



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

Yegor Bugayenko
yegor256@gmail.com

2022-12-16, 0.9.1

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

1 Introduction

This package helps you print formulas of φ -calculus, which is a formal foundation of [EO](#) programming language. The calculus was introduced by Bugayenko (2021) and later formalized by Kudasov et al. (2022). Here is how you render a simple expression:

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

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

*The sources are in GitHub at [objectionary/eolang.sty](https://github.com/objectionary/eolang.sty)

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

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

- “<<” maps to “ \langle ” (`\langle`),
- “>>” maps to “ \rangle ” (`\rangle`),
- “-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 `\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.

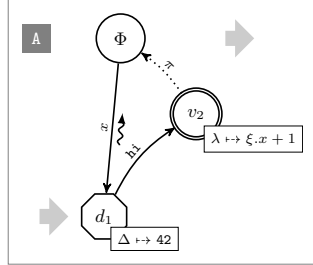
`\phiq` The command `\phiq` lets you inline a φ -calculus expressions using the same simple plain-text notation. You can use dollar sign directly too:

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

```

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

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



```

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

```

The content of the environment is parsed line by line. Markers in each line are separated by a single space. The first marker is either a unique name of a vertex, like “v1” in the example above, or an edge, like “v0->v1.” All other markers are either unary like “rho” or binary like “atom:\$\xi.x+1\$.” Binary markers have two parts, separated by colon.

The following markers are supported for a vertex:

- “tag: <math><math>” puts a custom label <math> into the circle,
- “data: [<box>]” makes it a data vertex with an optional attached “<box>” (the content of the box may only be numeric data),
- “atom: [<box>]” makes it an atom with an optional attached “<box>” (the content of the box is a math formula),
- “box: <txt>” attaches a “<box>” to it,
- “xy: <v>, <r>, <d>” places this vertex in a position relative to the vertex “<v>,” shifting it right by “<r>” and down by “<d>” centimetres.
- “+ : <v>” makes a copy of an existing vertex and all its kids.

The following markers are supported for an edge:

- “rho” places a backward snake arrow to the edge,
- “bend: <angle>” bend it right by the amount of “<angle>,”
- “a: <txt>” attaches label “<txt>” to it,
- “pi” makes it dotted, with 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{phiqation*}
7 [[ x -> [[D>42]] ]]
8 \end{phiqation*}
9 \begin{tabular}{p{.5in}l}
10 Free: & $\llbracket x -> ? \rrbracket$ \\
11 Bound: & \parbox{1in}{\input{a}} \\
12 \end{tabular}

```

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

2 Package Options

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

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

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

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

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

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

3 More Examples

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

| | |
|---|---|
| $\frac{x \mapsto [\varphi \mapsto y] \quad y \mapsto [z \mapsto 42]}{x.z \mapsto 42} \text{R1}$ | <pre> 6 \begin{phiquation*} 7 \dfrac \ 8 {x->[[@->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}`:

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

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

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

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

```

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

```

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

Discriminant can be calculated using the following simple formula:

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

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

```

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

```

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

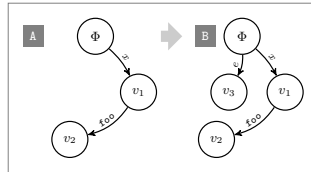
$$\begin{aligned} & \llbracket b \mapsto \emptyset \rrbracket, \\ & \llbracket \varphi \mapsto \text{TRUE}, \Delta \mapsto 42 \rrbracket, \\ & \psi = \langle \pi, 42 \rangle. \end{aligned}$$

```

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

```

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

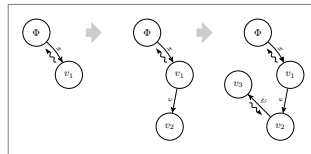


```

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

```

You can make a copy from a copy:

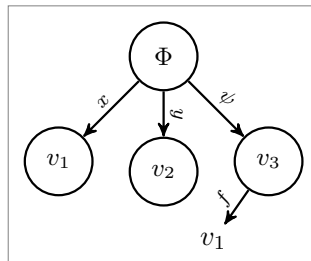


```

5 \begin{sodg}
6 v0
7 v1 xy:v0,.7,1
8 v0->v1 a:x bend:-10 rho
9 v0+a xy:v0,3,0 \\ v0=>v0a
10 v2a xy:v1a,-.8,1.3
11 v1a->v2a a:e
12 v0a+b xy:v0a,3,0 \\ v0a=>v0b
13 v3b xy:v2b,-1,-1
14 v2b->v3b a:\psi{} rho
15 \end{sodg}

```

You can have “broken” edges, using “break” attribute of an edge. The attribute must have a value, which is the percentage of the path between vertices that the arrow should take (can’t be more than 80 and less than 20). This may be convenient when you can’t fit all edges into the graph, for example:

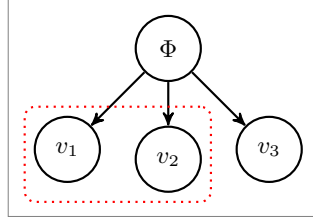


```

5 \begin{sodg}
6 v0
7 v1 xy:v0,-1,1
8 v0->v1 a:x
9 v2 xy:v0,0,1
10 v0->v2 a:y
11 v3 xy:v0,1,1
12 v0->v3 a:\psi{}
13 v3->v1 a:f bend:-75 break:30
14 \end{sodg}

```


You can add [TikZ](#) commands to `sodg` graph, for example:



```

6 \begin{sodg}
7 v0
8 v1 xy:v0,-1,1 \\\ v0->v1
9 v2 xy:v0,0,1 \\\ v0->v2
10 v3 xy:v0,1,1 \\\ v0->v3
11 \node[draw=red,rounded corners,\
12   dotted,fit=(v1) (v2)] {};
13 \end{sodg}

```

4 Implementation

First, we include a few packages. We need [stmaryrd](#) for `\llbracket` and `\rrbracket` commands:

```
1 \RequirePackage{stmaryrd}
```

We need [amsmath](#) for `equation*` environment:

```
2 \RequirePackage{amsmath}
```

We need [amssymb](#) for `\varnothing` command. We disable `\Bbbk` because it may conflict with some packages from [acmart](#):

```
3 \let\Bbbk\relax\RequirePackage{amssymb}
```

We need [fancyvrb](#) for `\VerbatimEnvironment` command:

```
4 \RequirePackage{fancyvrb}
```

We need [iexec](#) for executing Perl scripts:

```
5 \RequirePackage{iexec}
```

Then, we process package options:

```

6 \RequirePackage{pgfopts}
7 \RequirePackage{ifluatex}
8 \RequirePackage{ifxetex}
9 \pgfkeys{
10   /eolang/.cd,
11   tmpdir/.store in=\eolang@tmpdir,
12   tmpdir/.default=_eolang\ifxetex-xe\else\ifluatex-lua\fi\fi,
13   nocomments/.store in=\eolang@nocomments,
14   anonymous/.store in=\eolang@anonymous,
15   tmpdir
16 }
17 \ProcessPgfpPackageOptions{/eolang}

```

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

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

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

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

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

```

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

```

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

```

24 \makeatletter
25 \begin{VerbatimOut}{\eolang@tmpdir/eolang-phi.pl}
26 $macro = $ARGV[0];
27 open(my $fh, '<', $ARGV[1]);
28 my $tex; { local $/; $tex = <$fh>; }
29 print '% This file is auto-generated', "\n";
30 print '% There are ', length($tex),
31   ' chars in the input: ', $ARGV[1], "\n";
32 print '% ---', "\n";
33 if (index($tex, "\t") > 0) {
34   print "TABS are prohibited!";
35   exit 1;
36 }
37 my @lines = split (/\\n/g, $tex);
38 foreach my $t (@lines) {
39   print '% ', $t, "\n";
40 }
41 print '% ---', "\n";
42 $tex =~ s/\\.*/\\n\\n/g;
43 $tex =~ s/~/\\s+|\\s+$/g;
44 my $gathered = (0 == $tex =~ /\\n\\s+/g);
45 if ($gathered) {
46   print '% The "gathered" is used since all lines are left-aligned' . "\n";
47 }
48 my $align = 0;
49 print '% The "align" is NOT used by default' . "\n";
50 if (index($tex, '&&') >= 0) {
51   $macro =~ s/equation/align/g;
52   $align = 1;
53   print '% The "align" is used because of && seen in the text' . "\n";
54 }
55 if ($macro ne 'phiq') {
56   $tex =~ s/\\\\\\n\\n\\n/g;
57   $tex =~ s/\\\\\\n\\s*/g;
58   $tex =~ s/\\n*(\\label\\{[~\\}]+\\})\\n*/\\1/g;
59   $tex =~ s/\\n{3,}/\\n\\n/g;
60 }
61 my @texts = ();
62 sub trep {
63   my ($s) = @_ ;
64   my $open = 0;
65   my $p = 0;
66   for (; $p < length($s); $p++) {
67     $c = substr($s, $p, 1);
68     if ($c eq '}') {
69       if ($open eq 0) {
70         last;
71       }
72       $open--;
73     }
74     if ($c eq '{') {
75       $open++;

```

```

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

```

```

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

```

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

```

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

```

181 \makeatother

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

```

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

```

\phiq Then, we define \phiq command:

```

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

```

nodollar Then, we redefine dollar sign:

```

226 \ifdefined\eolang@nodollar\else
227   \begingroup

```

```

228 \catcode'\$=\active
229 \protected\gdef$#1${\phiq{#1}}
230 \endgroup
231 \AtBeginDocument{\catcode'\$=\active}
232 \fi

```

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

```

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

```

```

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

```

```

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

```



```

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

```

```

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

```

```

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

```

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

```
503 \setcounter{FancyVerbLine}{0}
```

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

```

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

```

picture Then, we define internal environment phicture:

```

514 \newenvironment{phicture}%
515 {\noindent\begin{tikzpicture}[
516   ->,>=stealth',node distance=0,thick,
517   pics/parallel arrow/.style={
518     code={\draw[-latex,phi-rho] (##1) -- (-##1);}}]}%
519 {\end{tikzpicture}}
520 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
521   minimum height=0.5cm, minimum width=0.5cm,
522   single arrow head extend=2mm]
523 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
524   minimum width=1.4em, font={\small\color{white}\ttfamily},
525   fill=gray]
526 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
527   draw,font={\small}]
528 \tikzstyle{phi-object} = [phi-thing,circle]
529 \tikzstyle{phi-data} = [phi-thing,regular polygon,
530   regular polygon sides=8]
531 \tikzstyle{phi-empty} = [phi-object]
532 \tikzset{%
533   phi-rho/.style={
534     postaction={%
535       decoration={
536         show path construction,
537         curveto code={
538           \tikzmath{
539             coordinate \I, \F, \v;
540             \I = (\tikzinputsegmentfirst);
541             \F = (\tikzinputsegmentlast);

```

```

542      \v = ($(\I) -(\F)$);
543      real \d, \a, \r, \t;
544      \d = 0.8;
545      \t = atan2(\vy, \vx);
546      if \vx<0 then { \a = 90; } else { \a = -90; };
547      {
548        \draw[arrows={-latex}, decorate,
549          decoration={%
550            snake, amplitude=.4mm,
551            segment length=2mm,
552            post length=1mm
553          }]
554        ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
555        -- ++(\t: 2*\d em);
556      };
557    }
558  },
559  lineto code={
560    \tikzmath{
561      coordinate \I, \F, \v;
562      \I = (\tikzinputsegmentfirst);
563      \F = (\tikzinputsegmentlast);
564      \v = ($(\I) -(\F)$);
565      real \d, \a, \r, \t;
566      \d = 0.8;
567      \t = atan2(\vy, \vx);
568      if \vx<0 then { \a = 90; } else { \a = -90; };
569      {
570        \draw[arrows={-latex}, decorate,
571          decoration={%
572            snake, amplitude=.4mm,
573            segment length=2mm,
574            post length=1mm}]
575        ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
576        -- ++(\t: 2*\d em);
577      };
578    }
579  }
580 },
581 decorate
582 }
583 }
584 }
585 \tikzstyle{phi-pi} = [draw,dotted]
586 \tikzstyle{phi-atom} = [phi-object,double]
587 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
588   rectangle,thin,minimum width=1.2em,anchor=north west,
589   font={\scriptsize}]
590 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
591   above=2pt,sloped/.append style={transform shape},
592   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:

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

636 \newcommand\xmir{%
637   \eoAnon[XML\(^+\)]{XMIR}}

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

658 \RequirePackage{trimclip}
659 \RequirePackage{amsfonts}
660 \makeatletter
661 \newcommand{\phiDotted}{%
662   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
663 \newcommand{\phiDotted@}[2]{%
664   \begingroup%
665   \settowidth{\dimen\z@}{$\m@th#1\rightharpoonup$}%
666   \settoheight{\dimen\tw@}{$\m@th#1\rightharpoonup$}%
667   \sbox\z@{%
668     \makebox[\dimen\z@][s]{%

```

```

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

| | | | |
|--|-------|--|---------------------------|
| eolang-sodg.pl: The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations. | 14 | | |
| 0.7.0 | | | |
| nodollar: Now it is possible to use dollar sign instead of the <code>\phiq</code> command. | 13 | | |
| eolang-phi.pl: New syntax sugar for 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. | 12 |
| | | <code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file. | 20 |
| | 0.9.0 | | |
| | | <code>\eoAnon</code> : New command <code>\eoAnon</code> added. | 21 |
| | | eolang-phi.pl: Proper handling of the <code>matrix</code> environment. | 10 |
| | | <code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code> | 22 |

Index

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

| Symbols | D | F |
|--|--|--|
| \backslash \$ 122, 219, 224, 228, 231, 606, 613 | \backslash d .. 142, 543, 544, 554, 555, 565, 566, 575, 576 | \backslash F 539, 541, 542, 554, 561, 563, 564, 575 |
| \backslash % 192, 223, 611 | \backslash dashrightarrow ... 671 | \backslash FancyVerbLine <u>503</u> |
| \backslash (..... 619, 635, 637 | \backslash def 180, 183, 195, 215, 594, 602, 615, 619, 620, 621, 622 | G |
| \backslash) 620, 635, 637 | \backslash Delta 446 | \backslash gdef 229 |
| \backslash * 88, 90, 119 | \backslash detokenize 212 | H |
| \backslash + 237, 364, 395, 401 | \backslash dimen 665, 666, 668, 670, 677 | \backslash hash ... 183, 186, 188, 191, 215, 218, 220, 222, 602, 605, 608, 610 |
| \backslash - 635 | \backslash dp 677 | \backslash hbox 649, 653 |
| \backslash 89, 91, 127, 137, 237 | \backslash draw ... 297, 518, 548, 570 | \backslash height 670, 673, 674 |
| \backslash / 87, 88 | E | \backslash hspace 639, 640 |
| \backslash ? 130 | \backslash E 119, 376, 384, 392, 395, 401, 404, 405, 406, 407 | \backslash hss 672 |
| \backslash [..... 87, 124 | \backslash end 167, 169, 172, 175, 202, 208, 492, 499, 519, 601 | \backslash ht 677 |
| \backslash { 58, 81, 93, 94, 119, 123, 127, 142 | \backslash endinginput 174, 498 | I |
| \backslash } 58, 93, 94, 119, 142 | \backslash eoAnon . <u>617</u> , 633, 635, 637 | \backslash I 539, 540, 542, 554, 561, 562, 564, 575 |
| \backslash] 87, 125 | \backslash eolang <u>632</u> | \backslash iexec ... 18, 185, 188, 213, 217, 220, 604, 608 |
| \backslash ^ 123 | \backslash eolang-phi.pl <u>24</u> | \backslash ifdefined 192, 193, 219, 223, 224, 226, 611, 612, 626 |
| \backslash 88, 140, 141, 198, 204, 242, 597 | \backslash eolang@sodg.pl ... <u>233</u> | \backslash ifluatex 12 |
| Numbers | \backslash eolang@anonymous 14, 626 | \backslash ifxetex 12 |
| \backslash 2 .. 87, 89, 91, 99, 127, 257 | \backslash eolang@lineno <u>19</u> | \backslash inputlineno ... 187, 607 |
| \backslash 3 89, 91 | \backslash eolang@mdfive <u>20</u> , 183, 215, 602 | J |
| \backslash 4 89 | \backslash eolang@nocomments ... 13, 192, 223, 611 | \backslash jobname . 18, 184, 185, 186, 188, 191, 201, 207, 213, 216, 217, 218, 220, 222, 600, 603, 604, 605, 608, 610 |
| A | \backslash eolang@nodollar 219, 224, 226 | K |
| \backslash a 543, 546, 554, 565, 568, 575 | \backslash eolang@phiSaveTo 180, 193, 195 | \backslash kern 648, 652 |
| \backslash active . 224, 228, 231, 613 | \backslash eolang@process 182, 202, 208 | L |
| \backslash alpha 621, 635 | \backslash eolang@sodgSaveTo 594, 612, 615 | \backslash lambda 458 |
| \backslash AtBeginDocument .. 231 | \backslash eolang@tmpdir 11, 18, 25, 177, 184, 185, 186, 188, 189, 191, 201, 207, 213, 216, 217, 218, 220, 221, 222, 234, 501, 600, 603, 604, 605, 608, 609, 610 | \backslash leadsto 642 |
| B | \backslash ex@ 648, 652 | \backslash limits 647, 651 |
| \backslash Bbbk 3 | | M |
| \backslash begin 25, 146, 149, 151, 200, 206, 234, 285, 515, 599 | | \backslash m@th ... 665, 666, 671, 674 |
| \backslash box 678 | | \backslash makeatletter 19, 21, 24, |
| C | | |
| \backslash catcode 198, 204, 219, 224, 228, 231, 597, 606, 613 | | |
| \backslash clean 212, 214 | | |
| \backslash clipbox 669, 673 | | |
| \backslash color 524 | | |

| | | | | | | | | | | | | | | | |
|---|---|--|---|-------------------------------------|--|---|--|---------------------------------|---------------------------|--|--|--|--|--|--|
| 179, 182, 211, 233, 593, 596, 624, 645, 660 | \makeatother 19, 23, 178, 181, 209, 225, 502, 595, 616, 631, 654, 681 | \makebox 668 | \mapsto 640 | \mapstochar . 640, 642, 662 | \mathop 647, 651 | \mathpalette 662 | \mathrel 639, 640, 642, 647, 651, 662 | \message 176, 187, 500, 607 | \mspace 642 | | | | | | |
| N | | | | | | | | | | | | | | | |
| \newcommand | 22, 180, 182, 211, 594, 632, 634, 636, 638, 641, 643, 646, 650, 655, 657, 661, 663 | \newcounter 19 | \newenvironment 197, 203, 514, 596 | \NewExpandableDocumentCommand | 625 | \node 336, 347, 350, 353, 360, 423 | \nodollar 226 | \noindent 515 | | | | | | | |
| P | | | | | | | | | | | | | | | |
| \pdf@filemdfivesum . 22 | \pdfstringdefDisableCommands | 618 | \pgfkeys 9 | \Phi 248 | \phic 634 | \phiConst 638 | \phicture 514 | \phiDotted . 446, 458, 658 | \phiDotted@ 662, 663 | | | | | | |
| \phiEOL 657 | \phiMany 655 | | | | | | | | | | | | | | |
| \phiOset 645, 656 | \phiq 210, 229 | \phiquation 182 | \phiSaveTo 179 | \phiSlot 643 | \phiUset 650, 656 | \phiWave 641 | \pi 301 | \ProcessPgfPackageOptions | 17 | | | | | | |
| \protected 229 | Q | | | | | | | | | | | | | | |
| \Q 119, 376, 384, 392, 395, 401, 404, 405, 406, 407 | R | | | | | | | | | | | | | | |
| \relax 3, 195, 615, 662 | \RequirePackage . . 1, 2, 3, 4, 5, 6, 7, 8, 20, 210, 504, 617, 658, 659 | \resizebox 670 | \rightarrow . 665, 666, 674 | S | | | | | | | | | | | |
| \sbox 667 | \scriptscriptstyle | 649, 653 | \scriptsize 589, 592 | \scshape 644 | \setcounter 194, 199, 205, 503, 598, 614 | \settoheight 666 | \settowidth 665 | \sffamily 633, 644 | \small 524, 527 | | | | | | |
| \sodg 596 | \sodgSaveTo 593 | \StrSubstitute 212 | \sxy 406 | T | | | | | | | | | | | |
| \t 33, 263, 543, 545, 554, 555, 565, 567, 575, 576 | Z | | | | | | | | | | | | | | |
| \z@ 665, 667, 668, 670, 677, 678 | V | | | | | | | | | | | | | | |
| \v 539, 542, 561, 564 | \value 194, 199, 205, 598, 614 | \varphi 622, 635 | \vbox 648, 652 | W | | | | | | | | | | | |
| \VerbatimEnvironment | 198, 204, 597 | \vss 649, 653 | \vx 545, 546, 567, 568 | \vy 545, 567 | X | | | | | | | | | | |
| \width 669, 673 | X | | | | | | | | | | | | | | |
| \xmir 636 | \xrightarrow 644 | Z | | | | | | | | | | | | | |
| \text 446, 644 | \textcolor 627 | \textnormal 447 | \texttt 447 | \the 187, 607 | \tikz 504 | \tikzinputsegmentfirst | 540, 562 | \tikzinputsegmentlast | 541, 563 | | | | | | |
| \tikzmath 538, 560 | \tikzset 532 | \tikzstyle 520, 523, 526, 528, 529, 531, 585, 586, 587, 590 | \ttfamily 524 | \tw@ 666, 677 | U | | | | | | | | | | |
| \usetikzlibrary ... 505, 506, 507, 508, 509, 510, 511, 512, 513 | U | | | | | | | | | | | | | | |