



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

Yegor Bugayenko
yegor256@gmail.com

2022-11-18, 0.7.0

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

1 Introduction

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

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

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

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

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

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

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

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

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

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

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

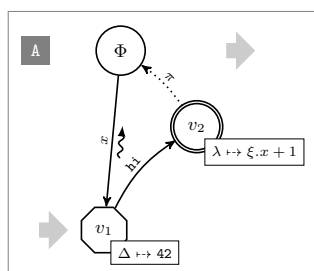
Object names are automatically converted to fixed-width font too, if they have more than one letter.

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

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

<p>A simple object $x \mapsto \llbracket \varphi \mapsto y \rrbracket$ is a decorator of the data object $y \mapsto \llbracket \Delta \mapsto 42 \rrbracket$.</p>	<pre> 4 \begin{document} 5 A simple object 6 \phiiq{x -> [[@ -> y]]} \ 7 is a decorator of 8 the data object \ 9 \$y -> [[\Delta .> 42]]\$. 10 \end{document} </pre>
---	--

`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 \\\ =>v1
7 v0->v1 a:x rho
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:

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

The following markers are supported for an edge:

- “rho” places a backward snake arrow to the edge,
- “bend: <angle>” bend it right by the amount of “<angle>,”
- “a: <txt>” attaches label “<txt>” to it,
- “pi” makes it dotted, with π label.

It is also possible to put transformation arrows to the graph, with the help of “v0=>v1” syntax. The arrow will be placed exactly between two vertices. You can also put an arrow from a vertex to the right, saying for example “v3=>”, of from the left to the vertex, by saying for example “=>v5.” If you want the arrow to stay further away from the vertex than usually, use a few “=” symbols, for example “==>v0.”

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

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

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

`\eolang` There is also a no-argument command `\eolang` to help you print the name of EO language. It understands anonymous mode of `\acmart` and prints itself differently, to `\xmir` double-blind your paper. There is also `\phic` command to print the name of φ -calculus, also sensitive to anonymous mode. The macro `\xmir` prints "XMIR".

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

```

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

```

`\phiConst` A few simple commands are defined to help you render arrows. It is recommended 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 arbitrary number of control-flow objects, while $x \vdash y$ makes it a special attribute.

```

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

```

`\phiMany` Sometimes you may need to simplify the way you describe an object:

The expression $\llbracket \alpha_1 \mapsto x_1, \alpha_2 \mapsto x_2, \dots, \alpha_n \mapsto x_n \rrbracket$ and expression $\llbracket \alpha_i \overset{n}{\rightsquigarrow} x_i \rrbracket$ 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.

```

2 Package Options

`tmpdir` The default location of temp files is `_eolang`. You can change this using `tmpdir` option:

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

`nodollar` You may disable special treatment of the dollar sign:

```
\usepackage[nodollar]{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:

$\frac{x \mapsto [\varphi \mapsto y] \quad y \mapsto [z \mapsto 42]}{x.z \mapsto 42} \text{R1}$	<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}`:

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

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

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

It's possible to use `\label` inside `phiqutation` environment:

<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} - 4ac. 10 \label{d} 11 \end{phiqutation} 12 Eq.\ref{d} is also widely used in 13 number theory and polynomial factoring. </pre>
--	---

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

$$\begin{aligned} x(\pi) &\mapsto [\lambda \mapsto f_1], \\ x(a,b,c) &\mapsto [\alpha_0 \mapsto \emptyset, \varphi \mapsto \text{hello}(\xi), x \mapsto \text{FALSE}], \\ \Delta &= 43-09. \end{aligned}$$

```

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

```

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

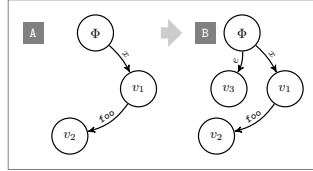
$$\begin{aligned} [b \mapsto \emptyset], \\ [\varphi \mapsto \text{TRUE}, \Delta \mapsto 42], \\ \Delta = 43-09. \end{aligned}$$

```

5 \begin{phi}
6 [[ b -> \emptyset ]], \\\
7 [[ @ -> \text{TRUE}, \Delta -> 42 ]], \\\
8 \Delta = |43-09|.
9 \end{phi}

```

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

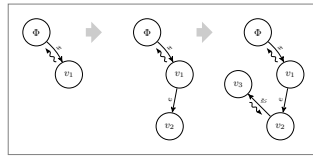


```

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

```

You can make a copy from a copy:

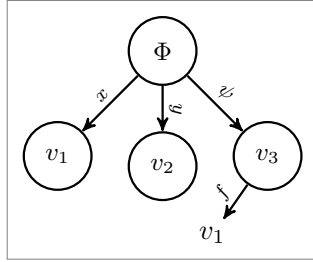


```

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

```

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

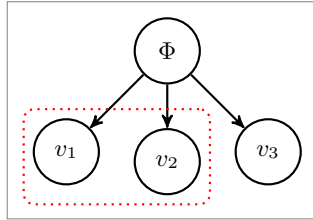


```

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

```

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



```

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

```

4 Implementation

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

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

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

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

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

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

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

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

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

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

Then, we process package options:

```

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

```

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

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

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

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

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

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

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

```
23 \makeatletter
24 \begin{VerbatimOut}{\eolang@tmpdir/eolang-phi.pl}
25 $env = $ARGV[0];
26 open(my $fh, '<', $ARGV[1]);
27 my $tex; { local $/; $tex = <$fh>; }
28 print '% This file is auto-generated', "\n";
29 print '% There are ', length($tex),
30 ' chars in the input: ', $ARGV[1], "\n";
31 print '% ---', "\n";
32 if (index($tex, "\t") > 0) {
33   print "TABS are prohibited!";
34   exit 1;
35 }
36 my @lines = split (/\\n/g, $tex);
37 foreach my $t (@lines) {
38   print '% ', $t, "\n";
39 }
40 print '% ---', "\n";
41 $tex =~ s/\\.*\\n/\\n/g;
42 $tex =~ s/\\s+|\\s+$//g;
43 my $gathered = (0 == $tex =~ /\\n\\s+/g);
44 if ($env ne 'phiq') {
45   $tex =~ s/\\s+\\n\\s*//g;
46   $tex =~ s/\\\\\\\\n/\\n\\n/g;
47   $tex =~ s/\\n*(\\label\\{[\\~]+\\})\\n*/\\1/g;
48 }
49 $tex =~ s/&/\\sigma{/g;
50 $tex =~ s/([\\~\\{a-z0-9]|\\^)Q(?:[a-z0-9])/\\1\\Phi{/g;
51 $tex =~ s/([\\~\\{a-z0-9]|\\^)D>/\\1\\Delta{}/g;
52 $tex =~ s/([\\~\\{a-z0-9]|\\^)L>/\\1\\lambda{}/g;
53 $tex =~ s/"([~]+)"/"\\1"/g;
54 $tex =~ s/([ ](>|/|\\^)([a-z0-9]{2,})(?=[ ](\\.))/\\1|2|/g;
55 $tex =~ s/([\\~_]|\\^)([0-9]+|\\*)/(\\[a-z]+|\\[a-z]+|\\)
56 (->|\\.\\.\\.>|>|:=|!->)/\\1\\alpha_{2}\\vert{}\\3\\space{}\\4/xg;
57 $tex =~ s/([\\~_]|\\^)([0-9]+|\\*)
58 (->|\\.\\.\\.>|>|:=|!->)/\\1\\alpha_{2}\\space{}\\3/xg;
59 if ($env ne 'phiq') {
60   $tex =~ s/\\begin\\{split\\}\\n/\\begin{split}&/g;
61   $tex =~ s/\\n\\s*\\end\\{split\\}/\\end{split}/g;
```



```

62 $tex =~ s/\n\n/\n\n/g;
63 $tex =~ s/\n\n\n\n[-4pt]/g;
64 $tex =~ s/([^\s])\s{2}([^\s])/1 2/g;
65 $tex =~ s/\s{2}/ \quad/g;
66 my @leads = $tex =~ /&[^\s]+\s(->|:=|)/g;
67 my @eols = $tex =~ /&/g;
68 $tex = '&' . $tex;
69 if (0+@leads == 0+@eols && 0+@eols > 0) {
70     $tex =~ s/&([^\s]+)\s/1&/g;
71     $gathered = 0;
72 }
73 }
74 $tex =~ s/\$/\xi{/g;
75 $tex =~ s/(?<!\{)\^{\rho}/g;
76 $tex =~ s/[\[\/\llbracket\mathrel{/g;
77 $tex =~ s/\]/\rrbracket\mathrel{/g;
78 $tex =~ s/([^\s,>()]{0-9A-F}{2}(?:-[0-9A-F]{2})+|
79 [0-9]+(?:\.[0-9]+)?)/1|2|/xg;
80 $tex =~ s/TRUE/|TRUE|/g;
81 $tex =~ s/FALSE/|FALSE|/g;
82 $tex =~ s/\?/\varnothing{/g;
83 $tex =~ s/@/\varphi{/g;
84 $tex =~ s/-([a-z]+)>/\mathrel{\phiSlot{1}}/g;
85 $tex =~ s/!->/\mathrel{\phiConst}/g;
86 $tex =~ s/->/\mathrel{\mapsto}/g;
87 $tex =~ s/~>/\mathrel{\phiWave}/g;
88 $tex =~ s/:=/\mathrel{\vDash}/g;
89 $tex =~ s/\.\. >/\mathrel{\phiDotted}/g;
90 $tex =~ s/\|{2,}/|/g;
91 $tex =~ s/|([^\|]+)\|/\textnormal{\texttt{1}}{/g;
92 if ($env eq 'phiq') {
93     print '$' if ($tex ne '');
94 } else {
95     print '\begin{', $env, '}';
96     if ($gathered) {
97         print '\begin{gathered}';
98     } else {
99         print '\begin{split}';
100     }
101 }
102 if ($gathered) {
103     $tex =~ s/(\n\n\n(?:\n\n)+)?&/1/g;
104     $tex =~ s/^&/g;
105 }
106 print $tex;
107 if ($env eq 'phiq') {
108     print '$' if ($tex ne '');
109 } else {
110     if ($gathered) {
111         print '\end{gathered}';
112     } else {
113         print '\end{split}';
114     }
115     print '\end{', $env, '}';

```

```

116 }
117 \print '\endinput';
118 \end{VerbatimOut}
119 \message{eolang: File with Perl script
120   '\eolang@tmpdir/eolang-phi.pl' saved~^J}%
121 \makeatother

```

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

```

122 \makeatletter\newcommand\eolang@process[1]{
123   \def\hash{\eolang@mdfive
124     {\eolang@tmpdir/\jobname/phiquation.tex}}%
125   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiquation.tex"
126     "\eolang@tmpdir/\jobname/\hash.tex"}%
127   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
128     perl "\eolang@tmpdir/eolang-phi.pl"
129     '#1'
130     "\eolang@tmpdir/\jobname/\hash.tex"
131     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi}%
132   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
133 }
134 \newenvironment{phiquation*}%
135 {\catcode'\|=12 \VerbatimEnvironment%
136 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
137 \begin{VerbatimOut}
138   {\eolang@tmpdir/\jobname/phiquation.tex}}
139 {\end{VerbatimOut}\eolang@process{equation*}}
140 \newenvironment{phiquation}%
141 {\catcode'\|=12 \VerbatimEnvironment%
142 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
143 \begin{VerbatimOut}
144   {\eolang@tmpdir/\jobname/phiquation.tex}}
145 {\end{VerbatimOut}\eolang@process{equation}}
146 \makeatother

```

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

```

147 \makeatletter\newcommand\phiq[1]{%
148   \iexec[trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{
149     /bin/echo '\detokenize{#1}'}%
150   \def\hash{\eolang@mdfive
151     {\eolang@tmpdir/\jobname/phiq.tex}}%
152   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiq.tex"
153     "\eolang@tmpdir/\jobname/\hash.tex"}%
154   \ifdefined\eolang@nodollar\else\catcode'\$=3 \fi%
155   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
156     perl \eolang@tmpdir/eolang-phi.pl '\phiq'
157     "\eolang@tmpdir/\jobname/\hash.tex"
158     \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi}%
159   \ifdefined\eolang@nodollar\else\catcode'\$=active\fi%
160 }\makeatother

```

nodollar Then, we redefine dollar sign:

```

161 \ifdefined\eolang@nodollar\else
162   \begingroup

```

```

163 \catcode'\$=\active
164 \protected\gdef$#1${\phiq{#1}}
165 \endgroup
166 \AtBeginDocument{\catcode'\$=\active}
167 \fi

```

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

```

168 \makeatletter
169 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
170 sub num {
171   my ($i) = @_;
172   $i =~ s/(\+|-)\./\10./g;
173   return $i;
174 }
175 sub fmt {
176   my ($tex) = @_;
177   $tex =~ s/\\|([^\|]+)\\|\\textnormal{\texttt{1}}/g;
178   return $tex;
179 }
180 sub vertex {
181   my ($v) = @_;
182   if (index($v, 'v0') == 0) {
183     return '\Phi';
184   } else {
185     $v =~ s/^v/v_/g;
186     $v =~ s/[0-9]$/g;
187     return $v;
188   }
189 }
190 sub tailor {
191   my ($t, $m) = @_;
192   $t =~ s/<([A-Z]?$m[A-Z]?):([>]+)>/\2/g;
193   $t =~ s/<[A-Z]+:[>]+>/g;
194   return $t;
195 }
196 open(my $fh, '<', $ARGV[0]);
197 my $tex; { local $/; $tex = <$fh>; }
198 if (index($tex, "\t") > 0) {
199   print "TABS are prohibited!";
200   exit 1;
201 }
202 print '% This file is auto-generated', "\n\n";
203 print '% --- there are ', length($tex),
204   ' chars in the input (', $ARGV[0], "):\n";
205 foreach my $t (split /\n/g, $tex) {
206   print '% ', $t, "\n";
207 }
208 print "% ---\n";
209 $tex =~ s/\\\\\\/\n/g;
210 $tex =~ s/\\\\\n//g;
211 $tex =~ s/(\\[a-zA-Z]+)\s+/\1/g;
212 $tex =~ s/\n{2,}/\n/g;
213 my @cmds = split /\n/g, $tex;

```

```

214 print '% --- before processing:' . "\n";
215 foreach my $t (split (/\\n/g, $tex)) {
216   print '% ', $t, "\n";
217 }
218 print '% ---';
219 print ' (' . (0+@cmds) . " lines)\n";
220 print '\begin{picture}', "\n";
221 for (my $c = 0; $c < 0+@cmds; $c++) {
222   my $cmd = $cmds[$c];
223   $cmd =~ s/^\\s+//g;
224   $cmd =~ s/\\%.*//g;
225   my ($head, $tail) = split(/ /, $cmd, 2);
226   my %opts = {};
227   foreach my $p (split(/ /, $tail)) {
228     my ($q, $t) = split(/:/, $p);
229     $opts{$q} = $t;
230   }
231   if (index($head, '->') >= 0) {
232     my $draw = '\draw[';
233     if (exists $opts{'pi'}) {
234       $draw = $draw . '<MB:phi-pi><F:draw=none>';
235       if (not exists $opts{'a'}) {
236         $opts{'a'} = '\pi';
237       }
238     }
239     if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
240       $draw = $draw . '<MB:,phi-rho>';
241     }
242     $draw = $draw . ']';
243     my ($from, $to) = split (/->/, $head);
244     $draw = $draw . " (${$from}) ";
245     if (exists $opts{'bend'}) {
246       $draw = $draw . 'edge [<F:draw=none><MF:,bend right=' .
247         num($opts{'bend'}) . '>';
248       if (exists $opts{'rho'}) {
249         $draw = $draw . '<MB:,phi-rho>';
250       }
251       $draw = $draw . ']';
252     } else {
253       $draw = $draw . '--';
254     }
255     if (exists $opts{'a'}) {
256       my $a = $opts{'a'};
257       if (index($a, '$') == -1) {
258         $a = '$' . fmt($a) . '$';
259       } else {
260         $a = fmt($a);
261       }
262       $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
263     }
264     if (exists $opts{'break'}) {
265       $draw = $draw . '<F: coordinate [pos=' .
266         ($opts{'break'} / 100) . '] (break)>';
267     }

```

```

268 $draw = $draw . " (<MF:${to}><B:break-v>);
269 if (exists $opts{'break'}) {
270     print tailor($draw, 'F') . ";\n";
271     print ' \node[outer sep=.1cm,inner sep=0cm] ' .
272         'at (break) (break-v) {$' . vertex($to) .
273         '$};' . "\n";
274     print ' ' . tailor($draw, 'B');
275 } else {
276     print tailor($draw, 'M');
277 }
278 } elsif (index($head, '=') >= 0) {
279     my ($from, $to) = split (/=>/, $head);
280     my $size = () = $head =~ /=/g;
281     if ($from eq '') {
282         print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
283             $to . '.center]';
284     } elsif ($to eq '') {
285         print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
286             $from . '.center]';
287     } else {
288         print '\node [phi-arrow] at ($(' .
289             $from . ')!0.5!(' . $to . ')$)';
290     }
291     print '{}';
292 } elsif (index($head, '!') >= 0) {
293     my ($v, $marker) = split (/!+/, $head);
294     my $size = () = $head =~ /*!/g;
295     print '\node [phi-marker, left=' .
296         ($size * 0.6) . 'cm of ' .
297         $v . '.center]{' . fmt($marker) . '}';
298 } elsif (index($head, '+') >= 0) {
299     my ($v, $suffix) = split (/+/, $head);
300     my @friends = ($v);
301     foreach my $c (@cmds) {
302         $e = $c;
303         $e =~ s/^\s+//g;
304         my $h = $e;
305         $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
306         foreach my $f (@friends) {
307             my $add = '';
308             if (index($h, $f . '->') >= 0) {
309                 $add = substr($h, index($h, '->') + 2);
310             }
311             if ($h =~ /\->\Q${f}\E/) {
312                 $add = substr($h, 0, index($h, '->'));
313             }
314             if (index($e, ' xy:' . $f . ',') >= 0) {
315                 $add = $h;
316             }
317             if (index($add, '+') == -1
318                 and $add ne ''
319                 and not(grep(/\Q${add}\E/, @friends))) {
320                 push(@friends, $add);
321             }

```

```

322     }
323 }
324 my @extra = ();
325 foreach my $e (@cmds) {
326     $m = $e;
327     if ($m =~ /^s*Q${v}\E\s/) {
328         next;
329     }
330     if ($m =~ /^s*[^\s]+\+/ and not($m =~ /^s*Q${head}\E\s/)) {
331         next;
332     }
333     foreach my $f (@friends) {
334         my $h = $f;
335         $h =~ s/[a-z]$//g;
336         if ($m =~ s/^(s*)Q${f}\E\+Q${suffix}\E\s?/\1${h}${suffix} /g) {
337             last;
338         }
339         $m =~ s/^(s*)Q${f}\E\s/\1${h}${suffix} /g;
340         $m =~ s/^(s*)Q${f}\E->/\1${h}${suffix}->/g;
341         $m =~ s/\sxy:Q${f}\E,/ xy:${h}${suffix},/g;
342         $m =~ s/->Q${f}\E\s/->${h}${suffix} /g;
343     }
344     if ($m ne $e) {
345         push(@extra, ' ' . $m);
346     }
347 }
348 splice(@extra, 0, 0, @extra[-1]);
349 splice(@extra, -1, 1);
350 splice(@extra, 0, 0, '% clone of ' . $v . ' (' . $head .
351     ') , friends: [' . join(', ', @friends) . ']' in ' .
352     (0+@cmds) . ' lines');
353 splice(@cmds, $c, 1, @extra);
354 print '% cloned ' . $v . ' at line no.' . $c .
355     ' (' . (0+@extra) . ' lines -> ' .
356     (0+@cmds) . ' lines total)';
357 } elsif ($head =~ /^v[0-9]+[a-z]?$/) {
358     print '\node[';
359     if (exists $opts{'xy'}) {
360         my ($v, $right, $down) = split(/,/, $opts{'xy'});
361         my $loc = '';
362         if ($down > 0) {
363             $loc = 'below ';
364         } elsif ($down < 0) {
365             $loc = 'above ';
366         }
367         if ($right > 0) {
368             $loc = $loc . 'right';
369         } elsif ($right < 0) {
370             $loc = $loc . 'left';
371         }
372         print ', ' . $loc . '=';
373         print abs(num($down)) . 'cm and ' .
374             abs(num($right)) . 'cm of ' . $v . '.center';
375     }

```

```

376   if (exists $opts{'data'}) {
377       print ',phi-data';
378       if (not $opts{'data'} eq '') {
379           my $d = $opts{'data'};
380           if (index($d, '|') == -1) {
381               $d = '$\Delta\phiDotted\text{' .
382                   '\textnormal{\texttt{' . fmt($d) . '}}}$';
383           } else {
384               $d = fmt($d);
385           }
386           $opts{'box'} = $d;
387       }
388   } elsif (exists $opts{'atom'}) {
389       print ',phi-atom';
390       if (not $opts{'atom'} eq '') {
391           my $a = $opts{'atom'};
392           if (index($a, '$') == -1) {
393               $a = '$\lambda\phiDotted{' . fmt($a) . '$';
394           } else {
395               $a = fmt($a);
396           }
397           $opts{'box'} = $a;
398       }
399   } else {
400       print ',phi-object';
401   }
402   print ']';
403   print '(', $head, ')';
404   print '{' . vertex($head) . '$}';
405   if (exists $opts{'box'}) {
406       print ' node[phi-box] at (';
407       print $head, '.south east) {';
408       print $opts{'box'}, ')';
409   }
410   } else {
411       print $cmd;
412   }
413   print ";\n";
414 }
415 print '\end{picture}%', "\n";
416 print "% --- after processing:\n%";
417 foreach my $c (@cmds) {
418     print '% ', $c, "\n";
419 }
420 print '% --- (' . (0+@cmds) . " lines)\n";
421 print '\endinput';
422 \end{VerbatimOut}
423 \message{eolang: File with Perl script
424     '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
425 \makeatother

```

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

```

426 \setcounter{FancyVerbLine}{0}

```

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

```

427 \RequirePackage{tikz}
428 \usetikzlibrary{arrows}
429 \usetikzlibrary{shapes}
430 \usetikzlibrary{snakes}
431 \usetikzlibrary{decorations}
432 \usetikzlibrary{decorations.pathmorphing}
433 \usetikzlibrary{decorations.pathreplacing}
434 \usetikzlibrary{positioning}
435 \usetikzlibrary{calc}
436 \usetikzlibrary{math}
437 \usetikzlibrary{arrows.meta}

```

picture Then, we define internal environment phicture:

```

438 \newenvironment{phicture}%
439 {\noindent\begin{tikzpicture}[
440   ->,>=stealth',node distance=0,thick,
441   pics/parallel arrow/.style={
442     code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
443 {\end{tikzpicture}}
444 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
445   minimum height=0.5cm, minimum width=0.5cm,
446   single arrow head extend=2mm]
447 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
448   minimum width=1.4em, font={\small\color{white}\ttfamily},
449   fill=gray]
450 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
451   draw,font={\small}]
452 \tikzstyle{phi-object} = [phi-thing,circle]
453 \tikzstyle{phi-data} = [phi-thing,regular polygon,
454   regular polygon sides=8]
455 \tikzstyle{phi-empty} = [phi-object]
456 \tikzset{%
457   phi-rho/.style={
458     postaction={%
459       decoration={
460         show path construction,
461         curveto code={
462           \tikzmath{
463             coordinate \I, \F, \v;
464             \I = (\tikzinputsegmentfirst);
465             \F = (\tikzinputsegmentlast);
466             \v = ($(\I) -(\F)$);
467             real \d, \a, \r, \t;
468             \d = 0.8;
469             \t = atan2(\vy, \vx);
470             if \vx<0 then { \a = 90; } else { \a = -90; };
471             {
472               \draw[arrows={-latex}, decorate,
473                 decoration={%
474                   snake, amplitude=.4mm,
475                   segment length=2mm,
476                   post length=1mm
477                 }

```



```

478         ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
479         -- ++(\t: 2*\d em);
480     };
481 }
482 },
483 lineto code={
484     \tikzmath{
485         coordinate \I, \F, \v;
486         \I = (\tikzinputsegmentfirst);
487         \F = (\tikzinputsegmentlast);
488         \v = ($(\I) -(\F)$);
489         real \d, \a, \r, \t;
490         \d = 0.8;
491         \t = atan2(\vy, \vx);
492         if \vx<0 then { \a = 90; } else { \a = -90; };
493         {
494             \draw[arrows={-latex}, decorate,
495                 decoration={%
496                     snake, amplitude=.4mm,
497                     segment length=2mm,
498                     post length=1mm}]
499                 ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
500                 -- ++(\t: 2*\d em);
501         };
502     }
503 }
504 },
505 decorate
506 }
507 }
508 }
509 \tikzstyle{phi-pi} = [draw,dotted]
510 \tikzstyle{phi-atom} = [phi-object,double]
511 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
512     rectangle,thin,minimum width=1.2em,anchor=north west,
513     font={\scriptsize}]
514 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
515     above=2pt,sloped/.append style={transform shape},
516     font={\scriptsize},color=black]

```

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

```

517 \makeatletter\newenvironment{sodg}%
518 {\catcode'\|=12 \VerbatimEnvironment%
519 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
520 \begin{VerbatimOut}
521   {\eolang@tmpdir/\jobname/sodg.tex}}
522 {\end{VerbatimOut}}%
523 \def\hash{\eolang@mdfive
524   {\eolang@tmpdir/\jobname/sodg.tex}}%
525 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
526   "\eolang@tmpdir/\jobname/\hash.tex"}%
527 \catcode'\$=3 %
528 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
529   perl "\eolang@tmpdir/eolang-sodg.pl"

```


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

```
565 \newcommand{\phiMany}[3]{%
566   \overunderset{\scriptscriptstyle #3}{\scriptscriptstyle #2}{#1}}
```

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

```
567 \RequirePackage{trimclip}
568 \RequirePackage{amsfonts}
569 \makeatletter
570 \newcommand{\phiDotted}{%
571   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
572 \newcommand{\phiDotted@}[2]{%
573   \begingroup%
574   \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
575   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
576   \sbox\z@{%
577     \makebox[\dimen\z@][s]{%
578       \clipbox{0 0 {0.4\width} 0}%
579       {\resizebox{\dimen\z@}{\height}%
580        {\m@th#1\dashrightarrow}}}%
581     \hss%
582     \clipbox{{0.69\width} {-0.1\height} 0
583      {-\height}}{\m@th#1\rightarrow}%
584   }%
585 }%
586 \ht\z@=\dimen\tw@ \dp\z@=\z@%
587 \box\z@%
588 \endgroup%
589 }
590 \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.	7	0.2.0	eolang-phi.pl: Numbers automatically render as \texttt. No need to use vertical bars around them anymore.	8
0.0.2	sodg: The environment phigure renamed to sodg for the sake of better semantic. The graph in the picture is solely a SODG graph, that's why the name sodg is better.	17		eolang-sodg.pl: The content of atom and data boxes is parsed automatically as formulas and numbers, respectively.	11
	eolang-phi.pl: New symbol added for basket slots	8		\xmirt: New command \xmirt prints XMIR in both normal and anonymous mode of acmart.	18
	Parsing of symbols “@,” “^,” and “&” enabled (\varphi, \rho, and \sigma)	8	0.3.0	\eolang@lineno: New counter for protecting lineno.	8
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets	8		eolang-phi.pl: New arrow added, that looks like \leadsto.	8
	eolang-sodg.pl: The Perl file now has a fixed name, which doesn't depend on the name of the TeX job. This file may be shared among jobs, no need to make it uniquely named.	11		\phiWave: New command \phiWave added to denote a link to a multi-layer attribute.	18
	\phiq: Parsing of additional symbols enabled.	10	0.4.0	eolang-sodg.pl: Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed with \Delta and \lambda commands.	11
0.1.0	General: Parsing of package options introduced.	7		Relative positioning of vertices fixed.	11
	\eolang: New command \eolang added to print the name of the language in both normal and anonymous mode of acmart.	18	0.5.0	eolang-phi.pl: Automated formatting of TRUE and FALSE added.	8
	\eolang@mdfive: New supplementary command added to calculate MD5 sum of a file.	8		eolang-sodg.pl: It is possible to use tikz commands inside sodg environment.	11
	eolang-phi.pl: A new Perl script “eolang-phi.pl” added for parsing of phi expressions.	8		New syntax introduced that allows to make clones of vertices and all their dependants.	11
	eolang-sodg.pl: There are two Perl scripts now: one for phiquation, another one for sodg.	11		Now edges may have break attribute, to make them shorter.	11
	\phic: New command \phic prints the name of φ -calculus in both normal and anonymous mode of acmart.	18		\phiMany: New command \phiMany enables iterating over an arrow.	19
	\phiConst: New command \phiConst added to denote a link to a constant attribute.	18		\phiSlot: New command \phiSlot added to denote a link to a slot in a basket.	18
	\phiDotted: New command \phiDotted added to denote a link to a special attribute.	19	0.6.0	General: Package option nocomments added in order to enable comments suppression in temporary .tex files (may be pretty important for .dtx documents).	7

0.7.0	<code>eolang-sodg.pl</code> : The attribute <code>rrho</code> is retired, now <code>rho</code> works just fine in all situations.	11	Φ , just using capital “Q” is enough. .	8
	<code>nodollar</code> : Now it is possible to use dollar sign instead of <code>\phi</code>	10	Object names are automatically converted to <code>\texttt</code> , provided their names include two or more symbols.	8
	<code>eolang-phi.pl</code> : New syntax sugar for		Text in quotes is automatically converted to <code>\texttt</code>	8

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>	74, 154, 159, 163, 166, 527, 532	
<code>\%</code>	131, 158, 531	
<code>\(</code>	78	
<code>*</code>	55, 57	
<code>\+</code>	172, 299, 330, 336	
<code>\.</code>	56, 58, 79, 89, 172	
<code>\/</code>	54, 55	
<code>\?</code>	82	
<code>\[</code>	76, 103	
<code>\{</code>	47, 60, 61, 75	
<code>\}</code>	47, 60, 61, 70	
<code>\]</code>	77, 103	
<code>\^</code>	75	
<code>\ </code>	55, 90, 91, 135, 141, 177, 518	
Numbers		
<code>\2</code> ...	54, 56, 58, 64, 79, 192	
<code>\3</code>	56, 58	
<code>\4</code>	56	
A		
<code>\a</code>	467, 470, 478, 489, 492, 499	
<code>\active</code> ..	159, 163, 166, 532	
<code>\alpha</code>	545	
<code>\anon</code>	536, 537, 544, 545, 552, 553	
<code>\AtBeginDocument</code> ..	166	
B		
<code>\Bbbk</code>	3	
<code>\begin</code> ..	24, 95, 97, 99, 137, 143, 169, 220, 439, 520	
<code>\box</code>	587	
C		
<code>\catcode</code>	135, 141, 154, 159, 163, 166, 518, 527, 532	
<code>\clipbox</code>	578, 582	
<code>\color</code>	448	
D		
<code>\d</code>	467, 468, 478, 479, 489, 490, 499, 500	
<code>\dashrightarrow</code> ...	580	
<code>\def</code>	123, 150, 523	
<code>\Delta</code>	381	
<code>\detokenize</code>	149	
<code>\dimen</code>	574, 575, 577, 579, 586	
<code>\dp</code>	586	
<code>\draw</code> ...	232, 442, 472, 494	
E		
<code>\E</code> ..	311, 319, 327, 330, 336, 339, 340, 341, 342	
<code>\end</code>	111, 113, 115, 118, 139, 145, 415, 422, 443, 522	
<code>\endinput</code>	117, 421	
<code>\eolang</code>	535	
<code>\eolang-phi.pl</code>	23	
<code>\eolang-sodg.pl</code> ...	168	
<code>\eolang@lineno</code>	18	
<code>\eolang@mdfive</code>	19, 123, 150, 523	
<code>\eolang@nocomments</code>	13, 131, 158, 531	
<code>\eolang@nodollar</code> ..	154, 159, 161	
<code>\eolang@process</code> ...	122, 139, 145	
<code>\eolang@tmpdir</code>	11, 17, 24, 120, 124, 125, 126, 127, 128, 130, 138, 144, 148, 151, 152, 153, 155, 156, 157, 169, 424, 521, 524, 525, 526, 528, 529, 530	
F		
<code>\F</code>	463, 465, 466, 478, 485, 487, 488, 499	
<code>\FancyVerbLine</code>	426	
G		
<code>\gdef</code>	164	
H		
<code>\hash</code> ...	123, 126, 127, 130, 150, 153, 155, 157, 523, 526, 528, 530	
<code>\height</code>	579, 582, 583	
<code>\hspace</code>	559, 560	
<code>\hss</code>	581	
<code>\ht</code>	586	
I		
<code>\I</code>	463, 464, 466, 478, 485, 486, 488, 499	
<code>\iexec</code> ...	17, 125, 127, 148, 152, 155, 525, 528	
<code>\ifdefined</code>	131, 154, 158, 159, 161, 531, 536, 544, 552	
<code>\ifluatex</code>	12	
<code>\ifxetex</code>	12	
J		
<code>\jobname</code> ..	17, 124, 125, 126, 127, 130, 138, 144, 148, 151, 152, 153, 155, 157, 521, 524, 525, 526, 528, 530	
L		
<code>\lambda</code>	393	
<code>\leadsto</code>	562	
M		
<code>\m@th</code> ...	574, 575, 580, 583	
<code>\makeatletter</code>	18, 20, 23, 122, 147, 168, 517, 569	
<code>\makeatother</code> ..	18, 22, 121, 146, 160, 425, 534, 590	
<code>\makebox</code>	577	
<code>\mapsto</code>	560	
<code>\mapstochar</code> ..	560, 562, 571	
<code>\mathpalette</code>	571	
<code>\mathrel</code>	559, 560, 562, 571	
<code>\message</code>	119, 423	
<code>\mspace</code>	562	
N		
<code>\newcommand</code> ..	21, 122, 147, 535, 543, 551, 558, 561, 563, 565, 570, 572	
<code>\newcounter</code>	18	
<code>\newenvironment</code> ...	134, 140, 438, 517	
<code>\node</code>	271, 282, 285, 288, 295, 358	

<code>\nodollar</code>	161	<code>\resizebox</code>	579	<code>\tikzstyle</code>	444 ,	
<code>\noindent</code>	439	<code>\rightarrow</code> .	574 , 575 , 583		447 , 450 , 452 , 453 ,	
					455 , 509 , 510 , 511 , 514	
O		S		<code>\ttfamily</code>		448
<code>\overunderset</code>	566	<code>\sbox</code>	576	<code>\tw@</code>	575 , 586	
P		<code>\scriptscriptstyle</code>	566			
<code>\pdf@filemdfivesum</code> .	21	<code>\scriptsize</code>	513 , 516			
<code>\pgfkeys</code>	9	<code>\scshape</code>	564			
<code>\Phi</code>	183	<code>\setcounter</code>	132 ,	U		
<code>\phic</code>	542		136 , 142 , 426 , 519 , 533	<code>\usetikzlibrary</code>	428 ,	
<code>\phiConst</code>	558	<code>\settoheight</code>	575		429 , 430 , 431 , 432 ,	
<code>\picture</code>	438	<code>\settowidth</code>	574		433 , 434 , 435 , 436 , 437	
<code>\phiDotted</code> .	381 , 393 , 567	<code>\sffamily</code> ..	537 , 539 , 564	V		
<code>\phiDotted@</code>	571 , 572	<code>\small</code>	448 , 451	<code>\v</code>	463 , 466 , 485 , 488	
<code>\phiMany</code>	565	<code>\sodg</code>	517	<code>\value</code>	132 , 136 , 142 , 519 , 533	
<code>\phiq</code>	147 , 164	<code>\sxy</code>	341	<code>\varphi</code>	546 , 548	
<code>\phiquation</code>	122	T		<code>\VerbatimEnvironment</code>		
<code>\phiSlot</code>	563	<code>\t</code>	32 , 198 , 467 , 469 , 478 ,		135 , 141 , 518	
<code>\phiWave</code>	561		479 , 489 , 491 , 499 , 500	<code>\vx</code>	469 , 470 , 491 , 492	
<code>\pi</code>	236	<code>\texorpdfstring</code> ...		<code>\vy</code>	469 , 491	
<code>\ProcessPgfoptions</code> .	16		545 , 546 , 548	W		
<code>\protected</code>	164	<code>\text</code>	381 , 564	<code>\width</code>	578 , 582	
Q		<code>\textnormal</code>	382	X		
<code>\Q</code> ..	311 , 319 , 327 , 330 ,	<code>\texttt</code>	382	<code>\xmir</code>	551	
	336 , 339 , 340 , 341 , 342	<code>\tikz</code>	427	<code>\xrightarrow</code>	564	
R		<code>\tikzinputsegmentfirst</code>	464 , 486	Z		
<code>\relax</code>	3 , 571	<code>\tikzinputsegmentlast</code>	465 , 487	<code>\z@</code>	574 , 576 , 577 , 579 , 586 , 587	
<code>\RequirePackage</code> ..	1 ,		462 , 484			
	2 , 3 , 4 , 5 , 6 , 7 , 8 ,	<code>\tikzmath</code>	456			
	19 , 427 , 542 , 567 , 568	<code>\tikzset</code>				