



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

Yegor Bugayenko
yegor256@gmail.com

2026-09-29, 0.25.0

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`),
- “:” maps to “ $\vdash$ ” (`\phiTerminal{:}`),
- “|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->x`. This is rendered as $\alpha_0|g \mapsto x$. Fixed-width words are likewise admissible, for instance `\phiq{0/|f|->x}` renders as “ $\alpha_0|f \mapsto x$ ”. An asterisk may replace the number, whereby `\phiq{* /g->x}` renders as “ $\alpha_*|g \mapsto x$ ”.

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.

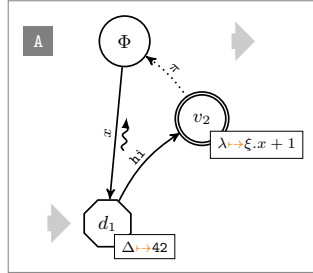
Texts within double quotes are likewise rendered in fixed-width font.

`\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:|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 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”.

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

```

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>
		6	<code>\begin{phiuation*}</code>
Bound:	$x \mapsto [\Delta \mapsto 42]$	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}
```

The dollar sign is turned into an active character only at the beginning of the document, so that packages loaded after `eolang` still read `$` at its normal math-shift catcode throughout the preamble. As a consequence, `$.$. $` written inside the body of a preamble-defined macro (for example, a `\newcommand`) is *not* converted to `\phiq` automatically. In that situation, use the `\phiq{...}` command directly, since it is catcode-independent and self-wraps its argument.

`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}
```

`nocolor` The package paints anonymised content and φ -calculus terminals orange. The `nocolor` option suppresses this, so that nothing is ever coloured — useful for greyscale printing and for venues that forbid colour:

```
\usepackage[nocolor]{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 `phiquation` 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:

$\frac{x \mapsto [\varphi \mapsto y] \quad y \mapsto [z \mapsto 42]}{x.z \mapsto \perp} R1$	<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>
---	--

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

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

The `matrix` environment may likewise be employed to group several lines:

$foo \mapsto \left\{ \begin{array}{c} \emptyset \\ [\lambda \mapsto \rho \times \xi.\alpha_0] \\ [\Delta \mapsto 42] \end{array} \right\}$	<pre> 5 \begin{phiquation*} 6 foo -> \left{\begin{matrix} \ 7 ? \\\ 8 [[L> ^ \times \$. \alpha_0]] \\\ 9 [[D> 42]] \ 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} \ 7 [v_2, @ -dtzd> 42] \\\ 8 [v_{33}] \ 9 \end{cases} 10 \end{phiquation*} 11 \end{document} </pre>
---	---

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

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

The discriminant may be computed by means of the following simple formula: $\Delta = b^2 - 4ac. \quad (1)$ Eq. 1 finds wide application in number theory and polynomial factoring.	<pre> 6 The discriminant may be computed by 7 means of the following simple formula: 8 \begin{phiquation} 9 D = b^{2} - {4}ac. 10 \label{d} 11 \end{phiquation} 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):

<table> <tr> <td>$\alpha_0 \mapsto x$</td> <td>This is formation</td> </tr> <tr> <td>$\alpha_0 \mapsto \emptyset$</td> <td>Abstraction</td> </tr> <tr> <td>$x(\Delta \mapsto 42)$</td> <td>Application</td> </tr> </table>	$\alpha_0 \mapsto x$	This is formation	$\alpha_0 \mapsto \emptyset$	Abstraction	$x(\Delta \mapsto 42)$	Application	<pre> 6 \begin{phiquation*} 7 [[~0 -> x]] && \text{This is formation} 8 [[~0 -> ?]] && \text{Abstraction} 9 x(D>42) && \text{Application} 10 \end{phiquation*} </pre>
$\alpha_0 \mapsto x$	This is formation						
$\alpha_0 \mapsto \emptyset$	Abstraction						
$x(\Delta \mapsto 42)$	Application						

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

The object formation $\alpha_0 \mapsto x$ may be replaced by the formula $Q \times a^2$.	<pre> 6 The object formation \$[[~0->x]]\$ 7 may be replaced by the formula 8 \((Q \times a^2 \)). </pre>
---	---

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

```

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

```

```

5 \begin{phiquation*}
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{phiquation*}

```

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

```


$$[b \mapsto \varnothing],$$


$$[\varphi \mapsto \text{TRUE}, \Delta \mapsto 42],$$


$$\psi = \langle \pi, 42 \rangle.$$


```

```

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

```

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

```

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

```

```

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

```

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

```

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

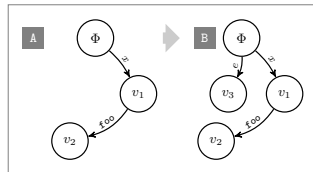
```

```

5 \begin{phiquation*}
6 x -> sprintf("Hello, \%s!", name)
7 \end{phiquation*}
8 \end{document}

```

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

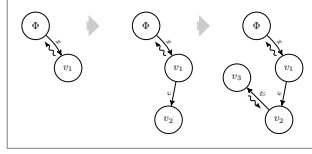


```

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}

```

A copy may itself be copied in turn:

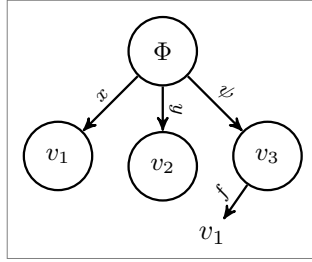


```

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}

```

“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:

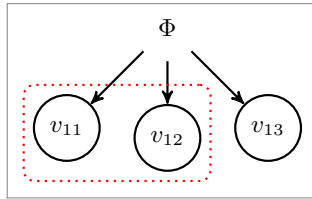


```

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:

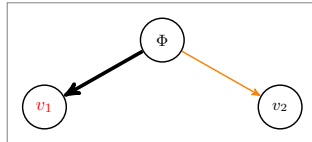


```

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 nocolor/.store in=\eolang@nocolor,
16 anonymous/.store in=\eolang@anonymous,
17 noshell/.store in=\eolang@noshell,
18 tmpdir
19 }
20 \ProcessPgfPackageOptions{/eolang}
```

`\eolang@ink` Then, we define an internal command that paints its argument orange, leaving it untouched when the `nocolor` package option is set:

```
21 \RequirePackage{xcolor}
22 \makeatletter
23 \ifdefined\eolang@nocolor
24 \newcommand\eolang@ink[1]{#1}
25 \else
26 \newcommand\eolang@ink[1]{\color{orange}#1}
27 \fi
28 \makeatother
```

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

```
29 \makeatletter
30 \ifdefined\eolang@noshell\else\RequirePackage{shellesc}\fi
31 \IfFileExists
32 {\eolang@tmpdir/\jobname}
33 {\message{\eolang: Temporary directory "\eolang@tmpdir/\jobname"
34 already exists^^J}}
35 {
```

```

36 \ifdefined\eolang@noshell
37 \message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
38 is not created, because of the "noshell" package option,
39 most probably the compilation will fail later^^J}
40 \else
41 \ifnum\ShellEscapeStatus=1
42 \iexec[null]{mkdir -p "\eolang@tmpdir/\jobname"}
43 \else
44 \message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
45 is not created, because -shell-escape is not set, and
46 it doesn't exist, most probably the compilation
47 will fail later^^J}
48 \fi
49 \fi
50 }
51 \makeatother

```

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

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

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

```

53 \RequirePackage{pdftexcmds}
54 \makeatletter
55 \newcommand\eolang@mdfive[1]{\pdf@filemdfivesum{#1}}
56 \makeatother

```

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

```

57 \makeatletter
58 \ifdefined\eolang@noshell
59 \message{eolang: Perl script is not going to be created,
60 at "\eolang@tmpdir/\jobname-phi.pl" because of the "noshell"
61 package option^^J}
62 \else
63 \openin 15=\eolang@tmpdir/\jobname-phi.pl
64 \ifeof 15
65 \message{eolang: Perl script is going to be created,
66 because it is absent at "\eolang@tmpdir/\jobname-phi.pl",
67 but if -shell-escape is not set, the compilation will
68 most likely fail now^^J}
69 \begin{VerbatimOut}{\eolang@tmpdir/\jobname-phi.pl}
70 $macro = $ARGV[0];
71 open(my $fh, '<', $ARGV[1]);
72 my $tex; { local $/; $tex = <$fh>; }
73 print "% This file is auto-generated by eolang.sty 0.25.0\n";
74 print '% There are ', length($tex),
75 ' chars in the input: ', $ARGV[1], "\n";
76 print '% ---', "\n";
77 if (index($tex, "\t") >= 0) {
78 print "TABS are prohibited!";
79 exit 1;
80 }
81 my @lines = split (/\\n/g, $tex);
82 foreach my $t (@lines) {

```

```

83 print '% ', $t, "\n";
84 }
85 print '% ---', "\n";
86 $tex =~ s/(?<!\n)%.*\n\n/g;
87 $tex =~ s/^\s+|\s+$//g;
88 my $splitting = $tex =~ /^\\begin\{split\}/;
89 if ($splitting) {
90   print '% The manual splitting mode is ON since \begin{split} started the text' . "\n";
91 }
92 my $indents = $tex =~ /\n +/g;
93 my $gathered = (0 == $indents);
94 if ($gathered) {
95   if ($splitting) {
96     print '% The "gathered" is NOT used because of manual splitting' . "\n";
97     $gathered = 0;
98   } else {
99     print '% The "gathered" is used since all lines are left-aligned' . "\n";
100   }
101 } else {
102   print '% The "gathered" is NOT used because ' .
103     $indents . " lines are indented\n";
104 }
105 my $align = 0;
106 print '% The "align" is NOT used by default' . "\n";
107 if (index($tex, '&&') >= 0) {
108   $macro =~ s/equation/align/g;
109   $align = 1;
110   print '% The "align" is used because of && seen in the text' . "\n";
111 }
112 if ($macro ne 'phiq') {
113   if (not $splitting) {
114     $tex =~ s/\\\\\n\n\n/g;
115     $tex =~ s/\\\\\n\s*//g;
116   }
117   $tex =~ s/\n*(\\label\[^\]]+\})\n*/\1/g;
118   $tex =~ s/\n{3,}/\n\n/g;
119 }
120 my @texts = ();
121 sub trep {
122   my ($s) = @_ ;
123   my $open = 0;
124   my $p = 0;
125   for (; $p < length($s); $p++) {
126     $c = substr($s, $p, 1);
127     if ($c eq '}') {
128       if ($open == 0) {
129         last;
130       }
131       $open--;
132     }
133     if ($c eq '{') {
134       $open++;
135     }
136   }

```

```

137 push@texts, substr($s, 0, $p));
138 return '{TEXT' . (0+@texts - 1) . '}' . substr($s, $p + 1);
139 }
140 $tex =~ s/\\text\{(.+)/trep("$1")/ge;
141 $tex =~ s/(?<=[\s,>()]{0-9A-F}{2}(?:-[0-9A-F]{2})+)(?!\{)/|1|/g;
142 $tex =~ s/(?<=[\s,>()]{0-9}++(?:\.[0-9]+)?)(?!\{)/|1|/g;
143 $tex =~ s/([^\{a-zA-Z0-9\\]|~)Q(?:[a-zA-Z0-9])/1 \\phiTerminal{\\Phi}}/g;
144 $tex =~ s/([^\{a-zA-Z0-9\\]|~)T(?:[a-zA-Z0-9])/1 \\phiTerminal{\\bot}}/g;
145 $tex =~ s/([^\{a-zA-Z0-9\\]|~)D>/1\\phiTerminal{\\Delta}} ..>/g;
146 $tex =~ s/([^\{a-zA-Z0-9\\]|~)L>/1\\phiTerminal{\\lambda}} ..>/g;
147 $tex =~ s/([^\{a-zA-Z0-9\\]|~)D(?:[a-zA-Z0-9])/1\\phiTerminal{\\Delta}}/g;
148 $tex =~ s/([^\{a-zA-Z0-9\\]|~)L(?:[a-zA-Z0-9])/1\\phiTerminal{\\lambda}}/g;
149 $tex =~ s/"([~"]+)"|"1"/g;
150 $tex =~ s/(?::~|(?<=[\s]([\\[,.>|/]))([a-zA-Z][a-zA-Z0-9]+)(?=[\s]([\\[,.-]|$))/|1|/g;
151 $tex =~ s/([_~]|~) ([0-9]+|*)\\(\\{[a-z]+|\\{[a-z]+|\\{
152 (->|\\.\\.>|~>|:=)/1\\alpha_{2}\\vert{\\3\\space}{4}/xg;
153 if ($macro ne 'phiq') {
154   if (not $splitting) {
155     $tex =~ s/\\begin{split}\\n/\\begin{split}&/g;
156     $tex =~ s/\\n\\s*\\end{split}/\\end{split}/g;
157     $tex =~ s/\\n\\n/\\&/g;
158     $tex =~ s/\\n/\\phiEOL{\\n}&/g;
159     $tex =~ s/\\\\$/&/g;
160     $tex =~ s/\\\\\\\\/\\n/g;
161     $tex =~ s/([~&\\s])s{2}([~\\s])/1 2/g;
162     $tex =~ s/s{2}/ \\quad}/g;
163     $tex = '&' . $tex;
164   }
165   my $lead = '[~\\s]+\\s(?:->|:=|==)\\s';
166   my @leads = $tex =~ /&{$lead}/g;
167   my @eols = $tex =~ /&/g;
168   if (0+@leads == 0+@eols && 0+@eols > 1) {
169     $tex =~ s/&({$lead})/1&~/g;
170     $gathered = 0;
171     print "% The \"gathered\" is NOT used because all ' .
172       (0+@eols) . ' lines are ' . (0+@leads) . \" leads\\n\";
173   }
174 }
175 if ($macro ne 'phiq') {
176   sub strip_tabs {
177     my ($env, $tex) = @_;
178     $tex =~ s/&/&/g;
179     return "\\begin{$env}" . $tex . "\\end{$env}";
180   }
181   foreach my $e (('matrix', 'cases')) {
182     $tex =~ s/\\begin{\\(Q$e$E*?)\\}(\\.+)/\\end{\\(Q$e$E*?)\\}/strip_tabs($1, $2)/sge;
183   }
184 }
185 $tex =~ s/\\$/\\phiTerminal{\\xi}}/g;
186 $tex =~ s/(?<![\\^](?!\\{)/\\phiTerminal{\\rho}}/g;
187 $tex =~ s/[\\[\\]/\\phiTerminal{\\llbracket}/g;
188 $tex =~ s/[\\]/\\phiTerminal{\\rrbracket}/g;
189 $tex =~ s/[?]/\\phiTerminal{\\varnothing}/g;
190 $tex =~ s/@/\\phiTerminal{\\varphi}}/g;

```

```

191 $tex =~ s/-([a-z]+)>/\mathrel{\phiSlot{1}}/g;
192 $tex =~ s/->/\mathbin{\phiTerminal{\mapsto}}/g;
193 $tex =~ s/~>/\mathbin{\phiWave}/g;
194 $tex =~ s/:=/\mathrel{\vDash}/g;
195 $tex =~ s/(?<![\{\}]):/\mathrel{\phiTerminal{:}}/g;
196 $tex =~ s/==/\mathrel{\equiv}/g;
197 $tex =~ s/\.\.\>/\mathbin{\phiTerminal{\phiDotted}}/g;
198 $tex =~ s/~([0-9]+)/\phiTerminal{\alpha_{1}}/g;
199 $tex =~ s/<</\langle/g;
200 $tex =~ s/>>/\rangle/g;
201 $tex =~ s/(?<![\{\}])\([\.]\)/\phiTerminal{1}/g;
202 $tex =~ s/(?<![\{\}]),/\phiTerminal{,}\;/g;
203 $tex =~ s/|{2,}/|/g;
204 $tex =~ s/|([\^|]+)\|/'\textnormal{\texttt{'}} . ($1 =~ s{-(?=-)}{-}\gr) . '}}{'}/ge;
205 $tex =~ s/{TEXT(\d+)\}/'\text{' . $texts[$1] . '}'/ge;
206 if ($macro eq 'phiq') {
207   print '\(' if ($tex ne '');
208 } else {
209   print '\begin{' , $macro, "}\n";
210   if (not($align)) {
211     if ($gathered) {
212       print '\begin{gathered}' . "\n";
213     } elsif (not $splitting) {
214       print '\begin{split}' . "\n";
215     }
216   }
217 }
218 if ($gathered and not($align)) {
219   $tex =~ s/^&//g;
220   $tex =~ s/\n&/\n/g;
221 }
222 print $tex;
223 if ($macro eq 'phiq') {
224   print '\)' if ($tex ne '');
225 } else {
226   if (not($align)) {
227     if ($gathered) {
228       print "\n" . '\end{gathered}';
229     } elsif (not $splitting) {
230       print "\n" . '\end{split}';
231     }
232   }
233   print "\n" . '\end{' . $macro . '}' ;
234 }
235 print '\endinput';
236 \end{VerbatimOut}
237 \message{eolang: File with Perl script
238   '\eolang@tmpdir/\jobname-phi.pl' saved^^J}
239 \else
240   \message{eolang: Perl script already exists at
241     "\eolang@tmpdir/\jobname-phi.pl"^^J}
242 \fi
243 \closein 15
244 \fi

```

245 \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:

```
246 \makeatletter
247 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
248 \makeatother
```

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

```
249 \makeatletter
250 \newcommand\eolang@tmp[1]{#1\ifxetex-xe\else\ifluatex-lua\fi\fi.tex}
251 \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:

```
252 \makeatletter
253 \newcommand\eolang@ifabsent[2]{%
254   \IfFileExists
255     {#1}
256     {%
257       \message{eolang: File "#1" already exists ^^J}%
258       \input{#1}}
259   {%
260     \ifdefined\eolang@noshell%
261       \message{eolang: Shell processing is disabled ^^J}%
262     \else%
263       \ifnum\ShellEscapeStatus=1\else%
264         \message{eolang: The -shell-escape command line
265           option is not provided, most probably compilation
266           will fail now: ^^J}%
267       \fi%
268       #2%
269     \fi%
270   }%
271 }
272 \makeatother
```

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

```
273 \makeatletter\newcommand\eolang@process[1]{
274   \def\hash{\eolang@mdfive
275     {\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}}-\the\inputlineno}%
276   \eolang@ifabsent
277     {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation-post}}
278     {%
279       \iexec[null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}"
280         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation}"}%
281       \message{Start parsing 'phi' at line no. \the\inputlineno ^^J}
282       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation-post}]{
283         perl "\eolang@tmpdir/\jobname-phi.pl"
284         '#1'
285         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation}"
286         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
```

```

287         \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
288     }%
289     \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
290     \def\eolang@phiSaveTo{\relax}%
291 }
292 %
293 \newenvironment{phiuation*}%
294 {\catcode'\|=12 \VerbatimEnvironment%
295 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
296 \begin{VerbatimOut}
297   {\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}}}}
298 {\end{VerbatimOut}\eolang@process{equation*}}
299 %
300 \newenvironment{phiuation}%
301 {\catcode'\|=12 \VerbatimEnvironment%
302 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
303 \begin{VerbatimOut}
304   {\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}}}}
305 {\end{VerbatimOut}\eolang@process{equation}}
306 \makeatother

```

\phiq Then, we define \phiq command:

```

307 \RequirePackage{xstring}
308 \makeatletter\newcommand\phiq[1]{%
309   \StrSubstitute{\detokenize{#1}}{'}{'"}{\clean}%
310   \def\hash{\pdfmdfivesum{\clean}-\the\inputlineno}%
311   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
312   \eolang@ifabsent
313     {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq-post}}
314     {%
315       \iexec[log,trace,quiet,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{phiq}]{
316         printf '\%s' '\clean'}%
317       \iexec[quiet,null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{phiq}"
318         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq}"}%
319       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq-post}]{
320         perl \eolang@tmpdir/\jobname-phi.pl 'phiq'
321         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq}"
322         \ifdefined\eolang@nocomments | perl -pe 's/\%.*((\n|$))//g' \fi}%
323       \message{eolang: Parsed 'phiq' at line no. \the\inputlineno^^J}%
324     }%
325   \ifdefined\eolang@nodollar\else\catcode'\$=\active\fi%
326 }\makeatother

```

nodollar Then, we redefine dollar sign:

```

327 \ifdefined\eolang@nodollar\else
328   \begingroup
329   \catcode'\$=\active
330   \protected\gdef\phiq#1{\phiq{#1}}
331   \endgroup
332   \AtBeginDocument{\catcode'\$=\active}
333 \fi

```

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

```

334 \makeatletter
335 \ifdefined\eolang@noshell
336 \message{eolang: Perl script is not going to be created
337   at "\eolang@tmpdir/\jobname-sodg.pl", because of the
338   "noshell" package option~^J}
339 \else
340 \openin 15=\eolang@tmpdir/\jobname-sodg.pl
341 \ifeof 15
342 \message{eolang: Perl script is going to be created,
343   because it is absent at "\eolang@tmpdir/\jobname-sodg.pl",
344   but if -shell-escape is not set, the compilation will
345   most likely fail now~^J}
346 \begin{VerbatimOut}{\eolang@tmpdir/\jobname-sodg.pl}
347 sub num {
348   my ($i) = @_;
349   $i =~ s/(\+|-)\./\10./g;
350   return $i;
351 }
352 sub fmt {
353   my ($tex) = @_;
354   $tex =~ s/\\|([\^\\|]+)\\|/'\textnormal{\texttt{' . ($1 =~ s{-(?=-)}{-{}}gr) . '}}'/ge;
355   return $tex;
356 }
357 sub toem {
358   my ($cm) = @_;
359   return $cm * 2.8;
360 }
361 sub vertex {
362   my ($v) = @_;
363   if (index($v, 'v0') == 0) {
364     return '\Phi';
365   } else {
366     $v =~ s/^v/v_/g;
367     $v =~ s/^[0-9]$//g;
368     return $v . '>';
369   }
370 }
371 sub tailor {
372   my ($t, $m) = @_;
373   $t =~ s/<([A-Z]?${m}[A-Z]?):([>]+)>/\2/g;
374   $t =~ s/<[A-Z]+:[>]+>/\2/g;
375   return $t;
376 }
377 open(my $fh, '<', $ARGV[0]);
378 my $tex; { local $/; $tex = <$fh>; }
379 if (index($tex, "\t") >= 0) {
380   print "TABS are prohibited!";
381   exit 1;
382 }
383 print '% This file is auto-generated', "\n%\n";
384 print '% --- there are ', length($tex),
385   ' chars in the input (', $ARGV[0], "):\n";
386 foreach my $t (split /\n/g, $tex) {
387   print '% ', $t, "\n";

```

```

388 }
389 print "% ---\n";
390 $tex =~ s/\\\\\\/\n/g;
391 $tex =~ s/\\\n//g;
392 $tex =~ s/(\\[a-zA-Z]+\)\s+/\1/g;
393 $tex =~ s/\n{2,}/\n/g;
394 my @cmds = split(/\n/g, $tex);
395 print '% --- before processing:' . "\n";
396 foreach my $t (split(/\n/g, $tex)) {
397   print '% ', $t, "\n";
398 }
399 print '% ---';
400 print ' (' . (0+@cmds) . " lines)\n";
401 print '\begin{picture}', "\n";
402 for (my $c = 0; $c < 0+@cmds; $c++) {
403   my $cmd = $cmds[$c];
404   $cmd =~ s/^\s+//g;
405   $cmd =~ s/(?<!\[)\%.*//g;
406   my ($head, $tail) = split(/ /, $cmd, 2);
407   my %opts = ();
408   my ($body, $style) = split(/style:/, $tail, 2);
409   $opts{'style'} = $style;
410   $tail = $body;
411   foreach my $p (split(/ /, $tail)) {
412     my ($q, $t) = split(/:/, $p);
413     $opts{$q} = $t;
414   }
415   if (index($head, '\\') == 0) {
416     print $cmd;
417   } elsif (index($head, '->') >= 0) {
418     my $draw = '\draw[';
419     if (exists $opts{'pi'}) {
420       $draw = $draw . '<MB:phi-pi><F:draw=none>';
421       if (not exists $opts{'a'}) {
422         $opts{'a'} = '\pi';
423       }
424     }
425     if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
426       $draw = $draw . '<MB:,phi-rho>';
427     }
428     $draw = $draw . ',' . $opts{'style'} . ']';
429     my ($from, $to) = split (/->/, $head);
430     $draw = $draw . " ($from) ";
431     if (exists $opts{'bend'}) {
432       $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
433         num($opts{'bend'}) . '>';
434       if (exists $opts{'rho'}) {
435         $draw = $draw . '<MB:,phi-rho>';
436       }
437       $draw = $draw . ']';
438     } else {
439       $draw = $draw . '--';
440     }
441     if (exists $opts{'a'}) {

```

```

442     my $a = $opts{'a'};
443     if (index($a, '$') == -1) {
444         $a = '$' . fmt($a) . '$';
445     } else {
446         $a = fmt($a);
447     }
448     $draw = $draw . '<MB: node [phi-attr] {' . $a . '}>';
449 }
450 if (exists $opts{'break'}) {
451     $draw = $draw . '<F: coordinate [pos=' .
452         ($opts{'break'} / 100) . ']' (break)>';
453 }
454 $draw = $draw . " (<MF:${to}><B:break-v>";
455 if (exists $opts{'break'}) {
456     print tailor($draw, 'F') . ";\n";
457     print ' \node[outer sep=' . toem(0.1) . 'em,inner sep=0em] ' .
458         'at (break) (break-v) {' . vertex($to) .
459         '$};' . "\n";
460     print ' ' . tailor($draw, 'B');
461 } else {
462     print tailor($draw, 'M');
463 }
464 } elsif (index($head, '=') >= 0) {
465     my ($from, $to) = split (/=/, $head);
466     my $size = () = $head =~ /=/g;
467     if ($from eq '') {
468         print '\node [phi-arrow, left=' . toem($size * 0.6) . 'em of ' .
469             $to . '.center]';
470     } elsif ($to eq '') {
471         print '\node [phi-arrow, right=' . toem($size * 0.6) . 'em of ' .
472             $from . '.center]';
473     } else {
474         print '\node [phi-arrow] at ($(' .
475             $from . ')!0.5!(' . $to . ')$)';
476     }
477     print '{}';
478 } elsif (index($head, '!') >= 0) {
479     my ($v, $marker) = split (/!/, $head);
480     my $size = () = $head =~ /*!/g;
481     print '\node [phi-marker, left=' .
482         toem($size * 0.6) . 'em of ' .
483         $v . '.center][' . fmt($marker) . ']';
484 } elsif (index($head, '+') >= 0) {
485     my ($v, $suffix) = split (/\/, $head);
486     my @friends = ($v);
487     foreach my $c (@cmds) {
488         $e = $c;
489         $e =~ s/^\s+//g;
490         my $h = $e;
491         $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
492         foreach my $f (@friends) {
493             my $add = '';
494             if (index($h, $f . '->') >= 0) {
495                 $add = substr($h, index($h, '->') + 2);

```

```

496     }
497     if ($h =~ /->\Q${f}\E$/) {
498         $add = substr($h, 0, index($h, '->'));
499     }
500     if (index($e, ' xy:' . $f . ',') >= 0) {
501         $add = $h;
502     }
503     if (index($add, '+') == -1
504         and $add ne ''
505         and not(grep(/^\\Q${add}\\E$/, @friends))) {
506         push(@friends, $add);
507     }
508 }
509 }
510 my @extra = ();
511 foreach my $e (@cmds) {
512     $m = $e;
513     if ($m =~ /^\\s*\\Q${v}\\E\\s/) {
514         next;
515     }
516     if ($m =~ /^\\s*[\\^\\s]+\\+\\/ and not($m =~ /^\\s*\\Q${head}\\E\\s/)) {
517         next;
518     }
519     foreach my $f (@friends) {
520         my $h = $f;
521         $h =~ s/[a-z]$/g;
522         if ($m =~ s/^\\s*\\Q${f}\\E\\+\\Q${suffix}\\E\\s?\\/1${h}${suffix} /g) {
523             last;
524         }
525         $m =~ s/^\\s*\\Q${f}\\E\\s\\/1${h}${suffix} /g;
526         $m =~ s/^\\s*\\Q${f}\\E->\\/1${h}${suffix}->/g;
527         $m =~ s\\/sxy:\\Q${f}\\E,/ xy:${h}${suffix},/g;
528         $m =~ s/->\\Q${f}\\E\\s/->${h}${suffix} /g;
529     }
530     if ($m ne $e) {
531         push(@extra, ' ' . $m);
532     }
533 }
534 splice(@extra, 0, 0, $extra[-1]);
535 splice(@extra, -1, 1);
536 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
537     '), friends: [' . join(', ', @friends) . '] in ' .
538     (0+@cmds) . ' lines');
539 splice(@cmds, $c, 1, @extra);
540 print '% cloned ' . $v . ' at line no.' . $c .
541     ' (+ ' . (0+@extra) . ' lines -> ' .
542     (0+@cmds) . ' lines total)';
543 } elsif ($head =~ /^v[0-9]+[a-z]?$/) {
544     print '\\node[';
545     if (exists $opts{'xy'}) {
546         my ($v, $right, $down) = split(/,/ , $opts{'xy'});
547         my $loc = '';
548         if ($down > 0) {
549             $loc = 'below ';

```

```

550     } elseif ($down < 0) {
551         $loc = 'above ';
552     }
553     if ($right > 0) {
554         $loc = $loc . 'right';
555     } elseif ($right < 0) {
556         $loc = $loc . 'left';
557     }
558     print ', ' . $loc . '=';
559     print toem(abs(num($down))) . 'em and ' .
560         toem(abs(num($right))) . 'em of ' . $v . '.center';
561 }
562 if (exists $opts{'data'}) {
563     print ',phi-data';
564     if ($opts{'data'} ne '') {
565         my $d = $opts{'data'};
566         if (index($d, '|') == -1) {
567             $d = '$\Delta\phiDotted\text{' .
568                 '\textnormal{\texttt{' . fmt($d) . '}}}$';
569         } else {
570             $d = fmt($d);
571         }
572         $opts{'box'} = $d;
573     }
574 } elseif (exists $opts{'atom'}) {
575     print ',phi-atom';
576     if ($opts{'atom'} ne '') {
577         my $a = $opts{'atom'};
578         if (index($a, '$') == -1) {
579             $a = '$\lambda\phiDotted{' . fmt($a) . '$';
580         } else {
581             $a = fmt($a);
582         }
583         $opts{'box'} = $a;
584     }
585 } else {
586     print ',phi-object';
587 }
588 if (exists $opts{'edgeless'}) {
589     print ',draw=none';
590 }
591 print ', ' . $opts{'style'} . ']';
592 print ' (' . $head . ')';
593 print '{';
594 if (exists $opts{'tag'}) {
595     my $t = $opts{'tag'};
596     if (index($t, '$') == -1) {
597         $t = '$' . $t . '$';
598     } else {
599         $t = fmt($t);
600     }
601     print $t;
602 } else {
603     print '$' . vertex($head) . '$';

```

```

604     }
605     print '>';
606     if (exists $opts{'box'}) {
607         print ' node[phi-box] at (';
608         print $head, '.south east) {';
609         print $opts{'box'}, '>';
610     }
611 }
612 print ";\n";
613 }
614 print '\end{picture}%', "\n";
615 print "% --- after processing:\n%";
616 foreach my $c (@cmds) {
617     print '% ', $c, "\n";
618 }
619 print '% --- (' . (0+@cmds) . " lines)\n";
620 print '\endinput';
621 \end{VerbatimOut}
622 \message{eolang: File with Perl script
623   '\eolang@tmpdir/\jobname-sodg.pl' saved^^J}
624 \else
625   \message{eolang: Perl script already exists at
626     "\eolang@tmpdir/\jobname-sodg.pl"^^J}
627 \fi
628 \closein 15
629 \fi
630 \makeatother

```

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

```

631 \setcounter{FancyVerbLine}{0}

```

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

```

632 \RequirePackage{tikz}
633 \usetikzlibrary{arrows}
634 \usetikzlibrary{shapes}
635 \usetikzlibrary{decorations}
636 \usetikzlibrary{decorations.pathmorphing}
637 \usetikzlibrary{decorations.pathreplacing}
638 \usetikzlibrary{positioning}
639 \usetikzlibrary{calc}
640 \usetikzlibrary{math}
641 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment `picture`:

```

642 \newenvironment{picture}%
643   {\noindent\begin{tikzpicture}[
644     ->,>stealth',node distance=0,line width=.08em,
645     pics/parallel arrow/.style={
646       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}]}%
647   {\end{tikzpicture}}
648 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
649   minimum height=0.05em, minimum width=0.05em,
650   single arrow head extend=2mm]

```

```

651 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
652   minimum width=1.4em, font={\small\color{white}\ttfamily},
653   fill=gray]
654 \tikzstyle{phi-thing} = [inner sep=0pt,minimum height=2.4em,
655   draw,font={\small}]
656 \tikzstyle{phi-object} = [phi-thing,circle]
657 \tikzstyle{phi-data} = [phi-thing,regular polygon,
658   regular polygon sides=8]
659 \tikzstyle{phi-empty} = [phi-object]
660 \tikzset{%
661   phi-rho/.style={
662     postaction={%
663       decoration={
664         show path construction,
665         curveto code={
666           \tikzmath{
667             coordinate \I, \F, \v;
668             \I = (\tikzinputsegmentfirst);
669             \F = (\tikzinputsegmentlast);
670             \v = ($(\I) -(\F)$);
671             real \d, \a, \r, \t;
672             \d = 0.8;
673             \t = atan2(\vy, \vx);
674             if \vx<0 then { \a = 90; } else { \a = -90; };
675             {
676               \draw[arrows={-latex}, decorate,
677                 decoration={%
678                   snake, amplitude=.4mm,
679                   segment length=2mm,
680                   post length=1mm
681                 }]]
682               ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
683               -- ++(\t: 2*\d em);
684             };
685           }
686         },
687         lineto code={
688           \tikzmath{
689             coordinate \I, \F, \v;
690             \I = (\tikzinputsegmentfirst);
691             \F = (\tikzinputsegmentlast);
692             \v = ($(\I) -(\F)$);
693             real \d, \a, \r, \t;
694             \d = 0.8;
695             \t = atan2(\vy, \vx);
696             if \vx<0 then { \a = 90; } else { \a = -90; };
697             {
698               \draw[arrows={-latex}, decorate,
699                 decoration={%
700                   snake, amplitude=.4mm,
701                   segment length=2mm,
702                   post length=1mm}]
703               ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
704               -- ++(\t: 2*\d em);

```

```

705         };
706     }
707 }
708 },
709     decorate
710 }
711 }
712 }
713 \tikzstyle{phi-pi} = [draw,dotted]
714 \tikzstyle{phi-atom} = [phi-object,double]
715 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
716     rectangle,line width=.04em,minimum width=1.2em,anchor=north west,
717     font={\scriptsize}]
718 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
719     above=2pt,sloped/.append style={transform shape},
720     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:

```

721 \makeatletter
722 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
723 \makeatother

```

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

```

724 \makeatletter\newenvironment{sodg}%
725 {\catcode'\|=12 \VerbatimEnvironment%
726 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
727 \begin{VerbatimOut}
728   {\eolang@tmpdir/\jobname/\eolang@tmp{sodg}}}
729 {\end{VerbatimOut}}%
730 \def\hash{\eolang@mdfive
731   {\eolang@tmpdir/\jobname/\eolang@tmp{sodg}}-\the\inputlineno}%
732 \catcode'\$=3 %
733 \eolang@ifabsent
734   {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg-post}}
735   {%
736     \iexec[null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{sodg}"
737       "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg}}"%
738     \message{eolang: Start parsing 'sodg' at line no. \the\inputlineno^^J}
739     \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg-post}]{
740       perl "\eolang@tmpdir/\jobname-sodg.pl"
741       "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg}"
742       \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\\n|$)//g'\fi
743       \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%
744   }
745 \catcode'\$=active%
746 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
747 \def\eolang@sodgSaveTo{\relax}%
748 }\makeatother

```

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

```

749 \RequirePackage{hyperref}
750 \pdfstringdefDisableCommands{
751   \def\({}%

```

```

752 \def\}{}%
753 \def\alpha{alpha}%
754 \def\varphi{phi}%
755 }
756 \makeatletter
757 \NewExpandableDocumentCommand{\eoAnon}{0{ANONYMIZED}m}{%
758   \ifdefined\eolang@anonymous%
759     \eolang@ink{#1}%
760   \else%
761     #2%
762   \fi%
763 }\makeatother

```

`\eolang` Then, we define a simple supplementary command for printing EO, the name of the language.

```

764 \newcommand\eolang{%
765   \eoAnon[XYZ]{\sffamily EO}}

```

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

```

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

```

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

```

768 \newcommand\xmirl{%
769   \eoAnon[XML\(^+\)]{XMIR}}

```

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

```

770 \newcommand\phiWave{%
771   \phiTerminal{\mapstochar\mathrel{\mspace{0.45mu}}\leadsto}}

```

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

```

772 \newcommand\phiSlot[1]{%
773   \xrightarrow{\text{\sffamily\scshape #1}}}

```

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

```

774 \makeatletter
775 \newcommand{\phi0set}[2]{%
776   \mathrel{\mathop{#2}\limits^{
777     \vbox to 0ex{\kern-2\ex@
778       \hbox{$\scriptscriptstyle#1$\vss}}}}}
779 \newcommand{\phiUset}[2]{%
780   \mathrel{\mathop{#2}\limits_{
781     \vbox to 0ex{\kern-6.3\ex@
782       \hbox{$\scriptscriptstyle#1$\vss}}}}}
783 \makeatother

```

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

```

784 \newcommand\phiMany[3]{%
785   \phi0set{#3}{\phiUset{#2}{#1}}}

```

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

```

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

```

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

```

787 \makeatletter
788 \newcommand\phiTerminal[1]{\eolang@ink{#1}}
789 \makeatother

```

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

```

790 \RequirePackage{trimclip}
791 \RequirePackage{amsfonts}
792 \makeatletter
793 \newcommand{\phiDotted}{%
794   \phiTerminal{\mapstochar\mathrel{\mathpalette\phiDotted@relax}}}
795 \newcommand{\phiDotted@}[2]{%
796   \begingroup%
797   \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
798   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
799   \sbox\z@{%
800     \makebox[\dimen\z@][s]{%
801       \clipbox{0 0 {0.4\width} 0}%
802       {\resizebox{\dimen\z@}{\height}%
803         {\m@th#1\dashrightarrow}}}%
804     \hss%
805     \clipbox{{0.69\width} {-0.1\height} 0
806       {-\height}}{\m@th#1\rightarrow}%
807   }%
808   }%
809   \ht\z@=\dimen\tw@ \dp\z@=\z@%
810   \box\z@%
811   \endgroup%
812 }
813 \makeatother
814 \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		phiquation , another one for sodg	17
	General: First draft.		11
0.0.2		0.12.1	
	sodg : The environment phigure renamed to sodg for the sake of better semantics. The graph in the picture is solely a SODG graph, that's why the name sodg is better.		17
	-phi.pl : New symbol added for basket slots		12
	Parsing of the symbols " @ ," " ^ ," and " & " enabled (\varphi , \rho , and \sigma)		12
	The symbols " [" and "] " replaced with " [[" and "]] " for abstract object brackets, because they conflicted with normal square brackets		12
	\phiq : Parsing of additional symbols enabled.		17
	-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		0.13.0	
	General: Parsing of package options introduced.		11
	\eolang : New command \eolang added to print the name of the language in both normal and the anonymous mode of acmart		26
	\eolang@mdfive : New supplementary command added to calculate MD5 sum of a file.		12
	-phi.pl : A new Perl script " eolang-phi.pl " added for parsing of phi expressions.		12
	\phic : New command \phic prints the name of φ -calculus in both normal and the anonymous mode of acmart		26
	\phiDotted : New command \phiDotted added to denote a link to a special attribute.		27
	-sodg.pl : There are two Perl scripts now: one for		
		phiquation , another one for sodg	17
		-sodg.pl : The bug is fixed related to the formatting of indexes of vertices.	17
		-phi.pl : Parsing of QQ into \dot{\Phi} implemented.	12
		-sodg.pl : The edgeless tag of a vertex removes the border of it.	17
		-sodg.pl : The style tag of vertices and edges.	17
		0.16.0	
		phiquation : The processing of phiquation data is done only if it's the first time processing, otherwise cache is used, thus making processing faster.	16
		sodg : The processing of sodg data is done only if it's the first time processing, otherwise cache is used, thus making processing faster.	25
		0.17.0	
		\eolang@ifabsent : A new supplementary eolang@ifabsent command added	16
		0.18.0	
		\eolang@ifabsent : The noshell package option added in order to enable complete prohibition of shell interactions.	16
		0.18.3	
		\eolang@tmp : A new supplementary eolang@tmp command added, to generate temporary file names.	16
		0.19.0	
		-phi.pl : Parsing of T into \bot implemented.	12
		0.2.0	
		-phi.pl : Numbers automatically render as \texttt . No need to use vertical bars around them anymore.	12
		-sodg.pl : The content of the atom and the data boxes is parsed	

automatically as formulas and numbers, respectively.	17	package option.	27
<code>\xmir</code> : New command <code>\xmir</code> prints XMIR in both normal and the anonymous mode of <code>acmart</code> . . .	26	0.25.0	
0.20.0		<code>-phi.pl</code> : Parsing of <code>:</code> into an orange colon (<code>\phiTerminal{:}</code>) implemented.	12
<code>-phi.pl</code> : Parsing of <code>0</code> into <code>\alpha_0</code> implemented.	12	0.3.0	
0.20.2		<code>\eolang@lineno</code> : New counter for protecting <code>lineno</code>	12
<code>-phi.pl</code> : Formatting of numbers fixed, now it works in any place inside an expression.	12	<code>-phi.pl</code> : New arrow added, that looks like <code>\leadsto</code>	12
0.21.0		<code>\phiWave</code> : New command <code>\phiWave</code> added to denote a link to a multi-layer attribute.	26
<code>-phi.pl</code> : Automated formatting of <code>TRUE</code> and <code>FALSE</code> removed. . . .	12	0.4.0	
<code>\phiTerminal</code> : New command <code>\phiTerminal</code> added, to highlight all terminals in phi expressions.	27	<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
0.22.0		Relative positioning of vertices fixed.	17
<code>-phi.pl</code> : Automated formatting of <code>D</code> and <code>L</code> added.	12	0.5.0	
0.23.0		<code>-phi.pl</code> : Automated formatting of <code>TRUE</code> and <code>FALSE</code> added.	12
<code>-phi.pl</code> : Parsing of <code>QQ</code> into <code>\dot{\Phi}</code> removed.	12	<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow. . .	26
0.23.1		<code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket.	26
<code>nodollar</code> : The dollar sign is made active at the end of the package (<code>\AtEndOfPackage</code>), so that <code>\$. . . \$</code> captured inside preamble-defined macros is also converted to <code>\phiq</code>	17	<code>-sodg.pl</code> : It is possible to use TikZ commands inside the <code>sodg</code> environment.	17
0.23.2		New syntax introduced that allows to make clones of vertices and all their dependants.	17
<code>nodollar</code> : The <code>\AtEndOfPackage</code> activation is removed, because it left <code>\$</code> active for the rest of the preamble and broke packages loaded after <code>eolang</code> . The dollar sign is now activated only at the beginning of the document; for <code>\$. . . \$</code> inside preamble-defined macros, use <code>\phiq{. . .}</code> instead, which is catcode-independent.	17	Now edges may have the <code>break</code> attribute, to make them shorter. .	17
0.24.0		0.6.0	
General: The <code>nocolor</code> package option added, in order to suppress the orange colour. . .	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
<code>\eolang@ink</code> : New internal command <code>\eolang@ink</code> paints its argument, honouring the <code>nocolor</code> package option.	11	<code>-sodg.pl</code> : The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations.	17
<code>\phiTerminal</code> : The <code>\phiTerminal</code> command respects the <code>nocolor</code>		0.7.0	
		<code>nodollar</code> : Now it is possible to use dollar sign instead of the <code>\phiq</code> command.	17
		<code>-phi.pl</code> : New syntax sugar for Φ , just using capital “Q” is enough. .	12

Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols.	12	<code>\phiSaveTo</code> : The output of the <code>\phiquation</code> environment can be redirected to a file.	16
Text in quotes is automatically converted to <code>\texttt</code>	12	<code>-sodg.pl</code> : The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.	17
0.8.0		<code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file.	25
General: The <code>anonymous</code> package option added.	11	0.9.0	
<code>-phi.pl</code> : Inside <code>\phiquation</code> any text inside the <code>\text</code> macro is not processed.	12	<code>\eoAnon</code> : New command <code>\eoAnon</code> added.	25
<code>\phi0set</code> : New commands <code>\phi0set</code> and <code>\phiUset</code> help position text over and under an arrow.	26	<code>-phi.pl</code> : Proper handling of the <code>matrix</code> environment.	12
		<code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code> . . .	27

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	
$\backslash$$ 185, 311, 325, 329, 332, 732, 745	$\backslash$ catcode .. 294, 301, 311, 325, 329, 332, 725, 732, 745	$\backslash$ eolang@phiSaveTo 247, 287, 290
$\backslash%$... 286, 316, 322, 742	$\backslash$ clean 309, 310, 316	$\backslash$ eolang@process 273, 298, 305
$\backslash($... 207, 751, 767, 769	$\backslash$ clipbox 801, 805	$\backslash$ eolang@sodgSaveTo 722, 743, 747
$\backslash)$... 224, 752, 767, 769	$\backslash$ closein 243, 628	$\backslash$ eolang@tmp 249, 275, 277, 279, 280, 282, 285, 297, 304, 313, 315, 317, 318, 319, 321, 728, 731, 734, 736, 737, 739, 741
$\backslash*$ 151, 182	$\backslash$ color 26, 652	$\backslash$ eolang@tmpdir 11, 32, 33, 37, 42, 44, 60, 63, 66, 69, 238, 241, 275, 277, 279, 280, 282, 283, 285, 297, 304, 313, 315, 317, 318, 319, 320, 321, 337, 340, 343, 346, 623, 626, 728, 731, 734, 736, 737, 739, 740, 741
$\backslash+$... 349, 485, 516, 522		$\backslash$ ex@ 777, 781
$\backslash-$ 767		
$\backslash$ -phi.pl <u>57</u>	D	F
$\backslash$ -sodg.pl <u>334</u>	$\backslash$ d 205, 671, 672, 682, 683, 693, 694, 703, 704	$\backslash$ F 667, 669, 670, 682, 689, 691, 692, 703
$\backslash.$... 142, 152, 197, 349	$\backslash$ dashrightarrow ... 803	$\backslash$ FancyVerbLine <u>631</u>
$\backslash/$ 150, 151	$\backslash$ def 247, 274, 290, 310, 722, 730, 747, 751, 752, 753, 754	
$\backslash?$ 189	$\backslash$ Delta 567	
$\backslash[$ 150, 187	$\backslash$ detokenize 309	
$\backslash{$ 88, 117, 140, 141, 142, 143, 144, 145, 146, 147, 148, 155, 156, 182, 186, 195, 201, 202, 205	$\backslash$ dimen 797, 798, 800, 802, 809	
$\backslash}$ 88, 117, 155, 156, 182, 205	$\backslash$ dp 809	
$\backslash]$ 150, 188	$\backslash$ draw . 418, 646, 676, 698	
$\backslash^$ 186	E	G
$\backslash $ 151, 203, 204, 294, 301, 354, 725	$\backslash$ E 182, 497, 505, 513, 516, 522, 525, 526, 527, 528	$\backslash$ gdef 330
	$\backslash$ end ... 228, 230, 233, 236, 298, 305, 614, 621, 647, 729	
Numbers	$\backslash$ endinput . 235, 620, 814	
$\backslash 2$ 152, 161, 373	$\backslash$ eoAnon <u>749</u> , 765, 767, 769	
$\backslash 3$ 152	$\backslash$ eolang <u>764</u>	
$\backslash 4$ 152	$\backslash$ eolang@anonymous 16, 758	
	$\backslash$ eolang@ifabsent .. . 252, 276, 312, 733	
A	$\backslash$ eolang@ink <u>21</u> , 759, 788	
$\backslash$ a 671, 674, 682, 693, 696, 703	$\backslash$ eolang@lineno <u>52</u>	
$\backslash$ active 325, 329, 332, 745	$\backslash$ eolang@mdfive <u>53</u> , 274, 730	H
$\backslash$ alpha 753, 767	$\backslash$ eolang@nocolor . 15, 23	$\backslash$ hash 274, 277, 280, 282, 285, 310, 313, 318, 319, 321, 730, 734, 737, 739, 741
$\backslash$ AtBeginDocument .. 332	$\backslash$ eolang@nocomments . . 13, 286, 322, 742	$\backslash$ hbox 778, 782
B	$\backslash$ eolang@nodollar 14, 311, 325, 327	$\backslash$ height ... 802, 805, 806
$\backslash$ Bbbk 3	$\backslash$ eolang@noshell 5, 17, 30, 36, 58, 260, 335	$\backslash$ hss 804
$\backslash$ begin 69, 90, 209, 212, 214, 296, 303, 346, 401, 643, 727		$\backslash$ ht 809
$\backslash$ box 810		

I		630, 723, 748,	<code>\phiUset</code> 779, 785
<code>\I</code>	667, 668, 670, 682, 689, 690, 692, 703	763, 783, 789, 813	<code>\phiWave</code> 770
<code>\iexec</code>	42, 279, 282, 315, 317, 319, 736, 739	<code>\makebox</code> 800	<code>\pi</code> 422
<code>\ifdefined</code>	5, 23, 30, 36, 58, 260, 286, 287, 311, 322, 325, 327, 335, 742, 743, 758	<code>\mapstochar</code> . . . 771, 794	<code>\ProcessPgfpPackageOptions</code> 20
<code>\ifeof</code>	64, 341	<code>\mathop</code> 776, 780	<code>\protected</code> 330
<code>\IfFileExists</code> . .	31, 254	<code>\mathpalette</code> 794	
<code>\ifluatex</code>	12, 250	<code>\mathrel</code> 771, 776, 780, 794	
<code>\ifnum</code>	41, 263	<code>\message</code> . 33, 37, 44, 59, 65, 237, 240, 257, 261, 264, 281, 323, 336, 342, 622, 625, 738	Q
<code>\ifxetex</code>	12, 250		<code>\Q</code> 182, 497, 505, 513, 516, 522, 525, 526, 527, 528
<code>\input</code>	258	<code>\mspace</code> 771	
<code>\inputlineno</code>	275, 281, 310, 323, 731, 738		R
J		N	<code>\relax</code> . . 3, 290, 747, 794
<code>\jobname</code>	32, 33, 37, 42, 44, 60, 63, 66, 69, 238, 241, 275, 277, 279, 280, 282, 283, 285, 297, 304, 313, 315, 317, 318, 319, 320, 321, 337, 340, 343, 346, 623, 626, 728, 731, 734, 736, 737, 739, 740, 741	<code>\newcommand</code> 24, 26, 55, 247, 250, 253, 273, 308, 722, 764, 766, 768, 770, 772, 775, 779, 784, 786, 788, 793, 795	<code>\RequirePackage</code> 1, 2, 3, 4, 5, 6, 7, 8, 21, 30, 53, 307, 632, 749, 790, 791
K			<code>\resizebox</code> 802
<code>\kern</code>	777, 781	<code>\newcounter</code> 52	<code>\rightarrow</code> 797, 798, 806
L		<code>\newenvironment</code> 293, 300, 642, 724	S
<code>\lambda</code>	579	<code>\NewExpandableDocumentCommand</code> 757	<code>\sbox</code> 799
<code>\leadsto</code>	771	<code>\node</code> 457, 468, 471, 474, 481, 544	<code>\scriptscriptstyle</code> 778, 782
<code>\limits</code>	776, 780	<code>\nodollar</code> 327	<code>\scriptsize</code> . . . 717, 720
M		<code>\noindent</code> 643	<code>\scshape</code> 773
<code>\m@th</code> . 797, 798, 803, 806		O	<code>\setcounter</code> 289, 295, 302, 631, 726, 746
<code>\makeatletter</code>	 22, 29, 52, 54, 57, 246, 249, 252, 273, 308, 334, 721, 724, 756, 774, 787, 792	<code>\openin</code> 63, 340	<code>\settoheight</code> 798
<code>\makeatother</code>	 28, 51, 52, 56, 245, 248, 251, 272, 306, 326,	P	<code>\settowidth</code> 797
		<code>\pdf@filemdfivesum</code> . 55	<code>\sffamily</code> 765, 773
		<code>\pdf@mdfivesum</code> . . . 310	<code>\ShellEscapeStatus</code> 41, 263
		<code>\pdfstringdefDisableCommands</code> 750	<code>\small</code> 652, 655
		<code>\pgfkeys</code> 9	<code>\sodg</code> 724
		<code>\Phi</code> 364	<code>\sodgSaveTo</code> 721
		<code>\phic</code> 766	<code>\StrSubstitute</code> . . . 309
		<code>\picture</code> 642	<code>\sxy</code> 527
		<code>\phiDotted</code> . 567, 579, 790	T
		<code>\phiDotted@</code> . . . 794, 795	<code>\t</code> 77, 379, 671, 673, 682, 683, 693, 695, 703, 704
		<code>\phiEOL</code> 786	<code>\text</code> 567, 773
		<code>\phiMany</code> 784	<code>\textnormal</code> 568
		<code>\phiOset</code> 774, 785	<code>\texttt</code> 568
		<code>\phii</code> 307, 330	<code>\the</code> 275, 281, 310, 323, 731, 738
		<code>\phiquation</code> 273	<code>\tikz</code> 632
		<code>\phiSaveTo</code> 246	<code>\tikzinputsegmentfirst</code> 668, 690
		<code>\phiSlot</code> 772	<code>\tikzinputsegmentlast</code> 669, 691
		<code>\phiTerminal</code> 771, 787, 794	<code>\tikzmath</code> 666, 688

