Partmac – typesetting set partitions

Daniel Gromada, 2020

The file `partmac.tex` contains macros for typesetting set partitions, which are used for example in free probability or the theory of easy quantum groups. The macros use PGF (the ground layer of Ti*k*Z) to draw the partitions. So, to use `partmac`, one has to install PGF (or Ti*k*Z) first. Then, one can write in plainTEX

```
\input pgfcore
\input partmac
```

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

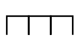
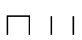
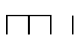
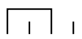
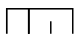
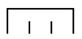
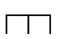
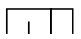
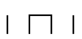
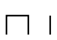
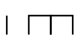
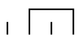
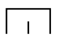
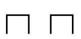
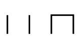
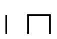
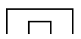
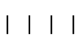
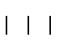
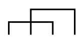
```
\usepackage{partmac}
```

The main aim of the package is to provide an easy to use tool for drawing set partitions. It provides an universal description of partitions in such a way that the result adjusts its proportions according to the context. So, one can use the same commands for drawing a partition in a title of a chapter, inside paragraph, or as an index of some mathematical symbol.

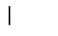
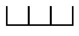
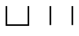
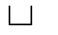
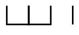
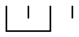
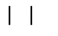
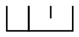
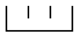
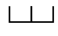
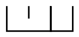
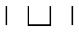
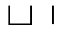
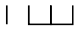
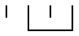
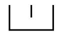
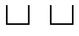
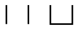
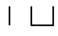
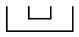
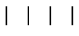
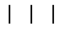
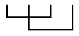
1 Basic usage

1.1 Predefined partitions

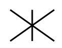
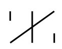
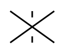
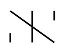
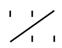
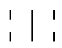
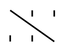
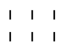
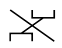
There are macros of the form `\Lxxxx` for partitions on lower line, `\Uxxxx` for partitions on upper line and `\Pxxxx` for partitions with the same ammount of upper and lower points. Here `xxxx` is the lexicographically smallest word representation of the partiiton. That is, we have the following macros for partitions with points on the lower line.

`\La`		`\Laaaa`		`\Laabc`	
`\Laa`		`\Laaab`		`\Labac`	
`\Lab`		`\Laaba`		`\Labca`	
`\Laaa`		`\Labaa`		`\Labbc`	
`\Laab`		`\Labbb`		`\Labcb`	
`\Laba`		`\Laabb`		`\Labcc`	
`\Labb`		`\Labba`		`\Labcd`	
`\Labc`		`\Labab`			

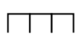
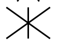
Then we have the following macros for partitions with points on the upper line.

`\Ua`		`\Uaaaa`		`\Uaabc`	
`\Uaa`		`\Uaaab`		`\Uabac`	
`\Uab`		`\Uaaba`		`\Uabca`	
`\Uaaa`		`\Uabaa`		`\Uabbc`	
`\Uaab`		`\Uabbb`		`\Uabcb`	
`\Uaba`		`\Uaabb`		`\Uabcc`	
`\Uabb`		`\Uabba`		`\Uabcd`	
`\Uabc`		`\Uabab`			

Finally partitions with equal number of points on lower and upper line.

`\Paa`		`\Pabab`		`\Pabcabc`	
`\Pab`		`\Paabc`		`\Pabcabd`	
`\Paaaa`		`\Pabac`		`\Pabcadc`	
`\Paaab`		`\Pabca`		`\Pabcdbc`	
`\Paaba`		`\Pabbc`		`\Pabcade`	
`\Pabaa`		`\Pabcb`		`\Pabcdbe`	
`\Pabbb`		`\Pabcc`		`\Pabcdec`	
`\Paabb`		`\Pabcd`		`\Pabcdef`	
`\Pabba`				`\Paabaab`	

In addition, we define the following synonyms.

`\singleton`		`\idpart`		`\fourpart`	
`\upsingleton`		`\disconnecterpart`		`\crosspart`	
`\pairpart`		`\positionerpart`		`\halflibpart`	
`\uppairpart`		`\connecterpart`			

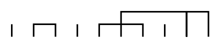
1.2 Partitions on one line

To define a general partition with points only on the lower or upper line, one can use macro `\Lpartition`, resp. `\Upartition`. The syntax is the following.

`\LPartition{`⟨*singletons*⟩`}{`⟨*remaining blocks*⟩`}`

The datum ⟨*singletons*⟩ should be of the form $h : i_1, i_2, \ldots, i_k$, where h is the height of the singleton blocks and $i_1, \ldots, i_k$ are the positions of the singletons. The datum ⟨*remaining blocks*⟩ consists of descriptions of other blocks. Each block is described similarly as the set of singletons, so in the format $h : i_1, \ldots, i_k$, where h is the height of the block and $i_1, \ldots, i_k$ are the positions of elements of the block. The data for the blocks are separated by semicolon.

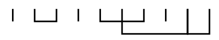
Let us show this on example.

`\LPartition{0.4:1,4,8}{0.4:2,3;0.4:5,7;0.8:6,9,10}` 

Here, the singletons are on position 1, 4, 8 and each of them is represented by a line of height 0.4. Then there are three additional blocks. First connecting points 2 and 3 is represented by a link of height 0.4 (that is, the same as the singletons). Second block connects points 5 and 7 and has the same height. Finally a block connecting points 6, 9, 10 has double height, that is, 0.8.

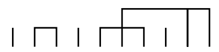
The macro `\UPartition` works the same. Except that instead of the height, we should put $1 -$ height, that is, we put there actually the y-coordinate of the point. So, to obtain the same result horizontally flipped, we have to write down

```
\UPartition{0.6:1,4,8}{0.6:2,3;0.6:5,7;0.2:6,9,10}
```



The units are chosen in such a way that one should keep the height between 0 and 1 to stick within the line in a paragraph. However, the macro works also if you put there higher numbers, which can be used especially in display mode. For example

```
\LPartition{0.6:1,4,8}{0.6:2,3;0.6:5,7;1.2:6,9,10}
```



1.3 General partitions

To draw general partitions with upper and lower points, one can use `\Partition{`$\langle data\rangle$`}`. The data can consist of the following commands.

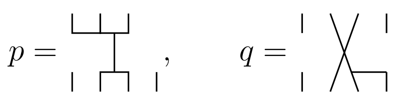
```
\Psingletons y1 to y2:i1,i2,...,ik     %draws singletons
\Pblock      y1 to y2:i1,i2,...,ik     %draws one block
\Pline       (x1,y1) (x2,y2)           %draws a line
```

Here, x_1 and x_2 represent the x coordinates (i.e. position of a point) and y_1 and y_2 the y-coordinates. Again, one is advised to keep the y coordinates between 0 and 1. As an example, we mention the definition of the connecter partition and the positioner partition .

```
\Partition{                   % connecter partition
\Pblock 0 to 0.3:1,2          % connecting two lower points
\Pblock 1 to 0.7:1,2          % connecting two upper points
\Pline (1.5,0.3) (1.5,0.7)    % connecting the two blocks together
}

\Partition{                  % positioner partition
\Psingletons 0to0.3:2        % singleton on lower line, pos. 2
\Psingletons 1to0.7:1        % singleton on upper line, pos. 1
\Pline (1,0) (2,1)           % line connecting lower pt 1 and upper pt 2
}
```

For more complicated partitions, one can use `\BigPartition{`$\langle data\rangle$`}`, which works exactly the same, but produces a larger result. Another difference is that `\BigPartition` aligns the middle of the partition, i.e. the point $y = 0.5$ with the equals sign. So, for example the result



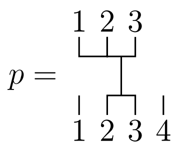
can be obtained writing

```
$$p=
\BigPartition{
\Pblock 0 to 0.25:2,3
\Pblock 1 to 0.75:1,2,3
\Psingletons 0 to 0.25:1,4
\Pline (2.5,0.25) (2.5,0.75)
},
\qquad
q=
\BigPartition{
\Psingletons 0 to 0.25:1,4
\Psingletons 1 to 0.75:1,4
\Pline (2,0) (3,1)
\Pline (3,0) (2,1)
\Pline (2.75,0.25) (4,0.25)
}$$
```

1.4 Adding text

To add text, one can use `\Ptext(`⟨*x*⟩`,`⟨*y*⟩`){`⟨*text*⟩`}`, where ⟨*x*⟩ and ⟨*y*⟩ are coordinates and ⟨*text*⟩ is any TEX code. The ⟨*text*⟩ is wrapped in a box, whose center is described by the coordinates. An example:

```
$$p=
\BigPartition{
\Pblock 0 to 0.25:2,3
\Pblock 1 to 0.75:1,2,3
\Psingletons 0 to 0.25:1,4
\Pline (2.5,0.25) (2.5,0.75)
\Ptext(1,1.2){1}
\Ptext(2,1.2){2}
\Ptext(3,1.2){3}
\Ptext(1,-0.2){1}
\Ptext(2,-0.2){2}
\Ptext(3,-0.2){3}
\Ptext(4,-0.2){4}
}$$
```



2 Coloring points

In this section, we describe how to assign different shapes to the set of partitioned points to obtain so-called colored partitions.

2.1 Bracket syntax

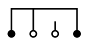
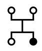
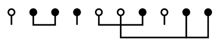
We start with coloring the points using white ∘ and black • circle. This can be achieved by encoding the partition coloring in brackets

`\Partition{...}[`⟨*coloring of the lower row*⟩`/`⟨*coloring of the upper row*⟩`]`
`\Partition{...}[`⟨*coloring of the lower row*⟩`]`

Here, ⟨*coloring of the lower/upper row*⟩ is a string made of letters `w` and `b` encoding the white and black circles. This notation can be used also for all the other macros for typesetting partitions like `\BigPartition`, `\LPartition`, `\UPartition`, `\Lxxxx`, `\Uxxxx`, even for user-defined macros.

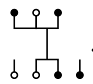
So, let us look on some examples:

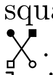
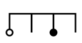
```
\Laaba[bwwwb]
\connecterpart[wb/ww]
\UPartition{0.6:1,4,8}{0.6:2,3;0.6:5,7;0.2:6,9,10}
    [/wbbbwwbwbb]
```

As mentioned above, even if we define some own partition macro such as[1]

```
\def\myfavouriteexample{
\BigPartition{
\Pblock 0 to 0.25:2,3
\Pblock 1 to 0.75:1,2,3
\Psingletons 0 to 0.25:1,4
\Pline (2.5,0.25) (2.5,0.75)
}}
```

Then we can just write `\myfavouriteexample[wwbb/bwb]` to obtain .

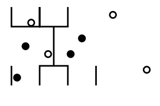
Besides black and white circles, you can also draw black and white square using letters `Q` and `q` instead of `b` and `w`. For example `\crosspart[qw/bQ]` gives . If you want to skip some point, write `0` in that place. For example `\fourpart[w0b0]` gives .

2.2 Finer placement of the points, additional shapes

The above described approach allows us to place the circles only to the points with coordinates $x = 1, 2, \ldots$ and $y = 0, 1$.[2] We may want to place the points also to different places. Here is how to do that.

First, we have the commands `\Pw(`$\langle x \rangle$`,`$\langle y \rangle$`)` and `\Pb(`$\langle x \rangle$`,`$\langle y \rangle$`)` that can be used inside `\Partition` or `\BigPartition`, so we can write

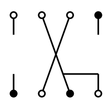
```
$$
\BigPartition{
\Pblock 0 to 0.25:2,3
\Pblock 1 to 0.75:1,2,3
\Psingletons 0 to 0.25:1,4
\Pline (2.5,0.25) (2.5,0.75)
\Pw (2.3,0.4) \Pw (4.6,0.9)
\Pw (1.7,0.8) \Pw (5.8,0.2)
\Pb (1.2,0.1) \Pb (3.5,0.6)
\Pb (1.5,0.5) \Pb (3.1,0.4)
}$$
```



Alternatively, one can use the construction `\Ppoint` $\langle y \rangle$ $\langle$*shape*$\rangle$`:`$\langle$*positions*$\rangle$, where $\langle$*shape*$\rangle$ is either `\Pw` or `\Pb`. For example,

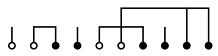
[1] If you are using LaTeX, you should maybe rather write `\newcommand{\myfavouriteexample}{...}`
[2] If you have the spirit of an experimenter, you may find out that we can actually reach any $y = 0, 1, 2, \ldots$

```
\BigPartition{
\Psingletons 0 to 0.25:1,4
\Psingletons 1 to 0.75:1,4
\Pline (2,0) (3,1)
\Pline (3,0) (2,1)
\Pline (2.75,0.25) (4,0.25)
\Ppoint0 \Pw:2,4
\Ppoint0 \Pb:1,3
\Ppoint1 \Pw:1,2,3
\Ppoint1 \Pb:4
}
```



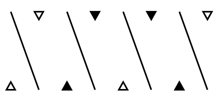
The commands `\Pw` and `\Pb` can also be used within `\LPartition` or `\UPartition` in the ⟨*singletons*⟩ parameter as follows:

```
\LPartition{0.6:1,4,8;\Pw:1,2,5,6;\Pb:3,4,7,8,9,10}
   {0.6:2,3;0.6:5,7;1.2:6,9,10}
```



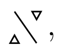
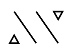
Finally, note that we have defined also a few other shapes of points that can be used in the same way as the commands `\Pw` and `\Pb`. First of all, we can use squares instead of circle simply by using the commands `\Pq` and `\PQ`. In addition, we define the triangle colors △, ▲ through the commands `\Lt`, `\LT` (lower triangle) and `\Ut`, `\UT` (upper triangle). The upper triangle are upside down. Example:

```
\BigPartition{
\Pline (1,1) (2,0)
\Pline (3,1) (4,0)
\Pline (5,1) (6,0)
\Pline (7,1) (8,0)
\Ppoint 0 \Lt:1,5
\Ppoint 0 \LT:3,7
\Ppoint 1 \Ut:2,8
\Ppoint 1 \UT:4,6
}
```

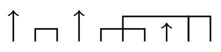


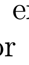
Actually we introduce some predefined partitions for working in the theory of *partitions with extra singletons* such as

`\singext` ▵, `\upsingext` ▿,

`\idext` , `\positionerext` , `\globcolext` 

Finally, we have the arrow colors represented by the command `\Ps`. Here, the syntax is slightly different: the command must, in addition, be followed by the height of the arrow. For instance, we can then write

`\LPartition{\Ps1:1,4;\Ps0.6:8}{0.4:2,3;0.4:5,7;0.8:6,9,10}` 

Also the command `\singleton` is not defined by `\uparrow` as one might expect, but using `\LPartition{\Ls1:1}{}`. This allows us to write `\singleton[w]` for  and `\singleton[b]` for .

2.3 Using color shapes in text

To write about the colors themself, we prepared the corresponding macros that just print the colored point.

`\wcol` ○, `\bcol` •, `\qcol` □, `\Qcol` ■, `\tcol` △, `\Tcol` ▲

3 Some tricks and modifications

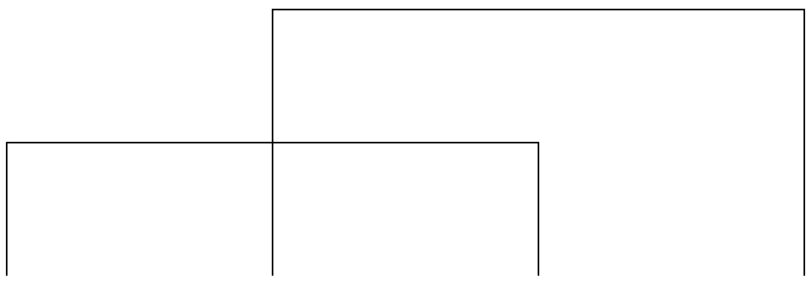
3.1 Size of the partition

The size and proportions of the partitions (that is, the coordinate system) are determined within the macro `\PartitionD` and `\BigPartitionD`. If you want to adjust those, just redefine the macro. Note that the size and the proportions depend on the context. In particular, it is different in text mode, in display mode, in subscripts and in subsubscripts.

Note also that the size of the partitions is defined relatively to the size of the font. So, if you increase the font size, then the partitions, such as ⌶, will also be large.

This is one way how to “ad hoc” enlarge a given partition. Another way is to use the macro `\PartitionA`⟨*depth*⟩⟨*y base length*⟩⟨*x base length*⟩, where ⟨*x/y base length*⟩ determines the length of the basis vectors, that is, the dinstance between 0 and 1. So, if you want to draw some gigantic partition, you can write something like

```
\PartitionA{0pt}{100pt}{100pt}{\Pblock 0to0.5:1,3 \Pblock 0to1:2,4}
```



3.2 Line thickness

The thickness of the lines in the partition is also determined relatively to the font size (cf. example above). The thickness is determined by the macro `\Pwidth` that contains a number (without units) that corresponds to the thickness in the unit `em`. By default, this is set to 0.05. So, when using 10pt font, the thickness is 0.5pt.

When typesetting in bold, one might want to increase this to roughly 0.08. Compare for example

pair partition ⊓ **pair partition** ⊓

In order to achieve that the thickness changes automatically when switching to the bold font, LaTeX users can write something like

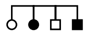
```
\expandafter\def\expandafter\bfseries
        \expandafter{\bfseries\def\Pwidth{0.08}}
```

somewhere to the beginning of the document. Nevertheless, I would actually not recommend doing so, because in the common LaTeX styles, `\textbf` does not change the shape of math to the bold. So, you might end up with something like **pair partition** $p =$ ⊓, which looks rather silly I would say. If you would like `\textbf` to change *everything* to bold, write the following instead.

```
\expandafter\def\expandafter\bfseries
        \expandafter{\bfseries\boldmath\def\Pwidth{0.08}}
```

3.3 Enlarging the points

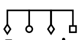
In a similar way, the size of the points can be adjusted. This is determined by the macro `\Pcolw`, which is by default set to 0.1. That is, the diameter of the circle and the edge of the square is by default 0.1em long. Changing this, we can enlarge the points.

For example, try `\def\Pcolw{0.15}\fourpart[wbqQ]`. You should obtain .

3.4 Defining new point shapes

One can define his own points using PGF commands. For example, let us define the dimond-shaped point

```
\def\Pd (#1,#2){
\pgftransformshift{\pgfpointxy{#1}{#2}}
{\pgfsetxvec{\pgfpoint{\Pcolw em}{0em}}
\pgfsetyvec{\pgfpoint{0em}{\Pcolw em}}
\pgfpathmoveto{\pgfpointxy{-1}{0}}
\pgfpathlineto{\pgfpointxy{0}{1.5}}
\pgfpathlineto{\pgfpointxy{1}{0}}
\pgfpathlineto{\pgfpointxy{0}{-1.5}}
\pgfpathclose
\pgfsetfillcolor{white}
\pgfusepath{stroke,fill}}
\pgftransformreset
}
```

Now, you can use this new point `\Pd` in the same way as any other of the ones discussed before. In addition, if you define your own command to be of the form `\P`x, where x is some single letter (here, $x =$ `d`), then you can also use the "bracket syntax". For instance, we can now write `\fourpart[dwdq]` and get .

Finally, we can define the command `\dcol` printing the diamond itself $\diamond$ by writting `\coldef\dcol\Pd`.

3.5 Using Phook

There is an empty macro `\Phook`, which is included at the beginning of every `\Partition` and `\BigPartition`. You can use it to change somehow the way the partitions are drawn. For instance, if you write

```
\def\Phook{\pgfsetstrokecolor{blue}}
```

then all the partitions you draw are going to be blue. See the next section for another application.

3.6 Drawing crossing strings to be interrupted

If you work with categories that have non-symmetric brainding, you may need to draw elements like or . In order to achieve that, we can define the following macro.

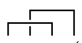
```
\def\addborder{
\pgfsetinnerlinewidth{\pgflinewidth}
\pgfsetlinewidth{4\pgflinewidth}
\pgfsetstrokecolor{white}
\pgfsetinnerstrokecolor{black}
}
```

Now, if you write `\addborder` in the beginning of the code inside `\Parition` or `\BigPartition`, every line will be typeset with an extra white border arround it. Then it depends on the order of the commands, which line or block will be the top one. The above two examples and were typeset using

```
\Partition{                       \Partition{
\addborder                        \addborder
\Pline (1,1)(2,0)                 \Pline (2,1)(1,0)
\Pline (2,1)(1,0)                 \Pline (1,1)(2,0)
}                                 }
```

Finally, if you want all partitions to be typeset in such a way, then you can simply write

```
\def\Phook{\addborder}
```

The command `\addborder` will then be included automatically. So, you can write `\crosspart` or `\Labab` to obtain or . It works well, but not perfect. If some block is composed using more than one command (such as `\Pblock` together with `\Pline`), then there will be some flaws. For instance, `\connecterpart` gives .

3.7 Using together with Ti*k*Z

Since this package is based on Ti*k*Z, any command you would use inside `\Partition` can also be used inside a Ti*k*Z picture. So, you can do stuff like

```
\tikzpicture
\Pblock 2to3:0,2,4
\Pblock 2to4:1,3,5
\draw[rounded corners=4pt]
(0,0) -- (0,1) -- (.5,1.5) -- (1,1)
    -- (1,0) -- (0,1) -- (1,1)
    -- (0,0) -- (1,0);
\shade[ball color=green]
    (3,.5) circle (.5cm);
\Ps-0.5 (3,-2)
\Pw (3,-2)
\endtikzpicture
```