



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

Yegor Bugayenko
yegor256@gmail.com

2022-11-28, 0.7.1

NB! You must run \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 φ -calculus, which is a formal foundation of [EO](#) programming language. The calculus was introduced by Bugayenko (2021) and later formalized by Kudasov et al. (2022). Here is how you render a simple expression:

<pre> app \mapsto [$\rho \mapsto \xi.b.\rho^2, \alpha_0 t \rightsquigarrow \text{TRUE},$ $b \mapsto [\alpha_* \mapsto \text{fn}(56),$ $\varphi \mapsto \Phi.\text{hello.bye}(\xi),$ $\Delta \mapsto \text{01-FE-C3}]$, $x \mapsto [\lambda \mapsto \emptyset]$. </pre>	<pre> 1 \documentclass{article} 2 \pagestyle{empty} 3 \usepackage{eolang} 4 \begin{document} 5 \begin{phiquestion*} 6 app -> [[% it's abstract! 7 ^ !-> \$.b.^{^2}, 0/t~> TRUE, 8 b -> [[*-> fn(56), 9 @ -> Q.hello.bye(\$), 10 D> 01-FE-C3]]],\ 11 x -> [[\lambda ..> ?]]. 12 \end{phiquestion*} 13 \end{document} </pre>
---	---

`phiquestion (env.)` The environment `phiquestion` lets you write a φ -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 “ φ ” (`\varphi`),
- “^” maps to “ ρ ” (`\rho`),
- “\$” maps to “ ξ ” (`\xi`),
- “&” maps to “ σ ” (`\sigma`),
- “?” maps to “ \emptyset ” (`\varnothing`),
- “Q” maps to “ Φ ” (`\Phi`),
- “->” maps to “ \mapsto ” (`\mapsto`),
- “~>” maps to “ \rightsquigarrow ” (`\phiWave`),
- “!->” maps to “ \multimap ” (`\phiConst`),
- “.>” maps to “ $\dot{\mapsto}$ ” (`\phiDotted`),
- “D>” maps to “ $\Delta \mapsto$ ” (`\Delta .>`),
- “L>” maps to “ $\lambda \mapsto$ ” (`\lambda .>`),
- “[[” maps to “ \llbracket ” (`\llbracket`),
- “]]” maps to “ \rrbracket ” (`\rrbracket`),
- “|abc|” maps to “abc” (`\texttt{abc}`).

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

- “-abc>” maps to “ \xrightarrow{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 `\phiiq{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 `\phiiq{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 `\phiiq{*/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.

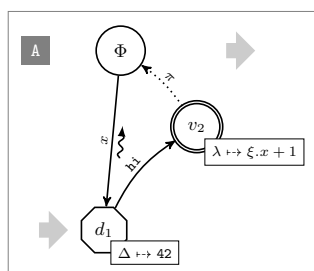
Object names are automatically converted to fixed-width font too, if they have more than one letter.

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

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

<p>A simple object $x \mapsto \llbracket \varphi \mapsto y \rrbracket$ is a decorator of the data object $y \mapsto \llbracket \Delta \mapsto 42 \rrbracket$.</p>	<pre> 4 \begin{document} 5 A simple object 6 \phiiq{x -> [[@ -> y]]} \ 7 is a decorator of 8 the data object \ 9 \$y -> [[\Delta .> 42]]\$. 10 \end{document} </pre>
---	--

`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 π 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 mode of `\acmart` and prints itself differently, to `\xmir` double-blind your paper. There is also `\phic` command to print the name of φ -calculus, also sensitive to anonymous mode. The macro `\xmir` prints “XMIR”.

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

```
4 \begin{document}
5 In our research we use \eolang{}, \
6 an experimental object-oriented \
7 dataflow language, \phic{}, as its \
8 formal foundation, and \xmir{} --- \
9 its XML-based presentation.
10 \end{document}
```

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

If x is an identifier and y is an object, then $x \mapsto y$ makes y a constant, $x \rightsquigarrow y$ makes it a decoratee of an arbitrary number of objects, while $x \dot{\mapsto} y$ makes it a special attribute.

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

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

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

$\llbracket \dot{\mapsto} \rrbracket$.

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

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

The expression $\llbracket \alpha_1 \mapsto x_1, \alpha_2 \mapsto x_2, \dots, \alpha_n \mapsto x_n \rrbracket$ and expression $\llbracket \alpha_i \xrightarrow{n} x_i \rrbracket$ are syntactically different but semantically equivalent.

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

`\phiSaveTo` If you want to use `phiqutation` or `sodg` environments inside `tabular` or any other environment or command, you won't be able to do this, because `phiqutation` and `sodg` are “verbatim” environments. `\phiSaveTo` and `\sodgSaveTo` commands will help you in this situation. You use them right before `\begin{phiqutation}` 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: $x \mapsto \emptyset$
Bound: $x \mapsto [\Delta \mapsto 42]$

```

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

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}
```

3 More Examples

The `phiqutation` 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:

$\frac{x \mapsto [\varphi \mapsto y] \quad y \mapsto [z \mapsto 42]}{x.z \mapsto 42} R1$
--

```

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

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

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

```

6 \begin{phiquestion*}
7 \dfrac{\begin{split}
8 x->[[@->y, z->42,
9 0/g->?, 1/foo->42]]
10 \end{split}}{\begin{split}
11 x->[[@->y, z->?, f ~> |pi|(
12 0->[[ \psi !-> |hello|(12) ]],
13 1->42)]]
14 \end{split}}\text{R2}.
15 \end{phiquestion*}

```

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

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

```

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

```

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

$$\text{Discriminant can be calculated using the following simple formula:}$$

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

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{phiquestion}
9 D = b^{2} - {4}ac.
10 \label{d}
11 \end{phiquestion}
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):

$$\begin{array}{ll} [\alpha_0 \mapsto x] & \text{This is formation} \\ [\alpha_0 \mapsto \emptyset] & \text{Abstraction} \\ x(\Delta \mapsto 42) & \text{Application} \end{array}$$

```

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

```

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  $\llbracket 0 \rightarrow x \rrbracket$ 
7 may be replaced with a formula
8  $\llbracket Q \times a^2 \rrbracket$ .
```

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

$\mapsto \llbracket \lambda \mapsto f_1 \rrbracket$,
 $\mapsto \llbracket \alpha_0 \mapsto \emptyset, \varphi \mapsto \text{hello}(\xi), x \mapsto \text{FALSE} \rrbracket$,
 $=43-09$.

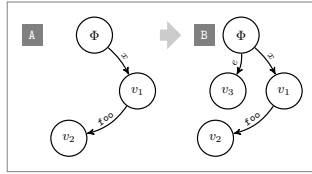
```
5 \begin{phiquation*}
6 x(\pi) \rightarrow \llbracket \lambda \mapsto f_1 \rrbracket, \llbracket
7 x(a,b,c) \rightarrow \llbracket \alpha_0 \rightarrow ?, \
8 @ \rightarrow |\text{hello}|(\$), x \rightarrow |\text{FALSE}| \rrbracket, \llbracket
9 \Delta = |43-09|.
10 \end{phiquation*}
```

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

$\llbracket b \mapsto \emptyset \rrbracket$,
 $\llbracket \varphi \mapsto \text{TRUE}, \Delta \mapsto 42 \rrbracket$,
 $\Delta = 43-09$.

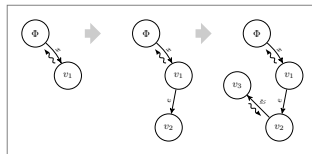
```
5 \begin{phiquation*}
6 \llbracket b \rightarrow ? \rrbracket,
7 \llbracket @ \rightarrow \text{TRUE}, \Delta \mapsto 42 \rrbracket, \llbracket
8 \Delta = |43-09|.
9 \end{phiquation*}
```

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



```
5 \begin{sodg}
6 v0 \ll 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 \ll v0a!B
15 \end{sodg}
```

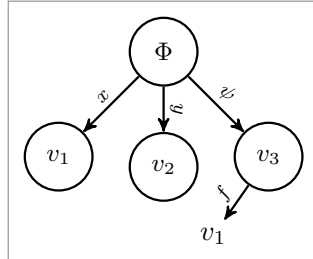
You can make a copy from a copy:



```
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 \ll v0=>v0a
10 v2a xy:v1a,-.8,1.3
11 v1a->v2a a:e
12 v0a+b xy:v0a,3,0 \ll 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:

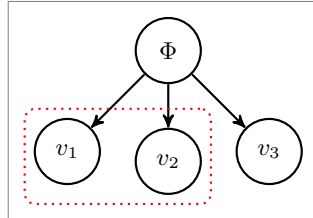


```

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 tmpdir
15 }
16 \ProcessPgfOptions{/eolang}

Then, we make a directory where all temporary files will be kept:
17 \iexec[null]{mkdir -p "\eolang@tmpdir/\jobname"}%

\eolang@lineno Then, we define an internal counter to protect line number from changing:
18 \makeatletter\newcounter{eolang@lineno}\makeatother

\eolang@mdfive Then, we define a command for MD5 hash calculating of a file:
19 \RequirePackage{pdftexcmds}
20 \makeatletter
21 \newcommand\eolang@mdfive[1]{\pdf@filemdfivesum{#1}}
22 \makeatother

eolang-phi.pl Then, we create a Perl script for phiquation processing using VerbatimOut environ-
ment from fancyvrb:
23 \makeatletter
24 \begin{VerbatimOut}{\eolang@tmpdir/eolang-phi.pl}
25 $macro = $ARGV[0];
26 open(my $fh, '<', $ARGV[1]);
27 my $tex; { local $/; $tex = <$fh>; }
28 print '% This file is auto-generated', "\n";
29 print '% There are ', length($tex),
30 ' chars in the input: ', $ARGV[1], "\n";
31 print '% ---', "\n";
32 if (index($tex, "\t") > 0) {
33   print "TABS are prohibited!";
34   exit 1;
35 }
36 my @lines = split (/\\n/g, $tex);
37 foreach my $t (@lines) {
38   print '% ', $t, "\n";
39 }
40 print '% ---', "\n";
41 $tex =~ s/\\.*/\\n\\n/g;
42 $tex =~ s/^\\s+|\\s+$/g;
43 my $gathered = (0 == $tex =~ /\\n\\s+/g);
44 if ($gathered) {
45   print '% The "gathered" is used since all lines are left-aligned' . "\n";
46 }
47 my $align = 0;
48 print '% The "align" is NOT used by default' . "\n";
49 if (index($tex, '&&') >= 0) {
50   $macro =~ s/equation/align/g;
51   $align = 1;
52   print '% The "align" is used because of && seen in the text' . "\n";
53 }
54 if ($macro ne 'phiq') {
55   $tex =~ s/\\\\\\\\\\n/\\n\\n/g;
56   $tex =~ s/\\\\\\n\\s*/g;
57   $tex =~ s/\\n*(\\\\label\\{[\\}]+\\})\\n*/\\1/g;

```

```

58 $tex =~ s/\n{3,}/\n\n/g;
59 }
60 my @texts = ();
61 sub trep {
62   my ($s) = @_;
63   my $open = 0;
64   my $p = 0;
65   for (; $p < length($s); $p++) {
66     $c = substr($s, $p, 1);
67     if ($c eq '}') {
68       if ($open eq 0) {
69         last;
70       }
71       $open--;
72     }
73     if ($c eq '{') {
74       $open++;
75     }
76   }
77   push(@texts, substr($s, 0, $p));
78   return '{TEXT' . (0+@texts - 1) . '}' . substr($s, $p + 1);
79 }
80 $tex =~ s/\\text\{(.+)/trep("$1")/ge;
81 $tex =~ s/(?<![&])&(?![&])\\sigma{/g;
82 $tex =~ s/([~\\{a-z0-9]|~)Q(?![a-z0-9])\\1\\Phi{/g;
83 $tex =~ s/([~\\{a-z0-9]|~)D>\\1\\Delta{}/g;
84 $tex =~ s/([~\\{a-z0-9]|~)L>\\1\\lambda{}/g;
85 $tex =~ s/"([~"]+)"/"\\1"/g;
86 $tex =~ s/(~|(?<=[\s](\\|[,.->|/]))([a-z][a-z0-9]+)(?=[\s](\\|[,.->|/))\\2/g;
87 $tex =~ s/([~_]|~)([0-9]+|~*)\\/(\\?[a-z]+|\\|[a-z]+|)
88   (->|\\.\\.\\.>|~>|:=|!->)/\\1\\alpha_{\\2}\\vert{\\3\\space{\\4}/g;
89 $tex =~ s/([~_]|~)([0-9]+|~*)
90   (->|\\.\\.\\.>|~>|:=|!->)/\\1\\alpha_{\\2}\\space{\\3}/g;
91 if ($macro ne 'phi') {
92   $tex =~ s/\\begin\\{split\\}\\n/\\begin{split}&/g;
93   $tex =~ s/\\n\\s*\\end\\{split\\}/\\end{split}/g;
94   $tex =~ s/\\n\\n/\\\\&/g;
95   $tex =~ s/\\n/\\phiEOL{\\n}&/g;
96   $tex =~ s/\\\\/\\\\\\n/g;
97   $tex =~ s/([~&\\s])\\s{2}([~\\s])/\\1 \\2/g;
98   $tex =~ s/\\s{2}/ \\quad{/g;
99   $tex = '&' . $tex;
100  my $lead = '[~\\s]+\\s(->|:=|!->)';
101  my @leads = $tex =~ /&${lead}/g;
102  my @eols = $tex =~ /&/g;
103  if (0+@leads == 0+@eols && 0+@eols > 0) {
104    $tex =~ s/&${lead}/\\1&/g;
105    $gathered = 0;
106    print '% The "gathered" is NOT used because all ' .
107      (0+@eols) . ' lines are ' . (0+@leads) . " leads\\n";
108  }
109 }
110 $tex =~ s/\\$/\\xi{/g;
111 $tex =~ s/(?<![{]\\^\\rho{/g;

```

```

112 $tex =~ s/\[/\[/\llbracket\mathrel{/g;
113 $tex =~ s/\]/\]/\rrbracket\mathrel{/g;
114 $tex =~ s/([\s,>()]{0-9A-F}{2}(?:-[0-9A-F]{2})+|
115 [0-9]+(?:\.[0-9]+)?)(?!\\)/\1\2/xg;
116 $tex =~ s/TRUE/|TRUE|/g;
117 $tex =~ s/FALSE/|FALSE|/g;
118 $tex =~ s/\?/\varnothing{/g;
119 $tex =~ s/@/\varphi{/g;
120 $tex =~ s/-([a-z]+)>/\mathrel{\phiSlot{1}}/g;
121 $tex =~ s/!->/\mathrel{\phiConst}/g;
122 $tex =~ s/->/\mathrel{\mapsto}/g;
123 $tex =~ s/~>/\mathrel{\phiWave}/g;
124 $tex =~ s/:=/\mathrel{\vDash}/g;
125 $tex =~ s/\.\. >/\mathrel{\phiDotted}/g;
126 $tex =~ s/|{2,}/|/g;
127 $tex =~ s/|([^\|]+)|/\textnormal{\texttt{1}}{/g;
128 $tex =~ s/{TEXT(d+)\}/'\text{' . @texts[$1] . '}'/ge;
129 if ($macro eq 'phiq') {
130   print '$' if ($tex ne '');
131 } else {
132   print '\begin{' , $macro, "\n";
133   if (not($align)) {
134     if ($gathered) {
135       print '\begin{gathered}';
136     } else {
137       print '\begin{split}';
138     }
139     print "\n";
140   }
141 }
142 if ($gathered and not($align)) {
143   $tex =~ s/~&/g;
144   $tex =~ s/\n&\n/g;
145 }
146 print $tex;
147 if ($macro eq 'phiq') {
148   print '$' if ($tex ne '');
149 } else {
150   if (not($align)) {
151     print "\n";
152     if ($gathered) {
153       print '\end{gathered}';
154     } else {
155       print '\end{split}';
156     }
157   }
158   print "\n" . '\end{' . $macro . '}' ;
159 }
160 print '\endinput';
161 \end{VerbatimOut}
162 \message{eolang: File with Perl script
163   '\eolang@tmpdir/eolang-phi.pl' saved^^J}%
164 \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:

```
165 \makeatletter
166 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
167 \makeatother
```

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

```
168 \makeatletter\newcommand\eolang@process[1]{
169   \def\hash{\eolang@mdfive
170     {\eolang@tmpdir/\jobname/phiuation.tex}}%
171   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiuation.tex"
172     "\eolang@tmpdir/\jobname/\hash.tex"}%
173   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
174     perl "\eolang@tmpdir/eolang-phi.pl"
175     '#1'
176     "\eolang@tmpdir/\jobname/\hash.tex"
177     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
178     \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
179   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
180   \def\eolang@phiSaveTo{\relax}%
181 }
182 \newenvironment{phiuation*}%
183 {\catcode'\|=12 \VerbatimEnvironment%
184 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
185 \begin{VerbatimOut}
186   {\eolang@tmpdir/\jobname/phiuation.tex}}
187 {\end{VerbatimOut}\eolang@process{equation*}}
188 \newenvironment{phiuation}%
189 {\catcode'\|=12 \VerbatimEnvironment%
190 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
191 \begin{VerbatimOut}
192   {\eolang@tmpdir/\jobname/phiuation.tex}}
193 {\end{VerbatimOut}\eolang@process{equation}}
194 \makeatother
```

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

```
195 \makeatletter\newcommand\phiiq[1]{%
196   \iexec[trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{
197     /bin/echo '\detokenize{#1}'}%
198   \def\hash{\eolang@mdfive
199     {\eolang@tmpdir/\jobname/phiq.tex}}%
200   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiq.tex"
201     "\eolang@tmpdir/\jobname/\hash.tex"}%
202   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
203   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
204     perl \eolang@tmpdir/eolang-phi.pl '\phiq'
205     "\eolang@tmpdir/\jobname/\hash.tex"
206     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi}%
207   \ifdefined\eolang@nodollar\else\catcode'\$=active\fi%
208 }\makeatother
```

`nodollar` Then, we redefine dollar sign:

```

209 \ifdefined\eolang@nodollar\else
210   \begin{group}
211     \catcode'\$=\active
212     \protected\gdef$#1$\{\phiq{#1}\}
213   \end{group}
214   \AtBeginDocument{\catcode'\$=\active}
215 \fi

```

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

```

216 \makeatletter
217 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
218 sub num {
219   my ($i) = @_;
220   $i =~ s/(\+|-)\./\10./g;
221   return $i;
222 }
223 sub fmt {
224   my ($tex) = @_;
225   $tex =~ s/|([^\|]+)|\\textnormal{\texttt{\1}}/g;
226   return $tex;
227 }
228 sub vertex {
229   my ($v) = @_;
230   if (index($v, 'v0') == 0) {
231     return '\Phi';
232   } else {
233     $v =~ s/^v/v_/g;
234     $v =~ s/[~0-9]$//g;
235     return $v;
236   }
237 }
238 sub tailor {
239   my ($t, $m) = @_;
240   $t =~ s/<([A-Z]?${m}[A-Z]?):([~>+)>/\2/g;
241   $t =~ s/<[A-Z]+:[~>+>/g;
242   return $t;
243 }
244 open(my $fh, '<', $ARGV[0]);
245 my $tex; { local $/; $tex = <$fh>; }
246 if (index($tex, "\t") > 0) {
247   print "TABS are prohibited!";
248   exit 1;
249 }
250 print '% This file is auto-generated', "\n%\n";
251 print '% --- there are ', length($tex),
252   ' chars in the input (', $ARGV[0], "):\n";
253 foreach my $t (split /\n/g, $tex) {
254   print '% ', $t, "\n";
255 }
256 print "% ---\n";
257 $tex =~ s/\\\\\\//\n/g;
258 $tex =~ s/\\n//g;
259 $tex =~ s/(\[a-zA-Z]+\s+)/\1/g;

```

```

260 $tex =~ s/\n{2,}/\n/g;
261 my @cmds = split(/\n/g, $tex);
262 print '% --- before processing:' . "\n";
263 foreach my $t (split(/\n/g, $tex)) {
264     print '% ', $t, "\n";
265 }
266 print '% ---';
267 print ' (' . (0+@cmds) . " lines)\n";
268 print '\begin{picture}', "\n";
269 for (my $c = 0; $c < 0+@cmds; $c++) {
270     my $cmd = $cmds[$c];
271     $cmd =~ s/^\s+//g;
272     $cmd =~ s/%.*//g;
273     my ($head, $tail) = split(/ /, $cmd, 2);
274     my %opts = {};
275     foreach my $p (split(/ /, $tail)) {
276         my ($q, $t) = split(/:/, $p);
277         $opts{$q} = $t;
278     }
279     if (index($head, '->') >= 0) {
280         my $draw = '\draw[';
281         if (exists $opts{'pi'}) {
282             $draw = $draw . '<MB:phi-pi><F:draw=none>';
283             if (not exists $opts{'a'}) {
284                 $opts{'a'} = '\pi';
285             }
286         }
287         if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
288             $draw = $draw . '<MB:,phi-rho>';
289         }
290         $draw = $draw . ']';
291         my ($from, $to) = split (/->/, $head);
292         $draw = $draw . " (${$from}) ";
293         if (exists $opts{'bend'}) {
294             $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
295                 num($opts{'bend'}) . '>';
296             if (exists $opts{'rho'}) {
297                 $draw = $draw . '<MB:,phi-rho>';
298             }
299             $draw = $draw . ']';
300         } else {
301             $draw = $draw . '--';
302         }
303         if (exists $opts{'a'}) {
304             my $a = $opts{'a'};
305             if (index($a, '$') == -1) {
306                 $a = '$' . fmt($a) . '$';
307             } else {
308                 $a = fmt($a);
309             }
310             $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
311         }
312         if (exists $opts{'break'}) {
313             $draw = $draw . '<F: coordinate [pos=' .

```

```

314         ($opts{'break'} / 100) . ']' (break)>';
315     }
316     $draw = $draw . " (<MF:${to}><B:break-v>";
317     if (exists $opts{'break'}) {
318         print tailor($draw, 'F') . ";\n";
319         print ' \node[outer sep=.1cm,inner sep=0cm] ' .
320             'at (break) (break-v) {$' . vertex($to) .
321             '$};' . "\n";
322         print ' ' . tailor($draw, 'B');
323     } else {
324         print tailor($draw, 'M');
325     }
326 } elsif (index($head, '=>') >= 0) {
327     my ($from, $to) = split (/=>/, $head);
328     my $size = () = $head =~ /=/g;
329     if ($from eq '') {
330         print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
331             $to . '.center]';
332     } elsif ($to eq '') {
333         print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
334             $from . '.center]';
335     } else {
336         print '\node [phi-arrow] at ($(' .
337             $from . ')!0.5!(' . $to . ')$)';
338     }
339     print '{}';
340 } elsif (index($head, '!'') >= 0) {
341     my ($v, $marker) = split (/!/, $head);
342     my $size = () = $head =~ /*!/g;
343     print '\node [phi-marker, left=' .
344         ($size * 0.6) . 'cm of ' .
345         $v . '.center]{' . fmt($marker) . '}';
346 } elsif (index($head, '+') >= 0) {
347     my ($v, $suffix) = split (/+/, $head);
348     my @friends = ($v);
349     foreach my $c (@cmds) {
350         $e = $c;
351         $e =~ s/^\s+//g;
352         my $h = $e;
353         $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
354         foreach my $f (@friends) {
355             my $add = '';
356             if (index($h, $f . '->') >= 0) {
357                 $add = substr($h, index($h, '->') + 2);
358             }
359             if ($h =~ /\->\Q${f}\E/) {
360                 $add = substr($h, 0, index($h, '->'));
361             }
362             if (index($e, ' xy:' . $f . ',') >= 0) {
363                 $add = $h;
364             }
365             if (index($add, '+') == -1
366                 and $add ne ''
367                 and not(grep(/^Q${add}\E/, @friends))) {

```

```

368         push(@friends, $add);
369     }
370 }
371 }
372 my @extra = ();
373 foreach my $e (@cmds) {
374     $m = $e;
375     if ($m =~ /\s*\Q${v}\E\s/) {
376         next;
377     }
378     if ($m =~ /\s*[\s]+\s/ and not($m =~ /\s*\Q${head}\E\s/)) {
379         next;
380     }
381     foreach my $f (@friends) {
382         my $h = $f;
383         $h =~ s/[a-z]$//g;
384         if ($m =~ s/^(.*)\Q${f}\E+\Q${suffix}\E\s?/\1${h}${suffix} /g) {
385             last;
386         }
387         $m =~ s/^(.*)\Q${f}\E\s/\1${h}${suffix} /g;
388         $m =~ s/^(.*)\Q${f}\E->/\1${h}${suffix}->/g;
389         $m =~ s/\sxy:\Q${f}\E,/ xy:${h}${suffix},/g;
390         $m =~ s/->\Q${f}\E\s/->${h}${suffix} /g;
391     }
392     if ($m ne $e) {
393         push(@extra, ' ' . $m);
394     }
395 }
396 splice(@extra, 0, 0, @extra[-1]);
397 splice(@extra, -1, 1);
398 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
399     '), friends: [' . join(', ', @friends) . ']' in ' .
400     (0+@cmds) . ' lines');
401 splice(@cmds, $c, 1, @extra);
402 print '% cloned ' . $v . ' at line no.' . $c .
403     ' (+ ' . (0+@extra) . ' lines -> ' .
404     (0+@cmds) . ' lines total)';
405 } elsif ($head =~ /\v[0-9]+[a-z]?$/) {
406     print '\node[';
407     if (exists $opts{'xy'}) {
408         my ($v, $right, $down) = split(/,/ , $opts{'xy'});
409         my $loc = '';
410         if ($down > 0) {
411             $loc = 'below';
412         } elsif ($down < 0) {
413             $loc = 'above';
414         }
415         if ($right > 0) {
416             $loc = $loc . 'right';
417         } elsif ($right < 0) {
418             $loc = $loc . 'left';
419         }
420         print ', ' . $loc . '=';
421         print abs(num($down)) . 'cm and ' .

```



```

422         abs(num($right)) . 'cm of ' . $v . '.center';
423     }
424     if (exists $opts{'data'}) {
425         print ',phi-data';
426         if (not $opts{'data'} eq '') {
427             my $d = $opts{'data'};
428             if (index($d, '|') == -1) {
429                 $d = '$\Delta\phiDotted\text{' .
430                     '\textnormal{\texttt{' . fmt($d) . '}}}$';
431             } else {
432                 $d = fmt($d);
433             }
434             $opts{'box'} = $d;
435         }
436     } elsif (exists $opts{'atom'}) {
437         print ',phi-atom';
438         if (not $opts{'atom'} eq '') {
439             my $a = $opts{'atom'};
440             if (index($a, '$') == -1) {
441                 $a = '$\lambda\phiDotted{' . fmt($a) . '$';
442             } else {
443                 $a = fmt($a);
444             }
445             $opts{'box'} = $a;
446         }
447     } else {
448         print ',phi-object';
449     }
450     print ']';
451     print ' (' . $head . ')';
452     print '{';
453     if (exists $opts{'tag'}) {
454         my $t = $opts{'tag'};
455         if (index($t, '$') == -1) {
456             $t = '$' . $t . '$';
457         } else {
458             $t = fmt($t);
459         }
460         print $t;
461     } else {
462         print '$' . vertex($head) . '$';
463     }
464     print '}';
465     if (exists $opts{'box'}) {
466         print ' node[phi-box] at (';
467         print $head, '.south east) {';
468         print $opts{'box'}, ')';
469     }
470 } else {
471     print $cmd;
472 }
473 print ";\n";
474 }
475 print '\end{picture}%', "\n";

```

```

476 print "% --- after processing:\n%";
477 foreach my $c (@cmds) {
478   print '% ', $c, "\n";
479 }
480 print '% --- (' . (0+@cmds) . " lines)\n";
481 print '\endinput';
482 \end{VerbatimOut}
483 \message{eolang: File with Perl script
484   '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
485 \makeatother

```

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

```

486 \setcounter{FancyVerbLine}{0}

```

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

```

487 \RequirePackage{tikz}
488 \usetikzlibrary{arrows}
489 \usetikzlibrary{shapes}
490 \usetikzlibrary{decorations}
491 \usetikzlibrary{decorations.pathmorphing}
492 \usetikzlibrary{decorations.pathreplacing}
493 \usetikzlibrary{positioning}
494 \usetikzlibrary{calc}
495 \usetikzlibrary{math}
496 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment picture:

```

497 \newenvironment{picture}%
498   {\noindent\begin{tikzpicture}[
499     ->,>=stealth',node distance=0,thick,
500     pics/parallel arrow/.style={
501       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
502   {\end{tikzpicture}}
503 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
504   minimum height=0.5cm, minimum width=0.5cm,
505   single arrow head extend=2mm]
506 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
507   minimum width=1.4em, font={\small\color{white}\ttfamily},
508   fill=gray]
509 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
510   draw,font={\small}]
511 \tikzstyle{phi-object} = [phi-thing,circle]
512 \tikzstyle{phi-data} = [phi-thing,regular polygon,
513   regular polygon sides=8]
514 \tikzstyle{phi-empty} = [phi-object]
515 \tikzset{%
516   phi-rho/.style={
517     postaction={%
518       decoration={
519         show path construction,
520         curveto code={
521           \tikzmath{
522             coordinate \I, \F, \v;

```

```

523 \I = (\tikzinputsegmentfirst);
524 \F = (\tikzinputsegmentlast);
525 \v = ($(\I) -(\F)$);
526 real \d, \a, \r, \t;
527 \d = 0.8;
528 \t = atan2(\vy, \vx);
529 if \vx<0 then { \a = 90; } else { \a = -90; };
530 {
531 \draw[arrows={-latex}, decorate,
532 decoration={%
533 snake, amplitude=.4mm,
534 segment length=2mm,
535 post length=1mm
536 }]
537 ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
538 -- ++(\t: 2*\d em);
539 };
540 }
541 },
542 lineto code={
543 \tikzmath{
544 coordinate \I, \F, \v;
545 \I = (\tikzinputsegmentfirst);
546 \F = (\tikzinputsegmentlast);
547 \v = ($(\I) -(\F)$);
548 real \d, \a, \r, \t;
549 \d = 0.8;
550 \t = atan2(\vy, \vx);
551 if \vx<0 then { \a = 90; } else { \a = -90; };
552 {
553 \draw[arrows={-latex}, decorate,
554 decoration={%
555 snake, amplitude=.4mm,
556 segment length=2mm,
557 post length=1mm}]
558 ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
559 -- ++(\t: 2*\d em);
560 };
561 }
562 }
563 },
564 decorate
565 }
566 }
567 }
568 \tikzstyle{phi-pi} = [draw,dotted]
569 \tikzstyle{phi-atom} = [phi-object,double]
570 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
571 rectangle,thin,minimum width=1.2em,anchor=north west,
572 font={\scriptsize}]
573 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
574 above=2pt,sloped/.append style={transform shape},
575 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:

```
576 \makeatletter
577 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
578 \makeatother
```

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

```
579 \makeatletter\newenvironment{sodg}%
580 {\catcode'\|=12 \VerbatimEnvironment%
581 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
582 \begin{VerbatimOut}
583   {\eolang@tmpdir/\jobname/sodg.tex}}
584 {\end{VerbatimOut}}%
585 \def\hash{\eolang@mdfive
586   {\eolang@tmpdir/\jobname/sodg.tex}}%
587 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
588   "\eolang@tmpdir/\jobname/\hash.tex"}%
589 \catcode'\$=3 %
590 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
591   perl "\eolang@tmpdir/eolang-sodg.pl"
592   "\eolang@tmpdir/\jobname/\hash.tex"
593   \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\\n|\\$)//g'\fi
594   \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%
595 \catcode'\$=active%
596 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
597 \def\eolang@sodgSaveTo{\relax}%
598 }\makeatother
```

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

```
599 \newcommand\eolang{%
600   \ifdefined\anon%
601     \anon[XYZ]{\sffamily EO}}%
602   \else%
603     {\sffamily EO}%
604   \fi%
605 }
```

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

```
606 \RequirePackage{hyperref}
607 \newcommand\phic{%
608   \ifdefined\anon%
609     \anon[\texorpdfstring{\alpha}{a}]{-cal\ -cu\ -lus}
610     {\texorpdfstring{\varphi}{phi}]{-cal\ -cu\ -lus}%
611   \else%
612     \texorpdfstring{\varphi}{phi}]{-cal\ -cu\ -lus}%
613   \fi%
614 }
```

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

```
615 \newcommand\xmirl{%
```

```

616 \ifdefined\anon%
617 \anon[XML$~+$]{XMIR}%
618 \else%
619 XMIR%
620 \fi%
621 }

```

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

```

622 \newcommand\phiConst{%
623 \mathrel{\hspace{.15em}}%
624 \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

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

```

625 \newcommand\phiWave{%
626 \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

```

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

```

627 \newcommand\phiSlot[1]{%
628 \xrightarrow{\text{\sffamily\scshape #1}}}

```

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

```

629 \makeatletter
630 \newcommand{\phiOset}[2]{%
631 \mathrel{\mathop{#2}\limits^{
632 \vbox to 0ex{\kern-2\ex@
633 \hbox{$\scriptscriptstyle#1$}\vss}}}}
634 \newcommand{\phiUset}[2]{%
635 \mathrel{\mathop{#2}\limits_{
636 \vbox to 0ex{\kern-6.3\ex@
637 \hbox{$\scriptscriptstyle#1$}\vss}}}}
638 \makeatother

```

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

```

639 \newcommand\phiMany[3]{%
640 \phiOset{#3}{\phiUset{#2}{#1}}}

```

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

```

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

```

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

```

642 \RequirePackage{trimclip}
643 \RequirePackage{amsfonts}
644 \makeatletter
645 \newcommand{\phiDotted}{%
646 \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
647 \newcommand{\phiDotted@}[2]{%
648 \begingroup%
649 \settowidth{\dimen\z@}{\m@th#1\rightarrow$}%
650 \settoheight{\dimen\tw@}{\m@th#1\rightarrow$}%
651 \sbox\z@{%

```

```

652 \makebox[\dimen\z@][s]{%
653   \clipbox{0 0 {0.4\width} 0}%
654   {\resizebox{\dimen\z@}{\height}%
655     {\$ \m@th#1 \dashrightarrow$}}%
656   \hss%
657   \clipbox{{0.69\width} {-0.1\height} 0
658     {-\height}}{\$ \m@th#1 \rightarrow$}%
659   }%
660 }%
661 \ht\z@=\dimen\tw@ \dp\z@=\z@%
662 \box\z@%
663 \endgroup%
664 }
665 \makeatother

```

References

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

Change History

0.0.1	General: First draft.	8	0.2.0	<code>eolang-phi.pl</code> : Numbers automatically render as <code>\texttt</code> . No need to use vertical bars around them anymore.	9
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.	20	<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.	13	
	<code>eolang-phi.pl</code> : New symbol added for basket slots	9	<code>\xmirt</code> : New command <code>\xmirt</code> prints XMIR in both normal and the anonymous mode of <code>acmart</code>	20	
	Parsing of the symbols “@,” “^,” and “&” enabled (<code>\varphi</code> , <code>\rho</code> , and <code>\sigma</code>)	9	0.3.0	<code>\eolang@lineno</code> : New counter for protecting <code>lineno</code>	9
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets	9	<code>eolang-phi.pl</code> : New arrow added, that looks like <code>\leadsto</code>	9	
	<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.	13	<code>\phiWave</code> : New command <code>\phiWave</code> added to denote a link to a multi-layer attribute.	21	
	<code>\phiq</code> : Parsing of additional symbols enabled.	12	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.	13
0.1.0	General: Parsing of package options introduced.	8		Relative positioning of vertices fixed.	13
	<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>	20	0.5.0	<code>eolang-phi.pl</code> : Automated formatting of <code>TRUE</code> and <code>FALSE</code> added.	9
	<code>\eolang@mdfive</code> : New supplementary command added to calculate MD5 sum of a file.	9	<code>eolang-sodg.pl</code> : It is possible to use TikZ commands inside the <code>sodg</code> environment.	13	
	<code>eolang-phi.pl</code> : A new Perl script “ <code>eolang-phi.pl</code> ” added for parsing of <code>phi</code> expressions.	9	New syntax introduced that allows to make clones of vertices and all their dependants.	13	
	<code>eolang-sodg.pl</code> : There are two Perl scripts now: one for <code>phi</code> uation, another one for <code>sodg</code>	13	Now edges may have the <code>break</code> attribute, to make them shorter.	13	
	<code>\phic</code> : New command <code>\phic</code> prints the name of φ -calculus in both normal and the anonymous mode of <code>acmart</code>	20	<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow.	21	
	<code>\phiConst</code> : New command <code>\phiConst</code> added to denote a link to a constant attribute.	21	<code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket.	21	
	<code>\phiDotted</code> : New command <code>\phiDotted</code> added to denote a link to a special attribute.	21	0.6.0	General: Package option <code>nocomments</code> added in order to enable comments suppression in temporary <code>.tex</code> files (may be pretty important for <code>.dtx</code> documents).	8

eolang-sodg.pl: The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations.	13	any text inside the <code>\text</code> macro is not processed.	9
0.7.0		eolang-sodg.pl: The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.	13
nodollar: Now it is possible to use dollar sign instead of the <code>\phiq</code> command.	12	<code>\phiOset</code> : New commands <code>\phiOset</code> and <code>\phiUset</code> help position text over and under an arrow.	21
eolang-phi.pl: New syntax sugar for Φ , just using capital “Q” is enough. .	9	<code>\phiSaveTo</code> : The output of the <code>phiqutation</code> environment can be redirected to a file.	11
Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols.	9	<code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file.	19
Text in quotes is automatically converted to <code>\texttt</code>	9	0.9.0	
0.8.0		<code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\\[-4pt]</code>	21
eolang-phi.pl: Inside <code>phiqutation</code>			

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			
$\backslash \$$	110, 202, 207, 211, 214, 589, 595	$\backslash \text{dashrightarrow}$...	655
$\backslash \%$	177, 206, 593	$\backslash \text{def}$	166, 169, 180, 198, 577, 585, 597
$\backslash *$	87, 89	$\backslash \Delta$	429
$\backslash +$	220, 347, 378, 384	$\backslash \text{detokenize}$	197
$\backslash -$	609, 610, 612	$\backslash \text{dimen}$ 649, 650, 652, 654, 661	
$\backslash .$	88, 90, 115, 125, 220	$\backslash \text{dp}$	661
$\backslash /$	86, 87	$\backslash \text{draw}$...	280, 501, 531, 553
$\backslash ?$	118	E	
$\backslash [$	86, 112	$\backslash \text{E}$..	359, 367, 375, 378, 384, 387, 388, 389, 390
$\backslash \{$ 57, 80, 92, 93, 111, 115, 128		$\backslash \text{end}$	153, 155, 158, 161, 187, 193, 475, 482, 502, 584
$\backslash \}$	57, 92, 93, 128	$\backslash \text{endinginput}$	160, 481
$\backslash]$	86, 113	$\backslash \text{eolang}$	599
$\backslash \sim$	111	$\backslash \text{eolang-phi.pl}$	23
$\backslash $	87, 126, 127, 183, 189, 225, 580	$\backslash \text{eolang-sodg.pl}$...	216
Numbers		$\backslash \text{eolang@lineno}$	18
$\backslash 2$..	86, 88, 90, 97, 115, 240	$\backslash \text{eolang@mdfive}$	
$\backslash 3$	88, 90	... 19, 169, 198, 585	
$\backslash 4$	88	$\backslash \text{eolang@nocomments}$	
A		... 13, 177, 206, 593	
$\backslash \text{a}$ 526, 529, 537, 548, 551, 558		$\backslash \text{eolang@nodollar}$..	
$\backslash \text{active}$.	207, 211, 214, 595 202, 207, 209	
$\backslash \alpha$	609	$\backslash \text{eolang@phiSaveTo}$.	
$\backslash \text{anon}$	600, 601, 608, 609, 616, 617 166, 178, 180	
$\backslash \text{AtBeginDocument}$..	214	$\backslash \text{eolang@process}$...	
B	 168, 187, 193	
$\backslash \text{Bbbk}$	3	$\backslash \text{eolang@sodgSaveTo}$	
$\backslash \text{begin}$	24, 132, 135, 137, 185, 191, 217, 268, 498, 582 577, 594, 597	
$\backslash \text{box}$	662	$\backslash \text{eolang@tmpdir}$	
C	 11, 17, 24, 163, 170, 171, 172, 173, 174, 176, 186, 192, 196, 199, 200, 201, 203, 204, 205, 217, 484, 583, 586, 587, 588, 590, 591, 592	
$\backslash \text{catcode}$		$\backslash \text{ex@}$	632, 636
183, 189, 202, 207, 211, 214, 580, 589, 595		F	
$\backslash \text{clipbox}$	653, 657	$\backslash \text{F}$	522, 524, 525, 537, 544, 546, 547, 558
$\backslash \text{color}$	507	$\backslash \text{FancyVerbLine}$	486
D		G	
$\backslash \text{d}$..	128, 526, 527, 537, 538, 548, 549, 558, 559	$\backslash \text{gdef}$	212
		H	
		$\backslash \text{hash}$...	169, 172, 173, 176, 198, 201, 203, 205, 585, 588, 590, 592
		$\backslash \text{hbox}$	633, 637
		$\backslash \text{height}$	654, 657, 658
		$\backslash \text{hspace}$	623, 624
		$\backslash \text{hss}$	656
		$\backslash \text{ht}$	661
		I	
		$\backslash \text{I}$	522, 523, 525, 537, 544, 545, 547, 558
		$\backslash \text{iexec}$...	17, 171, 173, 196, 200, 203, 587, 590
		$\backslash \text{ifdefined}$.	177, 178, 202, 206, 207, 209, 593, 594, 600, 608, 616
		$\backslash \text{ifluatex}$	12
		$\backslash \text{ifxetex}$	12
		J	
		$\backslash \text{jobname}$.	17, 170, 171, 172, 173, 176, 186, 192, 196, 199, 200, 201, 203, 205, 583, 586, 587, 588, 590, 592
		K	
		$\backslash \text{kern}$	632, 636
		L	
		$\backslash \lambda$	441
		$\backslash \text{leadsto}$	626
		$\backslash \text{limits}$	631, 635
		M	
		$\backslash \text{m@th}$...	649, 650, 655, 658
		$\backslash \text{makeatletter}$.	18, 20, 23, 165, 168, 195, 216, 576, 579, 629, 644
		$\backslash \text{makeatother}$..	18, 22, 164, 167, 194, 208, 485, 578, 598, 638, 665
		$\backslash \text{makebox}$	652
		$\backslash \text{mapsto}$	624
		$\backslash \text{mapstochar}$.	624, 626, 646
		$\backslash \text{mathop}$	631, 635

<code>\mathpalette</code>	646	<code>\pi</code>	284	<code>\texttt</code>	430
<code>\mathrel</code>	623, 624, 626, 631, 635, 646	<code>\ProcessPgfOptions</code> .	16	<code>\tikz</code>	487
<code>\message</code>	162, 483	<code>\protected</code>	212	<code>\tikzinputsegmentfirst</code>	523, 545
<code>\mspace</code>	626			<code>\tikzinputsegmentlast</code>	524, 546
N		Q		<code>\tikzmath</code>	521, 543
<code>\newcommand</code>	21, 166, 168, 195, 577, 599, 607, 615, 622, 625, 627, 630, 634, 639, 641, 645, 647	<code>\Q</code>	359, 367, 375, 378, 384, 387, 388, 389, 390	<code>\tikzset</code>	515
<code>\newcounter</code>	18			<code>\tikzstyle</code>	503, 506, 509, 511, 512, 514, 568, 569, 570, 573
<code>\newenvironment</code> . . .	182, 188, 497, 579	<code>\RequirePackage</code> . . .	1, 2, 3, 4, 5, 6, 7, 8, 19, 487, 606, 642, 643	<code>\ttfamily</code>	507
<code>\node</code>	319, 330, 333, 336, 343, 406	<code>\resizebox</code>	654	<code>\tw@</code>	650, 661
<code>\nodollar</code>	209	<code>\rightarrow</code>	649, 650, 658	U	
<code>\noindent</code>	498			<code>\usetikzlibrary</code> . . .	488, 489, 490, 491, 492, 493, 494, 495, 496
P		S		V	
<code>\pdf@filemdfivesum</code> .	21	<code>\sbox</code>	651	<code>\v</code>	522, 525, 544, 547
<code>\pgfkeys</code>	9	<code>\scriptscriptstyle</code>	633, 637	<code>\value</code> 179, 184, 190, 581, 596	
<code>\Phi</code>	231	<code>\scriptsize</code>	572, 575	<code>\varphi</code>	610, 612
<code>\phic</code>	606	<code>\scshape</code>	628	<code>\vbox</code>	632, 636
<code>\phiConst</code>	622	<code>\setcounter</code>	179, 184, 190, 486, 581, 596	<code>\VerbatimEnvironment</code>	183, 189, 580
<code>\picture</code>	497	<code>\settoheight</code>	650	<code>\vss</code>	633, 637
<code>\phiDotted</code>	429, 441, 642	<code>\settowidth</code>	649	<code>\vx</code>	528, 529, 550, 551
<code>\phiDotted@</code>	646, 647	<code>\sffamily</code>	601, 603, 628	<code>\vy</code>	528, 550
<code>\phiEOL</code>	641	<code>\small</code>	507, 510	W	
<code>\phiMany</code>	639	<code>\sodg</code>	579	<code>\width</code>	653, 657
<code>\phiOset</code>	629, 640	<code>\sodgSaveTo</code>	576	X	
<code>\phiq</code>	195, 212	<code>\sxy</code>	389	<code>\xmir</code>	615
<code>\phiquation</code>	168	T		<code>\xrightarrow</code>	628
<code>\phiSaveTo</code>	165	<code>\t</code> 32, 246, 526, 528, 537, 538, 548, 550, 558, 559		Z	
<code>\phiSlot</code>	627	<code>\texorpdfstring</code> . . .	609, 610, 612	<code>\z@</code> 649, 651, 652, 654, 661, 662	
<code>\phiUset</code>	634, 640	<code>\text</code>	429, 628		
<code>\phiWave</code>	625	<code>\textnormal</code>	430		