



eolang: \LaTeX Package for Formulas and Graphs of EO Programming Language and φ -calculus*

Yegor Bugayenko
yegor256@gmail.com

2024-01-02, 0.17.1

NB! You must run \TeX processor with `-shell-escape` option and you must have [Perl](#) installed. If you omit the `-shell-escape` option, the package will try to use cached files, if they exist. If they don't, compilation will fail. Thus, when you must prepare your document for a compilation without the `-shell-escape` option, run it locally with the option provided and then package all files (including the files in the `_eolang-*` directories) into a single ZIP archive. It is advised to use `tmpdir` package option in this case, in order to make the directory name not depend on the \LaTeX engine.

If `-shell-escape` is set, this package won't work on Windows, because it uses POSIX command line interface.

1 Introduction

This package helps you print formulas of φ -calculus, which is a formal foundation of [EO](#) programming language. The calculus was introduced by Bugayenko (2021) and later formalized by Kudasov et al. (2022). Here is how you render a simple expression:

*The sources are in GitHub at [objectionary/eolang.sty](https://github.com/objectionary/eolang.sty)

$\begin{aligned} \text{app} &\mapsto \llbracket \\ &\quad \rho \mapsto \xi.b.^2, \alpha_0 t \rightsquigarrow \text{TRUE}, \\ &\quad b \mapsto \llbracket \alpha_* \mapsto \Phi.\text{fn}(56), \\ &\quad \quad \varphi \mapsto \dot{\Phi}.\text{string.trim}(\xi), \\ &\quad \quad \Delta \mapsto 01\text{-FE-C3} \rrbracket, \\ &\quad x \mapsto \llbracket \lambda \mapsto \emptyset \rrbracket. \end{aligned}$	<pre> 1 \documentclass{minimal} 2 \usepackage{eolang} 3 \begin{document} 4 \begin{phiquestion*} 5 app -> [[% it's abstract! 6 ^ !-> \$.b.^{~2}, 0/t~> TRUE, 7 b -> [[*-> Q.fn(56), 8 @ -> QQ.string.trim(\$), 9 D> 01-FE-C3]]],\ 10 x -> [[\lambda .> ?]]. 11 \end{phiquestion*} 12 \end{document} </pre>
--	---

`phiquestion (env.)` The environment `phiquestion` lets you write a φ -calculus expressions using simple plain-text notation, where:

- “@” maps to “ φ ” (`\varphi`),
- “^” maps to “ ρ ” (`\rho`),
- “\$” maps to “ ξ ” (`\xi`),
- “&” maps to “ σ ” (`\sigma`),
- “?” maps to “ \emptyset ” (`\varnothing`),
- “Q” maps to “ Φ ” (`\Phi`),
- “QQ” maps to “ $\dot{\Phi}$ ” (`\dot{\Phi}`),
- “->” maps to “ \mapsto ” (`\mapsto`),
- “~>” maps to “ \rightsquigarrow ” (`\rightsquigarrow`),
- “!->” maps to “ \mapsto ” (`\mapsto`),
- “.>” maps to “ \mapsto ” (`\mapsto`),
- “D>” maps to “ $\Delta \mapsto$ ” (`\Delta \mapsto`),
- “L>” maps to “ $\lambda \mapsto$ ” (`\lambda \mapsto`),
- “[[” maps to “ \llbracket ” (`\llbracket`),
- “]]” maps to “ \rrbracket ” (`\rrbracket`),
- “==” maps to “ \equiv ” (`\equiv`),
- “|abc|” maps to “ abc ” (`\text{abc}`).

Also, a few symbols are supported for φ PU architecture:

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

Before any arrow you can put a number, which will be rendered as α with an index, for example `\phiq{0->x}` will render “ $\alpha_0 \mapsto x$ ”. Instead of a number you can use asterix too.

You can append a slash and a title to the number of an attribute, such as $0/g \mapsto x$. this will render as $\alpha_0|g \mapsto x$. You can use fixed-width words too, for example `\phiq{0/|f|->x}` will render as “ $\alpha_0|f \mapsto x$ ”. It’s also possible to use an asterix instead of a number, such that `\phiq{*/g->x}` renders as “ $\alpha_*|g \mapsto x$ ”

Numbers are automatically converted to fixed-width font, no need to always decorate them with vertical bars.

TRUE and FALSE are automatically converted to fixed-width font too.

Object names are automatically converted to fixed-width font too, if they have more than one letter.

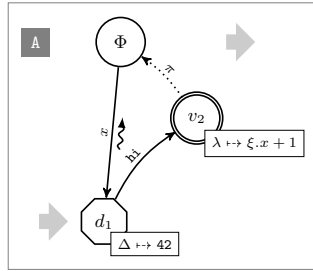
Texts in double quotes are automatically converted to fixed-width font too.

`\phiq` The command `\phiq` lets you inline a φ -calculus expressions using the same simple plain-text notation. You can use dollar sign directly too:

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 environment `sodg` allows you to draw 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 in each line are separated by a single space. The first marker is either a unique name of a vertex, like “ $v1$ ” in the example above, or an edge, like “ $v0 \rightarrow v1$.” All other markers are either unary like “ ρ ” or binary like “ $\text{atom:}\xi.x+1$.” Binary markers have two parts, separated by 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 to the edge,
- “bend:<angle>” bend 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.

It is also possible to put transformation arrows to the graph, with the help of “v0=>v1” syntax. The arrow will be placed exactly between two vertices. You can also put an arrow from a vertex to the right, saying for example “v3=>”, of from the left to the vertex, by saying for example “=>v5.” If you want the arrow to stay further away from the vertex than usually, use a few “=” symbols, for example “===>v0.”

You can also put a marker at the left side of a vertex, using “v5!A” syntax, where “v5” is the vertex and “A” is the text in the marker. They are useful when you put a few graphs on a picture explaining how one graph is transformed to another one and so forth. You can make a distance between the vertex and the marker a bit larger by using a few exclamation marks, for example “v5!!!A” will make a distance three times bigger.

You can make a clone of an existing vertex together with all its dependants, by using this syntax: “v0+a.” Here, we make a copy of “v0” and call it “v0a.” See the example below.

Be aware, unrecognized markers are simply ignored, without any error reporting.

`\eolang` There is also a no-argument command `\eolang` to help you print the name of EO
`\phic` language. It understands the anonymous package option and prints itself differently, to
`\xmirl` double-blind your paper. There is also `\phic` command to print the name of φ -calculus,
also sensitive to anonymous mode. The macro `\xmirl` 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 presentation.

```

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 presentation.
10 \end{document}

```

Without the anonymous option there will be no orange color:

In our research we use EO,
an experimental object-oriented
dataflow language, φ -calculus, as its
formal foundation, and XMIR —
its XML-based presentation.

```

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 presentation.
10 \end{document}

```

`\phiConst` A few simple commands are defined to help you render arrows. It is recommended
`\phiWave` not to use them directly, but use `!->` instead. However, if you want to use `\phiConst`,
`\phiDotted` wrap it in `\mathrel` for better display:

If x is an identifier and y is an object,
then $x \# \rightarrow y$ makes y a constant,
 $x \rightsquigarrow y$ makes it a decoratee
of an arbitrary number of objects,
while $x \vdash \rightarrow y$ makes it a special
attribute.

```

6 If $x$ is an identifier and $y$ is
7 an object, then $x \phiConst y$
8 makes $y$ a constant,
9 $x \phiWave y$ makes it a decoratee
10 of an arbitrary number of objects,
11 while $x \phiDotted y$ makes it
12 a special attribute.

```

`\phiOset` If you want to put a text over an arrow or under it, use `\phiOset` and `\phiUset`
`\phiUset` respectively:

When the names of attributes and
their values don't matter, we use
an arrow with a star, for example:

$$[[\vdash \rightarrow]]$$

```

6 When the names of attributes and their
7 values don't matter, we use an arrow
8 with a star, for example:
9 \begin{phiquestion*}
10 [[ \phiOset{*}{->} ]]
11 \end{phiquestion*}

```

`\phiMany` Sometimes you may need to simplify the way you describe an object (the typesetting
is a bit off, but this is not because of us, but because of [this](#)):

The expression $[\alpha_1 \mapsto x_1, \alpha_2 \mapsto x_2, \dots, \alpha_n \mapsto x_n]$
and expression $[\alpha_i \mapsto_{i=1}^n x_i]$ are
syntactically different but semanti-
cally equivalent.

```

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

```

`\phiSaveTo` If you want to use `phiquestion` or `sodg` environments inside `tabular` or any other
`\sodgSaveTo` environment or command, you won't be able to do this, because `phiquestion` and `sodg`
are “verbatim” environments. `\phiSaveTo` and `\sodgSaveTo` commands will help you
in this situation. You use them right before `\begin{phiquestion}` or `\begin{sodg}`
respectively — the content of the equation or the graph won't be rendered, but instead
saved to the file. Later, inside `tabular`, you can use it through the `\input` macro (don't
forget the `\parbox`):

Free:	$\llbracket x \mapsto \emptyset \rrbracket$
Bound:	$\llbracket x \mapsto \llbracket \Delta \mapsto 42 \rrbracket \rrbracket$

```

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` You may want to hide some of the content with the help of the anonymous package option. The command `\eoAnon` may help you with this. It has two parameters: one mandatory and one optional. The mandatory one is the content you want to show and the optional one is the substitution we will render if the anonymous package option is set.

2 Package Options

`tmpdir` The default location of temp files is `_eolang`. You can change this with the help of the `tmpdir` package option:

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

`nodollar` You may disable the special treatment of the dollar sign by using the `nodollar` package option:

```
\usepackage[nodollar]{eolang}
```

`anonymous` You may anonymize `\eolang`, `\XMIR`, and `\phic` commands by using anonymous package option (they all use the `\eoAnon` command mentioned earlier):

```
\usepackage[anonymous]{eolang}
```

3 More Examples

The `phiuation` environment treats ends of line as signals to start new lines in the formula. If you don't want this to happen and want to parse the next line as the a continuation of the current line, you can use a single backslash as it's done here:

$$\frac{x \mapsto \llbracket \varphi \mapsto y \rrbracket \quad y \mapsto \llbracket z \mapsto 42 \rrbracket}{x.z \mapsto 42} R1$$

```

6 \begin{phiuation*}
7 \dfrac \{
8 {x->[[@->y]] \quad y->[[z->42]]} \{
9 {x.z -> 42} \}
10 \text{\sffamily R1}
11 \end{phiuation*}

```

This is how you can use `\dfrac` from [amsmath](#) for large inference rules, with the help of `\begin{split}` and `\end{split}`:

$$\frac{x \mapsto [\varphi \mapsto y, z \mapsto 42, \alpha_0 | g \mapsto \emptyset, \alpha_1 | \text{foo} \mapsto 42]]}{x \mapsto [\varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto [\psi \mapsto \text{hello}(12)], \alpha_1 \mapsto 42)]} \text{R2.}$$

```

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{R2}.
15 \end{phiuation*}

```

You can use the `matrix` environment too, in order to group a few lines:

$$\text{foo} \mapsto \left\{ \begin{array}{c} \emptyset \\ [\lambda \mapsto \rho \times \xi. \alpha_0] \\ [\Delta \mapsto 42] \end{array} \right\}$$

```

5 \begin{phiuation*}
6 foo -> \left\{\begin{matrix} \backslash
7 ? \backslash
8 [[ L> ~ \times $. \alpha_0 ]] \backslash
9 [[ D> 42 ]] \backslash
10 \end{matrix}\right\}
11 \end{phiuation*}

```

The `cases` environment works too:

$$\beta \models \begin{cases} [v_2, \varphi \xrightarrow{\text{DTZD}} 42] \\ [v_{33}] \end{cases}$$

```

5 \begin{phiuation*}
6 \beta := \begin{cases}
7 [ v_2, @ -dtzd> 42 ] \backslash
8 [ v_{33} ] \backslash
9 \end{cases}
10 \end{phiuation*}
11 \end{document}

```

The `phiuation` environment may be used together with the [acmart](#) package:

$$\begin{array}{l} x \mapsto [\\ \quad y \mapsto [\\ \quad \quad z \mapsto \xi, f \mapsto \emptyset]], \\ \beta_1 \models [\psi \xrightarrow{\text{WAIT}} \emptyset]. \end{array}$$

```

1 \documentclass{acmart}
2 \usepackage{eolang}
3 \thispagestyle{empty}
4 \begin{document}
5 \begin{phiuation*}
6 x -> [[
7   y -> [[
8     z !-> $, f ..> ? ]]]],\backslash
9 \beta_1 := [ \psi -wait> ? ].
10 \end{phiuation*}
11 \end{document}

```

It's possible to use `\label` inside the `phiuation` environment (pay attention to how you can disable our custom parsing of math formulas by means of curled brackets around the “4” number):

Discriminant can be calculated using the following simple formula:

$$D = b^2 - 4ac. \quad (1)$$

Eq. 1 is also widely used in number theory and polynomial factoring.

```
6 Discriminant can be calculated using
7 the following simple formula:
8 \begin{phiuation}
9 D = b{^2} - {4}ac.
10 \label{d}
11 \end{phiuation}
12 Eq.\ref{d} is also widely used in
13 number theory and polynomial factoring.
```

You can add comments to your equations, using the `&&` command (pay attention, the text inside `\text{}` is not processed and treated like a 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*}
```

If you don't use `nodollar` package option, you can still use normal parsing of the dollar sign, by means of `\(...\)` syntax:

The object formation $\llbracket \alpha_0 \mapsto x \rrbracket$ may be replaced with a formula $Q \times a^2$.

```
6 The object formation $[[0->x]]$
7 may be replaced with a formula
8 \(( Q \times a^2 \)).
```

The `phiuation` environment will automatically align formulas by the first arrow, if there are only left-aligned formulas:

$x(\pi) \mapsto \llbracket \lambda \mapsto f_1 \rrbracket,$
$x(a, b, c) \mapsto \llbracket \alpha_0 \mapsto \emptyset, \varphi \mapsto \text{hello}(\xi), x \mapsto \text{FALSE} \rrbracket,$
$\Delta = 43-09,$
$x(y) \equiv x(\alpha_0 \mapsto y).$

```
5 \begin{phiuation*}
6 x(\pi) -> [[\lambda \mapsto f_1]], \ \backslash
7 x(a,b,c) -> [[ \alpha_0 -> ?, \
8 @ -> |hello|($), x -> |FALSE| ]], \backslash
9 \Delta = |43-09|,
10 x(y) == x(0-> y).
11 \end{phiuation*}
```

If not a single line is indented in `phiuation`, all formulas will be centered:

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

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

It is possible to use “manual splitting” mode in the `phiuation` environment by starting the body with `\begin{split}`:

$$\begin{aligned} x(\pi) &\mapsto 4 \\ x(a, b, c) &\mapsto [\![\alpha_0 \mapsto \emptyset]\!] \end{aligned}$$

```

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

```

When necessary to use a percentage sign, prepend it with a backward slash:

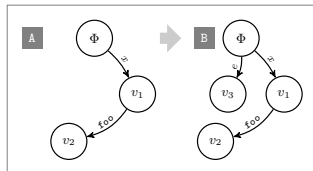
$$x \mapsto \text{sprintf}(\text{"Hello, \%s!"}, \text{name})$$

```

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

```

You can make a copy of a vertex together with its kids:

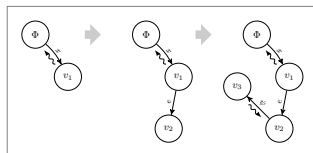


```

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}

```

You can make a copy from a copy:

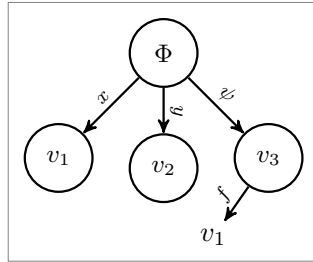


```

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:{\psif} rho
15 \end{sodg}

```

You can have “broken” edges, using “break” attribute of an edge. The attribute must have a value, which is the percentage of the path between vertices that the arrow should take (can’t be more than 80 and less than 20). This may be convenient when you can’t fit all edges into the graph, for example:

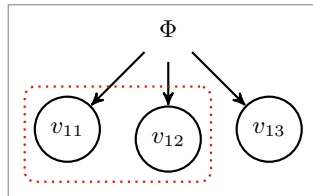


```

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}

```

You can add [TikZ](#) commands to `sodg` graph, for example:

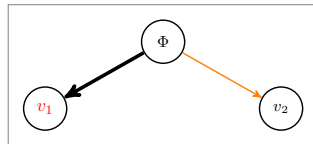


```

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}

```

You can modify TikZ style yourself (make sure `style:` stays at the end of the line!), for example:



```

6 \begin{sodg}
7 v0
8 v1 xy:v0,-2,1 style:font=\color{red}
9 v2 xy:v0,2,1
10 v0->v1 style:line width=2pt
11 v0->v2 style:draw=orange
12 \end{sodg}

```

4 Implementation

First, we include a few packages. We need [stmaryrd](#) for `\llbracket` and `\rrbracket` commands:

```
1 \RequirePackage{stmaryrd}
```

We need [amsmath](#) for `equation*` environment:

```
2 \RequirePackage{amsmath}
```

We need [amssymb](#) for `\varnothing` command. We disable `\Bbbk` because it may conflict with some packages from [acmart](#):

```
3 \let\Bbbk\relax\RequirePackage{amssymb}
```

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

```
4 \RequirePackage{fancyvrb}
```

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

```
5 \RequirePackage{iexec}
```

Then, we process package options:

```

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

```

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

```

18 \RequirePackage{shellesc}
19 \IfFileExists
20   {\eolang@tmpdir/\jobname}
21   {\message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
22     already exists^^J}}
23   {%
24     \ifnum\ShellEscapeStatus=1%
25       \iexec[null]{mkdir -p "\eolang@tmpdir/\jobname"}%
26     \else%
27       \message{eolang: Temporary directory "\eolang@tmpdir/\jobname"
28         is not created, because -shell-escape is not set, and
29         it doesn't exist, most probably the compilation
30         will fail later^^J}%
31     \fi%
32   }

```

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

```

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

```

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

```

34 \RequirePackage{pdftexcmds}
35 \makeatletter
36 \newcommand\eolang@mdfive[1]{\pdf@filemdfivesum{#1}}
37 \makeatother

```

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

```

38 \makeatletter
39 \openin 15=\eolang@tmpdir/\jobname-phi.pl
40 \ifeof 15
41 \message{eolang: Perl script is going to be created,
42   because it is absent at "\eolang@tmpdir/\jobname-phi.pl",
43   but if -shell-escape is not set, the compilation will
44   most likely fail now^^J}
45 \begin{VerbatimOut}{\eolang@tmpdir/\jobname-phi.pl}
46 $macro = $ARGV[0];
47 open(my $fh, '<', $ARGV[1]);
48 my $tex; { local $/; $tex = <$fh>; }
49 print "% This file is auto-generated by 0.17.1\n";

```

```

50 print '% There are ', length($tex),
51 ' chars in the input: ', $ARGV[1], "\n";
52 print '% ---', "\n";
53 if (index($tex, "\t") > 0) {
54   print "TABS are prohibited!";
55   exit 1;
56 }
57 my @lines = split (/\\n/g, $tex);
58 foreach my $t (@lines) {
59   print '% ', $t, "\n";
60 }
61 print '% ---', "\n";
62 $tex =~ s/(?<\\)%.*\\n\\n/g;
63 $tex =~ s/^\\s+|\\s+$/g;
64 my $splitting = $tex =~ /^\\begin\\{split\\}/;
65 if ($splitting) {
66   print '% The manual splitting mode is ON since \\begin{split} started the text' . "\n";
67 }
68 my $indents = $tex =~ /^\\n +/g;
69 my $gathered = (0 == $indents);
70 if ($gathered) {
71   if ($splitting) {
72     print '% The "gathered" is NOT used because of manual splitting' . "\n";
73     $gathered = 0;
74   } else {
75     print '% The "gathered" is used since all lines are left-aligned' . "\n";
76   }
77 } else {
78   print '% The "gathered" is NOT used because ' .
79     $indents . " lines are indented\n";
80 }
81 my $align = 0;
82 print '% The "align" is NOT used by default' . "\n";
83 if (index($tex, '&&') >= 0) {
84   $macro =~ s/equation/align/g;
85   $align = 1;
86   print '% The "align" is used because of && seen in the text' . "\n";
87 }
88 if ($macro ne 'phiq') {
89   if (not $splitting) {
90     $tex =~ s/\\\\\\n\\n\\n/g;
91     $tex =~ s/\\\\\\n\\s*/g;
92   }
93   $tex =~ s/\\n*(\\label\\{[~\\}]+\\})\\n*/\\1/g;
94   $tex =~ s/\\n{3,}/\\n\\n/g;
95 }
96 my @texts = ();
97 sub trep {
98   my ($s) = @_ ;
99   my $open = 0;
100   my $p = 0;
101   for (; $p < length($s); $p++) {
102     $c = substr($s, $p, 1);
103     if ($c eq '}') {

```

```

104     if ($open eq 0) {
105         last;
106     }
107     $open--;
108 }
109 if ($c eq '{') {
110     $open++;
111 }
112 }
113 push(@texts, substr($s, 0, $p));
114 return '{TEXT' . (0+@texts - 1) . '}' . substr($s, $p + 1);
115 }
116 $tex =~ s/\\text\\{(.+)/trep("$1")/ge;
117 if (not $splitting) {
118     $tex =~ s/(?![{&}&(?![&]))\\sigma{/g;
119 }
120 $tex =~ s/([~\\{a-z0-9]|^)^QQ(?![a-z0-9])/\\1\\dot{\\Phi}/g;
121 $tex =~ s/([~\\{a-z0-9]|^)^Q(?![a-z0-9])/\\1\\Phi/g;
122 $tex =~ s/([~\\{a-z0-9]|^)^D>/\\1\\Delta{..}/g;
123 $tex =~ s/([~\\{a-z0-9]|^)^L>/\\1\\lambda{..}/g;
124 $tex =~ s/"([~"]+)"|'|"1"/g;
125 $tex =~ s/(^|(?=[s])(\\[, .>\\|))([a-zA-Z][a-z0-9]+)(?=[s])(\\[, .-]|$)/|2|/g;
126 $tex =~ s/([~_]|^)([0-9]+|\\*)\\/(\\?[a-z]+|\\|[a-z]+|\\|)
127 (->|\\.\\.\\.>|>|:=|!->)/\\1\\alpha_{2}\\|\\vert{\\3\\space{\\4}/xg;
128 $tex =~ s/([~_]|^)([0-9]+|\\*)
129 (->|\\.\\.\\.>|>|:=|!->)/\\1\\alpha_{2}\\|\\space{\\3}/xg;
130 if ($macro ne 'phiq') {
131     if (not $splitting) {
132         $tex =~ s/\\begin\\{split\\}\\n\\begin{split}&/g;
133         $tex =~ s/\\n\\s*\\end\\{split\\}\\n\\end{split}/g;
134         $tex =~ s/\\n\\n\\|\\|\\|&/g;
135         $tex =~ s/\\n\\|\\phiEOL{\\}\\n&/g;
136         $tex =~ s/\\|\\|\\|$/g;
137         $tex =~ s/\\|\\|\\|\\|\\|\\n/g;
138         $tex =~ s/([~&s])\\s{2}([~s])/\\1 \\2/g;
139         $tex =~ s/\\s{2}/ \\quad/g;
140         $tex = '&' . $tex;
141     }
142     my $lead = '[~s]+\\s(?:->|:=|!>|==)\\s';
143     my @leads = $tex =~ /&${lead}/g;
144     my @eols = $tex =~ /&/g;
145     if (0+@leads == 0+@eols && 0+@eols > 1) {
146         $tex =~ s/&(${lead})/\\1&~/g;
147         $gathered = 0;
148         print '% The "gathered" is NOT used because all ' .
149             (0+@eols) . ' lines are ' . (0+@leads) . " leads\\n";
150     }
151 }
152 if ($macro ne 'phiq') {
153     sub strip_tabs {
154         my ($env, $tex) = @_;
155         $tex =~ s/&/g;
156         return "\\begin{$env}" . $tex . "\\end{$env}";
157     }

```

```

158 foreach my $e (('matrix', 'cases')) {
159     $tex =~ s/\\begin\{(\Q$e\E*?)\}(.)\\end\{(\Q$e\E*?)\}/strip_tabs($1, $2)/sge;
160 }
161 }
162 $tex =~ s/\$/\\xi{/g;
163 $tex =~ s/(?<!\{)\^(?!\\{)/\\rho{/g;
164 $tex =~ s/[\\[\lbracket\\mathbin{/g;
165 $tex =~ s/[\\]\rbracket\\mathbin{/g;
166 $tex =~ s/([s,>()]{0-9A-F}{2}(?:-[0-9A-F]{2})+|
167 [0-9]+(?:\.[0-9]+)?)(?!\\{)/1|2|/xg;
168 $tex =~ s/TRUE/|TRUE|/g;
169 $tex =~ s/FALSE/|FALSE|/g;
170 $tex =~ s/\?/\\varnothing{/g;
171 $tex =~ s/@/\\varphi{/g;
172 $tex =~ s/-([a-z]+)>/\\mathrel{\\phiSlot{1}}/g;
173 $tex =~ s/!->/\\mathbin{\\phiConst}/g;
174 $tex =~ s/->/\\mathbin{\\mapsto}/g;
175 $tex =~ s/~>/\\mathbin{\\phiWave}/g;
176 $tex =~ s/:=/\\mathrel{\\vDash}/g;
177 $tex =~ s/==/\\mathrel{\\equiv}/g;
178 $tex =~ s/\\.\\.\\.>/\\mathbin{\\phiDotted}/g;
179 $tex =~ s/<</\\langle/g;
180 $tex =~ s/>>/\\rangle/g;
181 $tex =~ s/|{2,}|/g;
182 $tex =~ s/|([~|]+)\\|/\\textnormal{\\texttt{1}}{/g;
183 $tex =~ s/\\{TEXT(d+)\\}/'\\text{' . @texts[$1] . '}'/ge;
184 if ($macro eq 'phiq') {
185     print '$' if ($tex ne '');
186 } else {
187     print '\\begin{' , $macro, "}\\n";
188     if (not($align)) {
189         if ($gathered) {
190             print '\\begin{gathered}' . "\\n";
191         } elsif (not $splitting) {
192             print '\\begin{split}' . "\\n";
193         }
194     }
195 }
196 if ($gathered and not($align)) {
197     $tex =~ s/^&/g;
198     $tex =~ s/\\n&/\\n/g;
199 }
200 print $tex;
201 if ($macro eq 'phiq') {
202     print '$' if ($tex ne '');
203 } else {
204     if (not($align)) {
205         if ($gathered) {
206             print "\\n" . '\\end{gathered}';
207         } elsif (not $splitting) {
208             print "\\n" . '\\end{split}';
209         }
210     }
211     print "\\n" . '\\end{' . $macro . '}' ;

```

```

212 }
213 print '\endinput';
214 \end{VerbatimOut}
215 \message{eolang: File with Perl script
216   '\eolang@tmpdir/\jobname-phi.pl' saved^^J}
217 \else
218   \message{eolang: Perl script already exists at
219     "\eolang@tmpdir/\jobname-phi.pl"^^J}
220 \fi
221 \closein 15
222 \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:

```

223 \makeatletter
224 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
225 \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:

```

226 \makeatletter
227 \newcommand\eolang@ifabsent[2]{%
228   \IfFileExists
229     {#1}
230     {%
231       \message{eolang: File "#1" already exists ^^J}%
232       \input{#1}}
233   {%
234     \ifnum\ShellEscapeStatus=1\else%
235       \message{eolang: The -shell-escape command line
236         option is not provided, most probably compilation
237         will fail now:^^J}%
238     \fi%
239     #2%
240   }%
241 }
242 \makeatother

```

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

```

243 \makeatletter\newcommand\eolang@process[1]{
244   \def\hash{\eolang@mdfive
245     {\eolang@tmpdir/\jobname/phiuation.tex}-\the\inputlineno}%
246   \eolang@ifabsent
247     {\eolang@tmpdir/\jobname/\hash-post.tex}
248     {%
249       \iexec[null]{cp "\eolang@tmpdir/\jobname/phiuation.tex"
250         "\eolang@tmpdir/\jobname/\hash.tex"}%
251       \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
252       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
253         perl "\eolang@tmpdir/\jobname-phi.pl"
254         ' #1 '
255         "\eolang@tmpdir/\jobname/\hash.tex"
256         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi

```

```

257         \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
258     }%
259     \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
260     \def\eolang@phiSaveTo{\relax}%
261 }
262 %
263 \newenvironment{phiqutation*}%
264 {\catcode'\|=12 \VerbatimEnvironment%
265 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
266 \begin{VerbatimOut}
267   {\eolang@tmpdir/\jobname/phiqutation.tex}}
268 {\end{VerbatimOut}\eolang@process{equation*}}
269 %
270 \newenvironment{phiqutation}%
271 {\catcode'\|=12 \VerbatimEnvironment%
272 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
273 \begin{VerbatimOut}
274   {\eolang@tmpdir/\jobname/phiqutation.tex}}
275 {\end{VerbatimOut}\eolang@process{equation}}
276 \makeatother

```

\phiq Then, we define \phiq command:

```

277 \RequirePackage{xstring}
278 \makeatletter\newcommand\phiq[1]{%
279   \StrSubstitute{\detokenize{#1}}{''}{'"''}[\clean]%
280   \def\hash{\pdf@mdfivesum{\clean}-\the\inputlineno}%
281   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
282   \eolang@ifabsent
283     {\eolang@tmpdir/\jobname/\hash-phiq-post.tex}
284     {%
285       \iexec[log,trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{
286         /bin/echo '\clean'}%
287       \iexec[quiet,null]{cp "\eolang@tmpdir/\jobname/phiq.tex"
288         "\eolang@tmpdir/\jobname/\hash-phiq.tex"%
289       \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-phiq-post.tex]{
290         perl \eolang@tmpdir/\jobname-phi.pl 'phiq'
291         "\eolang@tmpdir/\jobname/\hash-phiq.tex"
292         \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g' \fi}%
293     }%
294   \ifdefined\eolang@nodollar\else\catcode'\$=\active\fi%
295 }\makeatother

```

nodollar Then, we redefine dollar sign:

```

296 \ifdefined\eolang@nodollar\else
297   \begingroup
298   \catcode'\$=\active
299   \protected\gdef$#1${\phiq{#1}}
300   \endgroup
301   \AtBeginDocument{\catcode'\$=\active}
302 \fi

```

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

```

303 \makeatletter

```



```

304 \openin 15=\eolang@tmpdir/\jobname-sodg.pl
305 \ifeof 15
306 \message{eolang: Perl script is going to be created,
307   because it is absent at "\eolang@tmpdir/\jobname-sodg.pl",
308   but if -shell-escape is not set, the compilation will
309   most likely fail now^^J}
310 \begin{VerbatimOut}{\eolang@tmpdir/\jobname-sodg.pl}
311 sub num {
312   my ($i) = @_;
313   $i =~ s/(\+|-)\./\10./g;
314   return $i;
315 }
316 sub fmt {
317   my ($tex) = @_;
318   $tex =~ s/\\([^\|]+\|\\|\\textnormal{\\texttt{1}})/g;
319   return $tex;
320 }
321 sub vertex {
322   my ($v) = @_;
323   if (index($v, 'v0') == 0) {
324     return '\Phi';
325   } else {
326     $v =~ s/^v/v_/g;
327     $v =~ s/[0-9]$/g;
328     return $v . '>';
329   }
330 }
331 sub tailor {
332   my ($t, $m) = @_;
333   $t =~ s/<([A-Z]?${m}[A-Z]?):([>]+)>/\2/g;
334   $t =~ s/<[A-Z]+:[>]+>/\2/g;
335   return $t;
336 }
337 open(my $fh, '<', $ARGV[0]);
338 my $tex; { local $/; $tex = <$fh>; }
339 if (index($tex, "\t") > 0) {
340   print "TABS are prohibited!";
341   exit 1;
342 }
343 print '% This file is auto-generated', "\n%\n";
344 print '% --- there are ', length($tex),
345   ' chars in the input (', $ARGV[0], "):\n";
346 foreach my $t (split (/\\n/g, $tex)) {
347   print '% ', $t, "\n";
348 }
349 print "% ---\n";
350 $tex =~ s/\\\\\\/\n/g;
351 $tex =~ s/\\\\\n//g;
352 $tex =~ s/([a-zA-Z])\s+/\1/g;
353 $tex =~ s/\n{2,}/\n/g;
354 my @cmds = split (/\\n/g, $tex);
355 print '% --- before processing:' . "\n";
356 foreach my $t (split (/\\n/g, $tex)) {
357   print '% ', $t, "\n";

```

```

358 }
359 print '% ---';
360 print ' (' . (0+@cmds) . " lines)\n";
361 print '\begin{picture}', "\n";
362 for (my $c = 0; $c < 0+@cmds; $c++) {
363   my $cmd = $cmds[$c];
364   $cmd =~ s/^\s+//g;
365   $cmd =~ s/(?!\\)%.*//g;
366   my ($head, $tail) = split(/ /, $cmd, 2);
367   my %opts = {};
368   my ($body, $style) = split(/style:/, $tail, 2);
369   $opts{'style'} = $style;
370   $tail = $body;
371   foreach my $p (split(/ /, $tail)) {
372     my ($q, $t) = split(/:/, $p);
373     $opts{$q} = $t;
374   }
375   if (index($head, '\\') == 0) {
376     print $cmd;
377   } elsif (index($head, '->') >= 0) {
378     my $draw = '\draw[';
379     if (exists $opts{'pi'}) {
380       $draw = $draw . '<MB:phi-pi><F:draw=none>';
381       if (not exists $opts{'a'}) {
382         $opts{'a'} = '\pi';
383       }
384     }
385     if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
386       $draw = $draw . '<MB:.,phi-rho>';
387     }
388     $draw = $draw . ',' . $opts{'style'} . ']';
389     my ($from, $to) = split (/->/, $head);
390     $draw = $draw . " (${$from}) ";
391     if (exists $opts{'bend'}) {
392       $draw = $draw . 'edge [<F:draw=none><MF:.,bend right=' .
393         num($opts{'bend'}) . '>';
394       if (exists $opts{'rho'}) {
395         $draw = $draw . '<MB:.,phi-rho>';
396       }
397       $draw = $draw . ']';
398     } else {
399       $draw = $draw . '--';
400     }
401     if (exists $opts{'a'}) {
402       my $a = $opts{'a'};
403       if (index($a, '$') == -1) {
404         $a = '$' . fmt($a) . '$';
405       } else {
406         $a = fmt($a);
407       }
408       $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
409     }
410     if (exists $opts{'break'}) {
411       $draw = $draw . '<F: coordinate [pos=' .

```

```

412     ($opts{'break'} / 100) . ']' (break)>';
413 }
414 $draw = $draw . " (<MF:${to}><B:break-v>";
415 if (exists $opts{'break'}) {
416     print tailor($draw, 'F') . ";\n";
417     print ' \node[outer sep=.1cm,inner sep=0cm] ' .
418         'at (break) (break-v) {$' . vertex($to) .
419         '$};' . "\n";
420     print ' ' . tailor($draw, 'B');
421 } else {
422     print tailor($draw, 'M');
423 }
424 } elsif (index($head, '=>') >= 0) {
425     my ($from, $to) = split (/=>/, $head);
426     my $size = () = $head =~ /=/g;
427     if ($from eq '') {
428         print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
429             $to . '.center]';
430     } elsif ($to eq '') {
431         print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
432             $from . '.center]';
433     } else {
434         print '\node [phi-arrow] at ($(' .
435             $from . ')!0.5!(' . $to . ')$)';
436     }
437     print '{}';
438 } elsif (index($head, '!'') >= 0) {
439     my ($v, $marker) = split (/!+/, $head);
440     my $size = () = $head =~ /*!/g;
441     print '\node [phi-marker, left=' .
442         ($size * 0.6) . 'cm of ' .
443         $v . '.center]{' . fmt($marker) . '}';
444 } elsif (index($head, '+') >= 0) {
445     my ($v, $suffix) = split (/+/, $head);
446     my @friends = ($v);
447     foreach my $c (@cmds) {
448         $e = $c;
449         $e =~ s/^\s+//g;
450         my $h = $e;
451         $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
452         foreach my $f (@friends) {
453             my $add = '';
454             if (index($h, $f . '->') >= 0) {
455                 $add = substr($h, index($h, '->') + 2);
456             }
457             if ($h =~ /\->\Q${f}\E/) {
458                 $add = substr($h, 0, index($h, '->'));
459             }
460             if (index($e, ' xy:' . $f . ',') >= 0) {
461                 $add = $h;
462             }
463             if (index($add, '+') == -1
464                 and $add ne ''
465                 and not(grep(/\Q${add}\E/, @friends))) {

```

```

466         push(@friends, $add);
467     }
468 }
469 }
470 my @extra = ();
471 foreach my $e (@cmds) {
472     $m = $e;
473     if ($m =~ /\s*\Q${v}\E\s/) {
474         next;
475     }
476     if ($m =~ /\s*[^\s]+\s/ and not($m =~ /\s*\Q${head}\E\s/)) {
477         next;
478     }
479     foreach my $f (@friends) {
480         my $h = $f;
481         $h =~ s/[a-z]$//g;
482         if ($m =~ s/^(\\s*)\Q${f}\E+\Q${suffix}\E\s?/\1${h}${suffix} /g) {
483             last;
484         }
485         $m =~ s/^(\\s*)\Q${f}\E\s/\1${h}${suffix} /g;
486         $m =~ s/^(\\s*)\Q${f}\E->/\1${h}${suffix}->/g;
487         $m =~ s/\\sxy:\Q${f}\E,/ xy:${h}${suffix},/g;
488         $m =~ s/->\Q${f}\E\s/->${h}${suffix} /g;
489     }
490     if ($m ne $e) {
491         push(@extra, ' ' . $m);
492     }
493 }
494 splice(@extra, 0, 0, @extra[-1]);
495 splice(@extra, -1, 1);
496 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
497     '), friends: [' . join(', ', @friends) . ']' in ' .
498     '(0+@cmds) . ' lines');
499 splice(@cmds, $c, 1, @extra);
500 print '% cloned ' . $v . ' at line no.' . $c .
501     ' (+ ' . (0+@extra) . ' lines -> ' .
502     '(0+@cmds) . ' lines total)';
503 } elsif ($head =~ /\v[0-9]+[a-z]?$/) {
504     print '\node[';
505     if (exists $opts{'xy'}) {
506         my ($v, $right, $down) = split(/,/ , $opts{'xy'});
507         my $loc = '';
508         if ($down > 0) {
509             $loc = 'below ';
510         } elsif ($down < 0) {
511             $loc = 'above ';
512         }
513         if ($right > 0) {
514             $loc = $loc . 'right';
515         } elsif ($right < 0) {
516             $loc = $loc . 'left';
517         }
518         print ', ' . $loc . '=';
519         print abs(num($down)) . 'cm and ' .

```

```

520         abs(num($right)) . 'cm of ' . $v . '.center';
521     }
522     if (exists $opts{'data'}) {
523         print ',phi-data';
524         if ($opts{'data'} ne '') {
525             my $d = $opts{'data'};
526             if (index($d, '|') == -1) {
527                 $d = '$\Delta\phiDotted\text{' .
528                     '\textnormal{\texttt{' . fmt($d) . '}}}$';
529             } else {
530                 $d = fmt($d);
531             }
532             $opts{'box'} = $d;
533         }
534     } elsif (exists $opts{'atom'}) {
535         print ',phi-atom';
536         if ($opts{'atom'} ne '') {
537             my $a = $opts{'atom'};
538             if (index($a, '$') == -1) {
539                 $a = '$\lambda\phiDotted{' . fmt($a) . '$';
540             } else {
541                 $a = fmt($a);
542             }
543             $opts{'box'} = $a;
544         }
545     } else {
546         print ',phi-object';
547     }
548     if (exists $opts{'edgeless'}) {
549         print ',draw=none';
550     }
551     print ', ' . $opts{'style'} . ']';
552     print ' (' . $head . ')';
553     print '{';
554     if (exists $opts{'tag'}) {
555         my $t = $opts{'tag'};
556         if (index($t, '$') == -1) {
557             $t = '$' . $t . '$';
558         } else {
559             $t = fmt($t);
560         }
561         print $t;
562     } else {
563         print '$' . vertex($head) . '$';
564     }
565     print '}';
566     if (exists $opts{'box'}) {
567         print ' node[phi-box] at (';
568         print $head, '.south east) {';
569         print $opts{'box'}, ')';
570     }
571 }
572 print ";\n";
573 }

```

```

574 print '\end{phicture}%', "\n";
575 print "% --- after processing:\n%";
576 foreach my $c (@cmds) {
577   print '% ', $c, "\n";
578 }
579 print '% --- (' . (0+@cmds) . " lines)\n";
580 print '\endinput';
581 \end{VerbatimOut}
582 \message{eolang: File with Perl script
583   '\eolang@tmpdir/\jobname-sodg.pl' saved^^J}
584 \else
585   \message{eolang: Perl script already exists at
586     "\eolang@tmpdir/\jobname-sodg.pl"^^J}
587 \fi
588 \closein 15
589 \makeatother

```

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

```

590 \setcounter{FancyVerbLine}{0}

```

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

```

591 \RequirePackage{tikz}
592 \usetikzlibrary{arrows}
593 \usetikzlibrary{shapes}
594 \usetikzlibrary{decorations}
595 \usetikzlibrary{decorations.pathmorphing}
596 \usetikzlibrary{decorations.pathreplacing}
597 \usetikzlibrary{positioning}
598 \usetikzlibrary{calc}
599 \usetikzlibrary{math}
600 \usetikzlibrary{arrows.meta}

```

phicture Then, we define internal environment phicture:

```

601 \newenvironment{phicture}%
602   {\noindent\begin{tikzpicture}[
603     ->,>=stealth',node distance=0,thick,
604     pics/parallel arrow/.style={
605       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
606   {\end{tikzpicture}}
607 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
608   minimum height=0.5cm, minimum width=0.5cm,
609   single arrow head extend=2mm]
610 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
611   minimum width=1.4em, font={\small\color{white}\ttfamily},
612   fill=gray]
613 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
614   draw,font={\small}]
615 \tikzstyle{phi-object} = [phi-thing,circle]
616 \tikzstyle{phi-data} = [phi-thing,regular polygon,
617   regular polygon sides=8]
618 \tikzstyle{phi-empty} = [phi-object]
619 \tikzset{%
620   phi-rho/.style={

```

```

621 postaction={%
622   decoration={
623     show path construction,
624     curveto code={
625       \tikzmath{
626         coordinate \I, \F, \v;
627         \I = (\tikzinputsegmentfirst);
628         \F = (\tikzinputsegmentlast);
629         \v = ($(\I) -(\F)$);
630         real \d, \a, \r, \t;
631         \d = 0.8;
632         \t = atan2(\vy, \vx);
633         if \vx<0 then { \a = 90; } else { \a = -90; };
634         {
635           \draw[arrows={-latex}, decorate,
636             decoration={%
637               snake, amplitude=.4mm,
638               segment length=2mm,
639               post length=1mm
640             }]
641             ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
642             -- ++(\t: 2*\d em);
643         };
644       }
645     },
646     lineto code={
647       \tikzmath{
648         coordinate \I, \F, \v;
649         \I = (\tikzinputsegmentfirst);
650         \F = (\tikzinputsegmentlast);
651         \v = ($(\I) -(\F)$);
652         real \d, \a, \r, \t;
653         \d = 0.8;
654         \t = atan2(\vy, \vx);
655         if \vx<0 then { \a = 90; } else { \a = -90; };
656         {
657           \draw[arrows={-latex}, decorate,
658             decoration={%
659               snake, amplitude=.4mm,
660               segment length=2mm,
661               post length=1mm}]
662             ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
663             -- ++(\t: 2*\d em);
664         };
665       }
666     }
667   },
668   decorate
669 }
670 }
671 }
672 \tikzstyle{phi-pi} = [draw,dotted]
673 \tikzstyle{phi-atom} = [phi-object,double]
674 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,

```

```

675 rectangle,thin,minimum width=1.2em,anchor=north west,
676 font={\scriptsize}]
677 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
678 above=2pt,sloped/.append style={transform shape},
679 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:

```

680 \makeatletter
681 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
682 \makeatother

```

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

```

683 \makeatletter\newenvironment{sodg}%
684 {\catcode'\|=12 \VerbatimEnvironment%
685 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
686 \begin{VerbatimOut}
687   {\eolang@tmpdir/\jobname/sodg.tex}}
688 {\end{VerbatimOut}}%
689 \def\hash{\eolang@mdfive
690   {\eolang@tmpdir/\jobname/sodg.tex}-\the\inputlineno}%
691 \catcode'\$=3 %
692 \eolang@ifabsent
693   {\eolang@tmpdir/\jobname/\hash-sodg-post.tex}
694   {%
695     \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
696       "\eolang@tmpdir/\jobname/\hash.tex"}%
697     \message{eolang: Start parsing 'sodg' at line no. \the\inputlineno^^J}
698     \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-sodg-post.tex]{
699       perl "\eolang@tmpdir/\jobname-sodg.pl"
700       "\eolang@tmpdir/\jobname/\hash.tex"
701       \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
702       \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%
703   }
704 \catcode'\$=active%
705 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
706 \def\eolang@sodgSaveTo{\relax}%
707 }\makeatother

```

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

```

708 \RequirePackage{hyperref}
709 \pdfstringdefDisableCommands{
710   \def\<{}%
711   \def\>{}%
712   \def\alpha{alpha}%
713   \def\varphi{phi}%
714 }
715 \makeatletter
716 \NewExpandableDocumentCommand{\eoAnon}{O{ANONYMIZED}m}{%
717   \ifdefined\eolang@anonymous%
718     \textcolor{orange}{#1}%
719   \else%
720     #2%
721   \fi%

```



```

722 }\makeatother

\eoLang Then, we define a simple supplementary command to help you print EO, the name of
our language.
723 \newcommand\eoLang{%
724   \eoAnon[XYZ]{\sffamily EO}}

\phiC Then, we define a simple supplementary command to help you print  $\varphi$ -calculus, the
name of our formal apparatus.
725 \newcommand\phiC{%
726   \eoAnon[(\alpha)-cal-cu-lus]{(\varphi)-cal-cu-lus}}

\xmir Then, we define a simple supplementary command to help you print XMIR, the name of
our XML-based format of program representation.
727 \newcommand\xmir{%
728   \eoAnon[XML\(^+\)]{XMIR}}

\phiConst Then, we define a command to render an arrow for a constant attribute, as suggested
here:
729 \newcommand\phiConst{%
730   \mathrel{\hspace{.15em}}%
731   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

\phiWave Then, we define a command to render an arrow for a multi-layer attribute, as suggested
here:
732 \newcommand\phiWave{%
733   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

\phiSlot Then, we define a command to render an arrow for a slot in a basket:
734 \newcommand\phiSlot[1]{%
735   \xrightarrow{\text{\sffamily\scshape #1}}}

\phiOset Then, we define two commands to position a text over and under an arrow, as suggested
here:
736 \makeatletter
737 \newcommand{\phiOset}[2]{%
738   \mathrel{\mathop{#2}\limits^{
739     \vbox to 0ex{\kern-2\ex@
740       \hbox{$\scriptscriptstyle#1$}\vss}}}}
741 \newcommand{\phiUset}[2]{%
742   \mathrel{\mathop{#2}\limits_{
743     \vbox to 0ex{\kern-6.3\ex@
744       \hbox{$\scriptscriptstyle#1$}\vss}}}}
745 \makeatother

\phiMany Then, we define a command for an arrow with iterating indecies:
746 \newcommand\phiMany[3]{%
747   \phiOset{#3}{\phiUset{#2}{#1}}}

\phiEOL Then, we define a command for line breaks in formulas:
748 \newcommand\phiEOL{\\[-4pt]}

```

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

```

749 \RequirePackage{trimclip}
750 \RequirePackage{amsfonts}
751 \makeatletter
752 \newcommand{\phiDotted}{%
753   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
754 \newcommand{\phiDotted@}[2]{%
755   \begingroup%
756   \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
757   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
758   \sbox\z@{%
759     \makebox[\dimen\z@][s]{%
760       \clipbox{0 0 {0.4\width} 0}%
761       {\resizebox{\dimen\z@}{\height}%
762         {\m@th#1\dashrightarrow}}}%
763     \hss%
764     \clipbox{{0.69\width} {-0.1\height} 0
765       {-\height}}{\m@th#1\rightarrow}%
766   }%
767 }%
768 \ht\z@=\dimen\tw@ \dp\z@=\z@%
769 \box\z@%
770 \endgroup%
771 }
772 \makeatother

```

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	General: First draft.	10	0.12.1	-sodg.pl: The bug is fixed related to the formatting of indexes of vertices.	16
0.0.2	sodg: The environment phigure renamed to sodg for the sake of better semantic. The graph in the picture is solely a SODG graph, that's why the name sodg is better. 24		0.13.0	-phi.pl: Parsing of QQ into \dot{\Phi} implemented.	11
	-phi.pl: New symbol added for basket slots	11	0.14.0	-sodg.pl: The edgeless tag of a vertex removes the border of it. . .	16
	Parsing of the symbols “@,” “^,” and “&” enabled (\varphi, \rho, and \sigma)	11	0.15.0	-sodg.pl: The style tag of vertices and edges.	16
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets	11	0.16.0	phiquation: The processing of phiquation data is done only if it's the first time processing, otherwise cache is used, thus making processing faster.	15
	\phiq: Parsing of additional symbols enabled.	16	sodg: The processing of sodg data is done only if it's the first time processing, otherwise cache is used, thus making processing faster.	24	
	-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. .	16	0.17.0	\eolang@ifabsent: A new supplementary eolang@ifabsent command added	15
0.1.0	General: Parsing of package options introduced.	11	0.2.0	-phi.pl: Numbers automatically render as \texttt. No need to use vertical bars around them anymore. 11	
	\eolang: New command \eolang added to print the name of the language in both normal and the anonymous mode of acmart. . . .	25	-sodg.pl: The content of the atom and the data boxes is parsed automatically as formulas and numbers, respectively.	16	
	\eolang@mdfive: New supplementary command added to calculate MD5 sum of a file.	11	\xmirl: New command \xmirl prints XMIR in both normal and the anonymous mode of acmart. . . .	25	
	-phi.pl: A new Perl script "eolang-phi.pl" added for parsing of phi expressions.	11	0.3.0	\eolang@lineno: New counter for protecting lineno.	11
	\phic: New command \phic prints the name of φ-calculus in both normal and the anonymous mode of acmart.	25	-phi.pl: New arrow added, that looks like \leadsto.	11	
	\phiConst: New command \phiConst added to denote a link to a constant attribute.	25	\phiWave: New command \phiWave added to denote a link to a multi-layer attribute.	25	
	\phiDotted: New command \phiDotted added to denote a link to a special attribute.	26	0.4.0	-sodg.pl: Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed	
	-sodg.pl: There are two Perl scripts now: one for phiquation, another one for sodg.	16			

with the <code>\Delta</code> and the <code>\lambda</code> commands.	16	-phi.pl: New syntax sugar for Φ , just using capital “Q” is enough.	11
Relative positioning of vertices fixed.	16	Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols.	11
0.5.0		Text in quotes is automatically converted to <code>\texttt</code>	11
-phi.pl: Automated formatting of TRUE and FALSE added.	11	0.8.0	
<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow.	25	General: The anonymous package option added.	11
<code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket.	25	-phi.pl: Inside phiquation any text inside the <code>\text</code> macro is not processed.	11
-sodg.pl: It is possible to use TikZ commands inside the sodg environment.	16	<code>\phiOset</code> : New commands <code>\phiOset</code> and <code>\phiUset</code> help position text over and under an arrow.	25
New syntax introduced that allows to make clones of vertices and all their dependants.	16	<code>\phiSaveTo</code> : The output of the phiquation environment can be redirected to a file.	15
Now edges may have the <code>break</code> attribute, to make them shorter.	16	-sodg.pl: The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.	16
0.6.0		<code>\sodgSaveTo</code> : The output of the sodg environment can be redirected to a file.	24
General: Package option <code>nocomments</code> added in order to enable comments suppression in temporary <code>.tex</code> files (may be pretty important for <code>.dtx</code> documents).	11	0.9.0	
-sodg.pl: The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations.	16	<code>\eoAnon</code> : New command <code>\eoAnon</code> added.	24
0.7.0		-phi.pl: Proper handling of the <code>matrix</code> environment.	11
<code>nodollar</code> : Now it is possible to use dollar sign instead of the <code>\phiq</code> command.	16	<code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code>	25

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		<code>\color</code> 611	586, 687, 690, 693, 695, 696, 698, 699, 700
<code>\\$</code> 162, 281, 294, 298, 301, 691, 704		D	<code>\ex@</code> 739, 743
<code>\%</code> 256, 292, 701		<code>\d</code> .. 183, 630, 631, 641, 642, 652, 653, 662, 663	F
<code>\(</code> 710, 726, 728		<code>\dashrightarrow</code> ... 762	<code>\F</code> 626, 628, 629, 641, 648, 650, 651, 662
<code>\)</code> 711, 726, 728		<code>\def</code> 224, 244, 260, 280, 681, 689, 706, 710, 711, 712, 713	<code>\FancyVerbLine</code> 590
<code>*</code> 126, 128, 159		<code>\Delta</code> 527	G
<code>\+</code> 313, 445, 476, 482		<code>\detokenize</code> 279	<code>\gdef</code> 299
<code>\-</code> 726		<code>\dimen</code> 756, 757, 759, 761, 768	H
<code>\-phi.pl</code> 38		<code>\dp</code> 768	<code>\hash</code> 244, 247, 250, 252, 255, 280, 283, 288, 289, 291, 689, 693, 696, 698, 700
<code>\-sodg.pl</code> 303		<code>\draw</code> ... 378, 605, 635, 657	<code>\hbox</code> 740, 744
<code>\.</code> .. 127, 129, 167, 178, 313		E	<code>\height</code> 761, 764, 765
<code>\/</code> 125, 126		<code>\E</code> 159, 457, 465, 473, 476, 482, 485, 486, 487, 488	<code>\hspace</code> 730, 731
<code>\?</code> 170		<code>\end</code> 206, 208, 211, 214, 268, 275, 574, 581, 606, 688	<code>\hss</code> 763
<code>\[</code> 125, 164		<code>\endinput</code> 213, 580	<code>\ht</code> 768
<code>\{</code> 64, 93, 116, 132, 133, 159, 163, 167, 183		<code>\eoAnon</code> . 708, 724, 726, 728	I
<code>\}</code> 64, 93, 132, 133, 159, 183		<code>\eolang</code> 723	<code>\I</code> 626, 627, 629, 641, 648, 649, 651, 662
<code>\]</code> 125, 165		<code>\eolang@anonymous</code> 14, 717	<code>\iexec</code> ... 25, 249, 252, 285, 287, 289, 695, 698
<code>\^</code> 163		<code>\eolang@ifabsent</code> 226, 246, 282, 692	<code>\ifdefined</code> 256, 257, 281, 292, 294, 296, 701, 702, 717
<code>\ </code> 126, 181, 182, 264, 271, 318, 684		<code>\eolang@lineno</code> 33	<code>\ifeof</code> 40, 305
Numbers		<code>\eolang@mdfive</code> 34, 244, 689	<code>\IfFileExists</code> ... 19, 228
<code>\2</code> 125, 127, 129, 138, 167, 333		<code>\eolang@nocomments</code> ... 13, 256, 292, 701	<code>\ifluatex</code> 12
<code>\3</code> 127, 129		<code>\eolang@nodollar</code> 281, 294, 296	<code>\ifnum</code> 24, 234
<code>\4</code> 127		<code>\eolang@phiSaveTo</code> 224, 257, 260	<code>\ifxetex</code> 12
A		<code>\eolang@process</code> 243, 268, 275	<code>\input</code> 232
<code>\a</code> 630, 633, 641, 652, 655, 662		<code>\eolang@sodgSaveTo</code> 681, 702, 706	<code>\inputlineno</code> 245, 251, 280, 690, 697
<code>\active</code> . 294, 298, 301, 704		<code>\eolang@tmpdir</code> 11, 20, 21, 25, 27, 39, 42, 45, 216, 219, 245, 247, 249, 250, 252, 253, 255, 267, 274, 283, 285, 287, 288, 289, 290, 291, 304, 307, 310, 583,	J
<code>\alpha</code> 712, 726			<code>\jobname</code> 20, 21, 25, 27, 39, 42, 45, 216, 219, 245, 247, 249, 250, 252, 253, 255, 267, 274, 283, 285, 287, 288, 289, 290, 291, 304, 307, 310, 583,
<code>\AtBeginDocument</code> .. 301			
B			
<code>\Bbbk</code> 3			
<code>\begin</code> 45, 66, 187, 190, 192, 266, 273, 310, 361, 602, 686			
<code>\box</code> 769			
C			
<code>\catcode</code> 264, 271, 281, 294, 298, 301, 684, 691, 704			
<code>\clean</code> 279, 280, 286			
<code>\clipbox</code> 760, 764			
<code>\closein</code> 221, 588			

586, 687, 690, 693, 695, 696, 698, 699, 700					
K		P		T	
\kern 739, 743		\pdf@filemdfivesum . 36		\small 611, 614	
		\pdf@mdfivesum 280		\sodg 683	
		\pdfstringdefDisableCommand		\sodgSaveTo 680	
	 709		\strSubstitute 279	
		\pgfkeys 9		\sxy 487	
L		\Phi 324			
\lambda 539		\phic 725		T	
\leadsto 733		\phiConst 729		\t 53, 339, 630, 632, 641, 642, 652, 654, 662, 663	
\limits 738, 742		\picture 601		\text 527, 735	
M		\phiDotted . 527, 539, 749		\textcolor 718	
\m@th . . . 756, 757, 762, 765		\phiDotted@ 753, 754		\textnormal 528	
\makeatletter		\phiEOL 748		\texttt 528	
. . 33, 35, 38, 223,		\phiMany 746		\the . 245, 251, 280, 690, 697	
226, 243, 278, 303,		\phiOset 736, 747		\tikz 591	
680, 683, 715, 736, 751		\phiq 277, 299		\tikzinputsegmentfirst 627, 649	
\makeatother		\phiquation 243		\tikzinputsegmentlast 628, 650	
. 33, 37, 222, 225,		\phiSaveTo 223		\tikzmath 625, 647	
242, 276, 295, 589,		\phiSlot 734		\tikzset 619	
682, 707, 722, 745, 772		\phiUset 741, 747		\tikzstyle 607,	
\makebox 759		\phiWave 732		. 610, 613, 615, 616,	
\mapsto 731		\pi 382		. 618, 672, 673, 674, 677	
\mapstochar . 731, 733, 753		\ProcessPgfPackageOptions 17		\ttfamily 611	
\mathop 738, 742		\protected 299		\tw@ 757, 768	
\mathpalette 753					
\mathrel 730,		Q		U	
731, 733, 738, 742, 753		\Q 159, 457, 465, 473, 476, 482, 485, 486, 487, 488		\usetikzlibrary . . . 592, 593, 594, 595, 596, 597, 598, 599, 600	
\message . . . 21, 27, 41, 215, 218, 231, 235, 251, 306, 582, 585, 697		R		V	
\mspace 733		\relax 3, 260, 706, 753		\v 626, 629, 648, 651	
N		\RequirePackage 1, 2, 3, 4, 5, 6, 7, 8, 18, 34, 277, 591, 708, 749, 750		\value 259, 265, 272, 685, 705	
\newcommand 36, 224, 227, 243, 278, 681, 723, 725, 727, 729, 732, 734, 737, 741, 746, 748, 752, 754		\resizebox 761		\varphi 713, 726	
\newcounter 33		\rightarrow . 756, 757, 765		\vbox 739, 743	
\newenvironment 263, 270, 601, 683		S		\VerbatimEnvironment 264, 271, 684	
\NewExpandableDocumentCommand		\sbox 758		\vss 740, 744	
. 716		\scriptscriptstyle 740, 744		\vx 632, 633, 654, 655	
\node 417, 428, 431, 434, 441, 504		\scriptsize 676, 679		\vy 632, 654	
\nodollar 296		\scshape 735		W	
\noindent 602		\setcounter 259, 265, 272, 590, 685, 705		\width 760, 764	
O		\settoheight 757		X	
\openin 39, 304		\settowidth 756		\xmir 727	
		\sffamily 724, 735		\xrightarrow 735	
		\ShellEscapeStatus 24, 234		Z	
				\z@ 756, 758, 759, 761, 768, 769	