



# eolang: $\text{\LaTeX}$ Package for Formulas and Graphs of EO Programming Language and $\varphi$ -calculus\*

Yegor Bugayenko  
yegor256@gmail.com

2023-06-17, 0.12.2

**NB!** You must run  $\text{\TeX}$  processor with `--shell-escape` option and you must have [Perl](#) installed. This package doesn't work on Windows.

## 1 Introduction

This package helps you print formulas of  $\varphi$ -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:

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

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

---

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

- “@” maps to “ $\varphi$ ” (`\varphi`),
- “^” maps to “ $\rho$ ” (`\rho`),
- “\$” maps to “ $\xi$ ” (`\xi`),
- “&” maps to “ $\sigma$ ” (`\sigma`),
- “?” maps to “ $\emptyset$ ” (`\varnothing`),
- “Q” maps to “ $\Phi$ ” (`\Phi`),
- “->” maps to “ $\mapsto$ ” (`\mapsto`),
- “~>” maps to “ $\rightsquigarrow$ ” (`\phiWave`),
- “!->” maps to “ $\multimap$ ” (`\phiConst`),
- “. .>” maps to “ $\vdash$ ” (`\phiDotted`),
- “D>” maps to “ $\Delta \vdash$ ” (`\Delta . .>`),
- “L>” maps to “ $\lambda \vdash$ ” (`\lambda . .>`),
- “[[” maps to “ $\llbracket$ ” (`\llbracket`),
- “]]” maps to “ $\rrbracket$ ” (`\rrbracket`),
- “==” maps to “ $\equiv$ ” (`\equiv`),
- “|abc|” maps to “ $\text{abc}$ ” (`\texttt{abc}`).

Also, a few symbols are supported for  $\varphi$ PU architecture:

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

Before any arrow you can put a number, which will be rendered as `\alpha` 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->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  $\varphi$ -calculus expressions using the same simple plain-text notation. You can use dollar sign directly too:

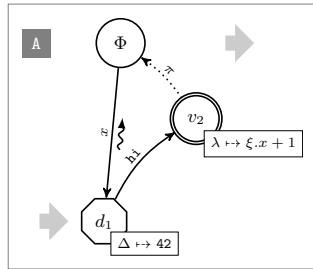
A simple object  $x \mapsto \llbracket \varphi \mapsto y \rrbracket$   
is a decorator of the data object  
 $y \mapsto \llbracket \Delta \mapsto 42 \rrbracket$ .

```

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->v1.” All other markers are either unary like “rho” or binary like “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.

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  $\pi$  label.

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 “ $v3 \Rightarrow$ ”, of from the left to the vertex, by saying for example “ $\Rightarrow v5$ .” If you want the arrow to stay further away from the vertex than usually, use a few “=” symbols, for example “ $=== \Rightarrow v0$ .”

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 language. It understands the anonymous package option and prints itself differently, to `\phic` double-blind your paper. There is also `\phic` command to print the name of  $\varphi$ -calculus, `\xmirl` also sensitive to anonymous mode. The macro `\xmirl` prints “XMIR”.

In our research we use XYZ,  
an experimental object-oriented  
dataflow language,  $\alpha$ -calculus, as its  
formal foundation, and XML<sup>+</sup> —  
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,  $\varphi$ -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 \xmirl{} --- \
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  $x \mapsto y$  makes  $y$  a constant,  $x \rightsquigarrow y$  makes it a decoratee of an arbitrary number of objects, while  $x \vdash y$  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.
```

`\phi0set` If you want to put a text over an arrow or under it, use `\phi0set` and `\phiUset` `\phiUset` respectively:

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

$\llbracket \mapsto^* \rrbracket$ .

```
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{phiuation*}
10 [[ \phiOset{*}{->} ]]
11 \end{phiuation*}
```

`\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  $\llbracket \alpha_1 \mapsto x_1, \alpha_2 \mapsto x_2, \dots, \alpha_n \mapsto x_n \rrbracket$  and expression  $\llbracket \alpha_i \xrightarrow{*} x_i \rrbracket$  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` `\sodgSaveTo` If you want to use `phiuation` or `sodg` environments inside `tabular` or any other environment or command, you won't be able to do this, because `phiuation` and `sodg` are “verbatim” environments. `\phiSaveTo` and `\sodgSaveTo` commands will help you in this situation. You use them right before `\begin{phiuation}` 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:  $\llbracket x \mapsto \llbracket \Delta \vdash 42 \rrbracket \rrbracket$

```
5 \phiSaveTo{a}
6 \begin{phiuation*}
7 [[ x -> [[D>42]] ]]
8 \end{phiuation*}
9 \begin{tabular}{p{.5in}l}
10 Free: & $\llbracket x -> ? \rrbracket$ \\
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:

$\frac{x \mapsto \llbracket \varphi \mapsto y \rrbracket \quad y \mapsto \llbracket z \mapsto 42 \rrbracket}{x.z \mapsto 42} R1$	<pre> 6 \begin{phiuation*} 7 \dfrac \ 8 {x-&gt;[[@-&gt;y]] \quad y-&gt;[[z-&gt;42]]} \ 9 {x.z -&gt; 42} \ 10 \text{\sffamily R1} 11 \end{phiuation*} </pre>
--	---

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

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

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

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

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} \backslash
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:

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

```

1 \documentclass{acmart}
2 \usepackage{eolang}
3 \thispagestyle{empty}
4 \begin{document}
5 \begin{phiuation*}
6 x -> \llbracket
7   y -> \llbracket
8     z !-> $, f ..> ? \rrbracket \rrbracket, \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 \llbracket 0->x \rrbracket && \text{This is formation}
8 \llbracket 0->? \rrbracket && \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 $[\llbracket 0->x \rrbracket]$
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:

$\begin{aligned} x(\pi) &\mapsto \llbracket \lambda \mapsto f_1 \rrbracket, \\ x(a, b, c) &\mapsto \llbracket \alpha_0 \mapsto \emptyset, \varphi \mapsto \text{hello}(\xi), x \mapsto \text{FALSE} \rrbracket, \\ \Delta &= 43-09, \\ x(y) &\equiv x(\alpha_0 \mapsto y). \end{aligned}$	<pre> 5 \begin{phiuation*} 6 x(\pi) -&gt; \llbracket \lambda \mapsto f_1 \rrbracket, \\\ 7 x(a,b,c) -&gt; \llbracket \alpha_0 \mapsto \emptyset, \varphi \mapsto \text{hello}(\xi), x \mapsto \text{FALSE} \rrbracket, \\\ 8 \Delta = 43-09, 9 x(y) == x(\alpha_0 \mapsto y). 10 \end{phiuation*} </pre>
---	--

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

$\begin{aligned} &\llbracket b \mapsto \emptyset \rrbracket, \\ &\llbracket \varphi \mapsto \text{TRUE}, \Delta \mapsto 42 \rrbracket, \\ &\psi = \langle \pi, 42 \rangle. \end{aligned}$	<pre> 5 \begin{phiuation*} 6 \llbracket b \mapsto \emptyset \rrbracket, 7 \llbracket \varphi \mapsto \text{TRUE}, \Delta \mapsto 42 \rrbracket, \\\ 8 \psi = \langle \pi, 42 \rangle. 9 \end{phiuation*} </pre>
---	---

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

$\begin{aligned} x(\pi) &\mapsto 4 \\ x(a, b, c) &\mapsto \llbracket \alpha_0 \mapsto \emptyset \rrbracket \end{aligned}$	<pre> 5 \begin{phiuation*} 6 \begin{split} 7 x(\pi) &amp;\mapsto 4 \\\ 8 x(a,b,c) &amp;\mapsto \llbracket \alpha_0 \mapsto \emptyset \rrbracket \\\ 9 \end{split} 10 \end{phiuation*} </pre>
---	--

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

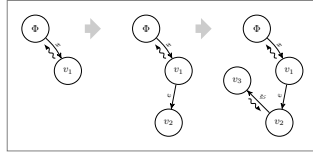
$x \mapsto \text{sprintf}(\text{"Hello, \%s!"}, \text{name})$	<pre> 5 \begin{phiuation*} 6 x -&gt; \text{sprintf}(\text{"Hello, \%s!"}, \text{name}) 7 \end{phiuation*} 8 \end{document} </pre>
---	---

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

	<pre> 5 \begin{sodg} 6 v0 \\\ v0!!A 7 v1 xy:v0,.7,1 8 v0-&gt;v1 a:x bend:-10 9 v2 xy:v1,-1.3,.8 10 v1-&gt;v2 a: foo  bend:-20 11 v0+a xy:v0,3,0 12 v3a xy:v0a,-.7,1 13 v0a-&gt;v3a a:e bend:-15 14 v0=&gt;v0a \\\ v0a!B 15 \end{sodg} </pre>
--	--

You can make a copy from a copy:



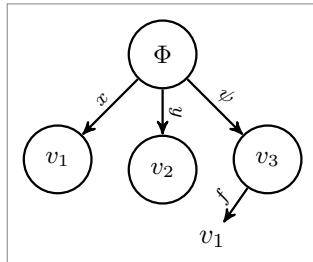


```

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:\psi{} 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:

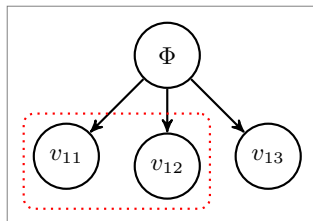


```

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
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}

```

## 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 \iexec[null]{mkdir -p "\eolang@tmpdir/\jobname"}%
```

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

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

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

```
20 \RequirePackage{pdftexcmds}
21 \makeatletter
22 \newcommand\eolang@mdfive[1]{\pdf@filemdfivesum{#1}}
23 \makeatother
```

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

```
24 \makeatletter
25 \begin{VerbatimOut}{\eolang@tmpdir/eolang-phi.pl}
26 $macro = $ARGV[0];
27 open(my $fh, '<', $ARGV[1]);
28 my $tex; { local $/; $tex = <$fh>; }
29 print "% This file is auto-generated by 0.12.2\n";
30 print '% There are ', length($tex),
31 ' chars in the input: ', $ARGV[1], "\n";
32 print '% ---', "\n";
33 if (index($tex, "\t") > 0) {
34   print "TABS are prohibited!";
35   exit 1;
36 }
37 my @lines = split /\n/g, $tex;
38 foreach my $t (@lines) {
39   print '% ', $t, "\n";
40 }
41 print '% ---', "\n";
42 $tex =~ s/(?<!\n)%.*\n/\n/g;
```

```

43 $tex =~ s/^~\s+|\s+$//g;
44 my $splitting = $tex =~ /^~\\begin\{split\}/;
45 if ($splitting) {
46   print '% The manual splitting mode is ON since \begin{split} started the text' . "\n";
47 }
48 my $indents = $tex =~ /\n +/g;
49 my $gathered = (0 == $indents);
50 if ($gathered) {
51   if ($splitting) {
52     print '% The "gathered" is NOT used because of manual splitting' . "\n";
53     $gathered = 0;
54   } else {
55     print '% The "gathered" is used since all lines are left-aligned' . "\n";
56   }
57 } else {
58   print '% The "gathered" is NOT used because ' .
59     $indents . " lines are indented\n";
60 }
61 my $align = 0;
62 print '% The "align" is NOT used by default' . "\n";
63 if (index($tex, '&&') >= 0) {
64   $macro =~ s/equation/align/g;
65   $align = 1;
66   print '% The "align" is used because of && seen in the text' . "\n";
67 }
68 if ($macro ne 'phiq') {
69   if (not $splitting) {
70     $tex =~ s/\\\\\\n/n\n/g;
71     $tex =~ s/\\\\n\s*//g;
72   }
73   $tex =~ s/\n*(\\label\{[^\}]+\})\n*/\1/g;
74   $tex =~ s/\n{3,}/\n\n/g;
75 }
76 my @texts = ();
77 sub trep {
78   my ($s) = @_ ;
79   my $open = 0;
80   my $p = 0;
81   for (; $p < length($s); $p++) {
82     $c = substr($s, $p, 1);
83     if ($c eq '}') {
84       if ($open eq 0) {
85         last;
86       }
87       $open--;
88     }
89     if ($c eq '{') {
90       $open++;
91     }
92   }
93   push(@texts, substr($s, 0, $p));
94   return '{TEXT' . (0+@texts - 1) . '}' . substr($s, $p + 1);
95 }
96 $tex =~ s/\\text\{(.+)/trep("$1")/ge;

```

```

97 if (not $splitting) {
98 $tex =~ s/(?<![{&}&(?![&])/\sigma{}/g;
99 }
100 $tex =~ s/([^\{a-z0-9]|^)Q(?![a-z0-9])/\1\Phi{}/g;
101 $tex =~ s/([^\{a-z0-9]|^)D>/\1\Delta{}/g;
102 $tex =~ s/([^\{a-z0-9]|^)L>/\1\lambda{}/g;
103 $tex =~ s/"([~"]+)"|"\1"/g;
104 $tex =~ s/([~|(?<=[\s](\[\,.\>\/]))([a-zA-Z][a-z0-9]+)(?=[\s](\[\,.\>\/)))/\2/g;
105 $tex =~ s/([~_]|^)([0-9]+|\*)\/\([a-z]+|\)[a-z]+|\)/g;
106 (->|\.\.\>|^>|:=|!->)/\1\alpha_{\2}\vert{\3}\space{\4}/g;
107 $tex =~ s/([~_]|^)([0-9]+|\*)/g;
108 (->|\.\.\>|^>|:=|!->)/\1\alpha_{\2}\space{\3}/g;
109 if ($macro ne 'phiq') {
110   if (not $splitting) {
111     $tex =~ s/\begin{\split}\n\\begin{split}&/g;
112     $tex =~ s/\n\s*\end{\split}\n\\end{split}&/g;
113     $tex =~ s/\n\n\\&/g;
114     $tex =~ s/\n\\phiEOL{}/n&/g;
115     $tex =~ s/\\&/g;
116     $tex =~ s/\\&/n/g;
117     $tex =~ s/([~&s])\s{2}([~&s])/\1 \2/g;
118     $tex =~ s/\s{2}/ \quad/g;
119     $tex = '&' . $tex;
120   }
121   my $lead = '[~&s]+\s(?:->|:=|!>)\s';
122   my @leads = $tex =~ /&{$lead}/g;
123   my @eols = $tex =~ /\n/g;
124   if (0+@leads == 0+@eols && 0+@eols > 1) {
125     $tex =~ s/&{$lead})/\1~/g;
126     $gathered = 0;
127     print '% The "gathered" is NOT used because all ' .
128       (0+@eols) . ' lines are ' . (0+@leads) . " leads\n";
129   }
130 }
131 if ($macro ne 'phiq') {
132 sub strip_tabs {
133   my ($env, $tex) = @_;
134   $tex =~ s/&/g;
135   return "\\begin{$env}" . $tex . "\\end{$env}";
136 }
137 foreach my $e (('matrix', 'cases')) {
138 $tex =~ s/\begin{(\Q$e\E*?)\}(.\+)\end{(\Q$e\E*?)\}/strip_tabs($1, $2)/sge;
139 }
140 }
141 $tex =~ s/\$/\xi{}/g;
142 $tex =~ s/(?<!\{)\^(?!\\)\}/\rho{}/g;
143 $tex =~ s/[[/\llbracket\mathbin{}/g;
144 $tex =~ s/[\\]\}/\rrbracket{}/g;
145 $tex =~ s/([~>])([0-9A-F]{2})(?:-[0-9A-F]{2})+|
146   [0-9]+(?:\.[0-9]+)?)(?!\\)/\1\2/g;
147 $tex =~ s/TRUE/|TRUE|/g;
148 $tex =~ s/FALSE/|FALSE|/g;
149 $tex =~ s/\?/\varnothing{}/g;
150 $tex =~ s/@/\varphi{}/g;

```

```

151 $tex =~ s/-([a-z]+)>/\mathrel{\phiSlot{1}}/g;
152 $tex =~ s/!->/\mathbin{\phiConst}/g;
153 $tex =~ s/->/\mathbin{\mapsto}/g;
154 $tex =~ s/~>/\mathbin{\phiWave}/g;
155 $tex =~ s/:=/\mathrel{\vDash}/g;
156 $tex =~ s/==/\mathrel{\equiv}/g;
157 $tex =~ s/\.\.\>/\mathbin{\phiDotted}/g;
158 $tex =~ s/<</\langle/g;
159 $tex =~ s/>>/\rangle/g;
160 $tex =~ s/|{2,}/|/g;
161 $tex =~ s/|([~|]+)|/\textnormal{\texttt{1}}{/g;
162 $tex =~ s/{TEXT(d+)}'/\text{'. @texts[1] . '}'/ge;
163 if ($macro eq 'phiq') {
164   print '$' if ($tex ne '');
165 } else {
166   print '\begin{', $macro, "}\n";
167   if (not($align)) {
168     if ($gathered) {
169       print '\begin{gathered}' . "\n";
170     } elsif (not $splitting) {
171       print '\begin{split}' . "\n";
172     }
173   }
174 }
175 if ($gathered and not($align)) {
176   $tex =~ s/~&/g;
177   $tex =~ s/\n&/\n/g;
178 }
179 print $tex;
180 if ($macro eq 'phiq') {
181   print '$' if ($tex ne '');
182 } else {
183   if (not($align)) {
184     if ($gathered) {
185       print "\n" . '\end{gathered}';
186     } elsif (not $splitting) {
187       print "\n" . '\end{split}';
188     }
189   }
190   print "\n" . '\end{' . $macro . '}'';
191 }
192 print '\endinput';
193 \end{VerbatimOut}
194 \message{eolang: File with Perl script
195   '\eolang@tmpdir/eolang-phi.pl' saved^^J}%
196 \makeatother

```

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

```

197 \makeatletter
198 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
199 \makeatother

```

`phiuation` Then, we define the `phiuation` and the `phiuation*` environments through a sup-

plementary \eolang@process command:

```

200 \makeatletter\newcommand\eolang@process[1]{
201   \def\hash{\eolang@mdfive
202     {\eolang@tmpdir/\jobname/phiuation.tex}}%
203   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiuation.tex"
204     "\eolang@tmpdir/\jobname/\hash.tex"}%
205   \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
206   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
207     perl "\eolang@tmpdir/eolang-phi.pl"
208     '#1'
209     "\eolang@tmpdir/\jobname/\hash.tex"
210     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\\n|$)//g'\fi
211     \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
212   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
213   \def\eolang@phiSaveTo{\relax}%
214 }
215 %
216 \newenvironment{phiuation*}%
217 {\catcode'\|=12 \VerbatimEnvironment%
218 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
219 \begin{VerbatimOut}
220   {\eolang@tmpdir/\jobname/phiuation.tex}}
221 {\end{VerbatimOut}\eolang@process{equation*}}
222 %
223 \newenvironment{phiuation}%
224 {\catcode'\|=12 \VerbatimEnvironment%
225 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
226 \begin{VerbatimOut}
227   {\eolang@tmpdir/\jobname/phiuation.tex}}
228 {\end{VerbatimOut}\eolang@process{equation}}
229 \makeatother

```

\phiq Then, we define \phiq command:

```

230 \RequirePackage{xstring}
231 \makeatletter\newcommand\phiq[1]{%
232   \StrSubstitute{\detokenize{#1}}{' '}{' '}{\clean}%
233   \iexec[log,trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{
234     /bin/echo '\clean'}%
235   \def\hash{\eolang@mdfive
236     {\eolang@tmpdir/\jobname/phiq.tex}}%
237   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiq.tex"
238     "\eolang@tmpdir/\jobname/\hash.tex"}%
239   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
240   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
241     perl \eolang@tmpdir/eolang-phi.pl '\phiq'
242     "\eolang@tmpdir/\jobname/\hash.tex"
243     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\\n|$)//g'\fi}%
244   \ifdefined\eolang@nodollar\else\catcode'\$=\active\fi%
245 } \makeatother

```

nodollar Then, we redefine dollar sign:

```

246 \ifdefined\eolang@nodollar\else
247   \begingroup
248   \catcode'\$=\active

```

```

249 \protected\gdef$#1${\phiq{#1}}
250 \endgroup
251 \AtBeginDocument{\catcode'\$=\active}
252 \fi

```

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

```

253 \makeatletter
254 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
255 sub num {
256   my ($i) = @_ ;
257   $i =~ s/(\+|-)\./\10./g;
258   return $i;
259 }
260 sub fmt {
261   my ($tex) = @_ ;
262   $tex =~ s/|([^\|]+)|\\textnormal{\texttt{\1}}/g;
263   return $tex;
264 }
265 sub vertex {
266   my ($v) = @_ ;
267   if (index($v, 'v0') == 0) {
268     return '\Phi';
269   } else {
270     $v =~ s/^v/v_/g;
271     $v =~ s/[~0-9]$//g;
272     return $v . ' ';
273   }
274 }
275 sub tailor {
276   my ($t, $m) = @_ ;
277   $t =~ s/<([A-Z]?${m}[A-Z]?):([~>+)>/\2/g;
278   $t =~ s/<[A-Z]+:[~>+>/\2/g;
279   return $t;
280 }
281 open(my $fh, '<', $ARGV[0]);
282 my $tex; { local $/; $tex = <$fh> }
283 if (index($tex, "\t") > 0) {
284   print "TABS are prohibited!";
285   exit 1;
286 }
287 print '% This file is auto-generated', "\n\n";
288 print '% --- there are ', length($tex),
289   ' chars in the input (', $ARGV[0], "):\n";
290 foreach my $t (split (/~n/g, $tex)) {
291   print '% ', $t, "\n";
292 }
293 print "% ---\n";
294 $tex =~ s/\\~n/g;
295 $tex =~ s/\\~n//g;
296 $tex =~ s/([a-zA-Z]+)\s+/\1/g;
297 $tex =~ s/~n{2,}/~n/g;
298 my @cmds = split(/~n/g, $tex);
299 print '% --- before processing:' . "\n";

```

```

300 foreach my $t (split (/\\n/g, $tex)) {
301   print '% ', $t, "\\n";
302 }
303 print '% ---';
304 print ' (' . (0+@cmds) . " lines)\\n";
305 print '\\begin{picture}', "\\n";
306 for (my $c = 0; $c < 0+@cmds; $c++) {
307   my $cmd = $cmds[$c];
308   $cmd =~ s/^\\s+//g;
309   $cmd =~ s/(?!\\)%.*//g;
310   my ($head, $tail) = split(/ /, $cmd, 2);
311   my %opts = {};
312   foreach my $p (split(/ /, $tail)) {
313     my ($q, $t) = split(/:/, $p);
314     $opts{$q} = $t;
315   }
316   if (index($head, '->') >= 0) {
317     my $draw = '\\draw[';
318     if (exists $opts{'pi'}) {
319       $draw = $draw . '<MB:phi-pi><F:draw=none>';
320       if (not exists $opts{'a'}) {
321         $opts{'a'} = '\\pi';
322       }
323     }
324     if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
325       $draw = $draw . '<MB:,phi-rho>';
326     }
327     $draw = $draw . ']';
328     my ($from, $to) = split (/->/, $head);
329     $draw = $draw . " (${$from}) ";
330     if (exists $opts{'bend'}) {
331       $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
332         num($opts{'bend'}) . '>';
333       if (exists $opts{'rho'}) {
334         $draw = $draw . '<MB:,phi-rho>';
335       }
336       $draw = $draw . ']';
337     } else {
338       $draw = $draw . '--';
339     }
340     if (exists $opts{'a'}) {
341       my $a = $opts{'a'};
342       if (index($a, '$') == -1) {
343         $a = '$' . fmt($a) . '$';
344       } else {
345         $a = fmt($a);
346       }
347       $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
348     }
349     if (exists $opts{'break'}) {
350       $draw = $draw . '<F: coordinate [pos=' .
351         ($opts{'break'} / 100) . '] (break)>';
352     }
353     $draw = $draw . " (<MF:${to}><B:break-v>)";

```



```

354   if (exists $opts{'break'}) {
355       print tailor($draw, 'F') . ";\n";
356       print ' \node[outer sep=.1cm,inner sep=0cm] ' .
357           'at (break) (break-v) {$' . vertex($to) .
358           '$};' . "\n";
359       print ' ' . tailor($draw, 'B');
360   } else {
361       print tailor($draw, 'M');
362   }
363 } elsif (index($head, '=') >= 0) {
364     my ($from, $to) = split (/=>/, $head);
365     my $size = () = $head =~ /=/g;
366     if ($from eq '') {
367         print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
368             $to . '.center]';
369     } elsif ($to eq '') {
370         print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
371             $from . '.center]';
372     } else {
373         print '\node [phi-arrow] at ($(' .
374             $from . ')!0.5!(' . $to . ')$)';
375     }
376     print '{}';
377 } elsif (index($head, '!') >= 0) {
378     my ($v, $marker) = split (/!+/, $head);
379     my $size = () = $head =~ /*!/g;
380     print '\node [phi-marker, left=' .
381         ($size * 0.6) . 'cm of ' .
382         $v . '.center]{' . fmt($marker) . '}';
383 } elsif (index($head, '+') >= 0) {
384     my ($v, $suffix) = split (/+/, $head);
385     my @friends = ($v);
386     foreach my $c (@cmds) {
387         $e = $c;
388         $e =~ s/^\s+//g;
389         my $h = $e;
390         $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
391         foreach my $f (@friends) {
392             my $add = '';
393             if (index($h, $f . '->') >= 0) {
394                 $add = substr($h, index($h, '->') + 2);
395             }
396             if ($h =~ /\->\Q${f}\E$/) {
397                 $add = substr($h, 0, index($h, '->'));
398             }
399             if (index($e, ' xy:' . $f . ',') >= 0) {
400                 $add = $h;
401             }
402             if (index($add, '+') == -1
403                 and $add ne ''
404                 and not(grep(/^\Q${add}\E$/, @friends))) {
405                 push(@friends, $add);
406             }
407         }

```

```

408 }
409 my @extra = ();
410 foreach my $e (@cmds) {
411     $m = $e;
412     if ($m =~ /\s*\Q${v}\E\s/) {
413         next;
414     }
415     if ($m =~ /\s*[^\s]+\s/ and not($m =~ /\s*\Q${head}\E\s/)) {
416         next;
417     }
418     foreach my $f (@friends) {
419         my $h = $f;
420         $h =~ s/[a-z]$/g;
421         if ($m =~ s/^(\\s*)\Q${f}\E+\Q${suffix}\E\s?/\1${h}${suffix} /g) {
422             last;
423         }
424         $m =~ s/^(\\s*)\Q${f}\E\s/\1${h}${suffix} /g;
425         $m =~ s/^(\\s*)\Q${f}\E->/\1${h}${suffix}->/g;
426         $m =~ s/\\sxy:\Q${f}\E,/ xy:${h}${suffix},/g;
427         $m =~ s/->\Q${f}\E\s/->${h}${suffix} /g;
428     }
429     if ($m ne $e) {
430         push(@extra, ' ' . $m);
431     }
432 }
433 splice(@extra, 0, 0, @extra[-1]);
434 splice(@extra, -1, 1);
435 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
436     '), friends: [' . join(', ', @friends) . ']' in ' .
437     (0+@cmds) . ' lines');
438 splice(@cmds, $c, 1, @extra);
439 print '% cloned ' . $v . ' at line no.' . $c .
440     ' (+ ' . (0+@extra) . ' lines -> ' .
441     (0+@cmds) . ' lines total)';
442 } elsif ($head =~ /\s*[0-9]+[a-z]?$/) {
443     print '\node[';
444     if (exists $opts{'xy'}) {
445         my ($v, $right, $down) = split(/,/ , $opts{'xy'});
446         my $loc = '';
447         if ($down > 0) {
448             $loc = 'below ';
449         } elsif ($down < 0) {
450             $loc = 'above ';
451         }
452         if ($right > 0) {
453             $loc = $loc . 'right';
454         } elsif ($right < 0) {
455             $loc = $loc . 'left';
456         }
457         print ', ' . $loc . '=';
458         print abs(num($down)) . 'cm and ' .
459             abs(num($right)) . 'cm of ' . $v . '.center';
460     }
461     if (exists $opts{'data'}) {

```

```

462     print ',phi-data';
463     if ($opts{'data'} ne '') {
464         my $d = $opts{'data'};
465         if (index($d, '|') == -1) {
466             $d = '$\Delta\phiDotted\text{' .
467                 '\textnormal{\texttt{' . fmt($d) . '}}}$';
468         } else {
469             $d = fmt($d);
470         }
471         $opts{'box'} = $d;
472     }
473 } elsif (exists $opts{'atom'}) {
474     print ',phi-atom';
475     if ($opts{'atom'} ne '') {
476         my $a = $opts{'atom'};
477         if (index($a, '$') == -1) {
478             $a = '$\lambda\phiDotted{' . fmt($a) . '$';
479         } else {
480             $a = fmt($a);
481         }
482         $opts{'box'} = $a;
483     }
484 } else {
485     print ',phi-object';
486 }
487 print ']';
488 print ' (' . $head . ')';
489 print '{';
490 if (exists $opts{'tag'}) {
491     my $t = $opts{'tag'};
492     if (index($t, '$') == -1) {
493         $t = '$' . $t . '$';
494     } else {
495         $t = fmt($t);
496     }
497     print $t;
498 } else {
499     print '$' . vertex($head) . '$';
500 }
501 print '}';
502 if (exists $opts{'box'}) {
503     print ' node[phi-box] at (';
504     print $head, '.south east) {';
505     print $opts{'box'}, ')';
506 }
507 } else {
508     print $cmd;
509 }
510 print ";\n";
511 }
512 print '\end{picture}%', "\n";
513 print "% --- after processing:\n%";
514 foreach my $c (@cmds) {
515     print '% ', $c, "\n";

```

```

516 }
517 print '% --- (' . (0+@cmds) . " lines)\n";
518 print '\endinput';
519 \end{VerbatimOut}
520 \message{eolang: File with Perl script
521   '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
522 \makeatother

```

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

```

523 \setcounter{FancyVerbLine}{0}

```

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

```

524 \RequirePackage{tikz}
525 \usetikzlibrary{arrows}
526 \usetikzlibrary{shapes}
527 \usetikzlibrary{decorations}
528 \usetikzlibrary{decorations.pathmorphing}
529 \usetikzlibrary{decorations.pathreplacing}
530 \usetikzlibrary{positioning}
531 \usetikzlibrary{calc}
532 \usetikzlibrary{math}
533 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment picture:

```

534 \newenvironment{picture}%
535   {\noindent\begin{tikzpicture}[
536     ->,>=stealth',node distance=0,thick,
537     pics/parallel arrow/.style={
538       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}]}%
539   {\end{tikzpicture}}
540 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
541   minimum height=0.5cm, minimum width=0.5cm,
542   single arrow head extend=2mm]
543 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
544   minimum width=1.4em, font={\small\color{white}\ttfamily},
545   fill=gray]
546 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
547   draw,font={\small}]
548 \tikzstyle{phi-object} = [phi-thing,circle]
549 \tikzstyle{phi-data} = [phi-thing,regular polygon,
550   regular polygon sides=8]
551 \tikzstyle{phi-empty} = [phi-object]
552 \tikzset{%
553   phi-rho/.style={
554     postaction={%
555       decoration={
556         show path construction,
557         curveto code={
558           \tikzmath{
559             coordinate \I, \F, \v;
560             \I = (\tikzinputsegmentfirst);
561             \F = (\tikzinputsegmentlast);
562             \v = ($(\I) -(\F)$);

```

```

563         real \d, \a, \r, \t;
564         \d = 0.8;
565         \t = atan2(\vy, \vx);
566         if \vx<0 then { \a = 90; } else { \a = -90; };
567         {
568             \draw[arrows={-latex}, decorate,
569                 decoration={%
570                     snake, amplitude=.4mm,
571                     segment length=2mm,
572                     post length=1mm
573                 }]
574             ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
575             -- ++(\t: 2*\d em);
576         };
577     }
578 },
579 \lineto code={
580     \tikzmath{
581         coordinate \I, \F, \v;
582         \I = (\tikzinputsegmentfirst);
583         \F = (\tikzinputsegmentlast);
584         \v = ($(\I) -(\F)$);
585         real \d, \a, \r, \t;
586         \d = 0.8;
587         \t = atan2(\vy, \vx);
588         if \vx<0 then { \a = 90; } else { \a = -90; };
589         {
590             \draw[arrows={-latex}, decorate,
591                 decoration={%
592                     snake, amplitude=.4mm,
593                     segment length=2mm,
594                     post length=1mm}]
595             ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
596             -- ++(\t: 2*\d em);
597         };
598     }
599 }
600 },
601 \decorate
602 }
603 }
604 }
605 \tikzstyle{phi-pi} = [draw,dotted]
606 \tikzstyle{phi-atom} = [phi-object,double]
607 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
608     rectangle,thin,minimum width=1.2em,anchor=north west,
609     font={\scriptsize}]
610 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
611     above=2pt,sloped/.append style={transform shape},
612     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:

```

613 \makeatletter

```

```

614 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
615 \makeatother

```

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

```

616 \makeatletter\newenvironment{sodg}%
617 {\catcode'\|=12 \VerbatimEnvironment%
618 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
619 \begin{VerbatimOut}
620   {\eolang@tmpdir/\jobname/sodg.tex}}
621 {\end{VerbatimOut}}%
622 \def\hash{\eolang@mdfive
623   {\eolang@tmpdir/\jobname/sodg.tex}}%
624 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
625   "\eolang@tmpdir/\jobname/\hash.tex"}%
626 \catcode'\$=3 %
627 \message{Start parsing 'sodg' at line no. \the\inputlineno^^J}
628 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
629   perl "\eolang@tmpdir/eolang-sodg.pl"
630   "\eolang@tmpdir/\jobname/\hash.tex"
631   \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
632   \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%
633 \catcode'\$=active%
634 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
635 \def\eolang@sodgSaveTo{\relax}%
636 }\makeatother

```

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

```

637 \RequirePackage{hyperref}
638 \pdfstringdefDisableCommands{
639   \def\({}%
640   \def\)}%
641   \def\alpha{alpha}%
642   \def\varphi{phi}%
643 }
644 \makeatletter
645 \NewExpandableDocumentCommand{\eoAnon}{O{ANONYMIZED}m}{%
646   \ifdefined\eolang@anonymous%
647     \textcolor{orange}{#1}%
648   \else%
649     #2%
650   \fi%
651 }\makeatother

```

\eolang Then, we define a simple supplementary command to help you print EO, the name of our language.

```

652 \newcommand\eolang{%
653   \eoAnon[XYZ]{\sffamily EO}}

```

\phic Then, we define a simple supplementary command to help you print  $\varphi$ -calculus, the name of our formal apparatus.

```

654 \newcommand\phic{%
655   \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.

```

656 \newcommand\xmir{%
657   \eoAnon[XML\(^+\)]{XMIR}}

```

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

```

658 \newcommand\phiConst{%
659   \mathrel{\hspace{.15em}}%
660   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

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

```

661 \newcommand\phiWave{%
662   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

```

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

```

663 \newcommand\phiSlot[1]{%
664   \xrightarrow{\text{\sffamily\scshape #1}}}

```

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

```

665 \makeatletter
666 \newcommand{\phiOset}[2]{%
667   \mathrel{\mathop{#2}\limits^{
668     \vbox to 0ex{\kern-2\ex@
669       \hbox{$\scriptscriptstyle#1$}\vss}}}}
670 \newcommand{\phiUset}[2]{%
671   \mathrel{\mathop{#2}\limits_{
672     \vbox to 0ex{\kern-6.3\ex@
673       \hbox{$\scriptscriptstyle#1$}\vss}}}}
674 \makeatother

```

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

```

675 \newcommand\phiMany[3]{%
676   \phiOset{#3}{\phiUset{#2}{#1}}}

```

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

```

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

```

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

```

678 \RequirePackage{trimclip}
679 \RequirePackage{amsfonts}
680 \makeatletter
681 \newcommand{\phiDotted}{%
682   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
683 \newcommand{\phiDotted@}[2]{%
684   \begingroup%
685   \settowidth{\dimen\z@}{$\m@th#1\rightharpoonup$}%
686   \settoheight{\dimen\tw@}{$\m@th#1\rightharpoonup$}%
687   \sbox\z@{%
688     \makebox[\dimen\z@][s]{%

```

```

689     \clipbox{0 0 {0.4\width} 0}%
690     {\resizebox{\dimen\z@}{\height}%
691      {\m@th#1\dashrightarrow$}}%
692     \hss%
693     \clipbox{{0.69\width} {-0.1\height} 0
694      {-\height}}{\m@th#1\rightarrow$}%
695   }%
696 }%
697 \ht\z@=\dimen\tw@ \dp\z@=\z@%
698 \box\z@%
699 \endgroup%
700 }
701 \makeatother

```



## References

- Bugayenko, Yegor (2021). *EOLANG and  $\varphi$ -calculus*. arXiv: [2111.13384](#) [cs.PL].
- Kudasov, Nikolai et al. (2022).  *$\varphi$ -calculus: a purely object-oriented calculus of decorated objects*. arXiv: [2204.07454](#) [cs.PL].

## Change History

0.0.1	General: First draft. . . . .	9	0.12.1	<code>eolang-sodg.pl</code> : The bug is fixed related to the formatting of indexes of vertices. . . . .	15
0.0.2	<code>sodg</code> : The environment <code>phigure</code> renamed to <code>sodg</code> for the sake of better semantic. The graph in the picture is solely a SODG graph, that's why the name <code>sodg</code> is better. . . . .	22	0.2.0	<code>eolang-phi.pl</code> : Numbers automatically render as <code>\texttt</code> . No need to use vertical bars around them anymore. . . . .	10
	<code>eolang-phi.pl</code> : New symbol added for basket slots . . . . .	10		<code>eolang-sodg.pl</code> : The content of the <code>atom</code> and the <code>data</code> boxes is parsed automatically as formulas and numbers, respectively. . . . .	15
	Parsing of the symbols “@,” “^,” and “&” enabled ( <code>\varphi</code> , <code>\rho</code> , and <code>\sigma</code> ) . . . . .	10		<code>\xm</code> : New command <code>\xm</code> prints XMIR in both normal and the anonymous mode of <code>acmart</code> . . . . .	23
	The symbols “[” and “]” replaced with “[” and “]” for abstract object brackets, because they conflicted with normal square brackets . . . . .	10	0.3.0	<code>\eolang@lineno</code> : New counter for protecting <code>lineno</code> . . . . .	10
	<code>eolang-sodg.pl</code> : 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. . . . .	15		<code>eolang-phi.pl</code> : New arrow added, that looks like <code>\leadsto</code> . . . . .	10
	<code>\phi</code> : Parsing of additional symbols enabled. . . . .	14		<code>\phiWave</code> : New command <code>\phiWave</code> added to denote a link to a multi-layer attribute. . . . .	23
0.1.0	General: Parsing of package options introduced. . . . .	10	0.4.0	<code>eolang-sodg.pl</code> : Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed with the <code>\Delta</code> and the <code>\lambda</code> commands. . . . .	15
	<code>\eolang</code> : New command <code>\eolang</code> added to print the name of the language in both normal and the anonymous mode of <code>acmart</code> . . . . .	22		Relative positioning of vertices fixed. . . . .	15
	<code>\eolang@mdfive</code> : New supplementary command added to calculate MD5 sum of a file. . . . .	10	0.5.0	<code>eolang-phi.pl</code> : Automated formatting of <code>TRUE</code> and <code>FALSE</code> added. . . . .	10
	<code>eolang-phi.pl</code> : A new Perl script “ <code>eolang-phi.pl</code> ” added for parsing of <code>phi</code> expressions. . . . .	10		<code>eolang-sodg.pl</code> : It is possible to use TikZ commands inside the <code>sodg</code> environment. . . . .	15
	<code>eolang-sodg.pl</code> : There are two Perl scripts now: one for <code>phi</code> quation, another one for <code>sodg</code> . . . . .	15		New syntax introduced that allows to make clones of vertices and all their dependants. . . . .	15
	<code>\phi</code> : New command <code>\phi</code> prints the name of $\varphi$ -calculus in both normal and the anonymous mode of <code>acmart</code> . . . . .	22		Now edges may have the <code>break</code> attribute, to make them shorter. . . . .	15
	<code>\phiConst</code> : New command <code>\phiConst</code> added to denote a link to a constant attribute. . . . .	23		<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow. . . . .	23
	<code>\phiDotted</code> : New command <code>\phiDotted</code> added to denote a link to a special attribute. . . . .	23		<code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket. . . . .	23
			0.6.0	General: Package option <code>nocomments</code> added in order to enable comments	

suppression in temporary .tex files (may be pretty important for .dtx documents). . . . .	10	eolang-phi.pl: Inside phiquation any text inside the \text macro is not processed. . . . .	10
eolang-sodg.pl: The rrho attribute is retired, now rho works just fine in all situations. . . . .	15	eolang-sodg.pl: The tag attribute is introduced for changing labels inside a vertex circle. . . . .	15
0.7.0		\phi0set: New commands \phi0set and \phiUset help position text over and under an arrow. . . . .	23
nodollar: Now it is possible to use dollar sign instead of the \phiq command. . . . .	14	\phiSaveTo: The output of the phiquation environment can be redirected to a file. . . . .	13
eolang-phi.pl: New syntax sugar for $\Phi$ , just using capital “Q” is enough. . . . .	10	\sodgSaveTo: The output of the sodg environment can be redirected to a file. . . . .	21
Object names are automatically converted to \texttt, provided their names include two or more symbols. . . . .	10	0.9.0	
Text in quotes is automatically converted to \texttt. . . . .	10	\eoAnon: New command \eoAnon added. . . . .	22
0.8.0		eolang-phi.pl: Proper handling of the matrix environment. . . . .	10
General: The anonymous package option added. . . . .	10	\phiEOL: New command \phiEOL added, instead of \[-4pt]. . . . .	23

# 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	D	F
<code>\\$</code> ..... 141, 239, 244, 248, 251, 626, 633	<code>\d</code> .. 162, 563, 564, 574, 575, 585, 586, 595, 596	<code>\F</code> ..... 559, 561, 562, 574, 581, 583, 584, 595
<code>\%</code> ..... 210, 243, 631	<code>\dashrightarrow</code> ... 691	<code>\FancyVerbLine</code> .... <u>523</u>
<code>\(</code> ..... 639, 655, 657	<code>\def</code> ..... 198, 201, 213, 235, 614, 622, 635, 639, 640, 641, 642	<b>G</b>
<code>\)</code> ..... 640, 655, 657	<code>\Delta</code> ..... 466	<code>\gdef</code> ..... 249
<code>\*</code> ..... 105, 107, 138	<code>\detokenize</code> ..... 232	<b>H</b>
<code>\+</code> ..... 257, 384, 415, 421	<code>\dimen</code> 685, 686, 688, 690, 697	<code>\hash</code> ... 201, 204, 206, 209, 235, 238, 240, 242, 622, 625, 628, 630
<code>\-</code> ..... 655	<code>\dp</code> ..... 697	<code>\hbox</code> ..... 669, 673
<code>\.</code> .. 106, 108, 146, 157, 257	<code>\draw</code> ... 317, 538, 568, 590	<code>\height</code> .... 690, 693, 694
<code>\/</code> ..... 104, 105	<b>E</b>	<code>\hspace</code> ..... 659, 660
<code>\?</code> ..... 149	<code>\E</code> 138, 396, 404, 412, 415, 421, 424, 425, 426, 427	<code>\hss</code> ..... 692
<code>\[</code> ..... 104, 143	<code>\end</code> ..... 185, 187, 190, 193, 221, 228, 512, 519, 539, 621	<code>\ht</code> ..... 697
<code>\{</code> .... 44, 73, 96, 111, 112, 138, 142, 146, 162	<code>\endinginput</code> .... 192, 518	<b>I</b>
<code>\}</code> 44, 73, 111, 112, 138, 162	<code>\eoAnon</code> . <u>637</u> , 653, 655, 657	<code>\I</code> ..... 559, 560, 562, 574, 581, 582, 584, 595
<code>\]</code> ..... 104, 144	<code>\eolang</code> ..... <u>652</u>	<code>\iexec</code> ... 18, 203, 206, 233, 237, 240, 624, 628
<code>\^</code> ..... 142	<code>\eolang-phi.pl</code> ..... <u>24</u>	<code>\ifdefined</code> ..... 210, 211, 239, 243, 244, 246, 631, 632, 646
<code>\ </code> ..... 105, 160, 161, 217, 224, 262, 617	<code>\eolang-sodg.pl</code> ... <u>253</u>	<code>\ifluatex</code> ..... 12
<b>Numbers</b>	<code>\eolang@anonymous</code> 14, 646	<code>\ifxetex</code> ..... 12
<code>\2</code> 104, 106, 108, 117, 146, 277	<code>\eolang@lineno</code> ..... <u>19</u>	<code>\inputlineno</code> ... 205, 627
<code>\3</code> ..... 106, 108	<code>\eolang@mdfive</code> .... ... <u>20</u> , 201, 235, 622	<b>J</b>
<code>\4</code> ..... 106	<code>\eolang@nocomments</code> ... 13, 210, 243, 631	<code>\jobname</code> . 18, 202, 203, 204, 206, 209, 220, 227, 233, 236, 237, 238, 240, 242, 620, 623, 624, 625, 628, 630
<b>A</b>	<code>\eolang@nodollar</code> .. ..... 239, 244, 246	<b>K</b>
<code>\a</code> 563, 566, 574, 585, 588, 595	<code>\eolang@phiSaveTo</code> . ..... 198, 211, 213	<code>\kern</code> ..... 668, 672
<code>\active</code> . 244, 248, 251, 633	<code>\eolang@process</code> ... ..... 200, 221, 228	<b>L</b>
<code>\alpha</code> ..... 641, 655	<code>\eolang@sodgSaveTo</code> ..... 614, 632, 635	<code>\lambda</code> ..... 478
<code>\AtBeginDocument</code> .. 251	<code>\eolang@tmpdir</code> .... ..... 11, 18, 25, 195, 202, 203, 204, 206, 207, 209, 220, 227, 233, 236, 237, 238, 240, 241, 242, 254, 521, 620, 623, 624, 625, 628, 629, 630	<code>\leadsto</code> ..... 662
<b>B</b>	<code>\ex@</code> ..... 668, 672	<code>\limits</code> ..... 667, 671
<code>\Bbbk</code> ..... 3		<b>M</b>
<code>\begin</code> ..... 25, 46, 166, 169, 171, 219, 226, 254, 305, 535, 619		<code>\m@th</code> ... 685, 686, 691, 694
<code>\box</code> ..... 698		<code>\makeatletter</code> ..... ..... 19, 21, 24,
<b>C</b>		
<code>\catcode</code> ..... 217, 224, 239, 244, 248, 251, 617, 626, 633		
<code>\clean</code> ..... 232, 234		
<code>\clipbox</code> ..... 689, 693		
<code>\color</code> ..... 544		

197, 200, 231, 253, 613, 616, 644, 665, 680	\makeatother 19, 23, 196, 199, 229, 245, 522, 615, 636, 651, 674, 701	\makebox ..... 688	\mapsto ..... 660	\mapstochar . 660, 662, 682	\mathop ..... 667, 671	\mathpalette ..... 682	\mathrel ..... 659, 660, 662, 667, 671, 682	\message 194, 205, 520, 627	\mspace ..... 662													
<b>N</b>																						
\newcommand ..... 22, 198, 200, 231, 614, 652, 654, 656, 658, 661, 663, 666, 670, 675, 677, 681, 683	\newcounter ..... 19	\newenvironment ... ... 216, 223, 534, 616	\NewExpandableDocumentCommand ..... 645	\node ..... 356, 367, 370, 373, 380, 443	\nodollar ..... 246	\noindent ..... 535	<b>P</b>															
\pdf@filemdfivesum . 22	\pdfstringdefDisableCommands ..... 638	\pgfkeys ..... 9	\Phi ..... 268	\phic ..... 654	\phiConst ..... 658	\phicture ..... 534	\phiDotted . 466, 478, 678	\phiDotted@ .... 682, 683	\phiEOL ..... 677													
\phiMany ..... 675	\phiOset ..... 665, 676	\phiq ..... 230, 249	\phiquation ..... 200	\phiSaveTo ..... 197	\phiSlot ..... 663	\phiUset ..... 670, 676	\phiWave ..... 661	\pi ..... 321	\ProcessPgfPackageOptions ..... 17													
\protected ..... 249	<b>Q</b>				\Q 138, 396, 404, 412, 415, 421, 424, 425, 426, 427	<b>R</b>																
\relax .... 3, 213, 635, 682	\RequirePackage .. 1, 2, 3, 4, 5, 6, 7, 8, 20, 230, 524, 637, 678, 679	\resizebox ..... 690	\rightarrow . 685, 686, 694	<b>S</b>																		
\sbox ..... 687	\scriptscriptstyle ..... 669, 673	\scriptsize .... 609, 612	\scshape ..... 664	\setcounter .... 212, 218, 225, 523, 618, 634	\settoheight ..... 686	\settowidth ..... 685	\sffamily ..... 653, 664	\small ..... 544, 547	\sodg ..... 616													
\sodgSaveTo ..... 613	\StrSubstitute .... 232	\sxy ..... 426	<b>T</b>						<b>Z</b>													
\t 33, 283, 563, 565, 574, 575, 585, 587, 595, 596	\text ..... 466, 664	\textcolor ..... 647	\textnormal ..... 467	\texttt ..... 467	\the ..... 205, 627	\tikz ..... 524	\tikzinputsegmentfirst ..... 560, 582	\tikzinputsegmentlast ..... 561, 583	\tikzmath ..... 558, 580													
\tikzset ..... 552	\tikzstyle .... 540, 543, 546, 548, 549, 551, 605, 606, 607, 610	\ttfamily ..... 544	\tw@ ..... 686, 697	<b>U</b>																		
\usetikzlibrary ... 525, 526, 527, 528, 529, 530, 531, 532, 533	<b>V</b>																					
\v ..... 559, 562, 581, 584	\value 212, 218, 225, 618, 634	\varphi ..... 642, 655	\vbox ..... 668, 672	<b>VerbatimEnvironment</b> ..... 217, 224, 617																		
\vss ..... 669, 673	\vx ..... 565, 566, 587, 588	\vy ..... 565, 587	<b>W</b>																			
\width ..... 689, 693	<b>X</b>																					
\xmir ..... 656	\xrightarrow ..... 664	<b>Z</b>																				
\z@ 685, 687, 688, 690, 697, 698																						