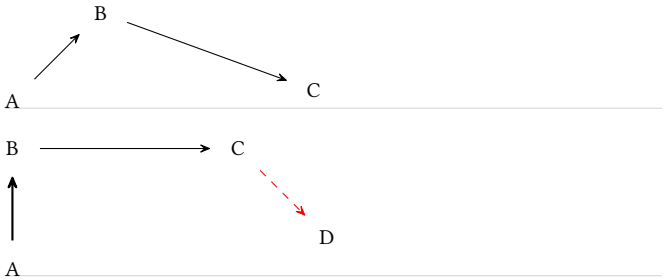
Pour le style, la règle est la suivante : le style spécifié dans l'argument entre crochets s'applique *après* le style par défaut, sans l'écraser ! C'est pour cette raison que seul l'attribut de couleur « black » parvient à écraser celui par défaut qui est « blue ».

Enfin, la commande `\arrow` admet un argument optionnel entre crochets de la forme [⟨*angle*⟩,⟨*coeff*⟩,⟨*style*⟩] pour modifier les caractéristiques de la flèche concernée. Pour le style, comme précédemment, celui est-ci est appliqué *après* le style par défaut et l'éventuel style spécifié dans l'argument optionnel de la commande `\schemestart`, le tout sans écrasement.

Caractéristiques des flèches

```
\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 Nom des composés

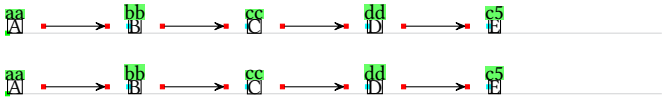
On peut passer outre le mécanisme automatique qui nomme les composés « c1 », « c2 », etc. Pour cela, la commande `\arrow` admet un argument entre parenthèses qui doit suivre immédiatement la commande. Cet argument est de la forme : `(n1--n2)`. Il nomme « `n1` » le composé se trouvant à l'origine de la flèche et « `n2` » celui se trouvant à sa fin. N'importe quelle chaîne de caractères alphanumériques peut être utilisée. La numérotation des noms « `c`⟨*n*⟩ » continue en interne ce qui signifie que si un composé porte un autre nom que celui par défaut, cela ne change pas le nom par défaut des composés suivants.

Les noms sont facultatifs et l'argument peut aussi bien être `(n1--)` que `(--n2)`.

Nom des composés

```
\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
```



On le voit, les deux méthodes sont équivalentes et l'on peut donc nommer un composé avec la flèche qui le précède ou avec la flèche qui le suit. En revanche, lorsqu'un composé est entouré de deux flèches spécifiant son nom, le premier nom est ignoré et un « warning » est émis :

Noms surabondants

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



Ici, le composé « B » est nommé « foo » par la flèche qui lui arrive dessus et « bar » par celle qui en part. chem*fig* émet donc le warning suivant annonçant que le nom « foo » sera ignoré :

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

6.5 Ancres

Comme on l'a vu, les flèches sont sur la ligne reliant les centre de boites englobant les composés. On peut spécifier un ancre autre que celui par défaut (qui, au sens de ti*k*z, s'appelle « `center` »). On utilise pour cela l'argument entre parenthèses :

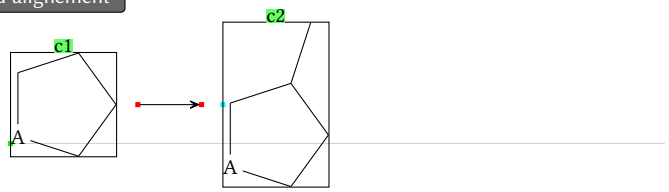
```
(n1.a1--n2.a2)
```

où l'ancre « `a1` » ou « `a2` » peut être : `north west`, `north`, `north east`, `west`, `center`, `east`, `mid west`, `mid`, `mid east`, `base west`, `base`, `base east`, `south west`, `south`, `south east`, `text`, ou n'importe quel angle.

Voici un exemple où l'alignement par défaut n'est pas bon car les deux « A » ne sont pas alignés verticalement. Les informations de débogage ont été rendues visibles pour constater que les ancres « `center` » par défaut ne conviennent pas :

Problèmes d'alignement

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

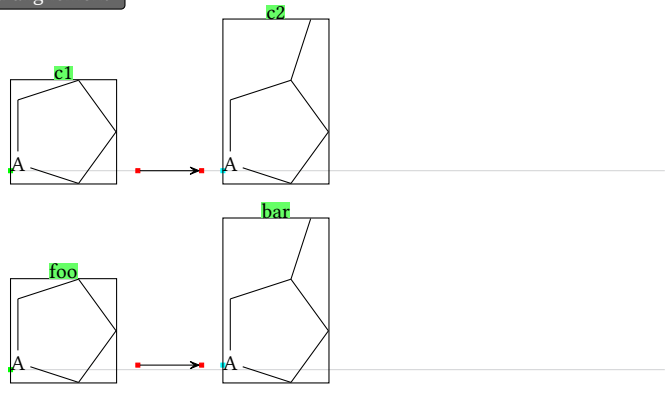


Pour que l'alignement soit correct, on va faire partir/arriver la flèche soit des ancres « `base east` »/« `base west` », soit des ancres « `mid east` »/« `mid west` » :

Problèmes d'alignement

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

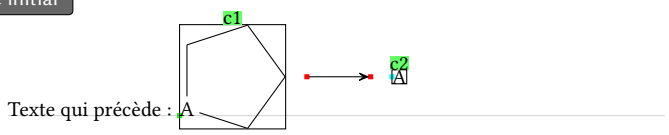
\schemestart
  \chemfig{A*5(-----)}
  \arrow(foo.mid east--bar.mid west)
  \chemfig{A*5(---(-)--)}
\schemestop
```



Un dernier ancre reste à spécifier, c'est celui du premier composé par rapport à la ligne de base du texte qui le précède, il s'agit du point vert de gauche sur le schéma qui suit :

Ancre initial

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



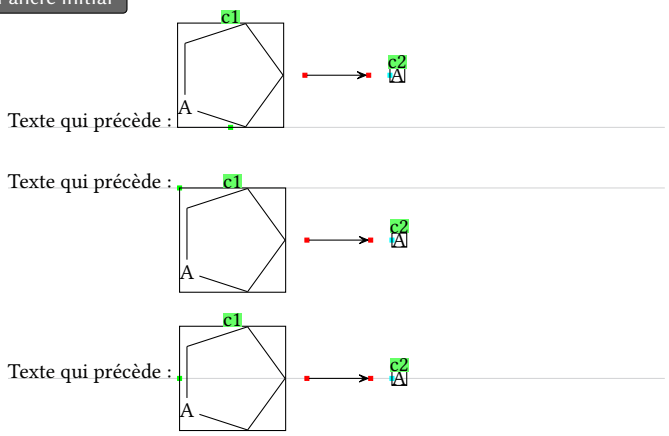
L'emplacement de cet ancre sur la boîte contenant le premier composé vaut par défaut « `text` ». Il est réglable avec la clé `init anchor` :

Réglage de l'ancre initial

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

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

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



6.6 Style des composés

Toujours par l'intermédiaire de l'argument entre parenthèses de la commande `\arrow`, on peut spécifier avec des instructions de ti*k*z le style « ⟨*s1*⟩ » ou « ⟨*s2*⟩ » de la boîte englobante du composé de départ ou de celui d'arrivée. La syntaxe complète de la commande `\arrow` est donc la suivante, où chaque spécification est optionnelle et peut être omise :

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

Tout comme les noms, si un style est spécifié à un composé par la flèche qui lui arrive dessus et par celle qui en part, le premier style sera ignoré avec un warning.

Style des composés

```
\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
```



La clé `compound style` = ⟨*code tikz*⟩ permet de définir de façon générale le style des composés affichés par la suite. Il faut bien évidemment entrer un argument vide pour qu'aucun style ne soit appliqué, ce qui est le cas par défaut.

Ici, nous allons définir un style en forme de boites aux coins arrondis, dont le fond sera semi transparent :

Styles globaux

```
\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 Embranchements

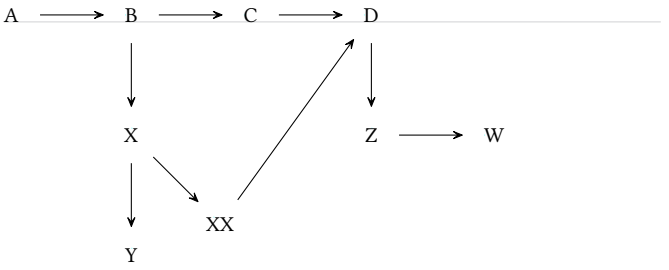
Jusqu'à présent, on n'a pu tracer que des schémas réactionnels linéaires. On peut également faire des embranchements et c'est là que le nom des composés va prendre toute son importance. Dans l'argument entre parenthèses de la commande `\arrow`, si un nom est précédé de « `@` », cela signifie que ce composé existe déjà. Plusieurs cas de figure peuvent se présenter :

- `(@n1--n2)` : la flèche sera tracée depuis le composé existant « `n1` » et le schéma continuera à la suite de cette flèche, créant donc un embranchement ;
- `(@n1--@n2)` : la flèche est tracée entre deux composés existants, qu'ils soient déjà définis ou qu'ils soient définis plus tard dans le schéma réactionnel : cette syntaxe peut donc se placer *n'importe où* dans le code du schéma réactionnel ;
- `(n1--@n2)` : cette syntaxe n'est pas admise ;

Voici un exemple où 3 embranchements sont créés, l'un partant de « B », l'autre de « D » et le dernier de « X ». À la fin, une flèche est tracée entre 2 composés existants « XX » et « D » :

Embranchements

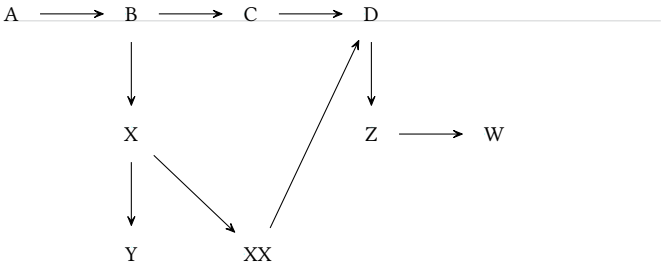
```
\schemestart
  A\arrow(aa--bb)B\arrow(--cc)C\arrow D
  \arrow(@bb--xx1)[-90]X\arrow[-90]Y% 1er embranchement
  \arrow(@c4--)[-90]Z\arrow W% 2è embranchement
  \arrow(@xx1--xx2)[-45]XX% 3è embranchement
  \arrow(@xx2--@c4)% flèche XX à D
\schemestop
```



On pourrait vouloir que « Y » et « XX » soient sur une même horizontale. Pour cela, on va tracer une liaison invisible horizontale de « Y » à « XX » et finir le schéma par une flèche entre les deux composés existants « XX » et « D » :

Embranchements

```
\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)% flèche de X à XX
  \arrow(@xx2--@c4)% flèche XX à D
\schemestop
```

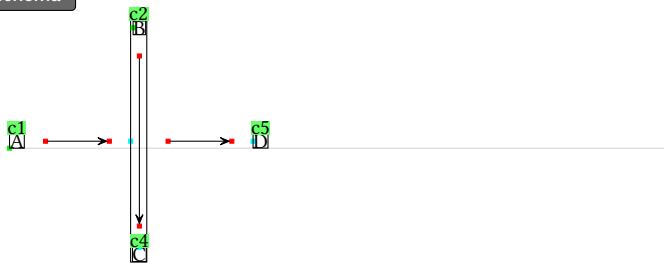


6.8 Sous-schéma

On peut placer une partie d'un schéma réactionnel dans une boîte englobante unique, considérée par chem*fig* comme un composé. La commande `\subscheme` dont l'argument obligatoire contient le sous-schéma entre accolades permet de grouper un sous-schéma comme un tout et lorsque `\subscheme` se trouve après une flèche, la commande enveloppe ce sous-schéma dans un composé portant le nom « `c`⟨*n*+*1*⟩ » :

Sous-schéma

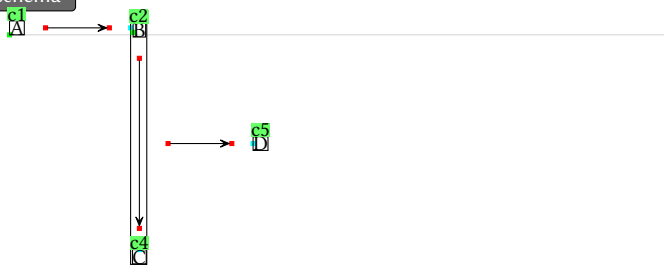
```
\setchemfig{scheme debug=true}
\schemestart
  A\arrow
  \subscheme{B\arrow[-90,2]C}
  \arrow
  D
\schemestop
```



Bien que ça ne soit pas clair car les noms se superposent, la boîte contenant le sous-schéma porte le nom « `c2` » et la numérotation se poursuit à l'intérieur du sous-schéma « `c3` » pour B, « `c4` » pour C. Comme le premier composé du sous-schéma est « B », la ligne de base du sous-schéma est celle de « B ». On peut le mettre en évidence en précisant les ancres :

Sous-schéma

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



Il faut noter que la commande `\subscheme{`⟨*scheme*⟩`}` n'étant rien d'autre qu'un raccourci pratique pour

`\schemestart`⟨*scheme*⟩`\schemestop`

Par conséquent, `\subscheme` admet les mêmes arguments optionnels que `\schemestart`.

chem*fig* fournit la paire de commandes `\chemleft` et `\chemright` qui permettent de placer des délimiteurs extensibles de part et d'autre d'un matériel. Ces commandes doivent être suivies de délimiteurs, tout comme on en mettrait à la suite des primitives de TEX `\left` et `\right` :

`\chemleft`⟨*car1*⟩⟨*matériel*⟩`\chemright`⟨*car2*⟩

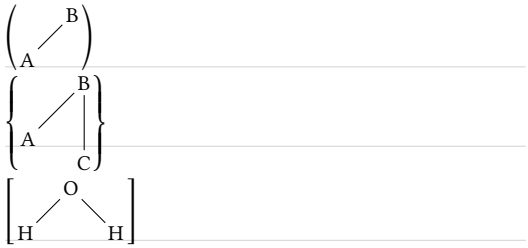
où ⟨*car1*⟩ et ⟨*car2*⟩ sont « (» et «) » ou « [» et «] » ou tout autre délimiteur extensible accepté par les commandes `\left` et `\right`.

Les macros \chemleft et \chemright

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

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

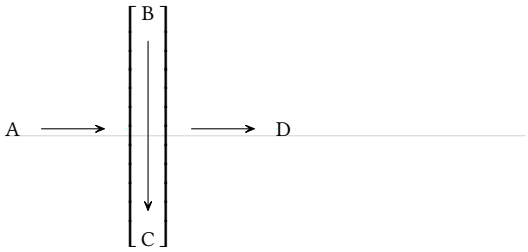
\chemleft[\chemfig{H-[1]O-[7]H}\chemright]
```



Voici le code du schéma réactionnel vu précédemment où l'on utilise `\chemleft` et `\chemright` :

Schéma réactionnel avec \chemleft et \chemright

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



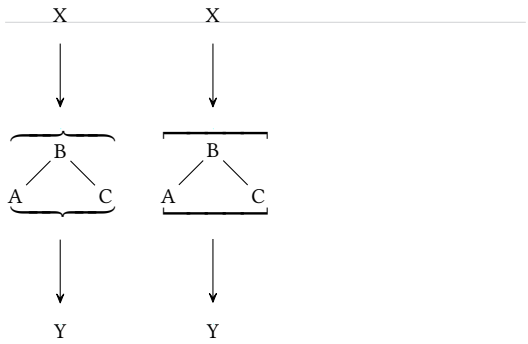
Dans le même ordre d'idée, les macros `\chemup` et `\chemdown` fonctionnent de la même façon sauf que les délimiteurs extensibles sont positionnés au dessus et au dessous du matériel :

`\chemup`⟨*car1*⟩⟨*matériel*⟩`\chemdown`⟨*car2*⟩

Voici un exemple :

Les macros \chemup et \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
```



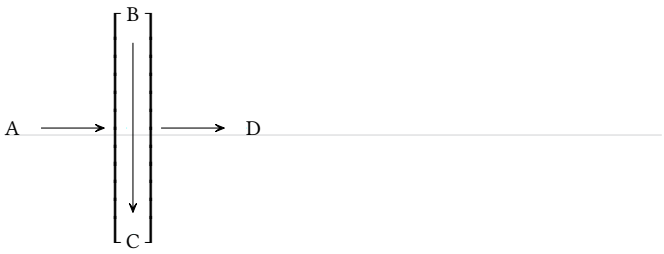
Avec les styles des composés, il est également possible de délimiter un composé quelconque (et donc un sous-schéma) par des délimiteurs extensibles (parenthèses, crochets, accolades), ceci après avoir chargé la librairie de ti*k*z « `matrix` » en mettant dans le préambule du document :

```
\usetikzlibrary{matrix}
```

Comme les commandes `\chemleft`, `\chemright`, `\chemup` et `\chemdown` sont disponibles, l'extension chem*fig* ne procède *pas* à ce chargement, il appartient donc à l'utilisateur de le faire s'il souhaite avoir accès à ces délimiteurs.
Voici le sous-schéma précédent délimité par des crochets :

les délimiteurs de la librairie «matrix»

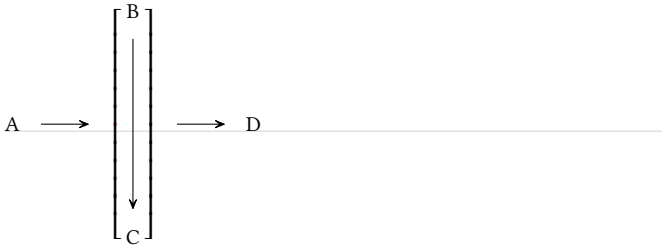
```
%\usetikzlibrary{matrix}
\schemestart
  A\arrow(--[left delimiter={[}, right delimiter={]}])
  \subscheme{B\arrow[-90,2]C}
  \arrow
  D
\schemestop
```



Les délimiteurs étant tracés en dehors de la boîte englobante, il est souhaitable ici de raccourcir un peu les flèches arrivant et partant de ces délimiteurs :

Sous-schéma

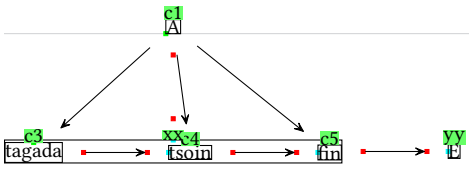
```
%\usetikzlibrary{matrix}
\schemestart
  A\arrow(--[left delimiter={[},
  right delimiter={]}])[,,shorten >=6pt]
  \subscheme{B\arrow[-90,2]C}
  \arrow[,,shorten <=6pt]
  D
\schemestop
```



Les sous-schémas doivent être utilisés à bon escient et peuvent parfois donner des résultats non désirés. Voici un exemple où l'on se sert d'un sous-schéma pour aligner horizontalement 3 composés :

Sous-schéma

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

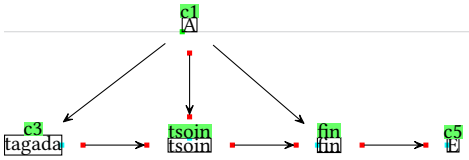


On constate que le centre du sous-schéma est sur la même verticale que le centre du composé « A »puisqu'on a tracé une flèche invisible avec un angle de −90°. Par contre, la flèche entre deux composés existants « A » et « tsoin » n'est *pas* verticale car « tsoin » ne se trouve pas sous le centre du sous-schéma puisque « tagada » est plus large que « fin ». Pour rendre cette flèche verticale, on ne peut pas s'en sortir avec un sous-schéma, sauf à spécifier un angle trouvé en tâtonnant pour l'ancre d'arrivée de la flèche invisible.

Il sera beaucoup plus simple de procéder sans sous-schéma et en utilisant un embranchement : tracer une flèche *visible* entre « A » et « tsoin », puis partir de « tsoin » pour tracer de part et d'autre des flèches horizontales, en créant un embranchement pour les flèches vers la droite.

Sous-schéma

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



6.9 Arguments optionnels des flèches

À l'intérieur de l'argument entre accolades de la commande `\arrow`, le nom des flèches peut être suivi d'arguments optionnels placés entre crochets. Voici ces arguments optionnels et leur signification pour les flèches définies par chem*fig* :

— les flèches « `->` », « `<-` », « `<->` », « `<=>` », « `<<->` »,« `<->>` », « `-/>` » ont trois arguments optionnels :

- le premier contient le « label » qui sera placé au dessus de la flèche ;
- le second contient le « label » qui sera placé sous la flèche. Ces deux labels sont orientés selon le même angle que la flèche. On peut régler le décalage perpendiculaire entre la flèche et l'ancre des labels avec la clé `arrow label sep` = ⟨*dim*⟩ qui vaut 3pt par défaut. Les labels contenus dans les deux arguments optionnels ne *sont pas* composés en mode mathématique.
- le troisième est une dimension qui correspond au décalage dans la direction perpendiculaire au sens de la flèche que l'on veut lui faire subir : cette dimension sera positive pour un décalage de la flèche (et ses éventuels labels) vers le haut et négative si l'on souhaite un décalage vers le bas.

— la flèche « `-U>` » dispose de 5 arguments optionnels :

- les trois premiers sont identiques à ceux des autres types de flèches ;
- le quatrième est le coefficient (qui vaut 0.333 par défaut) qui multiplie la longueur de la flèche pour obtenir le rayon de l'arc de cercle ;
- le cinquième est le demi angle partant du centre de l'arc de cercle tracé, il vaut 60 degrés par défaut.

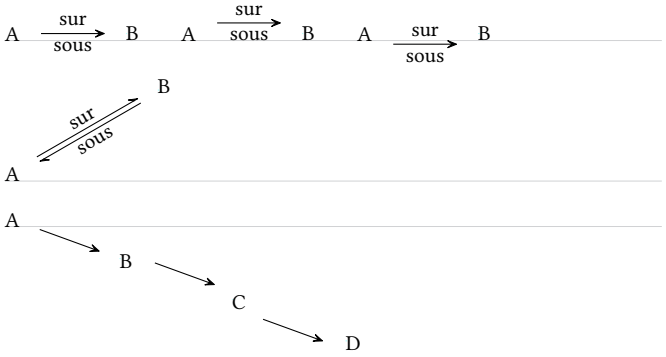
— la flèche invisible « `0` » admet 2 arguments optionnels qui sont du même type que les deux premiers vus au dessus ;

Arguments optionnels des flèches

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

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

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



Un problème survient lorsque les flèches sont verticales :

Flèches verticales

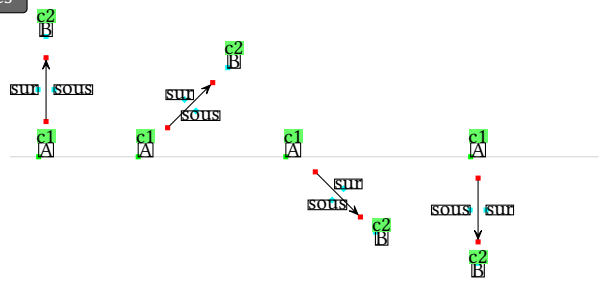
```
\schemestart
  A\arrow{->[sur][sous]}[-90]B
\schemestop
```



Par souci de lisibilité, il serait souhaitable que les labels placés au dessus et au dessous soient horizontaux. On peut choisir l'angle selon lequel les labels sont affichés, la valeur par défaut étant l'angle de la flèche. Pour spécifier un autre angle, il suffit de placer au début des arguments optionnels `*{`⟨*angle*⟩`}` :

Choix des angles

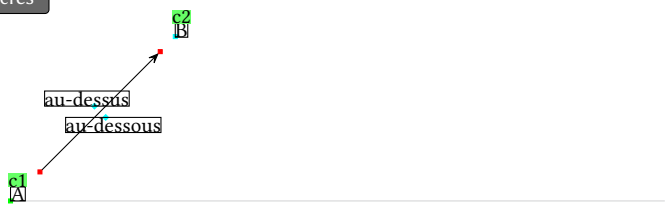
```
\setchemfig{scheme debug=true}
\schemestart A\arrow{->[*{0}sur][*{0}sous]}[90]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}sur][*{0}sous]}[45]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}sur][*{0}sous]}[-45]B\schemestop
\qquad
\schemestart A\arrow{->[*{0}sur][*{0}sous]}[-90]B\schemestop
```



L'ancre par défaut où est attaché chaque label provoque parfois des affichages indésirables :

Ancres

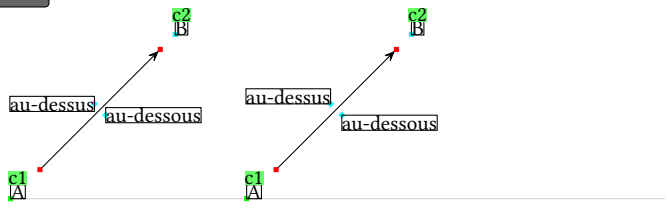
```
\setchemfig{scheme debug=true}
\schemestart
  A\arrow{->[*{0}au-dessus][*{0}au-dessous]}[45,2]B
\schemestop
```



On peut spécifier également l'ancre d'attache pour écraser celui choisi par chem*fig* par défaut. Il suffit d'utiliser la syntaxe suivante : `*{`⟨*angle*⟩`.`⟨*ancre*⟩`}`.

Ancres

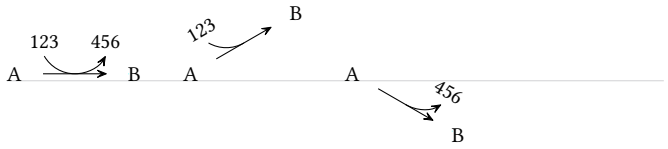
```
\setchemfig{scheme debug=true}
\schemestart
  A\arrow{->[*{0.0}au-dessus][*{0.180}au-dessous]}[45,2]B
\schemestop
\qquad
\schemestart
  A\arrow{->[*{0.south east}au-dessus]%
    [*{0.north west}au-dessous]}[45,2]B
\schemestop
```



Le cas particulier à envisager est la flèche « `-U>` ». Si l'un des deux labels contenus dans les deux premiers arguments optionnels est présent, l'arc de cercle correspondant est tracé :

La flèche -U>

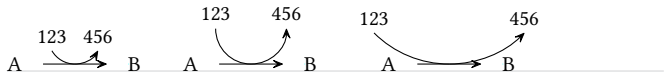
```
\schemestart A\arrow{-U>[123][456]}B\schemestop
\qquad
\schemestart A\arrow{-U>[123]}[30]B\schemestop
\qquad
\schemestart A\arrow{-U>[][456]}[-30]B\schemestop
```



Le quatrième et cinquième argument optionnel changent les dimensions de l'arc de cercle. Ils représentent respectivement le coefficient qui multiplie la longueur de la flèche pour obtenir le rayon et l'angle correspondant au demi arc de cercle (seule la valeur absolue de l'angle est prise en compte) :

La flèche -U>

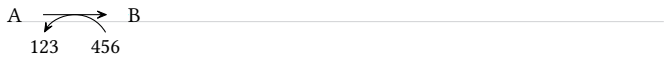
```
\schemestart A\arrow{-U>[123][456][][0.25]}B\schemestop
\qquad
\schemestart A\arrow{-U>[123][456][][][90]}B\schemestop
\qquad
\schemestart A\arrow{-U>[123][456][][1][45]}B\schemestop
```



En mettant un rayon et un angle négatif, l'arc de cercle se situe sous la flèche :

La flèche -U>

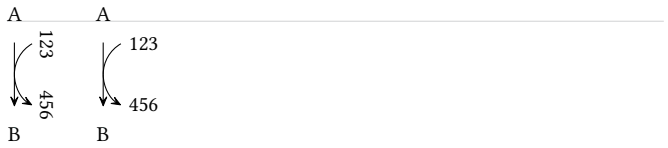
```
\schemestart
  A\arrow{-U>[123][456][][-0.333][-60]}B
\schemestop
```



Les deux premiers arguments disposent des mêmes fonctionnalités concernant les angles d'affichage et les ancres paramétrables que ceux des autres flèches :

La flèche -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 Créer ses propres flèches

Cette section, assez technique et demandant des connaissances sur ti*k*z, n'intéressera que les utilisateurs confirmés ayant besoin de définir leur propres flèches.

La commande `\definearrow` permet de construire des flèches personnalisées. La syntaxe de cette commande est la suivante :

`\definearrow{`⟨*nombre*⟩`}{`⟨*nom flèche*⟩`}{`⟨*code*⟩`}`

où ⟨*nombre*⟩ est le nombre d'arguments optionnels auxquels on fera référence dans le ⟨*code*⟩ avec la syntaxe habituelle `#1`, `#2`, etc. On ne peut pas définir de valeur par défaut pour ces arguments optionnels, et s'ils sont absents lorsque l'utilisateur utilise la macro `\arrow`, ils seront vides.

Avant d'aller plus loin, examinons les macros disponibles en interne lors du tracé des flèches. Comme ces macros ont le caractère « _ » dans leur nom, on ne peut y accéder qu'entre les commandes `\catcode‘\_=11` et `\catcode‘\_=8`.

- `\CF_arrowstartname` et `\CF_arrowendname` contiennent les noms des composés (qui sont des nœuds pour ti*k*z) entre lesquels doit être tracée la flèche ;
- `\CF_arrowstartnode` et `\CF_arrowendnode` contiennent les noms des nœuds où seront placés les extrémités des flèches. À la suite de ces noms viennent s'ajouter, s'ils ne sont pas vides, les ancres spécifiés par l'utilisateur dans l'argument entre parenthèses de la commande `\arrow` ;
- `\CF_arrowcurrentstyle` et `\CF_arrowcurrentangle` contiennent le style et l'angle de la flèche devant être tracée ;
- `\CF_arrowshiftnodes`⟨*dim*⟩ décale les nœuds « `\CF_arrowstartnode` » et « `\CF_arrowendnode` » de la dimension contenue dans son argument et ce, dans la direction perpendiculaire à la flèche ;
- `\CF_arrowdisplaylabel{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}` est la plus complexe et se charge de placer les labels avec les arguments suivants :
  - `#1` et `#5` sont les labels a écrire ;
  - `#2` et `#6` sont des réels compris entre 0 et 1 spécifiant à quel endroit de la flèche ces labels doivent être écrits, 0 étant le début et 1 étant la fin„ le tout étant supposé que la flèche est *rectiligne* ;
  - `#3` et `#7` sont les caractères « + » ou « - », « + » affichant le label au dessus et « - » l'affichant au dessous
  - `#4` et `#8` sont les nom des nœuds correspondant au début et à la fin de la flèche.
- les extrémités de flèches employées sont basées sur le modèle « `CF` » pour une flèche complète et avec l'option « harpoon » pour la demi flèche supérieure.

6.10.1 Une première flèche

Pour créer un exemple, construisons une flèche avec un disque en son milieu que l'on va appeler « `-.>` ». Décidons qu'elle admettra 4 arguments optionnels. Comme pour les flèches déjà définies, le 1er et le 2e seront les labels à mettre au dessus et au dessous ; le 3e sera le décalage de la flèche dans le sens perpendiculaire à sa direction. Enfin, le 4e argument sera la taille du point que l'on prendra égale à 2pt si le 4e argument est absent.

Nous allons d'abord écrire `\definearrow{4}{-.>}` pour déclarer que la flèche aura 4 arguments optionnels et s'appellera « `-.>` ». Tout d'abord, il faut modifier les positions des nœuds entre lesquels doit être tracée la flèche pour tenir compte du décalage contenu dans le 3e argument. Ceci est réalisé par la macro `\CF_arrowshiftnodes`, et il suffit donc de commencer le code de la flèche par : `\CF_arrowshiftnodes{#3}`. Ensuite, il faut tracer la flèche elle même et en profiter pour poser un nœud au milieu du segment que nous appelons « `mid@point` », puis tracer un cercle dont le centre est ce dernier nœud. Tout ceci est fait par le code ti*k*z suivant :

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

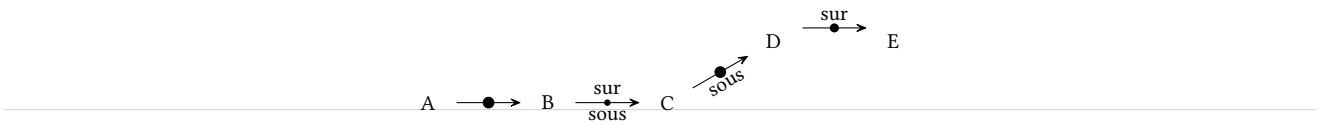
Il ne nous reste plus qu'à placer les labels s'ils existent avec la ligne suivante :

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

Voici notre flèche achevée :

Flèche «-.>»

```
\catcode`\_11
\definearrow4{-.>}{%
  \edef\pt_radius{\ifx\empty#4\empty 2pt\else #4\fi}% rayon du point
  \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{-.>[sur][sous][][1pt]} C \arrow{-.>[][sous]}[30] D \arrow{-.>[sur][][5pt][1.5pt]} E
\schemestop
```



6.10.2 Une flèche courbe

Pourquoi ne pas maintenant créer une flèche courbe ? Pour rendre les choses simples, mettons qu'elle n'aura qu'un seul argument optionnel contenant le code ti*k*z qui définira le (ou les) point de contrôle, et si cet argument est vide, une flèche du type « `-CF` » sera tracée.

Si l'argument `#1` n'est pas vide, ce qui nous intéresse ne sont pas `\CF_arrowstartnode` et `\CF_arrowendnode` qui contiennent les noms des nœuds où seront placés les extrémités des flèches parce que l'emplacement des ces nœuds est déjà déterminés par les ancres qui sont calculés pour des flèches *rectilignes* ! Nous allons plutôt prendre `\CF_arrowstartname` et `\CF_arrowendname` qui contiennent les noms des composés (qui sont des nœuds pour ti*k*z) entre lesquels doit être tracée la flèche. Voici le code ti*k*z qui va tracer la ligne courbe entre ces deux composés :

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

On doit rajouter du code ti*k*z pour raccourcir la flèche de la valeur `\CF_arrowoffset` définie par par la clé `arrow offset =` ⟨*p*⟩uisqu'on ne part pas des nœuds correspondant aux flèches rectilignes `\CF_arrowstartnode` et `\CF_arrowendnode`. Il faudra donc rajouter au début de `\CF_arrowcurrentstyle` le code suivant :

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

c'est ce que font les deux lignes qui suivent le `\else`.

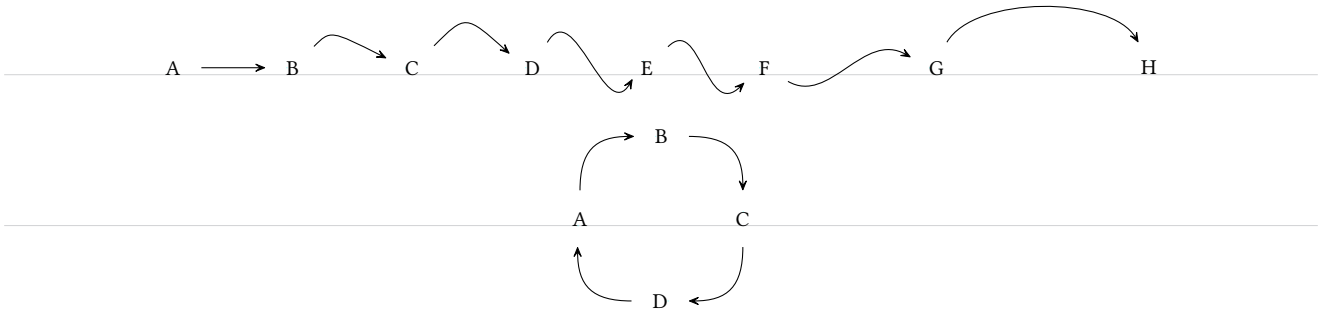
Voici donc notre flèche :

Flèche courbe

```
\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 La commande \merge

La commande `\merge` permet de tracer des flèches qui partent de plusieurs composés existants puis qui se rejoignent pour continuer le schéma réactionnel.

Immédiatement après la commande `\merge`, il faut spécifier dans quelle direction va se faire la progression. Pour cela, on utilise un des 4 caractères de direction : « `>` » (qui est celui par défaut si le caractère est absent), « `<` », « `^` » et « `v` ».

La syntaxe devient ensuite :

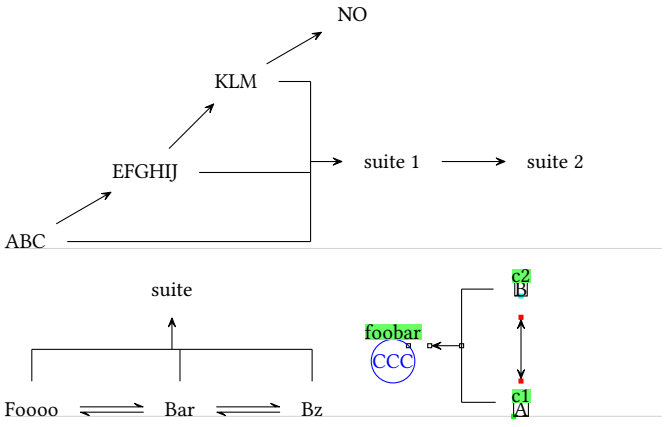
```
\merge{dir}(n1.a1)(n2.a2)(...)(ni.ai)--(n.a[s])
```

Où les noms « `ni` » avant le double tiret sont ceux que des composés déjà existant que l'on veut joindre. On peut également préciser l'ancre « `ai` » duquel doit partir la flèche si l'ancre par défaut ne convient pas. Comme pour la commande `\arrow`, « `n.a[s]` » contient le nom, l'ancre et le style du prochain composé.

La commande \merge

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

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



En ce qui concerne la géométrie de la flèche `\merge`, celle-ci se compose de *n* traits qui partent de chaque composé et vont jusqu'à la ligne de jonction qui leur est perpendiculaire : la longueur par défaut du plus court de ces traits est la moitié de l'espacement entre les composés défini avec `\setcompoundsep`. La flèche tracée depuis la ligne de jonction jusqu'au composé suivant a également cette longueur par défaut, et elle part du milieu de la ligne de jonction. Ces 3 caractéristiques géométriques peuvent être modifiées au moyen de l'argument optionnel qui vient après le nom de composé suivant :

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

où :

— la longueur de trait le plus court entre les composés à joindre et la ligne de jonction s'obtient en multipliant l'espacement entre les composés défini avec `\setcompoundsep` par le coefficient `c1`, qui vaut 0.5 par défaut ;

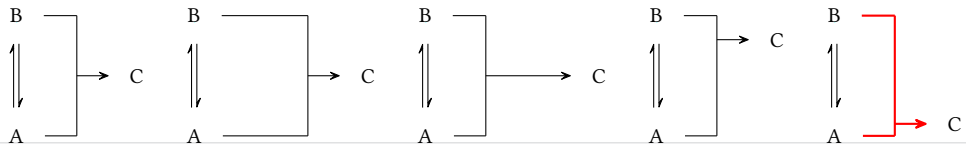
— la longueur de la flèche entre la ligne de jonction et le composé suivant s'obtient en l'espacement entre les composés par le coefficient `c2`, qui vaut 0.5 par défaut ;

— l'endroit d'où part la flèche depuis la ligne de jonction est déterminée par le coefficient `c`, sachant que s'il vaut 0, la flèche partira de la gauche de la ligne de jonction (ou du haut si la direction est `v` ou `^`) ;

— le style de la flèche tracée avec `\merge` est défini par le dernier argument ⟨*style*⟩.

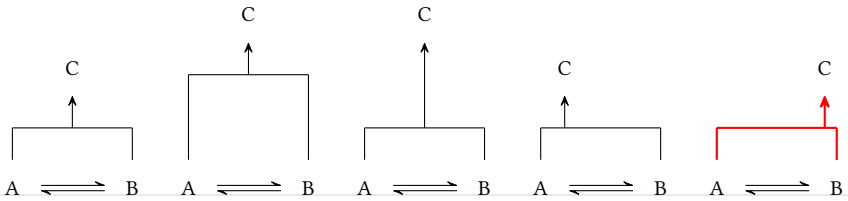
Paramètres géométriques de \merge

```
\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
```



Paramètres géométriques de \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
```



Enfin, il est possible d'écrire des labels au dessus ou au dessous de la flèche créée par la commande `\merge`. Pour cela, le caractère définissant la direction admet deux arguments optionnels entre crochets, le premier contenant le label mis au dessus et le second celui mis au dessous de la flèche. La syntaxe complète de la commande `\merge` est donc :

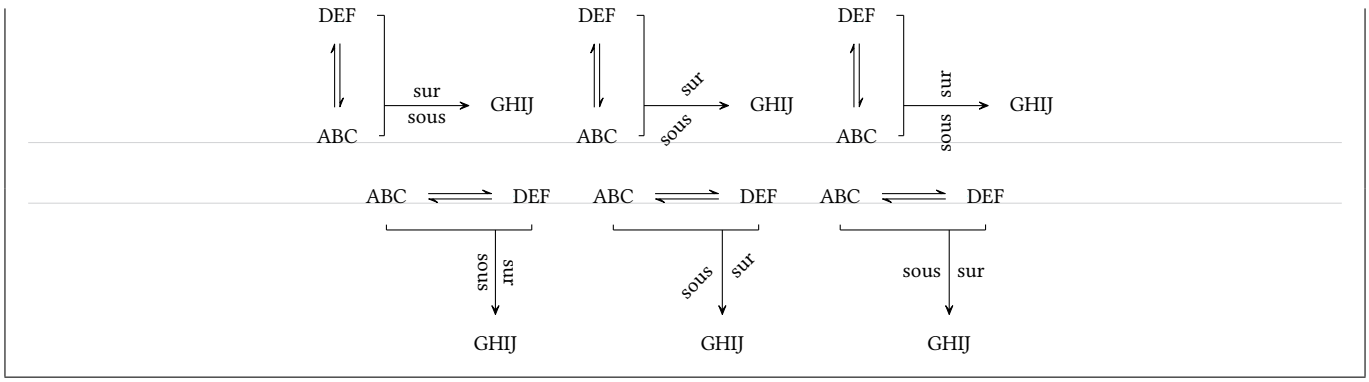
```
\merge{dir}[labelup][labeldow](n1.a1)(n2.a2)(...)(ni.ai)--(n.a[s])[c1,c2,c,style]
```

Toutes les fonctionnalités déjà présentées pour les labels des flèches sont ici possibles, à savoir choisir l'angle de rotation et l'ancre avec la syntaxe `*{angle.ancre}` placée juste avant le contenu du label.

Labels de la commande \merge

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

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



6.12 Le signe +

Entre les commandes `\schemestart` et `\schemestop`, on dispose de la macro « `\+` » qui affiche un signe + . Elle admet un argument optionnel *entre accolades* contenant 3 dimensions sous cette forme {⟨*dim1*⟩,⟨*dim2*⟩,⟨*dim3*⟩} où :

— ⟨*dim1*⟩ et ⟨*dim2*⟩ sont les dimensions qui vont être insérées avant et après le signe + ;

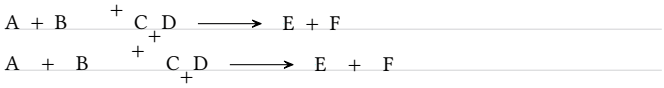
— ⟨*dim3*⟩ est le décalage vertical que l'on veut faire subir au signe.

On peut également spécifier ces dimensions pour tous les signes + à l'aide des clés `+ sep left` = ⟨*dim*⟩, `+ sep right` = ⟨*dim*⟩ et `+ vshift` = ⟨*dim*⟩. Ces dimensions ont pour valeur par défaut 0.5em pour les deux premières et 0pt pour la troisième.

La commande \+

```
\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
```



Comme on le voit sur l'exemple ci-dessous, il faut comprendre que le signe + inséré par la commande `\+` fait partie du composé :

Les composés et \+

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

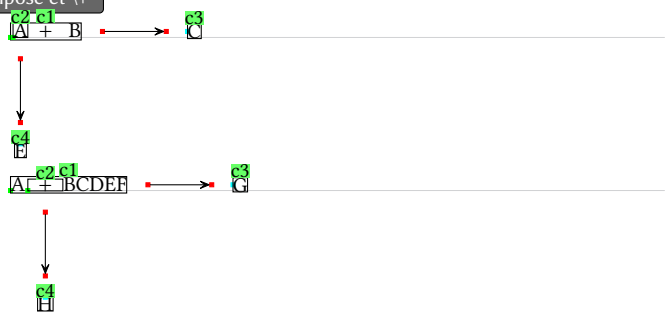


Il devient donc difficile de faire partir une flèche exactement sous la lettre « A » puisque cette lettre n'est pas un composé unique aux yeux de chem*fig*. On peut donc utiliser dans ce cas la commande `\subscheme` pour envelopper dans un composé unique la lettre « A », voire même le signe lui même et pouvoir y faire référence par la suite avec le nom qui lui est attribué :

Sous composé et \+

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

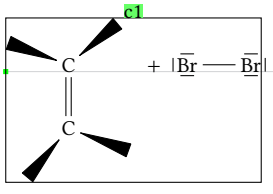
\schemestart
A\subscheme{\+}BCDEF \arrow G
\arrow(@c2--)[-90]H
\schemestop
```



Il peut arriver que l'alignement du signe « + » avec les molécules qui précèdent ou qui suivent pose problème. Voici un exemple :

Alignement du signe +

```
\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
```

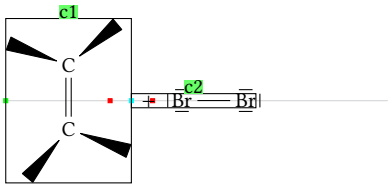


Ce qui se passe est que le signe « + » est sur la même ligne de base que le composé qui précède, et cette ligne de base est celle de l'atome « C » du haut. On pourrait bien sûr décaler le signe « + » mais cela ne changerait pas la position verticale de « $|\overline{Br} - \overline{Br}|$ ». En fait, le signe « + » ne stoppe pas la lecture d'un composé pour chem*fig* ce que l'on constate dans l'exemple ci-dessus où tout est englobé dans le composé « c1 ». On va donc être obligé de stopper le composé après la première molécule avec

un `\arrow{0}[,0]` qui produira une flèche invisible de longueur nulle. Pour centrer verticalement le tout, on va également préciser que l'ancre du premier composé doit être « `west` » (ou « `180` » qui est un synonyme) avec le deuxième argument optionnel de la commande `\schemestart` :

Alignement du signe +

```
\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
```

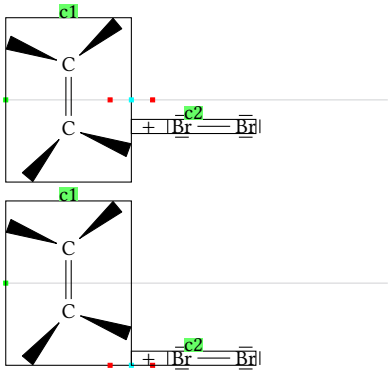


De cette façon, le premier composé « c1 » est la première molécule et le second composé est le reste, c'est-à-dire le signe « + » et la seconde molécule. On aurait pû jouer sur les ancres ou les styles via la commande `\arrow` pour placer le second composé à un autre endroit. Ici, par exemple on décale le second composé de 10pt vers le bas dans le premier cas et on fait coïncider l'ancre « south east » du premier composé avec l'ancre « south west » du second dans le deuxième cas :

Alignement du signe +

```
\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 Réactions horizontales

7.1 Syntaxe

Une réaction horizontale se tient dans un environnement ayant la syntaxe :

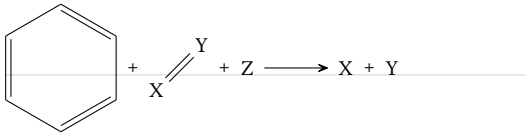
```
\hreac ⟨réaction⟩\endhreac
```

Les tokens `+` `>` `\>` `\^` sont réservés et ont une signification dans la ⟨*réaction*⟩ :

- un ⟨*composé*⟩ s'étend jusqu'à la prochaine occurrence de « + », « > » ou la fin de l'environnement ;
- dans la ⟨*réaction*⟩, tout code appartenant à un ⟨*composé*⟩ se trouvant entre accolades n'est pas examiné et est laissé tel quel entre accolades ;
- les espaces sont ignorés sauf lorsqu'ils se trouvent entre accolades, (voir point précédent) et dans les argument optionnels ou obligatoires des macros `\chemfig` ety `\polymerdelim` ;
- un ⟨*composé*⟩ est constitué d'un code quelconque à l'exception des tokens réservés vus ci-dessus sauf s'ils sont entre accolades ;
- le signe « + » affiche le signe + et constitue un ⟨*composé*⟩ à lui seul ;
- le signe « > » affiche une flèche et constitue un ⟨*composé*⟩ à lui seul ;

Exemple 1

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

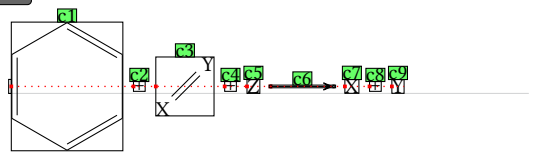


7.2 Placement des composés

Tous les composés sont numérotés `c`⟨*n*⟩ (où ⟨*n*⟩ est un entier) et sont placés dans des boites. Les centres de toutes les boites sont placés sur une même ligne horizontale, qui est tracée en points rouge lorsque l'option `hreac debug` est activée :

Alignement vertical des composés

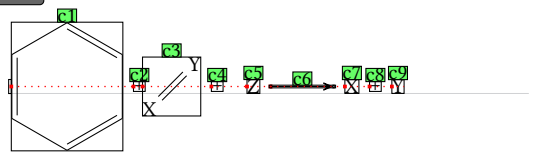
```
\hreac[hreac debug]
    \chemfig{*6(-=-=-=)}
    +
    \chemfig{X=[1]Y}
    +
    Z > X + Y
\endhreac
```



Lorsque `\>{`⟨*dimension*⟩`}` est rencontrée en cours de ⟨*composé*⟩ (mais il est préférable mieux que ce soit au début), celui-ci est déplacé horizontalement de ⟨*dimension*⟩.

Ajustement horizontal des composés

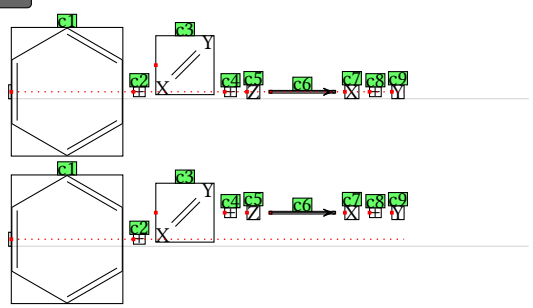
```
\hreac[hreac debug]
    \chemfig{*6(-=-=-=)}
    +
    \>{-5pt}\chemfig{X=[1]Y}
    +
    \>{5pt}Z > X + Y
\endhreac
```



Lorsque `\^{`⟨*dimension*⟩`}` est rencontré en début de ⟨*composé*⟩, il (et lui seul) est déplacé verticalement de `{`⟨*dimension*⟩`}`. Dans la version étoilée `\^*{`⟨*dimension*⟩`}`, le composé et tous les suivants sont concernés.

Ajustement vertical des composés

```
\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 Nom des composés

Pour afficher sous un ⟨*composé*⟩ son ⟨*nom*⟩, on peut utiliser indifféremment les syntaxes :

— `\chemname[`⟨*dimension*⟩`]{`⟨*composé*⟩`}{`⟨*nom*⟩`}` ;

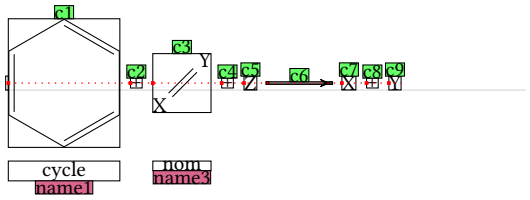
— ⟨*composé*⟩`\name[`⟨*dimension*⟩`]{`⟨*nom*⟩`}` ou `\name[`⟨*dimension*⟩`]{`⟨*nom*⟩`}`⟨*composé*⟩, l'ordre d'apparition n'ayant pas d'importance

La ⟨*dimension*⟩ optionnelle étant la distance minimale entre la boite du ⟨*composé*⟩ et celle du ⟨*nom*⟩. Lorsqu'elle est omise, sa valeur par défaut est la valeur de la clé `name sep`.

Tous les noms sont numérotés `name`⟨*n*⟩ où ⟨*n*⟩ est l'entier correspondant au ⟨*composé*⟩. Ils sont placés dans des boites dont la largeur est celle du ⟨*composé*⟩. Les frontières supérieures de ces boites sur une même ligne horizontale dont la position verticale est calculée pour se trouver sous le composé ayant la plus grande profondeur.

Noms des composés

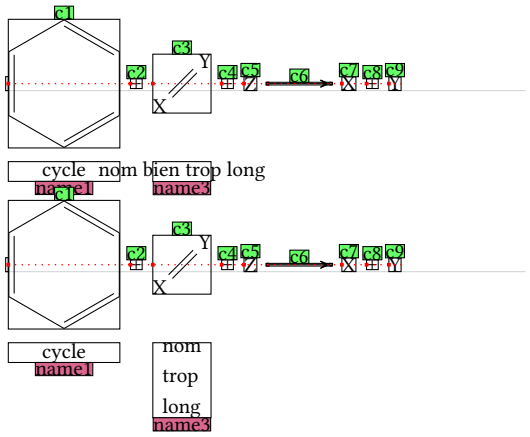
```
\hreac[hreac debug]
    \chemfig{*6(-=-=-=)}\name{cycle}
    +
    \chemfig{X=[1]Y}\name{nom}
    +
    Z > X + Y
\endhreac
```



Si la largeur d'un ⟨*nom*⟩ excède celle de son ⟨*composé*⟩, aucun ajustement de la largeur du ⟨*composé*⟩ n'est faite et le ⟨*nom*⟩ dépassera les frontières de la boite qui le contient. Il appartient à l'utilisateur d'utiliser la macro `\\` pour afficher le ⟨*nom*⟩ sur plusieurs lignes.

Noms des composés

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



7.4 Types de flèches

Si le token « > » est immédiatement suivi d'un texte entre accolades, ce texte optionnel spécifie le type de flèche. Sa valeur par défaut est la valeur de la clé `arrow head` qui contient ⟨*-CF*⟩ par défaut.

La syntaxe complète du token > est donc :

```
>{⟨type de flèche⟩}[⟨label haut⟩][⟨label bas⟩]
```

En utilisant ce paramètre optionnel entre accolades, on accède à ces autres types de flèches, déjà décrits dans la partie « Schémas réactionnels » : « -> », « <- », « <-> », « <=> », « <<-> » et « <->> ».

Les 6 types de flèches

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

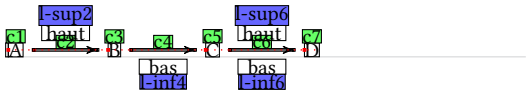


7.5 Label des flèches

Si le token « > » est immédiatement suivi d'un argument optionnel [⟨*label*⟩], ce ⟨*label*⟩ sera placé au-dessus de la flèche. Si un autre argument optionnel [⟨*label*⟩] est présent, l'autre ⟨*label*⟩ sera placé sous la flèche.

Labels de flèche

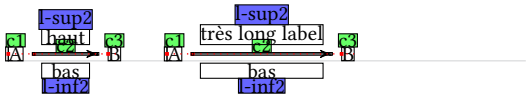
```
\hreac[hreac debug]
    A >[haut] B >[][bas] C >[haut][bas] D
\endhreac
```



La largeur par défaut d'une flèche est `3em`, mais si un label dépasse cette dimension, la flèche s'allonge.

Extension de flèche

```
\hreac[hreac debug]
    A >[haut][bas] B
\endhreac
\qquad
\hreac[hreac debug]
    A >[très long label][bas] B
\endhreac
```



Comme pour les noms de composés, il est possible d'utiliser `\\` pour composer un ⟨*label*⟩ sur plusieurs lignes.

Label sur plusieurs lignes

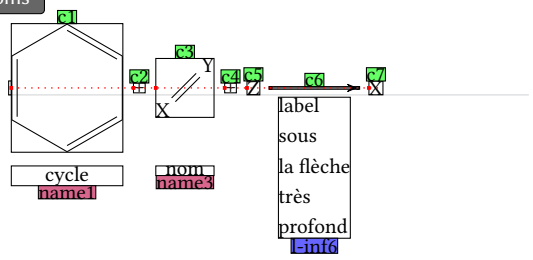
```
\hreac[hreac debug]
    A >[très\\long\\label][bas] B
\endhreac
```



La boite contenant le label du bas n'a aucune influence sur la position verticale des noms des composés.

Indépendance des labels et des noms

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



8 Liste des commandes

les macros créées par chem*fig* sont :

Macros	Description
`\chemfig[⟨paramètres⟩]{⟨code⟩}`	dessine la molécule dont le dessin est décrit par le ⟨*code*⟩
`\setchemfig{⟨paramètres⟩}`	règle les paramètres selon la syntaxe `clé` = ⟨*valeur*⟩, cf. p 4
`\resetchemfig`	Restaure les paramètres à leurs valeurs par défaut
`\printatom`	cette macro affiche les atomes dans les molécules, cf. p 23
`\hflipnext`	la prochaine molécule sera inversée horizontalement, cf. p 35
`\vflipnext`	la prochaine molécule sera inversée verticalement, cf. p 35
`\definesubmol {⟨nom⟩}⟨n⟩[⟨code1⟩]{⟨code2⟩}`	créé un alias ⟨*nom*⟩ que l'on peut placer dans le code des molécules à dessiner qui remplace le ⟨*code1*⟩ ou le ⟨*code2*⟩ selon l'inclinaison de la dernière liaison, cf. p 28
`\chemskipalign`	Ignore le groupe d'atomes en cours pour le mécanisme d'alignement vertical, cf. p 26
`\redefinesubmol {⟨nom⟩}⟨n⟩[⟨code1⟩]{⟨code2⟩}`	remplace l'alias déjà existant ⟨*nom*⟩ par le nouveau ⟨*code*⟩, cf. p 28
`\charge [⟨paramètres⟩]⟨pos⟩ [⟨tikz⟩]{⟨atome⟩}`	affiche l'⟨*atome*⟩ et positionne les charges selon leurs ⟨*positions*⟩ sans être prises en compte dans la boite englobante, cf. p 29
`\Charge [⟨paramètres⟩]⟨pos⟩ [⟨tikz⟩]{⟨atome⟩}`	identique à `\charge`, mais les charges sont comptabilisées dans la boite englobante.
`\chemmove[⟨options tikz⟩]⟨code tikz⟩`	ouvre un environnement `\tikzpicture` en y ajoutant à celles qui existent déjà les ⟨*options tikz*⟩, et relie avec le ⟨*code tikz*⟩ les nœuds posés dans les molécules à l'aide du caractère « `@` », cf. p 18
`\chemabove[⟨dim⟩]{⟨txt1⟩}{⟨txt2⟩}`	écrit le ⟨*txt1*⟩ et positionne le ⟨*txt2*⟩ au-dessus en laissant ⟨*dim*⟩ d'espacement vertical. Cette commande ne change pas la boîte englobante de ⟨*txt1*⟩, cf. p 31
`\chembelow[⟨dim⟩]{⟨txt1⟩}{⟨txt2⟩}`	écrit le ⟨*txt1*⟩ et positionne le ⟨*txt2*⟩ au-dessous en laissant ⟨*dim*⟩ d'espacement vertical. Cette commande ne change pas la boîte englobante de ⟨*txt1*⟩, cf. p 31
`\Chemabove[⟨dim⟩]{⟨txt1⟩}{⟨txt2⟩}`	écrit le ⟨*txt1*⟩ et positionne le ⟨*txt2*⟩ au-dessus en laissant ⟨*dim*⟩ d'espacement vertical, cf. p 31
`\Chembelow[⟨dim⟩]{⟨txt1⟩}{⟨txt2⟩}`	écrit le ⟨*txt1*⟩ et positionne le ⟨*txt2*⟩ au-dessous en laissant ⟨*dim*⟩ d'espacement vertical, cf. p 31
`\chemname[⟨dim⟩]{⟨molécule⟩}{⟨nom⟩}`	affiche le ⟨*nom*⟩ sous la ⟨*molécule*⟩, cf. p 20
`\chemnameinit`	initialise la plus grande profondeur des molécules rencontrées pour avoir un alignement correct de leurs noms
`\schemestart...\schemestop`	balises entre lesquelles un schéma réactionnel est tracé, cf. p 44
`\arrow`	trace une flèche dans un schéma réactionnel (la commande n'est définie que dans un schéma réactionnel)
`\+`	affiche un signe + dans un schéma réactionnel (la commande n'est définie que dans un schéma réactionnel), cf. p 45
`\subscheme{⟨code⟩}`	trace un sous-schéma réactionnel (la commande n'est définie que dans un schéma réactionnel), cf. p 49
`\definearrow`	définit une flèche pour un schéma réactionnel, cf. p 53
`\chemleft⟨car1⟩⟨matériel⟩ ...\chemright⟨car1⟩`	trace des délimiteurs extensibles définis par ⟨*car1*⟩ et ⟨*car2*⟩ à gauche et à droite du ⟨*matériel*⟩, `\cfpage{chemleft}`

`\chemup⟨car1⟩⟨matériel⟩ ...\chemdown⟨car1⟩`	trace des délimiteurs extensibles définis par ⟨*car1*⟩ et ⟨*car2*⟩ au-dessus et au-dessous du ⟨*matériel*⟩, cf. p 50
`\polymerdelim [⟨paramètres⟩]{⟨nœud1⟩}{⟨nœud2⟩}`	trace des délimiteurs aux nœuds spécifiés, cfpagepolymerdelim
`\hreac...\endhreac`	balises entre lesquelles une réaction horizontale est tracée, cf. p 59
`>{⟨type⟩}[⟨label haut⟩][⟨label bas⟩]`	trace une flèche horizontale, cf. p 60
`+`	affiche le signe + dans la réaction horizontale
`>{⟨dimension⟩}`	déplace le composé courant horizontalement, cf. p 59
`\^{⟨dimension⟩}` ou `\^*{⟨dimension⟩}`	déplace le composé courant verticalement, cf. p 59
`\name{⟨nom⟩}`	met le ⟨*nom*⟩ sous le composé courant, cf. p 60

9 Exemples

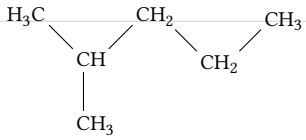
Ce manuel s'achève avec des dessins de molécules plus ou moins complexes.

L'utilisateur curieux pourra s'intéresser au ⟨*code*⟩ de chaque molécule, bien que celui-ci devienne parfois rebutant lorsqu'elles deviennent complexes. En effet, au delà d'un certain niveau de complexité, bien qu'il soit est assez facile d'écrire un ⟨*code*⟩ pour dessiner une molécule, il est assez ardu de le relire pour l'analyser a posteriori. On atteint rapidement les limites de la lisibilité immédiate du code d'un dessin complexe.

Quoi qu'il en soit, j'espère que cette extension aidera tous les utilisateurs de TEX, voire même ceux de LATEX qui souhaitent dessiner des molécules chimiques. Bien que chem*fig* ait été testé de façon approfondie, j'espère que vous serez indulgent quant aux bugs rencontrés. Un email pour me signaler un dysfonctionnement ou toute proposition d'amélioration sera bienvenu.

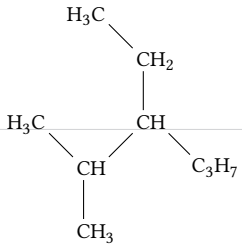
2-methylpentane

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



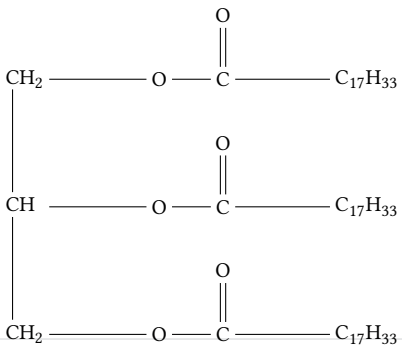
3-éthyl-2-méthylhexane

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



Stéarine, formule semi développée

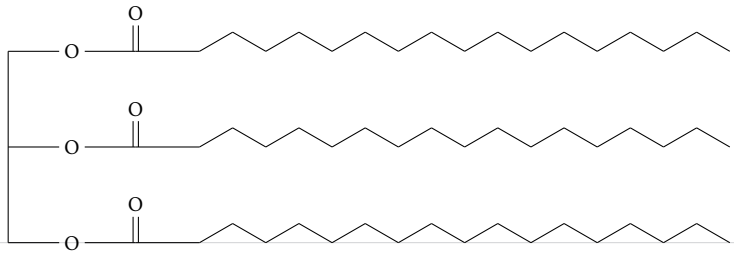
```
\definesubmol{@}{([0,2]-O-[0,1]C(=[2,1]O)-C_{17}H_{33})}
\chemfig{[2,2]CH_2!@-CH_{\phantom 2}!@-CH_2!@}
```



Stéarine, formule topologique

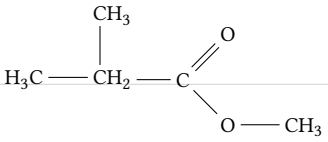
```
\definesubmol{x}{-[:+30,.6]-[:-30,.6]}
\definesubmol{y}{-O-(=[2,.6]O)-!x!x!x!x!x!x!x}
```

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



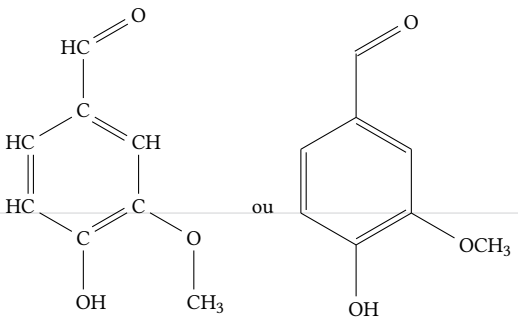
2-méthylpropanoate de méthyl

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



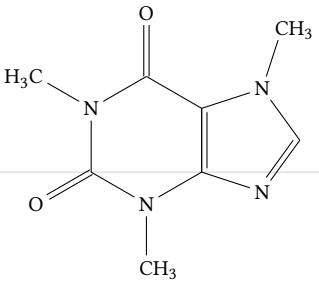
La vanilline

```
\chemfig{HC*6(-C(-OH)=C(-O-[::-60]CH_3)-CH=C(-[,,,2]HC=[::-60]O)-HC=[,,2])} \quad ou \quad
\chemfig{*6(-(-OH)=(-OCH_3)-=(-=[::-60]O)-=)}
```



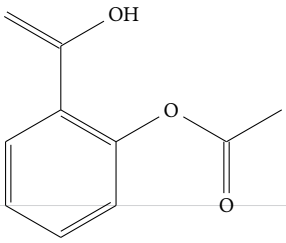
La caféine

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



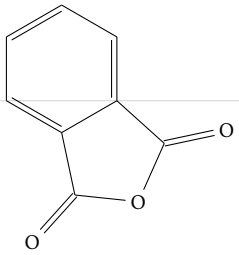
L'aspirine

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



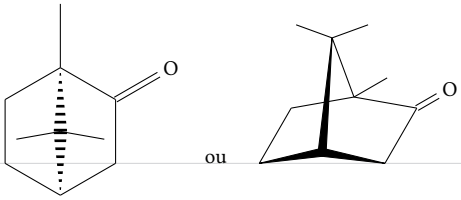
Anhydride phtalique

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



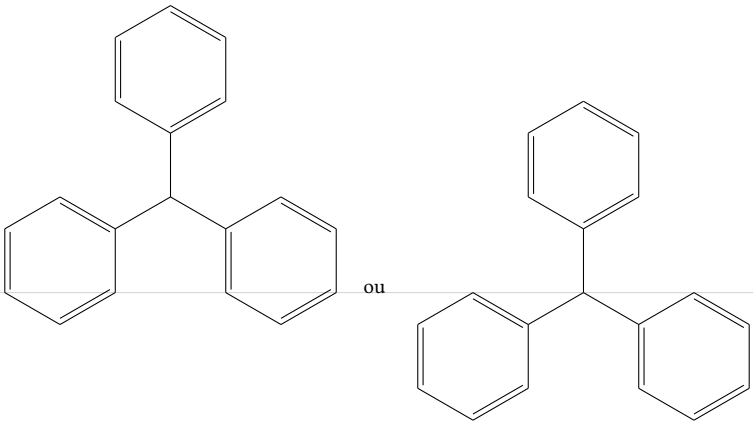
Camphre

```
\chemfig{*6(-(<:[::120](-[::-100,0.7])(-[::100,0.7]))--(=O)-(-)(<:[::120])--)}
\quad ou \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]}
```



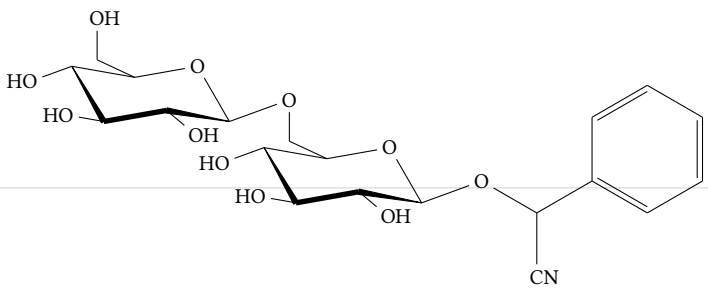
Triphenylméthane

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



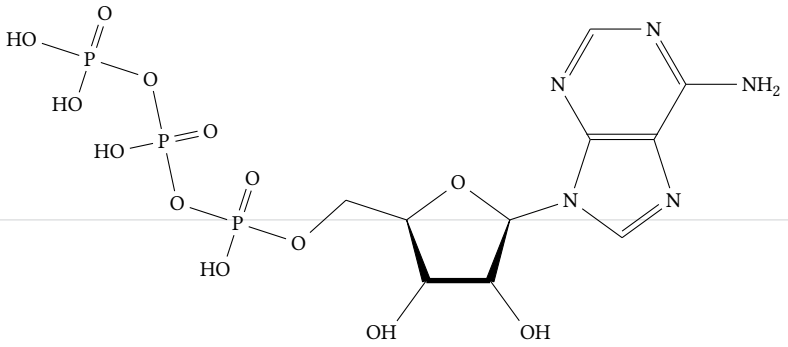
Amygdaline

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



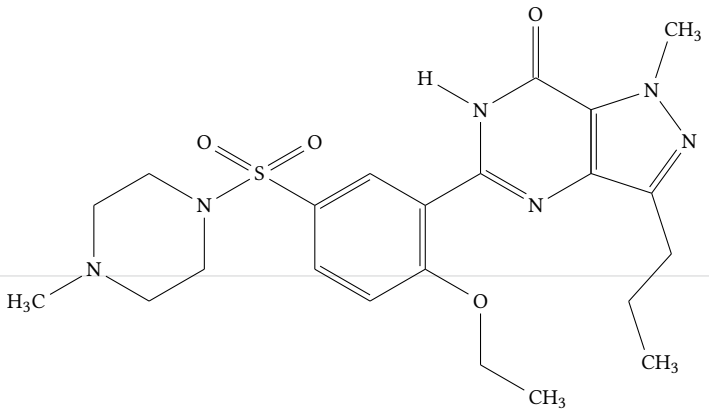
Adénosine triphosphate

```
\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-)}
```



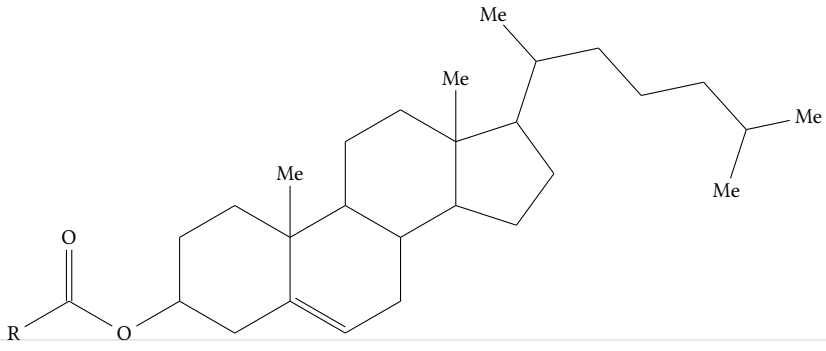
Viagra

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



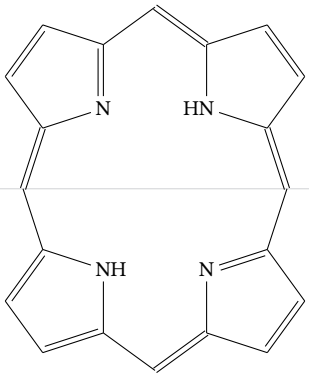
Ester de cholestérol

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



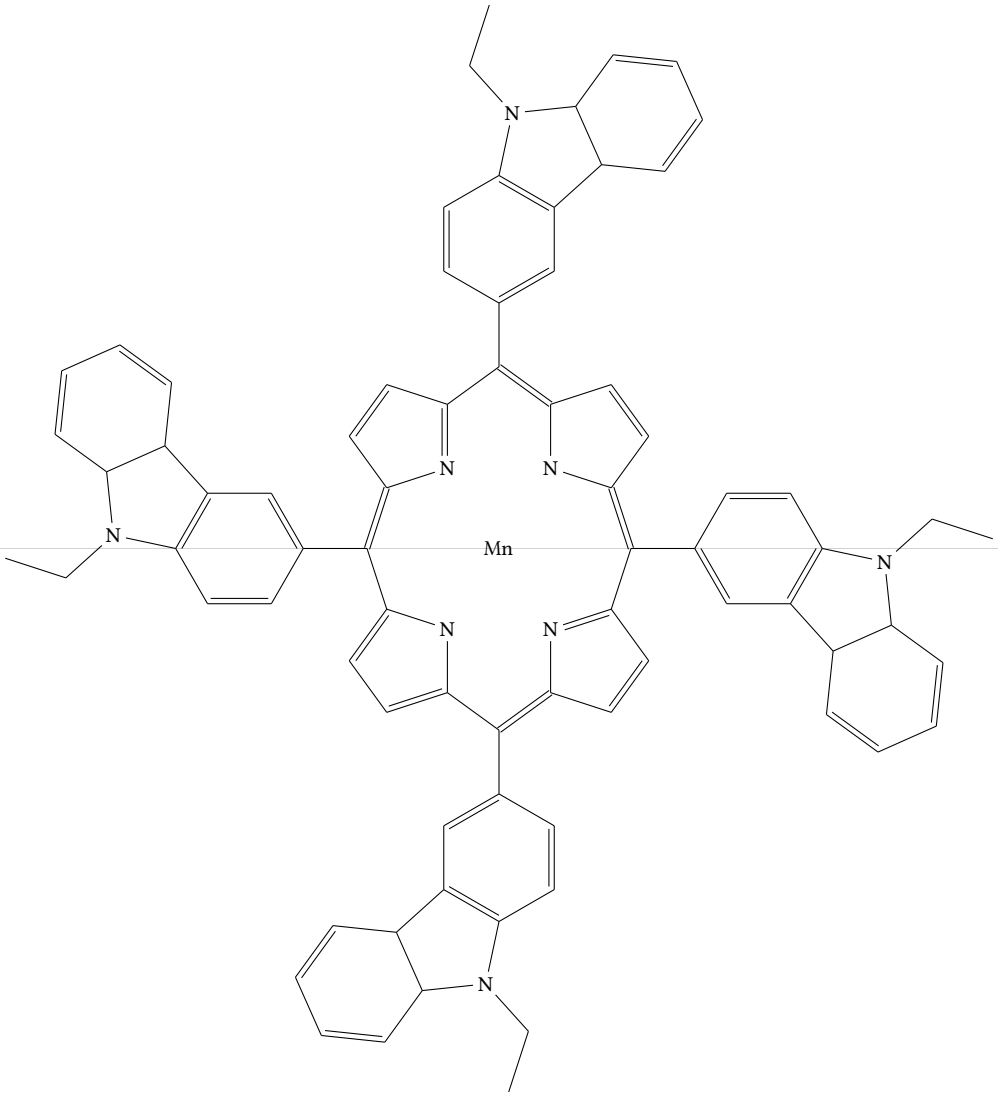
Porphyrine

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



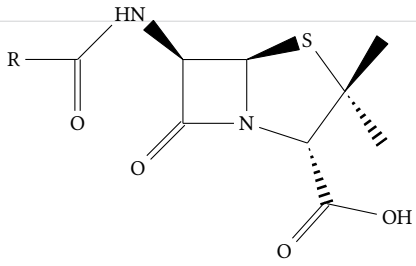
Manganese 5,10,15,20-tétra(N-ethyl-3-carbazolyl) porphyrine

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



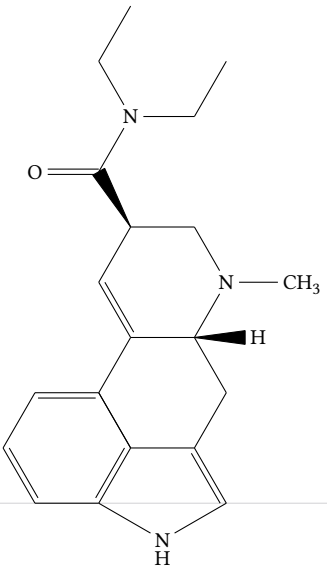
Pénicilline

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



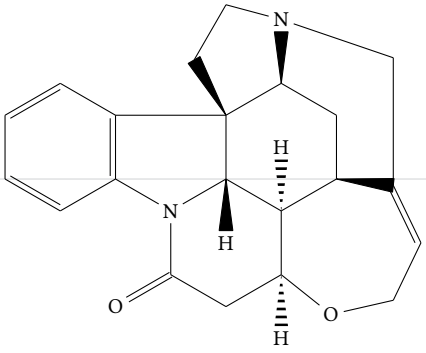
LSD

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



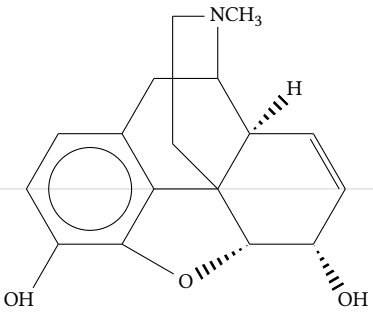
Strychnine

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



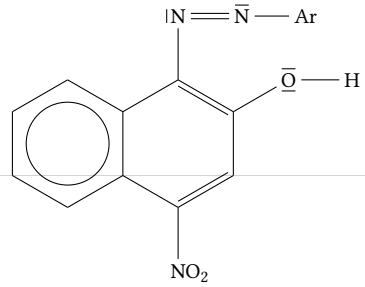
Codéine

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



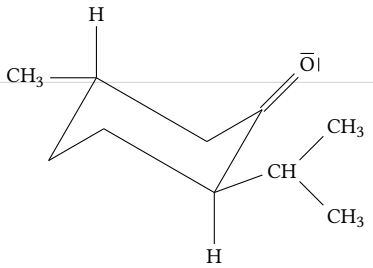
Colorant(rouge)

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



Menthone

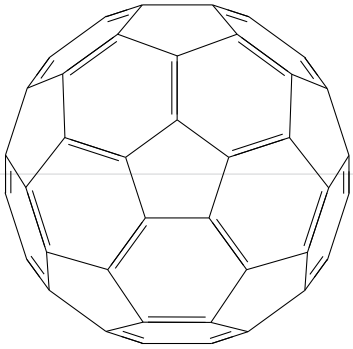
```
\chemfig{CH_3-?(-[2]H)(-[::-30,2]-[::+60](=[1]\charge{0=\|,90=\|}{O})
-[::-150,1.5](-[:20]CH(-[1]CH_3)(-[7]CH_3))(-[6]H)-[::-90,2]-[::+60]?)}
```



Fullerène

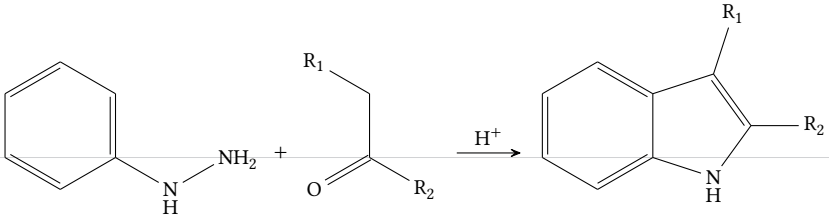
```
\definesubmol\fragment1{
    (-[:#1,0.85,,,draw=none]
    -[::126]-[::-54](=_#(2pt,2pt)[::180])
    -[::-70](-[::-56.2,1.07]=^#(2pt,2pt)[::180,1.07])
    -[::110,0.6](-[::-148,0.60](=^[::180,0.35])-[::-18,1.1])
    -[::50,1.1](-[::18,0.60]=_[::180,0.35])
    -[::50,0.6]
    -[::110])
}
\chemfig{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

```
}
```



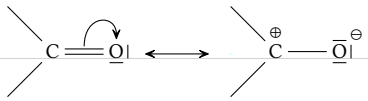
Synthèse de Fischer de l'indole

```
\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
```



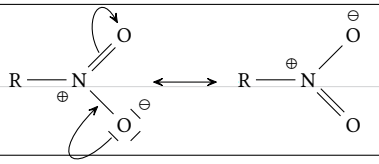
Mécanisme réactionnel : groupement carbonyle

```
\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);}
```



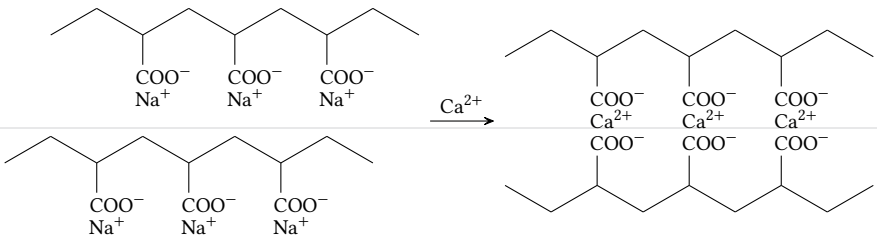
Mécanismes réactionnels : dérivés nitrés

```
\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
```



Addition nucléophile. Amines primaires

```
\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 élimination]}\chemfig{R-N=C(-[1]CH_3)(-[7]CH_3)}
    \+
    \chemfig{H_3\chemabove{O}{\scriptstyle\oplus}}
\schemestop
```

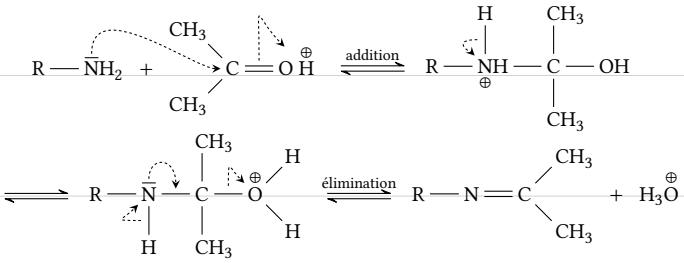
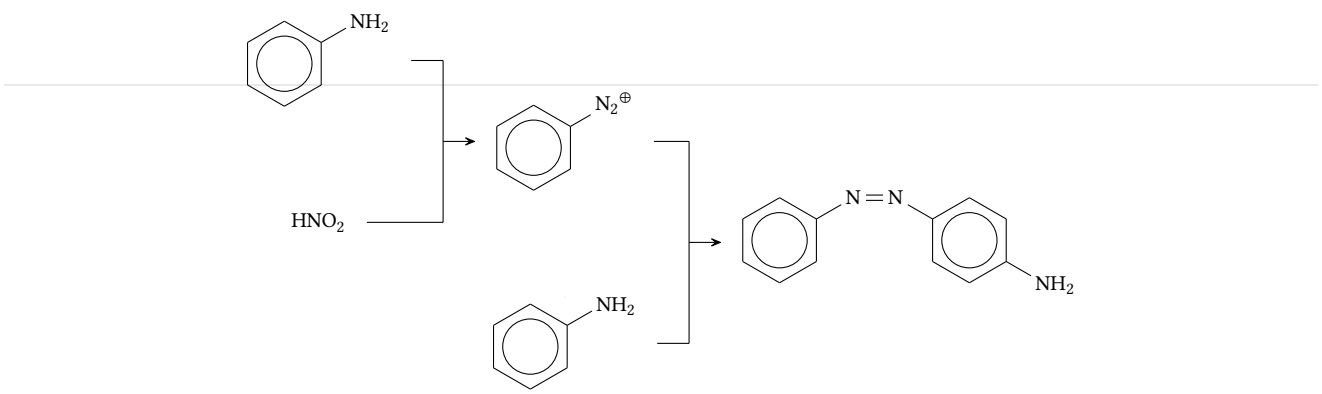


Schéma réactionnel

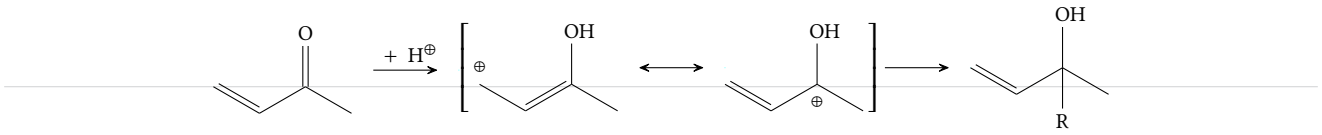
```
\setchemfig{atom sep=2em}
\schemestart[arrow angle=-90]
    \chemfig{**6(---(-NH _2)---)}\arrow{0}\chemfig{HNO_2}
    \merge(c1)(c2)--()
    \chemfig{**6(---(-N_2|{}^\oplus)---)}\arrow{0}\chemfig{**6(---(-NH _2)---)}
    \merge(c3)(c4)--()
    \chemfig{**6(---(-N=[::-30]N-[::-30]**6(---(-NH_2)---))---)}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

```
\schemestop
```



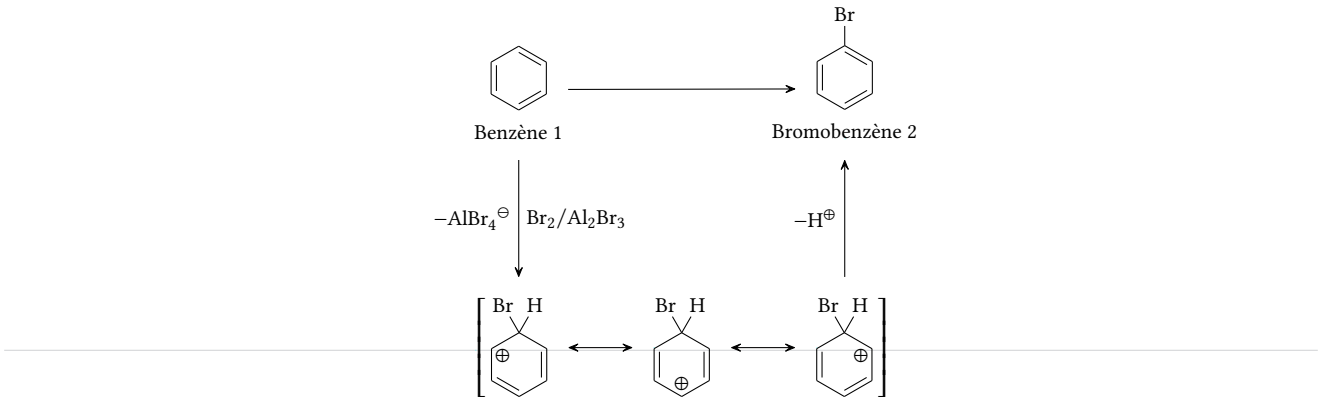
Réaction d'addition

```
\setchemfig{atom sep=2.5em}
\schemestart
    \chemfig{*6(=-(-)(=[2]O))}
    \arrow{->[\+\chemfig{H^\oplus}]}
    \chemleft[%
    \subscheme[90]{%
        \chemfig{*6((-[2,0.33,,,draw=none]\scriptstyle\oplus)-=(-)-OH)}
        \arrow{<->}
        \chemfig{*6(=-(-)(-[6,0.33,,,draw=none]\scriptstyle\oplus)-OH)}}%
    \chemright]
    \arrow(@c4--)\chemfig{*6(=-(-)(-[6]R)-OH)}
\schemestop
```



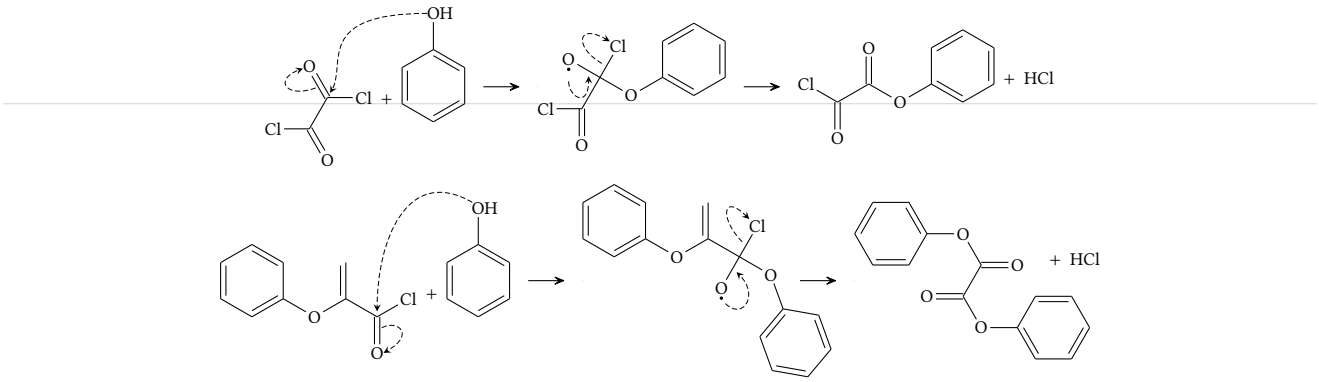
Substitution aromatique éléctrophile

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



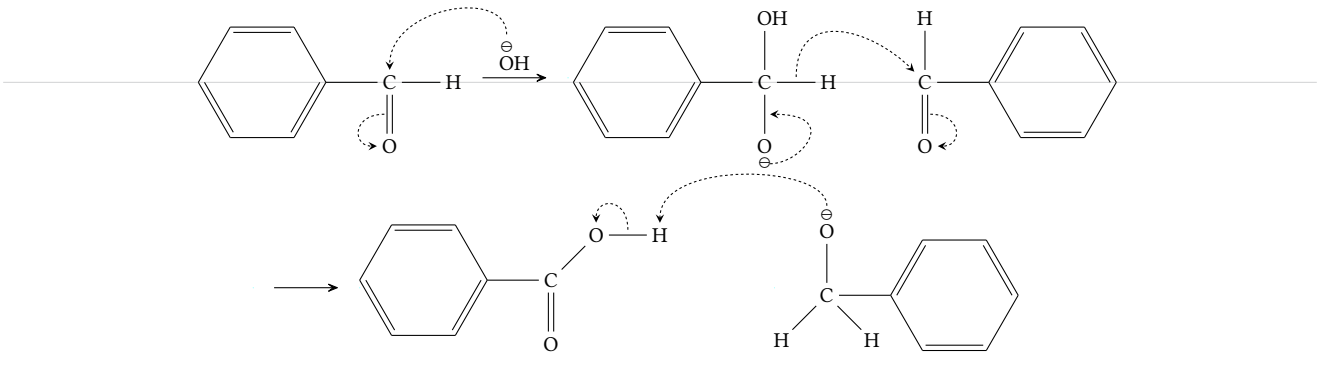
Mécanismes réactionnels de la chloration

```
\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
```



Réarrangement de Beckmann

```
\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
```

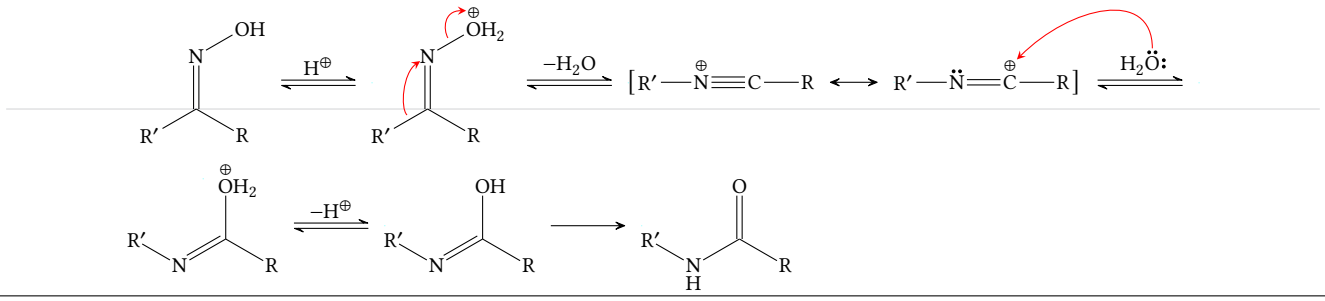
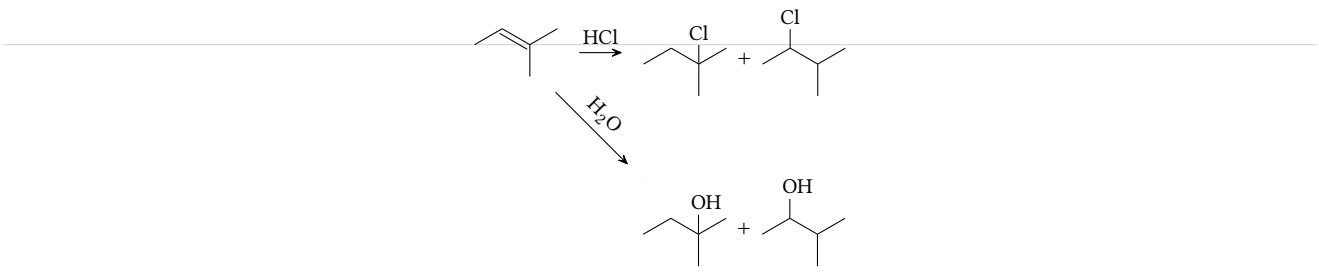


Schéma réactionnel

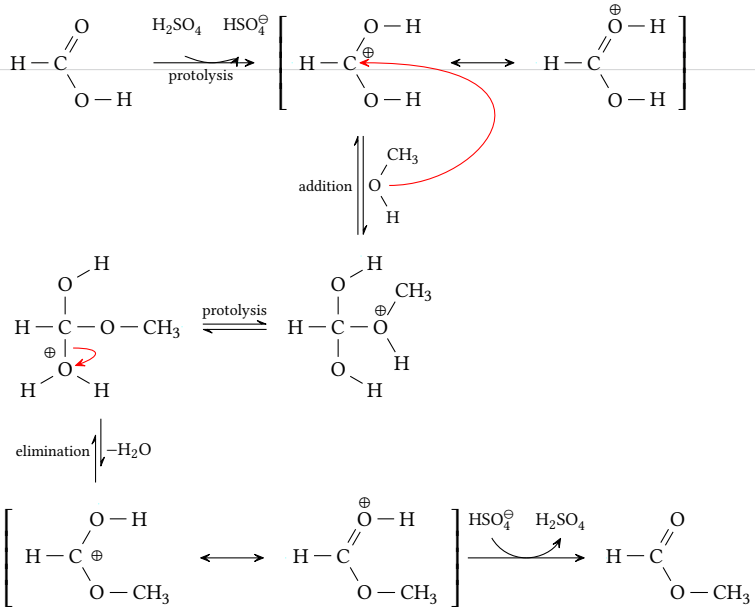
```
\setchemfig{atom sep=1.5em,compound sep=4em}
\schemestart
    \chemfig{-[::30]=_[::-60](-[:: -60])-[::60]}
    \arrow{->[\chemfig{HCl}]}
    \chemfig{-[::30]-[::-60](-[::120]Cl)(-[::-60])-[::60]}\+\chemfig{-[::30](-[::60]Cl)-[::-60](-[::-60])-[::60]}
    \arrow(@c1--.north west){->[\chemfig{H_2O}]}[-45,1.7]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



Estérification de l'acide formique

```
\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}
```

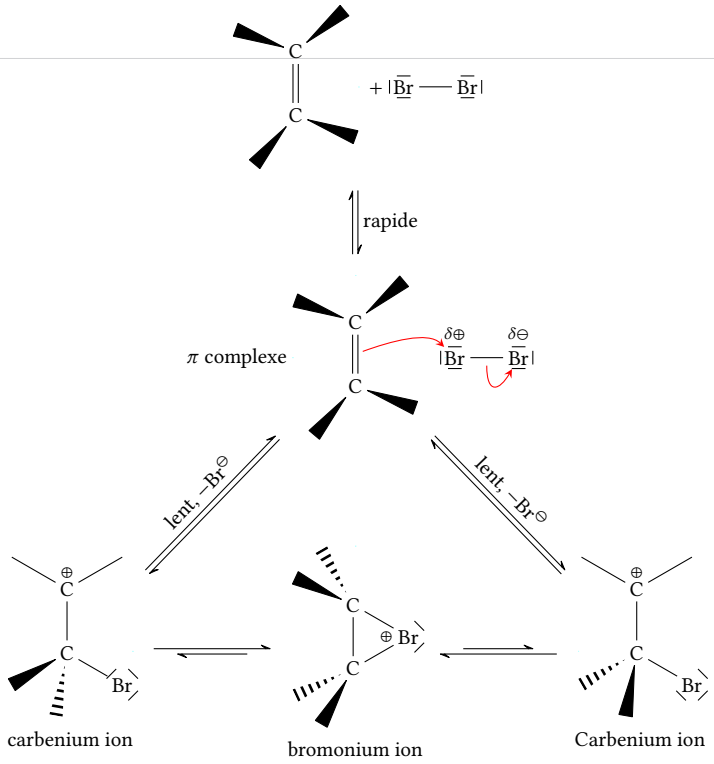
```
    \chemmove[red,shorten <=3pt,shorten >=1pt]{
        \draw(a1)..controls +(0:1.5cm)and+(0:3cm).. (a2);
        \draw(b1)..controls +(0:5mm)and+(20:5mm)..(a3);}
\schemestop
```



Addition électrophile d'halogène sur l'oléfine

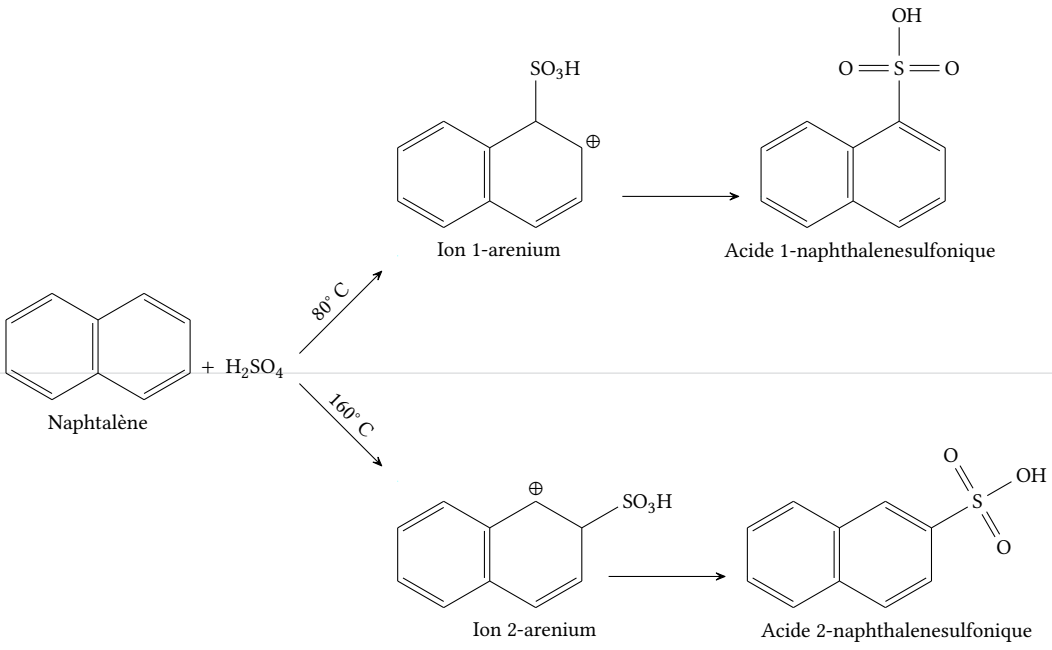
```
\schemestart
    \subscheme{%
        \chemfig{C(<[:40])(<[:160])=[6]C(<[:-130])<[:-20]}
        \arrow{0}[,0]\+\chemfig{\charge{90=\|,180=\|,270=\|}{Br}-\charge{90=\|,0=\|,270=\|}{Br}}}
    \arrow(@c1--olefin){<=>[*{0}rapide]}[-90]
    \chemfig{>[:-20]C(<[:40])=[@{db}6]C(<[:-130])<[:-20]}
    \arrow(--bromonium){0}[-90]
    \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){<=>[lent, \chemfig{{-}Br^\ominus}]}
    \arrow(@carbeniumB--@olefin){<=>[lent, \chemfig{{-}Br^\ominus}]}
    \chemmove[-stealth,red,shorten <=3pt,shorten >=2pt]{
        \draw(db) .. controls +(20:5mm) and +(135:5mm) .. (Br1);
        \draw(b2) .. controls +(-90:5mm) and +(-120:5mm) .. (Br2);}
\schemestop
```

```
\chemnameinit{}
```



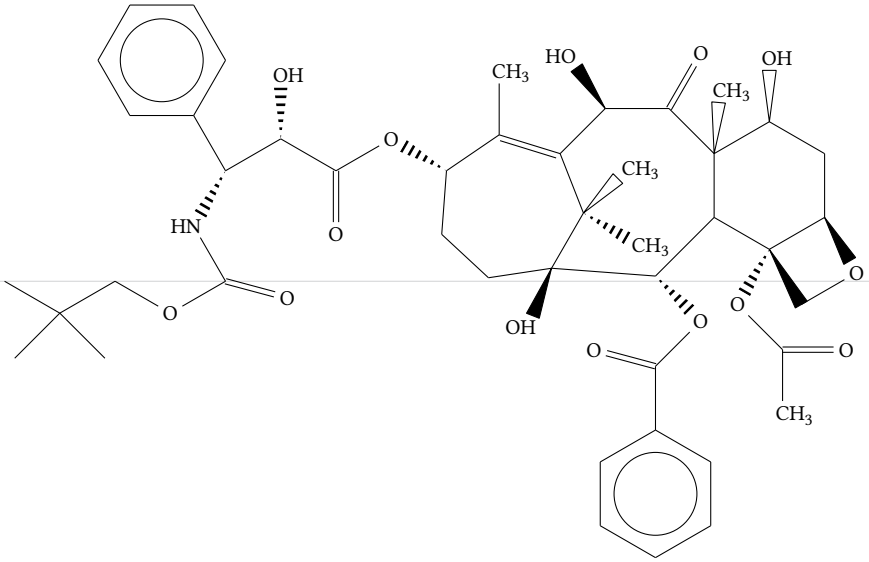
Sulfonation de la naphtalène

```
\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(-=-=-)=-=-)}}{Naphtalène}\+\chemfig{H_2SO_4}
    \arrow(nph.mid east--.south west){->[80° C]}[45]
    \chemname{\chemfig{*6(=-*6(-=-(!\cycleoplus)-(-SO_3H)-)=-=-)}}{Ion 1-arenium}
    \arrow(.mid east--.mid west)
    \chemname{\chemfig{*6(=-*6(-=-=(-!{so2oh})-)=-=-)}}{Acide 1-naphthalenesulfonique}
    \arrow(@nph.mid east--.north west){->[160° C]}[-45]
    \chemname{\chemfig{*6(=-*6(-=-(-SO_3H)-(!\cycleoplus)-)=-=-)}}{Ion 2-arenium}\kern-4em
    \arrow(.mid east--.mid west)
    \chemname{\chemfig{*6(=-*6(-=-(-!{so2oh})=-)=-=-)}}{Acide 2-naphthalenesulfonique}
\schemestop
\chemnameinit{}
```



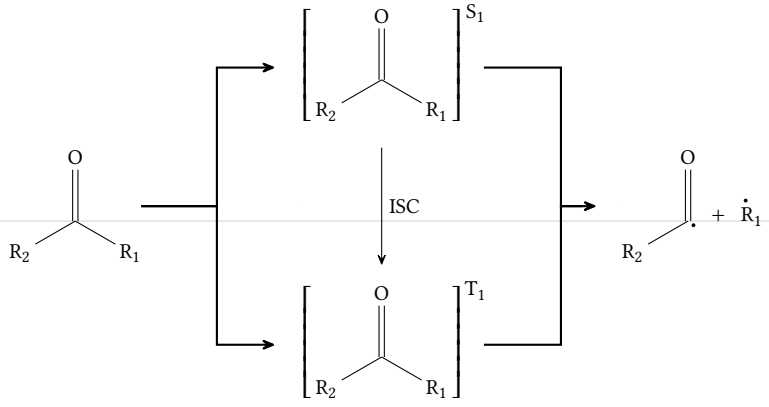
Taxotère

```
\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)-)}
```



Flèches divergentes

```
\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 Historique des changements

Changements (plus récents en premier) faits dans chem*fig* depuis la première version.

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