



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

Yegor Bugayenko
yegor256@gmail.com

2023-08-10, 0.15.0

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

1 Introduction

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

$\begin{aligned} \text{app} &\mapsto \llbracket \\ &\quad \rho \mapsto \xi.b.^2, \alpha_0 t \rightsquigarrow \text{TRUE}, \\ &\quad b \mapsto \llbracket \alpha_* \mapsto \Phi.\text{fn}(56), \\ &\quad \quad \varphi \mapsto \Phi.\text{string.trim}(\xi), \\ &\quad \quad \Delta \mapsto \text{01-FE-C3} \rrbracket \rrbracket, \\ x &\mapsto \llbracket \lambda \mapsto \emptyset \rrbracket. \end{aligned}$	<pre>1 \documentclass{minimal} 2 \usepackage{eolang} 3 \begin{document} 4 \begin{phiquestion*} 5 app -> [[% it's abstract! 6 ^ !-> \$.b.^{~2}, 0/t~> TRUE, 7 b -> [[*-> Q.fn(56), 8 @ -> QQ.string.trim(\$), 9 D> 01-FE-C3]]]],\ 10 x -> [[\lambda ..> ?]]. 11 \end{phiquestion*} 12 \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`),
- “QQ” maps to “ $\dot{\Phi}$ ” (`\dot{\Phi}`),
- “->” maps to “ \mapsto ” (`\mapsto`),
- “~>” maps to “ \rightsquigarrow ” (`\rightsquigarrow`),
- “!->” maps to “ $\# \mapsto$ ” (`\# \mapsto`),
- “. .>” maps to “ $\dot{\mapsto}$ ” (`\dot{\mapsto}`),
- “D>” maps to “ $\Delta \mapsto$ ” (`\Delta \mapsto`),
- “L>” maps to “ $\lambda \mapsto$ ” (`\lambda \mapsto`),
- “[[” maps to “ \llbracket ” (`\llbracket`),
- “]]” maps to “ \rrbracket ” (`\rrbracket`),
- “==” maps to “ \equiv ” (`\equiv`),
- “|abc|” maps to “ \texttt{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} ” (`\xrightarrow{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 asterisk 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 asterisk instead of a number, such that `\phiq{* /g->x}` renders as “ $\alpha_{*|g} \mapsto x$ ”

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

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

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

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

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

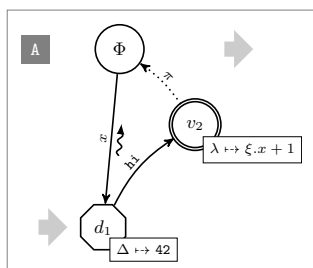
A simple object $x \mapsto [\varphi \mapsto y]$ is a decorator of the data object $y \mapsto [\Delta \mapsto 42]$.

```

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

```

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



```

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

```

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

The following markers are supported for a vertex:

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

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;

- “`style:{...}`” adds this TikZ style to the edge `\path`.

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

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

Without the anonymous option there will be no orange color:

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

`\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  $\$x \backslash\phi\text{Const } y\$$ 
8 makes  $\$y\$$  a constant,
9  $\$x \backslash\phi\text{Wave } y\$$  makes it a decoratee
10 of an arbitrary number of objects,
11 while  $\$x \backslash\phi\text{Dotted } y\$$  makes it
12 a special attribute.

```

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

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

$$[[\overset{*}{\mapsto}]].$$

```

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{phiqutation*}
10 [[ \phi0set{*}{->} ]].
11 \end{phiqutation*}

```

`\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 $[\alpha_1 \mapsto x_1, \alpha_2 \mapsto x_2, \dots, \alpha_n \mapsto x_n]$ and expression $[\alpha_i \overset{*}{\mapsto} x_i]$ 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 `phiqutation` or `sodg` environments inside `tabular` or any other
`\sodgSaveTo` environment or command, you won't be able to do this, because `phiqutation` and `sodg` are "verbatim" environments. `\phiSaveTo` and `\sodgSaveTo` commands will help you in this situation. You use them right before `\begin{phiqutation}` or `\begin{sodg}` respectively — the content of the equation or the graph won't be rendered, but instead saved to the file. Later, inside `tabular`, you can use it through the `\input` macro (don't forget the `\parbox`):

Free: $[[x \mapsto \emptyset]]$
Bound: $[[x \mapsto [[\Delta \mapsto 42]]]]$

```

5 \phiSaveTo{a}
6 \begin{phiqutation*}
7 [[ x -> [[D>42]] ]]
8 \end{phiqutation*}
9 \begin{tabular}{p{.5in}l}
10 Free: &  $[[x \rightarrow ?]]$  \\
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 `phiqutation` environment treats ends of line as signals to start new lines in the formula. If you don't want this to happen and want to parse the next line as the a continuation of the current line, you can use a single backslash as it's done here:

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

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

$\frac{x \mapsto [\varphi \mapsto y, z \mapsto 42, \alpha_0 g \mapsto \emptyset, \alpha_1 \text{foo} \mapsto 42]}{x \mapsto [\varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto [\psi \mapsto \text{hello}(12)], \alpha_1 \mapsto 42)]} R2.$	<pre>6 \begin{phiqutation*} 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 \rightsquigarrow pi (12 0->[[\psi !-> hello (12)]], 13 1->42]]] 14 \end{split}}\text{R2}. 15 \end{phiqutation*}</pre>
---	---

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

$\text{foo} \mapsto \left\{ \begin{array}{c} \emptyset \\ \llbracket \lambda \mapsto \rho \times \xi . \alpha_0 \rrbracket \\ \llbracket \Delta \mapsto 42 \rrbracket \end{array} \right\}$	<pre> 5 \begin{phiqutation*} 6 foo -> \left{\begin{matrix} \backslash 7 ? \backslash 8 \llbracket L > \sim \times \$. \alpha_0 \rrbracket \backslash 9 \llbracket D > 42 \rrbracket \backslash 10 \end{matrix}\right\} 11 \end{phiqutation*} </pre>
---	--

The cases environment works too:

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

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

$\begin{array}{l} x \mapsto \llbracket \\ y \mapsto \llbracket \\ z \mapsto \xi, f \mapsto \emptyset \rrbracket \rrbracket, \\ \beta_1 \models [\psi \xrightarrow{\text{WAIT}} \emptyset]. \end{array}$	<pre> 1 \documentclass{acmart} 2 \usepackage{eolang} 3 \thispagestyle{empty} 4 \begin{document} 5 \begin{phiqutation*} 6 x -> \llbracket 7 y -> \llbracket 8 z !-> \$, f ..> ? \rrbracket \rrbracket, \backslash 9 \beta_1 := [\psi -wait > ?]. 10 \end{phiqutation*} 11 \end{document} </pre>
---	---

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):

<p>Discriminant can be calculated using the following simple formula:</p> $D = b^2 - 4ac. \quad (1)$ <p>Eq. 1 is also widely used in number theory and polynomial factoring.</p>	<pre> 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. </pre>
--	---

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	6	<code>\begin{phiquation*}</code>
$\llbracket \alpha_0 \mapsto \emptyset \rrbracket$	Abstraction	7	<code>[[0->x]] && \text{This is formation}</code>
$x(\Delta \mapsto 42)$	Application	8	<code>[[0->?]] && \text{Abstraction}</code>
		9	<code>x(D>42) && \text{Application}</code>
		10	<code>\end{phiquation*}</code>

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

The object formation $\llbracket \alpha_0 \mapsto x \rrbracket$ may be replaced with a formula $Q \times a^2$.	6	The object formation $\$[\llbracket 0 \mapsto x \rrbracket]\$$
	7	may be replaced with a formula
	8	<code>\(Q \times a^2 \)</code> .

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

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

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

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

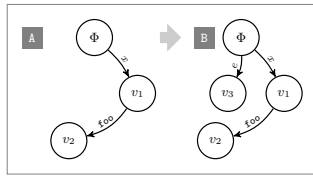
It is possible to use “manual splitting” mode in the `phiquation` environment by starting the body with `\begin{split}`:

$x(\pi) \mapsto 4$	5	<code>\begin{phiquation*}</code>
$x(a, b, c) \mapsto \llbracket \alpha_0 \mapsto \emptyset \rrbracket$	6	<code>\begin{split}</code>
	7	<code>x(\pi) & -> 4 \\\</code>
	8	<code>x(a,b,c) & -> [\llbracket \alpha_0 \mapsto \emptyset \rrbracket] \\\</code>
	9	<code>\end{split}</code>
	10	<code>\end{phiquation*}</code>

When necessary to use a percentage sign, prepend it with a backward slash:

$x \mapsto \text{sprintf}(\text{"Hello, \%s!"}, \text{name})$	5	<code>\begin{phiquation*}</code>
	6	<code>x -> \text{sprintf}(\text{"Hello, \%s!"}, \text{name})</code>
	7	<code>\end{phiquation*}</code>
	8	<code>\end{document}</code>

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

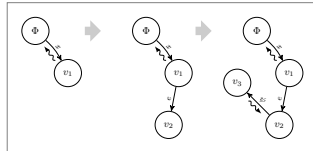


```

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:

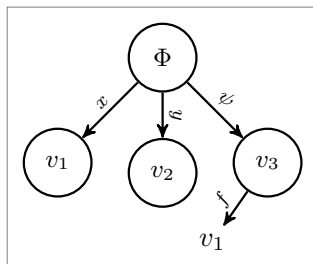


```

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:

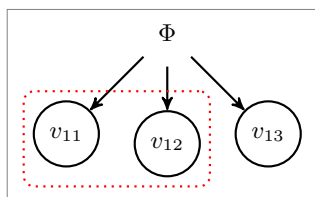


```

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:

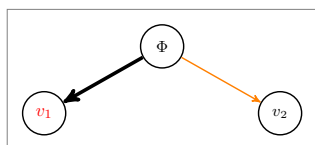


```

6 \begin{sodg}
7 v0 edgeless
8 v11 xy:v0,-1,1 \\\ v0->v11
9 v12 xy:v0,0,1 \\\ v0->v12
10 v13 xy:v0,1,1 \\\ v0->v13
11 \node[draw=red,rounded corners,\
12 dotted,fit=(v11) (v12)] {};
13 \end{sodg}

```

You can modify TikZ style yourself (make sure `style:` stays at the end of the line!), for example:



```

6 \begin{sodg}
7 v0
8 v1 xy:v0,-2,1 style:font=\color{red}
9 v2 xy:v0,2,1
10 v0->v1 style:line width=2pt
11 v0->v2 style:draw=orange
12 \end{sodg}

```

4 Implementation

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

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

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

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

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

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

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

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

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

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

Then, we process package options:

```

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

```

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

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

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

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

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

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

eolang-phi.pl Then, we create a Perl script for phiquation processing using VerbatimOut 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 0.15.0\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 $splitting = $tex =~ /^\\begin\\{split\\}/;
45 if ($splitting) {
46   print '% The manual splitting mode is ON since \\begin{split} started the text' . "\n";
47 }
48 my $indents = $tex =~ /\\n +/g;
49 my $gathered = (0 == $indents);
50 if ($gathered) {
51   if ($splitting) {
52     print '% The "gathered" is NOT used because of manual splitting' . "\n";
53     $gathered = 0;
54   } else {
55     print '% The "gathered" is used since all lines are left-aligned' . "\n";
56   }
57 } else {
58   print '% The "gathered" is NOT used because ' .
59     $indents . " lines are indented\n";
60 }
61 my $align = 0;
62 print '% The "align" is NOT used by default' . "\n";
```

```

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

```



```

171 } elsif (not $splitting) {
172   print '\begin{split}' . "\n";
173 }
174 }
175 }
176 if ($gathered and not($align)) {
177   $tex =~ s/^&//g;
178   $tex =~ s/\n&/\n/g;
179 }
180 print $tex;
181 if ($macro eq 'phiq') {
182   print '$' if ($tex ne '');
183 } else {
184   if (not($align)) {
185     if ($gathered) {
186       print "\n" . '\end{gathered}';
187     } elsif (not $splitting) {
188       print "\n" . '\end{split}';
189     }
190   }
191   print "\n" . '\end{' . $macro . '}';
192 }
193 print '\endinput';
194 \end{VerbatimOut}
195 \message{eolang: File with Perl script
196 '\eolang@tmpdir/eolang-phi.pl' saved^^J}%
197 \makeatother

```

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

```

198 \makeatletter
199 \newcommand\phiSaveTo[1]{\def\eolang@phiSaveTo{#1}}
200 \makeatother

```

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

```

201 \makeatletter\newcommand\eolang@process[1]{
202   \def\hash{\eolang@mdfive
203     {\eolang@tmpdir/\jobname/phiq. tex}}%
204   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiq. tex"
205     "\eolang@tmpdir/\jobname/\hash. tex"}%
206   \message{Start parsing 'phi' at line no. \the\inputlineno^^J}
207   \iexec[trace, stdout=\eolang@tmpdir/\jobname/\hash-post. tex]{
208     perl "\eolang@tmpdir/eolang-phi. pl"
209     '#1'
210     "\eolang@tmpdir/\jobname/\hash. tex"
211     \ifdefined\eolang@nocomments | perl -pe 's/\\%.*(\\n|$)//g'\fi
212     \ifdefined\eolang@phiSaveTo > \eolang@phiSaveTo\fi}%
213   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
214   \def\eolang@phiSaveTo{\relax}%
215 }
216 %
217 \newenvironment{phiq*}%
218 {\catcode'\|=12 \VerbatimEnvironment%

```

```

219 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
220 \begin{VerbatimOut}
221   {\eolang@tmpdir/\jobname/phiqutation.tex}
222 {\end{VerbatimOut}\eolang@process{equation*}}
223 %
224 \newenvironment{phiqutation}%
225 {\catcode'\|=12 \VerbatimEnvironment%
226 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
227 \begin{VerbatimOut}
228   {\eolang@tmpdir/\jobname/phiqutation.tex}}
229 {\end{VerbatimOut}\eolang@process{equation}}
230 \makeatother

```

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

```

231 \RequirePackage{xstring}
232 \makeatletter\newcommand\phiq[1]{%
233   \StrSubstitute{\detokenize{#1}}{'}'{''''}[\clean]%
234   \iexec[log,trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{
235     /bin/echo '\clean'}%
236   \def\hash{\eolang@mdfive
237     {\eolang@tmpdir/\jobname/phiq.tex}}%
238   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiq.tex"
239     "\eolang@tmpdir/\jobname/\hash.tex"}%
240   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
241   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
242     perl \eolang@tmpdir/eolang-phi.pl '\phiq'
243     "\eolang@tmpdir/\jobname/\hash.tex"
244     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$\)//g'\fi}%
245   \ifdefined\eolang@nodollar\else\catcode'\$=active\fi%
246 }\makeatother

```

`nodollar` Then, we redefine dollar sign:

```

247 \ifdefined\eolang@nodollar\else
248   \begingroup
249   \catcode'\$=\active
250   \protected\gdef\phiq#1{\phiq{#1}}
251   \endgroup
252   \AtBeginDocument{\catcode'\$=\active}
253 \fi

```

`eolang-sodg.pl` Then, we create a Perl script for `sodg` graphs processing using `VerbatimOut` from

[fancyvrb](#):

```

254 \makeatletter
255 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
256 sub num {
257   my ($i) = @_;
258   $i =~ s/(\+|-)\./\10./g;
259   return $i;
260 }
261 sub fmt {
262   my ($tex) = @_;
263   $tex =~ s/\\|([\^\\|]+)\\|\\textnormal{\\textttt{\\1}}/g;
264   return $tex;
265 }

```

```

266 sub vertex {
267   my ($v) = @_;
268   if (index($v, 'v0') == 0) {
269     return '\Phi';
270   } else {
271     $v =~ s/~/v/v_/g;
272     $v =~ s/[^0-9]$/g;
273     return $v . '>';
274   }
275 }
276 sub tailor {
277   my ($t, $m) = @_;
278   $t =~ s/<([A-Z]?${m}[A-Z]?):([>]+)/>/2/g;
279   $t =~ s/<[A-Z]+:[>+>/g;
280   return $t;
281 }
282 open(my $fh, '<', $ARGV[0]);
283 my $tex; { local $/; $tex = <$fh>; }
284 if (index($tex, "\t") > 0) {
285   print "TABS are prohibited!";
286   exit 1;
287 }
288 print '% This file is auto-generated', "\n\n";
289 print '% --- there are ', length($tex),
290   ' chars in the input (' , $ARGV[0], "):\n";
291 foreach my $t (split (/>/g, $tex)) {
292   print '% ', $t, "\n";
293 }
294 print "% ---\n";
295 $tex =~ s/\\\\/\\>/g;
296 $tex =~ s/\\n/\\n/g;
297 $tex =~ s/(\\[a-zA-Z]+)s+/\1/g;
298 $tex =~ s/\\n{2,}/>/g;
299 my @cmds = split(/>/g, $tex);
300 print '% --- before processing:' . "\n";
301 foreach my $t (split (/>/g, $tex)) {
302   print '% ', $t, "\n";
303 }
304 print '% ---';
305 print ' (' . (0+@cmds) . " lines)\n";
306 print '\begin{picture}', "\n";
307 for (my $c = 0; $c < 0+@cmds; $c++) {
308   my $cmd = $cmds[$c];
309   $cmd =~ s/~/s+//g;
310   $cmd =~ s/(?<!\)\%.*//g;
311   my ($head, $tail) = split(/ /, $cmd, 2);
312   my %opts = {};
313   my ($body, $style) = split(/style:/, $tail, 2);
314   $opts{'style'} = $style;
315   $tail = $body;
316   foreach my $p (split(/ /, $tail)) {
317     my ($q, $t) = split(/:/, $p);
318     $opts{$q} = $t;
319   }

```



```

320 if (index($head, '\\') == 0) {
321   print $cmd;
322 } elseif (index($head, '->') >= 0) {
323   my $draw = '\draw[';
324   if (exists $opts{'pi'}) {
325     $draw = $draw . '<MB:phi-pi><F:draw=none>';
326     if (not exists $opts{'a'}) {
327       $opts{'a'} = '\pi';
328     }
329   }
330   if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
331     $draw = $draw . '<MB:.,phi-rho>';
332   }
333   $draw = $draw . ', ' . $opts{'style'} . ']';
334   my ($from, $to) = split (/->/, $head);
335   $draw = $draw . " ($from) ";
336   if (exists $opts{'bend'}) {
337     $draw = $draw . 'edge [<F:draw=none><MF:.,bend right=' .
338       num($opts{'bend'}) . '>';
339     if (exists $opts{'rho'}) {
340       $draw = $draw . '<MB:.,phi-rho>';
341     }
342     $draw = $draw . ']';
343   } else {
344     $draw = $draw . '--';
345   }
346   if (exists $opts{'a'}) {
347     my $a = $opts{'a'};
348     if (index($a, '$') == -1) {
349       $a = '$' . fmt($a) . '$';
350     } else {
351       $a = fmt($a);
352     }
353     $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
354   }
355   if (exists $opts{'break'}) {
356     $draw = $draw . '<F: coordinate [pos=' .
357       ($opts{'break'} / 100) . '] (break)>';
358   }
359   $draw = $draw . " (<MF:${to}><B:break-v>";
360   if (exists $opts{'break'}) {
361     print tailor($draw, 'F') . ";\n";
362     print ' \node[outer sep=.1cm,inner sep=0cm] ' .
363       'at (break) (break-v) {'$' . vertex($to) .
364       '$};' . "\n";
365     print ' ' . tailor($draw, 'B');
366   } else {
367     print tailor($draw, 'M');
368   }
369 } elseif (index($head, '=>') >= 0) {
370   my ($from, $to) = split (/=>/, $head);
371   my $size = () = $head =~ /=/g;
372   if ($from eq '') {
373     print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .

```

```

374     $to . '.center]';
375 } elsif ($to eq '') {
376     print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
377         $from . '.center]';
378 } else {
379     print '\node [phi-arrow] at $(' .
380         $from . ')!0.5!(' . $to . ')$)';
381 }
382 print '{}';
383 } elsif (index($head, '!') >= 0) {
384     my ($v, $marker) = split (/!+/, $head);
385     my $size = () = $head =~ /!/g;
386     print '\node [phi-marker, left=' .
387         ($size * 0.6) . 'cm of ' .
388         $v . '.center]{' . fmt($marker) . '}';
389 } elsif (index($head, '+') >= 0) {
390     my ($v, $suffix) = split (/+/, $head);
391     my @friends = ($v);
392     foreach my $c (@cmds) {
393         $e = $c;
394         $e =~ s/^\s+//g;
395         my $h = $e;
396         $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
397         foreach my $f (@friends) {
398             my $add = '';
399             if (index($h, $f . '->') >= 0) {
400                 $add = substr($h, index($h, '->') + 2);
401             }
402             if ($h =~ /->\Q${f}\E/) {
403                 $add = substr($h, 0, index($h, '->'));
404             }
405             if (index($e, ' xy:' . $f . ',') >= 0) {
406                 $add = $h;
407             }
408             if (index($add, '+') == -1
409                 and $add ne ''
410                 and not(grep(/^Q${add}\E$/, @friends))) {
411                 push(@friends, $add);
412             }
413         }
414     }
415     my @extra = ();
416     foreach my $e (@cmds) {
417         $m = $e;
418         if ($m =~ /^s*\Q${v}\E\s/) {
419             next;
420         }
421         if ($m =~ /^s*[^\s]+\+/ and not($m =~ /^s*\Q${head}\E\s/)) {
422             next;
423         }
424         foreach my $f (@friends) {
425             my $h = $f;
426             $h =~ s/[a-z]$/g;
427             if ($m =~ s/^(s*)\Q${f}\E+\Q${suffix}\E\s?/\1${h}${suffix} /g) {

```

```

428         last;
429     }
430     $m =~ s/^(\\s*)\\Q${f}\\E\\s/\\1${h}${suffix} /g;
431     $m =~ s/^(\\s*)\\Q${f}\\E->/\\1${h}${suffix}->/g;
432     $m =~ s/\\sxy:\\Q${f}\\E,/ xy:${h}${suffix},/g;
433     $m =~ s/->\\Q${f}\\E\\s/->${h}${suffix} /g;
434 }
435 if ($m ne $e) {
436     push(@extra, ' ' . $m);
437 }
438 }
439 splice(@extra, 0, 0, @extra[-1]);
440 splice(@extra, -1, 1);
441 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
442 ') , friends: [' . join(', ', @friends) . ']' in ' .
443 (0+@cmds) . ' lines');
444 splice(@cmds, $c, 1, @extra);
445 print '% cloned ' . $v . ' at line no.' . $c .
446 ' (+ ' . (0+@extra) . ' lines -> ' .
447 (0+@cmds) . ' lines total)';
448 } elsif ($head =~ /^v[0-9]+[a-z]?$/) {
449     print '\\node[';
450     if (exists $opts{'xy'}) {
451         my ($v, $right, $down) = split(/,/, $opts{'xy'});
452         my $loc = '';
453         if ($down > 0) {
454             $loc = 'below ';
455         } elsif ($down < 0) {
456             $loc = 'above ';
457         }
458         if ($right > 0) {
459             $loc = $loc . 'right';
460         } elsif ($right < 0) {
461             $loc = $loc . 'left';
462         }
463         print ', ' . $loc . '=';
464         print abs(num($down)) . 'cm and ' .
465             abs(num($right)) . 'cm of ' . $v . '.center';
466     }
467     if (exists $opts{'data'}) {
468         print ',phi-data';
469         if ($opts{'data'} ne '') {
470             my $d = $opts{'data'};
471             if (index($d, '|') == -1) {
472                 $d = '$\\Delta\\phiDotted\\text{' .
473                     '\\textnormal{\\texttt{' . fmt($d) . '}}}$';
474             } else {
475                 $d = fmt($d);
476             }
477             $opts{'box'} = $d;
478         }
479     } elsif (exists $opts{'atom'}) {
480         print ',phi-atom';
481         if ($opts{'atom'} ne '') {

```

```

482     my $a = $opts{'atom'};
483     if (index($a, '$') == -1) {
484         $a = '$\lambda\phiDotted{}' . fmt($a) . '$';
485     } else {
486         $a = fmt($a);
487     }
488     $opts{'box'} = $a;
489 }
490 } else {
491     print ',phi-object';
492 }
493 if (exists $opts{'edgeless'}) {
494     print ',draw=none';
495 }
496 print ', ' . $opts{'style'} . '>';
497 print ' (' . $head . ')';
498 print '{';
499 if (exists $opts{'tag'}) {
500     my $t = $opts{'tag'};
501     if (index($t, '$') == -1) {
502         $t = '$' . $t . '$';
503     } else {
504         $t = fmt($t);
505     }
506     print $t;
507 } else {
508     print '$' . vertex($head) . '$';
509 }
510 print '>';
511 if (exists $opts{'box'}) {
512     print ' node[phi-box] at (';
513     print $head, '.south east) {';
514     print $opts{'box'}, '>';
515 }
516 }
517 print ";\n";
518 }
519 print '\end{picture}%', "\n";
520 print "% --- after processing:\n%";
521 foreach my $c (@cmds) {
522     print '% ', $c, "\n";
523 }
524 print '% --- (' . (0+@cmds) . " lines)\n";
525 print '\endinput';
526 \end{VerbatimOut}
527 \message{eolang: File with Perl script
528 '\eolang@tmpdir/eolang-sodg.pl' saved^~J}%
529 \makeatother

```

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

```
530 \setcounter{FancyVerbLine}{0}
```

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

```

531 \RequirePackage{tikz}
532 \usetikzlibrary{arrows}
533 \usetikzlibrary{shapes}
534 \usetikzlibrary{decorations}
535 \usetikzlibrary{decorations.pathmorphing}
536 \usetikzlibrary{decorations.pathreplacing}
537 \usetikzlibrary{positioning}
538 \usetikzlibrary{calc}
539 \usetikzlibrary{math}
540 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment phicture:

```

541 \newenvironment{phicture}%
542 {\noindent\begin{tikzpicture}[
543   ->,>=stealth',node distance=0,thick,
544   pics/parallel arrow/.style={
545     code={\draw[-latex,phi-rho] (##1) -- (-##1);}}]}%
546 {\end{tikzpicture}}
547 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
548   minimum height=0.5cm, minimum width=0.5cm,
549   single arrow head extend=2mm]
550 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
551   minimum width=1.4em, font={\small\color{white}\ttfamily},
552   fill=gray]
553 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
554   draw,font={\small}]
555 \tikzstyle{phi-object} = [phi-thing,circle]
556 \tikzstyle{phi-data} = [phi-thing,regular polygon,
557   regular polygon sides=8]
558 \tikzstyle{phi-empty} = [phi-object]
559 \tikzset{%
560   phi-rho/.style={
561     postaction={%
562       decoration={
563         show path construction,
564         curveto 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                 }
581               ]
582               ($\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$
583               -- ++(\t: 2*\d em);

```

```

583     };
584   }
585 },
586 lineto code={
587   \tikzmath{
588     coordinate \I, \F, \v;
589     \I = (\tikzinputsegmentfirst);
590     \F = (\tikzinputsegmentlast);
591     \v = ($(\I) -(\F)$);
592     real \d, \a, \r, \t;
593     \d = 0.8;
594     \t = atan2(\vy, \vx);
595     if \vx<0 then { \a = 90; } else { \a = -90; };
596     {
597       \draw[arrows={-latex}, decorate,
598         decoration={%
599           snake, amplitude=.4mm,
600           segment length=2mm,
601           post length=1mm}]
602         ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
603         -- ++(\t: 2*\d em);
604     };
605   }
606 }
607 },
608 decorate
609 }
610 }
611 }
612 \tikzstyle{phi-pi} = [draw,dotted]
613 \tikzstyle{phi-atom} = [phi-object,double]
614 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
615   rectangle,thin,minimum width=1.2em,anchor=north west,
616   font={\scriptsize}]
617 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
618   above=2pt,sloped/.append style={transform shape},
619   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:

```

620 \makeatletter
621 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
622 \makeatother

```

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

```

623 \makeatletter\newenvironment{sodg}%
624 {\catcode'\|=12 \VerbatimEnvironment%
625 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
626 \begin{VerbatimOut}
627   {\eolang@tmpdir/\jobname/sodg.tex}}
628 {\end{VerbatimOut}}%
629 \def\hash{\eolang@mdfive
630   {\eolang@tmpdir/\jobname/sodg.tex}}%
631 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"

```

```

632   "\eolang@tmpdir/\jobname/\hash.tex"%
633   \catcode'\$=3 %
634   \message{Start parsing 'sodg' at line no. \the\inputlineno^^J}
635   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
636     perl "\eolang@tmpdir/eolang-sodg.pl"
637     "\eolang@tmpdir/\jobname/\hash.tex"
638     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi
639     \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%
640   \catcode'\$active%
641   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
642   \def\eolang@sodgSaveTo{\relax}%
643 }\makeatother

```

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

```

644 \RequirePackage{hyperref}
645 \pdfstringdefDisableCommands{
646   \def\({}%
647   \def\)}%
648   \def\alpha{alpha}%
649   \def\varphi{phi}%
650 }
651 \makeatletter
652 \NewExpandableDocumentCommand{\eoAnon}{O{ANONYMIZED}m}{%
653   \ifdefined\eolang@anonymous%
654     \textcolor{orange}{#1}%
655   \else%
656     #2%
657   \fi%
658 }\makeatother

```

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

```

659 \newcommand\eolang{%
660   \eoAnon[XYZ]{\sffamily EO}}

```

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

```

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

```

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

```

663 \newcommand\xmirl{%
664   \eoAnon[XML\(^+\)]{XMIR}}

```

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

```

665 \newcommand\phiConst{%
666   \mathrel{\hspace{.15em}}%
667   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

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

```

668 \newcommand\phiWave{%
669   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

\phiSlot Then, we define a command to render an arrow for a slot in a basket:
670 \newcommand\phiSlot[1]{%
671   \xrightarrow{\text{\sffamily\scshape #1}}}

\phiOset Then, we define two commands to position a text over and under an arrow, as suggested
here:
672 \makeatletter
673 \newcommand{\phiOset}[2]{%
674   \mathrel{\mathop{#2}\limits^{
675     \vbox to 0ex{\kern-2\ex@
676       \hbox{\scriptscriptstyle#1}\vss}}}}
677 \newcommand{\phiUset}[2]{%
678   \mathrel{\mathop{#2}\limits_{
679     \vbox to 0ex{\kern-6.3\ex@
680       \hbox{\scriptscriptstyle#1}\vss}}}}
681 \makeatother

\phiMany Then, we define a command for an arrow with iterating indecies:
682 \newcommand\phiMany[3]{%
683   \phiOset{#3}{\phiUset{#2}{#1}}}

\phiEOL Then, we define a command for line breaks in formulas:
684 \newcommand\phiEOL{\[-4pt]}

\phiDotted Then, we define a command to render an arrow for a special attribute, as suggested here:
685 \RequirePackage{trimclip}
686 \RequirePackage{amsfonts}
687 \makeatletter
688 \newcommand{\phiDotted}{%
689   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
690 \newcommand{\phiDotted@}[2]{%
691   \begingroup%
692   \settowidth{\dimen\z@}{\m@th#1\rightarrow$}%
693   \settoheight{\dimen\tw@}{\m@th#1\rightarrow$}%
694   \sbox\z@{%
695     \makebox[\dimen\z@][s]{%
696       \clipbox{0 0 {0.4\width} 0}%
697       {\resizebox{\dimen\z@}{\height}%
698         {\m@th#1\dashrightarrow$}}%
699       \hss%
700       \clipbox{{0.69\width} {-0.1\height} 0
701         {-\height}}{\m@th#1\rightarrow$}%
702     }%
703   }%
704   \ht\z@=\dimen\tw@ \dp\z@=\z@%
705   \box\z@%
706   \endgroup%
707 }
708 \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.	10	0.12.1	<code>eolang-sodg.pl</code> : The bug is fixed related to the formatting of indexes of vertices.	15
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.	22	0.13.0	<code>eolang-phi.pl</code> : Parsing of <code>QQ</code> into <code>\dot{\Phi}</code> implemented.	11
	<code>eolang-phi.pl</code> : New symbol added for basket slots	11	0.14.0	<code>eolang-sodg.pl</code> : The <code>edgeless</code> tag of a vertex removes the border of it.	15
	Parsing of the symbols “@,” “^,” and “&” enabled (<code>\varphi</code> , <code>\rho</code> , and <code>\sigma</code>)	11	0.15.0	<code>eolang-sodg.pl</code> : The <code>style</code> tag of vertices and edges.	15
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets	11	0.2.0	<code>eolang-phi.pl</code> : Numbers automatically render as <code>\texttt</code> . No need to use vertical bars around them anymore.	11
	<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.	15		<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.	15
	<code>\phiq</code> : Parsing of additional symbols enabled.	15		<code>\xmirl</code> : New command <code>\xmirl</code> prints XMIR in both normal and the anonymous mode of <code>acmart</code>	23
0.1.0	General: Parsing of package options introduced.	10	0.3.0	<code>\eolang@lineno</code> : New counter for protecting <code>lineno</code>	11
	<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>	23		<code>eolang-phi.pl</code> : New arrow added, that looks like <code>\leadsto</code>	11
	<code>\eolang@mdfive</code> : New supplementary command added to calculate MD5 sum of a file.	11		<code>\phiWave</code> : New command <code>\phiWave</code> added to denote a link to a multi-layer attribute.	23
	<code>eolang-phi.pl</code> : A new Perl script “ <code>eolang-phi.pl</code> ” added for parsing of phi expressions.	11	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.	15
	<code>eolang-sodg.pl</code> : There are two Perl scripts now: one for <code>phiq</code> , another one for <code>sodg</code>	15		Relative positioning of vertices fixed.	15
	<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>	23	0.5.0	<code>eolang-phi.pl</code> : Automated formatting of <code>TRUE</code> and <code>FALSE</code> added.	11
	<code>\phiConst</code> : New command <code>\phiConst</code> added to denote a link to a constant attribute.	23		<code>eolang-sodg.pl</code> : It is possible to use TikZ commands inside the <code>sodg</code> environment.	15
	<code>\phiDotted</code> : New command <code>\phiDotted</code> added to denote a link to a special attribute.	24		New syntax introduced that allows to make clones of vertices and all their dependants.	15

Now edges may have the <code>break</code> attribute, to make them shorter.	15	Text in quotes is automatically converted to <code>\texttt</code>	11
<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow.	24	0.8.0	
<code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket.	24	General: The anonymous package option added.	10
0.6.0		<code>eolang-phi.pl</code> : Inside <code>phiqutation</code> any text inside the <code>\text</code> macro is not processed.	11
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).	10	<code>eolang-sodg.pl</code> : The <code>tag</code> attribute is introduced for changing labels inside a vertex circle.	15
<code>eolang-sodg.pl</code> : The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations.	15	<code>\phiOset</code> : New commands <code>\phiOset</code> and <code>\phiUset</code> help position text over and under an arrow.	24
0.7.0		<code>\phiSaveTo</code> : The output of the <code>phiqutation</code> environment can be redirected to a file.	14
<code>nodollar</code> : Now it is possible to use dollar sign instead of the <code>\phiq</code> command.	15	<code>\sodgSaveTo</code> : The output of the <code>sodg</code> environment can be redirected to a file.	22
<code>eolang-phi.pl</code> : New syntax sugar for Φ , just using capital “Q” is enough.	11	0.9.0	
Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols.	11	<code>\eoAnon</code> : New command <code>\eoAnon</code> added.	23
		<code>eolang-phi.pl</code> : Proper handling of the <code>matrix</code> environment.	11
		<code>\phiEOL</code> : New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code>	24

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 \$ 142, 240, 245, 249, 252, 633, 640	\backslash d .. 163, 570, 571, 581, 582, 592, 593, 602, 603	\backslash F 566, 568, 569, 581, 588, 590, 591, 602
\backslash % 211, 244, 638	\backslash dashrightarrow ... 698	\backslash FancyVerbLine ... <u>530</u>
\backslash (..... 646, 662, 664	\backslash def 199, 202, 214, 236, 621, 629, 642, 646, 647, 648, 649	G
\backslash) 647, 662, 664	\backslash Delta 472	\backslash gdef 250
\backslash * 106, 108, 139	\backslash detokenize 233	H
\backslash + 258, 390, 421, 427	\backslash dimen 692, 693, 695, 697, 704	\backslash hash ... 202, 205, 207, 210, 236, 239, 241, 243, 629, 632, 635, 637
\backslash - 662	\backslash dp 704	\backslash hbox 676, 680
\backslash . .. 107, 109, 147, 158, 258	\backslash draw ... 323, 545, 575, 597	\backslash height 697, 700, 701
\backslash / 105, 106	E	\backslash hspace 666, 667
\backslash ? 150	\backslash E 139, 402, 410, 418, 421, 427, 430, 431, 432, 433	\backslash hss 699
\backslash [..... 105, 144	\backslash end 186, 188, 191, 194, 222, 229, 519, 526, 546, 628	\backslash ht 704
\backslash { 44, 73, 96, 112, 113, 139, 143, 147, 163	\backslash endinginput 193, 525	I
\backslash } 44, 73, 112, 113, 139, 163	\backslash eoAnon . <u>644</u> , 660, 662, 664	\backslash I 566, 567, 569, 581, 588, 589, 591, 602
\backslash] 105, 145	\backslash eolang <u>659</u>	\backslash iexec ... 18, 204, 207, 234, 238, 241, 631, 635
\backslash ^ 143	\backslash eolang-phi.pl <u>24</u>	\backslash ifdefined 211, 212, 240, 244, 245, 247, 638, 639, 653
\backslash 106, 161, 162, 218, 225, 263, 624	\backslash eolang@sodg.pl ... <u>254</u>	\backslash ifluatex 12
Numbers	\backslash eolang@anonymous 14, 653	\backslash ifxetex 12
\backslash 2 105, 107, 109, 118, 147, 278	\backslash eolang@lineno <u>19</u>	\backslash inputlineno ... 206, 634
\backslash 3 107, 109	\backslash eolang@mdfive <u>20</u> , 202, 236, 629	J
\backslash 4 107	\backslash eolang@nocomments ... 13, 211, 244, 638	\backslash jobname . 18, 203, 204, 205, 207, 210, 221, 228, 234, 237, 238, 239, 241, 243, 627, 630, 631, 632, 635, 637
A	\backslash eolang@nodollar 240, 245, 247	K
\backslash a 570, 573, 581, 592, 595, 602	\backslash eolang@phiSaveTo 199, 212, 214	\backslash kern 675, 679
\backslash active . 245, 249, 252, 640	\backslash eolang@process 201, 222, 229	L
\backslash alpha 648, 662	\backslash eolang@sodgSaveTo 621, 639, 642	\backslash lambda 484
\backslash AtBeginDocument .. 252	\backslash eolang@tmpdir 11, 18, 25, 196, 203, 204, 205, 207, 208, 210, 221, 228, 234, 237, 238, 239, 241, 242, 243, 255, 528, 627, 630, 631, 632, 635, 636, 637	\backslash leadsto 669
B	\backslash ex@ 675, 679	\backslash limits 674, 678
\backslash Bbbk 3		M
\backslash begin 25, 46, 167, 170, 172, 220, 227, 255, 306, 542, 626		\backslash m@th ... 692, 693, 698, 701
\backslash box 705		\backslash makeatletter 19, 21, 24,
C		
\backslash catcode 218, 225, 240, 245, 249, 252, 624, 633, 640		
\backslash clean 233, 235		
\backslash clipbox 696, 700		
\backslash color 551		

198, 201, 232, 254, 620, 623, 651, 672, 687	<code>\phiOset</code> 672, 683	<code>\text</code> 472, 671	
<code>\makeatother</code> 19, 23, 197, 200, 230, 246, 529, 622, 643, 658, 681, 708	<code>\phiq</code> 231, 250	<code>\textcolor</code> 654	
<code>\makebox</code> 695	<code>\phiquation</code> 201	<code>\textnormal</code> 473	
<code>\mapsto</code> 667	<code>\phiSaveTo</code> 198	<code>\texttt</code> 473	
<code>\mapstochar</code> . 667, 669, 689	<code>\phiSlot</code> 670	<code>\the</code> 206, 634	
<code>\mathop</code> 674, 678	<code>\phiUset</code> 677, 683	<code>\tikz</code> 531	
<code>\mathpalette</code> 689	<code>\phiWave</code> 668	<code>\tikzinputsegmentfirst</code> 568, 589	
<code>\mathrel</code> 666, 667, 669, 674, 678, 689	<code>\pi</code> 327	<code>\tikzinputsegmentlast</code> 568, 590	
<code>\message</code> 195, 206, 527, 634	<code>\ProcessPgfPackageOptions</code> 17	<code>\tikzmath</code> 565, 587	
<code>\mspace</code> 669	<code>\protected</code> 250	<code>\tikzset</code> 559	
N			
<code>\newcommand</code> 22, 199, 201, 232, 621, 659, 661, 663, 665, 668, 670, 673, 677, 682, 684, 688, 690	Q		
<code>\newcounter</code> 19	<code>\Q</code> 139, 402, 410, 418, 421, 427, 430, 431, 432, 433	<code>\tikzstyle</code> 550, 553, 555, 556, 558, 612, 613, 614, 617	
<code>\newenvironment</code> 217, 224, 541, 623	R		
<code>\NewExpandableDocumentCommand</code> 652	<code>\relax</code> 3, 214, 642, 689	<code>\ttfamily</code> 551	
<code>\node</code> 362, 373, 376, 379, 386, 449	<code>\RequirePackage</code> . . . 1, 2, 3, 4, 5, 6, 7, 8, 20, 231, 531, 644, 685, 686	<code>\tw@</code> 693, 704	
<code>\nodollar</code> 247	<code>\resizebox</code> 697	U	
<code>\noindent</code> 542	<code>\rightarrow</code> . 692, 693, 701	<code>\usetikzlibrary</code> . . . 532, 533, 534, 535, 536, 537, 538, 539, 540	
P			
<code>\pdf@filemdfivesum</code> . 22	S		
<code>\pdfstringdefDisableCommands</code> 645	<code>\sbox</code> 694	<code>\v</code> 566, 569, 588, 591	
<code>\pgfkeys</code> 9	<code>\scriptscriptstyle</code> 676, 680	<code>\value</code> 213, 219, 226, 625, 641	
<code>\Phi</code> 269	<code>\scriptsize</code> 616, 619	<code>\varphi</code> 649, 662	
<code>\phic</code> 661	<code>\scshape</code> 671	<code>\vbox</code> 675, 679	
<code>\phiConst</code> 665	<code>\setcounter</code> 213, 219, 226, 530, 625, 641	<code>\VerbatimEnvironment</code> 218, 225, 624	
<code>\phicture</code> 541	<code>\settoheight</code> 693	<code>\vss</code> 676, 680	
<code>\phiDotted</code> . 472, 484, 685	<code>\settowidth</code> 692	<code>\vx</code> 572, 573, 594, 595	
<code>\phiDotted@</code> 689, 690	<code>\sffamily</code> 660, 671	<code>\vy</code> 572, 594	
<code>\phiEOL</code> 684	<code>\small</code> 551, 554	W	
<code>\phiMany</code> 682	<code>\sodg</code> 623	<code>\width</code> 696, 700	
	<code>\sodgSaveTo</code> 620	X	
	<code>\StrSubstitute</code> 233	<code>\xmir</code> 663	
	<code>\sxy</code> 432	<code>\xrightarrow</code> 671	
T			
<code>\t</code> 33, 284, 570, 572, 581, 582, 592, 594, 602, 603	Z		
	<code>\z@</code> 692, 694, 695, 697, 704, 705		