



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

Yegor Bugayenko
yegor256@gmail.com

2022-11-16, 0.6.0

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

1 Introduction

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

$$\begin{aligned} a &\mapsto \llbracket \\ &\quad \rho \mapsto \xi.b.\rho^2, \alpha_0 | t \mapsto \text{TRUE}, \\ &\quad b \mapsto \llbracket \alpha_* \mapsto \text{fn}(56), \\ &\quad \quad \varphi \mapsto \text{hello}(\xi), \\ &\quad \quad \Delta \mapsto 01\text{-FE-C3} \rrbracket \rrbracket, \\ x &\mapsto \llbracket \alpha_0 \mapsto \emptyset \rrbracket. \end{aligned}$$

```

1 \documentclass{article}
2 \pagestyle{empty}
3 \usepackage{eolang}
4 \begin{document}
5 \begin{phiquestion*}
6 a -> [[ % it's abstract!
7   ^ !-> $.b.^{^2}, 0/t-> TRUE,
8   b -> [[ *-> |fn|(56),
9     @ -> |hello|($),
10    \Delta ..> 01-FE-C3 ]]]],\\
11 x -> [[ \alpha_0 -> ? ]].
12 \end{phiquestion*}
13 \end{document}

```

`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`),
- “->” maps to “ \mapsto ” (`\mapsto`),
- “~>” maps to “ \rightsquigarrow ” (`\phiWave`),
- “!->” maps to “ \vDash ” (`\phiConst`),
- “. .>” maps to “ \vdash ” (`\phiDotted`),
- “[[” maps to “ \llbracket ” (`\llbracket`),
- “]]” maps to “ \rrbracket ” (`\rrbracket`),
- “|abc|” maps to “ \texttt{abc} ” (`\texttt{abc}`).

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

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

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

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

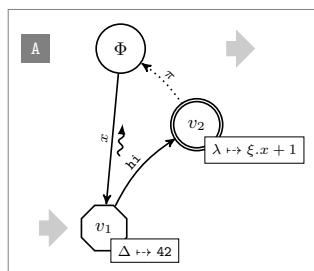
Numbers are automatically converted to fixed-width font, no need to always decorate them with vertical bars.

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

`\phiq` The command `\phiq` lets you inline a φ -calculus expressions using the same simple plain-text notation:

<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 \phiq{x -> [[@ -> y]]} \\\ 7 is a decorator of 8 the data object \\\ 9 \phiq{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 $\llbracket \alpha_i \xrightarrow{\alpha_i} x_i \rrbracket$ 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}
```

3 More Examples

The `phi`uation 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:

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

```

6 \begin{phiuation*}
7 \dfrac \{
8 {x->[[@->y]] \quad y->[[z->42]]} \{
9 {x.z -> 42} \} \{
10 \text{\textsf{family R1}}
11 \end{phiuation*}

```

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

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

```

6 \begin{phiuation*}
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)]]
14 \end{split}}{\text{R2}.}
15 \end{phiuation*}

```

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

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

```

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

```

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

$$\begin{array}{l} \text{Discriminant can be calculated using} \\ \text{the following simple formula:} \\ D = b^2 - 4ac. \quad (1) \\ \text{Eq. 1 is also widely used in number} \\ \text{theory and polynomial factoring.} \end{array}$$

```

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

```

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

```


$$x(\pi) \mapsto [\lambda \mapsto f_1],$$


$$x(a,b,c) \mapsto [\alpha_0 \mapsto \emptyset, \varphi \mapsto \text{hello}(\xi), x \mapsto \text{FALSE}],$$


$$\Delta = 43-09.$$


```

```

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

```

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

```


$$[b \mapsto \emptyset],$$


$$[\varphi \mapsto \text{TRUE}, \Delta \mapsto 42],$$


$$\Delta = 43-09.$$

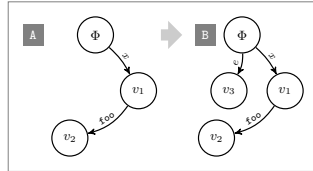

```

```

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

```

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

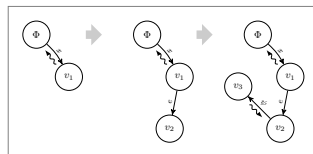


```

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:

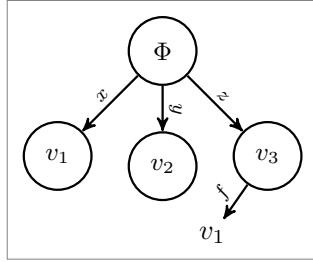


```

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:

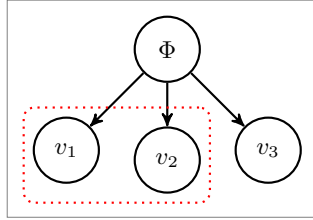


```

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:z
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 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/([^_]|^)([0-9]+|\\*)(/([a-z]+|\\|[a-z]+\\|)
51 (->|\\.\\.\\.>|~>|:=|!->)/\\1\\alpha_{2}\\vert{}\\3\\space{}\\4/xg;
52 $tex =~ s/([^_]|^)([0-9]+|\\*)(
53 (->|\\.\\.\\.>|~>|:=|!->)/\\1\\alpha_{2}\\space{}\\3/xg;
54 if ($env ne 'phiq') {
55   $tex =~ s/\\begin\\{split\\}\\n/\\begin{split}&/g;
56   $tex =~ s/\\n\\s*\\end\\{split\\}/\\end{split}/g;
57   $tex =~ s/\\n\\n/\\\\&/g;
58   $tex =~ s/\\n/\\\\[-4pt]&/g;
59   $tex =~ s/([^&\\s])\\s{2}([\\s])/\\1 \\2/g;
60   $tex =~ s/\\s{2}/ \\quad{/g;
61   my @leads = $tex =~ s/&[^\\s]+\\s(->|:=|!>)/g;
```



```

116 \iexec[trace,null]{perl -pi -e 's/(\\\\[a-zA-Z])\\s+/\1/g'
117   "\eolang@tmpdir/eolang-phi.pl"}
118 \makeatother

```

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

```

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

```

\phiq Then, we define \phiq command:

```

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

```

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

```

156 \makeatletter
157 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
158 sub num {
159   my ($i) = @_ ;
160   $i =~ s/(\\+|-)\\. /10./g;
161   return $i;

```

```

162 }
163 sub fmt {
164   my ($tex) = @_;
165   $tex =~ s/\\|([^\|]+)\\|\\textnormal{\\texttt{\\1}}/g;
166   return $tex;
167 }
168 sub vertex {
169   my ($v) = @_;
170   if (index($v, 'v0') == 0) {
171     return '\Phi';
172   } else {
173     $v =~ s/^v/v_/g;
174     $v =~ s/[^0-9]$/g;
175     return $v;
176   }
177 }
178 sub tailor {
179   my ($t, $m) = @_;
180   $t =~ s/<([A-Z]?${m}[A-Z]?):([>]+)>/\2/g;
181   $t =~ s/<[A-Z]+:[>]+>/g;
182   return $t;
183 }
184 open(my $fh, '<', $ARGV[0]);
185 my $tex; { local $/; $tex = <$fh>; }
186 if (index($tex, "\t") > 0) {
187   print "TABS are prohibited!";
188   exit 1;
189 }
190 print '% This file is auto-generated', "\n%\n";
191 print '% --- there are ', length($tex),
192   ' chars in the input (', $ARGV[0], "):\n";
193 foreach my $t (split /\n/g, $tex) {
194   print '% ', $t, "\n";
195 }
196 print "% ---\n";
197 $tex =~ s/\\\\\\/\n/g;
198 $tex =~ s/\\\\\n//g;
199 $tex =~ s/(\\[a-zA-Z]+) +/\1/g;
200 $tex =~ s/\n{2,}/\n/g;
201 my @cmds = split /\n/g, $tex;
202 print '% --- before processing:' . "\n";
203 foreach my $t (split /\n/g, $tex) {
204   print '% ', $t, "\n";
205 }
206 print '% ---';
207 print ' (' . (0+@cmds) . " lines)\n";
208 print '\begin{picture}', "\n";
209 for (my $c = 0; $c < 0+@cmds; $c++) {
210   my $cmd = $cmds[$c];
211   $cmd =~ s/^s+//g;
212   $cmd =~ s/%.*/g;
213   my ($head, $tail) = split(/ /, $cmd, 2);
214   my %opts = {};
215   foreach my $p (split(/ /, $tail)) {

```

```

216   my ($q, $t) = split(/:/, $p);
217   $opts{$q} = $t;
218 }
219 if (index($head, '->') >= 0) {
220   my $draw = '\draw[';
221   if (exists $opts{'pi'}) {
222     $draw = $draw . '<MB:phi-pi><F:draw=none>';
223     if (not exists $opts{'a'}) {
224       $opts{'a'} = '\pi';
225     }
226   }
227   if (exists $opts{'rho'} and not(exists $opts{'bend'})) {
228     $draw = $draw . '<MB:.,phi-rho>';
229   }
230   $draw = $draw . ']';
231   my ($from, $to) = split (/->/, $head);
232   $draw = $draw . " (${$from}) ";
233   if (exists $opts{'bend'}) {
234     $draw = $draw . 'edge [<F:draw=none><MF:.,bend right=' .
235       num($opts{'bend'}) . '>';
236     if (exists $opts{'rho'}) {
237       $draw = $draw . '<MB:.,phi-rho>';
238     }
239     $draw = $draw . ']';
240   } else {
241     $draw = $draw . '--';
242   }
243   if (exists $opts{'a'}) {
244     my $a = $opts{'a'};
245     if (index($a, '$') == -1) {
246       $a = '$' . fmt($a) . '$';
247     } else {
248       $a = fmt($a);
249     }
250     $draw = $draw . '<MB: node [phi-attr] {' . $a . '}>';
251   }
252   if (exists $opts{'break'}) {
253     $draw = $draw . '<F: coordinate [pos=' .
254       ($opts{'break'} / 100) . '] (break)>';
255   }
256   $draw = $draw . " (<MF:${to}><B:break-v>)" ;
257   if (exists $opts{'break'}) {
258     print tailor($draw, 'F') . ";\n";
259     print ' \node[outer sep=.1cm,inner sep=0cm] ' .
260       'at (break) (break-v) {$' . vertex($to) .
261       '$};' . "\n";
262     print ' ' . tailor($draw, 'B');
263   } else {
264     print tailor($draw, 'M');
265   }
266 } elsif (index($head, '=>') >= 0) {
267   my ($from, $to) = split (/=>/, $head);
268   my $size = () = $head =~ /=/g;
269   if ($from eq '') {

```

```

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

```

```

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

```

```

378     if (not $opts{'atom'} eq '') {
379         my $a = $opts{'atom'};
380         if (index($a, '$') == -1) {
381             $a = '$\lambda\phiDotted{}' . fmt($a) . '$';
382         } else {
383             $a = fmt($a);
384         }
385         $opts{'box'} = $a;
386     }
387 } else {
388     print ',phi-object';
389 }
390 print ']' ;
391 print ' (' , $head, ')';
392 print ' {'$' . vertex($head) . '$}';
393 if (exists $opts{'box'}) {
394     print ' node[phi-box] at (';
395     print $head, '.south east) {';
396     print $opts{'box'}, ')';
397 }
398 } else {
399     print $cmd;
400 }
401 print ";\n";
402 }
403 print '\end{picture}%', "\n";
404 print "% --- after processing:\n%";
405 foreach my $c (@cmds) {
406     print '% ', $c, "\n";
407 }
408 print '% --- (' . (0+@cmds) . " lines)\n";
409 print '\endinput';
410 \end{VerbatimOut}
411 \message{eolang: File with Perl script
412 '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
413 \iexec[trace,null]{perl -pi -e 's/(\\[a-zA-Z])\\s+\\/1/g'
414 "\eolang@tmpdir/eolang-sodg.pl"}
415 \makeatother

```

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

```

416 \setcounter{FancyVerbLine}{0}

```

tikz Then, we include tikz package and its libraries:

```

417 \RequirePackage{tikz}
418 \usetikzlibrary{arrows}
419 \usetikzlibrary{shapes}
420 \usetikzlibrary{snakes}
421 \usetikzlibrary{decorations}
422 \usetikzlibrary{decorations.pathmorphing}
423 \usetikzlibrary{decorations.pathreplacing}
424 \usetikzlibrary{positioning}
425 \usetikzlibrary{calc}
426 \usetikzlibrary{math}

```

```

427 \usetikzlibrary{arrows.meta}

picture Then, we define internal environment phicture:
428 \newenvironment{phicture}%
429   {\noindent\begin{tikzpicture}[
430     ->,>=stealth',node distance=0,thick,
431     pics/parallel arrow/.style={
432       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
433   {\end{tikzpicture}}
434 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
435   minimum height=0.5cm, minimum width=0.5cm,
436   single arrow head extend=2mm]
437 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
438   minimum width=1.4em, font={\small\color{white}\ttfamily},
439   fill=gray]
440 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
441   draw,font={\small}]
442 \tikzstyle{phi-object} = [phi-thing,circle]
443 \tikzstyle{phi-data} = [phi-thing,regular polygon,
444   regular polygon sides=8]
445 \tikzstyle{phi-empty} = [phi-object]
446 \tikzset{%
447   phi-rho/.style={
448     postaction={%
449       decoration={
450         show path construction,
451         curveto code={
452           \tikzmath{
453             coordinate \I, \F, \v;
454             \I = (\tikzinputsegmentfirst);
455             \F = (\tikzinputsegmentlast);
456             \v = ($(\I) -(\F)$);
457             real \d, \a, \r, \t;
458             \d = 0.8;
459             \t = atan2(\vy, \vx);
460             if \vx<0 then { \a = 90; } else { \a = -90; };
461             {
462               \draw[arrows={-latex}, decorate,
463                 decoration={%
464                   snake, amplitude=.4mm,
465                   segment length=2mm,
466                   post length=1mm
467                 }
468               ]
469               ($(\F)!.5!(\I) +(\t: -\d em) +(\t +\a: 1ex)$)
470               -- ++(\t: 2*\d em);
471             }
472           },
473         },
474         lineto code={
475           \tikzmath{
476             coordinate \I, \F, \v;
477             \I = (\tikzinputsegmentfirst);
478             \F = (\tikzinputsegmentlast);
479             \v = ($(\I) -(\F)$);

```



```

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

```

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

```

507 \makeatletter\newenvironment{sodg}%
508 {\catcode'\|=12 \VerbatimEnvironment%
509 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
510 \begin{VerbatimOut}
511   {\eolang@tmpdir/\jobname/sodg.tex}}
512 {\end{VerbatimOut}}%
513 \def\hash{\eolang@mdfive
514   {\eolang@tmpdir/\jobname/sodg.tex}}%
515 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
516   "\eolang@tmpdir/\jobname/\hash.tex"}%
517 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
518   perl "\eolang@tmpdir/eolang-sodg.pl"
519   "\eolang@tmpdir/\jobname/\hash.tex"
520   \ifdefined\eolang@nocomments | perl -pe 's/\%.*(\n|$)//g'\fi}%
521 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
522 }\makeatother

```

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

```

523 \newcommand\eolang{%
524   \ifdefined\anon%
525     \anon[XYZ]{\sffamily EO}%
526   \else%
527     {\sffamily EO}%

```

```

528 \fi%
529 }

```

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

```

530 \RequirePackage{hyperref}
531 \newcommand\phic{%
532   \ifdefined\anon%
533     \anon[\texorpdfstring{$\alpha$}{a}-calculus]
534     {\texorpdfstring{$\varphi$}{phi}-calculus}%
535   \else%
536     \texorpdfstring{$\varphi$}{phi}-calculus%
537   \fi%
538 }

```

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

```

539 \newcommand\xmirl{%
540   \ifdefined\anon%
541     \anon[XML$^+${XMIR}]%
542   \else%
543     XMIR%
544   \fi%
545 }

```

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

```

546 \newcommand\phiConst{%
547   \mathrel{\hspace{.15em}}%
548   \mapstochar\mathrel{\hspace{-.15em}}\mapsto}

```

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

```

549 \newcommand\phiWave{%
550   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto}

```

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

```

551 \newcommand\phiSlot[1]{%
552   \xrightarrow{\text{\sffamily\scshape #1}}}

```

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

```

553 \newcommand\phiMany[3]{%
554   \overunderset{\scriptscriptstyle #3}{\scriptscriptstyle #2}{#1}}

```

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

```

555 \RequirePackage{trimclip}
556 \RequirePackage{amsfonts}
557 \makeatletter
558 \newcommand{\phiDotted}{%
559   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
560 \newcommand{\phiDotted@}[2]{%
561   \begingroup
562   \settowidth{\dimen\z@}{\m@th#1\rightarrow$}%

```

```

563 \settoheight{\dimen\tw@}{\m@th#1\rightarrow$}%
564 \sbox\z@{%
565   \makebox[\dimen\z@][s]{%
566     \clipbox{0 0 {0.4\width} 0}%
567     {\resizebox{\dimen\z@}{\height}%
568       {\m@th#1\dashrightarrow$}}%
569     \hss%
570     \clipbox{{0.69\width} {-0.1\height} 0
571       {-\height}}{\m@th#1\rightarrow$}}%
572   }%
573 }%
574 \ht\z@=\dimen\tw@ \dp\z@=\z@%
575 \box\z@%
576 \endgroup}\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	<code>eolang-phi.pl</code> : Numbers automatically render as <code>\texttt</code> . No need to use vertical bars around them anymore.	8
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.	17		<code>eolang-sodg.pl</code> : The content of <code>atom</code> and <code>data</code> boxes is parsed automatically as formulas and numbers, respectively.	10
	<code>eolang-phi.pl</code> : New symbol added for basket slots	8		<code>\xmirt</code> : New command <code>\xmirt</code> prints XMIR in both normal and anonymous mode of <code>acmart</code>	18
	Parsing of symbols “@”, “^”, and “&” enabled (<code>\varphi</code> , <code>\rho</code> , and <code>\sigma</code>)	8	0.3.0	<code>\eolang@lineno</code> : New counter for protecting <code>lineno</code>	8
	The symbols “[” and “]” replaced with “[[” and “]]” for abstract object brackets, because they conflicted with normal square brackets	8		<code>eolang-phi.pl</code> : New arrow added, that looks like <code>\leadsto</code>	8
	<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.	10		<code>\phiWave</code> : New command <code>\phiWave</code> added to denote a link to a multi-layer attribute.	18
	<code>\phiq</code> : Parsing of additional symbols enabled.	10	0.4.0	<code>eolang-sodg.pl</code> : Labels on the edges are automatically printed as math formulas. Also, boxes are prefixed with <code>\Delta</code> and <code>\lambda</code> commands.	10
0.1.0	General: Parsing of package options introduced.	7		Relative positioning of vertices fixed.	10
	<code>\eolang</code> : New command <code>\eolang</code> added to print the name of the language in both normal and anonymous mode of <code>acmart</code>	17	0.5.0	<code>eolang-phi.pl</code> : Automated formatting of <code>TRUE</code> and <code>FALSE</code> added.	8
	<code>\eolang@mdfive</code> : New supplementary command added to calculate MD5 sum of a file.	8		<code>eolang-sodg.pl</code> : It is possible to use <code>tikz</code> commands inside <code>sodg</code> environment.	10
	<code>eolang-phi.pl</code> : A new Perl script “ <code>eolang-phi.pl</code> ” added for parsing of <code>phi</code> expressions.	8		New syntax introduced that allows to make clones of vertices and all their dependants.	10
	<code>eolang-sodg.pl</code> : There are two Perl scripts now: one for <code>phi</code> uation, another one for <code>sodg</code>	10		Now edges may have <code>break</code> attribute, to make them shorter.	10
	<code>\phic</code> : New command <code>\phic</code> prints the name of φ -calculus in both normal and anonymous mode of <code>acmart</code>	18		<code>\phiMany</code> : New command <code>\phiMany</code> enables iterating over an arrow.	18
	<code>\phiConst</code> : New command <code>\phiConst</code> added to denote a link to a constant attribute.	18		<code>\phiSlot</code> : New command <code>\phiSlot</code> added to denote a link to a slot in a basket.	18
	<code>\phiDotted</code> : New command <code>\phiDotted</code> added to denote a link to a special attribute.	18	0.6.0	General: Package option <code>nocomments</code> added in order to enable comments suppression in temporary <code>.tex</code> files (may be pretty important for <code>.dtx</code> documents).	7

eolang-sodg.pl: The attribute rrho is retired, now rho works just fine	in all situations.	10
---	----------------------------	----

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	E	
$\backslash \$$ 69	$\backslash E$... 299, 307, 315, 318, 324, 327, 328, 329, 330	$\backslash ifluatex$ 12
$\backslash \%$ 128, 154, 520	$\backslash end$ 106,	$\backslash ifxetex$ 12
$\backslash ($ 73	108, 110, 113, 136,	J
$\backslash *$ 50, 52	142, 403, 410, 433, 512	$\backslash jobname$. 17, 121, 122,
$\backslash +$ 160, 287, 318, 324	$\backslash endinput$ 112, 409	123, 124, 127, 135,
$\backslash .$ 51, 53, 74, 84, 160	$\backslash eolang$ <u>523</u>	141, 145, 148, 149,
$\backslash /$ 50	$\backslash eolang\text{-}\phi i.pl$ 23	150, 151, 153, 511,
$\backslash ?$ 77	$\backslash eolang\text{-}sodg.pl$... 156	514, 515, 516, 517, 519
$\backslash [$ 71, 98	$\backslash eolang@lineno$ <u>18</u>	L
$\backslash \{$ 47, 55, 56, 70	$\backslash eolang@mdfive$	$\backslash lambda$ 381
$\backslash \}$ 47, 55, 56, 65	... <u>19</u> , 120, 147, 513	$\backslash leadsto$ 550
$\backslash]$ 72, 98	$\backslash eolang@nocomments$.	
$\backslash ^$ 70	... 13, 128, 154, 520	M
$\backslash $ 50, 85, 86, 132, 138, 165, 508	$\backslash eolang@process$...	$\backslash m@th$... 562, 563, 568, 571
	... 119, 136, 142	$\backslash makeatletter$ 18, 20, 23,
Numbers	$\backslash eolang@tmpdir$	119, 144, 156, 507, 557
$\backslash 2$ 51, 53, 59, 74, 180	11, 17, 24, 115, 117,	$\backslash makeatother$ 18, 22, 118,
$\backslash 3$ 51, 53	121, 122, 123, 124,	143, 155, 415, 522, 576
$\backslash 4$ 51	125, 127, 135, 141,	$\backslash makebox$ 565
	145, 148, 149, 150,	$\backslash mapsto$ 548
A	151, 152, 153, 157,	$\backslash mapstochar$. 548, 550, 559
$\backslash a$ 457, 460, 468, 479, 482, 489	412, 414, 511, 514,	$\backslash mathpalette$ 559
$\backslash alpha$ 533	515, 516, 517, 518, 519	$\backslash mathrel$ 547, 548, 550, 559
$\backslash anon$ 524,		$\backslash message$ 114, 411
525, 532, 533, 540, 541		$\backslash mspace$ 550
B	F	
$\backslash Bbbk$ 3	$\backslash F$ 453, 455, 456,	N
$\backslash begin$ 24, 90, 92, 94, 134,	468, 475, 477, 478, 489	$\backslash newcommand$ 21, 119, 144,
140, 157, 208, 429, 510	$\backslash FancyVerbLine$ <u>416</u>	523, 531, 539, 546,
$\backslash box$ 575	H	549, 551, 553, 558, 560
C	$\backslash hash$... 120, 123, 124,	$\backslash newcounter$ 18
$\backslash catcode$... 132, 138, 508	127, 147, 150, 151,	$\backslash newenvironment$...
$\backslash clipbox$ 566, 570	153, 513, 516, 517, 519	... 131, 137, 428, 507
$\backslash color$ 438	$\backslash height$ 567, 570, 571	$\backslash node$ 259,
D	$\backslash hspace$ 547, 548	270, 273, 276, 283, 346
$\backslash d$ 457, 458, 468,	$\backslash hss$ 569	$\backslash noindent$ 429
469, 479, 480, 489, 490	$\backslash ht$ 574	O
$\backslash dashrightarrow$... 568	I	$\backslash overunderset$ 554
$\backslash def$ 120, 147, 513	$\backslash I$ 453, 454, 456,	P
$\backslash Delta$ 369	468, 475, 476, 478, 489	$\backslash pdf@filemdfivesum$.. 21
$\backslash detokenize$ 146	$\backslash iexec$ 17,	$\backslash pgfkeys$ 9
$\backslash dimen$ 562, 563, 565, 567, 574	116, 122, 124, 145,	$\backslash Phi$ 171
$\backslash dp$ 574	149, 151, 413, 515, 517	$\backslash phic$ <u>530</u>
$\backslash draw$... 220, 432, 462, 484	$\backslash ifdefined$ 128,	$\backslash phiConst$ <u>546</u>
	154, 520, 524, 532, 540	

<code>\picture</code>	428	<code>\scshape</code>	552	445, 499, 500, 501, 504
<code>\phiDotted</code>	369, 381, 555	<code>\setcounter</code>	129,	<code>\ttfamily</code> 438
<code>\phiDotted@</code>	559, 560		133, 139, 416, 509, 521	<code>\tw@</code> 563, 574
<code>\phiMany</code>	553	<code>\settoheight</code>	563	
<code>\phiq</code>	144	<code>\settowidth</code>	562	U
<code>\phiquation</code>	119	<code>\sffamily</code>	525, 527, 552	<code>\usetikzlibrary</code> 418,
<code>\phiSlot</code>	551	<code>\small</code>	438, 441	419, 420, 421, 422,
<code>\phiWave</code>	549	<code>\sodg</code>	507	423, 424, 425, 426, 427
<code>\pi</code>	224	<code>\sxy</code>	329	
<code>\ProcessPgfOptions</code> . .	16			V
		T		<code>\v</code> 453, 456, 475, 478
Q		<code>\t</code>	32, 186, 457, 459, 468,	<code>\value</code> 129, 133, 139, 509, 521
<code>\Q</code>	299, 307, 315, 318,		469, 479, 481, 489, 490	<code>\varphi</code> 534, 536
	324, 327, 328, 329, 330	<code>\texorpdfstring</code>		<code>\VerbatimEnvironment</code>
			533, 534, 536 132, 138, 508
R		<code>\text</code>	369, 552	<code>\vx</code> 459, 460, 481, 482
<code>\relax</code>	3, 559	<code>\textnormal</code>	370	<code>\vy</code> 459, 481
<code>\RequirePackage</code> . . .	1,	<code>\texttt</code>	370	W
	2, 3, 4, 5, 6, 7, 8,	<code>\tikz</code>	417	<code>\width</code> 566, 570
	19, 417, 530, 555, 556	<code>\tikzinputsegmentfirst</code>		
<code>\resizebox</code>	567		454, 476	
<code>\rightarrow</code>	562, 563, 571	<code>\tikzinputsegmentlast</code>		X
			455, 477	<code>\xmir</code> 539
S		<code>\tikzmath</code>	452, 474	<code>\xrightarrow</code> 552
<code>\sbox</code>	564	<code>\tikzset</code>	446	
<code>\scriptscriptstyle</code> . .	554	<code>\tikzstyle</code>	434,	Z
<code>\scriptsize</code>	503, 506		437, 440, 442, 443,	<code>\z@</code> 562, 564, 565, 567, 574, 575