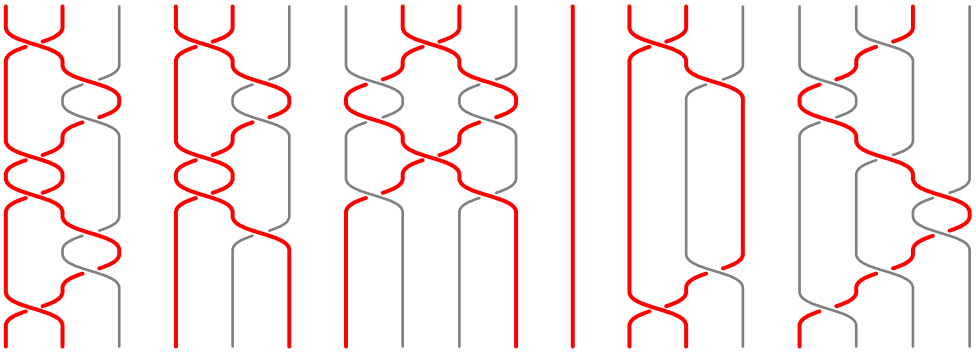
The braids Package: Documentation

Andrew Stacey
stacey@math.ntnu.no

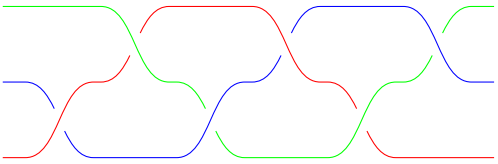
v1.0 from 2011/05/07



1 Introduction

This is a package for drawing braid diagrams using PGF/TikZ. An example follows.

```
\begin{center}
\begin{tikzpicture}
\braid[rotate=90,style strands={1}{red},style
    strands={2}{blue},style strands={3}{green}] s_1
    s_2^{-1} s_1 s_2^{-1} s_1 s_2^{-1};
\end{tikzpicture}
\end{center}
```



2 Usage

`\braid` A braid is specified by the command `\braid`. The syntax for this command is as follows:

```
\braid[style options] (name) at (coordinate) braid-word;
```

`braid-word` The `braid-word` is an expression in the braid group, such as `s_1 s_2^{-1}`. The generator labels are not significant. The exponent can be `1`, `{-1}`, or missing (in which case it defaults to `1`, note also that the exponent is read as a TEX-token so `{1}` is also legal). Certain other symbols are allowed in the `braid-word` which control the rendering of the braid. To get crossings to render at the same height, separate them with a hyphen (note: no check is made to ensure that the crossings can legally be put at the same height; *caveat emptor*). To draw a *floor*, precede the braid element by a vertical line. What happens then is that when the braid is rendered, the coordinates of the rectangle behind that crossing (wide enough to encompass all the strands) is passed to a command. The intention is that this command draw something behind the braid. The command is configurable by a key (see **??**).

`name` The (optional) `name` acts a little like the `name` of a TikZ node. When it is specified, the routine that renders the braid also saves certain coordinates as if they were node anchors. Specifically, `coordinate` nodes are placed at the centre of the braid diagram and at the ends of each strand. The centre has the label `name`, the strands are labelled `name-number-end` and `name-rev-number-end`, where `name` is the name given to the braid, `number` is the number of the strand counting from the left, and `end` is either `s` for the start or `e` for the end. If the version with `rev` is used then the numbers correspond to the *final* positions of the braids. The name can also be specified with the `name` key.

`at` The (optional) `at (coordinate)` syntax positions the braid at the `coordinate` in the current picture. Due to the implementation, the coordinate has to be known at the start, but the width and height of the braid are only known at the end. Therefore, the braid is positioned so that the start of the first strand is at `(coordinate)`. This can also be specified using the `at` key.

`style options` The `style options` set the style for the braid strands. They can be grouped into three types: options that set up the main parameters for the braid, options that set the default style for the strands, and options that set up styles for individual strands. The options are as follows.

2.1 Style Options

`number of strands` The key `number of strands` sets the minimum number of strands for the braid. The number of strands will grow according to the terms in the braid word so this merely sets a lower bound. If not set, the number of strands will be determined by the terms in the braid word.

`height` The key `height` sets the height of the piece of the braid corresponding to an element in the group.

`width` The key `width` sets the separation of the strands in the braid.

`border height` The key `border height` adds a little extra length to the strands at the start

and end of the braid.

`style strands` The style of the strands are controlled by two types of option. Style options that are set on the `\braid` command are passed to every strand. It is also possible to add style options to individual strands using the key `style strands`. This takes two options, a comma-delimited list of strand numbers (which could be just a single number) and a list of options to be applied to that strand. Thus, the syntax is `style strands={n,m,...}{options}`. The strands are numbered by their starting position. Not all of the standard TikZ style options are possible due to the way that the strands are constructed. Basically, the options that are allowed are those that do not require changing the path or drawing it more than once.

`floor command` When a floor is requested behind a crossing, the actual way to render it is determined by a command. This key allows the user to define that command. The argument to this key should be the code that should be executed for each floor. To avoid the hassle of getting the number of hashes right, the command should take no arguments. Rather, the coordinates of the rectangle are saved in to macros `\floorsx`, `\floorsy`, `\floorex`, `\floorey` (these macros will expand to something like `10pt`) and the command should use these to position the drawing. The default is to draw a line at the top and at the bottom of the rectangle.

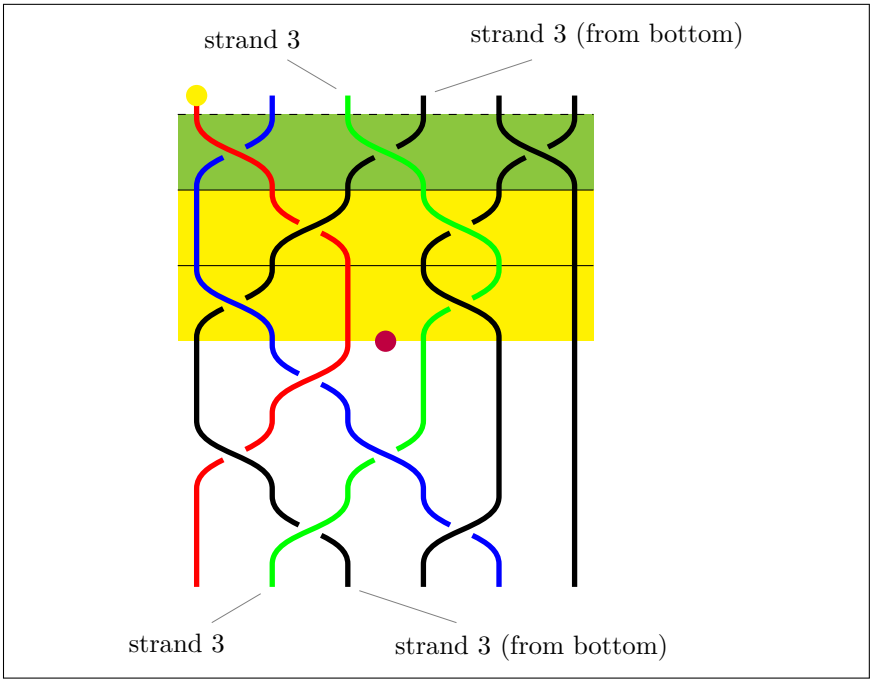
`style floors` `style all floors` In the spirit of separating *style* and *content*, the style options for the floors can be specified separately to the command (of course, they could be built in to the command). One advantage of this over building them in to the command is to allow them to be overridden for individual floors. The `style all floors` sets up options to be used for *all* floors, whilst the `style floors={n,m,...}{options}` sets up options to be used only for the listed floor. Anything specified in the `floor command` will take precedence over both of these.

Any other style options are passed to the underlying TikZ/PGF system and so may influence how the braid is drawn (but note that not all keys make sense due to the implementation).

3 Example

Here is a more detailed example.

```
\begin{center}
\begin{tikzpicture}
\braid[
  style all floors={fill=yellow},
  style floors={1}{dashed,fill=yellow!50!green},
  floor command={%
   \fill (\floorsx,\floorsy) rectangle
       (\floorex,\floorey);
   \draw (\floorsx,\floorsy) -- (\floorex,\floorsy);
  },
  line width=2pt,
  style strands={1}{red},
  style strands={2}{blue},
  style strands={3}{green}
] (braid) at (2,0) | s_1-s_3-s_5 | s_2^{-1}-s_4| s_1-s_4
    s_2^{-1} s_1-s_3 s_2^{-1}-s_4^{-1};
\fill[yellow] (2,0) circle (4pt);
\fill[purple] (braid) circle (4pt);
\node[at=(braid-3-s),pin=north west:strand 3] {};
\node[at=(braid-3-e),pin=south west:strand 3] {};
\node[at=(braid-rev-3-s),pin=north east:strand 3 (from
    bottom)] {};
\node[at=(braid-rev-3-e),pin=south east:strand 3 (from
    bottom)] {};
\end{tikzpicture}
\end{center}
```


strand 3
strand 3 (from bottom)
strand 3
strand 3 (from bottom)