



eolang: \LaTeX Package for Formulas and Graphs of EO Programming Language and φ -calculus*

Yegor Bugayenko
yegor256@gmail.com

2024-01-11, 0.18.0

NB! You must run \TeX processor with `-shell-escape` option and you must have [Perl](#) installed. If you omit the `-shell-escape` option, the package will try to use cached files, if they exist. If they don't, compilation will fail. Thus, when you must prepare your document for a compilation without the `-shell-escape` option, run it locally with the option provided and then package all files (including the files in the `_eolang-*` directories) into a single ZIP archive. It is advised to use `tmpdir` package option in this case, in order to make the directory name not depend on the \TeX engine.

If `-shell-escape` is set, this package won't work on Windows, because it uses POSIX command line interface.

1 Introduction

This package helps you print formulas of φ -calculus, which is a formal foundation of [EO](#) programming language. The calculus was introduced by Bugayenko (2021) and later formalized by Kudasov et al. (2022). Here is how you render a simple expression:

*The sources are in GitHub at [objectionary/eolang.sty](https://github.com/objectionary/eolang.sty)

$\begin{aligned} \text{app} &\mapsto \llbracket \\ &\quad \rho \mapsto \xi.b.^2, \alpha_0 t \rightsquigarrow \text{TRUE}, \\ &\quad b \mapsto \llbracket \alpha_* \mapsto \Phi.\text{fn}(56), \\ &\quad \quad \varphi \mapsto \dot{\Phi}.\text{string.trim}(\xi), \\ &\quad \quad \Delta \mapsto 01\text{-FE-C3} \rrbracket, \\ &\quad x \mapsto \llbracket \lambda \mapsto \emptyset \rrbracket. \end{aligned}$	<pre> 1 \documentclass{minimal} 2 \usepackage{eolang} 3 \begin{document} 4 \begin{phiquestion*} 5 app -> [[% it's abstract! 6 ^ !-> \$.b.^{~2}, 0/t~> TRUE, 7 b -> [[*-> Q.fn(56), 8 @ -> QQ.string.trim(\$), 9 D> 01-FE-C3]]],\ 10 x -> [[\lambda .> ?]]. 11 \end{phiquestion*} 12 \end{document} </pre>
--	---

`phiquestion (env.)` The environment `phiquestion` lets you write a φ -calculus expressions using simple plain-text notation, where:

- “@” maps to “ φ ” (`\varphi`),
- “^” maps to “ ρ ” (`\rho`),
- “\$” maps to “ ξ ” (`\xi`),
- “&” maps to “ σ ” (`\sigma`),
- “?” maps to “ \emptyset ” (`\varnothing`),
- “Q” maps to “ Φ ” (`\Phi`),
- “QQ” maps to “ $\dot{\Phi}$ ” (`\dot{\Phi}`),
- “->” maps to “ \mapsto ” (`\mapsto`),
- “~>” maps to “ \rightsquigarrow ” (`\rightsquigarrow`),
- “!->” maps to “ \mapsto ” (`\mapsto`),
- “.>” maps to “ \mapsto ” (`\mapsto`),
- “D>” maps to “ $\Delta \mapsto$ ” (`\Delta \mapsto`),
- “L>” maps to “ $\lambda \mapsto$ ” (`\lambda \mapsto`),
- “[[” maps to “ \llbracket ” (`\llbracket`),
- “]]” maps to “ \rrbracket ” (`\rrbracket`),
- “==” maps to “ \equiv ” (`\equiv`),
- “|abc|” maps to “ abc ” (`\text{abc}`).

Also, a few symbols are supported for φ PU architecture:

- “<<” maps to “ \langle ” (`\langle`),
- “>>” maps to “ \rangle ” (`\rangle`),
- “-abc>” maps to “ $\xrightarrow{\text{abc}}$ ” (`\xrightarrow{\text{abc}}`),
- “:=” maps to “ \vDash ” (`\vDash`).

Before any arrow you can put a number, which will be rendered as α with an index, for example `\phiq{0->x}` will render “ $\alpha_0 \mapsto x$ ”. Instead of a number you can use asterisk too.

You can append a slash and a title to the number of an attribute, such as $0/g \mapsto x$. this will render as $\alpha_0|g \mapsto x$. You can use fixed-width words too, for example $\backslash\mathrm{phiq}\{0/|f| \mapsto x\}$ will render as “ $\alpha_0|f \mapsto x$ ”. It’s also possible to use an asterix instead of a number, such that $\backslash\mathrm{phiq}\{*/g \mapsto x\}$ renders as “ $\alpha_*|g \mapsto x$ ”

Numbers are automatically converted to fixed-width font, no need to always decorate them with vertical bars.

TRUE and FALSE are automatically converted to fixed-width font too.

Object names are automatically converted to fixed-width font too, if they have more than one letter.

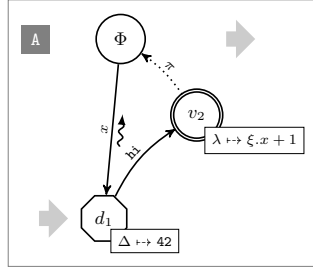
Texts in double quotes are automatically converted to fixed-width font too.

$\backslash\mathrm{phiq}$ The command $\backslash\mathrm{phiq}$ lets you inline a φ -calculus expressions using the same simple plain-text notation. You can use dollar sign directly too:

A simple object $x \mapsto \llbracket \varphi \mapsto y \rrbracket$ is a decorator of the data object $y \mapsto \llbracket \Delta \mapsto 42 \rrbracket$.

```
4 \begin{document}
5 A simple object
6 \mathrm{phiq}\{x \rightarrow \llbracket @ \rightarrow y \rrbracket\} \backslash
7 is a decorator of
8 the data object \backslash
9 $y \rightarrow \llbracket \Delta \mapsto 42 \rrbracket$.
10 \end{document}
```

$\mathrm{sodg} (env)$ The environment sodg allows you to draw a [SODG](#) graph:



```
1 \documentclass{standalone}
2 \usepackage{eolang}
3 \begin{document}
4 \begin{sodg}
5 v0 \backslash v0==> \backslash v0!!A
6 v1 xy:v0,-.8,2.8 data:42 tag:d_1
7 v0->v1 a:x rho \backslash =>v1
8 v2 xy:v0,+1,+1 atom:\xi.x+1
9 v1->v2 a:|hi| bend:-15
10 v2->v0 pi bend:10 % a comment
11 \end{sodg}
12 \end{document}
```

The content of the environment is parsed line by line. Markers in each line are separated by a single space. The first marker is either a unique name of a vertex, like “ $v1$ ” in the example above, or an edge, like “ $v0 \rightarrow v1$.” All other markers are either unary like “ ρ ” or binary like “ $\text{atom}:\xi.x+1$.” Binary markers have two parts, separated by colon.

The following markers are supported for a vertex:

- “ $\text{tag}:\langle\text{math}\rangle$ ” puts a custom label $\langle\text{math}\rangle$ into the circle;
- “ $\text{data}:[\langle\text{box}\rangle]$ ” makes it a data vertex with an optional attached “ $\langle\text{box}\rangle$ ” (the content of the box may only be numeric data);
- “ $\text{atom}:[\langle\text{box}\rangle]$ ” makes it an atom with an optional attached “ $\langle\text{box}\rangle$ ” (the content of the box is a math formula);
- “ $\text{box}:\langle\text{txt}\rangle$ ” attaches a “ $\langle\text{box}\rangle$ ” to it;

- “xy:<v>,<r>,<d>” places this vertex in a position relative to the vertex “<v>,” shifting it right by “<r>” and down by “<d>” centimetres;
- “+:<v>” makes a copy of an existing vertex and all its kids;
- “edgeless” removes the border from the vertex;
- “style:{...}” adds this TikZ style to the vertex \node.

The following markers are supported for an edge:

- “rho” places a backward snake arrow to the edge,
- “bend:<angle>” bend it right by the amount of “<angle>,”
- “a:<txt>” attaches label “<txt>” to it,
- “pi” makes it dotted, with π label;
- “style:{...}” adds this TikZ style to the edge \path.

It is also possible to put transformation arrows to the graph, with the help of “v0=>v1” syntax. The arrow will be placed exactly between two vertices. You can also put an arrow from a vertex to the right, saying for example “v3=>”, of from the left to the vertex, by saying for example “=>v5.” If you want the arrow to stay further away from the vertex than usually, use a few “=” symbols, for example “===>v0.”

You can also put a marker at the left side of a vertex, using “v5!A” syntax, where “v5” is the vertex and “A” is the text in the marker. They are useful when you put a few graphs on a picture explaining how one graph is transformed to another one and so forth. You can make a distance between the vertex and the marker a bit larger by using a few exclamation marks, for example “v5!!!A” will make a distance three times bigger.

You can make a clone of an existing vertex together with all its dependants, by using this syntax: “v0+a.” Here, we make a copy of “v0” and call it “v0a.” See the example below.

Be aware, unrecognized markers are simply ignored, without any error reporting.

`\eolang` There is also a no-argument command `\eolang` to help you print the name of EO
`\phic` language. It understands the anonymous package option and prints itself differently, to
`\xmirl` double-blind your paper. There is also `\phic` command to print the name of φ -calculus,
also sensitive to anonymous mode. The macro `\xmirl` prints “XMIR”.

In our research we use XYZ,
an experimental object-oriented
dataflow language, α -calculus, as its
formal foundation, and XML⁺ —
its XML-based presentation.

```

3 \usepackage[anonymous]{eolang}
4 \begin{document}
5 In our research we use \eolang{ }, \
6 an experimental object-oriented \
7 dataflow language, \phic{ }, as its \
8 formal foundation, and \xmirl{ } --- \
9 its XML-based presentation.
10 \end{document}

```

Without the anonymous option there will be no orange color:

In our research we use EO,
an experimental object-oriented
dataflow language, φ -calculus, as its
formal foundation, and XMIR —
its XML-based presentation.

```

3 \usepackage{eolang}
4 \begin{document}
5 In our research we use \eolang{}, \\
6 an experimental object-oriented \\
7 dataflow language, \phic{}, as its \\
8 formal foundation, and \xmir{} --- \\
9 its XML-based presentation.
10 \end{document}

```

`\phiConst` A few simple commands are defined to help you render arrows. It is recommended
`\phiWave` not to use them directly, but use `!->` instead. However, if you want to use `\phiConst`,
`\phiDotted` wrap it in `\mathrel` for better display:

If x is an identifier and y is an object,
then $x \# \rightarrow y$ makes y a constant,
 $x \rightsquigarrow y$ makes it a decoratee
of an arbitrary number of objects,
while $x \vdash \rightarrow y$ makes it a special
attribute.

```

6 If $x$ is an identifier and $y$ is
7 an object, then $x \phiConst y$
8 makes $y$ a constant,
9 $x \phiWave y$ makes it a decoratee
10 of an arbitrary number of objects,
11 while $x \phiDotted y$ makes it
12 a special attribute.

```

`\phiOset` If you want to put a text over an arrow or under it, use `\phiOset` and `\phiUset`
`\phiUset` respectively:

When the names of attributes and
their values don't matter, we use
an arrow with a star, for example:

$$[[\vdash \rightarrow]]$$

```

6 When the names of attributes and their
7 values don't matter, we use an arrow
8 with a star, for example:
9 \begin{phiquestion*}
10 [[ \phiOset{*}{->} ]]
11 \end{phiquestion*}

```

`\phiMany` Sometimes you may need to simplify the way you describe an object (the typesetting
is a bit off, but this is not because of us, but because of [this](#)):

The expression $[\alpha_1 \mapsto x_1, \alpha_2 \mapsto x_2, \dots, \alpha_n \mapsto x_n]$
and expression $[\alpha_i \mapsto_{i=1}^n x_i]$ are
syntactically different but semanti-
cally equivalent.

```

6 The expression
7 \phiiq{[[ 1-> x_1,
8 2-> x_2, \dots,
9 \alpha_n -> x_n ]]}
10 and expression
11 \phiiq{[[ \alpha_i
12 \phiMany{->}{i=1}{n} x_i ]]}
13 are syntactically different but
14 semantically equivalent.

```

`\phiSaveTo` If you want to use `phiquestion` or `sodg` environments inside `tabular` or any other
`\sodgSaveTo` environment or command, you won't be able to do this, because `phiquestion` and `sodg`
are “verbatim” environments. `\phiSaveTo` and `\sodgSaveTo` commands will help you
in this situation. You use them right before `\begin{phiquestion}` or `\begin{sodg}`
respectively — the content of the equation or the graph won't be rendered, but instead
saved to the file. Later, inside `tabular`, you can use it through the `\input` macro (don't
forget the `\parbox`):

Free:	$\llbracket x \mapsto \emptyset \rrbracket$
Bound:	$\llbracket x \mapsto \llbracket \Delta \mapsto 42 \rrbracket \rrbracket$

```

5 \phiSaveTo{a}
6 \begin{phiuation*}
7 [[ x -> [[D>42]] ]]
8 \end{phiuation*}
9 \begin{tabular}{p{.5in}l}
10 Free: & $\llbracket x \mapsto ? \rrbracket$ \\
11 Bound: & \parbox{1in}{\input{a}} \\
12 \end{tabular}

```

`\eoAnon` You may want to hide some of the content with the help of the anonymous package option. The command `\eoAnon` may help you with this. It has two parameters: one mandatory and one optional. The mandatory one is the content you want to show and the optional one is the substitution we will render if the anonymous package option is set.

2 Package Options

`tmpdir` The default location of temp files is `_eolang`. You can change this with the help of the `tmpdir` package option:

```
\usepackage[tmpdir=/tmp/foo]{eolang}
```

`nodollar` You may disable the special treatment of the dollar sign by using the `nodollar` package option:

```
\usepackage[nodollar]{eolang}
```

`anonymous` You may anonymize `\eolang`, `\XMIR`, and `\phic` commands by using anonymous package option (they all use the `\eoAnon` command mentioned earlier):

```
\usepackage[anonymous]{eolang}
```

`noshell` You may prohibit any interactions with the shell by using the `noshell` option. This may be helpful when you send your document for outside processing and want to make sure the compilation won't break due to shell errors:

```
\usepackage[noshell]{eolang}
```

3 More Examples

The `phiuation` environment treats ends of line as signals to start new lines in the formula. If you don't want this to happen and want to parse the next line as the a continuation of the current line, you can use a single backslash as it's done here:

$$\frac{x \mapsto \llbracket \varphi \mapsto y \rrbracket \quad y \mapsto \llbracket z \mapsto 42 \rrbracket}{x.z \mapsto 42} R1$$

```

6 \begin{phiuation*}
7 \dfrac \
8 {x->[[@->y]] \quad y->[[z->42]]} \
9 {x.z -> 42} \
10 \text{\sffamily R1}
11 \end{phiuation*}

```

This is how you can use `\dfrac` from [amsmath](#) for large inference rules, with the help of `\begin{split}` and `\end{split}`:

$$\frac{x \mapsto \llbracket \varphi \mapsto y, z \mapsto 42, \alpha_0 \mid g \mapsto \emptyset, \alpha_1 \mid \text{foo} \mapsto 42 \rrbracket}{x \mapsto \llbracket \varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto \llbracket \psi \rightsquigarrow \text{hello}(12) \rrbracket, \alpha_1 \mapsto 42) \rrbracket} \text{R2.}$$

```

6 \begin{phiuation*}
7 \dfrac{\begin{split}
8 x->[[@->y, z->42,
9 0/g->?, 1/foo->42]]
10 \end{split}}{\begin{split}
11 x->[[@->y, z->?, f ~> |pi|(
12 0->[[ \psi !-> |hello|(12) ]],
13 1->42)]]
14 \end{split}}\text{R2}.
15 \end{phiuation*}

```

You can use the `matrix` environment too, in order to group a few lines:

$$\text{foo} \mapsto \left\{ \begin{array}{c} \emptyset \\ \llbracket \lambda \mapsto \rho \times \xi. \alpha_0 \rrbracket \\ \llbracket \Delta \mapsto 42 \rrbracket \end{array} \right\}$$

```

5 \begin{phiuation*}
6 foo -> \left\{\begin{matrix} \backslash
7 ? \backslash\backslash
8 [[ L> ~ \times $. \alpha_0 ]] \backslash
9 [[ D> 42 ]] \backslash
10 \end{matrix}\right\}
11 \end{phiuation*}

```

The `cases` environment works too:

$$\beta \models \begin{cases} [v_2, \varphi \xrightarrow{\text{DTZD}} 42] \\ [v_{33}] \end{cases}$$

```

5 \begin{phiuation*}
6 \beta := \begin{cases} \backslash
7 [ v_2, @ -dtzd> 42 ] \backslash\backslash
8 [ v_{33} ] \backslash
9 \end{cases}
10 \end{phiuation*}
11 \end{document}

```

The `phiuation` environment may be used together with the [acmart](#) package:

$$\begin{array}{l} x \mapsto \llbracket \\ \quad y \mapsto \llbracket \\ \quad \quad z \rightsquigarrow \xi, f \mapsto \emptyset \rrbracket \rrbracket, \\ \beta_1 \models [\psi \xrightarrow{\text{WAIT}} \emptyset]. \end{array}$$

```

1 \documentclass{acmart}
2 \usepackage{eolang}
3 \thispagestyle{empty}
4 \begin{document}
5 \begin{phiuation*}
6 x -> [[
7   y -> [[
8     z !-> $, f ..> ? ]]]],\backslash
9 \beta_1 := [ \psi -wait> ? ].
10 \end{phiuation*}
11 \end{document}

```

It's possible to use `\label` inside the `phiuation` environment (pay attention to how you can disable our custom parsing of math formulas by means of curled brackets around the “4” number):

Discriminant can be calculated using the following simple formula:

$$D = b^2 - 4ac. \quad (1)$$

Eq. 1 is also widely used in number theory and polynomial factoring.

```
6 Discriminant can be calculated using
7 the following simple formula:
8 \begin{phiuation}
9 D = b{^2} - {4}ac.
10 \label{d}
11 \end{phiuation}
12 Eq.\ref{d} is also widely used in
13 number theory and polynomial factoring.
```

You can add comments to your equations, using the `&&` command (pay attention, the text inside `\text{}` is not processed and treated like a plain text):

$\llbracket \alpha_0 \mapsto x \rrbracket$	This is formation
$\llbracket \alpha_0 \mapsto \emptyset \rrbracket$	Abstraction
$x(\Delta \mapsto 42)$	Application

```
6 \begin{phiuation*}
7 [[ 0->x ]] && \text{This is formation}
8 [[ 0->? ]] && \text{Abstraction}
9 x(D>42) && \text{Application}
10 \end{phiuation*}
```

If you don't use `nodollar` package option, you can still use normal parsing of the dollar sign, by means of `\(...\)` syntax:

The object formation $\llbracket \alpha_0 \mapsto x \rrbracket$ may be replaced with a formula $Q \times a^2$.

```
6 The object formation $[[0->x]]$
7 may be replaced with a formula
8 \(( Q \times a^2 \)).
```

The `phiuation` environment will automatically align formulas by the first arrow, if there are only left-aligned formulas:

$x(\pi) \mapsto \llbracket \lambda \mapsto f_1 \rrbracket,$
$x(a, b, c) \mapsto \llbracket \alpha_0 \mapsto \emptyset, \varphi \mapsto \text{hello}(\xi), x \mapsto \text{FALSE} \rrbracket,$
$\Delta = 43-09,$
$x(y) \equiv x(\alpha_0 \mapsto y).$

```
5 \begin{phiuation*}
6 x(\pi) -> [[\lambda \mapsto f_1]], \\\
7 x(a,b,c) -> [[ \alpha_0 -> ?, \ \
8   @ -> |hello|($), x -> |FALSE| ]], \\\
9 \Delta = |43-09|,
10 x(y) == x(0-> y).
11 \end{phiuation*}
```

If not a single line is indented in `phiuation`, all formulas will be centered:

$\llbracket b \mapsto \emptyset \rrbracket,$
$\llbracket \varphi \mapsto \text{TRUE}, \Delta \mapsto 42 \rrbracket,$
$\psi = \langle \pi, 42 \rangle.$

```
5 \begin{phiuation*}
6 [[ b -> ? ]],
7 [[ @ -> TRUE, \Delta \mapsto 42 ]], \\\
8 \psi = << \pi, 42 >>.
9 \end{phiuation*}
```

It is possible to use “manual splitting” mode in the `phiuation` environment by starting the body with `\begin{split}`:

$$x(\pi) \mapsto 4$$

$$x(a, b, c) \mapsto \llbracket \alpha_0 \mapsto \emptyset \rrbracket$$

```

5 \begin{phiuation*}
6 \begin{split}
7 x(\pi) &\mapsto 4 \\
8 x(a,b,c) &\mapsto \llbracket \alpha_0 \mapsto \emptyset \rrbracket \\
9 \end{split}
10 \end{phiuation*}

```

When necessary to use a percentage sign, prepend it with a backward slash:

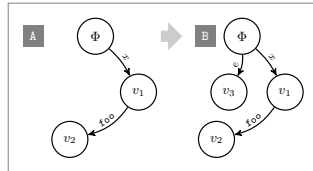
$$x \mapsto \text{sprintf}(\text{"Hello, \%s!"}, \text{name})$$

```

5 \begin{phiuation*}
6 x \mapsto \text{sprintf}(\text{"Hello, \%s!"}, \text{name})
7 \end{phiuation*}
8 \end{document}

```

You can make a copy of a vertex together with its kids:

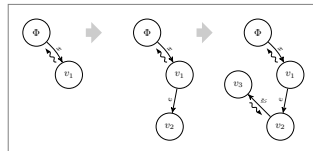


```

5 \begin{sodg}
6 v0 \ll v0!!A
7 v1 xy:v0,.7,1
8 v0->v1 a:x bend:-10
9 v2 xy:v1,-1.3,.8
10 v1->v2 a:|foo| bend:-20
11 v0+a xy:v0,3,0
12 v3a xy:v0a,-.7,1
13 v0a->v3a a:e bend:-15
14 v0=>v0a \ll v0a!B
15 \end{sodg}

```

You can make a copy from a copy:

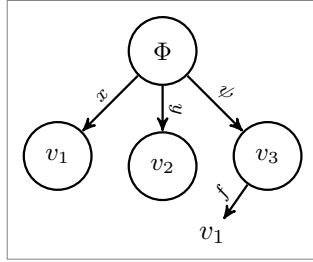


```

5 \begin{sodg}
6 v0
7 v1 xy:v0,.7,1
8 v0->v1 a:x bend:-10 rho
9 v0+a xy:v0,3,0 \ll v0=>v0a
10 v2a xy:v1a,-.8,1.3
11 v1a->v2a a:e
12 v0a+b xy:v0a,3,0 \ll v0a=>v0b
13 v3b xy:v2b,-1,-1
14 v2b->v3b a:\psi{} rho
15 \end{sodg}

```

You can have “broken” edges, using “break” attribute of an edge. The attribute must have a value, which is the percentage of the path between vertices that the arrow should take (can’t be more than 80 and less than 20). This may be convenient when you can’t fit all edges into the graph, for example:

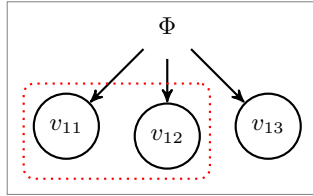


```

5 \begin{sodg}
6 v0
7 v1 xy:v0,-1,1
8 v0->v1 a:x
9 v2 xy:v0,0,1
10 v0->v2 a:y
11 v3 xy:v0,1,1
12 v0->v3 a:\psi{}
13 v3->v1 a:f bend:-75 break:30
14 \end{sodg}

```

You can add [TikZ](#) commands to `sodg` graph, for example:

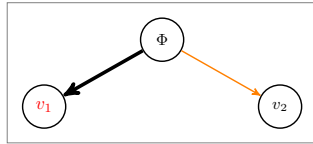


```

6 \begin{sodg}
7 v0 edgeless
8 v11 xy:v0,-1,1 \\\ v0->v11
9 v12 xy:v0,0,1 \\\ v0->v12
10 v13 xy:v0,1,1 \\\ v0->v13
11 \node[draw=red,rounded corners,\
12 dotted,fit=(v11) (v12)] {};
13 \end{sodg}

```

You can modify TikZ style yourself (make sure `style:` stays at the end of the line!), for example:



```

6 \begin{sodg}
7 v0
8 v1 xy:v0,-2,1 style:font=\color{red}
9 v2 xy:v0,2,1
10 v0->v1 style:line width=2pt
11 v0->v2 style:draw=orange
12 \end{sodg}

```

4 Implementation

First, we include a few packages. We need [stmaryrd](#) for `\llbracket` and `\rrbracket` commands:

```
1 \RequirePackage{stmaryrd}
```

We need [amsmath](#) for `equation*` environment:

```
2 \RequirePackage{amsmath}
```

We need [amssymb](#) for `\varnothing` command. We disable `\Bbbk` because it may conflict with some packages from [acmart](#):

```
3 \let\Bbbk\relax\RequirePackage{amssymb}
```

We need [fancyvrb](#) for `\VerbatimEnvironment` command:

```
4 \RequirePackage{fancyvrb}
```

We need [iexec](#) for executing Perl scripts:

```
5 \ifdefined\eolang@noshell\else\RequirePackage{iexec}\fi
```

Then, we process package options:

```
6 \RequirePackage{pgfopts}
7 \RequirePackage{ifluatex}
8 \RequirePackage{ifxetex}
9 \pgfkeys{
10   /eolang/.cd,
11   tmpdir/.store in=\eolang@tmpdir,
12   tmpdir/.default=_eolang\ifxetex-xe\else\ifluatex-lua\fi\fi,
13   nocomments/.store in=\eolang@nocomments,
14   anonymous/.store in=\eolang@anonymous,
15   noshell/.store in=\eolang@noshell,
16   tmpdir
17 }
18 \ProcessPgfPackageOptions{/eolang}
```

Then, we make a directory where all temporary files will be kept:

```
19 \makeatletter
20 \ifdefined\eolang@noshell\else\RequirePackage{shellesc}\fi
21 \IfFileExists
22   {\eolang@tmpdir/\jobname}
23   {\message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
24     already exists^^J}}
25   {
26     \ifdefined\eolang@noshell
27       \message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
28         is not created, because of the "noshell" package option,
29         most probably the compilation will fail later^^J}
30     \else
31       \ifnum\ShellEscapeStatus=1
32         \iexec[null]{mkdir -p "\eolang@tmpdir/\jobname"}
33       \else
34         \message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
35           is not created, because -shell-escape is not set, and
36           it doesn't exist, most probably the compilation
37           will fail later^^J}
38       \fi
39     \fi
40   }
41 \makeatother
```

\eolang@lineno Then, we define an internal counter to protect line number from changing:

```
42 \makeatletter\newcounter{eolang@lineno}\makeatother
```

\eolang@mdfive Then, we define a command for MD5 hash calculating of a file:

```
43 \RequirePackage{pdftexcmds}
44 \makeatletter
45 \newcommand\eolang@mdfive[1]{\pdf@filemdfivesum{#1}}
46 \makeatother
```

-phi.pl Then, we create a Perl script for phiquation processing using VerbatimOut environment from [fancyvrb](#):

```
47 \makeatletter
48 \ifdefined\eolang@noshell
49   \message{eolang: Perl script is not going to be created,
```

```

50 at "\eolang@tmpdir/\jobname-phi.pl" because of the "noshell"
51 package option^^J}
52 \else
53 \openin 15=\eolang@tmpdir/\jobname-phi.pl
54 \ifeof 15
55 \message{eolang: Perl script is going to be created,
56 because it is absent at "\eolang@tmpdir/\jobname-phi.pl",
57 but if -shell-escape is not set, the compilation will
58 most likely fail now^^J}
59 \begin{VerbatimOut}{\eolang@tmpdir/\jobname-phi.pl}
60 $macro = $ARGV[0];
61 open(my $fh, '<', $ARGV[1]);
62 my $tex; { local $/; $tex = <$fh>; }
63 print "% This file is auto-generated by 0.18.0\n";
64 print '% There are ', length($tex),
65 ' chars in the input: ', $ARGV[1], "\n";
66 print '% ---', "\n";
67 if (index($tex, "\t") > 0) {
68   print "TABS are prohibited!";
69   exit 1;
70 }
71 my @lines = split (/\\n/g, $tex);
72 foreach my $t (@lines) {
73   print '% ', $t, "\n";
74 }
75 print '% ---', "\n";
76 $tex =~ s/(?<!\\\)%.*\\n\\n/g;
77 $tex =~ s/^\\s+|\\s+$/g;
78 my $splitting = $tex =~ /^\\begin\\{split\\}/;
79 if ($splitting) {
80   print '% The manual splitting mode is ON since \\begin{split} started the text' . "\n";
81 }
82 my $indents = $tex =~ /\\n +/g;
83 my $gathered = (0 == $indents);
84 if ($gathered) {
85   if ($splitting) {
86     print '% The "gathered" is NOT used because of manual splitting' . "\n";
87     $gathered = 0;
88   } else {
89     print '% The "gathered" is used since all lines are left-aligned' . "\n";
90   }
91 } else {
92   print '% The "gathered" is NOT used because ' .
93     $indents . " lines are indented\n";
94 }
95 my $align = 0;
96 print '% The "align" is NOT used by default' . "\n";
97 if (index($tex, '&&') >= 0) {
98   $macro =~ s/equation/align/g;
99   $align = 1;
100   print '% The "align" is used because of && seen in the text' . "\n";
101 }
102 if ($macro ne 'phiq') {
103   if (not $splitting) {

```

```

104 $tex =~ s/\\\\\\\\n\\n\\n/g;
105 $tex =~ s/\\\\\\n\\s*//g;
106 }
107 $tex =~ s/\\n*(\\\\label\\{[^\\}+\\})\\n*/\\1/g;
108 $tex =~ s/\\n{3,}/\\n\\n/g;
109 }
110 my @texts = ();
111 sub trep {
112     my ($s) = @_ ;
113     my $open = 0;
114     my $p = 0;
115     for (; $p < length($s); $p++) {
116         $c = substr($s, $p, 1);
117         if ($c eq '}') {
118             if ($open eq 0) {
119                 last;
120             }
121             $open--;
122         }
123         if ($c eq '{') {
124             $open++;
125         }
126     }
127     push(@texts, substr($s, 0, $p));
128     return '{TEXT' . (0+@texts - 1) . '}' . substr($s, $p + 1);
129 }
130 $tex =~ s/\\text\\{(.+)/trep("$1")/ge;
131 if (not $splitting) {
132     $tex =~ s/(?<![&])&(?![&])/\\sigma{/g;
133 }
134 $tex =~ s/([\\~\\{a-z0-9]|^)QQ(?:[a-z0-9])/\\1\\dot{\\Phi{}}/g;
135 $tex =~ s/([\\~\\{a-z0-9]|^)Q(?:[a-z0-9])/\\1\\Phi{/g;
136 $tex =~ s/([\\~\\{a-z0-9]|^)D>/\\1\\Delta{...}/g;
137 $tex =~ s/([\\~\\{a-z0-9]|^)L>/\\1\\lambda{...}/g;
138 $tex =~ s/"([~"]+)"/|"1"/g;
139 $tex =~ s/(^|(?<=[s])(\\[\\[,.>|/))([a-zA-Z][a-z0-9]+)(?=[s])(\\[\\[,.-]|$)/|2/g;
140 $tex =~ s/([~_]|^)([0-9]+|\\*)\\/\\(\\?[a-z]+|\\1[a-z]+|\\1)
141 (->|\\.\\.\\.>|>|:=|!->)/\\1\\alpha_{2}\\\\vert{\\3\\space{\\4}/xg;
142 $tex =~ s/([~_]|^)([0-9]+|\\*)
143 (->|\\.\\.\\.>|>|:=|!->)/\\1\\alpha_{2}\\\\space{\\3}/xg;
144 if ($macro ne 'phi') {
145     if (not $splitting) {
146         $tex =~ s/\\begin\\{split\\}\\n\\begin{split}&/g;
147         $tex =~ s/\\n\\s*\\end\\{split\\}/\\end{split}/g;
148         $tex =~ s/\\n\\n\\\\\\&/g;
149         $tex =~ s/\\n\\\\\\phiEOL{\\}n\\&/g;
150         $tex =~ s/\\\\\\$/g;
151         $tex =~ s/\\\\\\//\\\\\\n/g;
152         $tex =~ s/([~&s])\\s{2}([~&s])/\\1 \\2/g;
153         $tex =~ s/\\s{2}/ \\quad{/g;
154         $tex = '&' . $tex;
155     }
156     my $lead = '[~s]+s(?:->|:=|==)s';
157     my @leads = $tex =~ /&${lead}/g;

```

```

158 my @eols = $tex =~ /&/g;
159 if (0+@leads == 0+@eols && 0+@eols > 1) {
160     $tex =~ s/&({lead})/\1~/g;
161     $gathered = 0;
162     print '% The "gathered" is NOT used because all ' .
163         (0+@eols) . ' lines are ' . (0+@leads) . " leads\n";
164 }
165 }
166 if ($macro ne 'phiq') {
167     sub strip_tabs {
168         my ($env, $tex) = @_;
169         $tex =~ s/& //g;
170         return "\\begin{$env}" . $tex . "\\end{$env}";
171     }
172     foreach my $e (('matrix', 'cases')) {
173         $tex =~ s/\\begin{\{Q$e\E*?\}\}(.)\\end{\Q$e\E*?\}}/strip_tabs($1, $2)/sge;
174     }
175 }
176 $tex =~ s/\$/\\xi{/g;
177 $tex =~ s/(?<!\{)\^(?!\\{)/\\rho{/g;
178 $tex =~ s/[\\[\\llbracket\\mathbin{/g;
179 $tex =~ s/[\\]\\mathbin{\\rrbracket}/g;
180 $tex =~ s/([\\s,>()]{0-9A-F}{2}{?:-[0-9A-F]{2})+|
181     [0-9]+(?:\\. [0-9]+)?)(?!\\{)/\\1|\\2|/xg;
182 $tex =~ s/TRUE/|TRUE|/g;
183 $tex =~ s/FALSE/|FALSE|/g;
184 $tex =~ s/\\?/\\varnothing{/g;
185 $tex =~ s/@/\\varphi{/g;
186 $tex =~ s/-([a-z]+)>/\\mathrel{\\phiSlot{1}}/g;
187 $tex =~ s/!->/\\mathbin{\\phiConst}/g;
188 $tex =~ s/->/\\mathbin{\\mapsto}/g;
189 $tex =~ s/~>/\\mathbin{\\phiWave}/g;
190 $tex =~ s/:=/\\mathrel{\\vDash}/g;
191 $tex =~ s/= /\\mathrel{\\equiv}/g;
192 $tex =~ s/\\.\\. >/\\mathbin{\\phiDotted}/g;
193 $tex =~ s/<</\\langle/g;
194 $tex =~ s/>>/\\rangle/g;
195 $tex =~ s/\\{2,\\}/|/g;
196 $tex =~ s/\\|([~|]+)\\|/\\textnormal{\\texttt{\\1}}{/g;
197 $tex =~ s/\\{TEXT(\\d+)\\}/'\\text{' . @texts[$1] . '}'/ge;
198 if ($macro eq 'phiq') {
199     print '\\(' if ($tex ne '');
200 } else {
201     print '\\begin{' . $macro, "}\\n";
202     if (not($align)) {
203         if ($gathered) {
204             print '\\begin{gathered}' . "\\n";
205         } elsif (not $splitting) {
206             print '\\begin{split}' . "\\n";
207         }
208     }
209 }
210 if ($gathered and not($align)) {
211     $tex =~ s/^& //g;

```

```

212 $tex =~ s/\n&/\n/g;
213 }
214 print $tex;
215 if ($macro eq 'phiq') {
216   print '\)' if ($tex ne '');
217 } else {
218   if (not($align)) {
219     if ($gathered) {
220       print "\n" . '\end{gathered}';
221     } elsif (not $splitting) {
222       print "\n" . '\end{split}';
223     }
224   }
225   print "\n" . '\end{' . $macro . '}';
226 }
227 print '\endinput';
228 \end{VerbatimOut}
229 \message{eolang: File with Perl script
230 '\eolang@tmpdir/\jobname-phi.pl' saved^^J}
231 \else
232   \message{eolang: Perl script already exists at
233     "\eolang@tmpdir/\jobname-phi.pl"^^J}
234 \fi
235 \closein 15
236 \fi
237 \makeatother

```

`\phiSaveTo` Then, we define the `\phiSaveTo` command to instruct the phiquation environment that the output should not be sent to the document but saved to the file instead:

```

238 \makeatletter
239 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
240 \makeatother

```

`\eolang@ifabsent` Then, we define the `\eolang@ifabsent` command, which if a given file is absent, runs a processing command, otherwise just inputs it:

```

241 \makeatletter
242 \newcommand\eolang@ifabsent[2]{%
243   \IfFileExists
244     {#1}
245     {%
246       \message{eolang: File "#1" already exists ^^J}%
247       \input{#1}}
248   {%
249     \ifdefined\eolang@noshell%
250       \message{eolang: Shell processing is disabled^^J}%
251     \else%
252       \ifnum\ShellEscapeStatus=1\else%
253         \message{eolang: The -shell-escape command line
254           option is not provided, most probably compilation
255           will fail now:^^J}%
256       \fi%
257       #2%
258     \fi%
259   }%

```

```

260 }
261 \makeatother

```

phiquation Then, we define the phiquation and the phiquation* environments through a supplementary \eolang@process command:

```

262 \makeatletter\newcommand\eolang@process[1]{
263   \def\hash{\eolang@mdfive
264     {\eolang@tmpdir/\jobname/phiquation.tex}-\the\inputlineno}%
265   \eolang@ifabsent
266     {\eolang@tmpdir/\jobname/\hash-phiquation-post.tex}
267     {%
268     \iexec[null]{cp "\eolang@tmpdir/\jobname/phiquation.tex"
269       "\eolang@tmpdir/\jobname/\hash-phiquation.tex"}%
270     \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
271     \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-phiquation-post.tex]{
272       perl "\eolang@tmpdir/\jobname-phi.pl"
273       '#1'
274       "\eolang@tmpdir/\jobname/\hash-phiquation.tex"
275       \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
276       \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
277     }%
278   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
279   \def\eolang@phiSaveTo{\relax}%
280 }
281 %
282 \newenvironment{phiquation*}%
283 {\catcode'\|=12 \VerbatimEnvironment%
284 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
285 \begin{VerbatimOut}
286   {\eolang@tmpdir/\jobname/phiquation.tex}}
287 {\end{VerbatimOut}\eolang@process{equation*}}
288 %
289 \newenvironment{phiquation}%
290 {\catcode'\|=12 \VerbatimEnvironment%
291 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
292 \begin{VerbatimOut}
293   {\eolang@tmpdir/\jobname/phiquation.tex}}
294 {\end{VerbatimOut}\eolang@process{equation}}
295 \makeatother

```

\phiq Then, we define \phiq command:

```

296 \RequirePackage{xstring}
297 \makeatletter\newcommand\phiq[1]{%
298   \StrSubstitute{\detokenize{#1}}{'}'{''''}[\clean]%
299   \def\hash{\pdf@mdfivesum{\clean}-\the\inputlineno}%
300   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
301   \eolang@ifabsent
302     {\eolang@tmpdir/\jobname/\hash-phiq-post.tex}
303     {%
304     \iexec[log,trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{
305       /bin/echo '\clean'}%
306     \iexec[quiet,null]{cp "\eolang@tmpdir/\jobname/phiq.tex"
307       "\eolang@tmpdir/\jobname/\hash-phiq.tex"}%
308     \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-phiq-post.tex]{

```



```

309 perl \eolang@tmpdir/\jobname-phi.pl 'phiq'
310 "\eolang@tmpdir/\jobname/\hash-phiq.tex"
311 \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g' \fi}%
312 \message{eolang: Parsed 'phiq' at line no. \the\inputlineno^^J}%
313 }%
314 \ifdefined\eolang@nodollar\else\catcode'\$=\active\fi%
315 }\makeatother

```

nodollar Then, we redefine dollar sign:

```

316 \ifdefined\eolang@nodollar\else
317 \begingroup
318 \catcode'\$=\active
319 \protected\gdef$#1${\phiq{#1}}
320 \endgroup
321 \AtBeginDocument{\catcode'\$=\active}
322 \fi

```

-sodg.pl Then, we create a Perl script for sodg graphs processing using VerbatimOut from

[fancyvrb](#):

```

323 \makeatletter
324 \ifdefined\eolang@noshell
325 \message{eolang: Perl script is not going to be created
326 at "\eolang@tmpdir/\jobname-sodg.pl", because of the
327 "noshell" package option^^J}
328 \else
329 \openin 15=\eolang@tmpdir/\jobname-sodg.pl
330 \ifeof 15
331 \message{eolang: Perl script is going to be created,
332 because it is absent at "\eolang@tmpdir/\jobname-sodg.pl",
333 but if -shell-escape is not set, the compilation will
334 most likely fail now^^J}
335 \begin{VerbatimOut}{\eolang@tmpdir/\jobname-sodg.pl}
336 sub num {
337 my ($i) = @_;
338 $i =~ s/(\+|-)\./\10./g;
339 return $i;
340 }
341 sub fmt {
342 my ($tex) = @_;
343 $tex =~ s/\\|([~\|]+)\\|\\textnormal{\texttt{\\1}}/g;
344 return $tex;
345 }
346 sub vertex {
347 my ($v) = @_;
348 if (index($v, 'v0') == 0) {
349 return '\Phi';
350 } else {
351 $v =~ s/^v/v_/g;
352 $v =~ s/[~0-9]$/g;
353 return $v . '>';
354 }
355 }
356 sub tailor {
357 my ($t, $m) = @_;

```

```

358 $t =~ s/<([A-Z]?${m}[A-Z]?):([~>]+)/\2/g;
359 $t =~ s/<[A-Z]+:[~>]+>/g;
360 return $t;
361 }
362 open(my $fh, '<', $ARGV[0]);
363 my $tex; { local $/; $tex = <$fh>; }
364 if (index($tex, "\t") > 0) {
365   print "TABS are prohibited!";
366   exit 1;
367 }
368 print '% This file is auto-generated', "\n%\n";
369 print '% --- there are ', length($tex),
370 ' chars in the input (', $ARGV[0], "):\n";
371 foreach my $t (split (/ \n/g, $tex)) {
372   print '% ', $t, "\n";
373 }
374 print "% ---\n";
375 $tex =~ s/\\\\/ \n/g;
376 $tex =~ s/\\n/ /g;
377 $tex =~ s/(\\[a-zA-Z]+)s+/\1/g;
378 $tex =~ s/\n{2,}/ \n/g;
379 my @cmds = split (/ \n/g, $tex);
380 print '% --- before processing:' . "\n";
381 foreach my $t (split (/ \n/g, $tex)) {
382   print '% ', $t, "\n";
383 }
384 print '% ---';
385 print ' (' . (0+@cmds) . " lines)\n";
386 print '\begin{picture}', "\n";
387 for (my $c = 0; $c < 0+@cmds; $c++) {
388   my $cmd = $cmds[$c];
389   $cmd =~ s/^s+//g;
390   $cmd =~ s/(?!\n)%.*//g;
391   my ($head, $tail) = split(/ /, $cmd, 2);
392   my %opts = {};
393   my ($body, $style) = split(/style:/, $tail, 2);
394   $opts{'style'} = $style;
395   $tail = $body;
396   foreach my $p (split(/ /, $tail)) {
397     my ($q, $t) = split(/:/, $p);
398     $opts{$q} = $t;
399   }
400   if (index($head, '\\') == 0) {
401     print $cmd;
402   } elsif (index($head, '->') >= 0) {
403     my $draw = '\draw[';
404     if (exists $opts{'pi'}) {
405       $draw = $draw . '<MB:phi-pi><F:draw=none>';
406       if (not exists $opts{'a'}) {
407         $opts{'a'} = '\pi';
408       }
409     }
410     if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
411       $draw = $draw . '<MB:,phi-rho>';

```

```

412 }
413 $draw = $draw . ',' . $opts{'style'} . ']';
414 my ($from, $to) = split (/>/, $head);
415 $draw = $draw . " (${from}) ";
416 if (exists $opts{'bend'}) {
417     $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
418         num($opts{'bend'}) . '>';
419     if (exists $opts{'rho'}) {
420         $draw = $draw . '<MB:,phi-rho>';
421     }
422     $draw = $draw . ']';
423 } else {
424     $draw = $draw . '--';
425 }
426 if (exists $opts{'a'}) {
427     my $a = $opts{'a'};
428     if (index($a, '$') == -1) {
429         $a = '$' . fmt($a) . '$';
430     } else {
431         $a = fmt($a);
432     }
433     $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
434 }
435 if (exists $opts{'break'}) {
436     $draw = $draw . '<F: coordinate [pos=' .
437         ($opts{'break'} / 100) . ']' (break)>';
438 }
439 $draw = $draw . " (<MF:${to}><B:break-v>)" ;
440 if (exists $opts{'break'}) {
441     print tailor($draw, 'F') . ";\n";
442     print ' \node[outer sep=.1cm,inner sep=0cm] ' .
443         'at (break) (break-v) {' . vertex($to) .
444         '$};' . "\n";
445     print ' ' . tailor($draw, 'B');
446 } else {
447     print tailor($draw, 'M');
448 }
449 } elsif (index($head, '>') >= 0) {
450     my ($from, $to) = split (/>/, $head);
451     my $size = () = $head =~ /=/g;
452     if ($from eq '') {
453         print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
454             $to . '.center]';
455     } elsif ($to eq '') {
456         print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
457             $from . '.center]';
458     } else {
459         print '\node [phi-arrow] at ($(' .
460             $from . ')!0.5!(' . $to . ')$)';
461     }
462     print '{}';
463 } elsif (index($head, '!') >= 0) {
464     my ($v, $marker) = split (/!+/, $head);
465     my $size = () = $head =~ /=/g;

```

```

466 print '\node [phi-marker, left=' .
467 ($size * 0.6) . 'cm of ' .
468 $v . '.center][{' . fmt($marker) . '}';
469 } elseif (index($head, '+') >= 0) {
470 my ($v, $suffix) = split (/+/, $head);
471 my @friends = ($v);
472 foreach my $c (@cmds) {
473     $e = $c;
474     $e =~ s/^\s+//g;
475     my $h = $e;
476     $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
477     foreach my $f (@friends) {
478         my $add = '';
479         if (index($h, $f . '->') >= 0) {
480             $add = substr($h, index($h, '->') + 2);
481         }
482         if ($h =~ /\->Q${f}\E$/) {
483             $add = substr($h, 0, index($h, '->'));
484         }
485         if (index($e, ' xy:' . $f . ',') >= 0) {
486             $add = $h;
487         }
488         if (index($add, '+') == -1
489             and $add ne ''
490             and not(grep(/\>Q${add}\E$/, @friends))) {
491             push(@friends, $add);
492         }
493     }
494 }
495 my @extra = ();
496 foreach my $e (@cmds) {
497     $m = $e;
498     if ($m =~ /\s*Q${v}\E\s/) {
499         next;
500     }
501     if ($m =~ /\s*[^\s]+\s/ and not($m =~ /\s*Q${head}\E\s/)) {
502         next;
503     }
504     foreach my $f (@friends) {
505         my $h = $f;
506         $h =~ s/[a-z]$//g;
507         if ($m =~ s/^(s*)Q${f}\E\+Q${suffix}\E\s?/\1${h}${suffix} /g) {
508             last;
509         }
510         $m =~ s/^(s*)Q${f}\E\s/\1${h}${suffix} /g;
511         $m =~ s/^(s*)Q${f}\E->/\1${h}${suffix}->/g;
512         $m =~ s/\sxy:Q${f}\E,/ xy:${h}${suffix},/g;
513         $m =~ s/->Q${f}\E\s/->${h}${suffix} /g;
514     }
515     if ($m ne $e) {
516         push(@extra, ' ' . $m);
517     }
518 }
519 splice(@extra, 0, 0, @extra[-1]);

```

```

520 splice(@extra, -1, 1);
521 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
522   '), friends: [' . join(', ', @friends) . ']' in ' .
523   (0+@cmds) . ' lines');
524 splice(@cmds, $c, 1, @extra);
525 print '% cloned ' . $v . ' at line no.' . $c .
526   ' (+ ' . (0+@extra) . ' lines -> ' .
527   (0+@cmds) . ' lines total)';
528 } elsif ($head =~ /^v[0-9]+[a-z]?$/ ) {
529   print '\node[';
530   if (exists $opts{'xy'}) {
531     my ($v, $right, $down) = split(/,/ , $opts{'xy'});
532     my $loc = '';
533     if ($down > 0) {
534       $loc = 'below ';
535     } elsif ($down < 0) {
536       $loc = 'above ';
537     }
538     if ($right > 0) {
539       $loc = $loc . 'right';
540     } elsif ($right < 0) {
541       $loc = $loc . 'left';
542     }
543     print ', ' . $loc . '=';
544     print abs(num($down)) . 'cm and ' .
545       abs(num($right)) . 'cm of ' . $v . '.center';
546   }
547   if (exists $opts{'data'}) {
548     print ',phi-data';
549     if ($opts{'data'} ne '') {
550       my $d = $opts{'data'};
551       if (index($d, '|') == -1) {
552         $d = '$\Delta\phiDotted\text{' .
553           '\textnormal{\texttt{' . fmt($d) . '}}}'$';
554       } else {
555         $d = fmt($d);
556       }
557       $opts{'box'} = $d;
558     }
559   } elsif (exists $opts{'atom'}) {
560     print ',phi-atom';
561     if ($opts{'atom'} ne '') {
562       my $a = $opts{'atom'};
563       if (index($a, '$') == -1) {
564         $a = '$\lambda\phiDotted{' . fmt($a) . '$';
565       } else {
566         $a = fmt($a);
567       }
568       $opts{'box'} = $a;
569     }
570   } else {
571     print ',phi-object';
572   }
573   if (exists $opts{'edgeless'}) {

```

```

574     print ',draw=none';
575 }
576 print ',, . $opts{style} . '];
577 print ' (' . $head . ')';
578 print '{';
579 if (exists $opts{tag}) {
580     my $t = $opts{tag};
581     if (index($t, '$') == -1) {
582         $t = '$' . $t . '$';
583     } else {
584         $t = fmt($t);
585     }
586     print $t;
587 } else {
588     print '$' . vertex($head) . '$';
589 }
590 print '}';
591 if (exists $opts{box}) {
592     print ' node[phi-box] at (';
593     print $head, '.south east) {';
594     print $opts{box}, '>';
595 }
596 }
597 print ";\n";
598 }
599 print '\end{picture}%', "\n";
600 print "% --- after processing:\n%";
601 foreach my $c (@cmds) {
602     print '% ', $c, "\n";
603 }
604 print '% --- (' . (0+@cmds) . " lines)\n";
605 print '\endinput';
606 \end{VerbatimOut}
607 \message{eolang: File with Perl script
608   '\eolang@tmpdir/\jobname-sodg.pl' saved^^J}
609 \else
610   \message{eolang: Perl script already exists at
611     "\eolang@tmpdir/\jobname-sodg.pl"^^J}
612 \fi
613 \closein 15
614 \fi
615 \makeatother

```

FancyVerbLine Then, we reset the counter for [fancyvrb](#), so that it starts counting lines from zero when the document starts rendering:

```
616 \setcounter{FancyVerbLine}{0}
```

tikz Then, we include [tikz](#) package and its libraries:

```

617 \RequirePackage{tikz}
618 \usetikzlibrary{arrows}
619 \usetikzlibrary{shapes}
620 \usetikzlibrary{decorations}
621 \usetikzlibrary{decorations.pathmorphing}
622 \usetikzlibrary{decorations.pathreplacing}

```

```

623 \usetikzlibrary{positioning}
624 \usetikzlibrary{calc}
625 \usetikzlibrary{math}
626 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment phicture:

```

627 \newenvironment{phicture}%
628 {\noindent\begin{tikzpicture}[
629   ->,>=stealth',node distance=0,thick,
630   pics/parallel arrow/.style={
631     code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
632 {\end{tikzpicture}}
633 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
634   minimum height=0.5cm, minimum width=0.5cm,
635   single arrow head extend=2mm]
636 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
637   minimum width=1.4em, font={\small\color{white}\ttfamily},
638   fill=gray]
639 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
640   draw,font={\small}]
641 \tikzstyle{phi-object} = [phi-thing,circle]
642 \tikzstyle{phi-data} = [phi-thing,regular polygon,
643   regular polygon sides=8]
644 \tikzstyle{phi-empty} = [phi-object]
645 \tikzset{%
646   phi-rho/.style={
647     postaction={%
648       decoration={
649         show path construction,
650         curveto code={
651           \tikzmath{
652             coordinate \I, \F, \v;
653             \I = (\tikzinputsegmentfirst);
654             \F = (\tikzinputsegmentlast);
655             \v = ($(\I) -(\F)$);
656             real \d, \a, \r, \t;
657             \d = 0.8;
658             \t = atan2(\vy, \vx);
659             if \vx<0 then { \a = 90; } else { \a = -90; };
660             {
661               \draw[arrows={-latex}, decorate,
662                 decoration={%
663                   snake, amplitude=.4mm,
664                   segment length=2mm,
665                   post length=1mm
666                 }
667               ]
668               ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
669               -- ++(\t: 2*\d em);
670             }
671           },
672   lineto code={
673     \tikzmath{
674       coordinate \I, \F, \v;

```

```

675 \I = (\tikzinputsegmentfirst);
676 \F = (\tikzinputsegmentlast);
677 \v = ($(\I) -(\F)$);
678 real \d, \a, \r, \t;
679 \d = 0.8;
680 \t = atan2(\vy, \vx);
681 if \vx<0 then { \a = 90; } else { \a = -90; };
682 {
683 \draw[arrows={-latex}, decorate,
684 decoration={%
685 snake, amplitude=.4mm,
686 segment length=2mm,
687 post length=1mm}]
688 ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
689 -- ++(\t: 2*\d em);
690 };
691 }
692 }
693 },
694 decorate
695 }
696 }
697 }
698 \tikzstyle{phi-pi} = [draw,dotted]
699 \tikzstyle{phi-atom} = [phi-object,double]
700 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
701 rectangle,thin,minimum width=1.2em,anchor=north west,
702 font={\scriptsize}]
703 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
704 above=2pt,sloped/.append style={transform shape},
705 font={\scriptsize},color=black]

```

\sodgSaveTo Then, we define the \sodgSaveTo command to instruct the sodg environment that the output should not be sent to the document but saved to the file instead:

```

706 \makeatletter
707 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
708 \makeatother

```

sodg Then, we create a new environment sodg, as suggested [here](#):

```

709 \makeatletter\newenvironment{sodg}%
710 {\catcode'\|=12 \VerbatimEnvironment%
711 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
712 \begin{VerbatimOut}
713 {\eolang@tmpdir/\jobname/sodg.tex}}
714 {\end{VerbatimOut}}%
715 \def\hash{\eolang@mdfive
716 {\eolang@tmpdir/\jobname/sodg.tex}-\the\inputlineno}%
717 \catcode'\$=3 %
718 \eolang@ifabsent
719 {\eolang@tmpdir/\jobname/\hash-sodg-post.tex}
720 {%
721 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
722 "\eolang@tmpdir/\jobname/\hash-sodg.tex"}%
723 \message{eolang: Start parsing 'sodg' at line no. \the\inputlineno^^J}

```



```

724 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-sodg-post.tex]{
725 perl "\eolang@tmpdir/\jobname-sodg.pl"
726 "\eolang@tmpdir/\jobname/\hash-sodg.tex"
727 \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
728 \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%
729 }
730 \catcode'\$ \active%
731 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
732 \def\eolang@sodgSaveTo{\relax}%
733 }\makeatother

```

`\eoAnon` Then, we define a supplementary command to help us anonymize some content.

```

734 \RequirePackage{hyperref}
735 \pdfstringdefDisableCommands{
736 \def\({}%
737 \def\)}%
738 \def\alpha{\alpha}%
739 \def\varphi{\phi}%
740 }
741 \makeatletter
742 \NewExpandableDocumentCommand{\eoAnon}{O{ANONYMIZED}m}{%
743 \ifdefined\eolang@anonymous%
744 \textcolor{orange}{#1}%
745 \else%
746 #2%
747 \fi%
748 }\makeatother

```

`\eolang` Then, we define a simple supplementary command to help you print EO, the name of our language.

```

749 \newcommand\eolang{%
750 \eoAnon[XYZ]{\sffamily EO}}

```

`\phic` Then, we define a simple supplementary command to help you print φ -calculus, the name of our formal apparatus.

```

751 \newcommand\phic{%
752 \eoAnon[(\alpha)-cal-cu-lus]{(\varphi)-cal-cu-lus}}

```

`\xmirl` Then, we define a simple supplementary command to help you print XMIR, the name of our XML-based format of program representation.

```

753 \newcommand\xmirl{%
754 \eoAnon[XML(\^+)]{XMIR}}

```

`\phiConst` Then, we define a command to render an arrow for a constant attribute, as suggested [here](#):

```

755 \newcommand\phiConst{%
756 \mathrel{\hspace{.15em}}%
757 \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

`\phiWave` Then, we define a command to render an arrow for a multi-layer attribute, as suggested [here](#):

```

758 \newcommand\phiWave{%
759 \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

```

`\phiSlot` Then, we define a command to render an arrow for a slot in a basket:

```
760 \newcommand\phiSlot[1]{%
761   \xrightarrow{\text{\sffamily\scshape #1}}}
```

`\phiOset` Then, we define two commands to position a text over and under an arrow, as suggested [here](#):

```
762 \makeatletter
763 \newcommand\phiOset[2]{%
764   \mathrel{\mathop{#2}\limits^{
765     \vbox to 0ex{\kern-2\ex@
766       \hbox{$\scriptscriptstyle#1$\vss}}}}
767 \newcommand\phiUset[2]{%
768   \mathrel{\mathop{#2}\limits_{
769     \vbox to 0ex{\kern-6.3\ex@
770       \hbox{$\scriptscriptstyle#1$\vss}}}}
771 \makeatother
```

`\phiMany` Then, we define a command for an arrow with iterating indecies:

```
772 \newcommand\phiMany[3]{%
773   \phiOset{#3}{\phiUset{#2}{#1}}}
```

`\phiEOL` Then, we define a command for line breaks in formulas:

```
774 \newcommand\phiEOL{\[-4pt]}
```

`\phiDotted` Then, we define a command to render an arrow for a special attribute, as suggested [here](#):

```
775 \RequirePackage{trimclip}
776 \RequirePackage{amsfonts}
777 \makeatletter
778 \newcommand\phiDotted{%
779   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
780 \newcommand\phiDotted@[2]{%
781   \begingroup%
782   \settowidth{\dimen\z@}{$\m@th#1\rightharpoonup$}%
783   \settoheight{\dimen\tw@}{$\m@th#1\rightharpoonup$}%
784   \sbox\z@{%
785     \makebox[\dimen\z@][s]{%
786       \clipbox{0 0 {0.4\width} 0}%
787       {\resizebox{\dimen\z@}{\height}%
788         {$\m@th#1\dashrightharpoonup$}}%
789       \hss%
790       \clipbox{{0.69\width} {-0.1\height} 0
791         {-\height}}{$\m@th#1\rightharpoonup$}%
792     }%
793   }%
794   \ht\z@=\dimen\tw@ \dp\z@=\z@%
795   \box\z@%
796   \endgroup%
797 }
798 \makeatother
```

References

- Bugayenko, Yegor (2021). *EOLANG and φ -calculus*. arXiv: [2111.13384](#) [cs.PL].
- Kudasov, Nikolai et al. (2022). *φ -calculus: a purely object-oriented calculus of decorated objects*. arXiv: [2204.07454](#) [cs.PL].

Change History

0.0.1	General: First draft.	10	0.12.1	-sodg.pl: The bug is fixed related to the formatting of indexes of vertices.	17
0.0.2	sodg: The environment <code>phigure</code> renamed to <code>sodg</code> for the sake of better semantic. The graph in the picture is solely a SODG graph, that's why the name <code>sodg</code> is better.	24	0.13.0	-phi.pl: Parsing of Φ into $\dot{\Phi}$ implemented.	11
	-phi.pl: New symbol added for basket slots	11	0.14.0	-sodg.pl: The <code>edgeless</code> tag of a vertex removes the border of it.	17
	Parsing of the symbols “@,” “^,” and “&” enabled (<code>\varphi</code> , <code>\rho</code> , and <code>\sigma</code>)	11	0.15.0	-sodg.pl: The <code>style</code> tag of vertices and edges.	17
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets	11	0.16.0	phiquation: The processing of phiquation data is done only if it's the first time processing, otherwise cache is used, thus making processing faster.	16
	<code>\phiq</code> : Parsing of additional symbols enabled.	16	sodg: The processing of <code>sodg</code> data is done only if it's the first time processing, otherwise cache is used, thus making processing faster.	24	
	-sodg.pl: The Perl file now has a fixed name, which doesn't depend on the name of the TeX job. This file may be shared among jobs, no need to make it uniquely named.	17	0.17.0	<code>\eolang@ifabsent</code> : A new supplementary <code>eolang@ifabsent</code> command added	15
0.1.0	General: Parsing of package options introduced.	11	0.18.0	<code>\eolang@ifabsent</code> : The <code>noshell</code> package option added in order to enable complete prohibition of shell interactions.	15
	<code>\eolang</code> : New command <code>\eolang</code> added to print the name of the language in both normal and the anonymous mode of <code>acmart</code>	25	0.2.0	-phi.pl: Numbers automatically render as <code>\texttt</code> . No need to use vertical bars around them anymore.	11
	<code>\eolang@mdfive</code> : New supplementary command added to calculate MD5 sum of a file.	11	-sodg.pl: The content of the <code>atom</code> and the <code>data</code> boxes is parsed automatically as formulas and numbers, respectively.	17	
	-phi.pl: A new Perl script “ <code>eolang-phi.pl</code> ” added for parsing of <code>phi</code> expressions.	11	<code>\xmirl</code> : New command <code>\xmirl</code> prints XMIR in both normal and the anonymous mode of <code>acmart</code>	25	
	<code>\phic</code> : New command <code>\phic</code> prints the name of φ -calculus in both normal and the anonymous mode of <code>acmart</code>	25	0.3.0	<code>\eolang@lineno</code> : New counter for protecting <code>lineno</code>	11
	<code>\phiConst</code> : New command <code>\phiConst</code> added to denote a link to a constant attribute.	25	-phi.pl: New arrow added, that looks like <code>\leadsto</code>	11	
	<code>\phiDotted</code> : New command <code>\phiDotted</code> added to denote a link to a special attribute.	26	<code>\phiWave</code> : New command <code>\phiWave</code> added to denote a link to a		
	-sodg.pl: There are two Perl scripts now: one for <code>phiquation</code> , another one for <code>sodg</code>	17			

multi-layer attribute.	25	dollar sign instead of the <code>\phi</code> command.	17
0.4.0		-phi.pl: New syntax sugar for Φ , just using capital “Q” is enough.	11
-sodg.pl: Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed with the <code>\Delta</code> and the <code>\lambda</code> commands.	17	Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols.	11
Relative positioning of vertices fixed.	17	Text in quotes is automatically converted to <code>\texttt</code>	11
0.5.0		0.8.0	
-phi.pl: Automated formatting of TRUE and FALSE added.	11	General: The anonymous package option added.	11
<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow.	26	-phi.pl: Inside <code>\phi</code> any text inside the <code>\text</code> macro is not processed.	11
<code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket.	26	<code>\phiOset</code> : New commands <code>\phiOset</code> and <code>\phiUset</code> help position text over and under an arrow.	26
-sodg.pl: It is possible to use TikZ commands inside the <code>sodg</code> environment.	17	<code>\phiSaveTo</code> : The output of the <code>\phi</code> environment can be redirected to a file.	15
New syntax introduced that allows to make clones of vertices and all their dependants.	17	-sodg.pl: The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.	17
Now edges may have the <code>break</code> attribute, to make them shorter.	17	<code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file.	24
0.6.0		0.9.0	
General: Package option <code>nocomments</code> added in order to enable comments suppression in temporary <code>.tex</code> files (may be pretty important for <code>.dtx</code> documents).	11	<code>\eoAnon</code> : New command <code>\eoAnon</code> added.	25
-sodg.pl: The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations.	17	-phi.pl: Proper handling of the <code>matrix</code> environment.	11
0.7.0		<code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\\[-4pt]</code>	26
<code>nodollar</code> : Now it is possible to use			

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols		<code>\color</code> 637	308, 309, 310, 326, 329, 332, 335, 608, 611, 713, 716, 719, 721, 722, 724, 725, 726
<code>\\$</code> 176, 300, 314, 318, 321, 717, 730	D	<code>\d</code> .. 197, 656, 657, 667, 668, 678, 679, 688, 689	<code>\ex@</code> 765, 769
<code>\%</code> 275, 311, 727	<code>\dashrightarrow</code> ... 788	<code>\def</code> 239, 263, 279, 299, 707, 715, 732, 736, 737, 738, 739	F
<code>\(</code> 199, 736, 752, 754	<code>\Delta</code> 552	<code>\F</code> 652, 654, 655, 667, 674, 676, 677, 688	<code>\FancyVerbLine</code> 616
<code>\)</code> 216, 737, 752, 754	<code>\detokenize</code> 298	G	
<code>*</code> 140, 142, 173	<code>\dimen</code> 782, 783, 785, 787, 794	<code>\gdef</code> 319	
<code>\+</code> 338, 470, 501, 507	<code>\dp</code> 794	H	
<code>\-</code> 752	<code>\draw</code> ... 403, 631, 661, 683	<code>\hash</code> 263, 266, 269, 271, 274, 299, 302, 307, 308, 310, 715, 719, 722, 724, 726	
<code>\-phi.pl</code> 47	E	<code>\hbox</code> 766, 770	
<code>\-sodg.pl</code> 323	<code>\E</code> 173, 482, 490, 498, 501, 507, 510, 511, 512, 513	<code>\height</code> 787, 790, 791	
<code>\.</code> .. 141, 143, 181, 192, 338	<code>\end</code> 220, 222, 225, 228, 287, 294, 599, 606, 632, 714	<code>\hspace</code> 756, 757	
<code>\/</code> 139, 140	<code>\endinput</code> 227, 605	<code>\hss</code> 789	
<code>\?</code> 184	<code>\eoAnon</code> . 734, 750, 752, 754	<code>\ht</code> 794	
<code>\[</code> 139, 178	<code>\eolang</code> 749	I	
<code>\{</code> ... 78, 107, 130, 146, 147, 173, 177, 181, 197	<code>\eolang@anonymous</code> 14, 743	<code>\I</code> 652, 653, 655, 667, 674, 675, 677, 688	
<code>\}</code> 78, 107, 146, 147, 173, 197	<code>\eolang@ifabsent</code> 241, 265, 301, 718	<code>\iexec</code> ... 32, 268, 271, 304, 306, 308, 721, 724	
<code>\]</code> 139, 179	<code>\eolang@lineno</code> 42	<code>\ifdefined</code> 5, 20, 26, 48, 249, 275, 276, 300, 311, 314, 316, 324, 727, 728, 743	
<code>\^</code> 177	<code>\eolang@mdfive</code> 43, 263, 715	<code>\ifeof</code> 54, 330	
<code>\ </code> 140, 195, 196, 283, 290, 343, 710	<code>\eolang@nocomments</code> ... 13, 275, 311, 727	<code>\IfFileExists</code> ... 21, 243	
Numbers	<code>\eolang@nodollar</code> 300, 314, 316	<code>\ifluatex</code> 12	
<code>\2</code> 139, 141, 143, 152, 181, 358	<code>\eolang@nosshell</code> .. 5, 15, 20, 26, 48, 249, 324	<code>\ifnum</code> 31, 252	
<code>\3</code> 141, 143	<code>\eolang@phiSaveTo</code> 239, 276, 279	<code>\ifxetex</code> 12	
<code>\4</code> 141	<code>\eolang@process</code> 262, 287, 294	<code>\input</code> 247	
A	<code>\eolang@sodgSaveTo</code> 707, 728, 732	<code>\inputlineno</code> ... 264, 270, 299, 312, 716, 723	
<code>\a</code> 656, 659, 667, 678, 681, 688	<code>\eolang@tmpdir</code> 11, 22, 23, 27, 32, 34, 50, 53, 56, 59, 230, 233, 264, 266, 268, 269, 271, 272, 274, 286, 293, 302, 304, 306, 307,	J	
<code>\active</code> . 314, 318, 321, 730		<code>\jobname</code> ... 22, 23, 27, 32, 34, 50, 53, 56, 59, 230, 233, 264, 266, 268, 269, 271,	
<code>\alpha</code> 738, 752			
<code>\AtBeginDocument</code> .. 321			
B			
<code>\Bbbk</code> 3			
<code>\begin</code> 59, 80, 201, 204, 206, 285, 292, 335, 386, 628, 712			
<code>\box</code> 795			
C			
<code>\catcode</code> 283, 290, 300, 314, 318, 321, 710, 717, 730			
<code>\clean</code> 298, 299, 305			
<code>\clipbox</code> 786, 790			
<code>\closein</code> 235, 613			

272, 274, 286, 293, 302, 304, 306, 307, 308, 309, 310, 326, 329, 332, 335, 608, 611, 713, 716, 719, 721, 722, 724, 725, 726			
K		O	<code>\small</code> 637, 640
<code>\kern</code> 765, 769		<code>\openin</code> 53, 329	<code>\sodg</code> 709
L		P	<code>\sodgSaveTo</code> 706
<code>\lambda</code> 564		<code>\pdf@filemdfivesum</code> . 45	<code>\StrSubstitute</code> 298
<code>\leadsto</code> 759		<code>\pdf@mdfivesum</code> 299	<code>\sxy</code> 512
<code>\limits</code> 764, 768		<code>\pdfstringdefDisableCommands</code>	T
M	 735	<code>\t</code> 67, 364, 656, 658, 667, 668, 678, 680, 688, 689
<code>\m@th</code> ... 782, 783, 788, 791		<code>\pgfkeys</code> 9	<code>\text</code> 552, 761
<code>\makeatletter</code>		<code>\Phi</code> 349	<code>\textcolor</code> 744
19, 42, 44, 47, 238, 241, 262, 297, 323, 706, 709, 741, 762, 777		<code>\phic</code> 751	<code>\textnormal</code> 553
<code>\makeatother</code>		<code>\phiConst</code> 755	<code>\texttt</code> 553
41, 42, 46, 237, 240, 261, 295, 315, 615, 708, 733, 748, 771, 798		<code>\picture</code> 627	<code>\the</code> 264, 270, 299, 312, 716, 723
<code>\makebox</code> 785		<code>\phiDotted</code> . 552, 564, 775	<code>\tikz</code> 617
<code>\mapsto</code> 757		<code>\phiDotted@</code> 779, 780	<code>\tikzinputsegmentfirst</code> 653, 675
<code>\mapstochar</code> . 757, 759, 779		<code>\phiEOL</code> 774	<code>\tikzinputsegmentlast</code> 654, 676
<code>\mathop</code> 764, 768		<code>\phiMany</code> 772	<code>\tikzmath</code> 651, 673
<code>\mathpalette</code> 779		<code>\phiOset</code> 762, 773	<code>\tikzset</code> 645
<code>\mathrel</code> 756, 757, 759, 764, 768, 779		<code>\phiiq</code> 296, 319	<code>\tikzstyle</code> 633, 636, 639, 641, 642, 644, 698, 699, 700, 703
<code>\message</code> 23, 27, 34, 49, 55, 229, 232, 246, 250, 253, 270, 312, 325, 331, 607, 610, 723		<code>\phiquation</code> 262	<code>\ttfamily</code> 637
<code>\mspace</code> 759		<code>\phiSaveTo</code> 238	<code>\tw@</code> 783, 794
N		<code>\phiSlot</code> 760	
<code>\newcommand</code> 45, 239, 242, 262, 297, 707, 749, 751, 753, 755, 758, 760, 763, 767, 772, 774, 778, 780		<code>\phiUset</code> 767, 773	U
<code>\newcounter</code> 42		<code>\phiWave</code> 758	<code>\usetikzlibrary</code> ... 618, 619, 620, 621, 622, 623, 624, 625, 626
<code>\newenvironment</code> 282, 289, 627, 709		<code>\pi</code> 407	V
<code>\NewExpandableDocumentCommand</code> 284, 291, 616, 711, 731 742		<code>\ProcessPgfPackageOptions</code> 18	<code>\v</code> 652, 655, 674, 677
<code>\node</code> 442, 453, 456, 459, 466, 529		<code>\protected</code> 319	<code>\value</code> 278, 284, 291, 711, 731
<code>\nodollar</code> 316		Q	<code>\varphi</code> 739, 752
<code>\noindent</code> 628		<code>\Q</code> 173, 482, 490, 498, 501, 507, 510, 511, 512, 513	<code>\vbox</code> 765, 769
		R	<code>\VerbatimEnvironment</code> 283, 290, 710
		<code>\relax</code> 3, 279, 732, 779	<code>\vss</code> 766, 770
		<code>\RequirePackage</code> 1, 2, 3, 4, 5, 6, 7, 8, 20, 43, 296, 617, 734, 775, 776	<code>\vx</code> 658, 659, 680, 681
		<code>\resizebox</code> 787	<code>\vy</code> 658, 680
		<code>\rightarrow</code> . 782, 783, 791	W
		S	<code>\width</code> 786, 790
		<code>\sbox</code> 784	X
		<code>\scriptscriptstyle</code> 766, 770	<code>\xmirc</code> 753
		<code>\scriptsize</code> 702, 705	<code>\xrightarrow</code> 761
		<code>\scshape</code> 761	
		<code>\setcounter</code> 278,	Z
		<code>\settoheight</code> 783	<code>\z@</code> 782, 784, 785, 787, 794, 795
		<code>\settowidth</code> 782	
		<code>\sffamily</code> 750, 761	
		<code>\ShellEscapeStatus</code> 31, 252	