



# eolang: $\text{\LaTeX}$ Package for Formulas and Graphs of EO Programming Language and $\varphi$ -calculus\*

Yegor Bugayenko  
yegor256@gmail.com

2023-08-02, 0.14.0

**NB!** You must run  $\text{\TeX}$  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  $\varphi$ -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:

$\begin{aligned} \text{app} &\mapsto \llbracket \\ &\quad \rho \mapsto \xi.b.^2, \alpha_0   t \rightsquigarrow \text{TRUE}, \\ &\quad b \mapsto \llbracket \alpha_* \mapsto \Phi.\text{fn}(56), \\ &\quad \quad \varphi \mapsto \Phi.\text{string.trim}(\xi), \\ &\quad \quad \Delta \mapsto 01\text{-FE-C3} \rrbracket, \\ &\quad x \mapsto \llbracket \lambda \mapsto \emptyset \rrbracket. \end{aligned}$	<pre> 1 \documentclass{minimal} 2 \usepackage{eolang} 3 \begin{document} 4 \begin{phiquestion*} 5 app -&gt; [[ % it's abstract! 6   ^ !-&gt; \$.b.^{^2}, 0/t~&gt; TRUE, 7   b -&gt; [[ *-&gt; Q.fn(56), 8     @ -&gt; QQ.string.trim(\$), 9     D&gt; 01-FE-C3 ]]],\ 10 x -&gt; [[ \lambda ..&gt; ? ]]. 11 \end{phiquestion*} 12 \end{document} </pre>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

`phiquestion (env.)` The environment `phiquestion` lets you write a  $\varphi$ -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 “ $\varphi$ ” (`\varphi`),
- “^” maps to “ $\rho$ ” (`\rho`),
- “\$” maps to “ $\xi$ ” (`\xi`),
- “&” maps to “ $\sigma$ ” (`\sigma`),
- “?” maps to “ $\emptyset$ ” (`\varnothing`),
- “Q” maps to “ $\Phi$ ” (`\Phi`),
- “QQ” maps to “ $\dot{\Phi}$ ” (`\dot{\Phi}`),
- “->” maps to “ $\mapsto$ ” (`\mapsto`),
- “~>” maps to “ $\rightsquigarrow$ ” (`\rightsquigarrow`),
- “!->” maps to “ $\multimap$ ” (`\multimap`),
- “..>” 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  $\varphi$ 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 `\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  $\varphi$ -calculus expressions using the same simple plain-text notation. You can use dollar sign directly too:

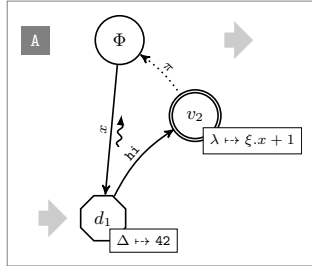
A simple object  $x \mapsto \llbracket \varphi \mapsto y \rrbracket$   
is a decorator of the data object  
 $y \mapsto \llbracket \Delta \mapsto 42 \rrbracket$ .

```

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

```

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



```

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

```

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

The following markers are supported for a vertex:

- “tag:<math>” 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.

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  $\pi$  label.

It is also possible to put transformation arrows to the graph, with the help of “ $v_0 \Rightarrow v_1$ ” 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 “ $v_3 \Rightarrow$ ”, or from the left to the vertex, by saying for example “ $\Rightarrow v_5$ .” If you want the arrow to stay further away from the vertex than usually, use a few “=” symbols, for example “ $=== \Rightarrow v_0$ .”

You can also put a marker at the left side of a vertex, using “ $v_5!A$ ” syntax, where “ $v_5$ ” 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 “ $v_5!!!A$ ” 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: “ $v_0+a$ .” Here, we make a copy of “ $v_0$ ” and call it “ $v_0a$ .” See the example below.

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

`\eolang` There is also a no-argument command `\eolang` to help you print the name of EO language. It understands the anonymous package option and prints itself differently, to `\phic` double-blind your paper. There is also `\phic` command to print the name of  $\varphi$ -calculus, `\xmirl` also sensitive to anonymous mode. The macro `\xmirl` prints “XMIR”.

In our research we use XYZ,  
an experimental object-oriented  
dataflow language,  $\alpha$ -calculus, as its  
formal foundation, and XML<sup>+</sup> —  
its XML-based presentation.

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

Without the anonymous option there will be no orange color:

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

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

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

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

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

`\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:

$$\llbracket \mapsto^* \rrbracket.$$

```
6 | When the names of attributes and their
7 | values don't matter, we use an arrow
8 | with a star, for example:
9 | \begin{phiuation*}
10 | \llbracket \phiOset{*}\{->\} \rrbracket.
11 | \end{phiuation*}
```

`\phiMany`     Sometimes you may need to simplify the way you describe an object (the typesetting is a bit off, but this is not because of us, but because of [this](#)):

The expression  $\llbracket \alpha_1 \mapsto x_1, \alpha_2 \mapsto x_2, \dots, \alpha_n \mapsto x_n \rrbracket$  and expression  $\llbracket \alpha_i \xrightarrow{*} x_i \rrbracket$  are syntactically different but semantically equivalent.

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

`\phiSaveTo`     If you want to use `phiuation` or `sodg` environments inside `tabular` or any other  
`\sodgSaveTo` environment or command, you won't be able to do this, because `phiuation` and `sodg` are “verbatim” environments. `\phiSaveTo` and `\sodgSaveTo` commands will help you in this situation. You use them right before `\begin{phiuation}` or `\begin{sodg}` respectively — the content of the equation or the graph won't be rendered, but instead saved to the file. Later, inside `tabular`, you can use it through the `\input` macro (don't forget the `\parbox`):

Free:      $\llbracket x \mapsto \emptyset \rrbracket$

Bound:     $\llbracket x \mapsto \llbracket \Delta \vdash 42 \rrbracket \rrbracket$

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

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

## 2 Package Options

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

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

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

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

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

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

### 3 More Examples

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

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

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

$\frac{\begin{array}{l} x \mapsto \llbracket \varphi \mapsto y, z \mapsto 42, \\ \alpha_0 \mid g \mapsto \varnothing, \alpha_1 \mid \text{foo} \mapsto 42 \rrbracket \\ x \mapsto \llbracket \varphi \mapsto y, z \mapsto \varnothing, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto \llbracket \psi \rightsquigarrow \text{hello}(12) \rrbracket, \\ \alpha_1 \mapsto 42) \rrbracket \end{array}}{R2.}$	<pre> 6 \begin{phiquation*} 7 \dfrac{\begin{split} 8 x-&gt;[[@-&gt;y, z-&gt;42, 9   0/g-&gt;?, 1/foo-&gt;42]] 10 \end{split}}{\begin{split} 11 x-&gt;[[@-&gt;y, z-&gt;?, f ~&gt;  pi ( 12   0-&gt;[[ \psi !-&gt;  hello (12) ]], 13   1-&gt;42)]] 14 \end{split}}\text{R2}. 15 \end{phiquation*} </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

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

The `cases` environment works too:

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

```

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

```

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

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

```

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

```

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

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

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

```

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

```

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

$$\begin{cases} \alpha_0 \mapsto x \\ \alpha_0 \mapsto \emptyset \\ x(\Delta \mapsto 42) \end{cases}$$

This is formation  
Abstraction  
Application

```

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

```

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

$$\text{The object formation } [\alpha_0 \mapsto x] \text{ may be replaced with a formula } Q \times a^2.$$

```

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

```

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

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

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

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

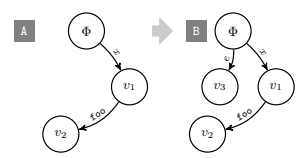
It is possible to use “manual splitting” mode in the `phiquation` environment by starting the body with `\begin{split}`:

$\begin{aligned} x(\pi) &\mapsto 4 \\ x(a, b, c) &\mapsto [\alpha_0 \mapsto \emptyset] \end{aligned}$	<pre> 5 \begin{phiquation*} 6 \begin{split} 7 x(\pi) &amp;\&amp; -&gt; 4 \\ 8 x(a,b,c) &amp;\&amp; -&gt; [[ \alpha_0 -&gt; ? ]] \\ 9 \end{split} 10 \end{phiquation*} </pre>
-------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

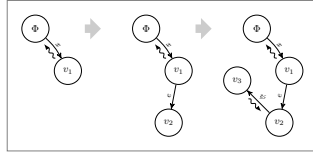
When necessary to use a percentage sign, prepend it with a backward slash:

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

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

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

You can make a copy from a copy:

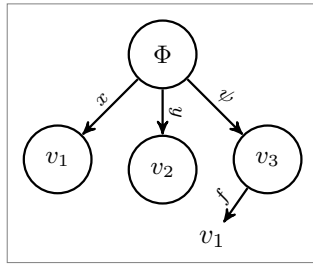


```

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:

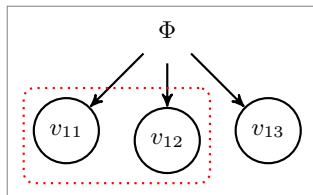


```

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}

```

## 4 Implementation

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

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

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

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

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

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

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

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

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

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

Then, we process package options:

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

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

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

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

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

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

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

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

```
24 \makeatletter
25 \begin{VerbatimOut}{\eolang@tmpdir/eolang-phi.pl}
26 $macro = $ARGV[0];
27 open(my $fh, '<', $ARGV[1]);
28 my $tex; { local $/; $tex = <$fh>; }
29 print "% This file is auto-generated by 0.14.0\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\\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   $tex =~ s/\\n{3,}/\\n\\n/g;
75 }
76 my @texts = ();
77 sub trep {
78   my ($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]|^)\QQ(?![a-z0-9])/\1\dot{\Phi{}}/g;
101 $tex =~ s/([~\{a-z0-9]|^)\Q(?![a-z0-9])/\1\Phi{/g;
102 $tex =~ s/([~\{a-z0-9]|^)\D>/\1\Delta{}/g;
103 $tex =~ s/([~\{a-z0-9]|^)\L>/\1\lambda{}/g;
104 $tex =~ s/"([~"]+)"|"\1"/g;
105 $tex =~ s/(\^(?<=[\s](\[\,.\>/)))([a-zA-Z][a-z0-9]+)(?=[\s](\[\,.\>|$\))/\2/g;
106 $tex =~ s/([~_]|^)([0-9]+\|*\)/(\^[a-z]+\|+[a-z]+\|)
107 (->|\.\>|^>|:=|!->)/\1\alpha_{2}\vert\3\space{4}/g;
108 $tex =~ s/([~_]|^)([0-9]+\|*)
109 (->|\.\>|^>|:=|!->)/\1\alpha_{2}\space{3}/g;
110 if ($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;
117     $tex =~ s/\\\n/g;
118     $tex =~ s/([~&\s])\s{2}([~\s])/\1 \2/g;
119     $tex =~ s/\s{2}/\quad/g;
120     $tex = '&' . $tex;
121   }
122   my $lead = '[~\s]+\s(?:->|:=|!>)\s';
123   my @leads = $tex =~ /&{$lead}/g;
124   my @eols = $tex =~ /\n/g;
125   if (0+@leads == 0+@eols && 0+@eols > 1) {
126     $tex =~ s/&{$lead})/\1~/g;
127     $gathered = 0;
128     print '% The "gathered" is NOT used because all ' .
129       (0+@eols) . ' lines are ' . (0+@leads) . " leads\n";
130   }
131 }
132 if ($macro ne 'phiq') {
133 sub strip_tabs {
134   my ($env, $tex) = @_;
135   $tex =~ s/&/g;
136   return "\begin{$env}" . $tex . "\end{$env}";
137 }
138 foreach my $e ('matrix', 'cases') {
139 $tex =~ s/\begin{($e\{*\})\}(.\+)\end{($e\{*\})}/strip_tabs($1, $2)/sge;
140 }
141 }
142 $tex =~ s/\$/\xi{/g;
143 $tex =~ s/(?<!\{)\^(?!\\)/\rho{/g;
144 $tex =~ s/[[/\llbracket\mathbin{/g;
145 $tex =~ s/[ ]/\mathbin{\rrbracket}/g;
146 $tex =~ s/([~\s,>])([0-9A-F]{2})(?:-[0-9A-F]{2})+|
147 [0-9]+(?:\.[0-9]+)?)(?!\\)/\1|\2|/g;
148 $tex =~ s/TRUE/|TRUE|/g;
149 $tex =~ s/FALSE/|FALSE|/g;
150 $tex =~ s/\?/\varnothing{/g;

```

```

151 $tex =~ s/@/\varphi{/g;
152 $tex =~ s/-([a-z]+)>/\mathrel{\phiSlot{1}}/g;
153 $tex =~ s/!->/\mathbin{\phiConst}/g;
154 $tex =~ s/->/\mathbin{\mapsto}/g;
155 $tex =~ s/~>/\mathbin{\phiWave}/g;
156 $tex =~ s/:=\mathrel{\vDash}/g;
157 $tex =~ s/==\mathrel{\equiv}/g;
158 $tex =~ s/\.\.\>/\mathbin{\phiDotted}/g;
159 $tex =~ s/<</\langle/g;
160 $tex =~ s/>>/\rangle/g;
161 $tex =~ s/\|{2,}/|/g;
162 $tex =~ s/|([^\|]+)|\textnormal{\texttt{1}}{}/g;
163 $tex =~ s/{\TEXT(d+)\}/'\text{' . @texts[1] . '}'/ge;
164 if ($macro eq 'phiq') {
165   print '$' if ($tex ne '');
166 } else {
167   print '\begin{' , $macro, "}\n";
168   if (not($align)) {
169     if ($gathered) {
170       print '\begin{gathered}' . "\n";
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 `phi`uation 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

```

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

```

201 \makeatletter\newcommand\eolang@process[1]{
202   \def\hash{\eolang@mdfive
203     {\eolang@tmpdir/\jobname/phiquation.tex}}%
204   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiquation.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{phiquation*}%
218 {\catcode'\|=12 \VerbatimEnvironment%
219 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
220 \begin{VerbatimOut}
221   {\eolang@tmpdir/\jobname/phiquation.tex}}
222 {\end{VerbatimOut}\eolang@process{equation*}}
223 %
224 \newenvironment{phiquation}%
225 {\catcode'\|=12 \VerbatimEnvironment%
226 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
227 \begin{VerbatimOut}
228   {\eolang@tmpdir/\jobname/phiquation.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$#1${\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{\texttt{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 /\n/g, $tex) {
292   print '% ', $t, "\n";
293 }
294 print "% ---\n";
295 $tex =~ s/\\\\\\/\n/g;
296 $tex =~ s/\\\\\n//g;
297 $tex =~ s/(\\[a-zA-Z]+)\s+/\1/g;
298 $tex =~ s/\n{2,}/\n/g;
299 my @cmds = split /\n/g, $tex;

```

```

300 print '% --- before processing:' . "\n";
301 foreach my $t (split (/\\n/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   $cmd =~ s/(?!\s)%.*//g;
311   my ($head, $tail) = split(/ /, $cmd, 2);
312   my %opts = {};
313   foreach my $p (split(/ /, $tail)) {
314     my ($q, $t) = split(/:/, $p);
315     $opts{$q} = $t;
316   }
317   if (index($head, '\\') == 0) {
318     print $cmd;
319   } elsif (index($head, '->') >= 0) {
320     my $draw = '\draw[';
321     if (exists $opts{'pi'}) {
322       $draw = $draw . '<MB:phi-pi><F:draw=none>';
323       if (not exists $opts{'a'}) {
324         $opts{'a'} = '\pi';
325       }
326     }
327     if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
328       $draw = $draw . '<MB:,phi-rho>';
329     }
330     $draw = $draw . ']';
331     my ($from, $to) = split (/->/, $head);
332     $draw = $draw . " (${$from}) ";
333     if (exists $opts{'bend'}) {
334       $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
335         num($opts{'bend'}) . '>';
336       if (exists $opts{'rho'}) {
337         $draw = $draw . '<MB:,phi-rho>';
338       }
339       $draw = $draw . ']';
340     } else {
341       $draw = $draw . '--';
342     }
343     if (exists $opts{'a'}) {
344       my $a = $opts{'a'};
345       if (index($a, '$') == -1) {
346         $a = '$' . fmt($a) . '$';
347       } else {
348         $a = fmt($a);
349       }
350       $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
351     }
352     if (exists $opts{'break'}) {
353       $draw = $draw . '<F: coordinate [pos=' .

```

```

354         ($opts{'break'} / 100) . ']' (break)>';
355     }
356     $draw = $draw . " (<MF:${to}><B:break-v>";
357     if (exists $opts{'break'}) {
358         print tailor($draw, 'F') . ";\n";
359         print ' \node[outer sep=.1cm,inner sep=0cm] ' .
360             'at (break) (break-v) {$' . vertex($to) .
361             '$};' . "\n";
362         print ' ' . tailor($draw, 'B');
363     } else {
364         print tailor($draw, 'M');
365     }
366 } elsif (index($head, '=>') >= 0) {
367     my ($from, $to) = split (/=>/, $head);
368     my $size = () = $head =~ /=/g;
369     if ($from eq '') {
370         print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
371             $to . '.center]';
372     } elsif ($to eq '') {
373         print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
374             $from . '.center]';
375     } else {
376         print '\node [phi-arrow] at ($(' .
377             $from . ')!0.5!(' . $to . ')$)';
378     }
379     print '{}';
380 } elsif (index($head, '!'') >= 0) {
381     my ($v, $marker) = split (/!/, $head);
382     my $size = () = $head =~ /*!/g;
383     print '\node [phi-marker, left=' .
384         ($size * 0.6) . 'cm of ' .
385         $v . '.center]{' . fmt($marker) . '}';
386 } elsif (index($head, '+') >= 0) {
387     my ($v, $suffix) = split (/+/, $head);
388     my @friends = ($v);
389     foreach my $c (@cmds) {
390         $e = $c;
391         $e =~ s/^\s+//g;
392         my $h = $e;
393         $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
394         foreach my $f (@friends) {
395             my $add = '';
396             if (index($h, $f . '->') >= 0) {
397                 $add = substr($h, index($h, '->') + 2);
398             }
399             if ($h =~ /\->\Q${f}\E/) {
400                 $add = substr($h, 0, index($h, '->'));
401             }
402             if (index($e, ' xy:' . $f . ',') >= 0) {
403                 $add = $h;
404             }
405             if (index($add, '+') == -1
406                 and $add ne ''
407                 and not(grep(/\Q${add}\E/, @friends))) {

```

```

408         push(@friends, $add);
409     }
410 }
411 }
412 my @extra = ();
413 foreach my $e (@cmds) {
414     $m = $e;
415     if ($m =~ /\s*\Q${v}\E\s/) {
416         next;
417     }
418     if ($m =~ /\s*[^\s]+\s/ and not($m =~ /\s*\Q${head}\E\s/)) {
419         next;
420     }
421     foreach my $f (@friends) {
422         my $h = $f;
423         $h =~ s/[a-z]$//g;
424         if ($m =~ s/^(s*)\Q${f}\E+\Q${suffix}\E\s?/\1${h}${suffix} /g) {
425             last;
426         }
427         $m =~ s/^(s*)\Q${f}\E\s/\1${h}${suffix} /g;
428         $m =~ s/^(s*)\Q${f}\E->/\1${h}${suffix}->/g;
429         $m =~ s/\sxy:\Q${f}\E,/ xy:${h}${suffix},/g;
430         $m =~ s/->\Q${f}\E\s/->${h}${suffix} /g;
431     }
432     if ($m ne $e) {
433         push(@extra, ' ' . $m);
434     }
435 }
436 splice(@extra, 0, 0, @extra[-1]);
437 splice(@extra, -1, 1);
438 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
439     '), friends: [' . join(', ', @friends) . ']' in ' .
440     (0+@cmds) . ' lines');
441 splice(@cmds, $c, 1, @extra);
442 print '% cloned ' . $v . ' at line no.' . $c .
443     ' (+ ' . (0+@extra) . ' lines -> ' .
444     (0+@cmds) . ' lines total)';
445 } elsif ($head =~ /\v[0-9]+[a-z]?$/) {
446     print '\node[';
447     if (exists $opts{'xy'}) {
448         my ($v, $right, $down) = split(/,/ , $opts{'xy'});
449         my $loc = '';
450         if ($down > 0) {
451             $loc = 'below ';
452         } elsif ($down < 0) {
453             $loc = 'above ';
454         }
455         if ($right > 0) {
456             $loc = $loc . 'right';
457         } elsif ($right < 0) {
458             $loc = $loc . 'left';
459         }
460         print ', ' . $loc . '=';
461         print abs(num($down)) . 'cm and ' .

```

```

462         abs(num($right)) . 'cm of ' . $v . '.center';
463     }
464     if (exists $opts{'data'}) {
465         print ',phi-data';
466         if ($opts{'data'} ne '') {
467             my $d = $opts{'data'};
468             if (index($d, '|') == -1) {
469                 $d = '$\Delta\phiDotted\text{' .
470                     '\textnormal{\texttt{' . fmt($d) . '}}}$';
471             } else {
472                 $d = fmt($d);
473             }
474             $opts{'box'} = $d;
475         }
476     } elsif (exists $opts{'atom'}) {
477         print ',phi-atom';
478         if ($opts{'atom'} ne '') {
479             my $a = $opts{'atom'};
480             if (index($a, '$') == -1) {
481                 $a = '$\lambda\phiDotted{' . fmt($a) . '$';
482             } else {
483                 $a = fmt($a);
484             }
485             $opts{'box'} = $a;
486         }
487     } else {
488         print ',phi-object';
489     }
490     if (exists $opts{'edgeless'}) {
491         print ',draw=none';
492     }
493     print ']';
494     print ' (' . $head . ')';
495     print '{';
496     if (exists $opts{'tag'}) {
497         my $t = $opts{'tag'};
498         if (index($t, '$') == -1) {
499             $t = '$' . $t . '$';
500         } else {
501             $t = fmt($t);
502         }
503         print $t;
504     } else {
505         print '$' . vertex($head) . '$';
506     }
507     print '}';
508     if (exists $opts{'box'}) {
509         print ' node[phi-box] at (';
510         print $head, '.south east) {';
511         print $opts{'box'}, ')';
512     }
513 }
514 print ";\n";
515 }

```

```

516 print '\end{phicture}%', "\n";
517 print "% --- after processing:\n%";
518 foreach my $c (@cmds) {
519   print '% ', $c, "\n";
520 }
521 print '% --- (' . (0+@cmds) . " lines)\n";
522 print '\endinput';
523 \end{VerbatimOut}
524 \message{eolang: File with Perl script
525   '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
526 \makeatother

```

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

```

527 \setcounter{FancyVerbLine}{0}

```

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

```

528 \RequirePackage{tikz}
529 \usetikzlibrary{arrows}
530 \usetikzlibrary{shapes}
531 \usetikzlibrary{decorations}
532 \usetikzlibrary{decorations.pathmorphing}
533 \usetikzlibrary{decorations.pathreplacing}
534 \usetikzlibrary{positioning}
535 \usetikzlibrary{calc}
536 \usetikzlibrary{math}
537 \usetikzlibrary{arrows.meta}

```

phicture Then, we define internal environment phicture:

```

538 \newenvironment{phicture}%
539   {\noindent\begin{tikzpicture}[
540     ->,>=stealth',node distance=0,thick,
541     pics/parallel arrow/.style={
542       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
543   {\end{tikzpicture}}
544 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
545   minimum height=0.5cm, minimum width=0.5cm,
546   single arrow head extend=2mm]
547 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
548   minimum width=1.4em, font={\small\color{white}\ttfamily},
549   fill=gray]
550 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
551   draw,font={\small}]
552 \tikzstyle{phi-object} = [phi-thing,circle]
553 \tikzstyle{phi-data} = [phi-thing,regular polygon,
554   regular polygon sides=8]
555 \tikzstyle{phi-empty} = [phi-object]
556 \tikzset{%
557   phi-rho/.style={
558     postaction={%
559       decoration={
560         show path construction,
561         curveto code={
562           \tikzmath{

```

```

563         coordinate \I, \F, \v;
564         \I = (\tikzinputsegmentfirst);
565         \F = (\tikzinputsegmentlast);
566         \v = ($(\I) -(\F)$);
567         real \d, \a, \r, \t;
568         \d = 0.8;
569         \t = atan2(\vy, \vx);
570         if \vx<0 then { \a = 90; } else { \a = -90; };
571         {
572             \draw[arrows={-latex}, decorate,
573                 decoration={%
574                     snake, amplitude=.4mm,
575                     segment length=2mm,
576                     post length=1mm
577                 }]
578             ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
579             -- ++(\t: 2*\d em);
580         };
581     },
582 },
583 \lineto code={
584     \tikzmath{
585         coordinate \I, \F, \v;
586         \I = (\tikzinputsegmentfirst);
587         \F = (\tikzinputsegmentlast);
588         \v = ($(\I) -(\F)$);
589         real \d, \a, \r, \t;
590         \d = 0.8;
591         \t = atan2(\vy, \vx);
592         if \vx<0 then { \a = 90; } else { \a = -90; };
593         {
594             \draw[arrows={-latex}, decorate,
595                 decoration={%
596                     snake, amplitude=.4mm,
597                     segment length=2mm,
598                     post length=1mm}]
599             ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
600             -- ++(\t: 2*\d em);
601         };
602     }
603 },
604 },
605 decorate
606 }
607 }
608 }
609 \tikzstyle{phi-pi} = [draw,dotted]
610 \tikzstyle{phi-atom} = [phi-object,double]
611 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
612     rectangle,thin,minimum width=1.2em,anchor=north west,
613     font={\scriptsize}]
614 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
615     above=2pt,sloped/.append style={transform shape},
616     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:

```
617 \makeatletter
618 \newcommand\sodgSaveTo[1]{\def\eolang@sodgSaveTo{#1}}
619 \makeatother
```

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

```
620 \makeatletter\newenvironment{sodg}%
621 {\catcode'\|=12 \VerbatimEnvironment%
622 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
623 \begin{VerbatimOut}
624   {\eolang@tmpdir/\jobname/sodg.tex}}
625 {\end{VerbatimOut}}%
626 \def\hash{\eolang@mdfive
627   {\eolang@tmpdir/\jobname/sodg.tex}}%
628 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
629   "\eolang@tmpdir/\jobname/\hash.tex"}%
630 \catcode'\$=3 %
631 \message{Start parsing 'sodg' at line no. \the\inputlineno^^J}
632 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
633   perl "\eolang@tmpdir/eolang-sodg.pl"
634   "\eolang@tmpdir/\jobname/\hash.tex"
635   \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\\n|$\)//g'\fi
636   \ifdefined\eolang@sodgSaveTo > \eolang@sodgSaveTo\fi}%
637 \catcode'\$=active%
638 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
639 \def\eolang@sodgSaveTo{\relax}%
640 }\makeatother
```

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

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

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

```
656 \newcommand\eolang{%
657   \eoAnon[XYZ]{\sffamily EO}}
```

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

```

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

\xmire Then, we define a simple supplementary command to help you print XMIR, the name of
our XML-based format of program representation.
660 \newcommand\xmir{%
661   \eoAnon[XML\(^+\)]{XMIR}}

\phiConst Then, we define a command to render an arrow for a constant attribute, as suggested
here:
662 \newcommand\phiConst{%
663   \mathrel{\hspace{.15em}}%
664   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

\phiWave Then, we define a command to render an arrow for a multi-layer attribute, as suggested
here:
665 \newcommand\phiWave{%
666   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

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

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

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

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

\phiDotted Then, we define a command to render an arrow for a special attribute, as suggested here:
682 \RequirePackage{trimclip}
683 \RequirePackage{amsfonts}
684 \makeatletter
685 \newcommand{\phiDotted}{%
686   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
687 \newcommand{\phiDotted@}[2]{%
688   \begingroup%
689   \settoheight{\dimen\z@}{\m@th#1\rightarrow}%
690   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%

```

```

691 \sbox\z@{%
692   \makebox[\dimen\z@][s]{%
693     \clipbox{0 0 {0.4\width} 0}%
694     {\resizebox{\dimen\z@}{\height}%
695       {\$ \m@th#1 \dashrightarrow$}}%
696     \hss%
697     \clipbox{{0.69\width} {-0.1\height} 0
698       {-\height}}{\$ \m@th#1 \rightarrow$}%
699   }%
700 }%
701 \ht\z@=\dimen\tw@ \dp\z@=\z@%
702 \box\z@%
703 \endgroup%
704 }
705 \makeatother

```

## References

- Bugayenko, Yegor (2021). *EOLANG and  $\varphi$ -calculus*. arXiv: [2111.13384](#) [cs.PL].
- Kudasov, Nikolai et al. (2022).  *$\varphi$ -calculus: a purely object-oriented calculus of decorated objects*. arXiv: [2204.07454](#) [cs.PL].

## Change History

0.0.1	General: First draft. . . . .	9	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 QQ into <code>\dot{\Phi}</code> implemented. . . . .	10
	<code>eolang-phi.pl</code> : New symbol added for basket slots . . . . .	10	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> ) . . . . .	10	0.2.0	<code>eolang-phi.pl</code> : Numbers automatically render as <code>\texttt</code> . No need to use vertical bars around them anymore. . . . .	10
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets . . . . .	10		<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>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>\xmirl</code> : New command <code>\xmirl</code> prints XMIR in both normal and the anonymous mode of <code>acmart</code> . . . . .	23
	<code>\phiq</code> : Parsing of additional symbols enabled. . . . .	14	0.3.0	<code>\eolang@lineno</code> : New counter for protecting <code>lineno</code> . . . . .	10
0.1.0	General: Parsing of package options introduced. . . . .	10		<code>eolang-phi.pl</code> : New arrow added, that looks like <code>\leadsto</code> . . . . .	10
	<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> . . . . .	22		<code>\phiWave</code> : New command <code>\phiWave</code> added to denote a link to a multi-layer attribute. . . . .	23
	<code>\eolang@mdfive</code> : New supplementary command added to calculate MD5 sum of a file. . . . .	10	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-phi.pl</code> : A new Perl script “ <code>eolang-phi.pl</code> ” added for parsing of <code>phi</code> expressions. . . . .	10		Relative positioning of vertices fixed. . . . .	15
	<code>eolang-sodg.pl</code> : There are two Perl scripts now: one for <code>phi</code> quation, another one for <code>sodg</code> . . . . .	15	0.5.0	<code>eolang-phi.pl</code> : Automated formatting of <code>TRUE</code> and <code>FALSE</code> added. . . . .	10
	<code>\phic</code> : New command <code>\phic</code> prints the name of $\varphi$ -calculus in both normal and the anonymous mode of <code>acmart</code> . . . . .	22		<code>eolang-sodg.pl</code> : It is possible to use TikZ commands inside the <code>sodg</code> environment. . . . .	15
	<code>\phiConst</code> : New command <code>\phiConst</code> added to denote a link to a constant attribute. . . . .	23		New syntax introduced that allows to make clones of vertices and all their dependants. . . . .	15
	<code>\phiDotted</code> : New command <code>\phiDotted</code> added to denote a link to a special attribute. . . . .	23		Now edges may have the <code>break</code> attribute, to make them shorter. . . . .	15
				<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow. . . . .	23

<p><code>\phiSlot</code>: New command <code>\phiSlot</code> added to denote a link to a slot in a basket. . . . . 23</p> <p>0.6.0</p> <p>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</p> <p><code>eolang-sodg.pl</code>: The <code>rrho</code> attribute is retired, now <code>rho</code> works just fine in all situations. . . . . 15</p> <p>0.7.0</p> <p><code>nodollar</code>: Now it is possible to use dollar sign instead of the <code>\phiq</code> command. . . . . 14</p> <p><code>eolang-phi.pl</code>: New syntax sugar for <math>\Phi</math>, just using capital “Q” is enough. 10</p> <p>Object names are automatically converted to <code>\texttt</code>, provided their names include two or more symbols. . . . . 10</p> <p>Text in quotes is automatically converted to <code>\texttt</code>. . . . . 10</p>	<p>0.8.0</p> <p>General: The anonymous package option added. . . . . 10</p> <p><code>eolang-phi.pl</code>: Inside <code>phiuation</code> any text inside the <code>\text</code> macro is not processed. . . . . 10</p> <p><code>eolang-sodg.pl</code>: The <code>tag</code> attribute is introduced for changing labels inside a vertex circle. . . . . 15</p> <p><code>\phi0set</code>: New commands <code>\phi0set</code> and <code>\phiUset</code> help position text over and under an arrow. . . . . 23</p> <p><code>\phiSaveTo</code>: The output of the <code>phiuation</code> environment can be redirected to a file. . . . . 13</p> <p><code>\sodgSaveTo</code>: The output of the <code>sodg</code> environment can be redirected to a file. . . . . 22</p> <p>0.9.0</p> <p><code>\eoAnon</code>: New command <code>\eoAnon</code> added. . . . . 22</p> <p><code>eolang-phi.pl</code>: Proper handling of the <code>matrix</code> environment. . . . . 10</p> <p><code>\phiEOL</code>: New command <code>\phiEOL</code> added, instead of <code>\[-4pt]</code>. . . . . 23</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# 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		
<code>\\$</code> .....	142, 240, 245, 249, 252, 630, 637	
<code>\%</code> .....	211, 244, 635	
<code>\(</code> .....	643, 659, 661	
<code>\)</code> .....	644, 659, 661	
<code>\*</code> .....	106, 108, 139	
<code>\+</code> .....	258, 387, 418, 424	
<code>\-</code> .....	659	
<code>\.</code> ..	107, 109, 147, 158, 258	
<code>\/</code> .....	105, 106	
<code>\?</code> .....	150	
<code>\[</code> .....	105, 144	
<code>\{</code> ....	44, 73, 96, 112, 113, 139, 143, 147, 163	
<code>\}</code> ....	44, 73, 112, 113, 139, 163	
<code>\]</code> .....	105, 145	
<code>\^</code> .....	143	
<code>\ </code> .....	106, 161, 162, 218, 225, 263, 621	
Numbers		
<code>\2</code> ..	105, 107, 109, 118, 147, 278	
<code>\3</code> .....	107, 109	
<code>\4</code> .....	107	
A		
<code>\a</code> ..	567, 570, 578, 589, 592, 599	
<code>\active</code> .	245, 249, 252, 637	
<code>\alpha</code> .....	645, 659	
<code>\AtBeginDocument</code> ..	252	
B		
<code>\Bbbk</code> .....	3	
<code>\begin</code> .....	25, 46, 167, 170, 172, 220, 227, 255, 306, 539, 623	
<code>\box</code> .....	702	
C		
<code>\catcode</code> .....	218, 225, 240, 245, 249, 252, 621, 630, 637	
<code>\clean</code> .....	233, 235	
<code>\clipbox</code> .....	693, 697	
<code>\color</code> .....	548	
D		
<code>\d</code> ..	163, 567, 568, 578, 579, 589, 590, 599, 600	
<code>\dashrightarrow</code> ...	695	
<code>\def</code> .....	199, 202, 214, 236, 618, 626, 639, 643, 644, 645, 646	
<code>\Delta</code> .....	469	
<code>\detokenize</code> .....	233	
<code>\dimen</code> ..	689, 690, 692, 694, 701	
<code>\dp</code> .....	701	
<code>\draw</code> ...	320, 542, 572, 594	
E		
<code>\E</code> ..	139, 399, 407, 415, 418, 424, 427, 428, 429, 430	
<code>\end</code> .....	186, 188, 191, 194, 222, 229, 516, 523, 543, 625	
<code>\endinginput</code> .....	193, 522	
<code>\eoAnon</code> .	641, 657, 659, 661	
<code>\eolang</code> .....	656	
<code>\eolang-phi.pl</code> .....	24	
<code>\eolang-sodg.pl</code> ...	254	
<code>\eolang@anonymous</code> ..	14, 650	
<code>\eolang@lineno</code> .....	19	
<code>\eolang@mdfive</code> ....	20, 202, 236, 626	
<code>\eolang@nocomments</code> ..	13, 211, 244, 635	
<code>\eolang@nodollar</code> ..	240, 245, 247	
<code>\eolang@phiSaveTo</code> .	199, 212, 214	
<code>\eolang@process</code> ...	201, 222, 229	
<code>\eolang@sodgSaveTo</code> ..	618, 636, 639	
<code>\eolang@tmpdir</code> ....	11, 18, 25, 196, 203, 204, 205, 207, 208, 210, 221, 228, 234, 237, 238, 239, 241, 242, 243, 255, 525, 624, 627, 628, 629, 632, 633, 634	
<code>\ex@</code> .....	672, 676	
F		
<code>\F</code> .....	563, 565, 566, 578, 585, 587, 588, 599	
<code>\FancyVerbLine</code> ....	527	
G		
<code>\gdef</code> .....	250	
H		
<code>\hash</code> ...	202, 205, 207, 210, 236, 239, 241, 243, 626, 629, 632, 634	
<code>\hbox</code> .....	673, 677	
<code>\height</code> ....	694, 697, 698	
<code>\hspace</code> .....	663, 664	
<code>\hss</code> .....	696	
<code>\ht</code> .....	701	
I		
<code>\I</code> .....	563, 564, 566, 578, 585, 586, 588, 599	
<code>\iexec</code> ...	18, 204, 207, 234, 238, 241, 628, 632	
<code>\ifdefined</code> .....	211, 212, 240, 244, 245, 247, 635, 636, 650	
<code>\ifluatex</code> .....	12	
<code>\ifxetex</code> .....	12	
<code>\inputlineno</code> ...	206, 631	
J		
<code>\jobname</code> .	18, 203, 204, 205, 207, 210, 221, 228, 234, 237, 238, 239, 241, 243, 624, 627, 628, 629, 632, 634	
K		
<code>\kern</code> .....	672, 676	
L		
<code>\lambda</code> .....	481	
<code>\leadsto</code> .....	666	
<code>\limits</code> .....	671, 675	
M		
<code>\m@th</code> ...	689, 690, 695, 698	
<code>\makeatletter</code> .....	19, 21, 24,	

198, 201, 232, 254, 617, 620, 648, 669, 684	\makeatother 19, 23, 197, 200, 230, 246, 526, 619, 640, 655, 678, 705	\makebox ..... 692	\mapsto ..... 664	\mapstochar . 664, 666, 686	\mathop ..... 671, 675	\mathpalette ..... 686	\mathrel ..... 663, 664, 666, 671, 675, 686	\message 195, 206, 524, 631	\mspace ..... 666		
<b>N</b>											
\newcommand ..... 22, 199, 201, 232, 618, 656, 658, 660, 662, 665, 667, 670, 674, 679, 681, 685, 687	\newcounter ..... 19	\newenvironment ... ... 217, 224, 538, 620	\NewExpandableDocumentCommand ..... 649	\node ..... 359, 370, 373, 376, 383, 446	\nodollar ..... 247	\noindent ..... 539	<b>P</b>				
\pdf@filemdfivesum . 22	\pdfstringdefDisableCommands ..... 642	\pgfkeys ..... 9	\Phi ..... 269	\phic ..... 658	\phiConst ..... 662	\phicture ..... 538	\phiDotted . 469, 481, 682	\phiDotted@ .... 686, 687	\phiEOL ..... 681		
\phiMany ..... 679	\phiOset ..... 669, 680	\phiq ..... 231, 250	\phiquation ..... 201	\phiSaveTo ..... 198	\phiSlot ..... 667	\phiUset ..... 674, 680	\phiWave ..... 665	\pi ..... 324	\ProcessPgfPackageOptions ..... 17		
\protected ..... 250	<b>Q</b>										
\Q 139, 399, 407, 415, 418, 424, 427, 428, 429, 430	<b>R</b>										
\relax .... 3, 214, 639, 686	\RequirePackage .. 1, 2, 3, 4, 5, 6, 7, 8, 20, 231, 528, 641, 682, 683	\resizebox ..... 694	\rightarrow . 689, 690, 698	<b>S</b>							
\sbox ..... 691	\scriptscriptstyle ..... 673, 677	\scriptsize .... 613, 616	\scshape ..... 668	\setcounter .... 213, 219, 226, 527, 622, 638	\settoheight ..... 690	\settowidth ..... 689	\sffamily ..... 657, 668	\small ..... 548, 551	\sodg ..... 620		
\sodgSaveTo ..... 617	\StrSubstitute .... 233	\sxy ..... 429	<b>T</b>								
\t 33, 284, 567, 569, 578, 579, 589, 591, 599, 600	<b>Z</b>										
\z@ 689, 691, 692, 694, 701, 702	\text ..... 469, 668	\textcolor ..... 651	\textnormal ..... 470	\texttt ..... 470	\the ..... 206, 631	\tikz ..... 528	\tikzinputsegmentfirst ..... 564, 586	\tikzinputsegmentlast ..... 565, 587	\tikzmath ..... 562, 584		
\tikzset ..... 556	\tikzstyle .... 544, 547, 550, 552, 553, 555, 609, 610, 611, 614	\ttfamily ..... 548	\tw@ ..... 690, 701	<b>U</b>							
\usetikzlibrary ... 529, 530, 531, 532, 533, 534, 535, 536, 537	<b>V</b>										
\v ..... 563, 566, 585, 588	\value 213, 219, 226, 622, 638	\varphi ..... 646, 659	\vbox ..... 672, 676	\VerbatimEnvironment ..... 218, 225, 621	\vss ..... 673, 677	\vx ..... 569, 570, 591, 592	\vy ..... 569, 591	<b>W</b>			
\width ..... 693, 697	<b>X</b>										
\xmir ..... 660	\xrightarrow ..... 668	<b>Z</b>									