



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

Yegor Bugayenko
yegor256@gmail.com

2022-11-29, 0.8.0

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.

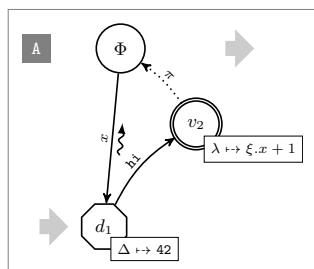
Object names are automatically converted to fixed-width font too, if they have more than one letter.

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

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

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

Without the anonymous option there will be no orange color:

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

```
3 \usepackage{eolang}
4 \begin{document}
5 In our research we use \eolang{ }, \
6 an experimental object-oriented \
7 dataflow language, \phic{ }, as its \
8 formal foundation, and \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}
```

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:

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

3 More Examples

The `phiq` 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:

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

```

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

```

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

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

```

6 \begin{phiquation*}
7 \dfrac{\begin{split}
8 x->[[@->y, z->42,
9 0/g->?, 1/foo->42]]
\end{split}}{\begin{split}
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{phiquation*}

```

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

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

```

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

```

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

$$\begin{array}{l} \text{Discriminant can be calculated using the following simple formula:} \\ D = b^2 - 4ac. \quad (1) \\ \text{Eq. 1 is also widely used in number theory and polynomial factoring.} \end{array}$$

```

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

```

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

$\llbracket \alpha_0 \mapsto x \rrbracket$	This is formation
$\llbracket \alpha_0 \mapsto \emptyset \rrbracket$	Abstraction
$x(\Delta \mapsto 42)$	Application

```

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

```

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

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

```

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

```

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

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

```

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

```

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

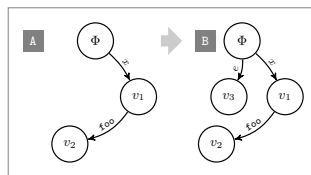
$\llbracket b \mapsto \emptyset \rrbracket$,
 $\llbracket \varphi \mapsto \text{TRUE}, \Delta \mapsto 42 \rrbracket$,
 $\Delta = 43-09$.

```

5 \begin{phiuation*}
6 [[ b -> ? ]],
7 [[ @ -> TRUE, \Delta \mapsto 42 ]], \ \backslash
8 \Delta = |43-09|.
9 \end{phiuation*}

```

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

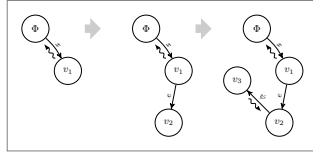


```

5 \begin{sodg}
6 v0 \ \backslash \ 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:

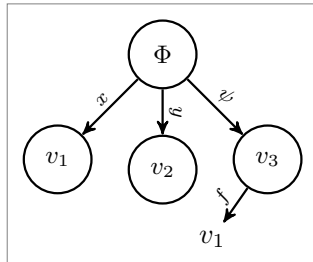


```

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:

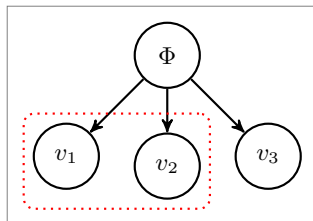


```

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 \ProcessPgfPackageOptions{/eolang}
```

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

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

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

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

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

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

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

```
24 \makeatletter
25 \begin{VerbatimOut}{\eolang@tmpdir/eolang-phi.pl}
26 $macro = $ARGV[0];
27 open(my $fh, '<', $ARGV[1]);
28 my $tex; { local $/; $tex = <$fh>; }
29 print '% This file is auto-generated', "\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/g;
57   $tex =~ s/\\\\n\s*//g;
58   $tex =~ s/\n*(\\label\{[^\}]+\})\n*/1/g;
59   $tex =~ s/\n{3,}/n/g;
60 }
61 my @texts = ();
62 sub trep {
63   my ($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]+|\\1[a-z]+|\\1)
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\\}/\\end{split}/g;
95   $tex =~ s/\\n\\n\\\\\\&/g;
96   $tex =~ s/\\n/\\phiEOL{\\n}/g;

```

```

97 $tex =~ s/\\\\\\\\\\\\\\\\n/g;
98 $tex =~ s/([^\s])\s{2}([^\s])/1 \2/g;
99 $tex =~ s/\s{2}/ \quad{/g;
100 $tex = '&' . $tex;
101 my $lead = '[^\s]+\s(?:->|:=|=';
102 my @leads = $tex =~ /&{\lead}/g;
103 my @eols = $tex =~ /\&/g;
104 if (0+@leads == 0+@eols && 0+@eols > 1) {
105     $tex =~ s/&({\lead})/\1&/g;
106     $gathered = 0;
107     print '% The "gathered" is NOT used because all ' .
108         (0+@eols) . ' lines are ' . (0+@leads) . " leads\n";
109 }
110 }
111 $tex =~ s/\$/\xi{/g;
112 $tex =~ s/(?<!\{)\^\\\\rho{/g;
113 $tex =~ s/[\\[\\llbracket\\mathrel{/g;
114 $tex =~ s/[\\]\\mathrel{\\rrbracket}/g;
115 $tex =~ s/([\\s,>()]{0-9A-F}{2}(?:-[0-9A-F]{2})+|
116 [0-9]+(?:\\. [0-9]+)?)?(?!\\{)/\\1|\\2|/xg;
117 $tex =~ s/TRUE/|TRUE|/g;
118 $tex =~ s/FALSE/|FALSE|/g;
119 $tex =~ s/\\?/\\varnothing{/g;
120 $tex =~ s/@/\\varphi{/g;
121 $tex =~ s/-([a-z]+)>/\\mathrel{\\phiSlot{\\1}}/g;
122 $tex =~ s/!->/\\mathrel{\\phiConst}/g;
123 $tex =~ s/->/\\mathrel{\\mapsto}/g;
124 $tex =~ s/~>/\\mathrel{\\phiWave}/g;
125 $tex =~ s/:=/\\mathrel{\\vDash}/g;
126 $tex =~ s/\\.\\.>/\\mathrel{\\phiDotted}/g;
127 $tex =~ s/\\{2,\\}/|/g;
128 $tex =~ s/\\|([^\|]+)\\|/\\textnormal{\\texttt{\\1}}{/g;
129 $tex =~ s/\\{TEXT(d+)}\\}'\\text{' . @texts[\\$1] . '}'/ge;
130 if ($macro eq 'phiq') {
131     print '$' if ($tex ne '');
132 } else {
133     print '\\begin{' , $macro, "}\\n";
134     if (not($align)) {
135         if ($gathered) {
136             print '\\begin{gathered}';
137         } else {
138             print '\\begin{split}';
139         }
140     }
141     print "\\n";
142 }
143 if ($gathered and not($align)) {
144     $tex =~ s/^&/g;
145     $tex =~ s/\\n&/\\n/g;
146 }
147 print $tex;
148 if ($macro eq 'phiq') {
149     print '$' if ($tex ne '');
150 } else {

```

```

151 if (not($align)) {
152   print "\n";
153   if ($gathered) {
154     print '\end{gathered}';
155   } else {
156     print '\end{split}';
157   }
158 }
159 print "\n" . '\end{' . $macro . '}';
160 }
161 print '\endinput';
162 \end{VerbatimOut}
163 \message{eolang: File with Perl script
164   '\eolang@tmpdir/eolang-phi.pl' saved^^J}%
165 \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:

```

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

```

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

```

169 \makeatletter\newcommand\eolang@process[1]{
170   \def\hash{\eolang@mdfive
171     {\eolang@tmpdir/\jobname/phiquation.tex}}%
172   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiquation.tex"
173     "\eolang@tmpdir/\jobname/\hash.tex"}%
174   \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
175   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
176     perl "\eolang@tmpdir/eolang-phi.pl"
177     '#1'
178     "\eolang@tmpdir/\jobname/\hash.tex"
179     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
180     \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
181   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
182   \def\eolang@phiSaveTo{\relax}%
183 }
184 \newenvironment{phiquation*}%
185 {\catcode'\|=12 \VerbatimEnvironment%
186 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
187 \begin{VerbatimOut}
188   {\eolang@tmpdir/\jobname/phiquation.tex}}
189 {\end{VerbatimOut}\eolang@process{equation*}}
190 \newenvironment{phiquation}%
191 {\catcode'\|=12 \VerbatimEnvironment%
192 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
193 \begin{VerbatimOut}
194   {\eolang@tmpdir/\jobname/phiquation.tex}}
195 {\end{VerbatimOut}\eolang@process{equation*}}
196 \makeatother

```

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

```

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

```

nodollar Then, we redefine dollar sign:

```

211 \ifdefined\eolang@nodollar\else
212   \begingroup
213   \catcode'\$=\active
214   \protected\gdef$#1${\phiq{#1}}
215   \endgroup
216   \AtBeginDocument{\catcode'\$=\active}
217 \fi

```

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

```

218 \makeatletter
219 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
220 sub num {
221   my ($i) = @_;
222   $i =~ s/(\+|-)\./\10./g;
223   return $i;
224 }
225 sub fmt {
226   my ($tex) = @_;
227   $tex =~ s/\\|([^\|]+)\\|\\textnormal{\texttt{\1}}/g;
228   return $tex;
229 }
230 sub vertex {
231   my ($v) = @_;
232   if (index($v, 'v0') == 0) {
233     return '\Phi';
234   } else {
235     $v =~ s/^v/v_/g;
236     $v =~ s/[0-9]$/g;
237     return $v;
238   }
239 }
240 sub tailor {
241   my ($t, $m) = @_;
242   $t =~ s/<([A-Z]?[A-Z]?):([>]+)>/\2/g;
243   $t =~ s/<[A-Z]+:[>]+>/g;
244   return $t;
245 }

```

```

246 open(my $fh, '<', $ARGV[0]);
247 my $tex; { local $/; $tex = <$fh>; }
248 if (index($tex, "\t") > 0) {
249     print "TABS are prohibited!";
250     exit 1;
251 }
252 print '% This file is auto-generated', "\n%\n";
253 print '% --- there are ', length($tex),
254     ' chars in the input (', $ARGV[0], "):\n";
255 foreach my $t (split (/\\n/g, $tex)) {
256     print '% ', $t, "\n";
257 }
258 print "% ---\n";
259 $tex =~ s/\\\\\\n/g;
260 $tex =~ s/\\\\n//g;
261 $tex =~ s/(\\[a-zA-Z]+)\\s+//1/g;
262 $tex =~ s/\\n{2,}/\\n/g;
263 my @cmds = split(/\\n/g, $tex);
264 print '% --- before processing:' . "\n";
265 foreach my $t (split (/\\n/g, $tex)) {
266     print '% ', $t, "\n";
267 }
268 print '% ---';
269 print ' (' . (0+@cmds) . " lines)\n";
270 print '\\begin{picture}', "\n";
271 for (my $c = 0; $c < 0+@cmds; $c++) {
272     my $cmd = $cmds[$c];
273     $cmd =~ s/^\\s+//g;
274     $cmd =~ s/%.*//g;
275     my ($head, $tail) = split(/ /, $cmd, 2);
276     my %opts = {};
277     foreach my $p (split(/ /, $tail)) {
278         my ($q, $t) = split(/:/, $p);
279         $opts{$q} = $t;
280     }
281     if (index($head, '->') >= 0) {
282         my $draw = '\\draw[';
283         if (exists $opts{'pi'}) {
284             $draw = $draw . '<MB:phi-pi><F:draw=none>';
285             if (not exists $opts{'a'}) {
286                 $opts{'a'} = '\\pi';
287             }
288         }
289         if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
290             $draw = $draw . '<MB:.,phi-rho>';
291         }
292         $draw = $draw . ']';
293         my ($from, $to) = split (/->/, $head);
294         $draw = $draw . " ($from) ";
295         if (exists $opts{'bend'}) {
296             $draw = $draw . 'edge [F:draw=none><MF:.,bend right=' .
297                 num($opts{'bend'}) . '>';
298             if (exists $opts{'rho'}) {
299                 $draw = $draw . '<MB:.,phi-rho>';

```

```

300     }
301     $draw = $draw . ']' ;
302 } else {
303     $draw = $draw . '--' ;
304 }
305 if (exists $opts{'a'}) {
306     my $a = $opts{'a'} ;
307     if (index($a, '$') == -1) {
308         $a = '$' . fmt($a) . '$' ;
309     } else {
310         $a = fmt($a) ;
311     }
312     $draw = $draw . '<MB: node [phi-attr] {' . $a . '}>' ;
313 }
314 if (exists $opts{'break'}) {
315     $draw = $draw . '<F: coordinate [pos=' .
316         ($opts{'break'} / 100) . ']' (break)>' ;
317 }
318 $draw = $draw . " (<MF:${to}><B:break-v>)" ;
319 if (exists $opts{'break'}) {
320     print tailord($draw, 'F') . ";\n";
321     print ' \node[outer sep=.1cm,inner sep=0cm] ' .
322         'at (break) (break-v) {' . vertex($to) .
323         '$};' . "\n";
324     print ' ' . tailord($draw, 'B');
325 } else {
326     print tailord($draw, 'M');
327 }
328 } elsif (index($head, '=>') >= 0) {
329     my ($from, $to) = split (/=>/, $head);
330     my $size = () = $head =~ /=/g;
331     if ($from eq '') {
332         print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
333             $to . '.center]';
334     } elsif ($to eq '') {
335         print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
336             $from . '.center]';
337     } else {
338         print '\node [phi-arrow] at ($(' .
339             $from . ')!0.5!(' . $to . ')$)';
340     }
341     print '{}';
342 } elsif (index($head, '!'') >= 0) {
343     my ($v, $marker) = split (/!+/, $head);
344     my $size = () = $head =~ /=/g;
345     print '\node [phi-marker, left=' .
346         ($size * 0.6) . 'cm of ' .
347         $v . '.center]{' . fmt($marker) . '}' ;
348 } elsif (index($head, '+') >= 0) {
349     my ($v, $suffix) = split (/+/, $head);
350     my @friends = ($v);
351     foreach my $c (@cmds) {
352         $e = $c;
353         $e =~ s/^\s+//g;

```

```

354     my $h = $e;
355     $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
356     foreach my $f (@friends) {
357         my $add = '';
358         if (index($h, $f . '->') >= 0) {
359             $add = substr($h, index($h, '->') + 2);
360         }
361         if ($h =~ /->\Q${f}\E$/) {
362             $add = substr($h, 0, index($h, '->'));
363         }
364         if (index($e, ' xy:' . $f . ',') >= 0) {
365             $add = $h;
366         }
367         if (index($add, '+') == -1
368             and $add ne ''
369             and not(grep(/^Q${add}\E$/, @friends))) {
370             push(@friends, $add);
371         }
372     }
373 }
374 my @extra = ();
375 foreach my $e (@cmds) {
376     $m = $e;
377     if ($m =~ /\s*\Q${v}\E\s/) {
378         next;
379     }
380     if ($m =~ /\s*[^\s]+\s/ and not($m =~ /\s*\Q${head}\E\s/)) {
381         next;
382     }
383     foreach my $f (@friends) {
384         my $h = $f;
385         $h =~ s/[a-z]$//g;
386         if ($m =~ s/^(s*)\Q${f}\E\+\Q${suffix}\E\s?/\1${h}${suffix} /g) {
387             last;
388         }
389         $m =~ s/^(s*)\Q${f}\E\s/\1${h}${suffix} /g;
390         $m =~ s/^(s*)\Q${f}\E->/\1${h}${suffix}->/g;
391         $m =~ s/\sxy:\Q${f}\E,/ xy:${h}${suffix},/g;
392         $m =~ s/->\Q${f}\E\s/->${h}${suffix} /g;
393     }
394     if ($m ne $e) {
395         push(@extra, ' ' . $m);
396     }
397 }
398 splice(@extra, 0, 0, @extra[-1]);
399 splice(@extra, -1, 1);
400 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
401     ' ), friends: [' . join(', ', @friends) . ' ] in ' .
402     (0+@cmds) . ' lines');
403 splice(@cmds, $c, 1, @extra);
404 print '% cloned ' . $v . ' at line no.' . $c .
405     ' (' . (0+@extra) . ' lines -> ' .
406     (0+@cmds) . ' lines total)';
407 } elsif ($head =~ /\v[0-9]+[a-z]?$/ ) {

```



```

408 print '\node[';
409 if (exists $opts{'xy'}) {
410     my ($v, $right, $down) = split(/,/ , $opts{'xy'});
411     my $loc = '';
412     if ($down > 0) {
413         $loc = 'below';
414     } elsif ($down < 0) {
415         $loc = 'above';
416     }
417     if ($right > 0) {
418         $loc = $loc . 'right';
419     } elsif ($right < 0) {
420         $loc = $loc . 'left';
421     }
422     print ', ' . $loc . '=';
423     print abs(num($down)) . 'cm and ' .
424         abs(num($right)) . 'cm of ' . $v . '.center';
425 }
426 if (exists $opts{'data'}) {
427     print ',phi-data';
428     if (not $opts{'data'} eq '') {
429         my $d = $opts{'data'};
430         if (index($d, '|') == -1) {
431             $d = '$\Delta\phiDotted\text{' .
432                 '\textnormal{\texttt{' . fmt($d) . '}}}$';
433         } else {
434             $d = fmt($d);
435         }
436         $opts{'box'} = $d;
437     }
438 } elsif (exists $opts{'atom'}) {
439     print ',phi-atom';
440     if (not $opts{'atom'} eq '') {
441         my $a = $opts{'atom'};
442         if (index($a, '$') == -1) {
443             $a = '$\lambda\phiDotted{' . fmt($a) . '$';
444         } else {
445             $a = fmt($a);
446         }
447         $opts{'box'} = $a;
448     }
449 } else {
450     print ',phi-object';
451 }
452 print ']';
453 print ' (' . $head . ')';
454 print '{';
455 if (exists $opts{'tag'}) {
456     my $t = $opts{'tag'};
457     if (index($t, '$') == -1) {
458         $t = '$' . $t . '$';
459     } else {
460         $t = fmt($t);
461     }

```

```

462     print $t;
463   } else {
464     print '$' . vertex($head) . '$';
465   }
466   print '}}';
467   if (exists $opts{'box'}) {
468     print ' node[phi-box] at (';
469     print $head, '.south east) {';
470     print $opts{'box'}, '}}';
471   }
472 } else {
473   print $cmd;
474 }
475 print ";\n";
476 }
477 print '\end{picture}%', "\n";
478 print "% --- after processing:\n%";
479 foreach my $c (@cmds) {
480   print '% ', $c, "\n";
481 }
482 print '% --- (' . (0+@cmds) . " lines)\n";
483 print '\endinput';
484 \end{VerbatimOut}
485 \message{eolang: File with Perl script
486   '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
487 \makeatother

```

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

```

488 \setcounter{FancyVerbLine}{0}

```

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

```

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

```

picture Then, we define internal environment `picture`:

```

499 \newenvironment{picture}%
500   {\noindent\begin{tikzpicture}[
501     ->,>=stealth',node distance=0,thick,
502     pics/parallel arrow/.style={
503       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
504   {\end{tikzpicture}}
505 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
506   minimum height=0.5cm, minimum width=0.5cm,
507   single arrow head extend=2mm]
508 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,

```

```

509 minimum width=1.4em, font={\small\color{white}\ttfamily},
510 fill=gray]
511 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
512 draw,font={\small}]
513 \tikzstyle{phi-object} = [phi-thing,circle]
514 \tikzstyle{phi-data} = [phi-thing,regular polygon,
515 regular polygon sides=8]
516 \tikzstyle{phi-empty} = [phi-object]
517 \tikzset{%
518 phi-rho/.style={
519 postaction={%
520 decoration={
521 show path construction,
522 curveto code={
523 \tikzmath{
524 coordinate \I, \F, \v;
525 \I = (\tikzinputsegmentfirst);
526 \F = (\tikzinputsegmentlast);
527 \v = ($(\I) -(\F)$);
528 real \d, \a, \r, \t;
529 \d = 0.8;
530 \t = atan2(\vy, \vx);
531 if \vx<0 then { \a = 90; } else { \a = -90; };
532 {
533 \draw[arrows={-latex}, decorate,
534 decoration={%
535 snake, amplitude=.4mm,
536 segment length=2mm,
537 post length=1mm
538 }]}
539 ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
540 -- ++(\t: 2*\d em);
541 };
542 }
543 },
544 lineto code={
545 \tikzmath{
546 coordinate \I, \F, \v;
547 \I = (\tikzinputsegmentfirst);
548 \F = (\tikzinputsegmentlast);
549 \v = ($(\I) -(\F)$);
550 real \d, \a, \r, \t;
551 \d = 0.8;
552 \t = atan2(\vy, \vx);
553 if \vx<0 then { \a = 90; } else { \a = -90; };
554 {
555 \draw[arrows={-latex}, decorate,
556 decoration={%
557 snake, amplitude=.4mm,
558 segment length=2mm,
559 post length=1mm]}
560 ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
561 -- ++(\t: 2*\d em);
562 };

```

```

563         }
564     }
565 },
566     decorate
567 }
568 }
569 }
570 \tikzstyle{phi-pi} = [draw,dotted]
571 \tikzstyle{phi-atom} = [phi-object,double]
572 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
573     rectangle,thin,minimum width=1.2em,anchor=north west,
574     font={\scriptsize}]
575 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
576     above=2pt,sloped/.append style={transform shape},
577     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:

```

578 \makeatletter
579 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
580 \makeatother

```

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

```

581 \makeatletter\newenvironment{sodg}%
582 {\catcode'\|=12 \VerbatimEnvironment%
583 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
584 \begin{VerbatimOut}
585   {\eolang@tmpdir/\jobname/sodg.tex}}
586 {\end{VerbatimOut}}%
587 \def\hash{\eolang@mdfive
588   {\eolang@tmpdir/\jobname/sodg.tex}}%
589 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
590   "\eolang@tmpdir/\jobname/\hash.tex"}%
591 \catcode'\$=3 %
592 \message{Start parsing 'sodg' at line no. \the\inputlineno^^J}
593 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
594   perl "\eolang@tmpdir/eolang-sodg.pl"
595   "\eolang@tmpdir/\jobname/\hash.tex"
596   \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
597   \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%
598 \catcode'\$active%
599 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
600 \def\eolang@sodgSaveTo{\relax}%
601 }\makeatother

```

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

```

602 \makeatletter
603 \NewExpandableDocumentCommand{\eolang@anon}{O{ANONYMIZED}m}{%
604   \ifdefined\eolang@anonymous%
605     \textcolor{orange}{#1}%
606   \else%
607     #2%
608   \fi%
609 }\makeatother

```

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

```

610 \makeatletter\newcommand\eolang{%
611   \eolang@anon[XYZ]{\sffamily EO}}
612 \makeatother

```

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

```

613 \RequirePackage{hyperref}
614 \makeatletter\newcommand\phic{%
615   \eolang@anon[\alpha$-cal\ -cu\ -lus]{\varphi$-cal\ -cu\ -lus}}
616 \makeatother

```

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

```

617 \makeatletter\newcommand\xmirl{%
618   \eolang@anon[XML$~+$]{XMIR}}
619 \makeatother

```

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

```

620 \newcommand\phiConst{%
621   \mathrel{\hspace{.15em}}%
622   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

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

```

623 \newcommand\phiWave{%
624   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

```

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

```

625 \newcommand\phiSlot[1]{%
626   \xrightarrow{\text{\sffamily\scshape #1}}}

```

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

```

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

```

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

```

637 \newcommand\phiMany[3]{%
638   \phiOset{#3}{\phiUset{#2}{#1}}}

```

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

```

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

```

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

```

640 \RequirePackage{trimclip}
641 \RequirePackage{amsfonts}
642 \makeatletter
643 \newcommand{\phiDotted}{%
644   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
645 \newcommand{\phiDotted@}[2]{%
646   \begingroup%
647   \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
648   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
649   \sbox\z@{%
650     \makebox[\dimen\z@][s]{%
651       \clipbox{0 0 {0.4\width} 0}%
652       {\resizebox{\dimen\z@}{\height}%
653        {\m@th#1\dashrightarrow}}}%
654     \hss%
655     \clipbox{{0.69\width} {-0.1\height} 0
656      {-\height}}{\m@th#1\rightarrow}%
657   }%
658 }%
659 \ht\z@=\dimen\tw@ \dp\z@=\z@%
660 \box\z@%
661 \endgroup%
662 }
663 \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	eolang-phi.pl: Numbers automatically render as \texttt. No need to use vertical bars around them anymore.	9
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.	20		eolang-sodg.pl: The content of the atom and the data boxes is parsed automatically as formulas and numbers, respectively.	13
	eolang-phi.pl: New symbol added for basket slots	9		\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)	9	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	9		eolang-phi.pl: New arrow added, that looks like \leadsto.	9
	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.	13		\phiWave: New command \phiWave added to denote a link to a multi-layer attribute.	21
	\phiq: Parsing of additional symbols enabled.	12	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.	13
0.1.0	General: Parsing of package options introduced.	9		Relative positioning of vertices fixed.	13
	\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.	9
	\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.	13
	eolang-phi.pl: A new Perl script “eolang-phi.pl” added for parsing of phi expressions.	9		New syntax introduced that allows to make clones of vertices and all their dependants.	13
	eolang-sodg.pl: There are two Perl scripts now: one for phiuation, another one for sodg.	13		Now edges may have the break attribute, to make them shorter.	13
	\phic: New command \phic prints the name of φ -calculus in both normal and the anonymous mode of acmart.	21		\phiMany: New command \phiMany enables iterating over an arrow.	21
	\phiConst: New command \phiConst added to denote a link to a constant attribute.	21		\phiSlot: New command \phiSlot added to denote a link to a slot in a basket.	21
	\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.	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.	13	<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.	12
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.	20
Text in quotes is automatically converted to <code>\texttt</code>	9	0.9.0	
0.8.0		<code>\eolang@anon</code> : New command <code>\eolang@anon</code> added.	20
General: The <code>anonymous</code> package option added.	9	<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		H	
$\backslash \$$	111, 204, 209, 213, 216, 591, 598	$\backslash \Delta$	431
$\backslash \%$	179, 208, 596	$\backslash \detokenize$	199
$\backslash *$	88, 90	$\backslash \dimen$ 647, 648, 650, 652, 659	
$\backslash +$	222, 349, 380, 386	$\backslash dp$	659
$\backslash -$	615	$\backslash draw$	282, 503, 533, 555
$\backslash .$	89, 91, 116, 126, 222	E	
$\backslash /$	87, 88	$\backslash E$	361, 369, 377, 380, 386, 389, 390, 391, 392
$\backslash ?$	119	$\backslash end$	154,
$\backslash [$	87, 113		156, 159, 162, 189,
$\backslash \{$ 58, 81, 93, 94, 112, 116, 129			195, 477, 484, 504, 586
$\backslash \}$	58, 93, 94, 129	$\backslash endinput$	161, 483
$\backslash]$	87, 114	$\backslash eolang$	610
$\backslash ^$	112	$\backslash eolang\text{-}\phi . pl$	24
$\backslash $	88, 127, 128, 185, 191, 227, 582	$\backslash eolang\text{-}sodg . pl$	218
Numbers		$\backslash eolang@anon$	
$\backslash 2$	87, 89, 91, 98, 116, 242		602, 611, 615, 618
$\backslash 3$	89, 91	$\backslash eolang@anonymous$ 14, 604	
$\backslash 4$	89	$\backslash eolang@lineno$	19
A		$\backslash eolang@mdfive$	
$\backslash a$ 528, 531, 539, 550, 553, 560			20, 170, 200, 587
$\backslash active$	209, 213, 216, 598	$\backslash eolang@nocomments$	
$\backslash alpha$	615		13, 179, 208, 596
$\backslash AtBeginDocument$	216	$\backslash eolang@nodollar$	204, 209, 211
B		$\backslash eolang@phiSaveTo$	167, 180, 182
$\backslash Bbbk$	3	$\backslash eolang@process$	
$\backslash begin$	25, 133, 136, 138, 187, 193, 219, 270, 500, 584		169, 189, 195
$\backslash box$	660	$\backslash eolang@sodgSaveTo$	
C			579, 597, 600
$\backslash catcode$		$\backslash eolang@tmpdir$	
	185, 191, 204, 209, 213, 216, 582, 591, 598		11, 18, 25,
$\backslash clipbox$	651, 655		164, 171, 172, 173, 175, 176, 178, 188, 194, 198, 201, 202, 203, 205, 206, 207, 219, 486, 585, 588, 589, 590, 593, 594, 595
$\backslash color$	509	$\backslash ex@$	630, 634
D		F	
$\backslash d$	129, 528, 529, 539, 540, 550, 551, 560, 561	$\backslash F$	524, 526, 527, 539, 546, 548, 549, 560
$\backslash dashrightarrow$	653	$\backslash FancyVerbLine$	488
$\backslash def$	167, 170, 182, 200, 579, 587, 600	G	
E		$\backslash gdef$	214
I		J	
$\backslash I$	524, 525, 527, 539, 546, 547, 549, 560	$\backslash jobname$	18, 171, 172, 173, 175, 178, 188, 194, 198, 201, 202, 203, 205, 207, 585, 588, 589, 590, 593, 595
$\backslash iexec$	18, 172, 175, 198, 202, 205, 589, 593	K	
$\backslash ifdefined$		$\backslash kern$	630, 634
	179, 180, 204, 208, 209, 211, 596, 597, 604	L	
$\backslash ifluatex$	12	$\backslash lambda$	443
$\backslash ifxetex$	12	$\backslash leadsto$	624
$\backslash inputlineno$	174, 592	$\backslash limits$	629, 633
M		M	
$\backslash m@th$	647, 648, 653, 656	$\backslash m@th$	647, 648, 653, 656
$\backslash makeatletter$	19, 21, 24, 166, 169, 197, 218, 578, 581, 602, 610, 614, 617, 627, 642	$\backslash makeatletter$	19, 21, 165, 168, 196, 210, 487, 580, 601, 609, 612, 616, 619, 636, 663
$\backslash makeatother$	19, 23, 165, 168, 196, 210, 487, 580, 601, 609, 612, 616, 619, 636, 663	$\backslash makebox$	650

