The Permute Package

Version 0.12

Abstract

The permute package inputs, outputs and composes permutations. For example, `$\pmt{(123)}\circ\pmt{(321)} = \pmt{(123)(321)}$` produces $(123)\circ(132) = id$. A misleading example is $(a \dots z)\circ(z \dots a) = id$ printed via `$(a\ldots z)\circ(z\ldots a)=\pmt{(a\ldots z)(z\ldots a)}$` — misleading since the package doesn't care about the human interpretation of "...".

1 User's guide

1.1 Software license

`permute.dtx` and `permute.ins` and all files generated from these files are referred to as 'the permute package'. It is distributed under the terms of the LaTeX Project Public License from CTAN archives in directory `macros/latex/base/lppl.txt`. Either version 1.0 or, at your option, any later version. The use of the package is completely free.

Permission is granted to modify the permute package. You are not allowed to distribute any changed version of the package, neither under the same name nor under a different one.

Send comments, ideas and bug reports via electronic mail to `cheinz@gmx.de`.

1.2 Installation

1. Following the TeX directory structure (TDS) you should put the files of the package into directories as follows:

`permute.dvi`	→	`texmf/doc/latex/permute`
`permute.dtx`, `permute.ins`	→	`texmf/source/latex/permute`

   Of course, you need not to use the TDS. Simply adjust the directories below.

2. Create the directory `texmf/tex/latex/permute`.

3. Change the working directory to `texmf/source/latex/permute` and run `permute.ins` through TeX.

4. Move the generated file to `texmf/tex/latex/permute` if this is not already done.

5. If your TEX implementation uses a filename database, update it.

1.3 Input and output formats

General notation. First we restrict ourselves to $S_1, \ldots, S_9$:

$$S_n := \{f : \{1, \ldots, n\} \to \{1, \ldots, n\} \mid f \text{ is one-to-one and onto}\}.$$

A permutation $f \in S_n$ can be written **verbose** as an explicit sequence of pre-image/image pairs like this:

$$f = \begin{pmatrix} 1 & 2 & \ldots & n \\ f(1) & f(2) & \ldots & f(n) \end{pmatrix}.$$

A permutation σ is a **cycle** and written $\sigma = (x_1 x_2 \ldots x_k)$ if and only if there are distinct numbers $x_1, x_2, \ldots, x_k \in \{1, \ldots, n\}$ satisfying

$$\begin{aligned} \sigma(x_i) &= x_{i+1} && \text{for all } 1 \le i < k \\ \sigma(x_k) &= x_1 \\ \sigma(x) &= x && \text{for all } x \in \{1, \ldots, n\} \setminus \{x_1, \ldots, x_k\}. \end{aligned}$$

This means that a cycle $(x_1 x_2 \ldots x_k)$ maps x_1 to x_2, x_2 to x_3, $\ldots$, x_k back to x_1, and fixes all other elements. Each permutation in S_n can be written as a composition of cycles, for example $\left(\begin{smallmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 5 & 3 & 4 \end{smallmatrix}\right) = (12) \circ (354) = (12)(354)$—we leave out the "$\circ$". Note that S_n is not commutative in general and that we compose from the right to the left. In the example 3 is mapped to 5 and then 5 stays 5 since unchanged by (12).

Input formats. If you want to enter a permutation cycle based, just write the cycles after each other. `(12)(354)` would be legal; there must not be a `\circ` in between. The verbose input format lists all pre-image/image pairs without any separators, but a space is allowed in between. The permutation above could also be entered as `12 21 35 43 54`. The package distinguishes the two formats by looking at the first token: If and only if it's a left parenthesis, the package accepts cycles.

Now we drop the restriction $n \le 9$ and the limitation of permuting numbers; we can do it with nearly arbitrary (token) strings. To enter a string as pre-image, image or inside a cycle, enclose the string in braces. That's all! For example, I typed

```
1{f(1)} 2{f(2)} \ldots\ldots n{f(n)}
```

for $\left(\begin{smallmatrix} 1 & 2 & \ldots & n \\ f(1) & f(2) & \ldots & f(n) \end{smallmatrix}\right)$, which actually isn't a permutation at all.

In the sequel $\langle pmt \rangle$ means either a verbose sequence or a sequence of cycles. *Do not use* `\empty` *or* `\relax` *or equivalent definitions inside* $\langle pmt \rangle$.

Output formats. The package provides a cycle based and two verbose output formats. Some of the commands described below have an optional ⟨*print order*⟩ argument. Now let ⟨*print order*⟩ equal $n_1 n_2 \dots n_k$ where each n_i is a single token or a braced string. If n_i and n_{i+1} both appear in a permutation and appear in different cycles, then the cycle containing n_i is printed first. Moreover the cycle starts with the element n_i. Cycles not covered by ⟨*print order*⟩ are printed as they appear in the internal data format. Some examples on printing (12)(34)(56):

⟨*print order*⟩	results in	⟨*print order*⟩	results in
no order	(12)(34)(56)	empty	(56)(34)(12)
`2`	(21)(56)(34)	`23`	(21)(34)(56)
`6`	(65)(34)(12)		

All results represent the same permutation. Note the difference between 'no order' and 'empty': The package uses a standard order if you don't request a special one.

There is some danger if you want to use a *single token string* as ⟨*print order*⟩. In this case you must enclose the string in *two level* of braces. Use ⟨*print order*⟩=`{{one}}` to control `({one}{two})({two}{three})({three}{four})`, for example. Since TEX discards one group level, `{one}` would lead to the order `o`, `n`, `e`. However, ⟨*print order*⟩=`{one}{two}` needs no extra braces.

For the verbose output formats, ⟨*print order*⟩ plays the role of domain. The package uses exactly the elements and order, i.e. all pre-images not appearing in ⟨*print order*⟩ are not printed, and we assume image=pre-image if pre-image appears in ⟨*print order*⟩ but not in the permutation. Some examples on printing `1{f(1)} 2{f(2)} \ldots\ldots n{f(n)}`:

⟨*print order*⟩	results in
no order	$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & a & b & c & d & e & f & g & h & i \\ f(1) & f(2) & 3 & 4 & 5 & 6 & 7 & 8 & 9 & a & b & c & d & e & f & g & h & i \end{pmatrix}$
`n\ldots 21`	$\begin{pmatrix} n & \dots & 2 & 1 \\ f(n) & \dots & f(2) & f(1) \end{pmatrix}$
`2\ldots n{n+1}`	$\begin{pmatrix} 2 & \dots & n & n+1 \\ f(2) & \dots & f(n) & n+1 \end{pmatrix}$
empty	()

The first example doesn't print '…' and '$f(n)$' since the pre-images '…' and 'n' don't appear in the standard printing order (which is the domain here). But it shows image=pre-image pairs not in the permutation since the standard domain defines the elements as pre-images.

Note: (a) The (full) verbose format uses math mode and the TEX-primitive `\atop`. The latter causes a warning if used together with `amsmath.sty`. (b) The package defines a short verbose output format, too. It prints the row of images only. Don't take it for the cycle based format!

Finally, some commands also have a star-form which separate the output like this: (1 2 3 4). It is useful if you use strings instead of numbers, for example the permutation (one two three four) is printed with a `*`-command.

1.4 User commands

`\pmt`[`*`][`[`⟨*print order*⟩`]`]`{`⟨*pmt*⟩`}`

calculates the composition of the cycles (if any) and prints the permutation: `\pmt{(12)(23)(34)}` prints (1234) and `\pmt{12 23 34 41}` gives (1234). Note that this command outputs cycles only. Some examples:

`\pmt{(12)(34)(56)}`	prints (12)(34)(56)
`\pmt*[2]{(12)(34)(56)}`	prints (2 1)(5 6)(3 4)
`\pmt[23]{(12)(34)(56)}`	prints (21)(34)(56)
`\pmt*[6]{(12)(34)(56)}`	prints (6 5)(3 4)(1 2)

The macro `\pmtprintorder` contains the standard printing order, see 1.5.

`\pmtv`[`*`][`[`⟨*print order*⟩`]`]`{`⟨*pmt*⟩`}`

prints a verbose form of the permutation. The commands `\pmtvshorttrue` and `\pmtvshortfalse` controls whether the package prints only the row of images or the full pre-image/image format.

`\pmttable`[`*`][`[`⟨*print order*⟩`]`]`{`⟨*list of pmts*⟩`}{`⟨*list of pmts*⟩`}`

`\pmtvtable`[`*`][`[`⟨*print order*⟩`]`]`{`⟨*list of pmts*⟩`}{`⟨*list of pmts*⟩`}`

The commands compose each σ_1 of the first list with each σ_2 of the second list and write the result $\sigma_1 \circ \sigma_2$ in row σ_1 and column σ_2. For example,

```
$$\pmttable{(),(12),(13),(23),(123),(132)}
           {(),(12),(13),(23),(123),(132)}$$
```

creates

∘	id	(12)	(13)	(23)	(123)	(132)
id	id	(12)	(13)	(23)	(123)	(132)
(12)	(12)	id	(132)	(123)	(23)	(13)
(13)	(13)	(123)	id	(132)	(12)	(23)
(23)	(23)	(132)	(123)	id	(13)	(12)
(123)	(123)	(13)	(23)	(12)	(132)	id
(132)	(132)	(23)	(12)	(13)	id	(123)

'()' stands for the identity map here. *Do not write* `\pmttable{,(12),...`. You may write `\pmttable{(12),(123)}{(),(23),(123),(132)}` to typeset a piece of the table or use `\pmtvtable` to print all permutations verbose. The optional arguments effect all printed permutations.

If you create really big tables like the one of S_5, you surely want to cut the whole table in pieces and produce subtables on different pages. This leads to alignment problems since the first column on the first page need not to have the width of the first column on the second page. Bad luck!

Now we discuss how to calculate with the permute package. Let ⟨*current pmt*⟩ denote the (internal) current permutation and ⟨*name*⟩ another internal (stored) permutation.

`\pmtload{`⟨*name*⟩`}`	⟨*current pmt*⟩ ← ⟨*name*⟩
`\pmtsave{`⟨*name*⟩`}`	⟨*name*⟩ ← ⟨*current pmt*⟩
`\pmtid[[`⟨*name*⟩`]]`	⟨*current pmt*⟩ ← identity map
`\pmtdo[[`⟨*name*⟩`]]{`⟨*pmt*⟩`}`	⟨*current pmt*⟩ ← ⟨*pmt*⟩ ∘ ⟨*current pmt*⟩
`\pmtcirc[[`⟨*name*⟩`]]{`⟨*pmt*⟩`}`	⟨*current pmt*⟩ ← ⟨*current pmt*⟩ ∘ ⟨*pmt*⟩
`\pmtprint[*][[`⟨*print order*⟩`]]`	prints ⟨*current pmt*⟩.
`\pmtvprint[*][[`⟨*print order*⟩`]]`	prints ⟨*current pmt*⟩.

`\pmtimageof[[`⟨*name*⟩`]]{`⟨*pre-image*⟩`}`

prints image of ⟨*pre-image*⟩ under ⟨*current pmt*⟩.

`\pmtpreimageof[[`⟨*name*⟩`]]{`⟨*image*⟩`}`

prints pre-image of ⟨*image*⟩ under ⟨*current pmt*⟩.

If you use any optional `[`⟨*name*⟩`]`, this permutation replaces ⟨*current pmt*⟩. For example, `\pmtid[a]` makes the permutation a to be the identity map or `\pmtdo[a]{`⟨*pmt*⟩`}` performs $a \leftarrow \langle pmt \rangle \circ a$.

Finally examples which all print the result of $\begin{pmatrix} 1 & \dots & k & l & \dots & n \\ f(1) & \dots & f(k) & f(l) & \dots & f(n) \end{pmatrix} \circ (kl)$, namely $\begin{pmatrix} 1 & \dots & k & l & \dots & n \\ f(1) & \dots & f(l) & f(k) & \dots & f(n) \end{pmatrix}$.

```
\pmtid
\pmtdo{1{f(1)} \ldots\ldots k{f(k)} l{f(l)} \ldots\ldots n{f(n)}}
\pmtcirc{(kl)}
\pmtvprint[1\ldots kl\ldots n]

\pmtid
\pmtdo{(kl)}
\pmtdo{1{f(1)} k{f(k)} l{f(l)} n{f(n)}}
\pmtvprint[1\ldots kl\ldots n]
```

We can drop the two `\ldots` pairs (also in the first example) since image and pre-image are equal.

1.5 Parameters

`\pmtprintorder`

contains the standard printing order which is used whenever you leave out or forget the optional ⟨*print order*⟩ argument. By default it contains the sequence `123456789abcdefghi`. You may adjust it to your needs, for example, `\renewcommand*\pmtprintorder{123456}` if you work with S_6. This is especially good if you use the verbose printing format. You can forget the optional ⟨*print order*⟩ in this case:

`\renewcommand*\pmtprintorder{123456}`

`\pmtv{()}`	prints $\left(\begin{smallmatrix}1&2&3&4&5&6\\1&2&3&4&5&6\end{smallmatrix}\right)$
`\pmtv{(123)}`	prints $\left(\begin{smallmatrix}1&2&3&4&5&6\\2&3&1&4&5&6\end{smallmatrix}\right)$
`\pmtv{(12)(23)(16)(34)(45)}`	prints $\left(\begin{smallmatrix}1&2&3&4&5&6\\6&3&4&5&1&2\end{smallmatrix}\right)$

`\pmtseparator`

contains the separator used for the optional star. By default it is a space: `\newcommand*\pmtseparator{ }`. Since it's not a backslashed space `\␣`, it is ignored in math mode, in particular in the verbose output format. But you may write `\renewcommand*\pmtseparator{\ }`.

`\pmtidname`

contains the string which is printed in the cycle based format if a permutation is the identity map. It is predefined via `\newcommand*\pmtidname{id}`.

`\pmtldelim`

`\pmtrdelim`

contain the left respectively right delimiter for the verbose format. `\left(` and `\right)` are used by default.

`\pmttableborders`

contains `lt` which makes a left border column and a top border line. You may redefine it to be empty or to contain `l`, `t`, `lt` or `tl`.

`\pmtarraystretch`

contains `2` and is used as `\arraystretch` for the table. You may write `\renewcommand*\pmtarraystretch{1.3}` and get more compact tables:

$\circ$	id	(12)	(23)
id	id	(12)	(23)
(12)	(12)	id	(123)
(23)	(23)	(132)	id