Abstract

The chemschemex package provides a comfortable method for the typesetting of (chemical) schemes based on TikZ code, including an automatical structure referencing.

Example 1: Chemical scheme (left) produced by a simple code (right).

Contents

1 Introduction and motivation 2
2 Usage 2
3 User commands 3
  3.1 Basic commands 3
  3.2 Structure commands 3
  3.3 The Chemscheme environment 6
  3.4 Ref commands 8
  3.5 Arrows and simples 8
4 Options 10
  4.1 The image option 10
  4.2 The labelseparator option 10
  4.3 The arrowadvance option 10

*This document corresponds to chemschemex v1.0, dated 2014/07/15.
1 Introduction and motivation

While \LaTeX{} is a powerful tool for mathematical or physical issues the typesetting of chemistry derived problems is still a little bit annoying. When I wrote my thesis in organic chemistry I missed a package which produces chemical schemes as easy as you include a graphic into your document. I simply wanted to draw my structures in ChemDraw, include them into my document and label them.

The packages \texttt{chemscheme} and \texttt{chemnum} offer a possibility to rerender image files for this purpose. Nevertheless, they only modify a scheme that already exists. This means, that arrows, margins, alignments and other parameters cannot be defined or changed globally in your document.

By using the TikZ and the \texttt{fancylabel} package (which has actually been written as slave of this package) the \texttt{chemschemex} package meets all these requirements (see example 1).

2 Usage

\begin{verbatim}
\usepackage[<options>]{chemschemex}
\end{verbatim}

The command above will load the \texttt{chemschemex} package. It requires the packages \texttt{xkeyval}, \texttt{etextools}, \texttt{xargs}, \texttt{ifthen}, \texttt{suffix}, TikZ, \texttt{graphicx}, and \texttt{fancylabel}. I strongly recommend to read the documentation of the \texttt{fancylabel} package because all referencing functions are provided by this package. It contains a lot of useful options that are not described in this documentation.
3 User commands

3.1 Basic commands

\customstruct \customstruct[<TikZ-capt>][<TikZ-obj>]{<capt>}{<obj>}

The \customstruct command typesets the object \textit{<obj>} in the first row of a \textit{TikZ} matrix and the caption \textit{<capt>} in the second row. The caption is supposed to be given as comma-separated list of label(s) and text. The two optional arguments \textit{<TikZ-capt>} and \textit{<TikZ-obj>} can be used to pass options to \textit{TikZ} elements (for further information see section 5.1). All the following structure commands are based on \customstruct.

\begin{verbatim}
\customstruct[nodes={draw=blue}][draw=red]
    {{label1:},{text}},
    {{ll2:},{a longer text}}
{object e.g. image}
\end{verbatim}

\CSXimage \CSXimage[<img-opt>]{<img>}

The \CSXimage command includes the image \textit{<img>} using the global options \textit{<global-img-opt>} defined by the \textit{image} option (see section 4.1) and the options given by \textit{<img-opt>}. This command is used in all the following structure commands and expands to \texttt{\includegraphics[<global-img-opt>,<img-opt>]{<img>}}.

3.2 Structure commands

\struct \struct[<capt>][<fam>][<img-opt>][<TikZ-capt>][<TikZ-obj>]{<img>}

The \struct command includes the image \textit{<img>}, sets a fancylabel (therefore it uses \textit{<img>} as marker and \textit{<fam>} as family, default: \textit{<fam>}=CSX; the use of families is described in the \textit{fancylabel} package) and prints it. If a \textit{<caption>} is given, it will also print the caption behind the label. The macro \texttt{\CSXlabelsep} can be changed with the \texttt{\labelseparator} option.

\begin{verbatim}
\struct[text]{maleimid}
    expands to:
\customstruct{\{\fancylabel[CSX]{maleimid}\}\CSXlabelsep},{}\}}
\end{verbatim}

\begin{verbatim}
\struct{N-R}
\end{verbatim}

\begin{verbatim}
\struct{text}{maleimid}
    expands to:
\customstruct{\{\fancylabel[CSX]{maleimid}\}\CSXlabelsep},{}\}}
\SFimage{maleimid}
\end{verbatim}
\structalt\structalt[<capt>][<fam>][<img-opt>][<TikZ-capt>][<TikZ-obj>]{<img>}{<alt-img>}

The \structalt command includes an image <alt-img> but the labeling corresponds to <img>.

\structalt{product}{product_num}
expands to:
\customstruct{\fancylabel[CSX]{product},{}\}
{\CSXimage{product_num}}

\struct\struct*[<capt>][<fam>][<img-opt>][<TikZ-capt>][<TikZ-obj>]{<img>}{<alt-img>}
\structalt*\structalt*[<capt>][<fam>][<img-opt>][<TikZ-capt>][<TikZ-obj>]{<img>}{<alt-img>}

The \struct* and \structalt* commands do the same as the corresponding nonstarred versions but use \fancylabel* instead of \fancylabel. This means, that a label for this structure will be defined but not printed.

\struct*[maleimid]
expands to:
\customstruct{\fancylabel*[CSX]{maleimid},{}\}
{\CSXimage{maleimid}}

\struct-\struct-\struct-\struct-\struct-\struct-\struct-\struct-\struct-\struct-
\struct-[<capt>][<img-opt>][<TikZ-capt>][<TikZ-obj>]{<img>}

The \struct- command includes an image <img> without any labeling.

\struct-[text]{maleimid}
expands to:
\customstruct{{{},{}\}
{\CSXimage{maleimid}}

\newstruct\newstruct[<sublabels>]{<img>}{<structname>}{<Structname>}{<abbr>}

If you want to use substructures you have to define them in the preamble. The optional argument <sublabels> is a comma-separated list of subcaptions. Whenever you want to refer to them you just have to use their numbers. In this example the first entry (R=H) gets the number 1, the second entry (R=Me) gets the number 2 and so on. If you use a structure without substructures it is not necessary to use \newstruct. However, the \newstruct command defines the name (and Name) and abbreviation of the structure what allows you to use the commands \structname, \Structname and \structabbr for this structure.

\newstruct{(R=H),(R=Me),(R=Ph)}{product}{}}

Note: All of the following structure commands assume that \newstruct has been used for the filename <img> before.
```
\Struct \\
\Struct[<fam>][<img-opt>][<TikZ-capt>][<TikZ-obj>]{<sublabels>}{<img>}

To use the $\text{	extbackslash Struct}$ command, the structure and its substructure have to be defined by the $\text{	extbackslash newstruct}$ command. It includes the image $\text{<img>}$, sets the sublabels given in the comma-separated list of $\text{<sublabels>}$ and prints each sublabel with the subcaption previously given to the $\text{	extbackslash newstruct}$ command (one line per sublabel).

\begin{figure}[ht]
\centering
\includegraphics[width=0.2\textwidth]{product}
\caption{\text{3a: $R=H$\hspace{0.5cm} 3b: $R=Me$\hspace{0.5cm} 3c: $R=Ph$}}
\end{figure}

$\text{\Struct[1,2,3]{product}}$ expands to:
\begin{verbatim}
\customstruct{{\fancysublabel[CSX]{product}{1}\CSXlabelsep},{R=H}},
\customstruct{{\fancysublabel[CSX]{product}{2}\CSXlabelsep},{R=Me}},
\customstruct{{\fancysublabel[CSX]{product}{3}\CSXlabelsep},{R=Ph}}
\end{verbatim}
\begin{figure}[ht]
\centering
\includegraphics[width=0.2\textwidth]{product}
\caption{\text{3a, c}}
\end{figure}
```

```
\Struct*
\\Struct*[<fam>][<img-opt>][<TikZ-capt>][<TikZ-obj>]{<sublabels>}{<img>}

To use the $\text{	extbackslash Struct*}$ command, the structure and its substructure have to be defined by the $\text{	extbackslash newstruct}$ command. It includes the image $\text{<filename>}$, sets the sublabels given in the comma-separated list of $\text{<sublabels>}$ and prints each sublabel without its subcaption previously given to the $\text{	extbackslash newstruct}$ command.

\begin{figure}[ht]
\centering
\includegraphics[width=0.2\textwidth]{product}
\caption{\text{3a, c}}
\end{figure}
```

```
\Structalt
\\Structalt[<fam>][<img-opt>][<TikZ-capt>][<TikZ-obj>]{<sublabels>}{<img>}{<alt-img>}

Works like the $\text{	extbackslash Struct}$ command but includes the image $\text{<alt-img>$. Labeling corresponds to $\text{<img>}$.

\begin{figure}[ht]
\centering
\includegraphics[width=0.2\textwidth]{product}
\caption{\text{3a, c}}
\end{figure}
```

```
\Structalt*
\\Structalt*[<fam>][<img-opt>][<TikZ-capt>][<TikZ-obj>]{<sublabels>}{<img>}{<alt-img>}

Works like the $\text{	extbackslash Struct*}$ command but includes the image $\text{<alt-img>$. Labeling corresponds to $\text{<img>}$.

\begin{figure}[ht]
\centering
\includegraphics[width=0.2\textwidth]{product}
\caption{\text{3a, c}}
\end{figure}
```

```
\structname
\\structname{<img>}

Prints the name of the structure $\text{<img>}$ that has been previously defined by the $\text{	extbackslash newstruct}$ command. This command is recommended for the chemical name without a leading capital letter (inside a sentence).

\begin{figure}[ht]
\centering
\includegraphics[width=0.2\textwidth]{product}
\caption{\text{3a, c}}
\end{figure}
```

```
\structname
\\structname{<img>}

Prints the name of the structure $\text{<img>}$ that has been previously defined by the $\text{	extbackslash newstruct}$ command. This command is recommended for the chemical name with a leading capital letter (at the beginning of a sentence).

\begin{figure}[ht]
\centering
\includegraphics[width=0.2\textwidth]{product}
\caption{\text{3a, c}}
\end{figure}
```
\structabbr{} \structabbr{<img>}

Prints the abbreviation of the structure <img> that has been previously defined by the \newstruct command.

### 3.3 The Chemscheme environment

\begin{Chemscheme}...structure code...\end{Chemscheme}

If a structure command appears outside a Chemscheme environment each command will typeset the image and caption in its own matrix. This causes no kind of adjustment.

\begin{center}
\begin{tikzpicture}
\draw[thick] (0,0) -- (0,1) -- (1,1) -- (1,0) -- cycle;
\draw (0.5,0.5) circle (0.25);
\node at (0.5,0.5) {N-R};
\node at (0,0) {O};
\node at (1,0) {O};
\end{tikzpicture}
\end{center}

\struct{cp}
\struct{long caption}{maleimid}

But if structure commands appear inside a Chemscheme environment all images and captions are printed in one matrix. This causes adjustment of the image (by default center) and the caption row (by default top) according to the TikZ style \CSXmatrix.

\begin{center}
\begin{tikzpicture}
\draw[thick] (0,0) -- (0,1) -- (1,1) -- (1,0) -- cycle;
\draw (0.5,0.5) circle (0.25);
\node at (0.5,0.5) {N-R};
\node at (0,0) {O};
\node at (1,0) {O};
\end{tikzpicture}
\end{center}

\begin{Chemscheme}
\struct{cp}
\struct{long caption}{maleimid}
\end{Chemscheme}

\ChemschemeNextRow
\ChemschemeNextRow[<row-sep>]

If you want to use the matrix adjustment over multiple lines you can produce a 'linebreak' using the \ChemschemeNextRow command. The optional argument <row-sep> allows you to define the space between the rows.
\begin{Chemscheme}
\struct{maleimid}
\RightArrow{\struct{cp}}{DEE}
\struct[80,\%]{product}
\end{Chemscheme}

\begin{Chemscheme}
\struct{maleimid}
\LeftArrow{\struct{cp}}{DEE}
\struct[80,\%]{product}
\end{Chemscheme}

\textbf{DEE}

\begin{Chemscheme}
\struct{maleimid}
\RightArrow{\struct{cp}}{DEE}
\struct[80,\%]{product}
\end{Chemscheme}

\begin{Chemscheme}
\struct[80,\%]{product}
\LeftArrow{\struct{cp}}{DEE}
\struct{maleimid}
\end{Chemscheme}

\CSXcommands{\begin{TikZ-code}\end{TikZ-code}}

The \texttt{CSXcommands} macro allows you to draw any \texttt{TiKZ} element(s) after the type-setting of the structure matrix.

\begin{Chemscheme}
\struct{maleimid}
\RightArrow{\struct{cp}}{DEE}
\Struct{1,2}{product}
\CSXcommands{
\draw[->,CSXallarrows,draw=red]
(Scheme\theCSXscheme Caption3Entry1.west) to
([xshift=-8pt]Scheme\theCSXscheme Caption3Entry1.west) to
node[auto,swap,CSXlabelfont,red]{a}
([xshift=-8pt]Scheme\theCSXscheme Caption3Entry2.west) to
(Scheme\theCSXscheme Caption3Entry2.west);
}
\end{Chemscheme}
3.4 Ref commands

\structref \structref[^{<fam>}]{{<img>}}
\structref* \structref*[^{<fam>}]{{<img>}}
\structsubref \structsubref[^{<fam>}]{{<img>}}{{<sublabels>}}
\structsubref- \structsubref-[^{<fam>}]{{<img>}}{{<sublabels>}}
\structsubref* \structsubref*[^{<fam>}]{{<img>}}{{<sublabels>}}

The chemschemex package defines ref commands that actually do exactly what their analogs from the fancylabel package do, but with CSX as default family.¹

\CSXstructref The \CSXstructref macro allows you to change the style of all referencing commands that are shown above. The definition is shown below and may be changed as required.

\newcommand{\CSXstructref}[1]{%
    \textbf{#1}%
}

3.5 Arrows and simples

\customarrow \customarrow[^{<length>}]{{<style>}}{{<upper-capt>}}{{<lower-capt>}}

The basic command for arrows is the \customarrow command. If the optional argument \texttt{<length>} is used, the arrow will have this length. Otherwise the arrow is stretched to the length of the widest caption advanced by the length globally defined via the \texttt{arrowadvance} option. The style argument \texttt{<style>} allows you to pass options to the Ti\kZ \texttt{draw} command.

\begin{tikzpicture}
    \draw[->,line width=1pt] (0,0) -- (1,0) node [above, midway] {this is a long caption} node [below, midway] {short};
\end{tikzpicture}

\customarrow[^{60pt}]{{<style>}}{{<upper-capt>}}{{<lower-capt>}}

\RightArrow \RightArrow[^{<length>}]{{<upper-capt>}}{{<lower-capt>}}
\arrow-cmd \arrow-cmd[^{<length>}]{{<upper-capt>}}{{<lower-capt>}}

The table below shows a couple of arrows that are based on the \customarrow command. You might define some other arrows using the \CSXdeclarearrow command.

\begin{tabular}{c}
\RightArrow \RightArrowDashed \DSRightArrow \\
\CSXRightArrow \CSXRightArrowDashed \DSXRightArrowDashed
\end{tabular}

¹For further information please have a look into the fancylabel package documentation.
All shown arrows have a normal, a crossed out (leading C), a striked out (leading S) and a double striked out (leading DS) version of the solid and the dashed (appending Dashed) arrow.

\begin{Chemscheme}
\struct{maleimid} \structplus \struct{cp} \RightArrow{}{DEE} \struct{product}
\end{Chemscheme}

\CSXdeclarearrow{\arrow-cmd}{<style>}
You can use the \CSXdeclarearrow command to declare arrows based on the customstruct command. The definition of \RightArrow is:

\CSXdeclarearrow{\RightArrow}{CSXnormalarrows,CSXRightArrow}

Considering the default setting of CSXnormalarrows and CSXRightArrow this means:

\CSXdeclarearrow{\RightArrow}{line width=0.7pt,->}

\structplus \structminus

\structplus and \structminus can be used like any structure or arrow command:

\begin{Chemscheme}
\begin{align*}
\struct{maleimid} & \structplus \\
\struct{cp} & \RightArrow{}{DEE} \\
\struct{product} & \end{align*}
\end{Chemscheme}
4 Options

4.1 The image option

\usepackage[image={<global-img-opt}>]{chemschemex}

All structure commands except \customstruct internally use the \CSXimage command to include pictures with \includegraphics. The image option allows you to define options that will be passed to any image that is inserted via \CSXimage. The default value is image={scale=0.7}.

4.2 The labelseparator option

\usepackage[labelseparator=<value>]{chemschemex}

The <value> given by the labelseparator option is set behind every \fancylabel inside a structure command if some text follows. The value is saved in \CSXlabelsep. The default value is labelseparator={:\,}.

\begin{verbatim}
\begin{structure}
  N-R
\end{structure}
\end{verbatim}

4.3 The arrowadvance option

\usepackage[arrowadvance=<length>]{chemschemex}

Every arrow with undefined length argument will be as long as its widest caption plus the length given by the arrowadvance option. This is also the minimal length of an arrow (when no captions are given). The default value is arrowadvance=10pt.

5 Customization and advanced examples

5.1 Predefined TikZ styles

There are a lot of TikZ styles that are already defined by the chemschemex package. The following two examples show the code that is generated by chemschemex and hopefully help you to understand the function of each style. Some of them can be changed (green), some of them must not be changed (red) to prevent strange results or even errors. If you want to change fonts please use the blue coloured styles instead of appending font= to any style since this will cause wrong calculations. The orange entries are styles that are optional arguments of \customstruct.
5.2 Style adjustment – some examples

As already mentioned above, please use the styles \texttt{CSXlabelfont} and \texttt{CSXtextfont} for any changes of the node font. This is necessary to ensure correct measurements. For local font adjustment of captions in structure commands use the optional argument \texttt{<TikZ-capt>}.

\begin{tikzpicture}
  \node (Scheme1Image2) at (0,0) {
    \struct\[text\]{maleimid}
  };

  \begin{scope}[nodes={draw=green},\texttt{CSXtextfont/.style={blue,font={\large}}}]
    \struct[\]
    \struct[\]
    \struct[\]
  \end{scope}

  \draw[red] (Scheme1Image2.north) -- (Scheme1Image2.south);

  \begin{scope}[\texttt{CSXlabelfont/.style={red,font={\large}}}]
    \struct[\]
    \struct[\]
  \end{scope}

  \node (Scheme1Image2Lower) at (Scheme1Image2.south) {
    \struct[\]
    \struct[\]
  };

  \draw[red] (Scheme1Image2Lower.north) -- (Scheme1Image2Lower.south);

\end{tikzpicture}
5.3 Chemical mechanisms

For more complex chemical mechanisms you can either use the matrix provided by the \texttt{Chemscheme} environment or the internal commands \texttt{\textbackslash CSXimage} and \texttt{\textbackslash fancylabel}:

\begin{tikzpicture}
\usetikzlibrary{matrix}
\tikzset{CSXmatrix/.append style={column sep=30pt}}
\begin{Chemscheme}
\struct{R2-CH=CH-R1}
\struct{Ru=CH2}
\struct{R1-CH=CH2}\ChemschemeNextRow[20pt]
\struct{CH2=CH-R2}
\struct{Ru=CH-R1}
\struct{C2H4}
\CSXcommands{
  \draw[->,draw=blue] (Scheme\theCSXscheme Image4)
  to [bend right=90,distance=22pt] (Scheme\theCSXscheme Image1);
  \draw[->,draw=red] (Scheme\theCSXscheme Image5)
  to [bend right=-90,distance=22pt] (Scheme\theCSXscheme Image2);
  \draw[->,draw=orange] (Scheme\theCSXscheme Image2)
  to [bend right=90,distance=22pt] (Scheme\theCSXscheme Image5);
  \draw[->,draw=green!75!black] (Scheme\theCSXscheme Image3)
  to [bend right=-90,distance=22pt] (Scheme\theCSXscheme Image6);
}
\end{Chemscheme}
\end{tikzpicture}