



eolang: L^AT_EX Package for Formulas and Graphs of EO Programming Language and φ -calculus*

Yegor Bugayenko
yegor256@gmail.com

2023-02-07, 0.11.1

NB! You must run L^AT_EX 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 ↪ [] ρ ↪ ξ.b. , α₀ t ↪ TRUE, b ↪ [α* ↪ fn(56), φ ↪ Φ.hello.bye(ξ), Δ ↪ 01-FE-C3], x ↪ [λ ↪ ∅]. </pre>	<pre> 1 \documentclass{minimal} 2 \usepackage{eolang} 3 \begin{document} 4 \begin{phiquation*} 5 app -> [[% it's abstract! 6 ^ !-> \$.b.^{^2}, 0/t^> TRUE, 7 b -> [[*-> fn(56), 8 @ -> Q.hello.bye(\$), 9 D> 01-FE-C3]]], \\ 10 x -> [[\lambda ..> ?]]. 11 \end{phiquation*} 12 \end{document} </pre>
---	---

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

*The sources are in GitHub at [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 “ \rightarrowtail ” (`\phiConst`),
- “..>” maps to “ \rightarrowtail ” (`\phiDotted`),
- “D>” maps to “ $\Delta \rightarrowtail$ ” (`\Delta ..>`),
- “L>” maps to “ $\lambda \rightarrowtail$ ” (`\lambda ..>`),
- “[[” maps to “[[” (`\llbracket`),
- ”]]” maps to ”]]” (`\rrbracket`),
- “==” maps to “ \equiv ” (`\equiv`),
- “|abc|” maps to “ abc ” (`\texttt{abc}`).

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

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

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

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

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

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

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

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

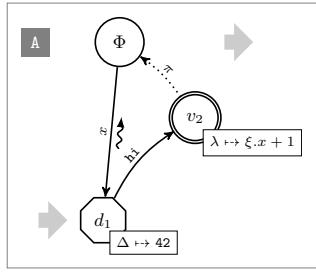
`\phiiq` The command `\phiiq` lets you inline a φ -calculus expressions using the same simple plain-text notation. You can use dollar sign directly too:

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 | \phi{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=>”, or 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 \phic language. It understands the anonymous package option and prints itself differently, to \xmir double-blind your paper. There is also \phic command to print the name of φ -calculus, also sensitive to anonymous mode. The macro \xmir prints "XMIR".

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

```

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 \xmir{} --- \\
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 \xmir{} --- \\
9 | its XML-based presentation.
10| \end{document}
```

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

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

```

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

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

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

$\llcorner \rightarrow \lrcorner$.

```

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{phiquation*}
10| [[ \phiiset{*}{->} ]].
11| \end{phiquation*}
```

\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 $[\alpha_1 \rightarrow x_1, \alpha_2 \rightarrow x_2, \dots, \alpha_n \rightarrow x_n]$ and expression $[\alpha_i \stackrel{n}{\rightarrow} x_i]$ are syntactically different but semantically equivalent.

```

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

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

```

5 | \phiSaveTo{a}
6 | \begin{phiquation*}
7 | [[ x -> [[D>42]] ]]
8 | \end{phiquation*}
9 | \begin{tabular}{p{.5in}l}
10| Free: & $[[x -> ?]]$ \\
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`, `\XMR`, and `\phic` commands by using `anonymous` package option (they all use the `\eoAnon` command mentioned earlier):

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

3 More Examples

The `phiuation` 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{phiuation*} 7 \dfrac { 8 {x->[@>y]} \quad y->[z->42]]} \ 9 {x.z -> 42} \ 10 \text{\sffamily R1} 11 \end{phiuation*}</pre>
--	---

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

$\frac{x \mapsto [\varphi \mapsto y, z \mapsto 42, \alpha_0 g \mapsto \emptyset, \alpha_1 foo \mapsto 42]}{x \mapsto [\varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto [\psi \mapsto \text{hello}(12)], \alpha_1 \mapsto 42)]} R2.$	<pre> 6 \begin{phiuation*} 7 \dfrac{\begin{split} 8 {x->[@>y, z->42,} 9 {0/g->?, 1/foo->42]} 10 \end{split}}{\begin{split} 11 {x->[@>y, z->?, f ~> \text{pi} (} 12 {0->[\text{ }\psi !-> \text{hello} (12)],} 13 {1->42)]}} 14 \end{split}}\text{R2}. 15 \end{phiuation*}</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{phiuation*} 6 x -> \left\{ \begin{array}{c} \emptyset \\ [\lambda \mapsto \rho \times \xi.\alpha_0] \\ [\Delta \mapsto 42] \end{array} \right\} \\ 7 ? \\ 8 [[L> ^ \times \\$.\alpha_0]] \\ 9 [[D> 42]] \\ 10 \end{phiuation*}</pre>
--	--

The `cases` environment works too:

$$\beta \models \left\{ \begin{array}{l} [v_2, \varphi \xrightarrow{\text{DTZD}} 42] \\ [v_{33}] \end{array} \right.$$

```

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

```

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

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

```

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

```

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

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{equation}
9 D = b\{^2} - \{4}ac.
10 \label{d}
11 \end{equation}
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{phiquation*}
7 [[ 0->x ]] && \text{This is formation}
8 [[ 0->? ]] && \text{Abstraction}
9 x(D>42) && \text{Application}
10 \end{phiquation*}
```

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 $Q \times a^2$.

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

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

```
x(π) ↦ [[λ ↦ f_1]],
x(a, b, c) ↦ [[α_0 ↦ ∅, φ ↦ hello(ξ), x ↦ FALSE]],
Δ = 43-09,
x(y) ≡ x(α_0 ↦ y).
```

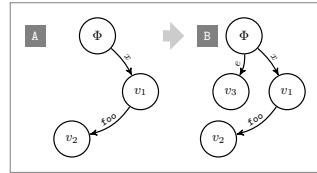
```
5 \begin{phiquation*}
6 x(\pi) \rightarrow [[\lambda \mapsto f_1]],
7 x(a,b,c) \rightarrow [[ \alpha_0 \rightarrow ?, \ \
8 @ \rightarrow |hello|($), x \rightarrow |FALSE| ]], \ \
9 \Delta = |43-09|,
10 x(y) == x(0 \rightarrow y).
11 \end{phiquation*}
```

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

```
[[b ↦ ∅]],
[[φ ↨ TRUE, Δ ↨ 42]],
ψ = ⟨π, 42⟩.
```

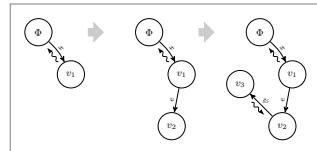
```
5 \begin{phiquation*}
6 [[ b \rightarrow ? ]],
7 [[ @ \rightarrow TRUE, \Delta \dots 42 ]], \ \
8 \psi = << \pi, 42 >>.
9 \end{phiquation*}
```

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



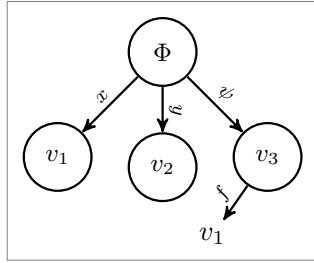
```
5 \begin{sodg}
6 v0 \\ v0!!A
7 v1 xy:v0,.7,1
8 v0->v1 a:x bend:-10
9 v2 xy:v1,-1.3,.8
10 v1->v2 a:|foo| bend:-20
11 v0+a xy:v0,3,0
12 v3a xy:v0a,-.7,1
13 v0a->v3a a:e bend:-15
14 v0=>v0a \\ v0a!B
15 \end{sodg}
```

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:

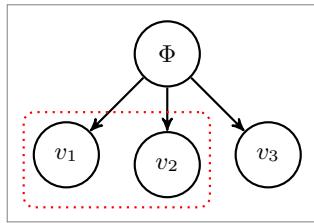


```

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 \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 phiuation 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 by 0.11.1\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 $indents = $tex =~ /\\n +/g;
45 my $gathered = (0 == $indents);
46 if ($gathered) {
47   print '% The "gathered" is used since all lines are left-aligned' . "\n";
48 } else {
49   print '% The "gathered" is NOT used because ' .
50     $indents . " lines are indented\n";
51 }
52 my $align = 0;
53 print '% The "align" is NOT used by default' . "\n";
54 if (index($tex, '\\&') >= 0) {
55   $macro =~ s/equation/align/g;
56   $align = 1;
57   print '% The "align" is used because of \\& seen in the text' . "\n";
58 }
59 if ($macro ne 'phiq') {
60   $tex =~ s/\\\\\\n\\n/g;

```

```

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

```

```

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

```

```

169 if (not($align)) {
170 print "\n";
171 if ($gathered) {
172     print '\end{gathered}';
173 } else {
174     print '\end{split}';
175 }
176 }
177 print "\n" . '\end{' . $macro . '}';
178 }
179 print '\endinput';
180 \end{VerbatimOut}
181 \message{eolang: File with Perl script
182   '\eolang@tmpdir/eolang-phi.pl' saved^{J}%
183 \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:

```

184 \makeatletter
185 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{\#1}}
186 \makeatother

```

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

```

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

```

\phiq Then, we define \phiq command:

```

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

```

`nodollar` Then, we redefine dollar sign:

```

231 \ifdefined\eolang@nodollar\else
232 \begingroup
233 \catcode`\$=\active
234 \protected\gdef$#1{\phiq{#1}}
235 \endgroup
236 \AtBeginDocument{\catcode`\$=\active}
237 \fi

```

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

```

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

```

```

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

```

```

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

```

```

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

```

```

426     (0+@cmds) . ' lines total)';
427 } elsif ($head =~ /~v[0-9]+[a-z]?$/) {
428     print '\node[';
429     if (exists $opts{'xy'}) {
430         my ($v, $right, $down) = split(/,/, $opts{'xy'});
431         my $loc = '';
432         if ($down > 0) {
433             $loc = 'below';
434         } elsif ($down < 0) {
435             $loc = 'above';
436         }
437         if ($right > 0) {
438             $loc = $loc . 'right';
439         } elsif ($right < 0) {
440             $loc = $loc . 'left';
441         }
442         print ',' . $loc . '=';
443         print abs(num($down)) . 'cm and ' .
444             abs(num($right)) . 'cm of ' . $v . '.center';
445     }
446     if (exists $opts{'data'}) {
447         print ',phi-data';
448         if ($opts{'data'} ne '') {
449             my $d = $opts{'data'};
450             if (index($d, '|') == -1) {
451                 $d = '$\Delta\phiDotted{text{' .
452                     '\textnormal{\texttt{' . fmt($d) . '}}}$';
453             } else {
454                 $d = fmt($d);
455             }
456             $opts{'box'} = $d;
457         }
458     } elsif (exists $opts{'atom'}) {
459         print ',phi-atom';
460         if ($opts{'atom'} ne '') {
461             my $a = $opts{'atom'};
462             if (index($a, '$') == -1) {
463                 $a = '$\lambda\phiDotted{}' . fmt($a) . '$';
464             } else {
465                 $a = fmt($a);
466             }
467             $opts{'box'} = $a;
468         }
469     } else {
470         print ',phi-object';
471     }
472     print ']';
473     print ' (' . $head . ')';
474     print '{';
475     if (exists $opts{'tag'}) {
476         my $t = $opts{'tag'};
477         if (index($t, '$') == -1) {
478             $t = '$' . $t . '$';
479         } else {

```

```

480     $t = fmt($t);
481 }
482     print $t;
483 } else {
484     print '$' . vertex($head) . '$';
485 }
486 print '}';
487 if (exists $opts{'box'}) {
488     print ' node[phi-box] at (';
489     print $head, '.south east) {';
490     print $opts{'box'}, '}';
491 }
492 } else {
493     print $cmd;
494 }
495 print ";\n";
496 }
497 print '\end{phicture}', "\n";
498 print "% --- after processing:\n%";
499 foreach my $c (@cmds) {
500     print '% ', $c, "\n";
501 }
502 print "% --- (' . (0+@cmds) . " lines)\n";
503 print '\endinput';
504 \end{VerbatimOut}
505 \message{eolang: File with Perl script
506   '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
507 \makeatother

```

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

```
508 \setcounter{FancyVerbLine}{0}
```

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

```

509 \RequirePackage{tikz}
510 \usetikzlibrary{arrows}
511 \usetikzlibrary{shapes}
512 \usetikzlibrary{decorations}
513 \usetikzlibrary{decorations.pathmorphing}
514 \usetikzlibrary{decorations.pathreplacing}
515 \usetikzlibrary{positioning}
516 \usetikzlibrary{calc}
517 \usetikzlibrary{math}
518 \usetikzlibrary{arrows.meta}

```

phicture Then, we define internal environment `phicture`:

```

519 \newenvironment{phicture}%
520   {\noindent\begin{tikzpicture}[
521     ->, >=stealth', node distance=0, thick,
522     pics/parallel arrow/.style={
523       code={\draw[-latex, phi-rho] (#1) -- (-##1);}}]}%
524   {\end{tikzpicture}}
525 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
526   minimum height=0.5cm, minimum width=0.5cm,

```

```

527   single arrow head extend=2mm]
528 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
529   minimum width=1.4em, font={\small\color{white}\ttfamily},
530   fill=gray]
531 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
532   draw,font={\small}]
533 \tikzstyle{phi-object} = [phi-thing,circle]
534 \tikzstyle{phi-data} = [phi-thing,regular polygon,
535   regular polygon sides=8]
536 \tikzstyle{phi-empty} = [phi-object]
537 \tikzset{%
538   phi-rho/.style={%
539     postaction={%
540       decoration={%
541         show path construction,
542         curveto code={%
543           \tikzmath{
544             coordinate \I, \F, \v;
545             \I = (\tikzinputsegmentfirst);
546             \F = (\tikzinputsegmentlast);
547             \v = ($(\I) -(\F)$);
548             real \d, \a, \r, \t;
549             \d = 0.8;
550             \t = atan2(\vy, \vx);
551             if \vx<0 then { \a = 90; } else { \a = -90; };
552             {
553               \draw[arrows=-latex], decorate,
554               decoration={%
555                 snake, amplitude=.4mm,
556                 segment length=2mm,
557                 post length=1mm
558               }];
559               ($(\F)! .5! (\I) +(\t: -\d em) +(\t +\a: 1ex)$)
560               -- ++(\t: 2*\d em);
561             };
562           }
563         },
564         lineto code={%
565           \tikzmath{
566             coordinate \I, \F, \v;
567             \I = (\tikzinputsegmentfirst);
568             \F = (\tikzinputsegmentlast);
569             \v = ($(\I) -(\F)$);
570             real \d, \a, \r, \t;
571             \d = 0.8;
572             \t = atan2(\vy, \vx);
573             if \vx<0 then { \a = 90; } else { \a = -90; };
574             {
575               \draw[arrows=-latex], decorate,
576               decoration={%
577                 snake, amplitude=.4mm,
578                 segment length=2mm,
579                 post length=1mm}]
580             ($(\F)! .5! (\I) +(\t: -\d em) +(\t +\a: 1ex)$)

```

```

581           -- ++(\t: 2*\d em);
582       };
583   }
584   },
585   },
586   decorate
587 }
588 }
589 }

590 \tikzstyle{phi-pi} = [draw,dotted]
591 \tikzstyle{phi-atom} = [phi-object,double]
592 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
593   rectangle,thin,minimum width=1.2em,anchor=north west,
594   font={\scriptsize}]
595 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
596   above=2pt,sloped/.append style={transform shape},
597   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:

```

598 \makeatletter
599 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{\#1}}
600 \makeatother

```

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

```

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

```

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

```

622 \RequirePackage{hyperref}
623 \pdfstringdefDisableCommands{
624 \def\({}%
625 \def\){}%
626 \def\alpha{\alpha}%
627 \def\varphi{\phi}%

```

```

628 }
629 \makeatletter
630 \NewExpandableDocumentCommand{\eoAnon}{O{ANONYMIZED}m}{%
631   \ifdefined\eolang@anonymous%
632   \textcolor{orange}{#1}%
633   \else%
634     #2%
635   \fi%
636 }\makeatother

```

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

```

637 \newcommand{\eolang}{%
638   \eoAnon[XYZ]{\sffamily EO}}

```

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

```

639 \newcommand{\phic}{%
640   \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.

```

641 \newcommand{\xmir}{%
642   \eoAnon[XML(+)]{XMIR}}

```

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

```

643 \newcommand{\phiConst}{%
644   \mathrel{\hspace{.15em}}\mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

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

```

646 \newcommand{\phiWave}{%
647   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

```

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

```

648 \newcommand{\phiSlot}[1]{%
649   \xrightarrow{\text{\sffamily\scshape #1}}}

```

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

```

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

```

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

```
660 \newcommand\phiMany[3]{%
661   \phi0set{\#3}{\phiUset{\#2}{\#1}}}
```

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

```
662 \newcommand\phiEOL{\\"[-4pt]}
```

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

```
663 \RequirePackage{trimclip}
664 \RequirePackage{amsfonts}
665 \makeatletter
666 \newcommand{\phiDotted}{%
667   \mapstochar{\mathrel{\mathpalette\phiDotted@\relax}}
668 \newcommand{\phiDotted@}[2]{%
669   \begingroup%
670   \settowidth{\dimen\z@}{$\m@th#1\rightarrow$}%
671   \settoheight{\dimen\tw@}{$\m@th#1\rightarrow$}%
672   \sbox\z@{%
673     \makebox[\dimen\z@][s]{%
674       \clipbox{0 0 {0.4\width} 0}%
675       {\resizebox{\dimen\z@}{\height}%
676        {$\m@th#1\dashrightarrow$}}%
677       \hss}%
678       \clipbox[{0.69\width} {-0.1\height} 0
679       {-\height}]{\m@th#1\rightarrow}%
680     }%
681   }%
682   \ht\z@=\dimen\tw@ \dp\z@=\z@%
683   \box\z@%
684   \endgroup%
685 }
686 \makeatother
```

References

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

Change History

0.0.1	General: First draft.	9	0.2.0	eolang-phi.pl: Numbers automatically render as <code>\texttt{}</code> . No need to use vertical bars around them anymore.	10
0.0.2	sodg: 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. <code>eolang-phi.pl:</code> New symbol added for basket slots	21	eolang-sodg.pl: The content of the atom and the data boxes is parsed automatically as formulas and numbers, respectively.	14	
	Parsing of the symbols “ <code>G</code> ”, “ <code>^</code> ”, and “ <code>&</code> ” enabled (<code>\varphi</code> , <code>\rho</code> , and <code>\sigma</code>)	10	\xmir: New command <code>\xmir</code> prints XMIR in both normal and the anonymous mode of <code>acmart</code>	22	
	The symbols “[” and ”]” replaced with “[[” and ”]]” for abstract object brackets, because they conflicted with normal square brackets	10	\eolang@lineno: New counter for protecting lineno.	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. <code>\phiq:</code> Parsing of additional symbols enabled.	14	eolang-phi.pl: New arrow added, that looks like <code>\leadsto</code>	10	
0.1.0	General: Parsing of package options introduced.	9	\phiWave: New command <code>\phiWave</code> added to denote a link to a multi-layer attribute.	22	
	\eolang: New command <code>\eolang</code> added to print the name of the language in both normal and the anonymous mode of <code>acmart</code>	22	0.3.0		
	\eolang@mdfive: New supplementary command added to calculate MD5 sum of a file.	10	eolang-sodg.pl: 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	
	eolang-phi.pl: A new Perl script “ <code>eolang-phi.pl</code> ” added for parsing of phi expressions.	10	Relative positioning of vertices fixed.	14	
	eolang-sodg.pl: There are two Perl scripts now: one for <code>phiuation</code> , another one for <code>sodg</code>	14	0.4.0		
	\phic: New command <code>\phic</code> prints the name of φ -calculus in both normal and the anonymous mode of <code>acmart</code>	22	eolang-phi.pl: Automated formatting of <code>TRUE</code> and <code>FALSE</code> added.	10	
	\phiConst: New command <code>\phiConst</code> added to denote a link to a constant attribute.	22	eolang-sodg.pl: It is possible to use TikZ commands inside the <code>sodg</code> environment.	14	
	\phiDotted: New command <code>\phiDotted</code> added to denote a link to a special attribute.	23	New syntax introduced that allows to make clones of vertices and all their dependants.	14	
			Now edges may have the <code>break</code> attribute, to make them shorter.	14	
			\phiMany: New command <code>\phiMany</code> enables iterating over an arrow.	23	
			\phiSlot: New command <code>\phiSlot</code> added to denote a link to a slot in a basket.	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.	14
eolang-phi.pl: New syntax sugar for Φ , just using capital “Q” is enough. 10 Object names are automatically converted to <code>\texttt{t}</code> , provided their names include two or more symbols.	10
Text in quotes is automatically converted to <code>\texttt{t}</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	
0.9.0	
not processed.	10
eolang-sodg.pl: The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.	14
\phi0set: New commands <code>\phi0set</code> and <code>\phi1set</code> help position text over and under an arrow.	22
\phiSaveTo: The output of the <code>\phiquation</code> environment can be redirected to a file.	13
\sodgSaveTo: The output of the <code>sodg</code> environment can be redirected to a file.	21
\eoAnon: New command <code>\eoAnon</code> added.	21
eolang-phi.pl: Proper handling of the <code>matrix</code> environment.	10
\phiEOL: 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	D	F
\\$ 126, 224, 229, 233, 236, 611, 618	\d 147, 548, 549, 559, 560, 570, 571, 580, 581	\F 544, 546, 547, 559, 566, 568, 569, 580
\% 197, 228, 616	\dashrightarrow 676	\FancyVerbLine 508
\(. 624, 640, 642	\def 185, 188, 200, 220, 599, 607, 620, 624, 625, 626, 627	
\) 625, 640, 642	\Delta 451	
* 92, 94, 123	\detokenize 217	
\+ 242, 369, 400, 406	\dimen 670, 671, 673, 675, 682	
\- 640	\dp 682	
\. 93, 95, 131, 142, 242	\draw 302, 523, 553, 575	
\/ 91, 92		
\? 134		
\[. 91, 128		
\{ 62, 85, 97, 98, 123, 127, 131, 147		
\} 62, 97, 98, 123, 147	\E 123, 381, 389, 397, 400, 406, 409, 410, 411, 412	
\] 91, 129	\end 172, 174, 177, 180, 207, 213, 497, 504, 524, 606	
\~ 127	\endinput 179, 503	
\ 92, 145, 146, 203, 209, 247, 602	\eoAnon 622, 638, 640, 642	
	\eolang 637	
	\eolang-phi.pl 24	
	\eolang-sodg.pl 238	
	\eolang@anonymous 14, 631	
	\eolang@lineno 19	
	\eolang@mdfive 20, 188, 220, 607	
	\eolang@nocomments 13, 197, 228, 616	
	\eolang@nodollar 224, 229, 231	
	\eolang@phiSaveTo 185, 198, 200	
	\eolang@process 187, 207, 213	
	\eolang@sodgSaveTo 599, 617, 620	
	\eolang@tmpdir 11, 18, 25, 182, 189, 190, 191, 193, 194, 196, 206, 212, 218, 221, 222, 223, 225, 227, 605, 608, 609, 610, 613, 615	
A		
\a 548, 551, 559, 570, 573, 580		
\active 229, 233, 236, 618		
\alpha 626, 640		
\AtBeginDocument 236		
B		
\Bbbk 3		
\begin 25, 151, 154, 156, 205, 211, 239, 290, 520, 604		
\box 683		
C		
\catcode		
203, 209, 224, 229, 233, 236, 602, 611, 618		
\clean 217, 219		
\clipbox 674, 678		
\color 529	\ex@ 653, 657	
D		
\d 147, 548, 549, 559, 560, 570, 571, 580, 581		
E		
\E 123, 381, 389, 397, 400, 406, 409, 410, 411, 412		
\end 172, 174, 177, 180, 207, 213, 497, 504, 524, 606		
\endinput 179, 503		
\eoAnon 622, 638, 640, 642		
\eolang 637		
\eolang-phi.pl 24		
\eolang-sodg.pl 238		
\eolang@anonymous 14, 631		
\eolang@lineno 19		
\eolang@mdfive 20, 188, 220, 607		
\eolang@nocomments 13, 197, 228, 616		
\eolang@nodollar 224, 229, 231		
\eolang@phiSaveTo 185, 198, 200		
\eolang@process 187, 207, 213		
\eolang@sodgSaveTo 599, 617, 620		
\eolang@tmpdir 11, 18, 25, 182, 189, 190, 191, 193, 194, 196, 206, 212, 218, 221, 222, 223, 225, 227, 605, 608, 609, 610, 613, 615		
F		
\F 544, 546, 547, 559, 566, 568, 569, 580		
\FancyVerbLine 508		
G		
\gdef 234		
H		
\hash 188, 191, 193, 196, 220, 223, 225, 227, 607, 610, 613, 615		
\hbox 654, 658		
\height 675, 678, 679		
\hspace 644, 645		
\hss 677		
\ht 682		
I		
\I 544, 545, 547, 559, 566, 567, 569, 580		
\iexec 18, 190, 193, 218, 222, 225, 609, 613		
\ifdefined		
197, 198, 224, 228, 229, 231, 616, 617, 631		
\ifluatex 12		
\ifxetex 12		
\inputlineno 192, 612		
J		
\jobname 18, 189, 190, 191, 193, 196, 206, 212, 218, 221, 222, 223, 225, 227, 605, 608, 609, 610, 613, 615		
K		
\kern 653, 657		
L		
\lambda 463		
\leadsto 647		
\limits 652, 656		
M		
\m@th 670, 671, 676, 679		
\makeatletter		
19, 21, 24,		

\phiIset 650, 661 \phiiq 215, 234 \phiquation 187 \phiISaveTo 184 \phiISlot 648 \phiIUsset 655, 661 \phiIWave 646 \pi 306 $\text{\ProcessPgfPackageOptions}$ 17 \protected 234 \Q 123, 381, 389, 397, 400, \Q 406, 409, 410, 411, 412 N \newcommand \relax 3, 200, 620, 667 \RequirePackage . . . 1, \relax 2, 3, 4, 5, 6, 7, 8, 20, \relax 215, 509, 622, 663, 664 \newcounter 19 \resizebox 675 \newenvironment \rightarrow 670, 671, 679 \rightarrow 202, 208, 519, 601 $\text{\NewExpandableDocumentCommand}$ 630 \sbox 672 \node 341, $\text{\scriptscriptstyle}$ 654, 658 \nodollar 231 \scriptsize 594, 597 \noindent 520 \scshape 649 \setcounter 199, P 204, 210, 508, 603, 619 $\text{\pdfatfilemdfivesum}$ 22 $\text{\pdfstringdefDisableCommand}$ \settoheight 670 \settowidth 623 \sffamily 638, 649 \pgfkeys 9 \small 529, 532 \Phi 253 \sodg 601 \phic 639 \sodgSaveTo 598 \phiConst 643 \StrSubstitute 217 \phicture 519 \sxy 411 \phiDotted 451, 463, 663 \phiDotted@ 667, 668 \phiEOL 662 \phiMany 660 S \v 544, 547, 566, 569 \value 199, 204, 210, 603, 619 \varphi 627, 640 \vbox 653, 657 $\text{\VerbatimEnvironment}$ 203, 209, 602 \vss 654, 658 \vx 550, 551, 572, 573 \vy 550, 572 T \t 33, 268, 548, 550, 559, \t 560, 570, 572, 580, 581 Z \z@ 670, 672, 673, 675, 682, 683	\textcolor 632 \textnormal 452 \texttt 452 \the 192, 612 \tikz 509 $\text{\tikzinsegmentfirst}$ 545, 567 $\text{\tikzinsegmentlast}$ 546, 568 \tikzmath 543, 565 \tikzset 537 \tikzstyle 525, \tikzstyle 528, 531, 533, 534, \tikzstyle 536, 590, 591, 592, 595 \ttfamily 529 \tw@ 671, 682 U \usetikzlibrary \usepackage 510, 511, 512, 513, \usepackage 514, 515, 516, 517, 518 V \v 544, 547, 566, 569 \value 199, 204, 210, 603, 619 \varphi 627, 640 \vbox 653, 657 $\text{\VerbatimEnvironment}$ 203, 209, 602 \vss 654, 658 \vx 550, 551, 572, 573 \vy 550, 572 W \width 674, 678 X \xmir 641 \xrightarrow 649 Z \z@ 670, 672, 673, 675, 682, 683
---	--