formcalc documentation

Ludwig Nittke

June 2025

Contents

1 Introduction

The Form calculus is a semi graphical calculus. It consists of cross ( ), void ( ) and variables. A variables is a names for any formula of the Form calculus. Variables can be divided into free and bound variables. Bound variables are variables that refer to a cross in the formula they are a part of (e.g.: ($h_1 =$ $\overline{h_1\,|}$ is bound). They can be represented by the so called re-entry form ($h_1 =$).

This package provides a simple toolkit to draw these graphical formulas. It is designed to provide simple out of the box functionality while being as customizable as possible.

This document contains a description of the environment and all the commands provided by the package including all possible options for customization. At the end it covers some tips and tricks for working with this package.

2 Options syntax

All options have to be given in the form `key=value`. If the value contains "`=`" or "`,`" it has to be surrounded by curly braces. Some specific options can be given without key or value.

All commands that take option do so with square braces. Thus if any option contains square closing braces, latex has to know which braces denote the end of the options. This can be given by additional curly braces inside the square ones (Example: `\formOptions[{key=value with ] braces}]`).

The options for the commands only modify the commands themselves unless stated otherwise.

3 Global options

The command `\formOptions[]` can be used to set global options until the end of the current latex group. Global options can also be set using package options, but package options get expanded automatically by latex, thus it is not recommended, especially if the set values contain commands.

input:

```
\begin{form}
    \cross{\cross{}}\cross{text}
\end{form}
\begin{form}[leveling=bottom up]
    \cross{\cross{}}\cross{text}
\end{form}
\begin{form}[leveling=top down]
    \cross{\cross{}}\cross{text}
\end{form}
```

output:

Possible keys:

- `cross`: a list of options as described in section 5.1
- `bound`: a list of options as described in section 5.2
- `void`: a list of options as described in section 5.3
- `leveling`: one of `top down`, `bottom up` or `off`; ensures that all crosses of the same level have the same height. `top down` counts level from top to bottom, `bottom up` the other way.
- `default`: restores default values; value will be ignored and is not mandatory

4 Environment

```
\begin{form}<content>\end{form}
\begin{form}[<options>]<content>\end{form}
```

Any formula has to be typed inside the `form` environment. It provides all the commands of the package. Inside the environment everything gets rendered before it is added to the document. As consequences the formcalc commands cannot be surrounded (e.g. by external commands, environments, latex groups, etc.) and end a latex group (all declarations get deleted). Additionally some commands may not work (e.g. `\label{}`).

Possible options are the same as global options.

The form environment preserves math mode. That means all its contents are in math mode if and only if the environment is in math mode. This includes numbers, names of bound variables and plain text. It does not include tags.

5 Commands

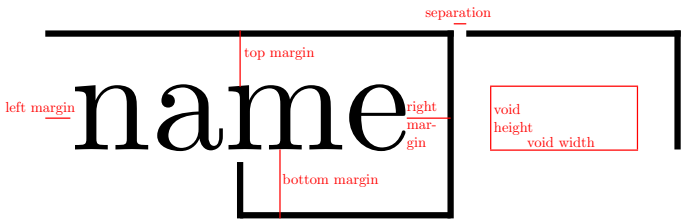
5.1 Cross

```
\cross{<content>}
\cross[<options>]{<content>}
```

input:

```
\begin{form}
    \cross[name=name]{\bound[visible]{1}}\cross{}
\end{form}
```

output:



Possible keys for the command and the global options:

- `left margin`
- `right margin`
- `top margin`
- `bottom margin`
- `separation`
- `linethickness`
- `number`: a list of options as described below
- `tag`: a list of options as described below
- `name`: the name of the cross as shown if the `visible` option of the corresponding `\bound{}` command is `true`; the in `number` determined displayed number can be used as `\#`
- `color`: only possible, if package `xcolor` is used; if not empty, `\color{value}` gets added before the lines of that cross and is available as `\crosscolor` in `name`, `number` and `tag`

Possible keys for the command only:

- `void width`: the width of the void inserted if this cross is empty
- `void height`: same as `void width`
- `recursive`: global options for this cross and its contents

input:

```
\begin{form}
    \cross[number={visible,right margin=0.5ex}]{}
\end{form}
```

output:

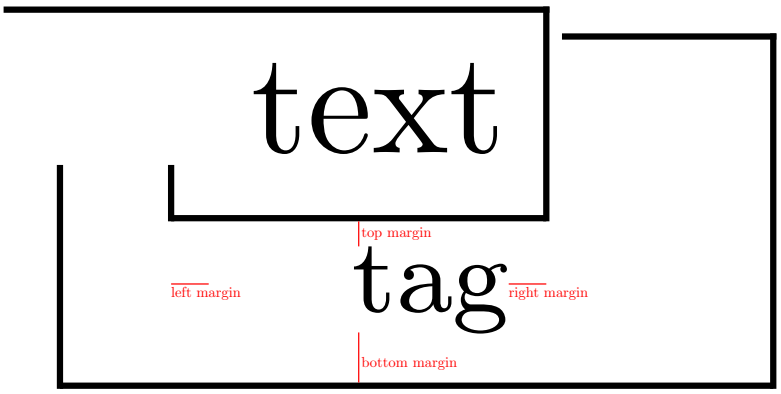
Possible keys for the number option:

- `right margin`
- `scale`: the scale factor by which the number gets smaller
- `visible`: shows the number if `true`; defaults to `true` if no value is given
- `display`: the number that should be shown if `visible` is `true`; key is not mandatory unless that leads to ambiguity; the number that the cross gets by counting can be used as `\#`; `\#` is the number that is the input for the `\bound{}` command.

input:

```
\begin{form}
    \cross[tag={\scriptsize tag,left margin=0.6ex,right margin=0.6ex}]
    {\bound{2}\bound{1}text}\cross{}
\end{form}
```

output:



Possible keys for the tag option:

- `left margin`
- `right margin`
- `top margin`
- `bottom margin`
- `display`: text to be shown below the re-entry of the cross; key is not mandatory unless that leads to ambiguity

5.2 Bound variable

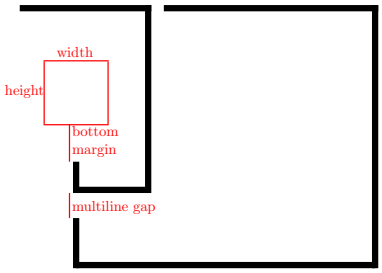
```
\bound{<cross number>}
\bound[<options>]{<cross number>}
```

`<cross number>` is a comma-separated list of at least one number referencing the corresponding cross. The numbers can be shown with the `cross={number={visible,\#}}` global option. If the list is longer than one item, for the purpos of collision detection (e.g. margins between re-entries, counting of crossings, etc.) it is treated as a series of bound variables (e.g. `\bound{2,1}` is the same as `\bound{2}\bound{1}`).

input:

```
\begin{form}
    \cross{\bound[bottom margin=0.6ex]{2,1}}\cross{}
\end{form}
```

output:



Possible keys for the bound command:

- `bottom margin`
- `visible`: shows the variable name if `true`; defaults to `true` if no value is given
- `width`
- `height`
- `multiline gap`
- `display`: override the variable name; key is not mandatory unless that leads to ambiguity

5.3 Void

input:

```
\begin{form}
    \void[frame]
\end{form}
```

output:

Possible keys for the void command:

- `frame`: shows a frame around the void if `true`; defaults to `true` if no value is given
- `width`
- `height`

6 Tips and tricks

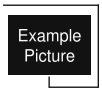
6.1 Commands in options

The content of the options gets copied as is. This allows options to include most commands.

input:

```
\begin{form}
    \cross[name={\includegraphics[height=2em]{picture.png}}]{\bound
    [visible]{1}}
\end{form}
```

output:



6.2 Length units

Latex has many units to specify lengths. Popular ones are `mm`, `cm`, `in` and `pt`. Those units reference lengths in the resulting document.

There are also units which reference font size. Those are `mu`, `ex` and `em`. One `mu` is $\frac{1}{18}$ `em` in math mode. As all lengths of this package are resolved during non-math mode this unit cannot be used. One `ex` is the height of `x` in the current font. One `em` is the height of `M` in the current font. The font referenced is the one at the beginning of the environment.

6.3 Spacing

In the form environment all space before and after commands is stripped and one space is added. This makes certain that newlines and spaces between commands don't add unwanted space, and ensures consistency between math mode and normal text. As an unfortunate side-effect commands like `\hspace{}` or `\hskip` don't work anymore.

A solution is to use `\makebox[]{}` and `\rule{}{}`. The command `\makebox[<width>]{\rule{0pt}{<height>}\hfill}` produces a box that only contains space but is of the given dimensions and gets treated as any other box (e.g. picture, character, table, etc.).

6.4 Spacing between lines

Latex tries to make the space between baselines `\baselineskip` wide. This dimension depends on font and font size. If the actual space between lines is smaller than `\lineskiplimit`, space is added to make it `\lineskip` wide. To set a consistent minimum spacing between lines both have to be modified:

```
\lineskip=0.8ex
\lineskiplimit=0.8ex
```

6.5 Output values

Use `\the` to show current values:

```
\the\lineskip
```

Use `\message{}` to print to the log:

```
\message{\the\lineskip}
```