



eolang: L^AT_EX Package
for Formulas and Graphs
of EO Programming Language
and φ -Calculus*

Yegor Bugayenko
yegor256@gmail.com

2026-06-30, 0.23.1

NB! The T_EX processor must be invoked with the `-shell-escape` option, and [Perl](#) must be installed on the host system. Should the `-shell-escape` option be omitted, the package falls back to cached files, if any are present; in their absence, compilation fails. When a document is to be prepared for compilation without the `-shell-escape` option, it should be compiled locally with the option enabled, whereupon all files (including those within the `_eolang-*` directories) must be bundled into a single ZIP archive. The use of the `tmpdir` package option is advised in such cases, so that the directory name is not made dependent upon the L^AT_EX engine.

When `-shell-escape` is set, this package does not operate on Windows, since it employs a POSIX command-line interface.

1 Introduction

This package typesets formulas of φ -calculus, the formal foundation of the [EO](#) programming language. The calculus was introduced by Bugayenko (2021) and subsequently formalised by Kudasov et al. (2022). A simple expression is rendered as follows:

*The sources are in GitHub at [objectionary/eolang.sty](#)

$\begin{aligned} \text{app} &\mapsto \llbracket \\ &\quad \rho \mapsto \xi.b.^2, 0/t \rightsquigarrow \text{TRUE}, \\ &\quad b \mapsto \llbracket * \mapsto \Phi.\text{fn}(56), \\ &\quad \quad \varphi \mapsto \Phi.\text{string.trim}(\xi), \\ &\quad \quad \Delta \mapsto 01\text{-FE-C3} \rrbracket, \\ &\quad x \mapsto \llbracket \lambda \mapsto \emptyset \rrbracket. \end{aligned}$	<pre> 1 \documentclass{minimal} 2 \usepackage{eolang} 3 \begin{document} 4 \begin{phiquation*} 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{phiquation*} 12 \end{document} </pre>
--	--

phiquation (*env.*) The **phiquation** environment permits the writing of φ -calculus expressions in a simple plain-text notation, in which:

- “@” 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 “ \mapsto ” (`\phiDotted`),
- “~>” maps to “ \rightsquigarrow ” (`\phiWave`),
- “D” maps to “ Δ ” (`\Delta`),
- “L” maps to “ λ ” (`\lambda`),
- “D>” maps to “ $\Delta \mapsto$ ” (`\Delta ..>`),
- “L>” maps to “ $\lambda \mapsto$ ” (`\lambda ..>`),
- “[[” maps to “ \llbracket ” (`\llbracket`),
- “]]” maps to “ \rrbracket ” (`\rrbracket`),
- “==” maps to “ \equiv ” (`\equiv`),
- “|abc|” maps to “abc” (`\texttt{abc}`).

Furthermore, several symbols are supported for the φ PU architecture:

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

A number may precede any arrow, in which case it is rendered as `\alpha` bearing an index; for instance, `\phiq{~0 -> x}` produces “ $\alpha_0 \mapsto x$ ”. The form `~0` likewise yields α_0 . An asterisk may be employed in place of a number.

A slash followed by a title may be appended to the number of an attribute, as in $0/g \rightarrow x$. This is rendered as $\alpha_0|g \rightarrow x$. Fixed-width words are likewise admissible, for instance `\phiq{0/|f|→x}` renders as “ $\alpha_0|f \rightarrow x$ ”. An asterisk may replace the number, whereby `\phiq{* /g→x}` renders as “ $\alpha_*|g \rightarrow x$ ”.

Numbers are automatically rendered in fixed-width font; vertical bars need not always be supplied.

Object names of more than one letter are likewise rendered in fixed-width font.

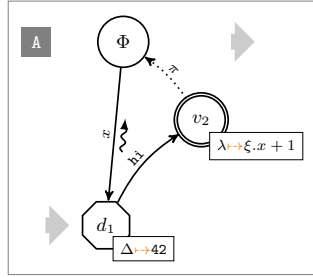
Texts within double quotes are likewise rendered in fixed-width font.

`\phiq` The `\phiq` command permits the inlining of φ -calculus expressions in the same simple plain-text notation. The dollar sign may be employed directly:

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 \phiq{x -> [[@ -> y]]} \\\
7 is a decorator of
8 the data object \\\
9 $y -> [[\Delta ..> 42]]$.
10 \end{document}
```

`sodg (env.)` The `sodg` environment permits the drawing of a [SODG](#) graph:



```
1 \documentclass{standalone}
2 \usepackage{eolang}
3 \begin{document}
4 \begin{sodg}
5 v0 \\\ v0==> \\\ v0!!A
6 v1 xy:v0,-.8,2.8 data:42 tag:d_1
7 v0->v1 a:x rho \\\ =>v1
8 v2 xy:v0,+1,+1 atom:\xi.x+1
9 v1->v2 a:|h1| 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 within each line are separated by a single space. The first marker is either the unique name of a vertex, such as “v1” in the example above, or an edge, such as “v0->v1”. All other markers are either unary, such as “rho”, or binary, such as “atom:\$\xi.x+1\$”. Binary markers consist of two parts separated by a colon.

The following markers are supported for a vertex:

- “tag:<math>” puts a custom label <math> into the circle;
- “data:[<box>]” makes it a data vertex with an optional attached “<box>” (the content of the box may only be numeric data);
- “atom:[<box>]” makes it an atom with an optional attached “<box>” (the content of the box is a math formula);
- “box:<txt>” attaches a “<box>” 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 on the edge;
- “`bend:<angle>`” bends 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`.

Transformation arrows may also be placed within the graph by means of the “`v0=>v1`” syntax. The arrow is positioned precisely between the two vertices. An arrow may likewise extend from a vertex to the right, as in “`v3=>`”, or from the left towards the vertex, as in “`=>v5`”. Whenever the arrow is to remain further from the vertex than usual, additional “`=`” symbols may be used, as in “`==>v0`”.

A marker may also be placed to the left of a vertex by means of the “`v5!A`” syntax, where “`v5`” denotes the vertex and “`A`” the text within the marker. Such markers prove useful whenever several graphs are juxtaposed within a single figure to illustrate the transformation of one graph into another. The distance between the vertex and its marker may be increased by additional exclamation marks; for instance, “`v5!!!A`” produces a distance three times the original.

A clone of an existing vertex together with all of its dependants may be created by means of the “`v0+a`” syntax: a copy of “`v0`” is thereby produced and named “`v0a`”. See the example below.

Note that unrecognised markers are silently ignored, without any error reporting.

`\eolang` A no-argument command `\eolang` is also provided for printing the name of
`\phic` the EO language. It is sensitive to the `anonymous` package option, rendering itself
`\xmirl` differently to support double-blind submissions. The `\phic` command similarly
prints the name of φ -calculus and is likewise sensitive to `anonymous` mode. The
`\xmirl` macro 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 representation.</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 \xmirl{ --- \ 9 its XML-based representation. 10 \end{document} </pre>
---	--

Without the `anonymous` option, no orange colour is applied:

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 \xmirc{} --- \\
9 its XML-based representation.
10 \end{document}

```

`\phiWave`
`\phiDotted`

Several simple commands are defined for the rendering of 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 $x$ is an identifier and $y$ 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` Should text be placed above or beneath an arrow, `\phi0set` and `\phiUset` are
`\phiUset` employed, respectively:

When the names of attributes and their values are immaterial, an arrow with a star is employed, for example:

$\langle \rightarrow^* \rangle$.

```

6 When the names of attributes and their
7 values are immaterial, an arrow
8 with a star is employed, for example:
9 \begin{phiuation*}
10 [[ \phi0set{*}{->} ]]
11 \end{phiuation*}

```

`\phiMany` On occasion, it is convenient to simplify the description of an object (the typesetting is somewhat imperfect, owing not to the package itself but rather to [this](#)):

The expression $\langle 1 \rightarrow x_1, 2 \rightarrow x_2, \dots, \alpha_n \rightarrow x_n \rangle$ and expression $\langle \alpha_i \xrightarrow{i=1}^n x_i \rangle$ 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` The `phiuation` and `sodg` environments cannot be used directly within `tabular`
`\sodgSaveTo` or any other environment or command, since they are “verbatim” environments. The `\phiSaveTo` and `\sodgSaveTo` commands address this difficulty. They are placed immediately before `\begin{phiuation}` or `\begin{sodg}`, respectively—the content of the equation or graph is then not rendered but saved to a file. Subsequently, within `tabular`, it may be retrieved by means of the `\input` macro (the `\parbox` should not be omitted):

Free:	$x \mapsto \emptyset$	5	<code>\phiSaveTo{a}</code>
Bound:	$x \mapsto [\Delta \mapsto 42]$	6	<code>\begin{phiuation*}</code>
		7	<code>[[x -> [[D>42]]]]</code>
		8	<code>\end{phiuation*}</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` Certain content may need to be concealed by means of the `anonymous` package option. The `\eoAnon` command serves this purpose. It accepts two parameters: one mandatory and one optional. The mandatory parameter denotes the content to be displayed; the optional parameter, the substitution rendered whenever the `anonymous` package option is set.

2 Package Options

`tmpdir` The default location for temporary files is `_eolang`. This may be altered by means of the `tmpdir` package option:

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

`nodollar` The special treatment of the dollar sign may be disabled by means of the `nodollar` package option:

`\usepackage[nodollar]{eolang}`

`anonymous` The `\eolang`, `\xmirc`, and `\phic` commands may be anonymised by means of the `anonymous` package option (each invokes the `\eoAnon` command mentioned earlier):

`\usepackage[anonymous]{eolang}`

`noshell` Any interaction with the shell may be prohibited by means of the `noshell` option. This proves useful whenever the document is dispatched for external processing, and assurance is required that compilation will not fail on account of shell errors:

`\usepackage[noshell]{eolang}`

3 More Examples

The `phiuation` environment treats line endings as signals to begin new lines within the formula. Should this behaviour be undesirable, and the subsequent line be parsed instead as a continuation of the current line, a single backslash may be employed as illustrated below:

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

The following illustrates the use of `\dfrac` from [amsmath](#) for large inference rules, by means of `\begin{split}` and `\end{split}`:

$\frac{x \mapsto [\varphi \mapsto y, z \mapsto 42, \quad 0/g \mapsto \emptyset, 1/\text{foo} \mapsto 42]}{x \mapsto [\varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto [\psi \mapsto \text{hello}(12)], \alpha_1 \mapsto 42)]} \text{R2.}$	<pre> 6 \begin{phiquation*} 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{phiquation*} </pre>
--	---

The `matrix` environment may likewise be employed to group several 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{phiquation*} 6 foo -> \left\{\begin{matrix} \backslash 7 ? \backslash 8 [[L> ^ \times \$. \alpha_0]] \backslash 9 [[D> 42]] \backslash 10 \end{matrix}\right\} 11 \end{phiquation*} </pre>
--	--

The `cases` environment is likewise supported:

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

The `phiquation` environment may likewise be employed in conjunction with the [acmart](#) package:

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

The `\label` command may be used within the `phiquation` environment (note how the bespoke parsing of math formulas may be suppressed by enclosing the “4”

number in curly brackets):

<div style="border: 1px solid black; padding: 5px;"> <p>The discriminant may be computed by means of the following simple formula:</p> $\Delta = b^2 - 4ac. \quad (1)$ <p>Eq. 1 finds wide application in number theory and polynomial factoring.</p> </div>	<pre> 6 The discriminant may be computed by 7 means of the following simple formula: 8 \begin{phiuation} 9 D = b{^2} - {4}ac. 10 \label{d} 11 \end{phiuation} 12 Eq.\ref{d} finds wide application in 13 number theory and polynomial factoring.</pre>
---	--

Comments may be appended to equations by means of the `&&` command (note that the text within `\text{}` is not processed but treated as plain text):

<div style="border: 1px solid black; padding: 5px;"> $\begin{array}{l} \alpha_0 \mapsto x \\ \alpha_0 \mapsto \emptyset \\ x(\Delta \mapsto 42) \end{array}$ <div style="display: flex; justify-content: space-between; padding: 0 10px;"> <div>This is formation</div> <div>Abstraction</div> <div>Application</div> </div> </div>	<pre> 6 \begin{phiuation*} 7 [[~0 -> x]] && \text{This is formation} 8 [[~0 -> ?]] && \text{Abstraction} 9 x(D>42) && \text{Application} 10 \end{phiuation*}</pre>
--	---

Even without the `nodollar` package option, the ordinary parsing of the dollar sign remains accessible by means of the `\(...\)` syntax:

<div style="border: 1px solid black; padding: 5px;"> <p>The object formation $\alpha_0 \mapsto x$ may be replaced by the formula $Q \times a^2$.</p> </div>	<pre> 6 The object formation $\alpha_0 \mapsto x$ 7 may be replaced by the formula 8 $Q \times a^2$.</pre>
---	--

The `phiuation` environment automatically aligns formulas at the first arrow whenever only left-aligned formulas are present:

<div style="border: 1px solid black; padding: 5px;"> $\begin{array}{l} x(\pi) \mapsto [\lambda \mapsto f_1], \\ x(a, b, c) \mapsto [\alpha_0 \mapsto \emptyset, \lambda \mapsto \text{Foo}, \varphi \mapsto \Phi \text{hello}(\zeta), x \mapsto \text{false}], \\ \Delta = 43-09, \\ x(y) \equiv x(\alpha_0 \mapsto y). \end{array}$ </div>	<pre> 5 \begin{phiuation*} 6 x(\pi) -> [[\lambda \mapsto f_1]], \\\ 7 x(a,b,c) -> [[~0 -> ?, L> Foo, \ 8 @ -> Q.hello(\$), x -> false]], \\\ 9 \Delta = 43-09 , 10 x(y) == x(~0 -> y). 11 \end{phiuation*}</pre>
--	--

Should no line in `phiuation` be indented, all formulas are centred:

<div style="border: 1px solid black; padding: 5px;"> $\begin{array}{l} b \mapsto \emptyset, \\ \varphi \mapsto \text{TRUE}, \Delta \mapsto 42, \\ \psi = \langle \pi, 42 \rangle. \end{array}$ </div>	<pre> 5 \begin{phiuation*} 6 [[b -> ?]], 7 [[@ -> TRUE, \Delta ..> 42]], \\\ 8 \psi = << \pi, 42 >>. 9 \end{phiuation*}</pre>
--	--

A “manual splitting” mode may be employed in the `phiuation` environment by commencing the body with `\begin{split}`:

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

Whenever a percentage sign is required, it must be prefixed with a backslash:

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

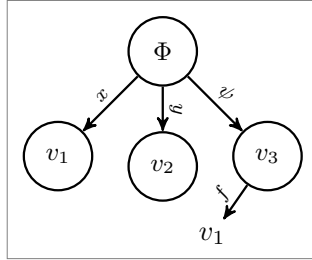
A copy of a vertex may be made together with its descendants:

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

A copy may itself be copied in turn:

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

“Broken” edges are supported by means of the “**break**” attribute of an edge. The attribute requires a value denoting the percentage of the path between vertices that the arrow is to traverse (the value cannot exceed 80 nor fall below 20). This proves convenient whenever not all edges can be accommodated within the graph, for example:

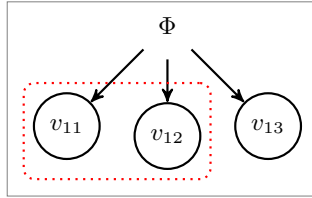


```

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}

```

[TikZ](#) commands may be added to a `sodg` graph, for example:

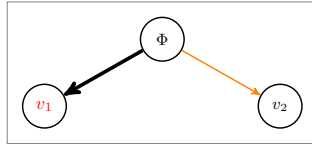


```

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}

```

The `TikZ` style may be modified directly (note that `style:` must remain 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 phiquation 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]);
63 my $tex; { local $/; $tex = <$fh>; }
64 print "% This file is auto-generated by eolang.sty 0.23.1\n";
65 print '% There are ', length($tex),
66 ' chars in the input: ', $ARGV[1], "\n";
67 print '% ---', "\n";
68 if (index($tex, "\t") >= 0) {
69   print "TABS are prohibited!";
70   exit 1;
71 }
72 my @lines = split (/\\n/g, $tex);
73 foreach my $t (@lines) {
74   print '% ', $t, "\n";
75 }
76 print '% ---', "\n";
77 $tex =~ s/(?<!\\\)%.*\\n\\n/g;
78 $tex =~ s/^\\s+|\\s+$//g;
79 my $splitting = $tex =~ /^\\begin\\{split\\}/;
80 if ($splitting) {
81   print '% The manual splitting mode is ON since \\begin{split} started the text' . "\n";
82 }
83 my $indents = $tex =~ /\\n +/g;
84 my $gathered = (0 == $indents);
85 if ($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 }
96 my $align = 0;
97 print '% The "align" is NOT used by default' . "\n";
98 if (index($tex, '&&') >= 0) {
99   $macro =~ s/equation/align/g;
100   $align = 1;
101   print '% The "align" is used because of && seen in the text' . "\n";
102 }
103 if ($macro ne 'phiq') {

```

```

104 if (not $splitting) {
105     $tex =~ s/\\\\\\n/n/g;
106     $tex =~ s/\\\\n\\s*/g;
107 }
108 $tex =~ s/\n*(\\label\{[^}\]+\\})\n*/1/g;
109 $tex =~ s/\n{3,}/\n/g;
110 }
111 my @texts = ();
112 sub trep {
113     my ($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]+|*)\V(\[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\\}/\\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/\$/\\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{,}\\;/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 `phiuation` 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

```

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

```

263 \makeatletter\newcommand\eolang@process[1]{
264   \def\hash{\eolang@mdfive
265     {\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}}-\the\inputlineno}%
266   \eolang@ifabsent
267     {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation-post}}
268     {%
269     \iexec[null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}"
270      "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation}"}%
271     \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
272     \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation-post}]{
273       perl "\eolang@tmpdir/\jobname-phi.pl"
274       '#1'
275       "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation}"
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{phiuation*}%
284 {\catcode'\|=12 \VerbatimEnvironment%
285 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
286 \begin{VerbatimOut}
287   {\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}}}
288 {\end{VerbatimOut}\eolang@process{equation*}}
289 %
290 \newenvironment{phiuation}%
291 {\catcode'\|=12 \VerbatimEnvironment%
292 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
293 \begin{VerbatimOut}
294   {\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}}}
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 \begingroup
319 \catcode'\$=\active
320 \protected\gdef$#1$#{\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     $cmd =~ s/(?<!\n)%.*//g;
396     my ($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 } elseif (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 } elseif (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.' . $c .
531     ' (+ ' . (0+@extra) . ' lines -> ' .
532     (0+@cmds) . ' lines total)';
533 } elsif ($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{picture}%', "\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 `picture`:

```

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               ($(\F)!1.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)

```

```

673         -- ++(\t: 2*\d em);
674     };
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 for printing EO, the name of the language.

```

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

```

\phic Then, we define a simple supplementary command for printing φ -calculus, the name of the formal apparatus.

```

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

```

\xmirt Then, we define a simple supplementary command for printing XMIR, the name of the XML-based format of program representation.

```

758 \newcommand\xmirt{%
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}}}
```

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

```

764 \makeatletter
765 \newcommand{\phi0set}[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   \phi0set{\#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		phiuation , another one for	
	General: First draft.	sodg	17
0.0.2		0.12.1	
	sodg : The environment phigure	- sodg.pl : The bug is fixed related	
	renamed to sodg for the sake of	to the formatting of indexes of	
	better semantics. The graph in	vertices.	17
	the picture is solely a SODG	0.13.0	
	graph, that's why the name	- phi.pl : Parsing of QQ into	
	sodg is better.	\dot{\Phi} implemented. . . .	11
	24	0.14.0	
	- phi.pl : New symbol added for	- sodg.pl : The edgeless tag of a	
	basket slots	vertex removes the border of it. 17	
	11	0.15.0	
	Parsing of the symbols " @ ," " ^ ,"	- sodg.pl : The style tag of	
	and " & " enabled (\varphi ,	vertices and edges.	17
	\rho , and \sigma)	0.16.0	
	11	phiuation : The processing of	
	The symbols " [" and "] "	phiuation data is done only if	
	replaced with " [[" and "]] " for	it's the first time processing,	
	abstract object brackets,	otherwise cache is used, thus	
	because they conflicted with	making processing faster. . . .	16
	normal square brackets	sodg : The processing of sodg data	
	11	is done only if it's the first time	
	\phiq : Parsing of additional	processing, otherwise cache is	
	symbols enabled.	used, thus making processing	
	16	faster.	24
	- sodg.pl : The Perl file now has a	0.17.0	
	fixed name, which doesn't	\eolang@ifabsent : A new	
	depend on the name of the TeX	supplementary	
	job. This file may be shared	eolang@ifabsent command	
	among jobs, no need to make it	added	15
	uniquely named.	0.18.0	
	17	\eolang@ifabsent : The noshell	
0.1.0		package option added in order	
	General: Parsing of package options	to enable complete prohibition	
	introduced.	of shell interactions.	15
	11	0.18.3	
	\eolang : New command \eolang	\eolang@tmp : A new	
	added to print the name of the	supplementary eolang@tmp	
	language in both normal and	command added, to generate	
	the anonymous mode of	temporary file names.	15
	acmart	0.19.0	
	25	- phi.pl : Parsing of T into \bot	
	\eolang@mdfive : New	implemented.	11
	supplementary command added	0.2.0	
	to calculate MD5 sum of a file. 11	- phi.pl : Numbers automatically	
	- phi.pl : A new Perl script	render as \texttt . No need to	
	" eolang-phi.pl " added for	use vertical bars around them	
	parsing of phi expressions. . . .	anymore.	11
	11	- sodg.pl : The content of the atom	
	\phic : New command \phic prints	and the data boxes is parsed	
	the name of φ -calculus in both		
	normal and the anonymous		
	mode of acmart		
	25		
	\phiDotted : New command		
	\phiDotted added to denote a		
	link to a special attribute. . . .		
	26		
	- sodg.pl : There are two Perl		
	scripts now: one for		

automatically as formulas and numbers, respectively.	17	New syntax introduced that allows to make clones of vertices and all their dependants.	17
<code>\xmir</code> : New command <code>\xmir</code> prints XMIR in both normal and the anonymous mode of <code>acmart</code>	25	Now edges may have the <code>break</code> attribute, to make them shorter.	17
0.20.0		0.6.0	
- <code>phi.pl</code> : Parsing of 0 into <code>\alpha_0</code> implemented.	11	General: Package option <code>nocomments</code> added in order to enable comments suppression in temporary <code>.tex</code> files (may be pretty important for <code>.dtx</code> documents).	11
0.20.2		- <code>sodg.pl</code> : The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations.	17
- <code>phi.pl</code> : Formatting of numbers fixed, now it works in any place inside an expression.	11	0.7.0	
0.21.0		<code>nodollar</code> : Now it is possible to use dollar sign instead of the <code>\phiq</code> command.	17
- <code>phi.pl</code> : Automated formatting of <code>TRUE</code> and <code>FALSE</code> removed.	11	- <code>phi.pl</code> : New syntax sugar for Φ , just using capital “Q” is enough.	11
<code>\phiTerminal</code> : New command <code>\phiTerminal</code> added, to highlight all terminals in phi expressions.	26	Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols.	11
0.22.0		Text in quotes is automatically converted to <code>\texttt</code>	11
- <code>phi.pl</code> : Automated formatting of <code>D</code> and <code>L</code> added.	11	0.8.0	
0.23.0		General: The <code>anonymous</code> package option added.	11
- <code>phi.pl</code> : Parsing of <code>QQ</code> into <code>\dot{\Phi}</code> removed.	11	- <code>phi.pl</code> : Inside <code>phiquation</code> any text inside the <code>\text</code> macro is not processed.	11
0.3.0		<code>\phi0set</code> : New commands <code>\phi0set</code> and <code>\phiUset</code> help position text over and under an arrow.	26
<code>\eolang@lineno</code> : New counter for protecting <code>lineno</code>	11	<code>\phiSaveTo</code> : The output of the <code>phiquation</code> environment can be redirected to a file.	15
- <code>phi.pl</code> : New arrow added, that looks like <code>\leadsto</code>	11	- <code>sodg.pl</code> : The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.	17
<code>\phiWave</code> : New command <code>\phiWave</code> added to denote a link to a multi-layer attribute.	25	<code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file.	24
0.4.0		0.9.0	
- <code>sodg.pl</code> : 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	<code>\eoAnon</code> : New command <code>\eoAnon</code> added.	25
Relative positioning of vertices fixed.	17	- <code>phi.pl</code> : Proper handling of the <code>matrix</code> environment.	11
0.5.0		<code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code>	26
- <code>phi.pl</code> : Automated formatting of <code>TRUE</code> and <code>FALSE</code> added.	11		
<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow.	26		
<code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket.	26		
- <code>sodg.pl</code> : It is possible to use TikZ commands inside the <code>sodg</code> environment.	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	C	\eolang@process ...
\\$ 176, 301, 315, 319, 322, 722, 735	\catcode .. 284, 291, 301, 315, 319, 322, 715, 722, 735 263, 288, 295
\% ... 276, 306, 312, 732	\clean 299, 300, 306	\eolang@sodgSaveTo 712, 733, 737
\(... 197, 741, 757, 759	\clipbox 790, 794	\eolang@tmp 239, 265, 267,
\) ... 214, 742, 757, 759	\closein 233, 618	269, 270, 272,
* 142, 173	\color 642, 778	275, 287, 294, 303, 305, 307, 308, 309, 311, 718, 721, 724, 726, 727, 729, 731
\+ ... 339, 475, 506, 512		\eolang@tmpdir 11, 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
\- 757	D	\ex@ 767, 771
\-phi.pl <u>48</u>	\d 195, 661, 662, 672, 673, 683, 684, 693, 694	F
\-sodg.pl <u>324</u>	\dashrightarrow ... 792	\F 657, 659, 660, 672, 679, 681, 682, 693
\. ... 133, 143, 187, 339	\def 237, 264, 280, 300, 712, 720, 737, 741, 742, 743, 744	\FancyVerbLine <u>621</u>
\/ 141, 142	\Delta 557	G
\? 180	\detokenize 299	\gdef 320
\[..... 141, 178	\dimen 786, 787, 789, 791, 798	H
\{ 79, 108, 131, 132, 133, 134, 135, 136, 137, 138, 139, 146, 147, 173, 177, 191, 192, 195	\dp 798	\hash 264, 267, 270, 272, 275, 300, 303, 308, 309, 311, 720, 724, 727, 729, 731
\} 79, 108, 146, 147, 173, 195	\draw . 408, 636, 666, 688	\hbox 768, 772
\] 141, 179	E	\height ... 791, 794, 795
\^ 177	\E 173, 487, 495, 503, 506, 512, 515, 516, 517, 518	\hss 793
\ 142, 193, 194, 284, 291, 344, 715	\end ... 218, 220, 223, 226, 288, 295, 604, 611, 637, 719	\ht 798
Numbers	\endinput . 225, 610, 803	I
\2 143, 152, 363	\eoAnon <u>739</u> , 755, 757, 759	\I 657, 658, 660, 672, 679, 680, 682, 693
\3 143	\eolang <u>754</u>	
\4 143	\eolang@anonymouse 15, 748	
A	\eolang@ifabsent .. . 242, 266, 302, 723	
\a 661, 664, 672, 683, 686, 693	\eolang@lineno <u>43</u>	
\active 315, 319, 322, 735	\eolang@mdfive <u>44</u> , 264, 720	
\alpha 743, 757	\eolang@nocomments . . 13, 276, 312, 732	
\AtBeginDocument .. 322	\eolang@nodollar 14, 301, 315, 317	
B	\eolang@noshell 5, 16, 21, 27, 49, 250, 325	
\Bbbk 3	\eolang@phiSaveTo 237, 277, 280	
\begin 60, 81, 199, 202, 204, 286, 293, 336, 391, 633, 717		
\box 799		

<code>\iexec</code>	33,	<code>\mathpalette</code>	783	<code>\protected</code>	320
	269, 272, 305,	<code>\mathrel</code> 761, 766, 770, 783			
	307, 309, 726, 729	<code>\message</code> . 24, 28, 35,			
<code>\ifdefined</code>	5,	50, 56, 227, 230,		Q	
	21, 27, 49, 250,	247, 251, 254,		<code>\Q</code>	173, 487, 495,
	276, 277, 301,	271, 313, 326,			503, 506, 512,
	312, 315, 317,	332, 612, 615, 728			515, 516, 517, 518
	325, 732, 733, 748	<code>\mspace</code>	761	R	
<code>\ifeof</code>	55, 331			<code>\relax</code>	3, 280, 737, 783
<code>\IfFileExists</code>	22, 244	N		<code>\RequirePackage</code>	1,
<code>\ifluatex</code>	12, 240	<code>\newcommand</code>			2, 3, 4, 5, 6, 7, 8,
<code>\ifnum</code>	32, 253	46, 237, 240,		21, 44, 297, 622,
<code>\ifxetex</code>	12, 240		243, 263, 298,		739, 777, 779, 780
<code>\input</code>	248		712, 754, 756,	<code>\resizebox</code>	791
<code>\inputlineno</code> 265, 271,			758, 760, 762,	<code>\rightarrow</code> 786, 787, 795	
	300, 313, 721, 728		765, 769, 774,		
			776, 778, 782, 784	S	
J		<code>\newcounter</code>	43	<code>\sbox</code>	788
<code>\jobname</code>	23,	<code>\newenvironment</code>		<code>\scriptscriptstyle</code>	768, 772
	24, 28, 33, 35, 51,	283, 290, 632, 714	<code>\scriptsize</code>	707, 710
	54, 57, 60, 228,	<code>\NewExpandableDocumentCommand</code>	747	<code>\scshape</code>	763
	231, 265, 267,			<code>\setcounter</code> 279, 285,	
	269, 270, 272,	<code>\node</code>	447, 458,		292, 621, 716, 736
	273, 275, 287,		461, 464, 471, 534	<code>\settoheight</code>	787
	294, 303, 305,	<code>\nodollar</code>	317	<code>\settowidth</code>	786
	307, 308, 309,	<code>\noindent</code>	633	<code>\sffamily</code>	755, 763
	310, 311, 327,			<code>\ShellEscapeStatus</code>	
	330, 333, 336,	O			32, 253
	613, 616, 718,	<code>\openin</code>	54, 330	<code>\small</code>	642, 645
	721, 724, 726,			<code>\sodg</code>	714
	727, 729, 730, 731	P		<code>\sodgSaveTo</code>	711
K		<code>\pdf@filemdfivesum</code>	46	<code>\StrSubstitute</code>	299
<code>\kern</code>	767, 771	<code>\pdf@mdfivesum</code>	300	<code>\TeX</code>	517
		<code>\pdfstringdefDisableCommands</code>	740	T	
L		<code>\pgfkeys</code>	9	<code>\t</code>	68, 369, 661,
<code>\lambda</code>	569	<code>\Phi</code>	354		663, 672, 673,
<code>\leadsto</code>	761	<code>\phic</code>	756		683, 685, 693, 694
<code>\limits</code>	766, 770	<code>\picture</code>	632	<code>\text</code>	557, 763
M		<code>\phiDotted</code>	557, 569, 779	<code>\textcolor</code>	749
<code>\m@th</code>	786, 787, 792, 795	<code>\phiDotted@</code>	783, 784	<code>\textnormal</code>	558
<code>\makeatletter</code>	20,	<code>\phiEOL</code>	776	<code>\texttt</code>	558
	43, 45, 48, 236,	<code>\phiMany</code>	774	<code>\the</code>	265, 271,
	239, 242, 263,	<code>\phiOset</code>	764, 775		300, 313, 721, 728
	298, 324, 711,	<code>\phiq</code>	297, 320	<code>\tikz</code>	622
	714, 746, 764, 781	<code>\phiquation</code>	263	<code>\tikzinputsegmentfirst</code>	658, 680
<code>\makeatother</code>	42,	<code>\phiSaveTo</code>	236	<code>\tikzinputsegmentlast</code>	659, 681
	43, 47, 235, 238,	<code>\phiSlot</code>	762	<code>\tikzmath</code>	656, 678
	241, 262, 296,	<code>\phiTerminal</code> 761, 777, 783		<code>\tikzset</code>	650
	316, 620, 713,	<code>\phiUset</code>	769, 775	<code>\tikzstyle</code>	
	738, 753, 773, 802	<code>\phiWave</code>	760		638, 641, 644,
<code>\makebox</code>	789	<code>\pi</code>	412		
<code>\mapstochar</code>	761, 783	<code>\ProcessPgfPackageOptions</code>	19		
<code>\mathop</code>	766, 770				

646, 647, 649, 703, 704, 705, 708		V		W
\ttfamily 642	\v ... 657, 660, 679, 682	\value 279,	\width 790, 794	
\tw@ 787, 798	285, 292, 716, 736			X
	\varphi 744, 757		\xmir 758	
	\vbox 767, 771		\xrightarrow 763	
U	\VerbatimEnvironment			
\usetikzlibrary 284, 291, 715			Z
..... 623, 624,	\vss 768, 772			
625, 626, 627,	\vx .. 663, 664, 685, 686	\z@ 786, 788,		
628, 629, 630, 631	\vy 663, 685	789, 791, 798, 799		