



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

Yegor Bugayenko
yegor256@gmail.com

2026-10-12, 0.26.4

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 ” (`\upvarphi`),
- “^” maps to “ ρ ” (`\uprho`),
- “\$” maps to “ ξ ” (`\upxi`),
- “?” 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 ” (`\uplambda`),
- “D>” maps to “ $\Delta \mapsto$ ” (`\Delta ..>`),
- “L>” maps to “ $\lambda \mapsto$ ” (`\uplambda ..>`),
- “[[” 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 `\upalpha` bearing an index; for instance, `\phiq{~0 -> x}` produces “ $\alpha_0 \mapsto x$ ”. The form `~0` likewise yields α_0 .

The Greek letters `\rho`, `\xi`, `\lambda`, `\alpha`, and `\varphi`, when typed inside `\phiq` or `phiqutation`, are rendered upright too, as `\uprho`, `\upxi`, `\uplambda`, `\upalpha`, and `\upvarphi` from [upgreek](#), since they are terminals of φ -calculus, not variables: `\phiq{\rho -> \xi}` produces “ $\rho \mapsto \xi$ ”. 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.

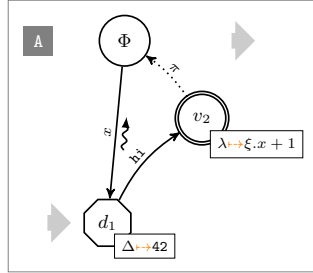
`\phiq` The `\phiq` command permits the inlining of φ -calculus expressions in the same simple plain-text notation. The dollar sign may be employed directly:

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></math>`” puts a custom label `<math></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 the EO language. It is sensitive to the `anonymous` package option, rendering itself 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 \xmirl{} --- \\
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}
```

`\phiNormalize` Five further commands render the arrows of the operational semantics of φ -
`\phiEvaluate` calculus, each painted red rather than the orange of `\phiTerminal`, so that a step
`\phiDataize` of computation is never mistaken for a terminal symbol. They are meant to stand
`\phiMorph` between two expressions, exactly as `\phiWave` and `\phiDotted` do above. All save
`\phiContextualize` `\phiEvaluate` accept an optional subscript, printed in the ordinary text colour,
for instance to name a step count or a substitution:

Normalization, $x \rightsquigarrow y$; evaluation,
 $x \rightarrow y$; dataization, $x \Downarrow_n y$; morphing,
 $x \Rightarrow_\sigma y$; contextualization,
 $x \Rightarrow y$.

```
6 Normalization, $x \phiNormalize y$;
7 evaluation, $x \phiEvaluate y$;
8 dataization, $x \phiDataize[n] y$;
9 morphing, $x \phiMorph[\sigma] y$;
10 contextualization, $x \phiContextualize y$.
11 \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:
 $[\rightarrow^*]$.

```
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 $1 \mapsto x_1, 2 \mapsto x_2, \dots, \alpha_n \mapsto x_n$ and expression $\alpha_i \mapsto x_i$ are syntactically different but semantically equivalent.

```

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

```

\phiSaveTo The **phiq** and **sodg** environments cannot be used directly within **tabular** 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{phiq}** 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: $x \mapsto \emptyset$

Bound: $x \mapsto [\Delta \mapsto 42]$

```

5 \phiSaveTo{a}
6 \begin{phiq*}
7 [[ x -> [[D>42]] ]]
8 \end{phiq*}
9 \begin{tabular}{p{.5in}l}
10 Free: & $ [[x -> ?]]$ \\\
11 Bound: & \parbox{1in}{\input{a}} \\\
12 \end{tabular}

```

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

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

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

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

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

$\text{foo} \mapsto \left\{ \begin{array}{c} \emptyset \\ \llbracket \lambda \mapsto \rho \times \xi. \alpha_0 \rrbracket \\ \llbracket \Delta \mapsto 42 \rrbracket \end{array} \right\}$	<pre> 5 \begin{phiquation*} 6 foo -> \left\{\begin{matrix} \backslash \\ ? \backslash \\ [[L> ^ \times \$. \alpha_0]] \backslash \\ [[D> 42]] \backslash \\ \end{matrix}\right\} 10 \end{matrix}\right\} 11 \end{phiquation*} </pre>
--	---

The `cases` environment is likewise supported:

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

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

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

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

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):

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

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

The object formation $\llbracket \alpha_0 \mapsto x \rrbracket$ may be replaced by the formula $Q \times a^2$.	6	The object formation $\$[\sim 0 \rightarrow x]\$$
	7	may be replaced by the formula
	8	<code>\(Q \times a^2 \)</code> .

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

$x(\pi) \mapsto [\lambda \mapsto f_1]$, $x(a, b, c) \mapsto [\alpha_0 \mapsto \emptyset, \lambda \mapsto \text{Foo}, \phi \mapsto \phi \text{hello}(\xi), x \mapsto \text{false}]$, $\Delta = 43-09$, $x(y) \equiv x(\alpha_0 \mapsto y)$.	5	<code>\begin{phiquation*}</code>
	6	<code>x(\pi) -> [[\lambda \mapsto f_1]], \\\</code>
	7	<code>x(a,b,c) -> [[~0 -> ?, L> Foo, \</code>
	8	<code>@ -> Q.hello(\$), x -> false]], \\\</code>
	9	<code>\Delta = 43-09 ,</code>
	10	<code>x(y) == x(~0 -> y).</code>
	11	<code>\end{phiquation*}</code>

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

$\llbracket b \mapsto \emptyset \rrbracket$,	5	<code>\begin{phiquation*}</code>
$\llbracket \phi \mapsto \text{TRUE}, \Delta \mapsto 42 \rrbracket$,	6	<code>[[b -> ?]],</code>
$\psi = \langle \pi, 42 \rangle$.	7	<code>[[@ -> TRUE, \Delta \mapsto 42]], \\\</code>
	8	<code>\psi = << \pi, 42 >>.</code>
	9	<code>\end{phiquation*}</code>

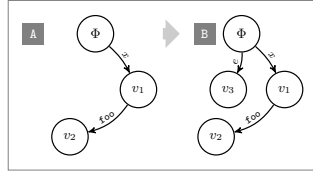
A “manual splitting” mode may be employed in the `phiquation` environment by commencing the body with `\begin{split}`:

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

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

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

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

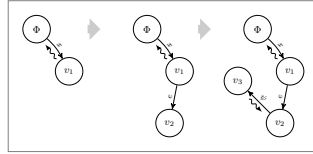


```

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:

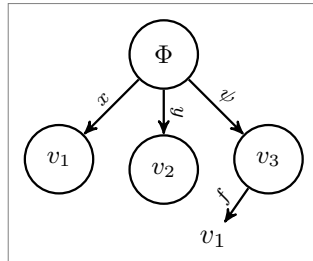


```

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:

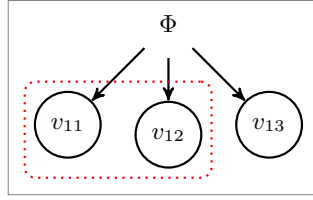


```

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:

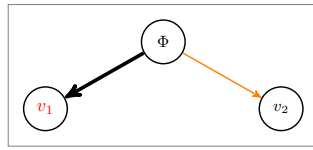


```

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 [upgreek](#) for upright Greek letters, such as `\uprho`:

```
4 \RequirePackage{upgreek}
```

We need [fancyvrb](#) for `\VerbatimEnvironment` command:

```
5 \RequirePackage{fancyvrb}
```

We need [iexec](#) for executing Perl scripts:

```
6 \ifdefined\eolang@noshell\else\RequirePackage{iexec}\fi
```

Then, we process package options:

```

7 \RequirePackage{pgfopts}
8 \RequirePackage{ifluatex}
9 \RequirePackage{ifxetex}
10 \pgfkeys{
11   /eolang/.cd,
12   tmpdir/.store in=\eolang@tmpdir,
13   tmpdir/.default=_eolang\ifxetex-xe\else\ifluatex-lua\fi\fi,
14   nocomments/.store in=\eolang@nocomments,
15   nodollar/.store in=\eolang@nodollar,
16   nocolor/.store in=\eolang@nocolor,

```

```

17 anonymous/.store in=\eolang@anonymous,
18 noshell/.store in=\eolang@noshell,
19 tmpdir
20 }
21 \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:

```

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

```

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

```

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

```

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

```

53 \makeatletter\newcounter{eolang@lineno}\makeatother

```

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

```

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

```

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

```

58 \makeatletter
59 \ifdefined\eolang@noshell
60   \message{eolang: Perl script is not going to be created,
61   at "\eolang@tmpdir/\jobname-phi.pl" because of the "noshell"
62   package option^^J}
63 \else
64   \openin 15=\eolang@tmpdir/\jobname-phi.pl
65   \ifeof 15
66   \message{eolang: Perl script is going to be created,
67   because it is absent at "\eolang@tmpdir/\jobname-phi.pl",
68   but if -shell-escape is not set, the compilation will
69   most likely fail now^^J}
70   \begin{VerbatimOut}{\eolang@tmpdir/\jobname-phi.pl}
71   \macro = $ARGV[0];
72   open(my $fh, '<', $ARGV[1]);
73   my $tex; { local $/; $tex = <$fh>; }
74   print "% This file is auto-generated by eolang.sty 0.26.4\n";
75   print '% There are ', length($tex),
76   ' chars in the input: ', $ARGV[1], "\n";
77   print '% ---', "\n";
78   if (index($tex, "\t") >= 0) {
79     print "TABS are prohibited!";
80     exit 1;
81   }
82   my @lines = split (/\\n/g, $tex);
83   foreach my $t (@lines) {
84     print '% ', $t, "\n";
85   }
86   print '% ---', "\n";
87   $tex =~ s/(?<!\\\)%.*\\n/\\n/g;
88   $tex =~ s/^\\s+|\\s+$///g;
89   my $splitting = $tex =~ /^\\begin\\{split\\}/;
90   if ($splitting) {
91     print '% The manual splitting mode is ON since \\begin{split} started the text' . "\n";
92   }
93   my $indents = $tex =~ /\\n +/g;
94   my $gathered = (0 == $indents);
95   if ($gathered) {
96     if ($splitting) {
97       print '% The "gathered" is NOT used because of manual splitting' . "\n";
98       $gathered = 0;
99     } else {
100       print '% The "gathered" is used since all lines are left-aligned' . "\n";
101     }
102   } else {
103     print '% The "gathered" is NOT used because ' .
104     $indents . " lines are indented\n";
105   }
106   my $align = 0;
107   print '% The "align" is NOT used by default' . "\n";
108   if (index($tex, '&&') >= 0) {
109     $macro =~ s/equation/align/g;
110     $align = 1;
111     print '% The "align" is used because of && seen in the text' . "\n";

```

```

112 }
113 if ($macro ne 'phiq') {
114   if (not $splitting) {
115     $tex =~ s/\\\\\\n\\n\\n/g;
116     $tex =~ s/\\\\n\\s*/g;
117   }
118   $tex =~ s/\\n*(\\label\\{[^\\}+\\})\\n*/\\1/g;
119   $tex =~ s/\\n{3,}\\n\\n/g;
120 }
121 my @texts = ();
122 sub trep {
123   my ($m, $s) = @_ ;
124   my $open = 0;
125   my $p = 0;
126   for (; $p < length($s); $p++) {
127     $c = substr($s, $p, 1);
128     if ($c eq '}') {
129       if ($open == 0) {
130         last;
131       }
132       $open--;
133     }
134     if ($c eq '{') {
135       $open++;
136     }
137   }
138   push(@texts, '\\\\' . $m . '{' . substr($s, 0, $p) . '}');
139   return '{TEXT' . (0+@texts - 1) . '}' . substr($s, $p + 1);
140 }
141 $tex =~ s/\\(text)\\{(\\+)/trep($1, "$2")/ge;
142 1 while $tex =~ s/\\(label|ref|eqref|nameref|pageref|phinoMeet|phinoAgain)\\{(\\+)/trep($1, "$2")/ge;
143 $tex =~ s/(?<[\\s,>()]( [0-9A-F]{2}(?:-[0-9A-F]{2})+)(?!\\{)/|\\1|/g;
144 $tex =~ s/(?<[\\s,>()]( [0-9]+(?:\\.[0-9]+)?)(?!\\{)/|\\1|/g;
145 $tex =~ s/([~\\{a-zA-Z0-9\\|\\^]Q(?:[a-zA-Z0-9])\\1 \\phiTerminal{\\Phi})/g;
146 $tex =~ s/([~\\{a-zA-Z0-9\\|\\^]T(?:[a-zA-Z0-9])\\1 \\phiTerminal{\\bot})/g;
147 $tex =~ s/([~\\{a-zA-Z0-9\\|\\^]D>\\1\\phiTerminal{\\Delta})\\.>/g;
148 $tex =~ s/([~\\{a-zA-Z0-9\\|\\^]L>\\1\\phiTerminal{\\lambda})\\.>/g;
149 $tex =~ s/([~\\{a-zA-Z0-9\\|\\^]D(?:[a-zA-Z0-9])\\1\\phiTerminal{\\Delta})/g;
150 $tex =~ s/([~\\{a-zA-Z0-9\\|\\^]L(?:[a-zA-Z0-9])\\1\\phiTerminal{\\lambda})/g;
151 $tex =~ s/"([~"]+)"|"\\1"/g;
152 $tex =~ s/(?:^|(?<[\\s](\\[\\[,>\\]))([a-zA-Z][a-zA-Z0-9]+)(?=[\\s](\\[\\[,>\\]))/|\\1|/g;
153 $tex =~ s/([~_~]|^)([0-9]+|\\*)\\/(\\{[a-z]+|\\}|[a-z]+|\\)|)
154 (->|\\.|\\.|>|~>|:=)\\1\\alpha_{2}\\\\vert}\\3\\space{\\4}/xg;
155 if ($macro ne 'phiq') {
156   if (not $splitting) {
157     $tex =~ s/\\begin{split}\\n\\begin{split}&/g;
158     $tex =~ s/\\n\\s*\\end{split}\\n\\end{split}/g;
159     $tex =~ s/\\n\\n\\\\\\\\&/g;
160     $tex =~ s/\\n\\n\\phiEOL}\\n&/g;
161     $tex =~ s/\\\\\\\\$/g;
162     $tex =~ s/\\\\\\\\\\\\\\\\\\n/g;
163     $tex =~ s/([~&\\s])\\s{2}([~\\s])/\\1 \\2/g;
164     $tex =~ s/\\s{2}/ \\quad/g;
165     $tex = '&' . $tex;

```

```

166 }
167 my $lead = '[^\\s]+\\s(?:->|:=|==)\\s';
168 my @leads = $tex =~ /&${lead}/g;
169 my @eols = $tex =~ /&/g;
170 if (0+@leads == 0+@eols && 0+@eols > 1) {
171     $tex =~ s/&(${lead})/\\1~/g;
172     $gathered = 0;
173     print '% The "gathered" is NOT used because all ' .
174         (0+@eols) . ' lines are ' . (0+@leads) . " leads\n";
175 }
176 }
177 if ($macro ne 'phiq') {
178     sub strip_tabs {
179         my ($env, $tex) = @_;
180         $tex =~ s/& //g;
181         return "\\begin{$env}" . $tex . "\\end{$env}";
182     }
183     foreach my $e (('matrix', 'cases')) {
184         $tex =~ s/\\begin\\{(\\Q$e\\E*?)\\}(\\.+)\\end\\{(\\Q$e\\E*?)\\}/strip_tabs($1, $2)/sge;
185     }
186 }
187 my @pipes = ();
188 $tex =~ s/\\|{2,}/|/g;
189 $tex =~ s/\\|([~|]+)\\|/push(@pipes, '\\textnormal{\\texttt{' . ($1 =~ s/-(?=-)}{-}}gr) . ')}'/g;
190 $tex =~ s/\\$/\\phiTerminal{\\xi}/g;
191 $tex =~ s/(?!\\{)\\^(?!\\{)/\\phiTerminal{\\rho}/g;
192 $tex =~ s/\\[\\[/\\phiTerminal{\\llbracket}/g;
193 $tex =~ s/\\]\\[/\\phiTerminal{\\rrbracket}/g;
194 $tex =~ s/\\?/\\phiTerminal{\\varnothing}/g;
195 $tex =~ s/\\@/\\phiTerminal{\\varphi}/g;
196 $tex =~ s/\\-([a-z]+)/\\mathrel{\\phiSlot{\\1}}/g;
197 $tex =~ s/\\->/\\mathbin{\\phiTerminal{\\mapsto}}/g;
198 $tex =~ s/\\->/\\mathbin{\\phiWave}/g;
199 $tex =~ s/\\:=/\\mathrel{\\vDash}/g;
200 $tex =~ s/\\(?<![\\{\\}):/\\mathord{\\phiTerminal{:}}/g;
201 $tex =~ s/\\==/\\mathrel{\\equiv}/g;
202 $tex =~ s/\\\\.\\>/\\mathbin{\\phiTerminal{\\phiDotted}}/g;
203 $tex =~ s/\\-([0-9]+)/\\phiTerminal{\\alpha_{\\1}}/g;
204 $tex =~ s/\\(\\rho|\\xi|\\lambda|\\alpha|\\varphi)(?![a-zA-Z])/\\up{1}/g;
205 $tex =~ s/\\<</\\langle/g;
206 $tex =~ s/\\>>/\\rangle/g;
207 $tex =~ s/\\(?<![\\{\\})([\\{\\.])/\\phiTerminal{\\1}/g;
208 $tex =~ s/\\(?<![\\{\\}),/\\phiTerminal{,}\\;/g;
209 $tex =~ s/\\{PIPE\\(d+\\)\\}/$pipes[$1]/g;
210 $tex =~ s/\\{TEXT\\(d+\\)\\}/$texts[$1]/g;
211 if ($macro eq 'phiq') {
212     print '\\(' if ($tex ne '');
213 } else {
214     print '\\begin{' . $macro . " \\n";
215     if (not($align)) {
216         if ($gathered) {
217             print '\\begin{gathered}' . "\\n";
218         } elsif (not $splitting) {
219             print '\\begin{split}' . "\\n";
220         }
221     }
222 }

```

```

220     }
221   }
222 }
223 if ($gathered and not($align)) {
224   $tex =~ s/^&//g;
225   $tex =~ s/\n&/\n/g;
226 }
227 print $tex;
228 if ($macro eq 'phiq') {
229   print '\)' if ($tex ne '');
230 } else {
231   if (not($align)) {
232     if ($gathered) {
233       print "\n" . '\end{gathered}';
234     } elsif (not $splitting) {
235       print "\n" . '\end{split}';
236     }
237   }
238   print "\n" . '\end{' . $macro . '}';
239 }
240 print '\endinput';
241 \end{VerbatimOut}
242 \message{eolang: File with Perl script
243   '\eolang@tmpdir/\jobname-phi.pl' saved^^J}
244 \else
245   \message{eolang: Perl script already exists at
246     "\eolang@tmpdir/\jobname-phi.pl"^^J}
247 \fi
248 \closein 15
249 \fi
250 \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:

```

251 \makeatletter
252 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
253 \let\eolang@phiSaveTo\relax
254 \makeatother

```

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

```

255 \makeatletter
256 \newcommand\eolang@tmp[1]{#1\ifxetex-xe\else\ifluatex-lua\fi\fi.tex}
257 \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:

```

258 \makeatletter
259 \newcommand\eolang@ifabsent[2]{%
260   \eolang@ifabsentwith{#1}{\input{#1}}{#2}%
261 }
262 \newcommand\eolang@ifabsentwith[3]{%
263   \IfFileExists
264     {#1}
265     {%

```

```

266     \message{eolang: File "#1" already exists ^^J}%
267     #2}
268     {%
269     \ifdefined\eolang@noshell%
270     \message{eolang: Shell processing is disabled^^J}%
271     \else%
272     \ifnum\ShellEscapeStatus=1\else%
273     \message{eolang: The -shell-escape command line
274     option is not provided, most probably compilation
275     will fail now:^^J}%
276     \fi%
277     #3%
278     \fi%
279     }%
280 }
281 \makeatother

```

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

```

282 \makeatletter\newcommand\eolang@process[1]{
283   \def\hash{\eolang@mdfive
284     {\eolang@tmpdir/\jobname/\eolang@tmp{phiquation}}-\the\inputlineno}%
285   \eolang@ifabsentwith
286     {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation-post}}
287     {\ifx\eolang@phiSaveTo\relax
288       \input{\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation-post}}%
289       \fi}
290   {%
291     \iexec[null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{phiquation}"
292       "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation}"}%
293     \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
294     \ifx\eolang@phiSaveTo\relax
295       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation-post}]{
296         perl "\eolang@tmpdir/\jobname-phi.pl"
297         '#1'
298         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation}"
299         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi}%
300     \else
301       \iexec[trace,quiet,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation-post}]{
302         perl "\eolang@tmpdir/\jobname-phi.pl"
303         '#1'
304         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation}"
305         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi}%
306     \fi
307   }%
308   \ifx\eolang@phiSaveTo\relax
309   \else
310     \ifdefined\eolang@noshell\else
311     \IfFileExists
312       {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation-post}}
313       {\iexec[null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation-post}" "\eolang@tmpdir/\jobname-phi.pl"}%
314       \fi
315     \aftergroup\eolang@clearPhiSaveTo
316   \fi

```

```

317 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
318 }
319 \def\eolang@clearPhiSaveTo{\let\eolang@phiSaveTo\relax}
320 %
321 \newenvironment{phiuation*}%
322 {\catcode'\|=12 \VerbatimEnvironment%
323 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
324 \begin{VerbatimOut}
325   {\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}}
326 {\end{VerbatimOut}\eolang@process{equation*}}
327 %
328 \newenvironment{phiuation}%
329 {\catcode'\|=12 \VerbatimEnvironment%
330 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
331 \begin{VerbatimOut}
332   {\eolang@tmpdir/\jobname/\eolang@tmp{phiuation}}
333 {\end{VerbatimOut}\eolang@process{equation}}
334 \makeatother

```

\phiq Then, we define \phiq command:

```

335 \RequirePackage{xstring}
336 \makeatletter\newcommand\phiq[1]{%
337   \StrSubstitute{\detokenize{#1}}{'',''}[\clean]%
338   \def\hash{\pdfmdfivesum{\clean}-\the\inputlineno}%
339   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
340   \eolang@ifabsent
341     {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq-post}}
342     {%
343       \iexec[log,trace,quiet,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{phiq}]{
344         printf '%s' '\clean'}%
345       \iexec[quiet,null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{phiq}"
346         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq}"}%
347       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq-post}]{
348         perl \eolang@tmpdir/\jobname-phi.pl 'phiq'
349         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq}"
350         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g' \fi}%
351       \message{eolang: Parsed 'phiq' at line no. \the\inputlineno^~J}%
352     }%
353   \ifdefined\eolang@nodollar\else\catcode'\$=\active\fi%
354 }\makeatother

```

nodollar Then, we redefine dollar sign:

```

355 \ifdefined\eolang@nodollar\else
356   \begingroup
357   \catcode'\$=\active
358   \protected\gdef\${#1}\phiq{#1}}
359   \endgroup
360   \AtBeginDocument{\catcode'\$=\active}
361 \fi

```

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

[fancyvrb](#):

```

362 \makeatletter
363 \ifdefined\eolang@noshell

```

```

364 \message{eolang: Perl script is not going to be created
365   at "\eolang@tmpdir/\jobname-sodg.pl", because of the
366   "noshell" package option^^J}
367 \else
368 \openin 15=\eolang@tmpdir/\jobname-sodg.pl
369 \ifeof 15
370 \message{eolang: Perl script is going to be created,
371   because it is absent at "\eolang@tmpdir/\jobname-sodg.pl",
372   but if -shell-escape is not set, the compilation will
373   most likely fail now^^J}
374 \begin{VerbatimOut}{\eolang@tmpdir/\jobname-sodg.pl}
375 sub num {
376   my ($i) = @_;
377   $i =~ s/(\+|-)\./\10./g;
378   return $i;
379 }
380 sub fmt {
381   my ($tex) = @_;
382   $tex =~ s/\\|([^\|]+)\\|'\textnormal{\texttt{' . ($1 =~ s{-(?=-)}{-{}}gr} . '}}'/ge;
383   return $tex;
384 }
385 sub toem {
386   my ($cm) = @_;
387   return $cm * 2.8;
388 }
389 sub vertex {
390   my ($v) = @_;
391   if (index($v, 'v0') == 0) {
392     return '\Phi';
393   } else {
394     $v =~ s/^v/v_{}/g;
395     $v =~ s/[^0-9]$//g;
396     return $v . '>';
397   }
398 }
399 sub tailor {
400   my ($t, $m) = @_;
401   $t =~ s/<([A-Z]?${m}[A-Z]?):([>]+)>/\2/g;
402   $t =~ s/<[A-Z]+:[>]+>/\2/g;
403   return $t;
404 }
405 open(my $fh, '<', $ARGV[0]);
406 my $tex; { local $/; $tex = <$fh>; }
407 if (index($tex, "\t") >= 0) {
408   print "TABS are prohibited!";
409   exit 1;
410 }
411 print '% This file is auto-generated', "\n\n";
412 print '% --- there are ', length($tex),
413   ' chars in the input (', $ARGV[0], "):\n";
414 foreach my $t (split /\n/g, $tex) {
415   print '% ', $t, "\n";
416 }
417 print "% ---\n";

```

```

418 $tex =~ s/\\\\\\/\n/g;
419 $tex =~ s/\\\n//g;
420 $tex =~ s/(\\[a-zA-Z]+\)\s+/\1/g;
421 $tex =~ s/\n{2,}/\n/g;
422 my @cmds = split(/\n/g, $tex);
423 print '% --- before processing:' . "\n";
424 foreach my $t (split(/\n/g, $tex)) {
425     print '% ', $t, "\n";
426 }
427 print '% ---';
428 print ' (' . (0+@cmds) . " lines)\n";
429 print '\begin{picture}', "\n";
430 for (my $c = 0; $c < 0+@cmds; $c++) {
431     my $cmd = $cmds[$c];
432     $cmd =~ s/^\s+//g;
433     $cmd =~ s/(?<!\)\%.*//g;
434     my ($head, $tail) = split(/ /, $cmd, 2);
435     my %opts = ();
436     my ($body, $style) = split(/style:/, $tail, 2);
437     $opts{'style'} = $style;
438     $tail = $body;
439     foreach my $p (split(/ /, $tail)) {
440         my ($q, $t) = split(/:/, $p);
441         $opts{$q} = $t;
442     }
443     if (index($head, '\\') == 0) {
444         print $cmd;
445     } elsif (index($head, '->') >= 0) {
446         my $draw = '\draw[';
447         if (exists $opts{'pi'}) {
448             $draw = $draw . '<MB:phi-pi><F:draw=none>';
449             if (not exists $opts{'a'}) {
450                 $opts{'a'} = '\pi';
451             }
452         }
453         if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
454             $draw = $draw . '<MB:,phi-rho>';
455         }
456         $draw = $draw . ',' . $opts{'style'} . ']';
457         my ($from, $to) = split (/->/, $head);
458         $draw = $draw . " ($from) ";
459         if (exists $opts{'bend'}) {
460             $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
461                 num($opts{'bend'}) . '>';
462             if (exists $opts{'rho'}) {
463                 $draw = $draw . '<MB:,phi-rho>';
464             }
465             $draw = $draw . ']';
466         } else {
467             $draw = $draw . '--';
468         }
469         if (exists $opts{'a'}) {
470             my $a = $opts{'a'};
471             if (index($a, '$') == -1) {

```

```

472     $a = '$' . fmt($a) . '$';
473 } else {
474     $a = fmt($a);
475 }
476 $draw = $draw . '<MB: node [phi-attr] {' . $a . '}>';
477 }
478 if (exists $opts{'break'}) {
479     $draw = $draw . '<F: coordinate [pos=' .
480         ($opts{'break'} / 100) . '] (break)>';
481 }
482 $draw = $draw . " (<MF:${to}><B:break-v>";
483 if (exists $opts{'break'}) {
484     print tailor($draw, 'F') . ";\n";
485     print ' \node[outer sep=' . toem(0.1) . 'em,inner sep=0em] ' .
486         'at (break) (break-v) {' . vertex($to) .
487         '$};' . "\n";
488     print ' ' . tailor($draw, 'B');
489 } else {
490     print tailor($draw, 'M');
491 }
492 } elsif (index($head, '=>') >= 0) {
493     my ($from, $to) = split (/=>/, $head);
494     my $size = () = $head =~ /=/g;
495     if ($from eq '') {
496         print '\node [phi-arrow, left=' . toem($size * 0.6) . 'em of ' .
497             $to . '.center]';
498     } elsif ($to eq '') {
499         print '\node [phi-arrow, right=' . toem($size * 0.6) . 'em of ' .
500             $from . '.center]';
501     } else {
502         print '\node [phi-arrow] at $(' .
503             $from . ')!0.5!(' . $to . ')$)';
504     }
505     print '{}';
506 } elsif (index($head, '!=') >= 0) {
507     my ($v, $marker) = split (/!=/, $head);
508     my $size = () = $head =~ /*!/g;
509     print '\node [phi-marker, left=' .
510         toem($size * 0.6) . 'em of ' .
511         $v . '.center'][' . fmt($marker) . ']';
512 } elsif (index($head, '+') >= 0) {
513     my ($v, $suffix) = split (/+/, $head);
514     my @friends = ($v);
515     foreach my $c (@cmds) {
516         $e = $c;
517         $e =~ s/^\s+//g;
518         my $h = $e;
519         $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
520         foreach my $f (@friends) {
521             my $add = '';
522             if (index($h, $f . '->') >= 0) {
523                 $add = substr($h, index($h, '->') + 2);
524             }
525             if ($h =~ /->\Q${f}\E$/) {

```

```

526         $add = substr($h, 0, index($h, '->'));
527     }
528     if (index($e, ' xy:' . $f . ',') >= 0) {
529         $add = $h;
530     }
531     if (index($add, '+') == -1
532         and $add ne ''
533         and not(grep(/^Q${add}\E$/, @friends))) {
534         push(@friends, $add);
535     }
536 }
537 }
538 my @extra = ();
539 foreach my $e (@cmds) {
540     $m = $e;
541     if ($m =~ /\s*Q${v}\E\s/) {
542         next;
543     }
544     if ($m =~ /\s*[^s]+\s/ and not($m =~ /\s*Q${head}\E\s/)) {
545         next;
546     }
547     foreach my $f (@friends) {
548         my $h = $f;
549         $h =~ s/[a-z]$//g;
550         if ($m =~ s/^(s*)Q${f}\E\+Q${suffix}\E\s?/\1${h}${suffix} /g) {
551             last;
552         }
553         $m =~ s/^(s*)Q${f}\E\s/\1${h}${suffix} /g;
554         $m =~ s/^(s*)Q${f}\E->/\1${h}${suffix}->/g;
555         $m =~ s/\sxy:Q${f}\E,/ xy:${h}${suffix},/g;
556         $m =~ s/->Q${f}\E\s/->${h}${suffix} /g;
557     }
558     if ($m ne $e) {
559         push(@extra, ' ' . $m);
560     }
561 }
562 splice(@extra, 0, 0, $extra[-1]);
563 splice(@extra, -1, 1);
564 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
565     '), friends: [' . join(', ', @friends) . ']' in ' .
566     '(0+@cmds) . ' lines');
567 splice(@cmds, $c, 1, @extra);
568 print '% cloned ' . $v . ' at line no.' . $c .
569     ' (+ ' . (0+@extra) . ' lines -> ' .
570     '(0+@cmds) . ' lines total)';
571 } elsif ($head =~ /\v[0-9]+[a-z]?$/) {
572     print '\node[';
573     if (exists $opts{'xy'}) {
574         my ($v, $right, $down) = split(/,/, $opts{'xy'});
575         my $loc = '';
576         if ($down > 0) {
577             $loc = 'below ';
578         } elsif ($down < 0) {
579             $loc = 'above ';

```

```

580     }
581     if ($right > 0) {
582         $loc = $loc . 'right';
583     } elseif ($right < 0) {
584         $loc = $loc . 'left';
585     }
586     print ', ' . $loc . '=';
587     print toem(abs(num($down))) . 'em and ' .
588         toem(abs(num($right))) . 'em of ' . $v . '.center';
589 }
590 if (exists $opts{'data'}) {
591     print ',phi-data';
592     if ($opts{'data'} ne '') {
593         my $d = $opts{'data'};
594         if (index($d, '|') == -1) {
595             $d = '$\Delta\phiDotted\text{' .
596                 '\textnormal{\texttt{' . fmt($d) . '}}}$';
597         } else {
598             $d = fmt($d);
599         }
600         $opts{'box'} = $d;
601     }
602 } elseif (exists $opts{'atom'}) {
603     print ',phi-atom';
604     if ($opts{'atom'} ne '') {
605         my $a = $opts{'atom'};
606         if (index($a, '$') == -1) {
607             $a = '$\lambda\phiDotted{' . fmt($a) . '$';
608         } else {
609             $a = fmt($a);
610         }
611         $opts{'box'} = $a;
612     }
613 } else {
614     print ',phi-object';
615 }
616 if (exists $opts{'edgeless'}) {
617     print ',draw=none';
618 }
619 print ', ' . $opts{'style'} . ']';
620 print ' (' . $head . ')';
621 print '{';
622 if (exists $opts{'tag'}) {
623     my $t = $opts{'tag'};
624     if (index($t, '$') == -1) {
625         $t = '$' . $t . '$';
626     } else {
627         $t = fmt($t);
628     }
629     print $t;
630 } else {
631     print '$' . vertex($head) . '$';
632 }
633 print '}';

```

```

634     if (exists $opts{'box'}) {
635         print ' node[phi-box] at (';
636         print $head, '.south east) {';
637         print $opts{'box'}, '}'';
638     }
639 }
640 print ";\n";
641 }
642 print '\end{picture}%', "\n";
643 print "% --- after processing:\n%";
644 foreach my $c (@cmds) {
645     print '% ', $c, "\n";
646 }
647 print '% --- (' . (0+@cmds) . " lines)\n";
648 print '\endinput';
649 \end{VerbatimOut}
650 \message{eolang: File with Perl script
651   '\eolang@tmpdir/\jobname-sodg.pl' saved^^J}
652 \else
653   \message{eolang: Perl script already exists at
654     "\eolang@tmpdir/\jobname-sodg.pl"^^J}
655 \fi
656 \closein 15
657 \fi
658 \makeatother

```

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

```

659 \setcounter{FancyVerbLine}{0}

```

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

```

660 \RequirePackage{tikz}
661 % Keep TikZ's inline math syntax available while the document uses active dollars.
662 \tikzset{every picture/.append style={execute at begin picture={\catcode'\$=3\relax}}}
663 \usetikzlibrary{arrows}
664 \usetikzlibrary{shapes}
665 \usetikzlibrary{decorations}
666 \usetikzlibrary{decorations.pathmorphing}
667 \usetikzlibrary{decorations.pathreplacing}
668 \usetikzlibrary{positioning}
669 \usetikzlibrary{calc}
670 \usetikzlibrary{math}
671 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment `picture`:

```

672 \newenvironment{picture}%
673   {\noindent\begin{tikzpicture}[
674     ->,>=stealth',node distance=0,line width=.08em,
675     pics/parallel arrow/.style={
676       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}]}%
677   {\end{tikzpicture}}
678 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
679   minimum height=0.05em, minimum width=0.05em,
680   single arrow head extend=2mm]

```

```

681 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
682   minimum width=1.4em, font={\small\color{white}\ttfamily},
683   fill=gray]
684 \tikzstyle{phi-thing} = [inner sep=0pt,minimum height=2.4em,
685   draw,font={\small}]
686 \tikzstyle{phi-object} = [phi-thing,circle]
687 \tikzstyle{phi-data} = [phi-thing,regular polygon,
688   regular polygon sides=8]
689 \tikzstyle{phi-empty} = [phi-object]
690 \tikzset{%
691   phi-rho/.style={
692     postaction={%
693       decoration={
694         show path construction,
695         curveto code={
696           \tikzmath{
697             coordinate \I, \F, \v;
698             \I = (\tikzinputsegmentfirst);
699             \F = (\tikzinputsegmentlast);
700             \v = ($(\I) -(\F)$);
701             real \d, \a, \r, \t;
702             \d = 0.8;
703             \t = atan2(\vy, \vx);
704             if \vx<0 then { \a = 90; } else { \a = -90; };
705             {
706               \draw[arrows={-latex}, decorate,
707                 decoration={%
708                   snake, amplitude=.4mm,
709                   segment length=2mm,
710                   post length=1mm
711                 }]]
712               ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
713               -- ++(\t: 2*\d em);
714             };
715           }
716         },
717         lineto code={
718           \tikzmath{
719             coordinate \I, \F, \v;
720             \I = (\tikzinputsegmentfirst);
721             \F = (\tikzinputsegmentlast);
722             \v = ($(\I) -(\F)$);
723             real \d, \a, \r, \t;
724             \d = 0.8;
725             \t = atan2(\vy, \vx);
726             if \vx<0 then { \a = 90; } else { \a = -90; };
727             {
728               \draw[arrows={-latex}, decorate,
729                 decoration={%
730                   snake, amplitude=.4mm,
731                   segment length=2mm,
732                   post length=1mm}]
733               ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
734               -- ++(\t: 2*\d em);

```

```

735         };
736     }
737 }
738 },
739     decorate
740 }
741 }
742 }
743 \tikzstyle{phi-pi} = [draw,dotted]
744 \tikzstyle{phi-atom} = [phi-object,double]
745 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
746     rectangle,line width=.04em,minimum width=1.2em,anchor=north west,
747     font={\scriptsize}]
748 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
749     above=2pt,sloped/.append style={transform shape},
750     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:

```

751 \makeatletter
752 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
753 \let\eolang@sodgSaveTo\relax
754 \makeatother

```

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

```

755 \makeatletter\newenvironment{sodg}%
756 {\catcode'\|=12 \VerbatimEnvironment%
757 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
758 \begin{VerbatimOut}
759   {\eolang@tmpdir/\jobname/\eolang@tmp{sodg}}}
760 {\end{VerbatimOut}%
761   \def\hash{\eolang@mdfive
762     {\eolang@tmpdir/\jobname/\eolang@tmp{sodg}}-\the\inputlineno}%
763   \catcode'\$=3 %
764   \eolang@ifabsentwith
765     {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg-post}}
766     {\ifx\eolang@sodgSaveTo\relax
767       \input{\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg-post}}%
768     \fi}
769   {%
770     \iexec[null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{sodg}"
771       "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg}"}%
772     \message{eolang: Start parsing 'sodg' at line no. \the\inputlineno^^J}
773     \ifx\eolang@sodgSaveTo\relax
774       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg-post}]{
775         perl "\eolang@tmpdir/\jobname-sodg.pl"
776         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg}"
777         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi}%
778     \else
779       \iexec[trace,quiet,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg-post}]{
780         perl "\eolang@tmpdir/\jobname-sodg.pl"
781         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg}"
782         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi}%
783     \fi

```

```

784     }
785     \ifx\eolang@sodgSaveTo\relax
786     \else
787         \ifdefined\eolang@noshell\else
788             \IfFileExists
789                 {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg-post}}
790                 {\iexec[null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-sodg-post}" "\eolang@sodg
791             \fi
792         \aftergroup\eolang@clearSodgSaveTo
793     \fi
794     \catcode'\$ \active%
795     \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
796 } \def\eolang@clearSodgSaveTo{\let\eolang@sodgSaveTo\relax}\makeatother

```

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

```

797 \RequirePackage{hyperref}
798 \pdfstringdefDisableCommands{
799     \def\({}%
800     \def\)}{%
801     \def\alpha{\alpha}%
802     \def\varphi{\phi}%
803     \def\upalpha{\alpha}%
804     \def\upvarphi{\phi}%
805 }
806 \makeatletter
807 \NewExpandableDocumentCommand{\eoAnon}{O{ANONYMIZED}m}{%
808     \ifdefined\eolang@anonymous%
809         \eolang@ink{#1}%
810     \else%
811         #2%
812     \fi%
813 } \makeatother

```

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

```

814 \newcommand\eolang{%
815     \eoAnon[XYZ]{\sffamily EO}}

```

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

```

816 \newcommand\phic{%
817     \eoAnon[(\upalpha)-cal-cu-lus]{\(\upvarphi)-cal-cu-lus}}

```

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

```

818 \newcommand\xmirl{%
819     \eoAnon[XML\(^+\)]{XMIR}}

```

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

```

820 \newcommand\phiWave{%
821     \phiTerminal{\mapstochar\mathrel{\mspace{0.45mu}}\leadsto}}

```

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

```
822 \newcommand\phiSlot[1]{%
823   \xrightarrow{\text{\sffamily\scshape #1}}}
```

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

```
824 \makeatletter
825 \newcommand\phi0set[2]{%
826   \mathrel{\mathop{#2}\limits^{
827     \vbox to 0ex{\kern-2\ex@
828       \hbox{\scriptscriptstyle#1}\vss}}}}
829 \newcommand\phiUset[2]{%
830   \mathrel{\mathop{#2}\limits_{
831     \vbox to 0ex{\kern-6.3\ex@
832       \hbox{\scriptscriptstyle#1}\vss}}}}
833 \makeatother
```

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

```
834 \newcommand\phiMany[3]{%
835   \phi0set{#3}{\phiUset{#2}{#1}}}
```

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

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

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

```
837 \makeatletter
838 \newcommand\phiTerminal[1]{\eolang@ink{#1}}
839 \makeatother
```

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

```
840 \RequirePackage{trimclip}
841 \RequirePackage{amsfonts}
842 \makeatletter
843 \newcommand\phiDotted{%
844   \phiTerminal{\mapstochar\mathrel{\mathpalette\phiDotted@relax}}}
845 \newcommand\phiDotted@[2]{%
846   \begingroup%
847   \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
848   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
849   \sbox\z@{%
850     \makebox[\dimen\z@][s]{%
851       \clipbox{0 0 {0.4\width} 0}%
852       {\resizebox{\dimen\z@}{\height}%
853         {\m@th#1\dashrightarrow}}}%
854     \hss%
855     \clipbox{{0.69\width} {-0.1\height} 0
856       {-\height}}{\m@th#1\rightarrow}%
857   }%
858   }%
859   \ht\z@=\dimen\tw@ \dp\z@=\z@%
860   \box\z@%
861   \endgroup%
```

```

862 }
863 \makeatother

\eolang@arrow Then, we define an internal command that paints a relational arrow red, leaving it
untouched when the nocolor package option is set, mirroring \eolang@ink above,
save for the colour, which is kept apart from that of \phiTerminal so that arrows
of operational semantics are never mistaken for terminals. The \makeatletter
below stays in effect until the matching \makeatother, well past the end of this
macro, since every command that follows refers to \eolang@arrow by name:

864 \makeatletter
865 \ifdefined\eolang@nocolor
866   \newcommand\eolang@arrow[1]{\mathrel{#1}}
867 \else
868   \newcommand\eolang@arrow[1]{\mathrel{\color{red}#1}}
869 \fi

\eolang@sub Then, we define an internal command that attaches an optional subscript to
an arrow, in the ordinary text colour, since a subscript is not itself a step of
computation:

870 \newcommand\eolang@sub[1]{\IfStrEq{#1}{-}{-}_{#1}}

\phiNormalize Then, we define a command for the normalization arrow of  $\phi$ -calculus, taking an
optional subscript, for instance a step count:

871 \newcommand\phiNormalize[1][]{\eolang@arrow\leadsto\eolang@sub{#1}}

\phiEvaluate Then, we define a command for the evaluation arrow of  $\phi$ -calculus:

872 \newcommand\phiEvaluate{\eolang@arrow{%
873   \vcenter{\hbox{\begin{tikzpicture}
874     \draw[-{Triangle[length=1.8mm,width=1.8mm]},line width=0.6pt]
875       (0,0)--(1.3em,0);
876     \end{tikzpicture}}}}}

\phiDataize Then, we define a command for the dataization arrow of  $\phi$ -calculus, taking an
optional subscript:

877 \newcommand\phiDataize[1][]{\eolang@arrow\Downarrow\eolang@sub{#1}}

\phiMorph Then, we define a command for the morphing arrow of  $\phi$ -calculus, taking an
optional subscript:

878 \newcommand\phiMorph[1][]{\eolang@arrow\Rightarrow\eolang@sub{#1}}

\phiContextualize Then, we define a command for the contextualization arrow of  $\phi$ -calculus, taking
an optional subscript:

879 \newcommand\phiContextualize[1][]{\eolang@arrow\Rightarrow\eolang@sub{#1}}
880 \makeatother

881 \endinput

```

References

- Bugayenko, Yegor (2021). *EOLANG and φ -Calculus*. arXiv: [2111.13384](#) [cs.PL].
- Kudasov, Nikolai et al. (2022). *φ -Calculus: A Purely Object-Oriented Calculus of Decorated Objects*. arXiv: [2204.07454](#) [cs.PL].

Change History

0.0.1		phiuation , another one for	
	General: First draft.	sodg	18
0.0.2		0.12.1	
	sodg : The environment phigure	- sodg.pl : The bug is fixed related	
	renamed to sodg for the sake of	to the formatting of indexes of	
	better semantics. The graph in	vertices.	18
	the picture is solely a SODG	0.13.0	
	graph, that's why the name	- phi.pl : Parsing of QQ into	
	sodg is better.	\dot{\Phi} implemented. . . .	12
	26	0.14.0	
	- phi.pl : New symbol added for	- sodg.pl : The edgeless tag of a	
	basket slots	vertex removes the border of it. 18	
	12	0.15.0	
	Parsing of the symbols " @ ," " ^ ,"	- sodg.pl : The style tag of	
	and " & " enabled (\varphi ,	vertices and edges.	18
	\rho , and \sigma)	0.16.0	
	12	phiuation : The processing of	
	The symbols " [" and "] "	phiuation data is done only if	
	replaced with " [[" and "]] " for	it's the first time processing,	
	abstract object brackets,	otherwise cache is used, thus	
	because they conflicted with	making processing faster. . . .	17
	normal square brackets	sodg : The processing of sodg data	
	12	is done only if it's the first time	
	\phiq : Parsing of additional	processing, otherwise cache is	
	symbols enabled.	used, thus making processing	
	18	faster.	26
	- sodg.pl : The Perl file now has a	0.17.0	
	fixed name, which doesn't	\eolang@ifabsent : A new	
	depend on the name of the TeX	supplementary	
	job. This file may be shared	eolang@ifabsent command	
	among jobs, no need to make it	added	16
	uniquely named.	0.18.0	
	18	\eolang@ifabsent : The noshell	
0.1.0		package option added in order	
	General: Parsing of package options	to enable complete prohibition	
	introduced.	of shell interactions.	16
	11	0.18.3	
	\eolang : New command \eolang	\eolang@tmp : A new	
	added to print the name of the	supplementary eolang@tmp	
	language in both normal and	command added, to generate	
	the anonymous mode of	temporary file names.	16
	acmart	0.19.0	
	27	- phi.pl : Parsing of T into \bot	
	\eolang@mdfive : New	implemented.	12
	supplementary command added	0.2.0	
	to calculate MD5 sum of a file. 12	- phi.pl : Numbers automatically	
	- phi.pl : A new Perl script	render as \texttt . No need to	
	" eolang-phi.pl " added for	use vertical bars around them	
	parsing of phi expressions. . . .	anymore.	12
	12	- sodg.pl : The content of the atom	
	\phic : New command \phic prints	and the data boxes is parsed	
	the name of ϕ -calculus in both		
	normal and the anonymous		
	mode of acmart		
	27		
	\phiDotted : New command		
	\phiDotted added to denote a		
	link to a special attribute. . . .		
	28		
	- sodg.pl : There are two Perl		
	scripts now: one for		

automatically as formulas and numbers, respectively.	18	package option.	28
\xmir: New command \xmir prints XMIR in both normal and the anonymous mode of acmart. . .	27	0.25.0	
0.20.0		-phi.pl: Parsing of : into an orange colon (\phiTerminal{:}) implemented.	12
-phi.pl: Parsing of 0 into \alpha_0 implemented.	12	0.26.0	
0.20.2		\eolang@arrow: New internal command \eolang@arrow paints a relational arrow red, honouring the nocolor package option.	29
-phi.pl: Formatting of numbers fixed, now it works in any place inside an expression.	12	\eolang@sub: New internal command \eolang@sub attaches an optional subscript, in the ordinary text colour, to an arrow.	29
0.21.0		-phi.pl: The Greek letters \rho, \xi, \lambda, \alpha, and \varphi are rendered upright, with the help of the upgreek package.	12
-phi.pl: Automated formatting of TRUE and FALSE removed. . .	12	\phiContextualize: New command \phiContextualize added, to render the contextualization arrow of ϕ -calculus, with an optional subscript.	29
\phiTerminal: New command \phiTerminal added, to highlight all terminals in phi expressions.	28	\phiDataize: New command \phiDataize added, to render the dataization arrow of ϕ -calculus, with an optional subscript.	29
0.22.0		\phiEvaluate: New command \phiEvaluate added, to render the evaluation arrow of ϕ -calculus.	29
-phi.pl: Automated formatting of D and L added.	12	\phiMorph: New command \phiMorph added, to render the morphing arrow of ϕ -calculus, with an optional subscript. . .	29
0.23.0		\phiNormalize: New command \phiNormalize added, to render the normalization arrow of ϕ -calculus, with an optional subscript.	29
-phi.pl: Parsing of QQ into \dot{\Phi} removed.	12	0.26.1	
0.23.1		-phi.pl: The arguments of \label, \ref, \eqref, \nameref, and \pageref are not processed inside phiuation, so label names may contain colons. . .	12
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.	18	0.26.2	
0.23.2		-phi.pl: The first arguments of	
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.	18		
0.24.0			
General: The nocolor package option added, in order to suppress the orange colour. . .	11		
\eolang@ink: New internal command \eolang@ink paints its argument, honouring the nocolor package option. . . .	12		
\phiTerminal: The \phiTerminal command respects the nocolor			

	<code>\phinoMeet</code> and <code>\phinoAgain</code> are not processed inside <code>phiquation</code> , so their label keys may contain colons.	12		<code>nocomments</code> added in order to enable comments suppression in temporary <code>.tex</code> files (may be pretty important for <code>.dtx</code> documents).	11
0.26.3	<code>-phi.pl</code> : The dollar sign keeps its math-shift catcode inside TikZ pictures.	12		<code>-sodg.pl</code> : The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations.	18
0.26.4	<code>-phi.pl</code> : The colon is rendered inside <code>phiquation</code> without spaces around it, like a dot. . .	12	0.7.0	<code>nodollar</code> : Now it is possible to use dollar sign instead of the <code>\phi</code> command.	18
0.3.0	<code>\eolang@lineno</code> : New counter for protecting <code>lineno</code>	12		<code>-phi.pl</code> : New syntax sugar for Φ , just using capital “Q” is enough. . .	12
	<code>-phi.pl</code> : New arrow added, that looks like <code>\leadsto</code>	12		Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols.	12
	<code>\phiWave</code> : New command <code>\phiWave</code> added to denote a link to a multi-layer attribute.	27		Text in quotes is automatically converted to <code>\texttt</code>	12
0.4.0	<code>-sodg.pl</code> : Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed with the <code>\Delta</code> and the <code>\lambda</code> commands.	18	0.8.0	General: The <code>anonymous</code> package option added.	11
	Relative positioning of vertices fixed.	18		<code>-phi.pl</code> : Inside <code>phiquation</code> any text inside the <code>\text</code> macro is not processed.	12
0.5.0	<code>-phi.pl</code> : Automated formatting of <code>TRUE</code> and <code>FALSE</code> added.	12		<code>\phiOset</code> : New commands <code>\phiOset</code> and <code>\phiUset</code> help position text over and under an arrow.	28
	<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow. . .	28		<code>\phiSaveTo</code> : The output of the <code>phiquation</code> environment can be redirected to a file.	16
	<code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket.	28		<code>-sodg.pl</code> : The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.	18
	<code>-sodg.pl</code> : It is possible to use TikZ commands inside the <code>sodg</code> environment.	18		<code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file.	26
	New syntax introduced that allows to make clones of vertices and all their dependants.	18	0.9.0	<code>\eoAnon</code> : New command <code>\eoAnon</code> added.	27
	Now edges may have the <code>break</code> attribute, to make them shorter. . .	18		<code>-phi.pl</code> : Proper handling of the <code>matrix</code> environment.	12
0.6.0	General: Package option			<code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code> . . .	28

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		324, 331, 374,	
$\backslash \$$	190, 339, 353, 357,	429, 673, 758, 873	$\backslash \text{eolang@clearSodgSaveTo}$
	360, 662, 763, 794		 792, 796
$\backslash \%$	 299, 305,	$\backslash \text{box}$	 860
	344, 350, 777, 782	C	
$\backslash ($	... 212, 799, 817, 819	$\backslash \text{catcode}$	322, 329, 339,
$\backslash)$	... 229, 800, 817, 819		353, 357, 360,
$\backslash *$	 153, 184		662, 756, 763, 794
$\backslash +$	... 377, 513, 544, 550	$\backslash \text{clean}$	 337, 338, 344
$\backslash -$	 817	$\backslash \text{clipbox}$	 851, 855
$\backslash \text{-phi.pl}$	 58	$\backslash \text{closein}$	 248, 656
$\backslash \text{-sodg.pl}$	 362	$\backslash \text{color}$	 27, 682, 868
$\backslash .$	... 144, 154, 202, 377	D	
$\backslash /$	 152, 153	$\backslash \text{d}$	 209, 210, 701,
$\backslash ?$	 194		702, 712, 713,
$\backslash [$	 152, 192		723, 724, 733, 734
$\backslash \{$	 89, 118,	$\backslash \text{dashrightarrow}$	... 853
	141, 142, 143,	$\backslash \text{def}$	... 252, 283, 319,
	144, 145, 146,		338, 752, 761,
	147, 148, 149,		796, 799, 800,
	150, 157, 158,		801, 802, 803, 804
	184, 191, 200,	$\backslash \Delta$	 595
	207, 208, 209, 210	$\backslash \text{detokenize}$	 337
$\backslash \}$	 89, 118, 157,	$\backslash \text{dimen}$	 847,
	158, 184, 209, 210		848, 850, 852, 859
$\backslash]$	 152, 193	$\backslash \text{Downarrow}$	 877
$\backslash ^$	 191	$\backslash \text{dp}$	 859
$\backslash $	 153, 188, 189,	$\backslash \text{draw}$	 446,
	322, 329, 382, 756		676, 706, 728, 874
Numbers		E	
$\backslash 2$	 154, 163, 401	$\backslash \text{E}$	 184, 525, 533,
$\backslash 3$	 154		541, 544, 550,
$\backslash 4$	 154		553, 554, 555, 556
A		$\backslash \text{end}$	233, 235, 238, 241,
$\backslash \text{a}$	 701, 704,		326, 333, 642,
	712, 723, 726, 733		649, 677, 760, 876
$\backslash \text{active}$	353, 357, 360, 794	$\backslash \text{endinput}$	. 240, 648, 881
$\backslash \text{aftergroup}$	... 315, 792	$\backslash \text{eoAnon}$	797, 815, 817, 819
$\backslash \alpha$	 801	$\backslash \text{eolang}$	 814
$\backslash \text{AtBeginDocument}$	.. 360	$\backslash \text{eolang@anonymous}$	17, 808
B		$\backslash \text{eolang@arrow}$	
$\backslash \text{Bbbk}$	 3		 864, 871,
$\backslash \text{begin}$	 70,	$\backslash \text{eolang@clearPhiSaveTo}$	872, 877, 878, 879
	91, 214, 217, 219,		 315, 319
		$\backslash \text{eolang@clearSodgSaveTo}$	 792, 796
		$\backslash \text{eolang@ifabsent}$	258, 340
		$\backslash \text{eolang@ifabsentwith}$	. 260, 262, 285, 764
		$\backslash \text{eolang@ink}$	22, 809, 838
		$\backslash \text{eolang@lineno}$	 53
		$\backslash \text{eolang@mdfive}$	
			 54, 283, 761
		$\backslash \text{eolang@nocolor}$	...
			 16, 24, 865
		$\backslash \text{eolang@nocomments}$	.
			 14, 299,
			305, 350, 777, 782
		$\backslash \text{eolang@nodollar}$	..
			. 15, 339, 353, 355
		$\backslash \text{eolang@noshell}$	...
			6, 18, 31, 37, 59,
			269, 310, 363, 787
		$\backslash \text{eolang@phiSaveTo}$	..
			. 252, 253, 287,
			294, 308, 313, 319
		$\backslash \text{eolang@process}$	...
			 282, 326, 333
		$\backslash \text{eolang@sodgSaveTo}$	.
			. 752, 753, 766,
			773, 785, 790, 796
		$\backslash \text{eolang@sub}$	 870,
			871, 877, 878, 879
		$\backslash \text{eolang@tmp}$	 255,
			284, 286, 288,
			291, 292, 295,
			298, 301, 304,
			312, 313, 325,
			332, 341, 343,
			345, 346, 347,
			349, 759, 762,
			765, 767, 770,
			771, 774, 776,
			779, 781, 789, 790
		$\backslash \text{eolang@tmpdir}$	12, 33,
			34, 38, 43, 45, 61,
			64, 67, 70, 243,
			246, 284, 286,
			288, 291, 292,
			295, 296, 298,
			301, 302, 304,
			312, 313, 325,

332, 341, 343, 345, 346, 347, 348, 349, 365, 368, 371, 374, 651, 654, 759, 762, 765, 767, 770, 771, 774, 775, 776, 779, 780, 781, 789, 790	\ex@ 827, 831		
F			
\F 697, 699, 700, 712, 719, 721, 722, 733			
\FancyVerbLine <u>659</u>			
G			
\gdef 358			
H			
\hash .. 283, 286, 288, 292, 295, 298, 301, 304, 312, 313, 338, 341, 346, 347, 349, 761, 765, 767, 771, 774, 776, 779, 781, 789, 790			
\hbox 828, 832, 873			
\height ... 852, 855, 856			
\hss 854			
\ht 859			
I			
\I 697, 698, 700, 712, 719, 720, 722, 733			
\iexec 43, 291, 295, 301, 313, 343, 345, 347, 770, 774, 779, 790			
\ifdefined 6, 24, 31, 37, 59, 269, 299, 305, 310, 339, 350, 353, 355, 363, 777, 782, 787, 808, 865			
\ifeof 65, 369			
\IfFileExists 32, 263, 311, 788			
\ifluatex 13, 256			
\ifnum 42, 272			
\IfStrEq 870			
\ifx 287, 294, 308, 766, 773, 785			
\ifxetex 13, 256			
		\input 260, 288, 767	293, 351, 364,
		\inputlineno 284, 293, 338, 351, 762, 772	370, 650, 653, 772
		\mspace 821	
		J	N
		\jobname 33, 34, 38, 43, 45, 61, 64, 67, 70, 243, 246, 284, 286, 288, 291, 292, 295, 296, 298, 301, 302, 304, 312, 313, 325, 332, 341, 343, 345, 346, 347, 348, 349, 365, 368, 371, 374, 651, 654, 759, 762, 765, 767, 770, 771, 774, 775, 776, 779, 780, 781, 789, 790	\newcommand 25, 27, 56, 252, 256, 259, 262, 282, 336, 752, 814, 816, 818, 820, 822, 825, 829, 834, 836, 838, 843, 845, 866, 868, 870, 871, 872, 877, 878, 879
			\newcounter 53
			\newenvironment 321, 328, 672, 755
			\NewExpandableDocumentCommand 807
			\node 485, 496, 499, 502, 509, 572
			\nodollar 355
			\noindent 673
		K	O
		\kern 827, 831	\openin 64, 368
		L	P
		\lambda 607	\pdf@filemdfivesum . 56
		\leadsto 821, 871	\pdf@mdfivesum 338
		\limits 826, 830	\pdfstringdefDisableCommands 798
		M	\pgfkeys 10
		\m@th . 847, 848, 853, 856	\Phi 392
		\makeatletter 23, 30, 53, 55, 58, 251, 255, 258, 282, 336, 362, 751, 755, 806, 824, 837, 842, 864	\phic 816
		\makeatother ... 29, 52, 53, 57, 250, 254, 257, 281, 334, 354, 658, 754, 796, 813, 833, 839, 863, 880	\phiContextualize .. 879
		\makebox 850	\picture 672
		\mapstochar ... 821, 844	\phiDataize 877
		\mathop 826, 830	\phiDotted . 595, 607, 840
		\mathpalette 844	\phiDotted@ ... 844, 845
		\mathrel .. 821, 826, 830, 844, 866, 868	\phiEOL 836
		\message . 34, 38, 45, 60, 66, 242, 245, 266, 270, 273,	\phiEvaluate 872
			\phiMany 834
			\phiMorph 878
			\phiNormalize 871
			\phiOset 824, 835
			\phiiq 335, 358
			\phiiquation 282
			\phiSaveTo 251
			\phiSlot 822
			\phiTerminal 821, 837, 844
			\phiUset 829, 835
			\phiWave 820

<code>\pi</code>	450	<code>\settowidth</code>	847		U
<code>\ProcessPgfpPackageOptions</code>		<code>\sffamily</code>	815, 823	<code>\upalpha</code>	803, 817
	21	<code>\ShellEscapeStatus</code> .		<code>\upvarphi</code>	804, 817
<code>\protected</code>	358		42, 272	<code>\usetikzlibrary</code> . . .	
		<code>\small</code>	682, 685		663, 664,
Q		<code>\sodg</code>	755		665, 666, 667,
<code>\Q</code>	184, 525, 533,	<code>\sodgSaveTo</code>	751		668, 669, 670, 671
	541, 544, 550,	<code>\StrSubstitute</code>	337		
	553, 554, 555, 556	<code>\sxy</code>	555	V	
				<code>\v</code>	697, 700, 719, 722
R		T		<code>\value</code>	317,
<code>\relax</code>	3, 253, 287,	<code>\t</code>	78, 407, 701,		323, 330, 757, 795
	294, 308, 319,		703, 712, 713,	<code>\varphi</code>	802
	662, 753, 766,		723, 725, 733, 734	<code>\vbox</code>	827, 831
	773, 785, 796, 844	<code>\text</code>	595, 823	<code>\vcenter</code>	873
<code>\RequirePackage</code>	1,	<code>\textnormal</code>	596	<code>\VerbatimEnvironment</code>	
	2, 3, 4, 5, 6, 7, 8,	<code>\texttt</code>	596		322, 329, 756
	9, 22, 31, 54, 335,	<code>\the</code>	284, 293,	<code>\vss</code>	828, 832
	660, 797, 840, 841		338, 351, 762, 772	<code>\vx</code>	703, 704, 725, 726
<code>\resizebox</code>	852	<code>\tikz</code>	660	<code>\vy</code>	703, 725
<code>\Rightarrow</code>	879	<code>\tikzinputsegmentfirst</code>			
<code>\rightarrow</code> 847, 848, 856			698, 720	W	
<code>\Rrightarrow</code>	878	<code>\tikzinputsegmentlast</code>		<code>\width</code>	851, 855
			699, 721		
S		<code>\tikzmath</code>	696, 718	X	
<code>\sbox</code>	849	<code>\tikzset</code>	662, 690	<code>\xmir</code>	818
<code>\scriptscriptstyle</code> .		<code>\tikzstyle</code>		<code>\xrightarrow</code>	823
	828, 832		678, 681, 684,		
<code>\scriptsize</code>	747, 750		686, 687, 689,	Z	
<code>\scshape</code>	823		743, 744, 745, 748	<code>\z@</code>	847, 849,
<code>\setcounter</code> 317, 323,		<code>\ttfamily</code>	682		850, 852, 859, 860
	330, 659, 757, 795	<code>\tw@</code>	848, 859		
<code>\settoheight</code>	848				