



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

Yegor Bugayenko  
yegor256@gmail.com

2022-12-15, 0.9.0

**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:

<pre> app <math>\mapsto</math> [   <math>\rho \mapsto \xi.b.\rho^2, \alpha_0   t \rightsquigarrow \text{TRUE},</math>   <math>b \mapsto [\alpha_* \mapsto \text{fn}(56),</math>     <math>\varphi \mapsto \Phi.\text{hello.bye}(\xi),</math>     <math>\Delta \mapsto 01\text{-FE-C3}]</math>,   <math>x \mapsto [\lambda \mapsto \emptyset]</math>. </pre>	<pre> 1 \documentclass{article} 2 \pagestyle{empty} 3 \usepackage{eolang} 4 \begin{document} 5 \begin{phiquestion*} 6 app -&gt; [[ % it's abstract! 7   ^ !-&gt; \$.b.^{^2}, 0/t~&gt; TRUE, 8   b -&gt; [[ *-&gt; fn(56), 9     @ -&gt; Q.hello.bye(\$), 10    D&gt; 01-FE-C3 ]]],\ 11 x -&gt; [[ \lambda ..&gt; ? ]]. 12 \end{phiquestion*} 13 \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](https://github.com/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 “ $\rightsquigarrow$ ” (`\rightsquigarrow`),
- “!->” maps to “ $\vDash$ ” (`\vDash`),
- “. .>” maps to “ $\vdash$ ” (`\vdash`),
- “D>” maps to “ $\Delta \vdash$ ” (`\Delta \vdash`),
- “L>” maps to “ $\lambda \vdash$ ” (`\lambda \vdash`),
- “[[” maps to “ $\llbracket$ ” (`\llbracket`),
- “]]” maps to “ $\rrbracket$ ” (`\rrbracket`),
- “|abc|” maps to “abc” (`\texttt{abc}`).

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

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

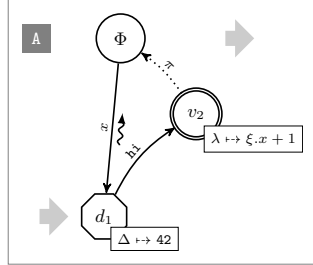
`\phiq` The command `\phiq` lets you inline a  $\varphi$ -calculus expressions using the same simple plain-text notation. You can use dollar sign directly too:

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><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=>”, 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 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, 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  $x \dashrightarrow y$  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` 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  $\llbracket \alpha_i \mapsto x_i \rrbracket$  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`     If you want to use `phiq` or `sodg` environments inside `tabular` or any other environment or command, you won't be able to do this, because `phiq` and `sodg` are “verbatim” environments. `\phiSaveTo` and `\sodgSaveTo` commands will help you in this situation. You use them right before `\begin{phiq}` 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{phiq}*
7 [[ x -> [[D>42]] ]]
8 \end{phiq}*
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 `phiquation` 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:

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

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

$\frac{\begin{array}{l} x \mapsto [\varphi \mapsto y, z \mapsto 42, \\ \alpha_0   g \mapsto \emptyset, \alpha_1   \text{foo} \mapsto 42 ] \end{array}}{x \mapsto [\varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto [\psi \rightsquigarrow \text{hello}(12)], \alpha_1 \mapsto 42)]} \text{R2.}$	<pre> 6 \begin{phiquation*} 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{\text{R2}.} 14 \end{phiquation*} </pre>
--	--

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

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

The `cases` environment works too:

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

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

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

```

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

```

It's possible to use `\label` inside the `phiqutation` 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{phiqutation}
9 D = b^{^2} - {4}ac.
10 \label{d}
11 \end{phiqutation}
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{phiqutation*}
7 [[ 0->x ]] && \text{This is formation}
8 [[ 0->? ]] && \text{Abstraction}
9 x(D>42) && \text{Application}
10 \end{phiqutation*}

```

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

$$\begin{array}{l}
x(\pi) \mapsto \llbracket \lambda \mapsto f_1 \rrbracket, \\
x(a, b, c) \mapsto \llbracket a_0 \mapsto \emptyset, \varphi \mapsto \text{hello}(\xi), x \mapsto \text{FALSE} \rrbracket, \\
\Delta = 43-09.
\end{array}$$

```

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

```

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

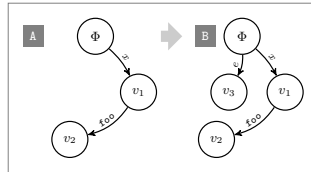
$$\begin{aligned} & \llbracket b \mapsto \emptyset \rrbracket, \\ & \llbracket \varphi \mapsto \text{TRUE}, \Delta \mapsto 42 \rrbracket, \\ & \psi = \langle \pi, 42 \rangle. \end{aligned}$$

```

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

```

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

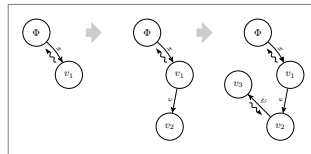


```

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:

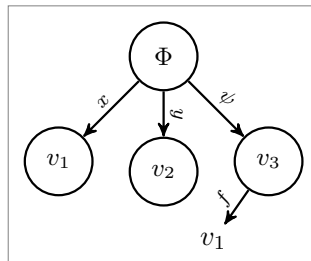


```

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:\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:



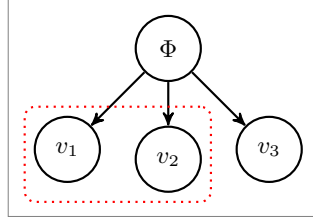
```

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 v1 xy:v0,-1,1 \\\ v0->v1
9 v2 xy:v0,0,1 \\\ v0->v2
10 v3 xy:v0,1,1 \\\ v0->v3
11 \node[draw=red,rounded corners,\
12 dotted,fit=(v1) (v2)] {};
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 \ProcessPgfpPackageOptions{/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 phiquation 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', "\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/g;
43 $tex =~ s/^\\s+|\\s+$//g;
44 my $gathered = (0 == $tex =~ /\\n\\s+/g);
45 if ($gathered) {
46   print '% The "gathered" is used since all lines are left-aligned' . "\n";
47 }
48 my $align = 0;
49 print '% The "align" is NOT used by default' . "\n";
50 if (index($tex, '&&') >= 0) {
51   $macro =~ s/equation/align/g;
52   $align = 1;
53   print '% The "align" is used because of && seen in the text' . "\n";
54 }
55 if ($macro ne 'phiq') {
56   $tex =~ s/\\\\\\n\\n\\n/g;
57   $tex =~ s/\\\\\\n\\s*/g;
58   $tex =~ s/\\n*(\\label\\{[~\\}]+\\})\\n*/\\1/g;
59   $tex =~ s/\\n{3,}/\\n\\n/g;
60 }
61 my @texts = ();
62 sub trep {
63   my ($s) = @_ ;
64   my $open = 0;
65   my $p = 0;
66   for (; $p < length($s); $p++) {
67     $c = substr($s, $p, 1);
68     if ($c eq '}') {
69       if ($open eq 0) {
70         last;
71       }
72       $open--;
73     }
74     if ($c eq '{') {
75       $open++;

```

```

76 }
77 }
78 push(@texts, substr($s, 0, $p));
79 return '{TEXT' . (0+@texts - 1) . '}' . substr($s, $p + 1);
80 }
81 $tex =~ s/\\text\{(.+)/trep("$1")/ge;
82 $tex =~ s/(?<![{&}]&(?![&}])\\/\\sigma{/g;
83 $tex =~ s/(\\[a-z0-9]|~)Q(?![a-z0-9])/\\1\\Phi{/g;
84 $tex =~ s/(\\[a-z0-9]|~)D>/\\1\\Delta{...}/g;
85 $tex =~ s/(\\[a-z0-9]|~)L>/\\1\\lambda{...}/g;
86 $tex =~ s/"([~]+)"|"1"/g;
87 $tex =~ s/(~|(?=<=[s](\\[, >,)/))([a-z][a-z0-9]+)(?=[s](\\[, -.]|$)/|\\2/g;
88 $tex =~ s/(~~_|~)([0-9]+|\\*)\\\\/\\\\?[a-z]+|\\|[a-z]+|\\|
89 (->|\\.\\.\\.>|>|:=|!->)/\\1\\alpha_{2}\\vert{}\\3\\space{}\\4/xg;
90 $tex =~ s/(~~_|~)([0-9]+|\\*)
91 (->|\\.\\.\\.>|>|:=|!->)/\\1\\alpha_{2}\\space{}\\3/xg;
92 if ($macro ne 'phiq') {
93   $tex =~ s/\\begin\\{split\\}\\n\\begin\\{split\\}&/g;
94   $tex =~ s/\\n\\s*\\end\\{split\\}\\n\\end\\{split\\}/g;
95   $tex =~ s/\\n\\n\\n\\n\\n&/g;
96   $tex =~ s/\\n\\n\\n\\phiEOL{\\}n&/g;
97   $tex =~ s/\\\\\\\\$/g;
98   $tex =~ s/\\\\\\\\\\\\\\\\n/g;
99   $tex =~ s/(\\[&s]\\s{2}(\\[~s])/\\1 \\2/g;
100  $tex =~ s/\\s{2}/ \\quad{/g;
101  $tex = '&' . $tex;
102  my $lead = '[~s]+s(?:->|:=|=)';
103  my @leads = $tex =~ /&{$lead}/g;
104  my @eols = $tex =~ /&/g;
105  if (0+@leads == 0+@eols && 0+@eols > 1) {
106    $tex =~ s/&({$lead})/\\1&/g;
107    $gathered = 0;
108    print '% The "gathered" is NOT used because all ' .
109      (0+@eols) . ' lines are ' . (0+@leads) . " leads\\n";
110  }
111 }
112 if ($macro ne 'phiq') {
113 sub strip_tabs {
114   my ($env, $tex) = @_ ;
115   $tex =~ s/&/g;
116   return "\\begin{$env}" . $tex . "\\end{$env}";
117 }
118 foreach my $e (('matrix', 'cases')) {
119 $tex =~ s/\\begin\\{(\\Q$e\\E\\*?)\\}\\.(+)\\end\\{(\\Q$e\\E\\*?)\\}/strip_tabs($1, $2)/sge;
120 }
121 }
122 $tex =~ s/\\$/\\xi{/g;
123 $tex =~ s/(?<![{&}]|^\\/\\rho{/g;
124 $tex =~ s/[\\[\\llbracket\\mathrel{/g;
125 $tex =~ s/[\\]\\rrbracket\\mathrel{/g;
126 $tex =~ s/(\\[s,>()([0-9A-F]{2})(?:-[0-9A-F]{2})+|
127 [0-9]+(?:\\. [0-9]+)?)(?!\\{)/\\1|\\2|xg;
128 $tex =~ s/TRUE/|TRUE|/g;
129 $tex =~ s/FALSE/|FALSE|/g;

```

```

130 $tex =~ s/\?/\varnothing{/g;
131 $tex =~ s/@/\varphi{/g;
132 $tex =~ s/-([a-z]+)>/\mathrel{\phiSlot{1}}/g;
133 $tex =~ s/!->/\mathrel{\phiConst}/g;
134 $tex =~ s/->/\mathrel{\mapsto}/g;
135 $tex =~ s/~>/\mathrel{\phiWave}/g;
136 $tex =~ s/:=\mathrel{\vDash}/g;
137 $tex =~ s/\.\.\>/\mathrel{\phiDotted}/g;
138 $tex =~ s/<</\langle/g;
139 $tex =~ s/>>/\rangle/g;
140 $tex =~ s/|{2,}/|/g;
141 $tex =~ s/|([~|]+)|/\textnormal{\texttt{1}}{/g;
142 $tex =~ s/{TEXT(d+)}'/\text{' . @texts[$1] . '}'/ge;
143 if ($macro eq 'phiq') {
144   print '$' if ($tex ne '');
145 } else {
146   print '\begin{' , $macro, "}\n";
147   if (not($align)) {
148     if ($gathered) {
149       print '\begin{gathered}';
150     } else {
151       print '\begin{split}';
152     }
153     print "\n";
154   }
155 }
156 if ($gathered and not($align)) {
157   $tex =~ s/^&/g;
158   $tex =~ s/\n&/\n/g;
159 }
160 print $tex;
161 if ($macro eq 'phiq') {
162   print '$' if ($tex ne '');
163 } else {
164   if (not($align)) {
165     print "\n";
166     if ($gathered) {
167       print '\end{gathered}';
168     } else {
169       print '\end{split}';
170     }
171   }
172   print "\n" . '\end{' . $macro . '}' ;
173 }
174 print '\endinput';
175 \end{VerbatimOut}
176 \message{eolang: File with Perl script
177   '\eolang@tmpdir/eolang-phi.pl' saved^^J}%
178 \makeatother

```

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

```

179 \makeatletter
180 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}

```

181 \makeatother

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

```

182 \makeatletter\newcommand\eolang@process[1]{
183   \def\hash{\eolang@mdfive
184     {\eolang@tmpdir/\jobname/phiquation.tex}}%
185   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiquation.tex"
186     "\eolang@tmpdir/\jobname/\hash.tex"}%
187   \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
188   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
189     perl "\eolang@tmpdir/eolang-phi.pl"
190     '#1'
191     "\eolang@tmpdir/\jobname/\hash.tex"
192     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\\n|$)//g'\fi
193     \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
194   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
195   \def\eolang@phiSaveTo{\relax}%
196 }
197 \newenvironment{phiquation*}%
198 {\catcode'\|=12 \VerbatimEnvironment%
199 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
200 \begin{VerbatimOut}
201   {\eolang@tmpdir/\jobname/phiquation.tex}}
202 {\end{VerbatimOut}\eolang@process{equation*}}
203 \newenvironment{phiquation}%
204 {\catcode'\|=12 \VerbatimEnvironment%
205 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
206 \begin{VerbatimOut}
207   {\eolang@tmpdir/\jobname/phiquation.tex}}
208 {\end{VerbatimOut}\eolang@process{equation}}
209 \makeatother

```

\phiq Then, we define \phiq command:

```

210 \makeatletter\newcommand\phiq[1]{%
211   \iexec[trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{
212     /bin/echo '\detokenize{#1}'}%
213   \def\hash{\eolang@mdfive
214     {\eolang@tmpdir/\jobname/phiq.tex}}%
215   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiq.tex"
216     "\eolang@tmpdir/\jobname/\hash.tex"}%
217   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
218   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
219     perl \eolang@tmpdir/eolang-phi.pl 'phiq'
220     "\eolang@tmpdir/\jobname/\hash.tex"
221     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\\n|$)//g'\fi}%
222   \ifdefined\eolang@nodollar\else\catcode'\$=active\fi%
223 }\makeatother

```

nodollar Then, we redefine dollar sign:

```

224 \ifdefined\eolang@nodollar\else
225   \begingroup
226   \catcode'\$=\active
227   \protected\gdef$#1${\phiq{#1}}

```

```

228 \endgroup
229 \AtBeginDocument{\catcode'\$=\active}
230 \fi

```

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

```

231 \makeatletter
232 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
233 sub num {
234   my ($i) = @_;
235   $i =~ s/(\+|-)\./\10./g;
236   return $i;
237 }
238 sub fmt {
239   my ($tex) = @_;
240   $tex =~ s/\\|([^\|]+)\\|\\textnormal{\\texttt{\1}}/g;
241   return $tex;
242 }
243 sub vertex {
244   my ($v) = @_;
245   if (index($v, 'v0') == 0) {
246     return '\Phi';
247   } else {
248     $v =~ s/^v/v_/g;
249     $v =~ s/[~0-9]$/g;
250     return $v;
251   }
252 }
253 sub tailor {
254   my ($t, $m) = @_;
255   $t =~ s/<([A-Z]?${m}[A-Z]?):([~>]+)>/\2/g;
256   $t =~ s/<[A-Z]+:[~>]+>/g;
257   return $t;
258 }
259 open(my $fh, '<', $ARGV[0]);
260 my $tex; { local $/; $tex = <$fh>; }
261 if (index($tex, "\t") > 0) {
262   print "TABS are prohibited!";
263   exit 1;
264 }
265 print '% This file is auto-generated', "\n\n";
266 print '% --- there are ', length($tex),
267   ' chars in the input (', $ARGV[0], "):\n";
268 foreach my $t (split (/\\n/g, $tex)) {
269   print '% ', $t, "\n";
270 }
271 print "% ---\n";
272 $tex =~ s/\\\\\\\\/\\n/g;
273 $tex =~ s/\\\\\\n//g;
274 $tex =~ s/([a-zA-Z]+)s+/\1/g;
275 $tex =~ s/\\n{2,}/\\n/g;
276 my @cmds = split (/\\n/g, $tex);
277 print '% --- before processing:' . "\n";
278 foreach my $t (split (/\\n/g, $tex)) {

```

```

279 print '% ', $t, "\n";
280 }
281 print '% ---';
282 print ' (' . (0+@cmds) . " lines)\n";
283 print '\begin{picture}', "\n";
284 for (my $c = 0; $c < 0+@cmds; $c++) {
285   my $cmd = $cmds[$c];
286   $cmd =~ s/^\s+//g;
287   $cmd =~ s/%.*//g;
288   my ($head, $tail) = split(/ /, $cmd, 2);
289   my %opts = {};
290   foreach my $p (split(/ /, $tail)) {
291     my ($q, $t) = split(/:/, $p);
292     $opts{$q} = $t;
293   }
294   if (index($head, '->') >= 0) {
295     my $draw = '\draw[';
296     if (exists $opts{'pi'}) {
297       $draw = $draw . '<MB:phi-pi><F:draw=none>';
298       if (not exists $opts{'a'}) {
299         $opts{'a'} = '\pi';
300       }
301     }
302     if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
303       $draw = $draw . '<MB:,phi-rho>';
304     }
305     $draw = $draw . ']';
306     my ($from, $to) = split (/->/, $head);
307     $draw = $draw . " (${$from}) ";
308     if (exists $opts{'bend'}) {
309       $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
310         num($opts{'bend'}) . '>';
311       if (exists $opts{'rho'}) {
312         $draw = $draw . '<MB:,phi-rho>';
313       }
314       $draw = $draw . ']';
315     } else {
316       $draw = $draw . '--';
317     }
318     if (exists $opts{'a'}) {
319       my $a = $opts{'a'};
320       if (index($a, '$') == -1) {
321         $a = '$' . fmt($a) . '$';
322       } else {
323         $a = fmt($a);
324       }
325       $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
326     }
327     if (exists $opts{'break'}) {
328       $draw = $draw . '<F: coordinate [pos=' .
329         ($opts{'break'} / 100) . '] (break)>';
330     }
331     $draw = $draw . " (<MF:${to}><B:break-v>)" ;
332     if (exists $opts{'break'}) {

```

```

333     print tailor($draw, 'F') . ";\n";
334     print ' \node[outer sep=.1cm,inner sep=0cm] ' .
335       'at (break) (break-v) {$' . vertex($to) .
336       '$};' . "\n";
337     print ' ' . tailor($draw, 'B');
338   } else {
339     print tailor($draw, 'M');
340   }
341 } elseif (index($head, '=') >= 0) {
342   my ($from, $to) = split (/=>/, $head);
343   my $size = () = $head =~ /=/g;
344   if ($from eq '') {
345     print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
346       $to . '.center]';
347   } elseif ($to eq '') {
348     print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
349       $from . '.center]';
350   } else {
351     print '\node [phi-arrow] at ($(' .
352       $from . ')!0.5!(' . $to . ')$)';
353   }
354   print '{}';
355 } elseif (index($head, '!') >= 0) {
356   my ($v, $marker) = split (/!+/, $head);
357   my $size = () = $head =~ /*!/g;
358   print '\node [phi-marker, left=' .
359     ($size * 0.6) . 'cm of ' .
360     $v . '.center]{' . fmt($marker) . '}';
361 } elseif (index($head, '+') >= 0) {
362   my ($v, $suffix) = split (/+/, $head);
363   my @friends = ($v);
364   foreach my $c (@cmds) {
365     $e = $c;
366     $e =~ s/\s+//g;
367     my $h = $e;
368     $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
369     foreach my $f (@friends) {
370       my $add = '';
371       if (index($h, $f . '->') >= 0) {
372         $add = substr($h, index($h, '->') + 2);
373       }
374       if ($h =~ /\->\Q${f}\E/) {
375         $add = substr($h, 0, index($h, '->'));
376       }
377       if (index($e, ' xy:' . $f . ',') >= 0) {
378         $add = $h;
379       }
380       if (index($add, '+') == -1
381         and $add ne ''
382         and not(grep(/^Q${add}\E$/, @friends))) {
383         push(@friends, $add);
384       }
385     }
386   }

```



```

387 my @extra = ();
388 foreach my $e (@cmds) {
389     $m = $e;
390     if ($m =~ /\s*\Q${v}\E\s/) {
391         next;
392     }
393     if ($m =~ /\s*[^\s]+\s/ and not($m =~ /\s*\Q${head}\E\s/)) {
394         next;
395     }
396     foreach my $f (@friends) {
397         my $h = $f;
398         $h =~ s/[a-z]$//g;
399         if ($m =~ s/^(\\s*)\Q${f}\E+\Q${suffix}\E\s?/\1${h}${suffix} /g) {
400             last;
401         }
402         $m = s/^(\\s*)\Q${f}\E\s/\1${h}${suffix} /g;
403         $m = s/^(\\s*)\Q${f}\E->/\1${h}${suffix}->/g;
404         $m = s/\\sxy:\Q${f}\E,/ xy:${h}${suffix},/g;
405         $m = s/->\Q${f}\E\s/->${h}${suffix} /g;
406     }
407     if ($m ne $e) {
408         push(@extra, ' ' . $m);
409     }
410 }
411 splice(@extra, 0, 0, @extra[-1]);
412 splice(@extra, -1, 1);
413 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
414     '), friends: [' . join(', ', @friends) . '] in ' .
415     (O+@cmds) . ' lines');
416 splice(@cmds, $c, 1, @extra);
417 print '% cloned ' . $v . ' at line no.' . $c .
418     ' (' . (O+@extra) . ' lines -> ' .
419     (O+@cmds) . ' lines total)';
420 } elsif ($head =~ /\v[0-9]+[a-z]?$/) {
421     print '\node[';
422     if (exists $opts{'xy'}) {
423         my ($v, $right, $down) = split(/,/, $opts{'xy'});
424         my $loc = '';
425         if ($down > 0) {
426             $loc = 'below ';
427         } elsif ($down < 0) {
428             $loc = 'above ';
429         }
430         if ($right > 0) {
431             $loc = $loc . 'right';
432         } elsif ($right < 0) {
433             $loc = $loc . 'left';
434         }
435         print ', ' . $loc . '=';
436         print abs(num($down)) . 'cm and ' .
437             abs(num($right)) . 'cm of ' . $v . '.center';
438     }
439     if (exists $opts{'data'}) {
440         print ',phi-data';

```

```

441     if ($opts{'data'} ne '') {
442         my $d = $opts{'data'};
443         if (index($d, '|') == -1) {
444             $d = '$\Delta\phiDotted\text{' .
445                 '\textnormal{\texttt{' . fmt($d) . '}}}$';
446         } else {
447             $d = fmt($d);
448         }
449         $opts{'box'} = $d;
450     }
451 } elsif (exists $opts{'atom'}) {
452     print ',phi-atom';
453     if ($opts{'atom'} ne '') {
454         my $a = $opts{'atom'};
455         if (index($a, '$') == -1) {
456             $a = '$\lambda\phiDotted{' . fmt($a) . '$';
457         } else {
458             $a = fmt($a);
459         }
460         $opts{'box'} = $a;
461     }
462 } else {
463     print ',phi-object';
464 }
465 print ']';
466 print ' (' . $head . ')';
467 print '{';
468 if (exists $opts{'tag'}) {
469     my $t = $opts{'tag'};
470     if (index($t, '$') == -1) {
471         $t = '$' . $t . '$';
472     } else {
473         $t = fmt($t);
474     }
475     print $t;
476 } else {
477     print '$' . vertex($head) . '$';
478 }
479 print '}';
480 if (exists $opts{'box'}) {
481     print ' node[phi-box] at (';
482     print $head, '.south east) {';
483     print $opts{'box'}, ')';
484 }
485 } else {
486     print $cmd;
487 }
488 print ";\n";
489 }
490 print '\end{picture}%', "\n";
491 print "% --- after processing:\n%";
492 foreach my $c (@cmds) {
493     print '% ', $c, "\n";
494 }

```

```

495 print '% --- (' . (0+@cmds) . " lines)\n";
496 print '\endinput';
497 \end{VerbatimOut}
498 \message{eolang: File with Perl script
499   '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
500 \makeatother

```

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

```

501 \setcounter{FancyVerbLine}{0}

```

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

```

502 \RequirePackage{tikz}
503 \usetikzlibrary{arrows}
504 \usetikzlibrary{shapes}
505 \usetikzlibrary{decorations}
506 \usetikzlibrary{decorations.pathmorphing}
507 \usetikzlibrary{decorations.pathreplacing}
508 \usetikzlibrary{positioning}
509 \usetikzlibrary{calc}
510 \usetikzlibrary{math}
511 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment picture:

```

512 \newenvironment{picture}%
513   {\noindent\begin{tikzpicture}[
514     ->,>=stealth',node distance=0,thick,
515     pics/parallel arrow/.style={
516       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
517   {\end{tikzpicture}}
518 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
519   minimum height=0.5cm, minimum width=0.5cm,
520   single arrow head extend=2mm]
521 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
522   minimum width=1.4em, font={\small\color{white}\ttfamily},
523   fill=gray]
524 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
525   draw,font={\small}]
526 \tikzstyle{phi-object} = [phi-thing,circle]
527 \tikzstyle{phi-data} = [phi-thing,regular polygon,
528   regular polygon sides=8]
529 \tikzstyle{phi-empty} = [phi-object]
530 \tikzset{%
531   phi-rho/.style={
532     postaction={%
533       decoration={
534         show path construction,
535         curveto code={
536           \tikzmath{
537             coordinate \I, \F, \v;
538             \I = (\tikzinputsegmentfirst);
539             \F = (\tikzinputsegmentlast);
540             \v = ($(\I) -(\F)$);
541             real \d, \a, \r, \t;

```

```

542         \d = 0.8;
543         \t = atan2(\vy, \vx);
544         if \vx<0 then { \a = 90; } else { \a = -90; };
545         {
546             \draw[arrows={-latex}, decorate,
547                 decoration={%
548                     snake, amplitude=.4mm,
549                     segment length=2mm,
550                     post length=1mm
551                 }]
552             ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
553             -- ++(\t: 2*\d em);
554         };
555     }
556 },
557 \lineto 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             ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
574             -- ++(\t: 2*\d em);
575         };
576     }
577 }
578 },
579 \decorate
580 }
581 }
582 }
583 \tikzstyle{phi-pi} = [draw,dotted]
584 \tikzstyle{phi-atom} = [phi-object,double]
585 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
586     rectangle,thin,minimum width=1.2em,anchor=north west,
587     font={\scriptsize}]
588 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
589     above=2pt,sloped/.append style={transform shape},
590     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:

```

591 \makeatletter
592 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}

```

593 \makeatother

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

```

594 \makeatletter\newenvironment{sodg}%
595 {\catcode'\|=12 \VerbatimEnvironment%
596 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
597 \begin{VerbatimOut}
598   {\eolang@tmpdir/\jobname/sodg.tex}}
599 {\end{VerbatimOut}}%
600 \def\hash{\eolang@mdfive
601   {\eolang@tmpdir/\jobname/sodg.tex}}%
602 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
603   "\eolang@tmpdir/\jobname/\hash.tex"}%
604 \catcode'\$=3 %
605 \message{Start parsing 'sodg' at line no. \the\inputlineno^^J}
606 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
607   perl "\eolang@tmpdir/eolang-sodg.pl"
608   "\eolang@tmpdir/\jobname/\hash.tex"
609   \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
610   \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%
611 \catcode'\$=active%
612 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
613 \def\eolang@sodgSaveTo{\relax}%
614 }\makeatother

```

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

```

615 \RequirePackage{hyperref}
616 \pdfstringdefDisableCommands{
617   \def\({}%
618   \def\)}{ }%
619   \def\alpha{\alpha}%
620   \def\varphi{\phi}%
621 }
622 \makeatletter
623 \NewExpandableDocumentCommand{\eoAnon}{O{ANONYMIZED}m}{%
624   \ifdefined\eolang@anonymous%
625     \textcolor{orange}{#1}%
626   \else%
627     #2%
628   \fi%
629 }\makeatother

```

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

```

630 \newcommand\eolang{%
631   \eoAnon[XYZ]{\sffamily EO}}

```

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

```

632 \newcommand\phic{%
633   \eoAnon[(\alpha)-cal-cu-lus]{(\varphi)-cal-cu-lus}}

```

\xmirl Then, we define a simple supplementary command to help you print XMIR, the name of our XML-based format of program representation.

```

634 \newcommand\xmir{%
635   \eoAnon[XML\(^+\)]{XMIR}}

```

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

```

636 \newcommand\phiConst{%
637   \mathrel{\hspace{.15em}}%
638   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

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

```

639 \newcommand\phiWave{%
640   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

```

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

```

641 \newcommand\phiSlot[1]{%
642   \xrightarrow{\text{\sffamily\scshape #1}}}

```

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

```

643 \makeatletter
644 \newcommand{\phi0set}[2]{%
645   \mathrel{\mathop{#2}\limits^{
646     \vbox to 0ex{\kern-2\ex@
647       \hbox{$\scriptscriptstyle#1$}\vss}}}}
648 \newcommand{\phiUset}[2]{%
649   \mathrel{\mathop{#2}\limits_{
650     \vbox to 0ex{\kern-6.3\ex@
651       \hbox{$\scriptscriptstyle#1$}\vss}}}}
652 \makeatother

```

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

```

653 \newcommand\phiMany[3]{%
654   \phi0set{#3}{\phiUset{#2}{#1}}}

```

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

```

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

```

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

```

656 \RequirePackage{trimclip}
657 \RequirePackage{amsfonts}
658 \makeatletter
659 \newcommand{\phiDotted}{%
660   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
661 \newcommand{\phiDotted@}[2]{%
662   \begin{group}
663     \settowidth{\dimen\z@}{\m@th#1\rightarrow$}%
664     \settoheight{\dimen\tw@}{\m@th#1\rightarrow$}%
665     \sbox\z@{%
666       \makebox[\dimen\z@][s]{%
667         \clipbox{0 0 {0.4\width} 0}%
668         {\resizebox{\dimen\z@}{\height}%
669           {\m@th#1\dashrightarrow$}}}%

```

```

670      \hss%
671      \clipbox{{0.69\width} {-0.1\height} 0
672      {-\height}}{${\m@th#1\rightarrow$}}%
673    }%
674  }%
675  \ht\z@=\dimen\tw@ \dp\z@=\z@%
676  \box\z@%
677  \endgroup%
678 }
679 \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	0.2.0	eolang-phi.pl: Numbers automatically render as \texttt. No need to use vertical bars around them anymore. . . . .	10
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. . . . .	21		eolang-sodg.pl: The content of the atom and the data boxes is parsed automatically as formulas and numbers, respectively. . . . .	14
	eolang-phi.pl: New symbol added for basket slots . . . . .	10		\xmirt: New command \xmirt prints XMIR in both normal and the anonymous mode of acmart. . . . .	21
	Parsing of the symbols “@,” “^,” and “&” enabled (\varphi, \rho, and \sigma) . . . . .	10	0.3.0	\eolang@lineno: New counter for protecting lineno. . . . .	9
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets . . . . .	10		eolang-phi.pl: New arrow added, that looks like \leadsto. . . . .	10
	eolang-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. . . . .	14		\phiWave: New command \phiWave added to denote a link to a multi-layer attribute. . . . .	22
	\phiq: Parsing of additional symbols enabled. . . . .	13	0.4.0	eolang-sodg.pl: Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed with the \Delta and the \lambda commands. . . . .	14
0.1.0	General: Parsing of package options introduced. . . . .	9		Relative positioning of vertices fixed. . . . .	14
	\eolang: New command \eolang added to print the name of the language in both normal and the anonymous mode of acmart. . . . .	21	0.5.0	eolang-phi.pl: Automated formatting of TRUE and FALSE added. . . . .	10
	\eolang@mdfive: New supplementary command added to calculate MD5 sum of a file. . . . .	9		eolang-sodg.pl: It is possible to use TikZ commands inside the sodg environment. . . . .	14
	eolang-phi.pl: A new Perl script “eolang-phi.pl” added for parsing of phi expressions. . . . .	10		New syntax introduced that allows to make clones of vertices and all their dependants. . . . .	14
	eolang-sodg.pl: There are two Perl scripts now: one for phiuation, another one for sodg. . . . .	14		Now edges may have the break attribute, to make them shorter. . . . .	14
	\phic: New command \phic prints the name of $\varphi$ -calculus in both normal and the anonymous mode of acmart. . . . .	21		\phiMany: New command \phiMany enables iterating over an arrow. . . . .	22
	\phiConst: New command \phiConst added to denote a link to a constant attribute. . . . .	22		\phiSlot: New command \phiSlot added to denote a link to a slot in a basket. . . . .	22
	\phiDotted: New command \phiDotted added to denote a link to a special attribute. . . . .	22	0.6.0	General: Package option nocomments added in order to enable comments suppression in temporary .tex files (may be pretty important for .dtx documents). . . . .	9

eolang-sodg.pl: The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations. . . . .	14		
0.7.0			
nodollar: Now it is possible to use dollar sign instead of the <code>\phiq</code> command. . . . .	13		
eolang-phi.pl: New syntax sugar for $\Phi$ , just using capital “Q” is enough. . . . .	10		
Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols. . . . .	10		
Text in quotes is automatically converted to <code>\texttt</code> . . . . .	10		
0.8.0			
General: The <code>anonymous</code> package option added. . . . .	9		
eolang-phi.pl: Inside <code>phiquation</code> any text inside the <code>\text</code> macro is			
			not processed. . . . . 10
		eolang-sodg.pl: The <code>tag</code> attribute is introduced for changing labels inside a vertex circle. . . . .	14
		<code>\phi0set</code> : New commands <code>\phi0set</code> and <code>\phiUset</code> help position text over and under an arrow. . . . .	22
		<code>\phiSaveTo</code> : The output of the <code>phiquation</code> environment can be redirected to a file. . . . .	12
		<code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file. . . . .	20
	0.9.0		
		<code>\eoAnon</code> : New command <code>\eoAnon</code> added. . . . .	21
		eolang-phi.pl: Proper handling of the <code>matrix</code> environment. . . . .	10
		<code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code> . . . . .	22

# 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.

<b>Symbols</b>		<code>\dashrightarrow</code> ... 669	<code>\FancyVerbLine</code> ... 501
<code>\\$</code> .....	122, 217, 222, 226, 229, 604, 611	<code>\def</code> .....	180, 183, 195, 213, 592, 600, 613, 617, 618, 619, 620
<code>\%</code> .....	192, 221, 609	<code>\Delta</code> .....	444
<code>\(</code> .....	617, 633, 635	<code>\detokenize</code> .....	212
<code>\)</code> .....	618, 633, 635	<code>\dimen</code> 663, 664, 666, 668, 675	
<code>\*</code> .....	88, 90, 119	<code>\dp</code> .....	675
<code>\+</code> .....	235, 362, 393, 399	<code>\draw</code> ...	295, 516, 546, 568
<code>\-</code> .....	633		
<code>\.</code> ....	89, 91, 127, 137, 235		
<code>\/</code> .....	87, 88	<b>E</b>	
<code>\?</code> .....	130	<code>\E</code> 119, 374, 382, 390, 393, 399, 402, 403, 404, 405	
<code>\[</code> .....	87, 124	<code>\end</code> .....	167, 169, 172, 175, 202, 208, 490, 497, 517, 599
<code>\{</code> .....	58, 81, 93, 94, 119, 123, 127, 142	<code>\endinput</code> .....	174, 496
<code>\}</code> ....	58, 93, 94, 119, 142	<code>\eoAnon</code> .	615, 631, 633, 635
<code>\]</code> .....	87, 125	<code>\eolang</code> .....	630
<code>\^</code> .....	123	<code>\eolang-phi.pl</code> .....	24
<code>\ </code> .....	88, 140, 141, 198, 204, 240, 595	<code>\eolang-sodg.pl</code> ...	231
<b>Numbers</b>		<code>\eolang@anonymous</code> 14, 624	
<code>\2</code> ..	87, 89, 91, 99, 127, 255	<code>\eolang@lineno</code> .....	19
<code>\3</code> .....	89, 91	<code>\eolang@mdfive</code> ....	20, 183, 213, 600
<code>\4</code> .....	89	<code>\eolang@nocomments</code> ... 13, 192, 221, 609	
<b>A</b>		<code>\eolang@nodollar</code> ..	
<code>\a</code> 541, 544, 552, 563, 566, 573		..... 217, 222, 224	
<code>\active</code> .	222, 226, 229, 611	<code>\eolang@phiSaveTo</code> .	
<code>\alpha</code> .....	619, 633	..... 180, 193, 195	
<code>\AtBeginDocument</code> ..	229	<code>\eolang@process</code> ...	
<b>B</b>		..... 182, 202, 208	
<code>\Bbbk</code> .....	3	<code>\eolang@sodgSaveTo</code> ..... 592, 610, 613	
<code>\begin</code> .....	25, 146, 149, 151, 200, 206, 232, 283, 513, 597	<code>\eolang@tmpdir</code> ....	
<code>\box</code> .....	676	..... 11, 18, 25, 177, 184, 185, 186, 188, 189, 191, 201, 207, 211, 214, 215, 216, 218, 219, 220, 232, 499, 598, 601, 602, 603, 606, 607, 608	
<b>C</b>		<code>\ex@</code> .....	646, 650
<code>\catcode</code> .....			
..... 198, 204, 217, 222, 226, 229, 595, 604, 611			
<code>\clipbox</code> .....	667, 671		
<code>\color</code> .....	522		
<b>D</b>		<b>F</b>	
<code>\d</code> ..	142, 541, 542, 552, 553, 563, 564, 573, 574	<code>\F</code> .....	537, 539, 540, 552, 559, 561, 562, 573
<b>D</b>			
		<b>G</b>	
		<code>\gdef</code> .....	227
		<b>H</b>	
		<code>\hash</code> ...	183, 186, 188, 191, 213, 216, 218, 220, 600, 603, 606, 608
		<code>\hbox</code> .....	647, 651
		<code>\height</code> ....	668, 671, 672
		<code>\hspace</code> .....	637, 638
		<code>\hss</code> .....	670
		<code>\ht</code> .....	675
		<b>I</b>	
		<code>\I</code> ....	537, 538, 540, 552, 559, 560, 562, 573
		<code>\iexec</code> ...	18, 185, 188, 211, 215, 218, 602, 606
		<code>\ifdefined</code> .....	
		..... 192, 193, 217, 221, 222, 224, 609, 610, 624	
		<code>\ifluatex</code> .....	12
		<code>\ifxetex</code> .....	12
		<code>\inputlineno</code> ...	187, 605
		<b>J</b>	
		<code>\jobname</code> .	18, 184, 185, 186, 188, 191, 201, 207, 211, 214, 215, 216, 218, 220, 598, 601, 602, 603, 606, 608
		<b>K</b>	
		<code>\kern</code> .....	646, 650
		<b>L</b>	
		<code>\lambda</code> .....	456
		<code>\leadsto</code> .....	640
		<code>\limits</code> .....	645, 649
		<b>M</b>	
		<code>\m@th</code> ...	663, 664, 669, 672
		<code>\makeatletter</code> .....	
		..... 19, 21, 24, 179, 182, 210, 231, 591, 594, 622, 643, 658	

<code>\makeatother</code> 19, 23, 178, 181, 209, 223, 500, 593, 614, 629, 652, 679	<code>\phiOset</code> ..... 643, 654	<code>\text</code> ..... 444, 642
<code>\makebox</code> ..... 666	<code>\phiq</code> ..... 210, 227	<code>\textcolor</code> ..... 625
<code>\mapsto</code> ..... 638	<code>\phiquation</code> ..... 182	<code>\textnormal</code> ..... 445
<code>\mapstochar</code> . 638, 640, 660	<code>\phiSaveTo</code> ..... 179	<code>\texttt</code> ..... 445
<code>\mathop</code> ..... 645, 649	<code>\phiSlot</code> ..... 641	<code>\the</code> ..... 187, 605
<code>\mathpalette</code> ..... 660	<code>\phiUset</code> ..... 648, 654	<code>\tikz</code> ..... 502
<code>\mathrel</code> ..... 637, 638, 640, 645, 649, 660	<code>\phiWave</code> ..... 639	<code>\tikzinputsegmentfirst</code> ..... 538, 560
<code>\message</code> 176, 187, 498, 605	<code>\pi</code> ..... 299	<code>\tikzinputsegmentlast</code> ..... 539, 561
<code>\mspace</code> ..... 640	<code>\ProcessPgfPackageOptions</code> ..... 17	<code>\tikzmath</code> ..... 536, 558
	<code>\protected</code> ..... 227	<code>\tikzset</code> ..... 530
		<code>\tikzstyle</code> .... 518, 521, 524, 526, 527, 529, 583, 584, 585, 588
	<b>Q</b>	<code>\ttfamily</code> ..... 522
<b>N</b>	<code>\Q</code> 119, 374, 382, 390, 393, 399, 402, 403, 404, 405	<code>\tw@</code> ..... 664, 675
<code>\newcommand</code> ..... 22, 180, 182, 210, 592, 630, 632, 634, 636, 639, 641, 644, 648, 653, 655, 659, 661	<b>R</b>	
<code>\newcounter</code> ..... 19	<code>\relax</code> .... 3, 195, 613, 660	<b>U</b>
<code>\newenvironment</code> ... ... 197, 203, 512, 594	<code>\RequirePackage</code> .. 1, 2, 3, 4, 5, 6, 7, 8, 20, 502, 615, 656, 657	<code>\usetikzlibrary</code> ... 503, 504, 505, 506, 507, 508, 509, 510, 511
<code>\NewExpandableDocumentCommand</code> ..... 623	<code>\resizebox</code> ..... 668	
<code>\node</code> ..... 334, 345, 348, 351, 358, 421	<code>\rightarrow</code> . 663, 664, 672	<b>V</b>
<code>\nodollar</code> ..... 224	<b>S</b>	<code>\v</code> ..... 537, 540, 559, 562
<code>\noindent</code> ..... 513	<code>\sbox</code> ..... 665	<code>\value</code> 194, 199, 205, 596, 612
	<code>\scriptscriptstyle</code> ..... 647, 651	<code>\varphi</code> ..... 620, 633
<b>P</b>	<code>\scriptsize</code> .... 587, 590	<code>\vbox</code> ..... 646, 650
<code>\pdf@filemdfivesum</code> . 22	<code>\scshape</code> ..... 642	<code>\VerbatimEnvironment</code> ..... 198, 204, 595
<code>\pdfstringdefDisableCommands</code> ..... 616	<code>\setcounter</code> .... 194, 199, 205, 501, 596, 612	<code>\vss</code> ..... 647, 651
<code>\pgfkeys</code> ..... 9	<code>\settoheight</code> ..... 664	<code>\vx</code> ..... 543, 544, 565, 566
<code>\Phi</code> ..... 246	<code>\settowidth</code> ..... 663	<code>\vy</code> ..... 543, 565
<code>\phic</code> ..... 632	<code>\sffamily</code> ..... 631, 642	<b>W</b>
<code>\phiConst</code> ..... 636	<code>\small</code> ..... 522, 525	<code>\width</code> ..... 667, 671
<code>\picture</code> ..... 512	<code>\sodg</code> ..... 594	
<code>\phiDotted</code> . 444, 456, 656	<code>\sodgSaveTo</code> ..... 591	<b>X</b>
<code>\phiDotted@</code> .... 660, 661	<code>\sxy</code> ..... 404	<code>\xmir</code> ..... 634
<code>\phiEOL</code> ..... 655	<b>T</b>	<code>\xrightarrow</code> ..... 642
<code>\phiMany</code> ..... 653	<code>\t</code> 33, 261, 541, 543, 552, 553, 563, 565, 573, 574	<b>Z</b>
		<code>\z@</code> 663, 665, 666, 668, 675, 676