



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

Yegor Bugayenko
yegor256@gmail.com

2023-01-22, 0.10.0

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 ” (`\rightsquigarrow`),
- “!->” maps to “ \multimap ” (`\multimap`),
- “. .>” maps to “ \vdash ” (`\vdash`),
- “D>” maps to “ $\Delta \vdash$ ” (`\Delta \vdash`),
- “L>” maps to “ $\lambda \vdash$ ” (`\lambda \vdash`),
- “[[” maps to “ \llbracket ” (`\llbracket`),
- “]]” maps to “ \rrbracket ” (`\rrbracket`),
- “==” maps to “ \equiv ” (`\equiv`),
- “|abc|” maps to “ abc ” (`\text{abc}`).

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

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

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

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

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

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

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

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

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

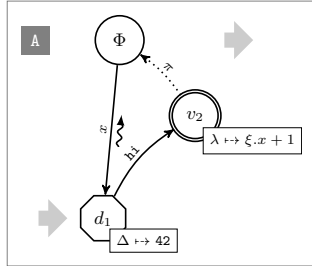
A simple object $x \mapsto [\varphi \mapsto y]$
is a decorator of the data object
 $y \mapsto [\Delta \mapsto 42]$.

```

4 \begin{document}
5 A simple object
6 \phiq{x -> [[@ -> y]]} \\\
7 is a decorator of
8 the data object \\\
9 $y -> [[\Delta ..> 42]]$.
10 \end{document}

```

`sodg (env.)` The environment `sodg` allows you to draw a [SODG](#) graph:



```

1 \documentclass{standalone}
2 \usepackage{eolang}
3 \begin{document}
4 \begin{sodg}
5 v0 \\\ v0==> \\\ v0!!A
6 v1 xy:v0,-.8,2.8 data:42 tag:d_1
7 v0->v1 a:x rho \\\ =>v1
8 v2 xy:v0,+1,+1 atom:\xi.x+1
9 v1->v2 a:|hi| bend:-15
10 v2->v0 pi bend:10 % a comment
11 \end{sodg}
12 \end{document}

```

The content of the environment is parsed line by line. Markers in each line are separated by a single space. The first marker is either a unique name of a vertex, like “v1” in the example above, or an edge, like “v0->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 “ $v3 \Rightarrow$ ”, of from the left to the vertex, by saying for example “ $\Rightarrow v5$.” If you want the arrow to stay further away from the vertex than usually, use a few “=” symbols, for example “ $=== \Rightarrow v0$.”

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

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

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

`\eolang` There is also a no-argument command `\eolang` to help you print the name of EO language. It understands the anonymous package option and prints itself differently, to `\phic` double-blind your paper. There is also `\phic` command to print the name of φ -calculus, `\xmirl` 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 $x \vdash y$ makes it a special attribute.

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

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

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

$\llbracket \mapsto^* \rrbracket$.

```
6 When the names of attributes and their
7 values don't matter, we use an arrow
8 with a star, for example:
9 \begin{phiuation*}
10 [[ \phi0set{*}{->} ]].
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 $\llbracket \alpha_i \stackrel{n}{\mapsto} x_i \rrbracket$ 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` `\sodgSaveTo` If you want to use `phiuation` or `sodg` environments inside `tabular` or any other environment or command, you won't be able to do this, because `phiuation` and `sodg` are “verbatim” environments. `\phiSaveTo` and `\sodgSaveTo` commands will help you in this situation. You use them right before `\begin{phiuation}` or `\begin{sodg}` respectively — the content of the equation or the graph won't be rendered, but instead saved to the file. Later, inside `tabular`, you can use it through the `\input` macro (don't forget the `\parbox`):

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

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

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

2 Package Options

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

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

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

package option:

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

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

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

3 More Examples

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

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

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

$\frac{\begin{array}{l} x \mapsto [\varphi \mapsto y, z \mapsto 42, \\ \alpha_0 g \mapsto \emptyset, \alpha_1 \text{foo} \mapsto 42] \\ x \mapsto [\varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi} (\\ \alpha_0 \mapsto [\psi \rightsquigarrow \text{hello}(12)], \\ \alpha_1 \mapsto 42)] \end{array}}{R2.}$	<pre> 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)]] \end{split}}\text{R2}. 14 \end{split}}\text{R2}. 15 \end{phiquation*} </pre>
--	--

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

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

The `cases` environment works too:

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

```

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

```

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

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

```

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

```

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

Discriminant can be calculated using the following simple formula:

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

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

```

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

```

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

$\alpha_0 \mapsto x$	This is formation
$\alpha_0 \mapsto \emptyset$	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 $\alpha_0 \mapsto x$ 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:

$\begin{aligned} x(\pi) &\mapsto [\lambda \mapsto f_1], \\ x(a, b, c) &\mapsto [\alpha_0 \mapsto \emptyset, \varphi \mapsto \text{hello}(\xi), x \mapsto \text{FALSE}], \\ \Delta &= 43-09, \\ x(y) &\equiv x(\alpha_0 \mapsto y). \end{aligned}$	<pre> 5 \begin{phiuation*} 6 x(\pi) -> [[\lambda \mapsto f_1]], \\ 7 x(a,b,c) -> [[\alpha_0 -> ?, \ 8 @ -> hello (\$), x -> FALSE]], \\ 9 \Delta = 43-09 , 10 x(y) == x(0-> y). 11 \end{phiuation*} </pre>
---	--

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

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

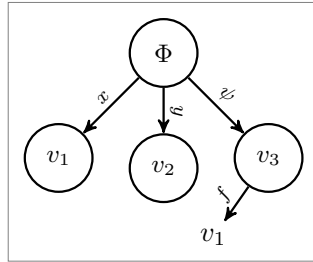
You can make a copy of a vertex together with its kids:

	<pre> 5 \begin{sodg} 6 v0 \\\ v0!!A 7 v1 xy:v0,.7,1 8 v0->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} </pre>
--	--

You can make a copy from a copy:

	<pre> 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} </pre>
--	--

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:

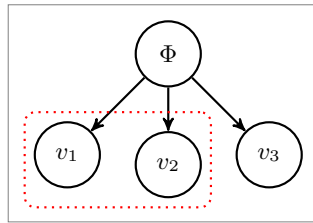


```

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 phiquation processing using VerbatimOut environ-
ment from fancyvrb:
24 \makeatletter
25 \begin{VerbatimOut}{\eolang@tmpdir/eolang-phi.pl}
26 $macro = $ARGV[0];
27 open(my $fh, '<', $ARGV[1]);
28 my $tex; { local $/; $tex = <$fh>; }
29 print '% This file is auto-generated', "\n";
30 print '% There are ', length($tex),
31   ' chars in the input: ', $ARGV[1], "\n";
32 print '% ---', "\n";
33 if (index($tex, "\t") > 0) {
34   print "TABS are prohibited!";
35   exit 1;
36 }
37 my @lines = split (/\\n/g, $tex);
38 foreach my $t (@lines) {
39   print '% ', $t, "\n";
40 }
41 print '% ---', "\n";
42 $tex =~ s/\\n\\n/g;
43 $tex =~ s/^\\s+|\\s+$//g;
44 my $gathered = (0 == $tex =~ /\\n\\s+/g);
45 if ($gathered) {
46   print '% The "gathered" is used since all lines are left-aligned' . "\n";
47 }
48 my $align = 0;
49 print '% The "align" is NOT used by default' . "\n";
50 if (index($tex, '&&') >= 0) {
51   $macro =~ s/equation/align/g;
52   $align = 1;
53   print '% The "align" is used because of && seen in the text' . "\n";
54 }
55 if ($macro ne 'phiq') {
56   $tex =~ s/\\\\\\n\\n\\n/g;
57   $tex =~ s/\\\\n\\s*/g;
58   $tex =~ s/\\n*(\\label\\{[\\}]+\\})\\n*/\\1/g;
59   $tex =~ s/\\n{3,}/\\n\\n/g;
60 }

```

```

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

```

```

115 $tex =~ s/&\/g;
116 return "\\begin{$env}" . $tex . "\\end{$env}";
117 }
118 foreach my $e (('matrix', 'cases')) {
119 $tex =~ s/\\begin{\(\\Q$e\E*?\)\\}(\\.+?)\\end{\\Q$e\E*?\\}/strip_tabs($1, $2)/sge;
120 }
121 }
122 $tex =~ s/\\$/\\xi{/g;
123 $tex =~ s/(?<!\{\\}\r\n)/g;
124 $tex =~ s/[\\[\\llbracket\\mathrel{}/g;
125 $tex =~ s/[\\]\\mathrel{\\rrbracket}/g;
126 $tex =~ s/([\\s,>()]{0-9A-F}{2}(?:-[0-9A-F]{2})+|
127 [0-9]+(?:\\. [0-9]+)?) (?!\\{\\})/1|2|/xg;
128 $tex =~ s/TRUE/|TRUE|/g;
129 $tex =~ s/FALSE/|FALSE|/g;
130 $tex =~ s/\\?/\\varnothing{/g;
131 $tex =~ s/@/\\varphi{/g;
132 $tex =~ s/-([a-z]+)/\\mathrel{\\phiSlot{\\1}}/g;
133 $tex =~ s/!->/\\mathrel{\\phiConst}/g;
134 $tex =~ s/->/\\mathrel{\\mapsto}/g;
135 $tex =~ s/~>/\\mathrel{\\phiWave}/g;
136 $tex =~ s/:=/\\mathrel{\\vDash}/g;
137 $tex =~ s/=\\mathrel{\\equiv}/g;
138 $tex =~ s/\\.\\.\\.\\mathrel{\\phiDotted}/g;
139 $tex =~ s/<</\\langle/g;
140 $tex =~ s/>>/\\rangle/g;
141 $tex =~ s/|{2,}|/g;
142 $tex =~ s/|([~|\\+)|\\|\\textnormal{\\texttt{\\1}}|}/g;
143 $tex =~ s/\\{TEXT(d+)}\\}/'\\text{' . @texts[$1] . '}';/ge;
144 if ($macro eq 'phiq') {
145   print '$' if ($tex ne '');
146 } else {
147   print '\\begin{' , $macro , "\\n";
148   if (not($align)) {
149     if ($gathered) {
150       print '\\begin{gathered}';
151     } else {
152       print '\\begin{split}';
153     }
154     print "\\n";
155   }
156 }
157 if ($gathered and not($align)) {
158   $tex =~ s/~&\/g;
159   $tex =~ s/\\n&/\\n/g;
160 }
161 print $tex;
162 if ($macro eq 'phiq') {
163   print '$' if ($tex ne '');
164 } else {
165   if (not($align)) {
166     print "\\n";
167   } if ($gathered) {
168     print '\\end{gathered}';

```

```

169 } else {
170   print '\end{split}';
171 }
172 }
173 print "\n" . '\end{' . $macro . '}'';
174 }
175 print '\endinput';
176 \end{VerbatimOut}
177 \message{eolang: File with Perl script
178   '\eolang@tmpdir/eolang-phi.pl' saved^^J}%
179 \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:

```

180 \makeatletter
181 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
182 \makeatother

```

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

```

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

```

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

```

211 \RequirePackage{xstring}
212 \makeatletter\newcommand\phiq[1]{%
213   \StrSubstitute{\detokenize{#1}}{'}{',''''}[\clean]%
214   \iexec[log,trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{

```

```

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

```

nodollar Then, we redefine dollar sign:

```

227 \ifdefined\eolang@nodollar\else
228   \begingroup
229   \catcode'\$=\active
230   \protected\gdef$#1${\phiq{#1}}
231   \endgroup
232   \AtBeginDocument{\catcode'\$=\active}
233 \fi

```

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

```

234 \makeatletter
235 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
236 sub num {
237   my ($i) = @_;
238   $i =~ s/(\+|-)\./\10./g;
239   return $i;
240 }
241 sub fmt {
242   my ($tex) = @_;
243   $tex =~ s/|([^\|]+)|\\textnormal{\texttt{\1}}/g;
244   return $tex;
245 }
246 sub vertex {
247   my ($v) = @_;
248   if (index($v, 'v0') == 0) {
249     return '\Phi';
250   } else {
251     $v =~ s/^v/v_/g;
252     $v =~ s/[0-9]$/g;
253     return $v;
254   }
255 }
256 sub tailor {
257   my ($t, $m) = @_;
258   $t =~ s/<([A-Z]?${m}[A-Z]?):([~>]+)>/\2/g;
259   $t =~ s/<[A-Z]+:[~>]+>/g;
260   return $t;
261 }
262 open(my $fh, '<', $ARGV[0]);
263 my $tex; { local $/; $tex = <$fh>; }

```

```

264 if (index($tex, "\t") > 0) {
265   print "TABS are prohibited!";
266   exit 1;
267 }
268 print '% This file is auto-generated', "\n%\n";
269 print '% --- there are ', length($tex),
270   ' chars in the input (', $ARGV[0], "):\n";
271 foreach my $t (split (/ \n/g, $tex)) {
272   print '% ', $t, "\n";
273 }
274 print "% ---\n";
275 $tex =~ s/\\\\\\/\n/g;
276 $tex =~ s/\\ \n//g;
277 $tex =~ s/(\\[a-zA-Z]+)\s+/\1/g;
278 $tex =~ s/\n{2,}/\n/g;
279 my @cmds = split(/ \n/g, $tex);
280 print '% --- before processing:' . "\n";
281 foreach my $t (split (/ \n/g, $tex)) {
282   print '% ', $t, "\n";
283 }
284 print '% ---';
285 print ' (' . (0+@cmds) . " lines)\n";
286 print '\begin{picture}', "\n";
287 for (my $c = 0; $c < 0+@cmds; $c++) {
288   my $cmd = $cmds[$c];
289   $cmd =~ s/^ \s+//g;
290   $cmd =~ s/%.*//g;
291   my ($head, $tail) = split(/ /, $cmd, 2);
292   my %opts = {};
293   foreach my $p (split(/ /, $tail)) {
294     my ($q, $t) = split(/:/, $p);
295     $opts{$q} = $t;
296   }
297   if (index($head, '->') >= 0) {
298     my $draw = '\draw[';
299     if (exists $opts{'pi'}) {
300       $draw = $draw . '<MB:phi-pi><F:draw=none>';
301       if (not exists $opts{'a'}) {
302         $opts{'a'} = '\pi';
303       }
304     }
305     if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
306       $draw = $draw . '<MB:,phi-rho>';
307     }
308     $draw = $draw . ']';
309     my ($from, $to) = split (/ -> /, $head);
310     $draw = $draw . " ($from) ";
311     if (exists $opts{'bend'}) {
312       $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
313         num($opts{'bend'}) . '>';
314       if (exists $opts{'rho'}) {
315         $draw = $draw . '<MB:,phi-rho>';
316       }
317       $draw = $draw . ']';

```

```

318 } else {
319     $draw = $draw . '--';
320 }
321 if (exists $opts{'a'}) {
322     my $a = $opts{'a'};
323     if (index($a, '$') == -1) {
324         $a = '$' . fmt($a) . '$';
325     } else {
326         $a = fmt($a);
327     }
328     $draw = $draw . '<MB: node [phi-attr] {' . $a . '}>';
329 }
330 if (exists $opts{'break'}) {
331     $draw = $draw . '<F: coordinate [pos=' .
332         ($opts{'break'} / 100) . ']' (break)>';
333 }
334 $draw = $draw . " (<MF:${to}><B:break-v>)" ;
335 if (exists $opts{'break'}) {
336     print tailor($draw, 'F') . ";\n";
337     print ' \node[outer sep=.1cm,inner sep=0cm] ' .
338         'at (break) (break-v) {' . vertex($to) .
339         '$};' . "\n";
340     print ' ' . tailor($draw, 'B');
341 } else {
342     print tailor($draw, 'M');
343 }
344 } elsif (index($head, '=') >= 0) {
345     my ($from, $to) = split (/=>/, $head);
346     my $size = () = $head =~ /=/g;
347     if ($from eq '') {
348         print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
349             $to . '.center]';
350     } elsif ($to eq '') {
351         print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
352             $from . '.center]';
353     } else {
354         print '\node [phi-arrow] at ($(' .
355             $from . ')!0.5!(' . $to . ')$)';
356     }
357     print '{}';
358 } elsif (index($head, '!') >= 0) {
359     my ($v, $marker) = split (/!+/, $head);
360     my $size = () = $head =~ /*!/g;
361     print '\node [phi-marker, left=' .
362         ($size * 0.6) . 'cm of ' .
363         $v . '.center']['' . fmt($marker) . ']' ;
364 } elsif (index($head, '+') >= 0) {
365     my ($v, $suffix) = split (/+/, $head);
366     my @friends = ($v);
367     foreach my $c (@cmds) {
368         $e = $c;
369         $e =~ s/^\s+//g;
370         my $h = $e;
371         $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;

```



```

372     foreach my $f (@friends) {
373         my $add = '';
374         if (index($h, $f . '->') >= 0) {
375             $add = substr($h, index($h, '->') + 2);
376         }
377         if ($h =~ /->\Q${f}\E/) {
378             $add = substr($h, 0, index($h, '->'));
379         }
380         if (index($e, ' xy:' . $f . ',') >= 0) {
381             $add = $h;
382         }
383         if (index($add, '+') == -1
384             and $add ne ''
385             and not(grep(/^Q${add}\E$/, @friends))) {
386             push(@friends, $add);
387         }
388     }
389 }
390 my @extra = ();
391 foreach my $e (@cmds) {
392     $m = $e;
393     if ($m =~ /^s*\Q${v}\E/s/) {
394         next;
395     }
396     if ($m =~ /^s*[^\s]+\+/ and not($m =~ /^s*\Q${head}\E/s/)) {
397         next;
398     }
399     foreach my $f (@friends) {
400         my $h = $f;
401         $h =~ s/[a-z]$/g;
402         if ($m =~ s/^(s*)\Q${f}\E+\Q${suffix}\E\s?/\1${h}${suffix} /g) {
403             last;
404         }
405         $m =~ s/^(s*)\Q${f}\E\s/\1${h}${suffix} /g;
406         $m =~ s/^(s*)\Q${f}\E->/\1${h}${suffix}->/g;
407         $m =~ s/\sxy:\Q${f}\E,/ xy:${h}${suffix},/g;
408         $m =~ s/->\Q${f}\E\s/->${h}${suffix} /g;
409     }
410     if ($m ne $e) {
411         push(@extra, ' ' . $m);
412     }
413 }
414 splice(@extra, 0, 0, @extra[-1]);
415 splice(@extra, -1, 1);
416 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
417     '), friends: [' . join(' ', @friends) . ']' in ' .
418     (0+@cmds) . ' lines');
419 splice(@cmds, $c, 1, @extra);
420 print '% cloned ' . $v . ' at line no.' . $c .
421     ' (' . (0+@extra) . ' lines -> ' .
422     (0+@cmds) . ' lines total)';
423 } elsif ($head =~ /^v[0-9]+[a-z]?$/ ) {
424     print '\node[';
425     if (exists $opts{'xy'}) {

```

```

426     my ($v, $right, $down) = split(/,/ , $opts{'xy'});
427     my $loc = '';
428     if ($down > 0) {
429         $loc = 'below ';
430     } elsif ($down < 0) {
431         $loc = 'above ';
432     }
433     if ($right > 0) {
434         $loc = $loc . 'right';
435     } elsif ($right < 0) {
436         $loc = $loc . 'left';
437     }
438     print ', ' . $loc . '=';
439     print abs(num($down)) . 'cm and ' .
440         abs(num($right)) . 'cm of ' . $v . '.center';
441 }
442 if (exists $opts{'data'}) {
443     print ',phi-data';
444     if ($opts{'data'} ne '') {
445         my $d = $opts{'data'};
446         if (index($d, '|') == -1) {
447             $d = '$\Delta\phiDotted\text{' .
448                 '\textnormal{\texttt{' . fmt($d) . '}}}'$';
449         } else {
450             $d = fmt($d);
451         }
452         $opts{'box'} = $d;
453     }
454 } elsif (exists $opts{'atom'}) {
455     print ',phi-atom';
456     if ($opts{'atom'} ne '') {
457         my $a = $opts{'atom'};
458         if (index($a, '$') == -1) {
459             $a = '$\lambda\phiDotted{' . fmt($a) . '$';
460         } else {
461             $a = fmt($a);
462         }
463         $opts{'box'} = $a;
464     }
465 } else {
466     print ',phi-object';
467 }
468 print ']';
469 print ' (' . $head . ')';
470 print '{';
471 if (exists $opts{'tag'}) {
472     my $t = $opts{'tag'};
473     if (index($t, '$') == -1) {
474         $t = '$' . $t . '$';
475     } else {
476         $t = fmt($t);
477     }
478     print $t;
479 } else {

```

```

480     print '$' . vertex($head) . '$';
481   }
482   print '>';
483   if (exists $opts{'box'}) {
484     print ' node[phi-box] at (';
485     print $head, '.south east) {';
486     print $opts{'box'}, '>';
487   }
488 } else {
489   print $cmd;
490 }
491 print ";\n";
492 }
493 print '\end{picture}%', "\n";
494 print "% --- after processing:\n%";
495 foreach my $c (@cmds) {
496   print '% ', $c, "\n";
497 }
498 print '% --- (' . (0+@cmds) . " lines)\n";
499 print '\endinput';
500 \end{VerbatimOut}
501 \message{eolang: File with Perl script
502   '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
503 \makeatother

```

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

```

504 \setcounter{FancyVerbLine}{0}

```

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

```

505 \RequirePackage{tikz}
506 \usetikzlibrary{arrows}
507 \usetikzlibrary{shapes}
508 \usetikzlibrary{decorations}
509 \usetikzlibrary{decorations.pathmorphing}
510 \usetikzlibrary{decorations.pathreplacing}
511 \usetikzlibrary{positioning}
512 \usetikzlibrary{calc}
513 \usetikzlibrary{math}
514 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment phicture:

```

515 \newenvironment{phicture}%
516 {\noindent\begin{tikzpicture}[
517   ->,>=stealth',node distance=0,thick,
518   pics/parallel arrow/.style={
519     code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
520 {\end{tikzpicture}}
521 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
522   minimum height=0.5cm, minimum width=0.5cm,
523   single arrow head extend=2mm]
524 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
525   minimum width=1.4em, font={\small\color{white}\ttfamily},
526   fill=gray]

```

```

527 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
528   draw,font={\small}]
529 \tikzstyle{phi-object} = [phi-thing,circle]
530 \tikzstyle{phi-data} = [phi-thing,regular polygon,
531   regular polygon sides=8]
532 \tikzstyle{phi-empty} = [phi-object]
533 \tikzset{%
534   phi-rho/.style={
535     postaction={%
536       decoration={
537         show path construction,
538         curveto code={
539           \tikzmath{
540             coordinate \I, \F, \v;
541             \I = (\tikzinputsegmentfirst);
542             \F = (\tikzinputsegmentlast);
543             \v = ($(\I) -(\F)$);
544             real \d, \a, \r, \t;
545             \d = 0.8;
546             \t = atan2(\vy, \vx);
547             if \vx<0 then { \a = 90; } else { \a = -90; };
548             {
549               \draw[arrows={-latex}, decorate,
550                 decoration={%
551                   snake, amplitude=.4mm,
552                   segment length=2mm,
553                   post length=1mm
554                 }]
555               ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
556               -- ++(\t: 2*\d em);
557             };
558           }
559         },
560         lineto code={
561           \tikzmath{
562             coordinate \I, \F, \v;
563             \I = (\tikzinputsegmentfirst);
564             \F = (\tikzinputsegmentlast);
565             \v = ($(\I) -(\F)$);
566             real \d, \a, \r, \t;
567             \d = 0.8;
568             \t = atan2(\vy, \vx);
569             if \vx<0 then { \a = 90; } else { \a = -90; };
570             {
571               \draw[arrows={-latex}, decorate,
572                 decoration={%
573                   snake, amplitude=.4mm,
574                   segment length=2mm,
575                   post length=1mm}]
576               ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
577               -- ++(\t: 2*\d em);
578             };
579           }
580         }

```

```

581     },
582     decorate
583   }
584 }
585 }
586 \tikzstyle{phi-pi} = [draw,dotted]
587 \tikzstyle{phi-atom} = [phi-object,double]
588 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
589   rectangle,thin,minimum width=1.2em,anchor=north west,
590   font={\scriptsize}]
591 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
592   above=2pt,sloped/.append style={transform shape},
593   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:

```

594 \makeatletter
595 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
596 \makeatother

```

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

```

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

```

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

```

618 \RequirePackage{hyperref}
619 \pdfstringdefDisableCommands{
620   \def\({}%
621   \def\)}{%
622   \def\alpha{alpha}%
623   \def\varphi{phi}%
624 }
625 \makeatletter
626 \NewExpandableDocumentCommand{\eoAnon}{O{ANONYMIZED}m}{%
627   \ifdefined\eolang@anonymous%

```

```

628 \textcolor{orange}{#1}%
629 \else%
630   #2%
631 \fi%
632 }\makeatother

```

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

```

633 \newcommand\eolang{%
634   \eoAnon[XYZ]{\sffamily EO}}

```

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

```

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

```

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

```

637 \newcommand\xmirl{%
638   \eoAnon[XML](~\){XMIR}}

```

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

```

639 \newcommand\phiConst{%
640   \mathrel{\hspace{.15em}}%
641   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

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

```

642 \newcommand\phiWave{%
643   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

```

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

```

644 \newcommand\phiSlot[1]{%
645   \xrightarrow{\text{\sffamily\scshape #1}}}

```

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

```

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

```

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

```

656 \newcommand\phiMany[3]{%
657   \phi0set{#3}{\phiUset{#2}{#1}}}

```

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

```
658 \newcommand\phiEOL{\[-4pt]}
```

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

```
659 \RequirePackage{trimclip}
660 \RequirePackage{amsfonts}
661 \makeatletter
662 \newcommand{\phiDotted}{%
663   \mapstochar\mathrel{\mathpalette\phiDotted@relax}}
664 \newcommand{\phiDotted@}[2]{%
665   \begingroup%
666   \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
667   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
668   \sbox\z@{%
669     \makebox[\dimen\z@][s]{%
670       \clipbox{0 0 {0.4\width} 0}{%
671         {\resizebox{\dimen\z@}{\height}{%
672           {\m@th#1\dashrightarrow}}}%
673       \hss%
674       \clipbox{{0.69\width} {-0.1\height} 0
675         {-\height}}{\m@th#1\rightarrow}}}%
676     }%
677   }%
678   \ht\z@=\dimen\tw@ \dp\z@=\z@%
679   \box\z@%
680 \endgroup%
681 }
682 \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.	9	0.2.0	eolang-phi.pl: Numbers automatically render as \texttt. No need to use vertical bars around them anymore.	10
0.0.2	sodg: The environment phigure renamed to sodg for the sake of better semantic. The graph in the picture is solely a SODG graph, that's why the name sodg is better.	21		eolang-sodg.pl: The content of the atom and the data boxes is parsed automatically as formulas and numbers, respectively.	14
	eolang-phi.pl: New symbol added for basket slots	10		\xmirt: New command \xmirt prints XMIR in both normal and the anonymous mode of acmart.	22
	Parsing of the symbols “@,” “^,” and “&” enabled (\varphi, \rho, and \sigma)	10	0.3.0	\eolang@lineno: New counter for protecting lineno.	10
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets	10		eolang-phi.pl: New arrow added, that looks like \leadsto.	10
	eolang-sodg.pl: The Perl file now has a fixed name, which doesn't depend on the name of the TeX job. This file may be shared among jobs, no need to make it uniquely named.	14		\phiWave: New command \phiWave added to denote a link to a multi-layer attribute.	22
	\phiq: Parsing of additional symbols enabled.	13	0.4.0	eolang-sodg.pl: Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed with the \Delta and the \lambda commands.	14
0.1.0	General: Parsing of package options introduced.	9		Relative positioning of vertices fixed.	14
	\eolang: New command \eolang added to print the name of the language in both normal and the anonymous mode of acmart.	22	0.5.0	eolang-phi.pl: Automated formatting of TRUE and FALSE added.	10
	\eolang@mdfive: New supplementary command added to calculate MD5 sum of a file.	10		eolang-sodg.pl: It is possible to use TikZ commands inside the sodg environment.	14
	eolang-phi.pl: A new Perl script “eolang-phi.pl” added for parsing of phi expressions.	10		New syntax introduced that allows to make clones of vertices and all their dependants.	14
	eolang-sodg.pl: There are two Perl scripts now: one for phiuation, another one for sodg.	14		Now edges may have the break attribute, to make them shorter.	14
	\phic: New command \phic prints the name of φ -calculus in both normal and the anonymous mode of acmart.	22		\phiMany: New command \phiMany enables iterating over an arrow.	22
	\phiConst: New command \phiConst added to denote a link to a constant attribute.	22		\phiSlot: New command \phiSlot added to denote a link to a slot in a basket.	22
	\phiDotted: New command \phiDotted added to denote a link to a special attribute.	23	0.6.0	General: Package option nocomments added in order to enable comments suppression in temporary .tex files (may be pretty important for .dtx documents).	9

eolang-sodg.pl: The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations.	14		
0.7.0			
nodollar: Now it is possible to use dollar sign instead of the <code>\phiq</code> command.	14		
eolang-phi.pl: New syntax sugar for Φ , just using capital “Q” is enough.	10		
Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols.	10		
Text in quotes is automatically converted to <code>\texttt</code>	10		
0.8.0			
General: The <code>anonymous</code> package option added.	9		
eolang-phi.pl: Inside <code>phiquation</code> any text inside the <code>\text</code> macro is			
			not processed. 10
		eolang-sodg.pl: The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.	14
		<code>\phi0set</code> : New commands <code>\phi0set</code> and <code>\phiUset</code> help position text over and under an arrow.	22
		<code>\phiSaveTo</code> : The output of the <code>phiquation</code> environment can be redirected to a file.	13
		<code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file.	21
	0.9.0		
		<code>\eoAnon</code> : New command <code>\eoAnon</code> added.	21
		eolang-phi.pl: Proper handling of the <code>matrix</code> environment.	10
		<code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code>	23

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols		
$\backslash \$$	122, 220, 225, 229, 232, 607, 614	
$\backslash \%$	193, 224, 612	
$\backslash ($	620, 636, 638	
$\backslash)$	621, 636, 638	
$\backslash *$	88, 90, 119	
$\backslash +$	238, 365, 396, 402	
$\backslash -$	636	
$\backslash .$	89, 91, 127, 138, 238	
$\backslash /$	87, 88	
$\backslash ?$	130	
$\backslash [$	87, 124	
$\backslash \{$	58, 81, 93, 94, 119, 123, 127, 143	
$\backslash \}$	58, 93, 94, 119, 143	
$\backslash]$	87, 125	
$\backslash \sim$	123	
$\backslash $	88, 141, 142, 199, 205, 243, 598	
Numbers		
$\backslash 2$..	87, 89, 91, 99, 127, 258	
$\backslash 3$	89, 91	
$\backslash 4$	89	
A		
$\backslash a$	544, 547, 555, 566, 569, 576	
$\backslash active$.	225, 229, 232, 614	
$\backslash alpha$	622, 636	
$\backslash AtBeginDocument$..	232	
B		
$\backslash Bbbk$	3	
$\backslash begin$	25, 147, 150, 152, 201, 207, 235, 286, 516, 600	
$\backslash box$	679	
C		
$\backslash catcode$	199, 205, 220, 225, 229, 232, 598, 607, 614	
$\backslash clean$	213, 215	
$\backslash clipbox$	670, 674	
$\backslash color$	525	
D		
$\backslash d$..	143, 544, 545, 555, 556, 566, 567, 576, 577	
$\backslash dashrightarrow$...	672	
$\backslash def$	181, 184, 196, 216, 595, 603, 616, 620, 621, 622, 623	
$\backslash Delta$	447	
$\backslash detokenize$	213	
$\backslash dimen$	666, 667, 669, 671, 678	
$\backslash dp$	678	
$\backslash draw$...	298, 519, 549, 571	
E		
$\backslash E$	119, 377, 385, 393, 396, 402, 405, 406, 407, 408	
$\backslash end$	168, 170, 173, 176, 203, 209, 493, 500, 520, 602	
$\backslash endinginput$	175, 499	
$\backslash eoAnon$.	618, 634, 636, 638	
$\backslash eolang$	633	
$\backslash eolang-phi.pl$	24	
$\backslash eolang-sodg.pl$...	234	
$\backslash eolang@anonymous$	14, 627	
$\backslash eolang@lineno$	19	
$\backslash eolang@mdfive$	20, 184, 216, 603	
$\backslash eolang@nocomments$	13, 193, 224, 612	
$\backslash eolang@nodollar$..	220, 225, 227	
$\backslash eolang@phiSaveTo$.	181, 194, 196	
$\backslash eolang@process$...	183, 203, 209	
$\backslash eolang@sodgSaveTo$	595, 613, 616	
$\backslash eolang@tmpdir$	11, 18, 25, 178, 185, 186, 187, 189, 190, 192, 202, 208, 214, 217, 218, 219, 221, 222, 223, 235, 502, 601, 604, 605, 606, 609, 610, 611	
$\backslash ex@$	649, 653	
F		
$\backslash F$	540, 542, 543, 555, 562, 564, 565, 576	
$\backslash FancyVerbLine$	504	
G		
$\backslash gdef$	230	
H		
$\backslash hash$...	184, 187, 189, 192, 216, 219, 221, 223, 603, 606, 609, 611	
$\backslash hbox$	650, 654	
$\backslash height$	671, 674, 675	
$\backslash hspace$	640, 641	
$\backslash hss$	673	
$\backslash ht$	678	
I		
$\backslash I$	540, 541, 543, 555, 562, 563, 565, 576	
$\backslash iexec$...	18, 186, 189, 214, 218, 221, 605, 609	
$\backslash ifdefined$	193, 194, 220, 224, 225, 227, 612, 613, 627	
$\backslash ifluatex$	12	
$\backslash ifxetex$	12	
$\backslash inputlineno$...	188, 608	
J		
$\backslash jobname$.	18, 185, 186, 187, 189, 192, 202, 208, 214, 217, 218, 219, 221, 223, 601, 604, 605, 606, 609, 611	
K		
$\backslash kern$	649, 653	
L		
$\backslash lambda$	459	
$\backslash leadsto$	643	
$\backslash limits$	648, 652	
M		
$\backslash m@th$...	666, 667, 672, 675	
$\backslash makeatletter$	19, 21, 24,	

