Name

chem – embed chemical structure diagrams in *groff* documents

Synopsis

chem [**−−**] [*file* . . .]

chem −h
chem −−help

chem −v
chem −−version

Description

chem produces chemical structure diagrams. Today's version is best suited for organic chemistry (bonds, rings). The *chem* program is a *groff* preprocessor like *eqn*, *pic*, *tbl*, etc. It generates *pic* output such that all *chem* parts are translated into diagrams of the *pic* language.

If no operands are given, or if *file* is "**−**", *chem* reads the standard input stream. **−h** and **−−help** display a usage message, whereas **−v** and **−−version** display version information; all exit.

The program *chem* originates from the Perl source file *chem.pl*. It tells *pic* to include a copy of the macro file *chem.pic*. Moreover the *groff* source file *pic.tmac* is loaded.

In a style reminiscent of *eqn* and *pic*, the *chem* diagrams are written in a special language.

A set of *chem* lines looks like this

```
.cstart
chem data
.cend
```

Lines containing the keywords **.cstart** and **.cend** start and end the input for *chem*, respectively. In *pic* context, i.e., after the call of **.PS**, *chem* input can optionally be started by the line **begin chem** and ended by the line with the single word **end** instead.

Anything outside these initialization lines is copied through without modification; all data between the initialization lines is converted into *pic* commands to draw the diagram.

As an example,

```
.cstart
CH3
bond
CH3
.cend
```

prints two **CH3** groups with a bond between them.

If you want to create just *groff* output, you must run *chem* followed by *groff* with the option **−p** for the activation of *pic*:

chem [*file* . . .] **| groff −p** . . .

Language

The *chem* input language is rather small. It provides rings of several styles and a way to glue them together as desired, bonds of several styles, moieties (e.g., **C**, **NH3**, . . ., and strings.

Setting variables

There are some variables that can be set by commands. Such commands have two possible forms, either

variable value

or

variable **=** *value*

This sets the given *variable* to the argument *value*. If more arguments are given only the last argument is taken, all other arguments are ignored.

There are only a few variables to be set by these commands:

textht *arg*
Set the height of the text to *arg*; default is 0.16.

cwid *arg*
Set the character width to *arg*; default is 0.12.

db *arg* Set the bond length to *arg*; default is 0.2.

size *arg*
Scale the diagram to make it look plausible at point size *arg*; default is 10 point.

Bonds

This

bond [*direction*] [*length n*] [**from** *Name*|*picstuff*]

draws a single bond in direction from nearest corner of *Name*. **bond** can also be **double bond**, **front bond**, **back bond**, etc. (We will get back to *Name* soon.)

direction is the angle in degrees (0 up, positive clockwise) or a direction word like **up**, **down**, **sw** (= southwest), etc. If no direction is specified, the bond goes in the current direction (usually that of the last bond).

Normally the bond begins at the last object placed; this can be changed by naming a **from** place. For instance, to make a simple alkyl chain:

CH3	
bond	(this one goes right from the CH3)
C	(at the right end of the bond)
double bond up	(from the C)
O	(at the end of the double bond)
bond right from C	
CH3	

A length in inches may be specified to override the default length. Other *pic* commands can be tacked on to the end of a bond command, to created dotted or dashed bonds or to specify a **to** place.

Rings

There are lots of rings, but only five- and six-sided rings get much support. **ring** by itself is a six-sided ring; **benzene** is the benzene ring with a circle inside. **aromatic** puts a circle into any kind of ring.

ring [**pointing** (**up**|**right**|**left**|**down**)] [**aromatic**] [**put Mol at** *n*] [**double** *i*,*j k*,*l* ... [*picstuff*]

The vertices of a ring are numbered 1, 2, ... from the vertex that points in the natural compass direction. So for a hexagonal ring with the point at the top, the top vertex is 1, while if the ring has a point at the east side, that is vertex 1. This is expressed as

```
R1: ring pointing up
R2: ring pointing right
```

The ring vertices are named **.V1**, ..., **.V***n*, with **.V1** in the pointing direction. So the corners of **R1** are **R1.V1** (the *top*), **R1.V2**, **R1.V3**, **R1.V4** (the *bottom*), etc., whereas for **R2**, **R2.V1** is the rightmost vertex and **R2.V4** the leftmost. These vertex names are used for connecting bonds or other rings. For example,

```
R1: benzene pointing right
R2: benzene pointing right with .V6 at R1.V2
```

creates two benzene rings connected along a side.

Interior double bonds are specified as **double** *n1*,*n2 n3*,*n4* ...**;** each number pair adds an interior bond. So the alternate form of a benzene ring is

ring double 1,2 3,4 5,6

Heterocycles (rings with something other than carbon at a vertex) are written as **put** *X* **at** *V*, as in

R: ring put N at 1 put O at 2

In this heterocycle, **R.N** and **R.O** become synonyms for **R.V1** and **R.V2**.

There are two five-sided rings. **ring5** is pentagonal with a side that matches the six-sided ring; it has four natural directions. A **flatring** is a five-sided ring created by chopping one corner of a six-sided ring so that it exactly matches the six-sided rings.

The description of a ring has to fit on a single line.

Moieties and strings

A moiety is a string of characters beginning with a capital letter, such as N(C2H5)2. Numbers are converted to subscripts (unless they appear to be fractional values, as in N2.5H). The name of a moiety is determined from the moiety after special characters have been stripped out: e.g., N(C2H5)2) has the name NC2H52.

Moieties can be specified in two kinds. Normally a moiety is placed right after the last thing mentioned, separated by a semicolon surrounded by spaces, e.g.,

B1: bond ; OH

Here the moiety is **OH**; it is set after a bond.

As the second kind a moiety can be positioned as the first word in a *pic*-like command, e.g.,

CH3 at C + (0.5,0.5)

Here the moiety is **CH3**. It is placed at a position relative to **C**, a moiety used earlier in the chemical structure.

So moiety names can be specified as *chem* positions everywhere in the *chem* code. Beneath their printing moieties are names for places.

The moiety **BP** is special. It is not printed but just serves as a mark to be referred to in later *chem* commands. For example,

bond ; BP

sets a mark at the end of the bond. This can be used then for specifying a place. The name **BP** is derived from *branch point* (i.e., line crossing).

A string within double quotes **"** is interpreted as a part of a *chem* command. It represents a string that should be printed (without the quotes). Text within quotes **"**. . .**"** is treated more or less like a moiety except that no changes are made to the quoted part.

Names

In the alkyl chain above, notice that the carbon atom **C** was used both to draw something and as the name for a place. A moiety always defines a name for a place; you can use your own names for places instead, and indeed, for rings you will have to. A name is just

*Name***:** . . .

Name is often the name of a moiety like **CH3**, but it need not to be. Any name that begins with a capital letter and which contains only letters and numbers is valid:

First: bond
bond 30 from First

Miscellaneous

The specific construction

bond . . . **; moiety**

is equivalent to

```
bond
moiety
```

Otherwise, each item has to be on a separate line (and only one line). Note that there must be whitespace

after the semicolon which separates the commands.

A period character **.** or a single quote **'** in the first column of a line signals a *troff* command, which is copied through as-is.

A line whose first non-blank character is a hash character (**#**) is treated as a comment and thus ignored. However, hash characters within a word are kept.

A line whose first word is **pic** is copied through as-is after the word **pic** has been removed.

The command

size *n*

scales the diagram to make it look plausible at point size *n* (default is 10 point).

Anything else is assumed to be *pic* code, which is copied through with a label.

Since *chem* is a *pic* preprocessor, it is possible to include *pic* statements in the middle of a diagram to draw things not provided for by *chem* itself. Such *pic* statements should be included in *chem* code by adding **pic** as the first word of this line for clarity.

The following *pic* commands are accepted as *chem* commands, so no **pic** command word is needed:

define Start the definition of *pic* macro within *chem*.

[Start a block composite.

] End a block composite.

{ Start a macro definition block.

} End a macro definition block.

The macro names from **define** statements are stored and their call is accepted as a *chem* command as well.

Wish list

This TODO list was collected by Brian Kernighan.

Error checking is minimal; errors are usually detected and reported in an oblique fashion by *pic*.

There is no library or file inclusion mechanism, and there is no shorthand for repetitive structures.

The extension mechanism is to create *pic* macros, but these are tricky to get right and don't have all the properties of built-in objects.

There is no in-line chemistry yet (e.g., analogous to the **$**. . .**$** construct of *eqn*).

There is no way to control entry point for bonds on groups. Normally a bond connects to the carbon atom if entering from the top or bottom and otherwise to the nearest corner.

Bonds from substituted atoms on heterocycles do not join at the proper place without adding a bit of *pic*.

There is no decent primitive for brackets.

Text (quoted strings) doesn't work very well.

A squiggle bond is needed.

Files

/usr/local/share/groff/1.23.0/pic/chem.pic
A collection of *pic* macros needed by *chem*.

/usr/local/share/groff/1.23.0/tmac/pic.tmac
A macro file which redefines **.PS**, **.PE**, and **.PF** to center *pic* diagrams.

/usr/local/share/doc/groff−1.23.0/examples/chem/.chem*
Example files for *chem*.

/usr/local/share/doc/groff−1.23.0/examples/chem/122/.chem*
Example files from the *chem* article by its authors, "CHEM—A Program for Typesetting Chemical Structure Diagrams: User Manual" (CSTR #122).

Authors

The GNU version of *chem* was written by Bernd Warken ⟨groff−bernd.warken−72@web.de⟩. It is based on the documentation of Brian Kernighan's original *awk* version of *chem*.

See also

"CHEM—A Program for Typesetting Chemical Diagrams: User Manual" by Jon L. Bentley, Lynn W. Jelinski, and Brian W. Kernighan, 1992, AT&T Bell Laboratories Computing Science Technical Report No. 122

groff(1), *pic*(1)