



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

Yegor Bugayenko
yegor256@gmail.com

2026-09-30, 0.26.1

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

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

1 Introduction

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

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

<pre> app $\mapsto$ [$\rho \mapsto \xi, b.^2, 0/t \rightsquigarrow \text{TRUE},$ $b \mapsto [* \mapsto \Phi.\text{fn}(56),$ $\varphi \mapsto \Phi.\text{string.trim}(\xi),$ $\Delta \mapsto 01\text{-FE-C3}]$, $x \mapsto [\lambda \mapsto \emptyset]$. </pre>	<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`),
- “]” maps to “ $]$ ” (`\rrbracket`),
- “==” maps to “ $\equiv$ ” (`\equiv`),
- “:” maps to “ $:$ ” (`\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.

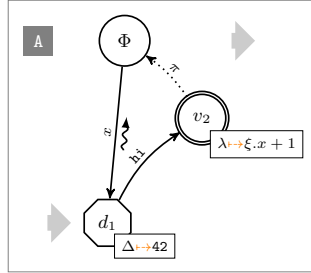
Texts within double quotes are likewise rendered in fixed-width font.

`\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>`” 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 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 \xmir{} --- \\
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 -calculus, each painted red rather than the orange of `\phiTerminal`, so that a step of computation is never mistaken for a terminal symbol. They are meant to stand between two expressions, exactly as `\phiWave` and `\phiDotted` do above. All save `\phiContextualize` 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 employed, respectively:

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

$\llbracket \mapsto^* \rrbracket$.

```
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 **phi**uation 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{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: $x \mapsto \emptyset$

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

```

5 \phiSaveTo{a}
6 \begin{phiuation*}
7 [[ x -> [[D>42]] ]]
8 \end{phiuation*}
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 **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 \} \backslash 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}`:

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

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

$\text{foo} \mapsto \left\{ \begin{array}{c} \varnothing \\ \llbracket \lambda \mapsto \rho \times \xi. \alpha_0 \rrbracket \\ \llbracket \Delta \mapsto 42 \rrbracket \end{array} \right\}$	<pre> 5 \begin{phiquation*} 6 foo -> \left\{\begin{matrix} \backslash 7 ? \backslash 8 [[L> ^ \times \$. \alpha_0]] \backslash 9 [[D> 42]] \backslash 10 \end{matrix}\right\} 11 \end{phiquation*} </pre>
--	--

The `cases` environment is likewise supported:

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

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

$\begin{array}{l} x \mapsto \llbracket \\ y \mapsto \llbracket \\ z \mapsto \xi, f \mapsto \varnothing \rrbracket, \\ \beta_1 \models [\psi \xrightarrow{\text{WAIT}} \varnothing]. \end{array}$	<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
$\llbracket \alpha_0 \mapsto \emptyset \rrbracket$	Abstraction
$x(\Delta \mapsto 42)$	Application

```

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

```

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  $\backslash( Q \times a^2 \backslash)$ .

```

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

```

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

$\llbracket b \mapsto \emptyset \rrbracket$,
 $\llbracket \phi \mapsto \text{TRUE}, \Delta \mapsto 42 \rrbracket$,
 $\psi = \langle \pi, 42 \rangle$.

```

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

```

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

$x(\pi) \mapsto 4$
 $x(a, b, c) \mapsto \llbracket \alpha_0 \mapsto \emptyset \rrbracket$

```

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

```

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

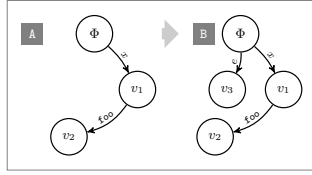
$x \mapsto \text{sprintf}(\text{"Hello, \%s!"}, \text{name})$

```

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

```

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

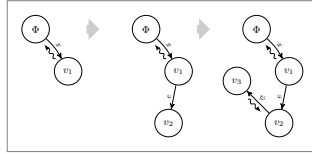


```

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:

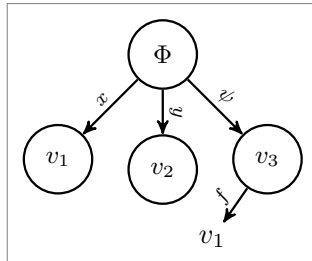


```

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:

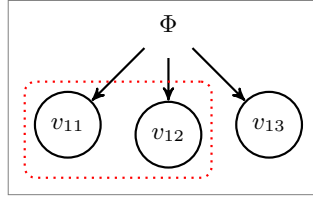


```

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:

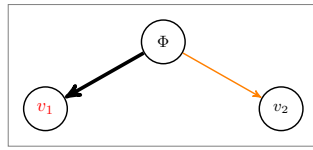


```

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 [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 0.26.1\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)\\{(.+)/trep($1, "$2")/e;
143 $tex =~ s/(?<=[\\s,>()](0-9A-F){2}(?:-[0-9A-F]{2})+)(?!\\)/|1|/g;
144 $tex =~ s/(?<=[\\s,>()](0-9){2}(?:\\.[0-9]{1,})+)(?!\\)/|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 $tex =~ s/([_~]|^)([0-9]+|\\*)\\/(\\{[a-z]+|\\{[a-z]+\\})/g;
154 (->|\\.\\.\\.>|~>|:=)/1\\alpha_{2}\\vert\\}3\\space{}4/xg;
155 if ($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\\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 $tex =~ s/\\$/\\phiTerminal{\\xi{}}/g;
188 $tex =~ s/(?!\\{)\\^(?!\\{)/\\phiTerminal{\\rho{}}/g;
189 $tex =~ s/[\\[/\\phiTerminal{\\llbracket}/g;
190 $tex =~ s/\\]/\\phiTerminal{\\rrbracket}/g;
191 $tex =~ s/\\?/\\phiTerminal{\\varnothing}/g;
192 $tex =~ s/@/\\phiTerminal{\\varphi}/g;
193 $tex =~ s/-([a-z]+)/\\mathrel{\\phiSlot{\\1}}/g;
194 $tex =~ s/->/\\mathbin{\\phiTerminal{\\mapsto}}/g;
195 $tex =~ s/~>/\\mathbin{\\phiWave}/g;
196 $tex =~ s/:=/\\mathrel{\\vDash}/g;
197 $tex =~ s/(?![\\{\\}]):/\\mathrel{\\phiTerminal{:}}/g;
198 $tex =~ s/= /\\mathrel{\\equiv}/g;
199 $tex =~ s/\\.\\.\\. /\\mathbin{\\phiTerminal{\\phiDotted}}/g;
200 $tex =~ s/-([0-9]+)/\\phiTerminal{\\alpha_{\\1}}/g;
201 $tex =~ s/\\(\\rho|\\xi|\\lambda|\\alpha|\\varphi)(?![a-zA-Z])/\\up\\1/g;
202 $tex =~ s/<</\\langle/g;
203 $tex =~ s/>>/\\rangle/g;
204 $tex =~ s/(?![\\{\\}])\\(\\)/\\phiTerminal{\\1}/g;
205 $tex =~ s/(?![\\{\\}]),/\\phiTerminal{,}/g;
206 $tex =~ s/\\|{2,}/|/g;
207 $tex =~ s/\\|([~\\|]+)\\|/'\\textnormal{\\texttt{}}' . ($1 =~ s/(?=-)}{-}\\}gr) . '}}{}/ge;
208 $tex =~ s/\\{TEXT\\(d+\\)/$texts[$1]/g;
209 if ($macro eq 'phiq') {
210     print '\\(' if ($tex ne '');
211 } else {
212     print '\\begin{' . $macro, "\\n";
213     if (not($align)) {
214         if ($gathered) {
215             print '\\begin{gathered}' . "\\n";
216         } elsif (not $splitting) {
217             print '\\begin{split}' . "\\n";
218         }
219     }

```

```

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

```

`\phiSaveTo` Then, we define the `\phiSaveTo` command to instruct the `phi`uation environment that the output should not be sent to the document but saved to the file instead:

```

249 \makeatletter
250 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
251 \makeatother

```

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

```

252 \makeatletter
253 \newcommand\eolang@tmp[1]{#1\ifxetex-xe\else\ifluatex-lua\fi\fi.tex}
254 \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:

```

255 \makeatletter
256 \newcommand\eolang@ifabsent[2]{%
257   \IfFileExists
258     {#1}
259     {%
260       \message{eolang: File "#1" already exists ^^J}%
261       \input{#1}}
262   {%
263     \ifdefined\eolang@noshell%
264       \message{eolang: Shell processing is disabled^^J}%
265     \else%

```

```

266         \ifnum\ShellEscapeStatus=1\else%
267         \message{eolang: The -shell-escape command line
268         option is not provided, most probably compilation
269         will fail now:^^J}%
270     \fi%
271     #2%
272     \fi%
273 }%
274 }
275 \makeatother

```

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

```

276 \makeatletter\newcommand\eolang@process[1]{
277   \def\hash{\eolang@mdfive
278     {\eolang@tmpdir/\jobname/\eolang@tmp{phiquation}}-\the\inputlineno}%
279   \eolang@ifabsent
280     {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation-post}}
281     {%
282       \iexec[null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{phiquation}"
283         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation}"}%
284       \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
285       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation-post}]{
286         perl "\eolang@tmpdir/\jobname-phi.pl"
287         '#1'
288         "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiquation}"
289         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\\n|\\$)//g'\fi
290         \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
291     }%
292   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
293   \def\eolang@phiSaveTo{\relax}%
294 }
295 %
296 \newenvironment{phiquation*}%
297 {\catcode'\|=12 \VerbatimEnvironment%
298 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
299 \begin{VerbatimOut}
300   {\eolang@tmpdir/\jobname/\eolang@tmp{phiquation}}}
301 {\end{VerbatimOut}\eolang@process{equation*}}
302 %
303 \newenvironment{phiquation}%
304 {\catcode'\|=12 \VerbatimEnvironment%
305 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
306 \begin{VerbatimOut}
307   {\eolang@tmpdir/\jobname/\eolang@tmp{phiquation}}}
308 {\end{VerbatimOut}\eolang@process{equation}}
309 \makeatother

```

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

```

310 \RequirePackage{xstring}
311 \makeatletter\newcommand\phiq[1]{%
312   \StrSubstitute{\detokenize{#1}}{'}{'"}{\clean}%
313   \def\hash{\pdf@mdfivesum{\clean}-\the\inputlineno}%
314   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%

```

```

315 \eolang@ifabsent
316   {\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq-post}}
317   {%
318     \iexec[log,trace,quiet,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{phiq}]{
319       printf '\%s' '\clean'}%
320     \iexec[quiet,null]{cp "\eolang@tmpdir/\jobname/\eolang@tmp{phiq}"
321       "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq}"}%
322     \iexec[trace,stdout=\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq-post}]{
323       perl \eolang@tmpdir/\jobname-phi.pl 'phiq'
324       "\eolang@tmpdir/\jobname/\eolang@tmp{\hash-phiq}"
325       \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\\n|$)//g' \fi}%
326     \message{\eolang: Parsed 'phiq' at line no. \the\inputlineno^~J}%
327   }%
328   \ifdefined\eolang@nodollar\else\catcode'\$=\active\fi%
329 } \makeatother

```

nodollar Then, we redefine dollar sign:

```

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

```

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

```

337 \makeatletter
338 \ifdefined\eolang@noshell
339 \message{\eolang: Perl script is not going to be created
340   at "\eolang@tmpdir/\jobname-sodg.pl", because of the
341   "noshell" package option^~J}
342 \else
343 \openin 15=\eolang@tmpdir/\jobname-sodg.pl
344 \ifeof 15
345 \message{\eolang: Perl script is going to be created,
346   because it is absent at "\eolang@tmpdir/\jobname-sodg.pl",
347   but if -shell-escape is not set, the compilation will
348   most likely fail now^~J}
349 \begin{VerbatimOut}{\eolang@tmpdir/\jobname-sodg.pl}
350 sub num {
351   my ($i) = @_;
352   $i =~ s/(\+|-)\./\10./g;
353   return $i;
354 }
355 sub fmt {
356   my ($tex) = @_;
357   $tex =~ s/\\|([^\|]+)\\|/'\textnormal{\texttt{' . ($1 =~ s/{-(?=-)}{-}}gr) . '}}'/ge;
358   return $tex;
359 }
360 sub toem {
361   my ($cm) = @_;
362   return $cm * 2.8;
363 }

```

```

364 sub vertex {
365   my ($v) = @_ ;
366   if (index($v, 'v0') == 0) {
367     return '\Phi';
368   } else {
369     $v =~ s/^v/v_/g;
370     $v =~ s/[^0-9]$//g;
371     return $v . '>';
372   }
373 }
374 sub tailor {
375   my ($t, $m) = @_ ;
376   $t =~ s/<([A-Z]?${m}[A-Z]?):([>]+)>/\2/g;
377   $t =~ s/<[A-Z]+:[>]+>/\2/g;
378   return $t;
379 }
380 open(my $fh, '<', $ARGV[0]);
381 my $tex; { local $/; $tex = <$fh>; }
382 if (index($tex, "\t") >= 0) {
383   print "TABS are prohibited!";
384   exit 1;
385 }
386 print '% This file is auto-generated', "\n\n";
387 print '% --- there are ', length($tex),
388   ' chars in the input (' , $ARGV[0] , "):\n";
389 foreach my $t (split(/\n/g, $tex)) {
390   print '% ', $t, "\n";
391 }
392 print "% ---\n";
393 $tex =~ s/\\\\/\n/g;
394 $tex =~ s/\\n//g;
395 $tex =~ s/(\\[a-zA-Z]+)\s+/\1/g;
396 $tex =~ s/\n{2,}/\n/g;
397 my @cmds = split(/\n/g, $tex);
398 print '% --- before processing:' . "\n";
399 foreach my $t (split(/\n/g, $tex)) {
400   print '% ', $t, "\n";
401 }
402 print '% ---';
403 print ' (' . (0+@cmds) . " lines)\n";
404 print '\begin{picture}', "\n";
405 for (my $c = 0; $c < 0+@cmds; $c++) {
406   my $cmd = $cmds[$c];
407   $cmd =~ s/^s+//g;
408   $cmd =~ s/(?!\n)%.*//g;
409   my ($head, $tail) = split(/ /, $cmd, 2);
410   my %opts = ();
411   my ($body, $style) = split(/style:/, $tail, 2);
412   $opts{'style'} = $style;
413   $tail = $body;
414   foreach my $p (split(/ /, $tail)) {
415     my ($q, $t) = split(/:/, $p);
416     $opts{$q} = $t;
417   }

```

```

418 if (index($head, '\\') == 0) {
419   print $cmd;
420 } elsif (index($head, '->') >= 0) {
421   my $draw = '\draw[';
422   if (exists $opts{'pi'}) {
423     $draw = $draw . '<MB:phi-pi><F:draw=none>';
424     if (not exists $opts{'a'}) {
425       $opts{'a'} = '\pi';
426     }
427   }
428   if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
429     $draw = $draw . '<MB:,phi-rho>';
430   }
431   $draw = $draw . ',' . $opts{'style'} . ']';
432   my ($from, $to) = split (/>/, $head);
433   $draw = $draw . " ($from) ";
434   if (exists $opts{'bend'}) {
435     $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
436       num($opts{'bend'}) . '>';
437     if (exists $opts{'rho'}) {
438       $draw = $draw . '<MB:,phi-rho>';
439     }
440     $draw = $draw . ']';
441   } else {
442     $draw = $draw . '--';
443   }
444   if (exists $opts{'a'}) {
445     my $a = $opts{'a'};
446     if (index($a, '$') == -1) {
447       $a = '$' . fmt($a) . '$';
448     } else {
449       $a = fmt($a);
450     }
451     $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
452   }
453   if (exists $opts{'break'}) {
454     $draw = $draw . '<F: coordinate [pos=' .
455       ($opts{'break'} / 100) . '] (break)>';
456   }
457   $draw = $draw . " (<MF:${to}><B:break-v>";
458   if (exists $opts{'break'}) {
459     print tailor($draw, 'F') . ";\n";
460     print ' \node[outer sep=' . toem(0.1) . 'em,inner sep=0em] ' .
461       'at (break) (break-v) {' . vertex($to) .
462       '$};' . "\n";
463     print ' ' . tailor($draw, 'B');
464   } else {
465     print tailor($draw, 'M');
466   }
467 } elsif (index($head, '=>') >= 0) {
468   my ($from, $to) = split (/=>/, $head);
469   my $size = () = $head =~ /=/g;
470   if ($from eq '') {
471     print '\node [phi-arrow, left=' . toem($size * 0.6) . 'em of ' .

```

```

472     $to . '.center]';
473 } elsif ($to eq '') {
474     print '\node [phi-arrow, right=' . toem($size * 0.6) . 'em of ' .
475         $from . '.center]';
476 } else {
477     print '\node [phi-arrow] at ($( ' .
478         $from . ')!0.5!( ' . $to . ')$)';
479 }
480 print '{}';
481 } elsif (index($head, '!') >= 0) {
482     my ($v, $marker) = split (/!+/, $head);
483     my $size = () = $head =~ /!/g;
484     print '\node [phi-marker, left=' .
485         toem($size * 0.6) . 'em of ' .
486         $v . '.center]{' . fmt($marker) . '}';
487 } elsif (index($head, '+') >= 0) {
488     my ($v, $suffix) = split (/+/, $head);
489     my @friends = ($v);
490     foreach my $c (@cmds) {
491         $e = $c;
492         $e =~ s/^\s+//g;
493         my $h = $e;
494         $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
495         foreach my $f (@friends) {
496             my $add = '';
497             if (index($h, $f . '->') >= 0) {
498                 $add = substr($h, index($h, '->') + 2);
499             }
500             if ($h =~ /\->\Q${f}\E$/) {
501                 $add = substr($h, 0, index($h, '->'));
502             }
503             if (index($e, ' xy:' . $f . ',') >= 0) {
504                 $add = $h;
505             }
506             if (index($add, '+') == -1
507                 and $add ne ''
508                 and not(grep(/^\Q${add}\E$/, @friends))) {
509                 push(@friends, $add);
510             }
511         }
512     }
513     my @extra = ();
514     foreach my $e (@cmds) {
515         $m = $e;
516         if ($m =~ /\s*\Q${v}\E$/) {
517             next;
518         }
519         if ($m =~ /\s*[\s]+\+\/ and not($m =~ /\s*\Q${head}\E$/)) {
520             next;
521         }
522         foreach my $f (@friends) {
523             my $h = $f;
524             $h =~ s/[a-z]$/g;
525             if ($m =~ s/^(s*)\Q${f}\E\+\Q${suffix}\E?s?\/1${h}${suffix} /g) {

```

```

526         last;
527     }
528     $m =~ s/^(\\s*)\\Q${f}\\E\\s/\\1${h}${suffix} /g;
529     $m =~ s/^(\\s*)\\Q${f}\\E->/\\1${h}${suffix}->/g;
530     $m =~ s/\\sxy:\\Q${f}\\E,/ xy:${h}${suffix},/g;
531     $m =~ s/->\\Q${f}\\E\\s/->${h}${suffix} /g;
532 }
533 if ($m ne $e) {
534     push(@extra, ' ' . $m);
535 }
536 }
537 splice(@extra, 0, 0, $extra[-1]);
538 splice(@extra, -1, 1);
539 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
540     '), friends: [' . join(', ', @friends) . ']' in ' .
541     (0+@cmds) . ' lines');
542 splice(@cmds, $c, 1, @extra);
543 print '% cloned ' . $v . ' at line no.' . $c .
544     ' (' . (0+@extra) . ' lines -> ' .
545     (0+@cmds) . ' lines total)';
546 } elsif ($head =~ /^v[0-9]+[a-z]?$/) {
547     print '\\node[';
548     if (exists $opts{'xy'}) {
549         my ($v, $right, $down) = split(/,/ , $opts{'xy'});
550         my $loc = '';
551         if ($down > 0) {
552             $loc = 'below ';
553         } elsif ($down < 0) {
554             $loc = 'above ';
555         }
556         if ($right > 0) {
557             $loc = $loc . 'right';
558         } elsif ($right < 0) {
559             $loc = $loc . 'left';
560         }
561         print ', ' . $loc . '=';
562         print toem(abs(num($down))) . 'em and ' .
563             toem(abs(num($right))) . 'em of ' . $v . '.center';
564     }
565     if (exists $opts{'data'}) {
566         print ',phi-data';
567         if ($opts{'data'} ne '') {
568             my $d = $opts{'data'};
569             if (index($d, '|') == -1) {
570                 $d = '\\Delta\\phiDotted\\text{' .
571                     '\\textnormal{\\texttt{' . fmt($d) . '}}}'$';
572             } else {
573                 $d = fmt($d);
574             }
575             $opts{'box'} = $d;
576         }
577     } elsif (exists $opts{'atom'}) {
578         print ',phi-atom';
579         if ($opts{'atom'} ne '') {

```

```

580     my $a = $opts{'atom'};
581     if (index($a, '$') == -1) {
582         $a = '$\lambda\phiDotted{' . fmt($a) . '$';
583     } else {
584         $a = fmt($a);
585     }
586     $opts{'box'} = $a;
587 }
588 } else {
589     print ',phi-object';
590 }
591 if (exists $opts{'edgeless'}) {
592     print ',draw=none';
593 }
594 print ',' . $opts{'style'} . ']';
595 print ' (' . $head . ')';
596 print '{';
597 if (exists $opts{'tag'}) {
598     my $t = $opts{'tag'};
599     if (index($t, '$') == -1) {
600         $t = '$' . $t . '$';
601     } else {
602         $t = fmt($t);
603     }
604     print $t;
605 } else {
606     print '$' . vertex($head) . '$';
607 }
608 print '}';
609 if (exists $opts{'box'}) {
610     print ' node[phi-box] at (';
611     print $head, '.south east) {';
612     print $opts{'box'}, ')';
613 }
614 }
615 print ";\n";
616 }
617 print '\end{picture}%', "\n";
618 print "% --- after processing:\n";
619 foreach my $c (@cmds) {
620     print '% ', $c, "\n";
621 }
622 print '% --- (' . (0+@cmds) . " lines)\n";
623 print '\endinput';
624 \end{VerbatimOut}
625 \message{eolang: File with Perl script
626 '\eolang@tmpdir/\jobname-sodg.pl' saved^^J}
627 \else
628 \message{eolang: Perl script already exists at
629 "\eolang@tmpdir/\jobname-sodg.pl"^^J}
630 \fi
631 \closein 15
632 \fi
633 \makeatother

```

FancyVerbLine Then, we reset the counter for `fancyvrb`, so that it starts counting lines from zero when the document starts rendering:

```
634 \setcounter{FancyVerbLine}{0}
```

tikz Then, we include `tikz` package and its libraries:

```
635 \RequirePackage{tikz}
636 \usetikzlibrary{arrows}
637 \usetikzlibrary{shapes}
638 \usetikzlibrary{decorations}
639 \usetikzlibrary{decorations.pathmorphing}
640 \usetikzlibrary{decorations.pathreplacing}
641 \usetikzlibrary{positioning}
642 \usetikzlibrary{calc}
643 \usetikzlibrary{math}
644 \usetikzlibrary{arrows.meta}
```

picture Then, we define internal environment `picture`:

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

```

682         segment length=2mm,
683         post length=1mm
684     ]]
685     ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
686     -- ++(\t: 2*\d em);
687 };
688 }
689 },
690 lineto code={
691     \tikzmath{
692         coordinate \I, \F, \v;
693         \I = (\tikzinputsegmentfirst);
694         \F = (\tikzinputsegmentlast);
695         \v = ($(\I) -(\F)$);
696         real \d, \a, \r, \t;
697         \d = 0.8;
698         \t = atan2(\vy, \vx);
699         if \vx<0 then { \a = 90; } else { \a = -90; };
700         {
701             \draw[arrows={-latex}, decorate,
702                 decoration={%
703                     snake, amplitude=.4mm,
704                     segment length=2mm,
705                     post length=1mm}]
706             ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
707             -- ++(\t: 2*\d em);
708         };
709     }
710 }
711 },
712 decorate
713 }
714 }
715 }
716 \tikzstyle{phi-pi} = [draw,dotted]
717 \tikzstyle{phi-atom} = [phi-object,double]
718 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
719     rectangle,line width=.04em,minimum width=1.2em,anchor=north west,
720     font={\scriptsize}]
721 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
722     above=2pt,sloped/.append style={transform shape},
723     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:

```

724 \makeatletter
725 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
726 \makeatother

```

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

```

727 \makeatletter\newenvironment{sodg}%
728 {\catcode'\|=12 \VerbatimEnvironment%
729 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
730 \begin{VerbatimOut}

```

```

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

```

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

```

752 \RequirePackage{hyperref}
753 \pdfstringdefDisableCommands{
754   \def\{\}%
755   \def\}\}%
756   \def\alpha{\alpha}%
757   \def\varphi{\phi}%
758 }
759 \makeatletter
760 \NewExpandableDocumentCommand{\eoAnon}{O{ANONYMIZED}m}{%
761   \ifdefined\eolang@anonymous%
762     \eolang@ink{#1}%
763   \else%
764     #2%
765   \fi%
766 }\makeatother

```

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

```

767 \newcommand\eolang{%
768   \eoAnon[XYZ]{\sffamily EO}}

```

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

```

769 \newcommand\phic{%
770   \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.

```

771 \newcommand\xmirl{%
772   \eoAnon[XML\(^+\)]{XMIR}}

```

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

```
773 \newcommand\phiWave{%
774   \phiTerminal{\mapstochar\mathrel{\mspace{0.45mu}}\leadsto}}
```

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

```
775 \newcommand\phiSlot[1]{%
776   \xrightarrow{\text{\sffamily\scshape #1}}}
```

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

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

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

```
787 \newcommand\phiMany[3]{%
788   \phi0set{\#3}{\phiUset{\#2}{\#1}}}
```

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

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

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

```
790 \makeatletter
791 \newcommand\phiTerminal[1]{\eolang@ink{\#1}}
792 \makeatother
```

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

```
793 \RequirePackage{trimclip}
794 \RequirePackage{amsfonts}
795 \makeatletter
796 \newcommand{\phiDotted}{%
797   \phiTerminal{\mapstochar\mathrel{\mathpalette\phiDotted@relax}}}
798 \newcommand{\phiDotted@}[2]{%
799   \begingroup%
800   \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
801   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
802   \sbox\z@{%
803     \makebox[\dimen\z@][s]{%
804       \clipbox{0 0 {0.4\width} 0}%
805       {\resizebox{\dimen\z@}{\height}%
806         {\m@th#1\dashrightarrow}}}%
807     \hss%
808     \clipbox{{0.69\width} {-0.1\height} 0
809       {-\height}}{\m@th#1\rightarrow}}%
```

```

810  }%
811  }%
812   \ht\z@=\dimen\tw@ \dp\z@=\z@%
813   \box\z@%
814   \endgroup%
815 }
816 \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:

817 \makeatletter
818 \ifdefined\eolang@nocolor
819   \newcommand\eolang@arrow[1]{\mathrel{#1}}
820 \else
821   \newcommand\eolang@arrow[1]{\mathrel{\color{red}{#1}}}
822 \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:

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

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

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

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

825 \newcommand\phiEvaluate{\eolang@arrow{%
826   \vcenter{\hbox{\begin{tikzpicture}
827     \draw[-{Triangle[length=1.8mm,width=1.8mm]},line width=0.6pt]
828       (0,0)--(1.3em,0);
829     \end{tikzpicture}}}}}

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

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

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

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

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

832 \newcommand\phiContextualize[1][]{\eolang@arrow\Rightarrow\eolang@sub{#1}}
833 \makeatother

834 \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 sodg	18
	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.		25
	-phi.pl : New symbol added for basket slots		12
	Parsing of the symbols " @ ," " ^ ," and " & " enabled (\varphi , \rho , and \sigma)		12
	The symbols " [" and "] " replaced with " [[" and "]] " for abstract object brackets, because they conflicted with normal square brackets		12
	\phiq : Parsing of additional symbols enabled.		17
	-sodg.pl : The Perl file now has a fixed name, which doesn't depend on the name of the TeX job. This file may be shared among jobs, no need to make it uniquely named.		18
0.1.0		0.12.1	
	General: Parsing of package options introduced.		11
	\eolang : New command \eolang added to print the name of the language in both normal and the anonymous mode of acmart		26
	\eolang@mdfive : New supplementary command added to calculate MD5 sum of a file.		12
	-phi.pl : A new Perl script " eolang-phi.pl " added for parsing of phi expressions.		12
	\phic : New command \phic prints the name of φ -calculus in both normal and the anonymous mode of acmart		26
	\phiDotted : New command \phiDotted added to denote a link to a special attribute.		27
	-sodg.pl : There are two Perl scripts now: one for		
		phiuation , another one for sodg	18
		-sodg.pl : The bug is fixed related to the formatting of indexes of vertices.	18
		0.13.0	
		-phi.pl : Parsing of QQ into \dot{\Phi} implemented.	12
		0.14.0	
		-sodg.pl : The edgeless tag of a vertex removes the border of it.	18
		0.15.0	
		-sodg.pl : The style tag of vertices and edges.	18
		0.16.0	
		phiuation : The processing of phiuation data is done only if it's the first time processing, otherwise cache is used, thus making processing faster.	17
		sodg : The processing of sodg data is done only if it's the first time processing, otherwise cache is used, thus making processing faster.	25
		0.17.0	
		\eolang@ifabsent : A new supplementary eolang@ifabsent command added	16
		0.18.0	
		\eolang@ifabsent : The noshell package option added in order to enable complete prohibition of shell interactions.	16
		0.18.3	
		\eolang@tmp : A new supplementary eolang@tmp command added, to generate temporary file names.	16
		0.19.0	
		-phi.pl : Parsing of T into \bot implemented.	12
		0.2.0	
		-phi.pl : Numbers automatically render as \texttt . No need to use vertical bars around them anymore.	12
		-sodg.pl : The content of the atom and the data boxes is parsed	

automatically as formulas and numbers, respectively.	18	package option.	27
\xmir: New command \xmir prints XMIR in both normal and the anonymous mode of acmart. . .	26	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.	28
-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.	28
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.	28
\phiTerminal: New command \phiTerminal added, to highlight all terminals in phi expressions.	27	\phiDataize: New command \phiDataize added, to render the dataization arrow of φ -calculus, with an optional subscript.	28
0.22.0		\phiEvaluate: New command \phiEvaluate added, to render the evaluation arrow of φ -calculus.	28
-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. . .	28
0.23.0		\phiNormalize: New command \phiNormalize added, to render the normalization arrow of φ -calculus, with an optional subscript.	28
-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 phiquation, 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	0.3.0	
0.23.2		\eolang@lineno: New counter for	
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			

protecting lineno.	12	in all situations.	18
-phi.pl: New arrow added, that looks like \leadsto.	12	0.7.0 nodollar: Now it is possible to use dollar sign instead of the \phiq command.	18
\phiWave: New command \phiWave added to denote a link to a multi-layer attribute.	27	-phi.pl: New syntax sugar for Φ , just using capital “Q” is enough.	12
0.4.0 -sodg.pl: Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed with the \Delta and the \lambda commands.	18	Object names are automatically converted to \texttt, provided their names include two or more symbols.	12
Relative positioning of vertices fixed.	18	Text in quotes is automatically converted to \texttt.	12
0.5.0 -phi.pl: Automated formatting of TRUE and FALSE added.	12	0.8.0 General: The anonymous package option added.	11
\phiMany: New command \phiMany enables iterating over an arrow.	27	-phi.pl: Inside phiquation any text inside the \text macro is not processed.	12
\phiSlot: New command \phiSlot added to denote a link to a slot in a basket.	27	\phi0set: New commands \phi0set and \phiUset help position text over and under an arrow.	27
-sodg.pl: It is possible to use TikZ commands inside the sodg environment.	18	\phiSaveTo: The output of the phiquation environment can be redirected to a file.	16
New syntax introduced that allows to make clones of vertices and all their dependants.	18	-sodg.pl: The tag attribute is introduced for changing labels inside a vertex circle.	18
Now edges may have the break attribute, to make them shorter.	18	\sodgSaveTo: The output of the sodg environment can be redirected to a file.	25
0.6.0 General: Package option nocomments added in order to enable comments suppression in temporary .tex files (may be pretty important for .dtx documents).	11	0.9.0 \eoAnon: New command \eoAnon added.	26
-sodg.pl: The rrho attribute is retired, now rho works just fine		-phi.pl: Proper handling of the matrix environment.	12
		\phiEOL: New command \phiEOL added, instead of \[-4pt].	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	
<code>\\$</code> 187, 314, 328, 332, 335, 735, 748	<code>\catcode</code> . . 297, 304, 314, 328, 332, 335, 728, 735, 748	<code>\eolang@nocomments</code> . . 14, 289, 325, 745
<code>\%</code> . . . 289, 319, 325, 745	<code>\clean</code> 312, 313, 319	<code>\eolang@nodollar</code> . . . 15, 314, 328, 330
<code>\(</code> . . . 210, 754, 770, 772	<code>\clipbox</code> 804, 808	<code>\eolang@noshell</code> 6, 18, 31, 37, 59, 263, 338
<code>\)</code> . . . 227, 755, 770, 772	<code>\closein</code> 246, 631	<code>\eolang@phiSaveTo</code> 250, 290, 293
<code>*</code> 153, 184	<code>\color</code> 27, 655, 821	<code>\eolang@process</code> 276, 301, 308
<code>\+</code> . . . 352, 488, 519, 525		<code>\eolang@sodgSaveTo</code> 725, 746, 750
<code>\-</code> 770		<code>\eolang@sub</code> 823, 824, 830, 831, 832
<code>\-phi.pl</code> 58		<code>\eolang@tmp</code> 252, 278, 280, 282, 283, 285, 288, 300, 307, 316, 318, 320, 321, 322, 324, 731, 734, 737, 739, 740, 742, 744
<code>\-sodg.pl</code> 337		<code>\eolang@tmpdir</code> 12, 33, 34, 38, 43, 45, 61, 64, 67, 70, 241, 244, 278, 280, 282, 283, 285, 286, 288, 300, 307, 316, 318, 320, 321, 322, 323, 324, 340, 343, 346, 349, 626, 629, 731, 734, 737, 739, 740, 742, 743, 744
<code>\.</code> . . . 144, 154, 199, 352		<code>\ex@</code> 780, 784
<code>\/</code> 152, 153		
<code>\?</code> 191		
<code>\[</code> 152, 189		
<code>\{</code> . 89, 118, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 157, 158, 184, 188, 197, 204, 205, 208		
<code>\}</code> 89, 118, 157, 158, 184, 208		
<code>\]</code> 152, 190		
<code>\^</code> 188		
<code>\ </code> 153, 206, 207, 297, 304, 357, 728		
Numbers	E	
<code>\2</code> 154, 163, 376	<code>\E</code> 184, 500, 508, 516, 519, 525, 528, 529, 530, 531	
<code>\3</code> 154	<code>\end</code> 231, 233, 236, 239, 301, 308, 617, 624, 650, 732, 829	
<code>\4</code> 154	<code>\endinput</code> . 238, 623, 834	
A	<code>\eoAnon</code> 752, 768, 770, 772	
<code>\a</code> 674, 677, 685, 696, 699, 706	<code>\eolang</code> 767	
<code>\active</code> 328, 332, 335, 748	<code>\eolang@anonymous</code> 17, 761	
<code>\alpha</code> 756, 770	<code>\eolang@arrow</code> 817, 824, 825, 830, 831, 832	
<code>\AtBeginDocument</code> . . 335	<code>\eolang@ifabsent</code> . . . 255, 279, 315, 736	
B	<code>\eolang@ink</code> . 22, 762, 791	
<code>\Bbbk</code> 3	<code>\eolang@lineno</code> 53	
<code>\begin</code> 70, 91, 212, 215, 217, 299, 306, 349, 404, 646, 730, 826	<code>\eolang@mdfive</code> 54, 277, 733	
<code>\box</code> 813	<code>\eolang@nocolor</code> 16, 24, 818	
	F	
	<code>\F</code> 670, 672, 673, 685, 692, 694, 695, 706	
	<code>\FancyVerbLine</code> 634	
	G	
	<code>\gdef</code> 333	
	H	
	<code>\hash</code> 277, 280, 283, 285, 288,	

313, 316, 321, 322, 324, 733, 737, 740, 742, 744	\hbox 781, 785, 826	\makeatletter 23, 30, 53, 55, 58, 249, 252, 255, 276, 311, 337, 724, 727, 759, 777, 790, 795, 817	\pgfkeys 10
\height . . . 805, 808, 809	\hss 807	\makeatother . . . 29, 52, 53, 57, 248, 251, 254, 275, 309, 329, 633, 726, 751, 766, 786, 792, 816, 833	\Phi 367
\ht 812		\makebox 803	\phic 769
I		\mapstochar . . . 774, 797	\phiContextualize . . 832
\I 670, 671, 673, 685, 692, 693, 695, 706	\iexec 43, 282, 285, 318, 320, 322, 739, 742	\mathop 779, 783	\phiDataize 830
\ifdefined . 6, 24, 31, 37, 59, 263, 289, 290, 314, 325, 328, 330, 338, 745, 746, 761, 818	\ifEOF 65, 344	\mathpalette 797	\phiDotted . 570, 582, 793
\IfFileExists . . 32, 257	\ifluatex 13, 253	\mathrel . . 774, 779, 783, 797, 819, 821	\phiDotted@ . . . 797, 798
\ifluatex 13, 253	\ifnum 42, 266	\message . 34, 38, 45, 60, 66, 240, 243, 260, 264, 267, 284, 326, 339, 345, 625, 628, 741	\phiEOL 789
\IfStrEq 823	\ifxetex 13, 253	\message . 34, 38, 45, 60, 66, 240, 243, 260, 264, 267, 284, 326, 339, 345, 625, 628, 741	\phiEvaluate 825
\ifxetex 13, 253	\input 261	\mspace 774	\phiMany 787
\inputlineno 278, 284, 313, 326, 734, 741			\phiMorph 831
J		N	\phiNormalize 824
\jobname 33, 34, 38, 43, 45, 61, 64, 67, 70, 241, 244, 278, 280, 282, 283, 285, 286, 288, 300, 307, 316, 318, 320, 321, 322, 323, 324, 340, 343, 346, 349, 626, 629, 731, 734, 737, 739, 740, 742, 743, 744		\newcommand 25, 27, 56, 250, 253, 256, 276, 311, 725, 767, 769, 771, 773, 775, 778, 782, 787, 789, 791, 796, 798, 819, 821, 823, 824, 825, 830, 831, 832	\phiOset 777, 788
		\newcounter 53	\phiq 310, 333
		\newenvironment 296, 303, 645, 727	\phiquation 276
		\NewExpandableDocumentCommand 760	\phiSaveTo 249
		\node 460, 471, 474, 477, 484, 547	\phiSlot 775
		\nodollar 330	\phiTerminal 774, 790, 797
		\noindent 646	\phiUset 782, 788
K		O	\phiWave 773
\kern 780, 784		\openin 64, 343	\pi 425
L			\ProcessPgfPackageOptions 21
\lambda 582		P	\protected 333
\leadsto 774, 824		\pdf@filemdfivesum . 56	
\limits 779, 783		\pdf@mdfivesum . . . 313	Q
M		\pdfstringdefDisableCommand 753	\Q 184, 500, 508, 516, 519, 525, 528, 529, 530, 531
\m@th . 800, 801, 806, 809			R
			\relax . . 3, 293, 750, 797
			\RequirePackage . 1, 2, 3, 4, 5, 6, 7, 8, 9, 22, 31, 54, 310, 635, 752, 793, 794
			\resizebox 805
			\Rightarrow 832
			\rightarrow 800, 801, 809
			\rightarrow 831
			S
			\sbox 802
			\scriptscriptstyle 781, 785
			\scriptsize . . . 720, 723
			\scshape 776
			\setcounter 292, 298, 305, 634, 729, 749
			\settoheight 801
			\settowidth 800
			\sffamily 768, 776
			\shellEscapeStatus 42, 266

