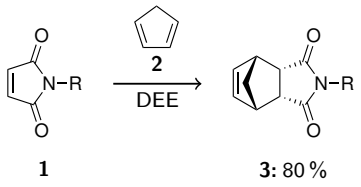
The chemschemex package*

Dominik Siegel
dominik.siegel@yahoo.de

2018/01/20

Abstract

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




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

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

Contents

*This document corresponds to chemschemex v1.1.1, dated 2017/04/03.

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, inlcude them into my document and label them.

The packages chemscheme and 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 fancylabel package (which has actually been written as slave of this package) the chemschemex package meets all these requirements (see example 1).

2 Usage

`\usepackage` `\usepackage[<options>]{chemschemex}`

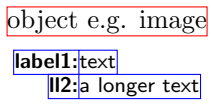
The command above will load the chemschemex package. It requires the packages xkeyval, etextools, xargs, ifthen, suffix, TikZ, graphicx, and fancylabel. I strongly recommend to read the documentation of the fancylabel package because all referencing functions are provided by this package. It contains 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 `<obj>` in the first row of a TikZ matrix and the caption `<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 `<TikZ-capt>` and `<TikZ-obj>` can be used to pass options to TikZ elements (for further information see section 5.1). All the following structure commands are based on `\customstruct`.




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

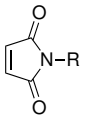
`\CSXimage` `\CSXimage[<img-opt>]{<img>}`

The `\CSXimage` command includes the image `<img>` using the global options `<global-img-opt>` defined by the `image` option (see section 4.1) and the options given by `<img-opt>`. This command is used in all the following structure commands and expands to `\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 inlcudes the image `<img>`, sets a fancylabel (therefore it uses `<img>` as marker and `<fam>` as family, default: `<fam>`=`CSX`; the use of families is described in the fancylabel package) and prints it. If a `<caption>` is given, it will also print the caption behind the label. The macro `\CSXlabelsep` can be changed with the `labelseparator` option.

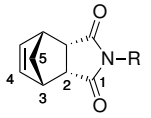


1: text

```
\struct[text]{maleimid}
   expands to:
\customstruct{{{\fancylabel[CSX]{maleimid}\CSXlabelsep},{}}}
             {\CSXimage{maleimid}}
```

`\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>`.



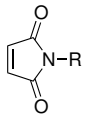
3

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

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

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

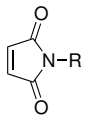
The `\struct*` and `\structalt*` commands do the same as the corresponding nonstarred versions but use `\fancylabel*` instead of `\fancylabel`. This means, that a label for this structure will be defined but not printed.




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

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

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



text

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

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

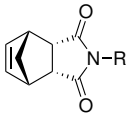
If you want to use substructures you have to define them in the preamble. The optional argument `<sublabels>` is a comma-separated list of subcaptions. Whenever you want to refer to them you just have to use their numbers. In this example the first entry (`R=H`) gets the number 1, the second entry (`R=Me`) 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) und 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 `\Struct` command, the structure and its substructure have to be defined by the `\newstruct` command. It inlcudes the image `<img>`, sets the sublabels given in the comma-separated list of `<sublabels>` and prints each sublabel with the subcaption previously given to the `\newstruct` command (one line per sublabel).

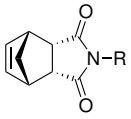


3a: R=H
3b: R=Me
3c: R=Ph

```
\Struct{1,2,3}{product}
   expands to:
\customstruct{{{\fancysublabel[CSX]{product}{1}\CSXlabelsep},{R=H}},
              {{\fancysublabel[CSX]{product}{2}\CSXlabelsep},{R=Me}},
              {{\fancysublabel[CSX]{product}{3}\CSXlabelsep},{R=Ph}}}
             {\CSXimage{product}}
```

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

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



3a, c

```
\Struct*{1,3}{product}
   expands to:
\customstruct{{{\fancysublabel[CSX]{product}{1,3}},{}}}
             {\CSXimage{product}}
```

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

Works like the `\Struct` command but includes the image `<alt-img>`. Labeling corresponds to `<img>`.

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

Works like the `\Struct*` command but includes the image `<alt-img>`. Labeling corresponds to `<img>`.

\structname `\structname{<img>}`

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

\Structname `\Structname{<img>}`

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

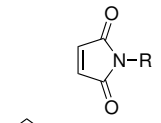
`\structabbr` `\structabbr{<img>}`

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

3.3 The Chemscheme environment

`Chemscheme` `\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.

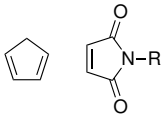


1: long caption

2

```
\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`.

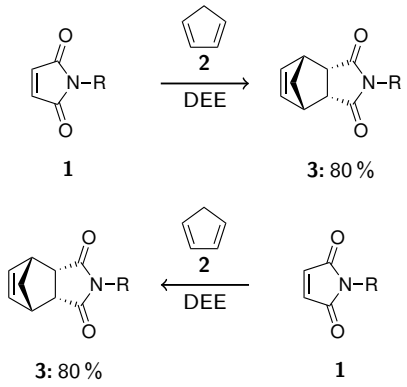


2 **1:** long caption

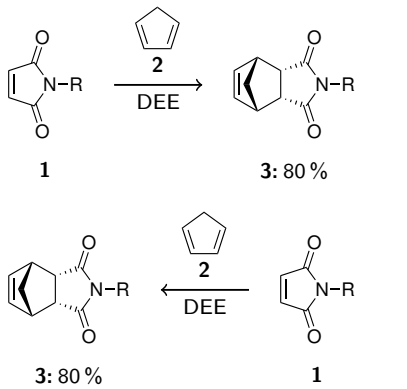
```
\begin{Chemscheme}
   \struct{cp}
   \struct[long caption]{maleimid}
\end{Chemscheme}
```

`\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.



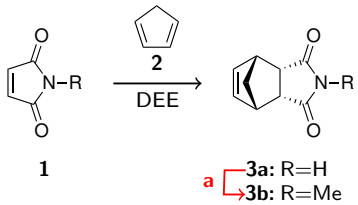

```
\begin{Chemscheme}
   \struct{maleimid}
   \RightArrow{\struct{cp}}{DEE}
   \struct[80\,\%]{product}
   \ChemschemeNextRow[10pt]
   \struct[80\,\%]{product}
   \LeftArrow{\struct{cp}}{DEE}
   \struct{maleimid}
\end{Chemscheme}
```




```
\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` `\CSXcommands{<TikZ-code>}`

The `CSXcommands` macro allows you to draw any TikZ element(s) after the typesetting 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]{%
    % #1=fancyref command
    \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 `<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 `arrowadvance` option. The style argument `<style>` allows you to pass options to the TikZ `\draw` command.

this is a long caption
short

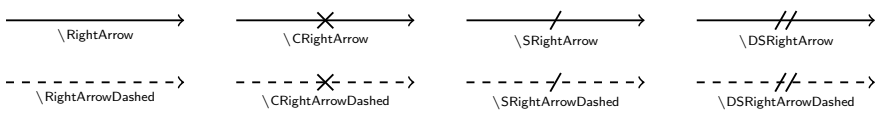
```
\customarrow{->,line width=1pt}
            {this is a long caption}
            {short}
```

this is a long caption
short

```
\customarrow[60pt]
            {->,CSXarrowupper/.append style={red}}
            {this is a long caption}
            {short}
```

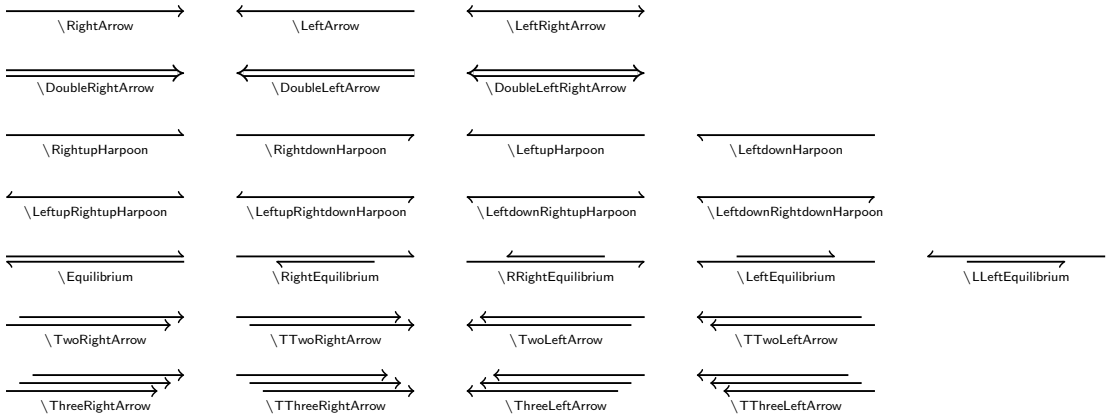
`\RightArrow` `\RightArrow[<length>]{<upper-capt>}{<lower-capt>}`
`\<arrow-cmd>` `\<arrow-cmd>[<length>]{<upper-capt>}{<lower-capt>}`

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.



[1] For 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.



`\CSXdeclarearrow` `\CSXdeclarearrow{<arrow-cmd>}{<style>}`

You can use the `\CSXdeclarearrow` command to declare arrows based on the customstruct command. The definition of `\RightArrow` is:

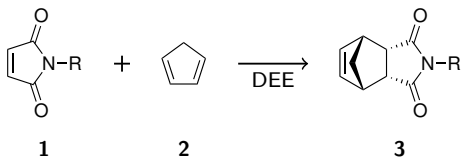
```
\CSXdeclarearrow{\RightArrow}{CSXnormalarrows,CSXRightArrow}
```

Considering the default setting of `CSXnormalarrows` and `CSXRightArrow` this means:

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

`\structplus` `\structplus`
`\structminus` `\structminus`

The simples (this is how I call arrow-like elements without a upper or lower caption) `\structplus` and `\structminus` can be used like any structure or arrow command:




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

4 Options

4.1 The image option

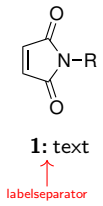
`image=` `\usepackage[image={<global-img-opt>}]{chemschemex}`

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

`labelseparator=` `\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={:\,}`.



4.3 The arrowadvance option

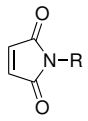
`arrowadvance=` `\usepackage[arrowadvance=<length>]{chemschemex}`

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

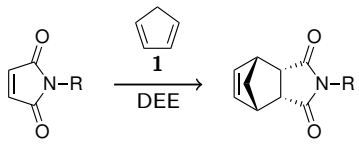
There are a lot of TikZ styles that are already defined by the chemschemex package. The following two examples show the code that is generated by chemschemex and hopefully help you to understand the function of each style. Some of them can be changed (green), some of them must not be changed (red) to 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`.



text

```
\customstruct{{{text}}}{\CSXimage{maleimid}}
```

```
\begin{tikzpicture}[CSX]
   \matrix [CSXmatrix]{
      % Image row
      \node[<TikZ-obj>]
           (Scheme1Image1)
           {\CSXimage{maleimid}};\\
      % Caption row
      \node(Scheme1Caption1)
           {\tikz[CSXcaption,<TikZ-capt>]
                 {\node[CSXtextfont,CSXsettextwidth,CSXcaptionTextOnly]
                       (Scheme1Caption1Entry1)
                       {text};
                 }
           };\\
   };
\end{tikzpicture}
```



text

label 1text 1
label 2text 2
text 3

```
\begin{Chemscheme}
   \customstruct{{{text}}}{\CSXimage{maleimid}}
   \RightArrow{\customstruct{{{1},{}}}
              {\CSXimage{cp}}}{DEE}
   \customstruct{{{label 1},{text 1}},
                 {{label 2},{text 2}},
                 {text 3}}
                {\CSXimage{product}}
\end{Chemscheme}
```

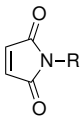
```
\begin{tikzpicture}[CSX]
   \matrix [CSXmatrix]{
      % Image row
      \node[<TikZ-obj>]
           (Scheme1Image1)
           {\CSXimage{maleimid}};&
      \node(Scheme1Image2)
           {\hbox to <arrowlength>{}};&
      \node[<TikZ-obj>]
           (Scheme1Image3)
           {\CSXimage{product}};\\
      % Caption row
      \node(Scheme1Caption1)
           {\tikz[CSXcaption,<TikZ-capt>]
                 {\node[CSXtextfont,CSXsettextwidth,CSXcaptionTextOnly]
```

```
                    (Scheme1Caption1Entry1)
                    {text};
              }
        };&
   \node(Scheme1Caption2)
        {};&
   \node(Scheme1Caption3)
        {\tikz[CSXcaption,<TikZ-capt>]
              {\node[CSXlabelfont,CSXsetlabelwidth,CSXcaptionLabelandText,
                     rectangle split,rectangle split horizontal,
                     rectangle split parts=2,rectangle split part align=base,
                     every two node part/.style={CSXtextfont,CSXsettextwidth}]
                    (Scheme1Caption3Entry1)
                    {label 1\nodepart{two}text 1};
               \node[CSXlabelfont,CSXsetlabelwidth,CSXcaptionLabelandText,
                     rectangle split,rectangle split horizontal,
                     rectangle split parts=2,rectangle split part align=base,
                     every two node part/.style={CSXtextfont,CSXsettextwidth}
                     below left=of Scheme1Caption3Entry1.one split south,
                     anchor=one split north]
                    (Scheme1Caption3Entry2)
                    {label 2\nodepart{two}text 2};
               \node[CSXtextfont,CSXsettextwidth,CSXcaptionTextOnly%
                     below=of Scheme1Caption3Entry2.south]
                    (Scheme1Caption3Entry3)
                    {text 3};
              }
        };\\
};
\draw[CSXallarrows,CSXRightArrow,text width=(<arrowlength>-\CSXarrowadvance)]
     ([CSXshiftA]Scheme1Image2.west) to
     node(Scheme1Image2Upper)
         [CSXarrowfont,CSXarrowupper,auto]
         {\tikz[remember picture,CSXStructInArrow]
               {\node[<TikZ-obj>]
                     (Scheme1Image2UpperImage1)
                     {\CSXimage{cp}};
                \node[below=of Scheme1Image2UpperImage1]
                     (Scheme1Image2UpperCaption1)
                     {\tikz[CSXcaption,<TikZ-capt>]
                           {\node[CSXlabelfont,CSXsetlabelwidth,
                                  CSXcaptionLabelandText,rectangle split,
                                  rectangle split horizontal,
                                  rectangle split parts=2,
                                  rectangle split part align=base,
                                  every two node part/.style={
                                     CSXtextfont,
                                     CSXsettextwidth
                                  }]
                                 (Scheme1Image2UpperCaption1Entry1)
```

```
                                        {1\nodepart{two}};
                                    }
                                };
                            }
                        }
                    node(Scheme1Image2Lower)
                        [CSXarrowfont,CSXarrowlower,auto,swap]
                        {DEE}
                    ([CSXshiftB]Scheme1Image2.east);
\end{tikzpicture}
```

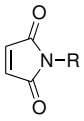
5.2 Style adjustment – some examples

As already mentioned above, please use the styles `CSXlabelfont` and `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 `<TikZ-capt>`.



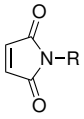
1: text

```
\struct[text]{maleimid}
```



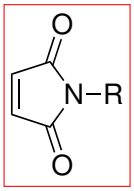
1: text

```
\struct[text]
       []
       []
       [CSXlabelfont/.style={red,font={\large}}]
       {maleimid}
```



1: text

```
\struct[text]
       []
       []
       [nodes={draw=green},
        CSXtextfont/.style={blue,font={\large}}]
       {maleimid}
```

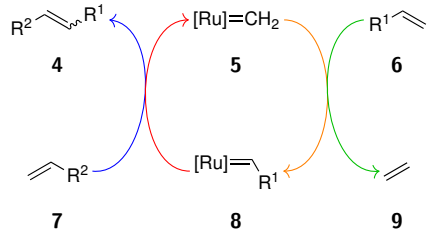


1

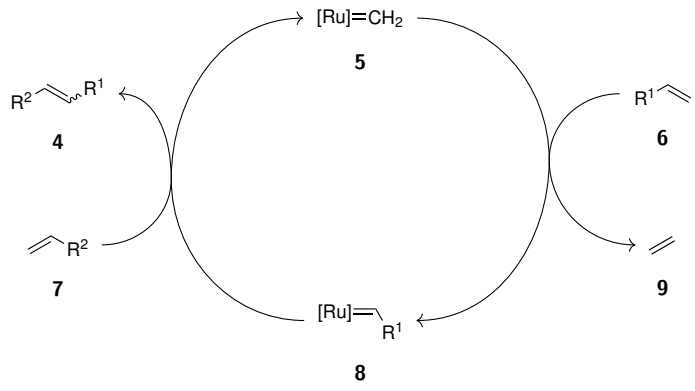
```
\struct[][][scale=1.5][][draw=red]{maleimid}
```

5.3 Chemical mechanisms

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




```
\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}
```




```
\tikz[node distance=1pt,mycaption/.style={CSXlabelfont}]{
   \node (n) at (0,2){\CSXimage{Ru=CH2}};
```

```
    \node (ncapt) [mycaption,below=of n] {\fancylabel[CSX]{Ru=CH2}};
    \node (s) at (0,-2){\CSXimage{Ru=CH-R1}};
    \node (scapt) [mycaption,below=of s] {\fancylabel[CSX]{Ru=CH-R1}};
    \node (nw) at (-4,1){\CSXimage{R2-CH=CH-R1}};
    \node (nwcapt) [mycaption,below=of nw] {\fancylabel[CSX]{R2-CH=CH-R1}};
    \node (sw) at (-4,-1){\CSXimage{CH2=CH-R2}};
    \node (swcapt) [mycaption,below=of sw] {\fancylabel[CSX]{CH2=CH-R2}};
    \node (ne) at (4,1){\CSXimage{R1-CH=CH2}};
    \node (necapt) [mycaption,below=of ne] {\fancylabel[CSX]{R1-CH=CH2}};
    \node (se) at (4,-1){\CSXimage{C2H4}};
    \node (ecapt) [mycaption,below=of se] {\fancylabel[CSX]{C2H4}};
    \node (w) at (-2.5,0) {};
    \node (e) at (2.5,0) {};
    \draw [->] (s) to [out=180,in=270] (w) to [out=90,in=180] (n);
    \draw [->] (n) to [out=0,in=90] (e) to [out=270,in=0] (s);
    \draw [->] (sw) to [out=0,in=270] (w) to [out=90,in=0] (nw);
    \draw [->] (ne) to [out=180,in=90] (e) to [out=270,in=180] (se);
}
```

6 Change history

2014/07/15 Initial version

2017/04/03 Bug within the \Struct command fixed

2018/01/20 Dependencies changed (xifthen instead of ifthen, etoolbox instead of etextools). Loading etextools caused compatibility issues with some other packages.