



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

Yegor Bugayenko
yegor256@gmail.com

2026-08-11, 0.24.0

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`),
- “|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.

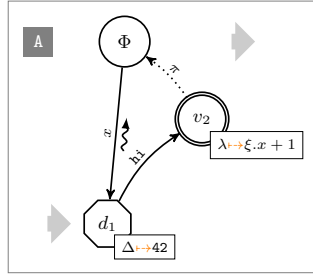
Texts within double quotes are likewise rendered in fixed-width font.

`\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:|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 $\langle \alpha_i \stackrel{n}{\rightarrow} x_i \rangle$ 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}} \text{\sffamily R2}. 15 \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 \mapsto 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:

```

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

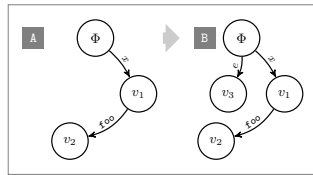
```

```

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:

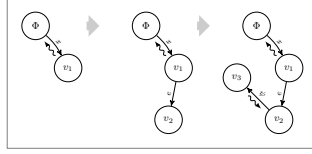


```

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:

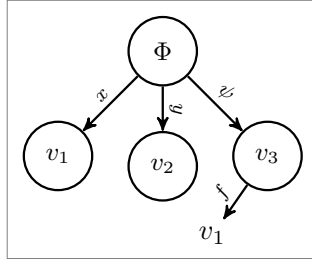


```

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:

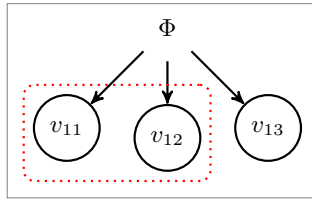


```

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:

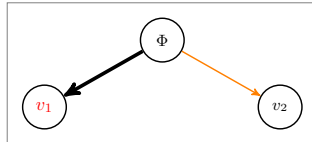


```

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

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

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

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

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

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

```
272 \makeatletter\newcommand\eolang@process[1]{
273   \def\hash{\eolang@mdfive
274     {\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}}-\the\inputlineno}%
275   \eolang@ifabsent
276     {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation-post}}
277     {%
278       \iexec[null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}"
279         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation}"}%
280       \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
281       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation-post}]{
282         perl "\eolang@tmpdir/\jobname-phi.pl"
283         '#1'
284         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiuation}"
285         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\\n|$\n)//g'\fi
286         \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
287     }%
```

```

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

```

`\phiq` Then, we define `\phiq` command:

```

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

```

`nodollar` Then, we redefine dollar sign:

```

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

```

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

[fancyvrb](#):

```

333 \makeatletter
334 \ifdefined\eolang@noshell

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

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

```

630 \setcounter{FancyVerbLine}{0}

```

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

```

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

```

picture Then, we define internal environment `picture`:

```

641 \newenvironment{picture}%
642   {\noindent\begin{tikzpicture}[
643     ->,>=stealth',node distance=0,line width=.08em,
644     pics/parallel arrow/.style={
645       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
646   {\end{tikzpicture}}
647 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
648   minimum height=0.05em, minimum width=0.05em,
649   single arrow head extend=2mm]
650 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
651   minimum width=1.4em, font={\small\color{white}\ttfamily},

```

```

652 fill=gray]
653 \tikzstyle{phi-thing} = [inner sep=0pt,minimum height=2.4em,
654 draw,font={\small}]
655 \tikzstyle{phi-object} = [phi-thing,circle]
656 \tikzstyle{phi-data} = [phi-thing,regular polygon,
657 regular polygon sides=8]
658 \tikzstyle{phi-empty} = [phi-object]
659 \tikzset{%
660 phi-rho/.style={
661 postaction={%
662 decoration={
663 show path construction,
664 curveto code={
665 \tikzmath{
666 coordinate \I, \F, \v;
667 \I = (\tikzinputsegmentfirst);
668 \F = (\tikzinputsegmentlast);
669 \v = ($(\I) -(\F)$);
670 real \d, \a, \r, \t;
671 \d = 0.8;
672 \t = atan2(\vy, \vx);
673 if \vx<0 then { \a = 90; } else { \a = -90; };
674 {
675 \draw[arrows={-latex}, decorate,
676 decoration={%
677 snake, amplitude=.4mm,
678 segment length=2mm,
679 post length=1mm
680 }]}
681 ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
682 -- ++(\t: 2*\d em);
683 };
684 }
685 },
686 lineto code={
687 \tikzmath{
688 coordinate \I, \F, \v;
689 \I = (\tikzinputsegmentfirst);
690 \F = (\tikzinputsegmentlast);
691 \v = ($(\I) -(\F)$);
692 real \d, \a, \r, \t;
693 \d = 0.8;
694 \t = atan2(\vy, \vx);
695 if \vx<0 then { \a = 90; } else { \a = -90; };
696 {
697 \draw[arrows={-latex}, decorate,
698 decoration={%
699 snake, amplitude=.4mm,
700 segment length=2mm,
701 post length=1mm}]
702 ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
703 -- ++(\t: 2*\d em);
704 };
705 }

```

```

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

```

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

```

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

```

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

```

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

```

748 \RequirePackage{hyperref}
749 \pdfstringdefDisableCommands{
750   \def\({}%
751   \def\)}{%
752   \def\alpha{\alpha}%

```

```

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

```

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

```

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

```

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

```

765 \newcommand\phic{%
766 \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.

```

767 \newcommand\xmirl{%
768 \eoAnon[XML{^+}]{XMIR}}

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

789 \RequirePackage{trimclip}
790 \RequirePackage{amsfonts}
791 \makeatletter
792 \newcommand{\phiDotted}{%
793   \phiTerminal{\mapstochar\mathrel{\mathpalette\phiDotted@relax}}}
794 \newcommand{\phiDotted@}[2]{%
795   \begingroup%
796   \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
797   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
798   \sbox\z@{%
799     \makebox[\dimen\z@][s]{%
800       \clipbox{0 0 {0.4\width} 0}%
801       {\resizebox{\dimen\z@}{\height}%
802         {\m@th#1\dashrightarrow}}}%
803     \hss%
804     \clipbox{{0.69\width} {-0.1\height} 0
805       {-\height}}{\m@th#1\rightarrow}%
806   }%
807 }%
808 \ht\z@=\dimen\tw@ \dp\z@=\z@%
809 \box\z@%
810 \endgroup%
811 }
812 \makeatother
813 \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	0.13.0	
	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	0.14.0	
	\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.	0.15.0	
			17
0.1.0		0.16.0	
	General: Parsing of package options introduced.		16
	\eolang : New command \eolang added to print the name of the language in both normal and the anonymous mode of acmart	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
	\eolang@mdfive : New supplementary command added to calculate MD5 sum of a file.	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
	-phi.pl : A new Perl script " eolang-phi.pl " added for parsing of phi expressions.	0.17.0	
	\phic : New command \phic prints the name of φ -calculus in both normal and the anonymous mode of acmart		16
	\phiDotted : New command \phiDotted added to denote a link to a special attribute.	0.18.0	
	-sodg.pl : There are two Perl scripts now: one for		16
		\eolang@ifabsent : A new supplementary eolang@ifabsent command added	16
		0.18.3	
			16
		\eolang@tmp : A new supplementary eolang@tmp command added, to generate temporary file names.	16
		0.19.0	
			12
		0.2.0	
			12
		-sodg.pl : The content of the atom and the data boxes is parsed	

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

Text in quotes is automatically converted to <code>\texttt</code>	12	be redirected to a file.	16
0.8.0		<code>-sodg.pl</code> : The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.	17
General: The <code>anonymous</code> package option added.	11	<code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file.	25
<code>-phi.pl</code> : Inside <code>phiquation</code> any text inside the <code>\text</code> macro is not processed.	12	0.9.0	
<code>\phi0set</code> : New commands <code>\phi0set</code> and <code>\phiUset</code> help position text over and under an arrow.	26	<code>\eoAnon</code> : New command <code>\eoAnon</code> added.	25
<code>\phiSaveTo</code> : The output of the <code>phiquation</code> environment can		<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		\eolang@phiSaveTo . .	
\\$	185, 310, 324, 328, 331, 731, 744	\catcode . .	293, 300, 310, 324, 328, 331, 724, 731, 744		 246, 286, 289
\%	285, 315, 321, 741	\clean	308, 309, 315	\eolang@process	272, 297, 304
\(.	206, 750, 766, 768	\clipbox	800, 804	\eolang@sodgSaveTo	721, 742, 746
\)	223, 751, 766, 768	\closein	242, 627	\eolang@tmp	. 248, 274, 276, 278, 279, 281, 284, 296, 303, 312, 314, 316, 317, 318, 320, 727, 730, 733, 735, 736, 738, 740
*	151, 182	\color	26, 651	\eolang@tmpdir 11, 32, 33, 37, 42, 44, 60, 63, 66, 69, 237, 240, 274, 276, 278, 279, 281, 282, 284, 296, 303, 312, 314, 316, 317, 318, 319, 320, 336, 339, 342, 345, 622, 625, 727, 730, 733, 735, 736, 738, 739, 740	
\+	348, 484, 515, 521			\ex@	776, 780
\-	766	D			
\-phi.pl	<u>57</u>	\d	204, 670, 671, 681, 682, 692, 693, 702, 703		
\-sodg.pl	<u>333</u>	\dashrightarrow . . .	802		
\.	142, 152, 196, 348	\def 246, 273, 289, 309, 721, 729, 746, 750, 751, 752, 753			
\/.	150, 151	\Delta	566		
\?	189	\detokenize	308		
\[.	150, 187	\dimen	796, 797, 799, 801, 808		
\{	88, 117, 140, 141, 142, 143, 144, 145, 146, 147, 148, 155, 156, 182, 186, 200, 201, 204	\dp	808		
\}	88, 117, 155, 156, 182, 204	\draw . 417, 645, 675, 697			
\].	150, 188	E			
\^	186	\E	182, 496, 504, 512, 515, 521, 524, 525, 526, 527		
\ 	151, 202, 203, 293, 300, 353, 724	\end . . . 227, 229, 232, 235, 297, 304, 613, 620, 646, 728			
Numbers		\endinput . 234, 619, 813			
\2	152, 161, 372	\eoAnon <u>748</u> , 764, 766, 768			
\3	152	\eolang	<u>763</u>		
\4	152	\eolang@anonymous 16, <u>757</u>			
A		\eolang@ifabsent	251, 275, 311, 732		
\a	670, 673, 681, 692, 695, 702	\eolang@ink <u>21</u> , 758, 787			
\active 324, 328, 331, 744		\eolang@lineno	<u>52</u>		
\alpha	752, 766	\eolang@mdfive	<u>53</u> , 273, 729		
\AtBeginDocument . .	331	\eolang@nocolor . 15, 23			
B		\eolang@nocomments	13, 285, 321, 741		
\Bbbk	3	\eolang@nodollar	14, 310, 324, 326		
\begin 69, 90, 208, 211, 213, 295, 302, 345, 400, 642, 726		\eolang@noshell 5, 17, 30, 36, 58, 259, 334			
\box	809				
		F			
		\F 666, 668, 669, 681, 688, 690, 691, 702			
		\FancyVerbLine	<u>630</u>		
		G			
		\gdef	329		
		H			
		\hash	273, 276, 279, 281, 284, 309, 312, 317, 318, 320, 729, 733, 736, 738, 740		
		\hbox	777, 781		
		\height . . . 801, 804, 805			
		\hss	803		
		\ht	808		

I		629, 722, 747,	\phiUset 778, 784
\I 666, 667, 669, 681,		762, 782, 788, 812	\phiWave 769
688, 689, 691, 702	\makebox 799		\pi 421
\iexec 42,	\mapstochar . . . 770, 793		\ProcessPgfpPackageOptions
278, 281, 314,	\mathop 775, 779		 20
316, 318, 735, 738	\mathpalette 793		\protected 329
\ifdefined 5, 23,	\mathrel 770, 775, 779, 793	Q	
30, 36, 58, 259,	\message . 33, 37, 44,	\Q 182, 496, 504,	
285, 286, 310,	59, 65, 236, 239,	512, 515, 521,	
321, 324, 326,	256, 260, 263,	524, 525, 526, 527	
334, 741, 742, 757	280, 322, 335,		
\ifeof 64, 340	341, 621, 624, 737	R	
\IfFileExists . . 31, 253	\mspace 770	\relax . . 3, 289, 746, 793	
\ifluatex 12, 249	N	\RequirePackage 1, 2,	
\ifnum 41, 262	\newcommand 24,	3, 4, 5, 6, 7, 8,	
\ifxetex 12, 249	26, 55, 246, 249,	21, 30, 53, 306,	
\input 257	252, 272, 307,	631, 748, 789, 790	
\inputlineno 274, 280,	721, 763, 765,	\resizebox 801	
309, 322, 730, 737	767, 769, 771,	\rightarrow 796, 797, 805	
	774, 778, 783,		
J	785, 787, 792, 794	S	
\jobname 32,	\newcounter 52	\sbox 798	
33, 37, 42, 44, 60,	\newenvironment . . .	\scriptscriptstyle .	
63, 66, 69, 237,	. 292, 299, 641, 723	 777, 781	
240, 274, 276,	\NewExpandableDocumentCommand	\scriptsize . . . 716, 719	
278, 279, 281,	 756	\scshape 772	
282, 284, 296,	\node 456, 467,	\setcounter 288, 294,	
303, 312, 314,	470, 473, 480, 543	301, 630, 725, 745	
316, 317, 318,	\nodollar 326	\settoheight 797	
319, 320, 336,	\noindent 642	\settowidth 796	
339, 342, 345,		\sffamily 764, 772	
622, 625, 727,	O	\ShellEscapeStatus .	
730, 733, 735,	\openin 63, 339	 41, 262	
736, 738, 739, 740		\small 651, 654	
	P	\sodg 723	
K	\pdf@filemdfivesum . 55	\sodgSaveTo 720	
\kern 776, 780	\pdf@mdfivesum . . . 309	\StrSubstitute 308	
	\pdfstringdefDisableCommands	\sxy 526	
L	 749	T	
\lambda 578	\pgfkeys 9	\t 77, 378, 670,	
\leadsto 770	\Phi 363	672, 681, 682,	
\limits 775, 779	\phic 765	692, 694, 702, 703	
	\picture 641	\text 566, 772	
M	\phiDotted . 566, 578, 789	\textnormal 567	
\m@th . 796, 797, 802, 805	\phiDotted@ . . . 793, 794	\texttt 567	
\makeatletter	\phiEOL 785	\the 274, 280,	
. . . . 22, 29, 52,	\phiMany 783	309, 322, 730, 737	
54, 57, 245, 248,	\phiOset 773, 784	\tikz 631	
251, 272, 307,	\phiq 306, 329	\tikzinputsegmentfirst	
333, 720, 723,	\phiquation 272	 667, 689	
755, 773, 786, 791	\phiSaveTo 245	\tikzinputsegmentlast	
\makeatother	\phiSlot 771	 668, 690	
. . . . 28, 51, 52,	\phiTerminal 770, 786, 793	\tikzmath 665, 687	
56, 244, 247, 250,			
271, 305, 325,			

