



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

Yegor Bugayenko
yegor256@gmail.com

2025/12/24, 0.23.0

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](#)

<pre> app ↦ [ρ ↦ ξ.b.², 0/t ↦ TRUE, b ↦ [* ↦ Φ.fn(56), φ ↦ Φ.string.trim(ξ), Δ ↦ 01-FE-C3]], x ↦ [λ ↦ ∅]. </pre>	<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 @ -> Q.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 “ \emptyset ” (`\varnothing`),
- “T” maps to “ \perp ” (`\bot`),
- “Q” maps to “ Φ ” (`\Phi`),
- “->” maps to “ \mapsto ” (`\mapsto`),
- “..>” maps to “ $\dot{\mapsto}$ ” (`\phiDotted`),
- “~>” maps to “ \rightsquigarrow ” (`\phiWave`),
- “D” maps to “ Δ ” (`\Delta`),
- “L” maps to “ λ ” (`\lambda`),
- “D>” maps to “ $\Delta\dot{\mapsto}$ ” (`\Delta ..>`),
- “L>” maps to “ $\lambda\dot{\mapsto}$ ” (`\lambda ..>`),
- “[[” maps to “ \llbracket ” (`\llbracket`),
- “]]” maps to “ \rrbracket ” (`\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} ” (`\phiSlot{abc}`),
- “:=” maps to “ \vDash ” (`\vDash`).

Before any arrow you can put a number, which will be rendered as `\alpha` with an index, for example `\phiiq{~0 -> x}` will render “ $\alpha_0 \mapsto x$ ”. It’s also possible to use `~0` to render α_0 . 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 $0/g \rightarrow x$. 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.

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

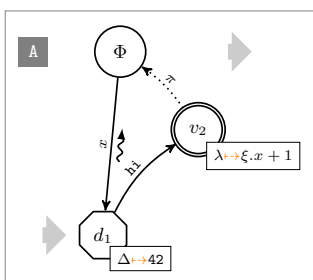
Texts in double quotes are automatically converted to fixed-width font too.

$\backslash\text{phiq}$ 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:

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

```
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 $\text{y} \rightarrow [[\Delta \mapsto 42]]$.
10 \end{document}
```

$\text{sodg} (env.)$ 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 “ 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;
- “ $\text{xy}:\langle v\rangle, \langle r\rangle, \langle d\rangle$ ” places this vertex in a position relative to the vertex “ $\langle v\rangle$,” shifting it right by “ $\langle r\rangle$ ” and down by “ $\langle d\rangle$ ” 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=>”, or from the left to the vertex, by saying for example “=>v5.” If you want the arrow to stay further away from the vertex than usual, 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 the 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”.

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

```

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 representation.
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 representation.

```

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 representation.
10 \end{document}

```

`\phiWave`
`\phiDotted`

A few simple commands are defined to help you render arrows.

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

```

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

```

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

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

$[\overset{*}{\rightarrow}]$.

```

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{phiqutation*}
10 [[ \phi0set{*}{->} ]] .
11 \end{phiqutation*}

```

`\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 rather because of [this](#)):

The expression $[1 \rightarrow x_1, 2 \rightarrow x_2, \dots, \alpha_n \rightarrow x_n]$ and expression $[\alpha_i \overset{n}{\rightarrow} x_i]$ are syntactically different but semantically equivalent.

```

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

```

`\phiSaveTo` If you want to use `phiqutation` or `sodg` environments inside `tabular` or any other
`\sodgSaveTo` environment or command, you won't be able to do this, because `phiqutation` and `sodg` are "verbatim" environments. `\phiSaveTo` and `\sodgSaveTo` commands will help you in this situation. You use them right before `\begin{phiqutation}` 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: $[x \mapsto \emptyset]$
Bound: $[x \mapsto [\Delta \mapsto 42]]$

```

5 \phiSaveTo{a}
6 \begin{phiqutation*}
7 [[ x -> [[D>42]] ]]
8 \end{phiqutation*}
9 \begin{tabular}{p{.5in}l}
10 Free: & $ [[x -> ?]] $ \
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 `phiquation` 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 a continuation of the current line, you can use a single backslash as it's done here:

$\frac{x \mapsto [\varphi \mapsto y], \quad y \mapsto [z \mapsto 42]}{x.z \mapsto \perp} R1$	<pre>6 \begin{phiquation*} 7 \dfrac \ 8 {x->[[@->y]] \quad y->[[z->42]]} \ 9 {x.z -> T} \ 10 \text{\sffamily R1} 11 \end{phiquation*}</pre>
--	--

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

$\frac{x \mapsto [\varphi \mapsto y, z \mapsto 42, \quad 0/g \mapsto \emptyset, 1/foo \mapsto 42]}{x \mapsto [\varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \pi(\alpha_0 \mapsto [\psi \mapsto \text{hello}(12)], \alpha_1 \mapsto 42)]} R2.$	<pre>6 \begin{phiquation*} 7 \dfrac{\begin{split} 8 x->[[@->y, z->42, 9 0/g->?, 1/foo->42]] \end{split}}{\begin{split} 10 \end{split}}{\begin{split} 11 x->[[@->y, z->?, f \rightsquigarrow pi (12 ~0 ->[[\psi -> hello (12)]], 13 ~1 -> 42)]] 14 \end{split}}\text{R2}. 15 \end{phiquation*}</pre>
--	--

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

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

The `cases` environment works too:

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

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

$\begin{array}{l} x \mapsto [\\ y \mapsto [\\ z \mapsto \xi, f \mapsto \emptyset]], \\ \beta_1 \models [\psi \xrightarrow{\text{WAIT}} \emptyset]. \end{array}$	<pre> 1 \documentclass{acmart} 2 \usepackage{eolang} 3 \thispagestyle{empty} 4 \begin{document} 5 \begin{phiqutation*} 6 x -> [[7 y -> [[8 z -> \$, f ..> ?]]],\backslash 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):

<p>Discriminant can be calculated using the following simple formula:</p> $\Delta = b^2 - 4ac. \quad (1)$ <p>Eq. 1 is also widely used in number theory and polynomial factoring.</p>	<pre> 6 Discriminant can be calculated using 7 the following simple formula: 8 \begin{phiqutation} 9 D = b^{^2} - {4}ac. 10 \label{d} 11 \end{phiqutation} 12 Eq.\ref{d} is also widely used in 13 number theory and polynomial factoring. </pre>
---	---

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

```

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

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

```

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

```

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

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

```

5 \begin{phiqutation*}
6 x(\pi) -> [[\lambda \dots \mapsto f_1]], \backslash
7 x(a,b,c) -> [[ ~0 -> ?, L> Foo, \
8   @ -> Q.hello($), x -> false ]], \backslash
9 \Delta = |43-09|,
10 x(y) == x(~0 -> y).
11 \end{phiqutation*}

```

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

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

```

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

```

It is possible to use “manual splitting” mode in the phiqutation environment by starting the body with $\backslash \text{begin}\{\text{split}\}$:

$x(\pi) \mapsto 4$
$x(a, b, c) \mapsto [[\alpha_0 \mapsto \emptyset]]$

```

5 \begin{phiqutation*}
6 \begin{split}
7 x(\pi) &\& -> 4 \backslash\backslash
8 x(a,b,c) &\& -> [[ \alpha_0 -> ? ]] \backslash\backslash
9 \end{split}
10 \end{phiqutation*}

```

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

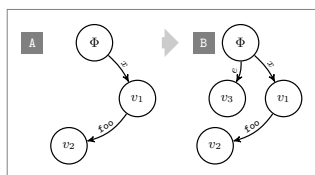
$x \mapsto \text{sprintf}(\text{"Hello, \%s!"}, \text{name})$

```

5 \begin{phiqutation*}
6 x -> \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:

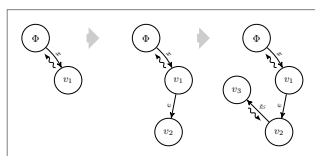


```

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:

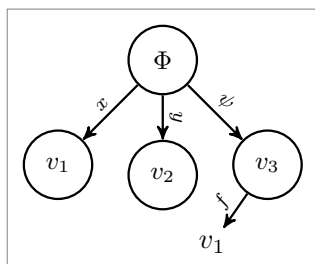


```

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:

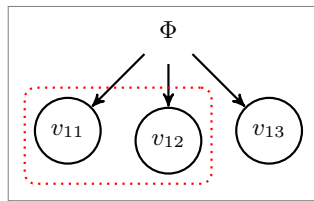


```

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:

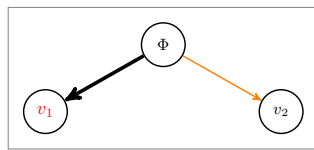


```

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 nodollar/.store in=\eolang@nodollar,
15 anonymous/.store in=\eolang@anonymous,
16 noshell/.store in=\eolang@noshell,
17 tmpdir

```

```

18 }
19 \ProcessPgfPackageOptions{/eolang}
    Then, we make a directory where all temporary files will be kept:
20 \makeatletter
21 \ifdefined\eolang@noshell\else\RequirePackage{shellesc}\fi
22 \IfFileExists
23   {\eolang@tmpdir/\jobname}
24   {\message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
25     already exists^^J}}
26   {
27     \ifdefined\eolang@noshell
28       \message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
29         is not created, because of the "noshell" package option,
30         most probably the compilation will fail later^^J}
31     \else
32       \ifnum\ShellEscapeStatus=1
33         \iexec[null]{mkdir -p "\eolang@tmpdir/\jobname"}
34       \else
35         \message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
36           is not created, because -shell-escape is not set, and
37           it doesn't exist, most probably the compilation
38           will fail later^^J}
39       \fi
40     \fi
41   }
42 \makeatother

```

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

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

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

```

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

```

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

```

48 \makeatletter
49 \ifdefined\eolang@noshell
50   \message{eolang: Perl script is not going to be created,
51     at "\eolang@tmpdir/\jobname-phi.pl" because of the "noshell"
52     package option^^J}
53 \else
54 \openin 15=\eolang@tmpdir/\jobname-phi.pl
55 \ifeof 15
56 \message{eolang: Perl script is going to be created,
57   because it is absent at "\eolang@tmpdir/\jobname-phi.pl",
58   but if -shell-escape is not set, the compilation will
59   most likely fail now^^J}
60 \begin{VerbatimOut}{\eolang@tmpdir/\jobname-phi.pl}
61 \macro = $ARGV[0];
62 open(my $fh, '<', $ARGV[1]);

```

```

63my $tex; { local $/; $tex = <$fh>; }
64print "% This file is auto-generated by eolang.sty 0.23.0\n";
65print '% There are ', length($tex),
66 ' chars in the input: ', $ARGV[1], "\n";
67print '% ---', "\n";
68if (index($tex, "\t") >= 0) {
69  print "TABS are prohibited!";
70  exit 1;
71}
72my @lines = split (/\\n/g, $tex);
73foreach my $t (@lines) {
74  print '% ', $t, "\n";
75}
76print '% ---', "\n";
77$tex =~ s/(?<!\|\\)%.*\\n\\n/g;
78$tex =~ s/^\\s+|\\s+$//g;
79my $splitting = $tex =~ /^\\begin\\{split\\}/;
80if ($splitting) {
81  print '% The manual splitting mode is ON since \\begin{split} started the text' . "\n";
82}
83my $indents = $tex =~ /\\n +/g;
84my $gathered = (0 == $indents);
85if ($gathered) {
86  if ($splitting) {
87    print '% The "gathered" is NOT used because of manual splitting' . "\n";
88    $gathered = 0;
89  } else {
90    print '% The "gathered" is used since all lines are left-aligned' . "\n";
91  }
92} else {
93  print '% The "gathered" is NOT used because ' .
94    $indents . " lines are indented\n";
95}
96my $align = 0;
97print '% The "align" is NOT used by default' . "\n";
98if (index($tex, '&&') >= 0) {
99  $macro =~ s/equation/align/g;
100  $align = 1;
101  print '% The "align" is used because of && seen in the text' . "\n";
102}
103if ($macro ne 'phiq') {
104  if (not $splitting) {
105    $tex =~ s/\\\\\\n\\n\\n/g;
106    $tex =~ s/\\\\\\n\\s*//g;
107  }
108  $tex =~ s/\\n*(\\label\\{[-\\}+\\})\\n*/\\1/g;
109  $tex =~ s/\\n{3,}/\\n\\n/g;
110}
111my @texts = ();
112sub trep {
113  my ($s) = @_ ;
114  my $open = 0;
115  my $p = 0;
116  for (; $p < length($s); $p++) {

```

```

117   $c = substr($s, $p, 1);
118   if ($c eq '}') {
119     if ($open == 0) {
120       last;
121     }
122     $open--;
123   }
124   if ($c eq '{') {
125     $open++;
126   }
127 }
128 push(@texts, substr($s, 0, $p));
129 return '{TEXT' . (0+@texts - 1) . '}' . substr($s, $p + 1);
130 }
131 $tex =~ s/\\text\{(.+)/trep("$1")/ge;
132 $tex =~ s/(?<=[\s,>()]{0-9A-F}{2}(?:-[0-9A-F]{2})+)(?!\\)/|1|/g;
133 $tex =~ s/(?<=[\s,>()]{0-9}(?:\.[0-9]+)?)(?!\\)/|1|/g;
134 $tex =~ s/([\^\\{a-zA-Z0-9\\}|\^)Q(?:[a-zA-Z0-9])\1 \\phiTerminal{\\Phi}/g;
135 $tex =~ s/([\^\\{a-zA-Z0-9\\}|\^)T(?:[a-zA-Z0-9])\1 \\phiTerminal{\\bot}/g;
136 $tex =~ s/([\^\\{a-zA-Z0-9\\}|\^)D>/\1 \\phiTerminal{\\Delta} ..>/g;
137 $tex =~ s/([\^\\{a-zA-Z0-9\\}|\^)L>/\1 \\phiTerminal{\\lambda} ..>/g;
138 $tex =~ s/([\^\\{a-zA-Z0-9\\}|\^)D(?:[a-zA-Z0-9])\1 \\phiTerminal{\\Delta}/g;
139 $tex =~ s/([\^\\{a-zA-Z0-9\\}|\^)L(?:[a-zA-Z0-9])\1 \\phiTerminal{\\lambda}/g;
140 $tex =~ s/"([\^"]+)"/|"1"|/g;
141 $tex =~ s/(?:\^|(?<=[\s)(\)[\.,>\/]))([a-zA-Z][a-zA-Z0-9]+)(?=[\s)(\)[\.,>\/])|$/|1|/g;
142 $tex =~ s/([\^_~|\^)([0-9]+|\^*)\\(\[?[\a-z]+\|[\a-z]+\|)
143 (->|\.[\.]>|\^>|:=)/\1 \\alpha_{2} \\vert{}^3 \\space{}^4/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/\\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/\$/\phiTerminal{\xi}/g;
177 $tex =~ s/(?<\{)\^(?!\{)/\phiTerminal{\rho}/g;
178 $tex =~ s/[[/\phiTerminal{\llbracket}/g;
179 $tex =~ s/[/\phiTerminal{\rrbracket}/g;
180 $tex =~ s/?/\phiTerminal{\varnothing}/g;
181 $tex =~ s/@/\phiTerminal{\varphi}/g;
182 $tex =~ s/-([a-z]+)/\mathrel{\phiSlot{1}}/g;
183 $tex =~ s/->/\mathbin{\phiTerminal{\mapsto}}/g;
184 $tex =~ s/~>/\mathbin{\phiWave}/g;
185 $tex =~ s/:=\mathrel{\vDash}/g;
186 $tex =~ s/==\mathrel{\equiv}/g;
187 $tex =~ s/\.\./\mathbin{\phiTerminal{\phiDotted}}/g;
188 $tex =~ s/~([0-9]+)/\phiTerminal{\alpha_{1}}/g;
189 $tex =~ s/<</\langle/g;
190 $tex =~ s/>>/\rangle/g;
191 $tex =~ s/(?<![\{\}])([.])\phiTerminal{1}/g;
192 $tex =~ s/(?<![\{\}],)\phiTerminal{,}\backslash/g;
193 $tex =~ s/|{2,}/|/g;
194 $tex =~ s/|([~|]+)|\textnormal{\texttt{1}}{/g;
195 $tex =~ s/{TEXT(d+)\}/'\text{' . $texts[1] . '}'/ge;
196 if ($macro eq 'phiq') {
197     print '\(' if ($tex ne '');
198 } else {
199     print '\begin{' , $macro, "}\n";
200     if (not($align)) {
201         if ($gathered) {
202             print '\begin{gathered}' . "\n";
203         } elsif (not $splitting) {
204             print '\begin{split}' . "\n";
205         }
206     }
207 }
208 if ($gathered and not($align)) {
209     $tex =~ s/^&/g;
210     $tex =~ s/\n&/\n/g;
211 }
212 print $tex;
213 if ($macro eq 'phiq') {
214     print '\)' if ($tex ne '');
215 } else {
216     if (not($align)) {
217         if ($gathered) {
218             print "\n" . '\end{gathered}';
219         } elsif (not $splitting) {
220             print "\n" . '\end{split}';
221         }
222     }
223     print "\n" . '\end{' . $macro . '}' ;
224 }

```

```

225 print '\endinput';
226 \end{VerbatimOut}
227 \message{eolang: File with Perl script
228   '\eolang@tmpdir/\jobname-phi.pl' saved^^J}
229 \else
230   \message{eolang: Perl script already exists at
231     "\eolang@tmpdir/\jobname-phi.pl"^^J}
232 \fi
233 \closein 15
234 \fi
235 \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:

```

236 \makeatletter
237 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
238 \makeatother

```

`\eolang@tmp` Then, we define the `\eolang@tmp` command, which generates temporary file names:

```

239 \makeatletter
240 \newcommand\eolang@tmp[1]{#1\ifxetex-xe\else\ifluatex-lua\fi\fi.tex}
241 \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:

```

242 \makeatletter
243 \newcommand\eolang@ifabsent[2]{%
244   \IfFileExists
245     {#1}
246     {%
247       \message{eolang: File "#1" already exists ^^J}%
248       \input{#1}}
249   {%
250     \ifdefined\eolang@noshell%
251       \message{eolang: Shell processing is disabled^^J}%
252     \else%
253       \ifnum\ShellEscapeStatus=1\else%
254         \message{eolang: The -shell-escape command line
255           option is not provided, most probably compilation
256           will fail now:^^J}%
257       \fi%
258       #2%
259     \fi%
260   }%
261 }
262 \makeatother

```

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

```

263 \makeatletter\newcommand\eolang@process[1]{
264   \def\hash{\eolang@mdfive
265     {\eolang@tmpdir/\jobname/\eolang@tmp{phi}}-\the\inputlineno}%
266   \eolang@ifabsent

```

```

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

```

\phiq Then, we define \phiq command:

```

297 \RequirePackage{xstring}
298 \makeatletter\newcommand\phiq[1]{%
299 \StrSubstitute{\detokenize{#1}}{'}'{'''''}[\clean]%
300 \def\hash{\pdf@mdfivesum{\clean}-\the\inputlineno}%
301 \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
302 \eolang@ifabsent
303 {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq-post}}
304 {%
305 \iexec[log,trace,quiet,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{phiq}]{
306 printf '\%s' '\clean'}%
307 \iexec[quiet,null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{phiq}"
308 "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq}}}%
309 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq-post}]{
310 perl \eolang@tmpdir/\jobname-phi.pl 'phiq'
311 "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq}"
312 \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g' \fi}%
313 \message{eolang: Parsed 'phiq' at line no. \the\inputlineno^^J}%
314 }%
315 \ifdefined\eolang@nodollar\else\catcode'\$=active\fi%
316 }\makeatother

```

nodollar Then, we redefine dollar sign:


```

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

```

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

```

324 \makeatletter
325 \ifdefined\eolang@noshell
326 \message{eolang: Perl script is not going to be created
327 at "\eolang@tmpdir/\jobname-sodg.pl", because of the
328 "noshell" package option^^J}
329 \else
330 \openin 15=\eolang@tmpdir/\jobname-sodg.pl
331 \ifeof 15
332 \message{eolang: Perl script is going to be created,
333 because it is absent at "\eolang@tmpdir/\jobname-sodg.pl",
334 but if -shell-escape is not set, the compilation will
335 most likely fail now^^J}
336 \begin{VerbatimOut}{\eolang@tmpdir/\jobname-sodg.pl}
337 sub num {
338 my ($i) = @_;
339 $i =~ s/(\+|-)\./\10./g;
340 return $i;
341 }
342 sub fmt {
343 my ($tex) = @_;
344 $tex =~ s/\\|([\^\\|]+)\\|\\textnormal{\\texttt{1}}/g;
345 return $tex;
346 }
347 sub toem {
348 my ($cm) = @_;
349 return $cm * 2.8;
350 }
351 sub vertex {
352 my ($v) = @_;
353 if (index($v, 'v0') == 0) {
354 return '\Phi';
355 } else {
356 $v =~ s/^v/v_/g;
357 $v =~ s/[^0-9]$/g;
358 return $v . '>';
359 }
360 }
361 sub tailor {
362 my ($t, $m) = @_;
363 $t =~ s/<([A-Z]?${m}[A-Z]?):([\^>]+)>/\2/g;
364 $t =~ s/<[A-Z]+:([\^>]+)>/\2/g;
365 return $t;
366 }
367 open(my $fh, '<', $ARGV[0]);

```

```

368 my $tex; { local $/; $tex = <$fh>; }
369 if (index($tex, "\t") >= 0) {
370   print "TABS are prohibited!";
371   exit 1;
372 }
373 print '% This file is auto-generated', "\n%\n";
374 print '% --- there are ', length($tex),
375   ' chars in the input (', $ARGV[0], "):\n";
376 foreach my $t (split (/\\n/g, $tex)) {
377   print '% ', $t, "\n";
378 }
379 print "% ---\n";
380 $tex =~ s/\\\\\\n/g;
381 $tex =~ s/\\n/g;
382 $tex =~ s/(\\[a-zA-Z]+)\\s+\\1/g;
383 $tex =~ s/\\n{2,}/n/g;
384 my @cmds = split(/\\n/g, $tex);
385 print '% --- before processing:' . "\n";
386 foreach my $t (split (/\\n/g, $tex)) {
387   print '% ', $t, "\n";
388 }
389 print '% ---';
390 print ' (' . (0+@cmds) . " lines)\n";
391 print '\\begin{picture}', "\n";
392 for (my $c = 0; $c < 0+@cmds; $c++) {
393   my $cmd = $cmds[$c];
394   $cmd =~ s/^\\s+//g;
395   $cmd =~ s/(?!\\)%.*/g;
396   my ($head, $tail) = split(/ /, $cmd, 2);
397   my %opts = ();
398   my ($body, $style) = split(/style:/, $tail, 2);
399   $opts{'style'} = $style;
400   $tail = $body;
401   foreach my $p (split(/ /, $tail)) {
402     my ($q, $t) = split(/:/, $p);
403     $opts{$q} = $t;
404   }
405   if (index($head, '\\') == 0) {
406     print $cmd;
407   } elsif (index($head, '->') >= 0) {
408     my $draw = '\\draw[';
409     if (exists $opts{'pi'}) {
410       $draw = $draw . '<MB:phi-pi><F:draw=none>';
411       if (not exists $opts{'a'}) {
412         $opts{'a'} = '\\pi';
413       }
414     }
415     if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
416       $draw = $draw . '<MB:.,phi-rho>';
417     }
418     $draw = $draw . ', ' . $opts{'style'} . ']';
419     my ($from, $to) = split (/->/, $head);
420     $draw = $draw . " ($from) ";
421     if (exists $opts{'bend'}) {

```

```

422     $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
423         num($opts{'bend'}) . '>';
424     if (exists $opts{'rho'}) {
425         $draw = $draw . '<MB:,phi-rho>';
426     }
427     $draw = $draw . '>';
428 } else {
429     $draw = $draw . '--';
430 }
431 if (exists $opts{'a'}) {
432     my $a = $opts{'a'};
433     if (index($a, '$') == -1) {
434         $a = '$' . fmt($a) . '$';
435     } else {
436         $a = fmt($a);
437     }
438     $draw = $draw . '<MB: node [phi-attr] {' . $a . '}>';
439 }
440 if (exists $opts{'break'}) {
441     $draw = $draw . '<F: coordinate [pos=' .
442         ($opts{'break'} / 100) . '] (break)>';
443 }
444 $draw = $draw . " (<MF:${to}><B:break-v>);";
445 if (exists $opts{'break'}) {
446     print tailor($draw, 'F') . ";\n";
447     print '\node[outer sep=' . toem(0.1) . 'em,inner sep=0em] ' .
448         'at (break) (break-v) {'$' . vertex($to) .
449         '$};' . "\n";
450     print ' ' . tailor($draw, 'B');
451 } else {
452     print tailor($draw, 'M');
453 }
454 } elsif (index($head, '>') >= 0) {
455     my ($from, $to) = split (/=>/, $head);
456     my $size = () = $head =~ /=/g;
457     if ($from eq '') {
458         print '\node [phi-arrow, left=' . toem($size * 0.6) . 'em of ' .
459             $to . '.center]';
460     } elsif ($to eq '') {
461         print '\node [phi-arrow, right=' . toem($size * 0.6) . 'em of ' .
462             $from . '.center]';
463     } else {
464         print '\node [phi-arrow] at ($(' .
465             $from . ')!0.5!(' . $to . ')$)';
466     }
467     print '{}';
468 } elsif (index($head, '!') >= 0) {
469     my ($v, $marker) = split (/!+/, $head);
470     my $size = () = $head =~ /*!/g;
471     print '\node [phi-marker, left=' .
472         toem($size * 0.6) . 'em of ' .
473         $v . '.center]{' . fmt($marker) . '}';
474 } elsif (index($head, '+') >= 0) {
475     my ($v, $suffix) = split (/+/, $head);

```

```

476 my @friends = ($v);
477 foreach my $c (@cmds) {
478     $e = $c;
479     $e =~ s/^\s+//g;
480     my $h = $e;
481     $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
482     foreach my $f (@friends) {
483         my $add = '';
484         if (index($h, $f . '->') >= 0) {
485             $add = substr($h, index($h, '->') + 2);
486         }
487         if ($h =~ /-\>\Q${f}\E/) {
488             $add = substr($h, 0, index($h, '->'));
489         }
490         if (index($e, ' xy:' . $f . ',') >= 0) {
491             $add = $h;
492         }
493         if (index($add, '+') == -1
494             and $add ne ''
495             and not(grep(/^Q${add}\E/, @friends))) {
496             push(@friends, $add);
497         }
498     }
499 }
500 my @extra = ();
501 foreach my $e (@cmds) {
502     $m = $e;
503     if ($m =~ /\s*\Q${v}\E\s/) {
504         next;
505     }
506     if ($m =~ /\s*[^\s]+\s/ and not($m =~ /\s*\Q${head}\E\s/)) {
507         next;
508     }
509     foreach my $f (@friends) {
510         my $h = $f;
511         $h =~ s/[a-z]$/g;
512         if ($m =~ s/^(\\s*)\Q${f}\E+\Q${suffix}\E\s?/\1${h}${suffix} /g) {
513             last;
514         }
515         $m =~ s/^(\\s*)\Q${f}\E\s/\1${h}${suffix} /g;
516         $m =~ s/^(\\s*)\Q${f}\E->/\1${h}${suffix}->/g;
517         $m =~ s/\\sxy:\Q${f}\E,/ xy:${h}${suffix},/g;
518         $m =~ s/->\Q${f}\E\s/->${h}${suffix} /g;
519     }
520     if ($m ne $e) {
521         push(@extra, ' ' . $m);
522     }
523 }
524 splice(@extra, 0, 0, $extra[-1]);
525 splice(@extra, -1, 1);
526 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
527     ')', friends: [' . join(', ', @friends) . ']' in ' .
528     (0+@cmds) . ' lines');
529 splice(@cmds, $c, 1, @extra);

```

```

530 print '% cloned ' . $v . ' at line no.' . $c .
531 ' (+ ' . (0+@extra) . ' lines -> ' .
532 (0+@cmds) . ' lines total)';
533 } elseif ($head =~ /^v[0-9]+[a-z]?$/) {
534 print '\node[';
535 if (exists $opts{'xy'}) {
536 my ($v, $right, $down) = split(/,/ , $opts{'xy'});
537 my $loc = '';
538 if ($down > 0) {
539 $loc = 'below ' ;
540 } elsif ($down < 0) {
541 $loc = 'above ' ;
542 }
543 if ($right > 0) {
544 $loc = $loc . 'right';
545 } elsif ($right < 0) {
546 $loc = $loc . 'left';
547 }
548 print ', ' . $loc . '=';
549 print toem(abs(num($down))) . 'em and ' .
550 toem(abs(num($right))) . 'em of ' . $v . '.center';
551 }
552 if (exists $opts{'data'}) {
553 print ',phi-data';
554 if ($opts{'data'} ne '') {
555 my $d = $opts{'data'};
556 if (index($d, '|') == -1) {
557 $d = '$\Delta\phiDotted\text{' .
558 '\textnormal{\texttt{' . fmt($d) . '}}}$';
559 } else {
560 $d = fmt($d);
561 }
562 $opts{'box'} = $d;
563 }
564 } elsif (exists $opts{'atom'}) {
565 print ',phi-atom';
566 if ($opts{'atom'} ne '') {
567 my $a = $opts{'atom'};
568 if (index($a, '$') == -1) {
569 $a = '$\lambda\phiDotted{' . fmt($a) . '$';
570 } else {
571 $a = fmt($a);
572 }
573 $opts{'box'} = $a;
574 }
575 } else {
576 print ',phi-object';
577 }
578 if (exists $opts{'edgeless'}) {
579 print ',draw=none';
580 }
581 print ', ' . $opts{'style'} . ']';
582 print ' (' . $head . ')';
583 print '{';

```

```

584   if (exists $opts{'tag'}) {
585     my $t = $opts{'tag'};
586     if (index($t, '$') == -1) {
587       $t = '$' . $t . '$';
588     } else {
589       $t = fmt($t);
590     }
591     print $t;
592   } else {
593     print '$' . vertex($head) . '$';
594   }
595   print '}}';
596   if (exists $opts{'box'}) {
597     print ' node[phi-box] at (';
598     print $head, '.south east) {';
599     print $opts{'box'}, '}}';
600   }
601 }
602 print ";\n";
603 }
604 print '\end{phicture}}', "\n";
605 print "% --- after processing:\n%";
606 foreach my $c (@cmds) {
607   print '% ', $c, "\n";
608 }
609 print '% --- (' . (0+@cmds) . " lines)\n";
610 print '\endinput';
611 \end{VerbatimOut}
612 \message{eolang: File with Perl script
613   '\eolang@tmpdir/\jobname-sodg.pl' saved^^J}
614 \else
615   \message{eolang: Perl script already exists at
616     "\eolang@tmpdir/\jobname-sodg.pl"^^J}
617 \fi
618 \closein 15
619 \fi
620 \makeatother

```

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

```
621 \setcounter{FancyVerbLine}{0}
```

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

```

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

```

picture Then, we define internal environment phicture:

```

632 \newenvironment{picture}%
633   {\noindent\begin{tikzpicture}[
634     ->,>=stealth',node distance=0,line width=.08em,
635     pics/parallel arrow/.style={
636       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}]}%
637   {\end{tikzpicture}}
638 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
639   minimum height=0.05em, minimum width=0.05em,
640   single arrow head extend=2mm]
641 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
642   minimum width=1.4em, font={\small\color{white}\ttfamily},
643   fill=gray]
644 \tikzstyle{phi-thing} = [inner sep=0pt,minimum height=2.4em,
645   draw,font={\small}]
646 \tikzstyle{phi-object} = [phi-thing,circle]
647 \tikzstyle{phi-data} = [phi-thing,regular polygon,
648   regular polygon sides=8]
649 \tikzstyle{phi-empty} = [phi-object]
650 \tikzset{%
651   phi-rho/.style={
652     postaction={%
653       decoration={
654         show path construction,
655         curveto code={
656           \tikzmath{
657             coordinate \I, \F, \v;
658             \I = (\tikzinputsegmentfirst);
659             \F = (\tikzinputsegmentlast);
660             \v = ($(\I) -(\F)$);
661             real \d, \a, \r, \t;
662             \d = 0.8;
663             \t = atan2(\vy, \vx);
664             if \vx<0 then { \a = 90; } else { \a = -90; };
665             {
666               \draw[arrows={-latex}, decorate,
667                 decoration={%
668                   snake, amplitude=.4mm,
669                   segment length=2mm,
670                   post length=1mm
671                 }
672               ]
673               ($(\F)!5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
674               -- ++(\t: 2*\d em);
675             }
676           },
677         lineto code={
678           \tikzmath{
679             coordinate \I, \F, \v;
680             \I = (\tikzinputsegmentfirst);
681             \F = (\tikzinputsegmentlast);
682             \v = ($(\I) -(\F)$);
683             real \d, \a, \r, \t;
684             \d = 0.8;

```

```

685         \t = atan2(\vy, \vx);
686         if \vx<0 then { \a = 90; } else { \a = -90; };
687         {
688             \draw[arrows={-latex}, decorate,
689                 decoration={%
690                     snake, amplitude=.4mm,
691                     segment length=2mm,
692                     post length=1mm}]
693                 ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
694                 -- ++(\t: 2*\d em);
695         };
696     }
697 }
698 },
699 decorate
700 }
701 }
702 }
703 \tikzstyle{phi-pi} = [draw,dotted]
704 \tikzstyle{phi-atom} = [phi-object,double]
705 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
706     rectangle,line width=.04em,minimum width=1.2em,anchor=north west,
707     font={\scriptsize}]
708 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
709     above=2pt,sloped/.append style={transform shape},
710     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:

```

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

```

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

```

714 \makeatletter\newenvironment{sodg}%
715 {\catcode'\|=12 \VerbatimEnvironment%
716 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
717 \begin{VerbatimOut}
718   {\eolang@tmpdir/\jobname/\eolang@tmp{sodg}}
719 {\end{VerbatimOut}}%
720   \def\hash{\eolang@mdfive
721     {\eolang@tmpdir/\jobname/\eolang@tmp{sodg}}-\the\inputlineno}%
722   \catcode'\$=3 %
723   \eolang@ifabsent
724     {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg-post}}
725     {%
726     \iexec[null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{sodg}"
727       "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg}"}%
728     \message{eolang: Start parsing 'sodg' at line no. \the\inputlineno^^J}
729     \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg-post}]{
730       perl "\eolang@tmpdir/\jobname-sodg.pl"
731       "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg}"
732       \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
733       \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%

```



```

734   }
735   \catcode'\$ \active%
736   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
737   \def\eolang@sodgSaveTo{\relax}%
738 }\makeatother

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

758 \newcommand\xmir{%
759   \eoAnon[XML^(~+)]{XMIR}}

```

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

```

760 \newcommand\phiWave{%
761   \phiTerminal{\mapstochar\mathrel{\mspace{0.45mu}}\leadsto}}

```

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

```

762 \newcommand\phiSlot[1]{%
763   \xrightarrow{\text{\sffamily\scshape #1}}}

```

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

```

764 \makeatletter
765 \newcommand{\phiOset}[2]{%
766   \mathrel{\mathop{#2}\limits^{
767     \vbox to 0ex{\kern-2\ex@

```

```

768 \hbox{\$ \scriptscriptstyle#1$ \vss}}}}
769 \newcommand{\phiUset}[2]{%
770 \mathrel{\mathop{#2}\limits_{
771 \vbox to 0ex{\kern-6.3\ex@
772 \hbox{\$ \scriptscriptstyle#1$ \vss}}}}
773 \makeatother

```

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

```

774 \newcommand{\phiMany}[3]{%
775 \phiOset{#3}{\phiUset{#2}{#1}}

```

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

```

776 \newcommand{\phiEOL}{\[-4pt]}

```

`\phiTerminal` Then, we define a command to wrap all terminals in all expressions:

```

777 \RequirePackage{xcolor}
778 \newcommand{\phiTerminal}[1]{\color{orange}#1}

```

`\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 \phiTerminal{\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

803 \endinput

```

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	vertices.	17
0.0.2	General: First draft.	10
	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
	-phi.pl: New symbol added for basket slots	11
	Parsing of the symbols “@,” “^,” and “&” enabled (<code>\varphi</code> , <code>\rho</code> , and <code>\sigma</code>)	11
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets	11
	<code>\phiq</code> : Parsing of additional symbols enabled.	16
	-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
0.1.0	General: Parsing of package options introduced.	10
	<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
	<code>\eolang@mdfive</code> : New supplementary command added to calculate MD5 sum of a file.	11
	-phi.pl: A new Perl script “ <code>eolang-phi.pl</code> ” added for parsing of <code>phi</code> expressions.	11
	<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
	<code>\phiDotted</code> : New command <code>\phiDotted</code> added to denote a link to a special attribute.	26
	-sodg.pl: There are two Perl scripts now: one for <code>phiqutation</code> , another one for <code>sodg</code>	17
0.12.1	-sodg.pl: The bug is fixed related to the formatting of indexes of	
0.13.0	-phi.pl: Parsing of <code>QQ</code> into <code>\dot{\Phi}</code> implemented.	11
0.14.0	-sodg.pl: The <code>edgeless</code> tag of a vertex removes the border of it.	17
0.15.0	-sodg.pl: The <code>style</code> tag of vertices and edges.	17
0.16.0	<code>phiqutation</code> : The processing of <code>phiqutation</code> data is done only if it's the first time processing, otherwise cache is used, thus making processing faster.	15
	<code>sodg</code> : 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
0.17.0	<code>\eolang@ifabsent</code> : A new supplementary <code>eolang@ifabsent</code> command added	15
0.18.0	<code>\eolang@ifabsent</code> : The <code>noshell</code> package option added in order to enable complete prohibition of shell interactions.	15
0.18.3	<code>\eolang@tmp</code> : A new supplementary <code>eolang@tmp</code> command added, to generate temporary file names.	15
0.19.0	-phi.pl: Parsing of <code>T</code> into <code>\bot</code> implemented.	11
0.2.0	-phi.pl: Numbers automatically render as <code>\texttt</code> . No need to use vertical bars around them anymore.	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
	<code>\xmire</code> : New command <code>\xmire</code> prints XMIR in both normal and the anonymous mode of <code>acmart</code>	25
0.20.0	-phi.pl: Parsing of <code>0</code> into <code>\alpha_0</code> implemented.	11

0.20.2	-phi.pl: Formatting of numbers fixed, now it works in any place inside an expression.	11	0.6.0	General: Package option nocomments added in order to enable comments suppression in temporary .tex files (may be pretty important for .dtx documents).	10
0.21.0	-phi.pl: Automated formatting of TRUE and FALSE removed.	11	-sodg.pl: The rrho attribute is retired, now rho works just fine in all situations.	17	
	\phiTerminal: New command \phiTerminal added, to highlight all terminals in phi expressions. . .	26	0.7.0	nodollar: Now it is possible to use dollar sign instead of the \phiq command.	16
0.22.0	-phi.pl: Automated formatting of D and L added.	11	-phi.pl: New syntax sugar for Φ , just using capital "Q" is enough.	11	
0.23.0	-phi.pl: Parsing of QQ into \dot{\Phi} removed.	11	Object names are automatically converted to \texttt, provided their names include two or more symbols.	11	
0.3.0	\eolang@lineno: New counter for protecting lineno.	11	Text in quotes is automatically converted to \texttt.	11	
	-phi.pl: New arrow added, that looks like \leadsto.	11	0.8.0	General: The anonymous package option added.	10
	\phiWave: New command \phiWave added to denote a link to a multi-layer attribute.	25	-phi.pl: Inside phiquation any text inside the \text macro is not processed.	11	
0.4.0	-sodg.pl: Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed with the \Delta and the \lambda commands.	17	\phiOset: New commands \phiOset and \phiUset help position text over and under an arrow.	25	
	Relative positioning of vertices fixed.	17	\phiSaveTo: The output of the phiquation environment can be redirected to a file.	15	
0.5.0	-phi.pl: Automated formatting of TRUE and FALSE added.	11	-sodg.pl: The tag attribute is introduced for changing labels inside a vertex circle.	17	
	\phiMany: New command \phiMany enables iterating over an arrow. . .	26	\sodgSaveTo: The output of the sodg environment can be redirected to a file.	24	
	\phiSlot: New command \phiSlot added to denote a link to a slot in a basket.	25	0.9.0	\eoAnon: New command \eoAnon added.	25
	-sodg.pl: It is possible to use TikZ commands inside the sodg environment.	17	-phi.pl: Proper handling of the matrix environment.	11	
	New syntax introduced that allows to make clones of vertices and all their dependants.	17	\phiEOL: New command \phiEOL added, instead of \[-4pt].	26	
	Now edges may have the break attribute, to make them shorter. . .	17			

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>\clipboard</code> 790, 794	309, 311, 718, 721,
<code>\\$</code> 176, 301,	<code>\closein</code> 233, 618	724, 726, 727, 729, 731
315, 319, 322, 722, 735	<code>\color</code> 642, 778	<code>\eolang@tmpdir</code>
<code>\%</code> 276, 306, 312, 732		. . . 11, 23, 24, 28,
<code>\(</code> 197, 741, 757, 759	D	33, 35, 51, 54, 57,
<code>\)</code> 214, 742, 757, 759	<code>\d</code> . . 195, 661, 662, 672,	60, 228, 231, 265,
<code>*</code> 142, 173	673, 683, 684, 693, 694	267, 269, 270, 272,
<code>\+</code> 339, 475, 506, 512	<code>\dashrightarrow</code> . . . 792	273, 275, 287, 294,
<code>\-</code> 757	<code>\def</code> 237, 264,	303, 305, 307, 308,
<code>\-phi.pl</code> 48	280, 300, 712, 720,	309, 310, 311, 327,
<code>\-sodg.pl</code> 324	737, 741, 742, 743, 744	330, 333, 336, 613,
<code>\.</code> 133, 143, 187, 339	<code>\Delta</code> 557	616, 718, 721, 724,
<code>\/</code> 141, 142	<code>\detokenize</code> 299	726, 727, 729, 730, 731
<code>\?</code> 180	<code>\dimen</code> 786, 787, 789, 791, 798	<code>\ex@</code> 767, 771
<code>\[</code> 141, 178	<code>\dp</code> 798	F
<code>\{</code> 79, 108, 131, 132, 133,	<code>\draw</code> . . . 408, 636, 666, 688	<code>\F</code> 657, 659, 660,
134, 135, 136, 137,	E	672, 679, 681, 682, 693
138, 139, 146, 147,	<code>\E</code> 173, 487, 495, 503, 506,	<code>\FancyVerbLine</code> 621
173, 177, 191, 192, 195	512, 515, 516, 517, 518	G
<code>\}</code> 79, 108, 146, 147, 173, 195	<code>\end</code> 218,	<code>\gdef</code> 320
<code>\]</code> 141, 179	220, 223, 226, 288,	H
<code>\^</code> 177	295, 604, 611, 637, 719	<code>\hash</code> 264, 267,
<code>\ </code> 142, 193,	<code>\endinput</code> . . 225, 610, 803	270, 272, 275, 300,
194, 284, 291, 344, 715	<code>\eoAnon</code> . 739, 755, 757, 759	303, 308, 309, 311,
Numbers	<code>\eolang</code> 754	720, 724, 727, 729, 731
<code>\2</code> 143, 152, 363	<code>\eolang@anonymous</code> 15, 748	<code>\hbox</code> 768, 772
<code>\3</code> 143	<code>\eolang@ifabsent</code> . .	<code>\height</code> 791, 794, 795
<code>\4</code> 143	. . . 242, 266, 302, 723	<code>\hss</code> 793
A	<code>\eolang@lineno</code> 43	<code>\ht</code> 798
<code>\a</code> 661, 664, 672, 683, 686, 693	<code>\eolang@mdfive</code> 44, 264, 720	I
<code>\active</code> . 315, 319, 322, 735	<code>\eolang@nocomments</code>	<code>\I</code> 657, 658, 660,
<code>\alpha</code> 743, 757	. . . 13, 276, 312, 732	672, 679, 680, 682, 693
<code>\AtBeginDocument</code> . . 322	<code>\eolang@nodollar</code> . .	<code>\iexec</code> . . . 33, 269, 272,
B	. . . 14, 301, 315, 317	305, 307, 309, 726, 729
<code>\Bbbk</code> 3	<code>\eolang@noshell</code> . . 5,	<code>\ifdefined</code> 5,
<code>\begin</code> 60, 81,	16, 21, 27, 49, 250, 325	21, 27, 49, 250, 276,
199, 202, 204, 286,	<code>\eolang@phiSaveTo</code> .	277, 301, 312, 315,
293, 336, 391, 633, 717 237, 277, 280	317, 325, 732, 733, 748
<code>\box</code> 799	<code>\eolang@process</code> . . .	<code>\ifeof</code> 55, 331
C 263, 288, 295	<code>\IfFileExists</code> . . . 22, 244
<code>\catcode</code>	<code>\eolang@sodgSaveTo</code>	<code>\ifluatex</code> 12, 240
284, 291, 301, 315, 712, 733, 737	<code>\ifnum</code> 32, 253
319, 322, 715, 722, 735	<code>\eolang@tmp</code> 239,	<code>\ifxetex</code> 12, 240
<code>\clean</code> 299, 300, 306	265, 267, 269, 270,	<code>\input</code> 248
	272, 275, 287, 294,	
	303, 305, 307, 308,	

<code>\inputlineno</code> ... 265, 271, 300, 313, 721, 728	<code>\nodollar</code> 317	<code>\ShellEscapeStatus</code> 32, 253
	<code>\noindent</code> 633	<code>\small</code> 642, 645
J	O	<code>\sodg</code> 714
<code>\jobname</code> ... 23, 24, 28, 33, 35, 51, 54, 57, 60, 228, 231, 265, 267, 269, 270, 272, 273, 275, 287, 294, 303, 305, 307, 308, 309, 310, 311, 327, 330, 333, 336, 613, 616, 718, 721, 724, 726, 727, 729, 730, 731	<code>\openin</code> 54, 330	<code>\sodgSaveTo</code> 711
	P	<code>\StrSubstitute</code> ... 299
	<code>\pdf@filemdfivesum</code> . 46	<code>\sxy</code> 517
	<code>\pdf@mdfivesum</code> 300	T
	<code>\pdfstringdefDisableCommands</code> 740	<code>\t</code> 68, 369, 661, 663, 672, 673, 683, 685, 693, 694
	<code>\pgfkeys</code> 9	<code>\text</code> 557, 763
	<code>\Phi</code> 354	<code>\textcolor</code> 749
	<code>\phic</code> 756	<code>\textnormal</code> 558
	<code>\picture</code> 632	<code>\texttt</code> 558
	<code>\phiDotted</code> . 557, 569, 779	<code>\the</code> 265, 271, 300, 313, 721, 728
	<code>\phiDotted@</code> 783, 784	<code>\tikz</code> 622
	<code>\phiEOL</code> 776	<code>\tikzinputsegmentfirst</code> 658, 680
	<code>\phiMany</code> 774	<code>\tikzinputsegmentlast</code> 659, 681
	<code>\phiOset</code> 764, 775	<code>\tikzmath</code> 656, 678
	<code>\phiIq</code> 297, 320	<code>\tikzset</code> 650
	<code>\phiIquation</code> 263	<code>\tikzstyle</code> 638, 641, 644, 646, 647, 649, 703, 704, 705, 708
	<code>\phiSaveTo</code> 236	<code>\ttfamily</code> 642
	<code>\phiSlot</code> 762	<code>\tw@</code> 787, 798
	<code>\phiTerminal</code> 761, 777, 783	U
	<code>\phiUset</code> 769, 775	<code>\usetikzlibrary</code> ... 623, 624, 625, 626, 627, 628, 629, 630, 631
	<code>\phiWave</code> 760	V
	<code>\pi</code> 412	<code>\v</code> 657, 660, 679, 682
	<code>\ProcessPgfPackageOptions</code> 19	<code>\value</code> 279, 285, 292, 716, 736
	<code>\protected</code> 320	<code>\varphi</code> 744, 757
	Q	<code>\vbox</code> 767, 771
	<code>\Q</code> 173, 487, 495, 503, 506, 512, 515, 516, 517, 518	<code>\VerbatimEnvironment</code> 284, 291, 715
	R	<code>\vss</code> 768, 772
	<code>\relax</code> ... 3, 280, 737, 783	<code>\vx</code> 663, 664, 685, 686
	<code>\RequirePackage</code> 1, 2, 3, 4, 5, 6, 7, 8, 21, 44, 297, 622, 739, 777, 779, 780	<code>\vy</code> 663, 685
	<code>\resizebox</code> 791	W
	<code>\rightarrow</code> . 786, 787, 795	<code>\width</code> 790, 794
	S	X
	<code>\sbox</code> 788	<code>\xmir</code> 758
	<code>\scriptscriptstyle</code> 768, 772	<code>\xrightarrow</code> 763
	<code>\scriptsize</code> 707, 710	Z
	<code>\scshape</code> 763	<code>\z@</code> 786, 788, 789, 791, 798, 799
	<code>\setcounter</code> ... 279, 285, 292, 621, 716, 736	
	<code>\settoheight</code> 787	
	<code>\settowidth</code> 786	
	<code>\sffamily</code> 755, 763	
N		
<code>\newcommand</code> .. 46, 237, 240, 243, 263, 298, 712, 754, 756, 758, 760, 762, 765, 769, 774, 776, 778, 782, 784		
<code>\newcounter</code> 43		
<code>\newenvironment</code> 283, 290, 632, 714		
<code>\NewExpandableDocumentCommand</code> 747		
<code>\node</code> 447, 458, 461, 464, 471, 534		