strdiag – typesetting string diagrams

Daniel Gromada, 2024

The file `strdiag.tex` contains macros for typesetting string diagrams used in quantum groups or categorical quantum information. The macros are based on Ti*k*Z. So, to use `strdiag`, one has to install Ti*k*Z first. Then, one can write in plainTEX

```
\input tikz
\input partmac
```

For LATEXusers, I've prepared the file `strdiag.sty`, which includes PGF automatically, so it is sufficient to write

```
\usepackage{strdiag}
```

This is still work in progress, I am happy for any sort of comments and suggestions.

1 Basic usage

1.1 Predefined diagrams

The following macros are defined

`\cup`		`\cap`		`\Did`		`\Dcross`	
`\Rcap`		`\Lcap`		`\Fid`		`\Bid`	

1.2 Strings

The central macro of this package is `\Diagram{`⟨*data*⟩`}`, which essentailly opens a Ti*k*Z picture and sets the coordinates. You can use any Ti*k*Z macros you want inside ⟨*data*⟩. As an example, the `\cap` macro is defined by

```
\Diagram{
   \draw (1,0) .. controls +(up:0.5) and +(up:0.5) .. (2,0);
}
```

1.3 Boxes

There are two similar macros to typeset morphisms, namely `\Dmor{`⟨*shape*⟩`}`⟨*in*⟩`/`⟨*out*⟩ `(`⟨*x*⟩`,`⟨*y*⟩`)` and `\DMor{`⟨*shape*⟩`}[`⟨*in*⟩`/`⟨*out*⟩`] (`⟨*x*⟩`,`⟨*y*⟩`) {`⟨*label*⟩`}`.

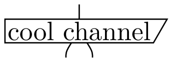
Here, ⟨*shape*⟩ can be one of

`circ`		`bcirc`		`square`	
`vec`		`covec`		`map`	
`mapC`		`mapT`		`mapA`	
`selfC`		`selfT`		`selfA`	
`selfCR`		`selfTR`		`selfAR`	

The data ⟨*x*⟩ and ⟨*y*⟩ are the coordinates of the box. The value $y = 0.5$ should correspond to the middle of the line.

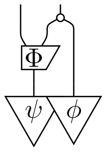
The data ⟨*in*⟩ and ⟨*out*⟩ can be numbers standing for the number of inputs and outputs. This part of declaration is optional. If you skip it, there will be no strings attached to the box. Using `\Dmor`, the endpoints of the inputs and outputs will be placed at distance $\Delta y = 0.5$ (if the box is placed at $y = 0.5$, then the inputs are at $y = 0$ and outputs at $y = 1$). The distance is $\Delta y = 1$ if you use `\DMor`. The inputs and outputs are spaced by the distance $\Delta x = 1$.

The extra parameter ⟨*label*⟩ of `\DMor` stands for a text that should be placed inside a box. So that in total, you can type

`\Diagram{\DMor{map}2/1 (0,0.5) {cool channel}}` for 

You can of course have more than one box in a diagram and you can combine them with simple lines drawn by the Ti*k*Z `\draw` command to obtain something like

```
\Diagram{
   \DMor{vec}0/1 (0,-1.5) {$\psi$}
   \DMor{vec}0/1 (1.5,-1.5) {$\phi$}
   \DMor{map}1/2 (0,.5)  {$\Phi$}
   \Dmor{circ}2/1 (1,2)
   \draw (1.5,-.5) - (1.5,1.5);
   \draw (-.5,1.5) - (-.5,2.5);
}
```



There is also a shorthand for drawing single spiders: `\spider{⟨in⟩/⟨out⟩}` stands for

`\Diagram{\Dmor{bcirc}⟨in⟩/⟨out⟩ (1,0.5)}`.

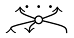
Similarly works `\wsipder` just using `circ` instead of `bcirc`.

So, for instance, `\spider{3/2}` creates .

1.4 Arrows

Considering the `\Dmor` and `\DMor` macros, one can replace ⟨*in*⟩/⟨*out*⟩ by `[⟨in⟩/⟨out⟩]`, where ⟨*in*⟩ and ⟨*out*⟩ are not numbers, but strings of characters from the following list

`>`	outgoing arrow
`<`	incoming arrow
`-`	a string without arrow
`0`	no string
`.`	three dots

In the same way, you can use `\spider[⟨in⟩/⟨out⟩]` or `\wsipder[⟨in⟩/⟨out⟩]`. So, for instance, you can write `\wspider[<>0-/>.>]` to obtain .

We also define new Ti*k*Z string styles `mid arrow` and `late arrow` to add arrows in the middle of a string. As an example, let us use the above to arrowise the example from the previous section. Note that if you want to avoid doubling the arrows within one string, you have to pay special attention and it becomes unfortunately rather less straightforward.

```
\Diagram{
   \DMor{vec}0/1 (0,-1.5) {$\psi$}
   \DMor{vec}0/1 (1.5,-1.5) {$\phi$}
   \DMor{map}1/2 (0,.5)  {$\Phi$}
   \Dmor{circ}2/1 (1,2)
   \draw (1.5,-.5) - (1.5,1.5);
   \draw (-.5,1.5) - (-.5,2.5);
}
```