



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

Yegor Bugayenko
yegor256@gmail.com

2024/10/31, 0.18.2

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](#)

<pre> app \mapsto [$\rho \mapsto \xi.b.^2, \alpha_0 t \rightsquigarrow \text{TRUE}$, $b \mapsto [\alpha_* \mapsto \Phi.\text{fn}(56)$, $\varphi \mapsto \dot{\Phi}.\text{string.trim}(\xi)$, $\Delta \mapsto 01\text{-FE-C3}]$, $x \mapsto [\lambda \mapsto \emptyset]$. </pre>	<pre> 1 \documentclass{minimal} 2 \usepackage{eolang} 3 \begin{document} 4 \begin{phiqutation*} 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{phiqutation*} 12 \end{document} </pre>
--	---

`phiqutation (env)` The environment `phiqutation` 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`),
- “]]” maps to “]]” (`\rrbracket`),
- “==” maps to “ \equiv ” (`\equiv`),
- “|abc|” maps to “abc” (`\texttt{abc}`).

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

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

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

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

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

<p>A simple object $x \mapsto [\varphi \mapsto y]$ is a decorator of the data object $y \mapsto [\Delta \mapsto 42]$.</p>	<pre> 4 \begin{document} 5 A simple object 6 \backslash\text{phiq}\{x \rightarrow [[@ \rightarrow y]]\} \backslash 7 is a decorator of 8 the data object \backslash 9 \\$y \rightarrow [[\Delta \mapsto 42]]\$. 10 \end{document} </pre>
---	--

$\text{sodg} (env)$ The environment sodg allows you to draw a **SODG** graph:

	<pre> 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} </pre>
--	--

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 “ rho ” 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
`\xmir` double-blind your paper. There is also `\phic` command to print the name of φ -calculus,
also sensitive to anonymous mode. The macro `\xmir` prints “XMIR”.

<p>In our research we use XYZ, an experimental object-oriented dataflow language, α-calculus, as its formal foundation, and XML⁺ — its XML-based presentation.</p>	<pre> 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 \xmir{ } --- \ 9 its XML-based presentation. 10 \end{document} </pre>
---	---

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 \xmirc{} --- \
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 $x \mapsto y$ makes y a constant, $x \rightsquigarrow y$ makes it a decoratee of an arbitrary number of objects, while $x \dot{\mapsto} y$ makes it a special attribute.

```

6 If $$ is an identifier and $$ is
7 an object, then $x \phiConst y$
8 makes $$ 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.

```

`\phi0set` If you want to put a text over an arrow or under it, use `\phi0set` and `\phiUset` respectively:

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

$[\dot{\mapsto}]$.

```

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 [[ \phi0set{*}{->} ]]
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 $[\alpha_i \dot{\mapsto} x_i]$ are syntactically different but semantically 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: $\llbracket x \mapsto \emptyset \rrbracket$	5 <code>\phiSaveTo{a}</code>
Bound: $\llbracket x \mapsto \llbracket \Delta \mapsto 42 \rrbracket \rrbracket$	6 <code>\begin{phiqutation*}</code>
	7 <code>[[x -> [[D>42]]]]</code>
	8 <code>\end{phiqutation*}</code>
	9 <code>\begin{tabular}{p{.5in}l}</code>
	10 <code>Free: & \$[[x -> ?]]\$ \\\</code>
	11 <code>Bound: & \parbox{1in}{\input{a}} \\\</code>
	12 <code>\end{tabular}</code>

`\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 `phiqutation` 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 <code>\begin{phiqutation*}</code>
	7 <code>\frac \</code>
	8 <code>{x->[[@->y]] \quad y->[[z->42]]} \</code>
	9 <code>{x.z -> 42} \</code>
	10 <code>\text{\sffamily R1}</code>
	11 <code>\end{phiqutation*}</code>

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

$\frac{x \mapsto [\varphi \mapsto y, z \mapsto 42, \alpha_0 g \mapsto \emptyset, \alpha_1 \text{foo} \mapsto 42]}{x \mapsto [\varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto [\psi \rightsquigarrow \text{hello}(12)], \alpha_1 \mapsto 42)]} \text{R2.}$	<pre> 6 \begin{phiqutation*} 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{phiqutation*} </pre>
---	---

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

$\text{foo} \mapsto \left\{ \begin{array}{c} \emptyset \\ [\lambda \mapsto \rho \times \xi . \alpha_0] \\ [\Delta \mapsto 42] \end{array} \right\}$	<pre> 5 \begin{phiqutation*} 6 foo -> \left\{\begin{matrix} \ 7 ? \\ \ 8 [[L> ~ \times \$. \alpha_0]]\ \ 9 [[D> 42]]\ \ 10 \end{matrix}\right\} 11 \end{phiqutation*} </pre>
---	---

The `cases` environment works too:

$\beta \models \begin{cases} [v_2, \varphi \xrightarrow{\text{DTZD}} 42] \\ [v_{33}] \end{cases}$	<pre> 5 \begin{phiqutation*} 6 \beta := \begin{cases} \ 7 [v_2, @ -dtzd> 42] \\ \ 8 [v_{33}] \ 9 \end{cases} 10 \end{phiqutation*} 11 \end{document} </pre>
---	--

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

$x \mapsto [[y \mapsto [z \rightsquigarrow \xi, f \mapsto \emptyset]], \beta_1 \models [\psi \xrightarrow{\text{WAIT}} \emptyset].$	<pre> 1 \documentclass{acmart} 2 \usepackage{eolang} 3 \thispagestyle{empty} 4 \begin{document} 5 \begin{phiqutation*} 6 x -> [[7 y -> [[8 z !-> \$, f ..> ?]]]],\ 9 \beta_1 := [\psi -wait> ?]. 10 \end{phiqutation*} 11 \end{document} </pre>
--	---

It's possible to use `\label` inside the `phiqutation` 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):

$[\alpha_0 \mapsto x]$	This is formation
$[\alpha_0 \mapsto \emptyset]$	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 $[\alpha_0 \mapsto x]$ 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:

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

```

5 \begin{phiuation*}
6 x(\pi) -> [[\lambda .> 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:

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

```

5 \begin{phiuation*}
6 [[ b -> ? ]],
7 [[ @ -> TRUE, \Delta .> 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{phiqutation*}
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{phiqutation*}

```

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

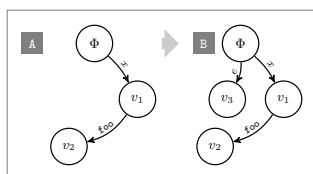
$$x \mapsto \text{sprintf}(\text{"Hello, \%s!"}, \text{name})$$

```

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

```

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

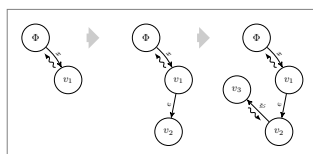


```

5 \begin{sodg}
6 v0 \\\ 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 \\\ v0a!B
15 \end{sodg}

```

You can make a copy from a copy:

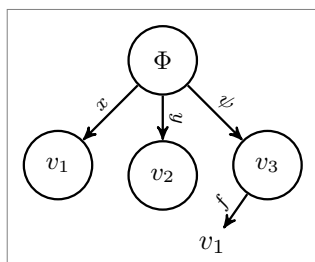


```

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 \\\ v0=>v0a
10 v2a xy:v1a,-.8,1.3
11 v1a->v2a a:e
12 v0a+b xy:v0a,3,0 \\\ 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:

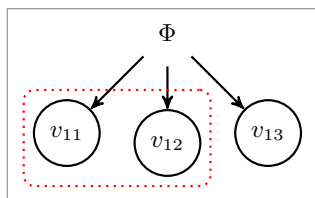


```

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:

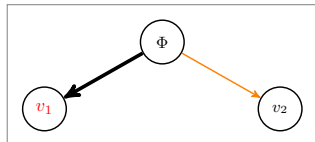


```

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];
61open(my $fh, '<', $ARGV[1]);
62my $tex; { local $/; $tex = <$fh>; }
63print "% This file is auto-generated by eolang.sty 0.18.2\n";
64print '% There are ', length($tex),
65 ' chars in the input: ', $ARGV[1], "\n";
66print '% ---', "\n";
67if (index($tex, "\t") > 0) {
68 print "TABS are prohibited!";
69 exit 1;
70}
71my @lines = split (/\\n/g, $tex);
72foreach my $t (@lines) {
73 print '% ', $t, "\n";
74}
75print '% ---', "\n";
76$tex =~ s/(?<!\n)\%.*\\n\\n/g;
77$tex =~ s/^\\s+|\\s+$//g;
78my $splitting = $tex =~ /-\\begin\\{split\\}/;
79if ($splitting) {
80 print '% The manual splitting mode is ON since \\begin{split} started the text' . "\n";
81}
82my $indents = $tex =~ /\\n +/g;
83my $gathered = (0 == $indents);
84if ($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}
95my $align = 0;
96print '% The "align" is NOT used by default' . "\n";
97if (index($tex, '&&') >= 0) {
98 $macro =~ s/equation/align/g;
99 $align = 1;
100 print '% The "align" is used because of && seen in the text' . "\n";
101}
102if ($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]+|\\[a-z]+|\\)
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 'phiq') {
145   if (not $splitting) {
146     $tex =~ s/\\begin\\{split}\\n/\\begin{split}&/g;
147     $tex =~ s/\\n\\s*\\end\\{split}\\n/\\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 `phi` 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@nosshell%
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

```

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

```

262 \makeatletter\newcommand\eolang@process[1]{
263   \def\hash{\eolang@mdfive
264     {\eolang@tmpdir/\jobname/phiqutation.tex}-\the\inputlineno}%
265   \eolang@ifabsent
266     {\eolang@tmpdir/\jobname/\hash-phiqutation-post.tex}
267     {%
268     \iexec[null]{cp "\eolang@tmpdir/\jobname/phiqutation.tex"
269       "\eolang@tmpdir/\jobname/\hash-phiqutation.tex"}%
270     \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
271     \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-phiqutation-post.tex]{
272       perl "\eolang@tmpdir/\jobname-phi.pl"
273         '#1'
274         "\eolang@tmpdir/\jobname/\hash-phiqutation.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{phiqutation*}%
283 {\catcode'\|=12 \VerbatimEnvironment%
284 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
285 \begin{VerbatimOut}
286   {\eolang@tmpdir/\jobname/phiqutation.tex}}
287 {\end{VerbatimOut}\eolang@process{equation*}}
288 %
289 \newenvironment{phiqutation}%
290 {\catcode'\|=12 \VerbatimEnvironment%
291 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
292 \begin{VerbatimOut}
293   {\eolang@tmpdir/\jobname/phiqutation.tex}}
294 {\end{VerbatimOut}\eolang@process{equation}}
295 \makeatother

```

\phiiq Then, we define \phiiq command:

```

296 \RequirePackage{xstring}
297 \makeatletter\newcommand\phiiq[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       printf '\%s' '\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 $tex;
345 }
346 sub toem {
347 my ($cm) = @_;
348 return $cm * 2.8;
349 }
350 sub vertex {
351 my ($v) = @_;
352 if (index($v, 'v0') == 0) {
353 return '\Phi';
354 } else {
355 $v =~ s/^v/v_/g;
356 $v =~ s/[^0-9]$/g;
357 return '$v . '};

```

```

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

```

```

412     }
413   }
414   if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
415     $draw = $draw . '<MB:,phi-rho>';
416   }
417   $draw = $draw . ',' . $opts{'style'} . ']';
418   my ($from, $to) = split (/>/, $head);
419   $draw = $draw . " ($from) ";
420   if (exists $opts{'bend'}) {
421     $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
422       num($opts{'bend'}) . '>';
423     if (exists $opts{'rho'}) {
424       $draw = $draw . '<MB:,phi-rho>';
425     }
426     $draw = $draw . ']';
427   } else {
428     $draw = $draw . '--';
429   }
430   if (exists $opts{'a'}) {
431     my $a = $opts{'a'};
432     if (index($a, '$') == -1) {
433       $a = '$' . fmt($a) . '$';
434     } else {
435       $a = fmt($a);
436     }
437     $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
438   }
439   if (exists $opts{'break'}) {
440     $draw = $draw . '<F: coordinate [pos=' .
441       ($opts{'break'} / 100) . '] (break)>';
442   }
443   $draw = $draw . " (<MF:${to}><B:break-v>";
444   if (exists $opts{'break'}) {
445     print tailer($draw, 'F') . ";\n";
446     print ' \node[outer sep=' . toem(0.1) . 'em,inner sep=0em] ' .
447       'at (break) (break-v) {$' . vertex($to) .
448       '$};' . "\n";
449     print ' ' . tailer($draw, 'B');
450   } else {
451     print tailer($draw, 'M');
452   }
453 } elsif (index($head, '>') >= 0) {
454   my ($from, $to) = split (/=>/, $head);
455   my $size = () = $head =~ /=/g;
456   if ($from eq '') {
457     print '\node [phi-arrow, left=' . toem($size * 0.6) . 'em of ' .
458       $to . '.center]';
459   } elsif ($to eq '') {
460     print '\node [phi-arrow, right=' . toem($size * 0.6) . 'em of ' .
461       $from . '.center]';
462   } else {
463     print '\node [phi-arrow] at ($(' .
464       $from . ')!0.5!(' . $to . ')$)';
465   }

```

```

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

```

```

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

```

```

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

```

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

```
620 \setcounter{FancyVerbLine}{0}
```

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

```
621 \RequirePackage{tikz}
622 \usetikzlibrary{arrows}
```

```

623 \usetikzlibrary{shapes}
624 \usetikzlibrary{decorations}
625 \usetikzlibrary{decorations.pathmorphing}
626 \usetikzlibrary{decorations.pathreplacing}
627 \usetikzlibrary{positioning}
628 \usetikzlibrary{calc}
629 \usetikzlibrary{math}
630 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment picture:

```

631 \newenvironment{picture}%
632 {\noindent\begin{tikzpicture}[
633   ->,>=stealth',node distance=0,line width=.08em,
634   pics/parallel arrow/.style={
635     code={\draw[-latex,phi-rho] (##1) -- (-##1);}}]}%
636 {\end{tikzpicture}}
637 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
638   minimum height=0.05em, minimum width=0.05em,
639   single arrow head extend=2mm]
640 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
641   minimum width=1.4em, font={\small\color{white}\ttfamily},
642   fill=gray]
643 \tikzstyle{phi-thing} = [inner sep=0pt,minimum height=2.4em,
644   draw,font={\small}]
645 \tikzstyle{phi-object} = [phi-thing,circle]
646 \tikzstyle{phi-data} = [phi-thing,regular polygon,
647   regular polygon sides=8]
648 \tikzstyle{phi-empty} = [phi-object]
649 \tikzset{%
650   phi-rho/.style={
651     postaction={%
652       decoration={
653         show path construction,
654         curveto code={
655           \tikzmath{
656             coordinate \I, \F, \v;
657             \I = (\tikzinputsegmentfirst);
658             \F = (\tikzinputsegmentlast);
659             \v = ($(\I) -(\F)$);
660             real \d, \a, \r, \t;
661             \d = 0.8;
662             \t = atan2(\vy, \vx);
663             if \vx<0 then { \a = 90; } else { \a = -90; };
664             {
665               \draw[arrows={-latex}, decorate,
666                 decoration={%
667                   snake, amplitude=.4mm,
668                   segment length=2mm,
669                   post length=1mm
670                 }]}
671             ($(\F)!5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
672             -- ++(\t: 2*\d em);
673           };
674         }

```

```

675     },
676     lineto code={
677         \tikzmath{
678             coordinate \I, \F, \v;
679             \I = (\tikzinputsegmentfirst);
680             \F = (\tikzinputsegmentlast);
681             \v = ($(\I) -(\F)$);
682             real \d, \a, \r, \t;
683             \d = 0.8;
684             \t = atan2(\vy, \vx);
685             if \vx<0 then { \a = 90; } else { \a = -90; };
686             {
687                 \draw[arrows={-latex}, decorate,
688                     decoration={%
689                         snake, amplitude=.4mm,
690                         segment length=2mm,
691                         post length=1mm}]
692                     ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
693                     -- ++(\t: 2*\d em);
694             };
695         }
696     }
697 },
698 decorate
699 }
700 }
701 }
702 \tikzstyle{phi-pi} = [draw,dotted]
703 \tikzstyle{phi-atom} = [phi-object,double]
704 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
705     rectangle,line width=.04em,minimum width=1.2em,anchor=north west,
706     font={\scriptsize}]
707 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
708     above=2pt,sloped/.append style={transform shape},
709     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:

```

710 \makeatletter
711 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
712 \makeatother

```

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

```

713 \makeatletter\newenvironment{sodg}%
714 {\catcode'\|=12 \VerbatimEnvironment%
715 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
716 \begin{VerbatimOut}
717   {\eolang@tmpdir/\jobname/sodg.tex}}
718 {\end{VerbatimOut}}%
719 \def\hash{\eolang@mdfive
720   {\eolang@tmpdir/\jobname/sodg.tex}-\the\inputlineno}%
721 \catcode'\$=3 %
722 \eolang@ifabsent
723   {\eolang@tmpdir/\jobname/\hash-sodg-post.tex}

```



```

724   {%
725     \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
726       "\eolang@tmpdir/\jobname/hash-sodg.tex"}%
727     \message{eolang: Start parsing 'sodg' at line no. \the\inputlineno^^J}
728     \iexec[trace,stdout=\eolang@tmpdir/\jobname/hash-sodg-post.tex]{
729       perl "\eolang@tmpdir/\jobname-sodg.pl"
730       "\eolang@tmpdir/\jobname/hash-sodg.tex"
731       \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
732       \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%
733   }
734   \catcode'\$ \active%
735   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
736   \def\eolang@sodgSaveTo{\relax}%
737 } \makeatother

```

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

```

738 \RequirePackage{hyperref}
739 \pdfstringdefDisableCommands{
740   \def\({}%
741   \def\)}{%
742   \def\alpha{alpha}%
743   \def\varphi{phi}%
744 }
745 \makeatletter
746 \NewExpandableDocumentCommand{\eoAnon}{0{ANONYMIZED}m}{%
747   \ifdefined\eolang@anonymous%
748     \textcolor{orange}{#1}%
749   \else%
750     #2%
751   \fi%
752 } \makeatother

```

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

```

753 \newcommand\eolang{%
754   \eoAnon[XYZ]{\sffamily EO}}

```

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

```

755 \newcommand\phic{%
756   \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.

```

757 \newcommand\xmirl{%
758   \eoAnon[XML^(+)\{XMIR}}

```

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

```

759 \newcommand\phiConst{%
760   \mathrel{\hspace{.15em}}%
761   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

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

```
762 \newcommand\phiWave{%
763   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}
```

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

```
764 \newcommand\phiSlot[1]{%
765   \xrightarrow{\text{\sffamily\scshape #1}}}
```

`\phi0set` Then, we define two commands to position a text over and under an arrow, as suggested [here](#):

```
766 \makeatletter
767 \newcommand{\phi0set}[2]{%
768   \mathrel{\mathop{#2}\limits^~{
769     \vbox to 0ex{\kern-2\ex@
770       \hbox{\$ \scriptscriptstyle#1$ \vss}}}}
771 \newcommand{\phiUset}[2]{%
772   \mathrel{\mathop{#2}\limits_~{
773     \vbox to 0ex{\kern-6.3\ex@
774       \hbox{\$ \scriptscriptstyle#1$ \vss}}}}
775 \makeatother
```

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

```
776 \newcommand\phiMany[3]{%
777   \phi0set{#3}{\phiUset{#2}{#1}}}
```

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

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

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

```
779 \RequirePackage{trimclip}
780 \RequirePackage{amsfonts}
781 \makeatletter
782 \newcommand{\phiDotted}{%
783   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
784 \newcommand{\phiDotted@}[2]{%
785   \begingroup%
786   \settowidth{\dimen\z@}{\$ \m@th#1 \rightarrow$}%
787   \settoheight{\dimen\tw@}{\$ \m@th#1 \rightarrow$}%
788   \sbox\z@{%
789     \makebox[\dimen\z@][s]{%
790       \clipbox{0 0 {0.4\width} 0}%
791       {\resizebox{\dimen\z@}{\height}%
792         {\$ \m@th#1 \dashrightarrow$}}%
793       \hss%
794       \clipbox{{0.69\width} {-0.1\height} 0
795         {-\height}}{\$ \m@th#1 \rightarrow$}%
796     }%
797   }%
798   \ht\z@=\dimen\tw@ \dp\z@=\z@%
799   \box\z@%
800 \endgroup%
801 }
802 \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 <code>QQ</code> into <code>\dot{\Phi}</code> 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 <code>phiquation</code> data is done only if it's the first time processing, otherwise cache is used, thus making processing faster.	16
	<code>\phiiq</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>\xmir</code> : New command <code>\xmir</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.	26		dollar sign instead of the <code>\phiq</code> command.	17
0.4.0	-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		-phi.pl: New syntax sugar for Φ , just using capital “Q” is enough.	11
	Relative positioning of vertices fixed.	17		Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols.	11
0.5.0	-phi.pl: Automated formatting of TRUE and FALSE added.	11	0.8.0	Text in quotes is automatically converted to <code>\texttt</code>	11
	<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow.	26	General: The anonymous package option added.		11
	<code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket.	26	-phi.pl: Inside <code>\phiquation</code> any text inside the <code>\text</code> macro is not processed.		11
	-sodg.pl: It is possible to use TikZ commands inside the <code>sodg</code> environment.	17	<code>\phi0set</code> : New commands <code>\phi0set</code> and <code>\phiUset</code> help position text over and under an arrow.		26
	New syntax introduced that allows to make clones of vertices and all their dependants.	17	<code>\phiSaveTo</code> : The output of the <code>\phiquation</code> environment can be redirected to a file.		15
	Now edges may have the <code>break</code> attribute, to make them shorter.	17	-sodg.pl: The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.		17
0.6.0	General: Package option <code>no comments</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>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file.		24
	-sodg.pl: The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations.	17	0.9.0		
0.7.0	<code>nodollar</code> : Now it is possible to use		<code>\eoAnon</code> : New command <code>\eoAnon</code> added.		25
			-phi.pl: Proper handling of the <code>matrix</code> environment.		11
			<code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code>		26

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		
\backslash \$ 176, 300, 314, 318, 321, 721, 734	\backslash color 641 308, 309, 310, 326, 329, 332, 335, 612, 615, 717, 720, 723, 725, 726, 728, 729, 730
\backslash % 275, 305, 311, 731	D	
\backslash (..... 199, 740, 756, 758	\backslash d	.. 197, 660, 661, 671, 672, 682, 683, 692, 693
\backslash) 216, 741, 756, 758	\backslash dashrightarrow	... 792
\backslash * 140, 142, 173	\backslash def 239, 263, 279, 299, 711, 719, 736, 740, 741, 742, 743
\backslash + 338, 474, 505, 511	\backslash Delta 556
\backslash - 756	\backslash detokenize 298
\backslash -phi.pl <u>47</u>	\backslash dimen	786, 787, 789, 791, 798
\backslash -sodg.pl <u>323</u>	\backslash dp 798
\backslash .	.. 141, 143, 181, 192, 338	\backslash draw	... 407, 635, 665, 687
\backslash / 139, 140		
\backslash ? 184	E	
\backslash [..... 139, 178	\backslash E	173, 486, 494, 502, 505, 511, 514, 515, 516, 517
\backslash {	... 78, 107, 130, 146, 147, 173, 177, 181, 197	\backslash end 220, 222, 225, 228, 287, 294, 603, 610, 636, 718
\backslash }	78, 107, 146, 147, 173, 197	\backslash endinginput 227, 609
\backslash] 139, 179	\backslash eoAnon	.. <u>738</u> , 754, 756, 758
\backslash ^ 177	\backslash eolang <u>753</u>
\backslash 140, 195, 196, 283, 290, 343, 714	\backslash eolang@anonymous	14, 747
		\backslash eolang@ifabsent <u>241</u> , 265, 301, 722
	Numbers	\backslash eolang@lineno <u>42</u>
\backslash 2	139, 141, 143, 152, 181, 362	\backslash eolang@mdfive	<u>43</u> , 263, 719
\backslash 3 141, 143	\backslash eolang@nocomments	... 13, 275, 311, 731
\backslash 4 141	\backslash eolang@nodollar 300, 314, 316
		\backslash eolang@noshell	.. 5, 15, 20, 26, 48, 249, 324
	A	\backslash eolang@phiSaveTo 239, 276, 279
\backslash a	660, 663, 671, 682, 685, 692	\backslash eolang@process 262, 287, 294
\backslash active	.. 314, 318, 321, 734	\backslash eolang@sodgSaveTo 711, 732, 736
\backslash alpha 742, 756	\backslash eolang@tmpdir 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,
\backslash AtBeginDocument	.. 321		
	B		
\backslash Bbbk 3		
\backslash begin 59, 80, 201, 204, 206, 285, 292, 335, 390, 632, 716		
\backslash box 799		
	C		
\backslash catcode 283, 290, 300, 314, 318, 321, 714, 721, 734		
\backslash clean 298, 299, 305		
\backslash clipbox 790, 794		
\backslash closein 235, 617		
		F	
		\backslash F 656, 658, 659, 671, 678, 680, 681, 692
		\backslash FancyVerbLine <u>620</u>
		G	
		\backslash gdef 319
		H	
		\backslash hash 263, 266, 269, 271, 274, 299, 302, 307, 308, 310, 719, 723, 726, 728, 730
		\backslash hbox 770, 774
		\backslash height 791, 794, 795
		\backslash hspace 760, 761
		\backslash hss 793
		\backslash ht 798
		I	
		\backslash I 656, 657, 659, 671, 678, 679, 681, 692
		\backslash iexec	... 32, 268, 271, 304, 306, 308, 725, 728
		\backslash ifdefined 5, 20, 26, 48, 249, 275, 276, 300, 311, 314, 316, 324, 731, 732, 747
		\backslash ifeof 54, 330
		\backslash IfFileExists	... 21, 243
		\backslash ifluatex 12
		\backslash ifnum 31, 252
		\backslash ifxetex 12
		\backslash input 247
		\backslash inputlineno	... 264, 270, 299, 312, 720, 727
		J	
		\backslash jobname	... 22, 23, 27, 32, 34, 50, 53, 56, 59, 230, 233, 264, 266, 268, 269, 271,

272, 274, 286, 293, 302, 304, 306, 307, 308, 309, 310, 326, 329, 332, 335, 612, 615, 717, 720, 723, 725, 726, 728, 729, 730			
	K		
\kern	769, 773		
	L		
\lambda	568		
\leadsto	763		
\limits	768, 772		
	M		
\m@th	786, 787, 792, 795		
\makeatletter	19, 42, 44, 47, 238, 241, 262, 297, 323, 710, 713, 745, 766, 781		
\makeatother	41, 42, 46, 237, 240, 261, 295, 315, 619, 712, 737, 752, 775, 802		
\makebox	789		
\mapsto	761		
\mapstochar	761, 763, 783		
\mathop	768, 772		
\mathpalette	783		
\mathrel	760, 761, 763, 768, 772, 783		
\message	23, 27, 34, 49, 55, 229, 232, 246, 250, 253, 270, 312, 325, 331, 611, 614, 727		
\mspace	763		
	N		
\newcommand	45, 239, 242, 262, 297, 711, 753, 755, 757, 759, 762, 764, 767, 771, 776, 778, 782, 784		
\newcounter	42		
\newenvironment	282, 289, 631, 713		
\NewExpandableDocumentCommand	284, 291, 620, 715, 735, 746		
\node	446, 457, 460, 463, 470, 533		
\nodollar	316		
\noindent	632		
	O		
\openin	53, 329		
	P		
\pdf@filemdfivesum	45		
\pdf@mdfivesum	299		
\pdfstringdefDisableCommands	739		
\pgfkeys	9		
\Phi	353		
\phic	755		
\phiConst	759		
\picture	631		
\phiDotted	556, 568, 779		
\phiDotted@	783, 784		
\phiEOL	778		
\phiMany	776		
\phiOset	766, 777		
\phii	296, 319		
\phiquation	262		
\phiSaveTo	238		
\phiSlot	764		
\phiUset	771, 777		
\phiWave	762		
\pi	411		
\ProcessPgfPackageOptions	18		
\protected	319		
	Q		
\Q	173, 486, 494, 502, 505, 511, 514, 515, 516, 517		
	R		
\relax	3, 279, 736, 783		
\RequirePackage	1, 2, 3, 4, 5, 6, 7, 8, 20, 43, 296, 621, 738, 779, 780		
\resizebox	791		
\rightarrow	786, 787, 795		
	S		
\sbox	788		
\scriptscriptstyle	770, 774		
\scriptsize	706, 709		
\scshape	765		
\setcounter	278,		
\settoheight	787		
\settowidth	786		
\sffamily	754, 765		
\ShellEscapeStatus	31, 252		
	T		
\small	641, 644		
\sodg	713		
\sodgSaveTo	710		
\StrSubstitute	298		
\sxy	516		
	T		
\t	67, 368, 660, 662, 671, 672, 682, 684, 692, 693		
\text	556, 765		
\textcolor	748		
\textnormal	557		
\texttt	557		
\the	264, 270, 299, 312, 720, 727		
\tikz	621		
\tikzinputsegmentfirst	657, 679		
\tikzinputsegmentlast	658, 680		
\tikzmath	655, 677		
\tikzset	649		
\tikzstyle	637, 640, 643, 645, 646, 648, 702, 703, 704, 707		
\ttfamily	641		
\tw@	787, 798		
	U		
\usetikzlibrary	622, 623, 624, 625, 626, 627, 628, 629, 630		
	V		
\v	656, 659, 678, 681		
\value	278, 284, 291, 715, 735		
\varphi	743, 756		
\vbox	769, 773		
\VerbatimEnvironment	283, 290, 714		
\vss	770, 774		
\vx	662, 663, 684, 685		
\vy	662, 684		
	W		
\width	790, 794		
	X		
\xmir	757		
\xrightarrow	765		
	Z		
\z@	786, 788, 789, 791, 798, 799		