



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

Yegor Bugayenko
yegor256@gmail.com

2022-11-13, 0.5.0

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

1 Introduction

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

$ \begin{aligned} a &\mapsto \llbracket \\ &\quad \rho \mapsto \xi.b.\rho^2, \alpha_0 \mapsto \text{TRUE}, \\ &\quad b \mapsto \llbracket c \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} $	<pre> 1 \documentclass{article} 2 \pagestyle{empty} 3 \usepackage{eolang} 4 \begin{document} 5 \begin{phiquestion*} 6 a -> [[% it's abstract! 7 ^ !-> \$.b.^{^2}, 0-> TRUE, 8 b -> [[c -> fn (56), 9 @ -> hello (\$), 10 \Delta ..> 01-FE-C3]]]],\\ 11 x -> [[\alpha_0 -> ?]]. 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`),
- “->” 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 “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.

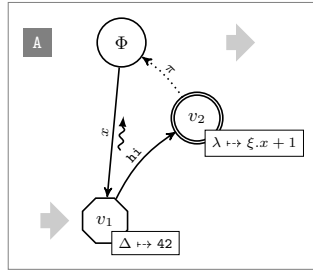
Numbers are automatically converted to fixed-width font, no need to always decorate them with vertical bars.

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

`\phiiq` The command `\phiiq` 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 \phiiq{x -> [[@ -> y]]} \\\ 7 is a decorator of 8 the data object \\\ 9 \phiiq{y -> [[\Delta .> 42]]}. 10 \end{document} </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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



```

1 \documentclass{article}
2 \pagestyle{empty}
3 \usepackage{eolang}
4 \begin{document}
5 \begin{sodg}
6 v0 \\\ v0==> \\\ v0!!A
7 v1 xy:v0,-.8,2.8 data:42 \\\ =>v1
8 v0->v1 a:x rho
9 v2 xy:v0,+1,+1 atom:\xi.x+1
10 v1->v2 a:|h1| bend:-15
11 v2->v0 pi bend:10 % a comment
12 \end{sodg}
13 \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,
- “rrho” places a reverse rho,
- “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 $\llbracket \alpha_i \xrightarrow{n} x_i \rrbracket$ 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 `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:

$\frac{x \mapsto [\varphi \mapsto y] \quad y \mapsto [z \mapsto 42]}{x.z \mapsto 42} \text{R1}$	<pre> 6 \begin{phiquation*} 7 \dfrac \ 8 {x->[[@->y]] \quad y->[[z->42]]} \ 9 {x.z -> 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}`:

$\frac{\begin{array}{l} x \mapsto [\varphi \mapsto y, z \mapsto 42, \\ \alpha_0 \mapsto \emptyset, \alpha_1 \mapsto 42] \end{array}}{\begin{array}{l} x \mapsto [\varphi \mapsto y, z \mapsto \emptyset, f \rightsquigarrow \text{pi}(\alpha_0 \mapsto [\psi \rightsquigarrow \text{hello}(12)], \\ \alpha_1 \mapsto 42)] \end{array}} \text{R2.}$	<pre> 6 \begin{phiquation*} 7 \dfrac{\begin{split} 8 x->[[@->y, z->42, 9 0->?, 1->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{\textsf{R2}.} 14 \end{phiquation*} 15 \end{phiquation*} </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

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

It's possible to use `\label` inside `phiquation` 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{phiquation} 9 D = b^{^2} - 4ac. 10 \label{d} 11 \end{phiquation} 12 Eq.\ref{d} is also widely used in 13 number theory and polynomial factoring. </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

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

```

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

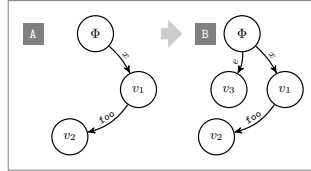
$$\begin{aligned} [b \mapsto \emptyset], \\ [\varphi \mapsto \text{TRUE}, \Delta \mapsto 42], \\ \Delta = 43-09. \end{aligned}$$

```

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

```

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

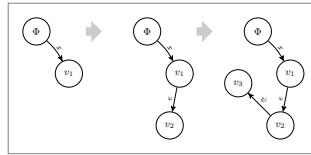


```

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:

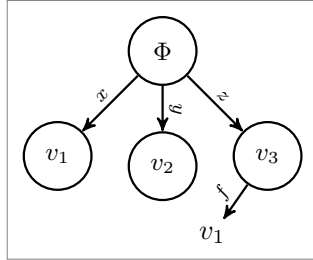


```

5 \begin{sodg}
6 v0
7 v1 xy:v0,.7,1
8 v0->v1 a:x bend:-10
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
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:

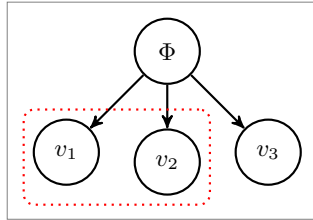


```

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 tmpdir
14 }
15 \ProcessPgfOptions{/eolang}

```

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

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

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

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

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

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

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

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



```

61 $tex =~ s/&([^\s\\]+)\s\/1&/g;
62 $gathered = 0;
63 }
64 }
65 $tex =~ s/\$/\\xi{/g;
66 $tex =~ s/(?<!\{)\^\/\rho{/g;
67 $tex =~ s/\[/\\lbracket\\mathrel{/g;
68 $tex =~ s/\]/\\rrbracket{/g;
69 $tex =~ s/([^\s,>()]{0-9A-F}{2}(?:-[0-9A-F]{2})?|[0-9]+(?:\.[0-9]+)?)/1|2|/g;
70 $tex =~ s/TRUE/|TRUE|/g;
71 $tex =~ s/FALSE/|FALSE|/g;
72 $tex =~ s/\?/\\varnothing{/g;
73 $tex =~ s/@/\\varphi{/g;
74 $tex =~ s/-([a-z]+)>/\\mathrel{\\phiSlot{1}}/g;
75 $tex =~ s/!->/\\mathrel{\\phiConst}/g;
76 $tex =~ s/->/\\mathrel{\\mapsto}/g;
77 $tex =~ s/~>/\\mathrel{\\phiWave}/g;
78 $tex =~ s/:=/\\mathrel{\\vDash}/g;
79 $tex =~ s/\\.\\.\\.>/\\mathrel{\\phiDotted}/g;
80 $tex =~ s/\\|{2,}/|/g;
81 $tex =~ s/\\|([^\|]+)\\|/\\textnormal{\\texttt{1}}{/g;
82 if ($env eq 'phiq') {
83   print '$' if ($tex ne '');
84 } else {
85   print '\\begin{', $env, '}';
86   if ($gathered) {
87     print '\\begin{gathered}';
88   } else {
89     print '\\begin{split}';
90   }
91 }
92 if ($gathered) {
93   $tex =~ s/(\\\\(?:\\[[^\]]+\\])?)&/1/g;
94   $tex =~ s/^&/g;
95 }
96 print $tex;
97 if ($env eq 'phiq') {
98   print '$' if ($tex ne '');
99 } else {
100   if ($gathered) {
101     print '\\end{gathered}';
102   } else {
103     print '\\end{split}';
104   }
105   print '\\end{', $env, '}';
106 }
107 print '\\endinput%';
108 \\end{VerbatimOut}
109 \\message{eolang: File with Perl script
110   'eolang@tmpdir/eolang-phi.pl' saved^^J}%
111 \\xexec[trace,null]{perl -pi -e 's/(\\\\[a-zA-Z])\\s+\\/1/g'
112   "eolang@tmpdir/eolang-phi.pl"}
113 \\makeatother

```

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

```

114 \makeatletter\newcommand\eolang@process[1]{
115   \def\hash{\eolang@mdfive
116     {\eolang@tmpdir/\jobname/phiquation.tex}}%
117   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiquation.tex"
118     "\eolang@tmpdir/\jobname/\hash.tex"}%
119   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
120     perl "\eolang@tmpdir/eolang-phi.pl"
121     '#1'
122     "\eolang@tmpdir/\jobname/\hash.tex"}%
123   \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
124 }
125 \newenvironment{phiquation*}%
126 {\VerbatimEnvironment%
127 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
128 \begin{VerbatimOut}
129   {\eolang@tmpdir/\jobname/phiquation.tex}}
130 {\end{VerbatimOut}\eolang@process{equation*}}
131 \newenvironment{phiquation}%
132 {\VerbatimEnvironment%
133 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
134 \begin{VerbatimOut}
135   {\eolang@tmpdir/\jobname/phiquation.tex}}
136 {\end{VerbatimOut}\eolang@process{equation}}
137 \makeatother

```

\phiq Then, we define \phiq command:

```

138 \makeatletter\newcommand\phiq[1]{%
139   \iexec[trace,quiet,stdout=\eolang@tmpdir/\jobname/phiq.tex]{
140     /bin/echo '\detokenize{#1}'}%
141   \def\hash{\eolang@mdfive
142     {\eolang@tmpdir/\jobname/phiq.tex}}%
143   \iexec[null]{cp "\eolang@tmpdir/\jobname/phiq.tex"
144     "\eolang@tmpdir/\jobname/\hash.tex"}%
145   \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
146     perl \eolang@tmpdir/eolang-phi.pl '\phiq'
147     "\eolang@tmpdir/\jobname/\hash.tex"}%
148 }\makeatother

```

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

[fancyvrb](#):

```

149 \makeatletter
150 \begin{VerbatimOut}{\eolang@tmpdir/eolang-sodg.pl}
151 sub num {
152   my ($i) = @_;
153   $i =~ s/(\+|-)\./\10./g;
154   return $i;
155 }
156 sub fmt {
157   my ($tex) = @_;
158   $tex =~ s/\\|([^\|]+)\\|\\textnormal{\\texttt{\\1}}/g;
159   return $tex;
160 }

```

```

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

```

```

215     $draw = $draw . '<MB:phi-pi><F:draw=none>';
216     if (not exists $opts{'a'}) {
217         $opts{'a'} = '\pi';
218     }
219 }
220 $draw = $draw . ']'';
221 my ($from, $to) = split (/>/, $head);
222 $draw = $draw . " ($from) ";
223 if (exists $opts{'bend'}) {
224     $draw = $draw . 'edge [<F:draw=none,><MF:bend right=' .
225         num($opts{'bend'}) . '>]';
226 } else {
227     $draw = $draw . '--';
228 }
229 if (exists $opts{'rho'} or exists $opts{'rrho'}) {
230     $draw = $draw . ' pic[sloped,phi-rho]{parallel arrow={';
231     $draw = $draw . '-' if not exists $opts{'rrho'};
232     $draw = $draw . '0.3,-0.15}}';
233 }
234 if (exists $opts{'a'}) {
235     my $a = $opts{'a'};
236     if (index($a, '$') == -1) {
237         $a = '$' . fmt($a) . '$';
238     } else {
239         $a = fmt($a);
240     }
241     $draw = $draw . '<MB: node [phi-attr] {' . $a . '>';
242 }
243 if (exists $opts{'break'}) {
244     $draw = $draw . '<F: coordinate [pos=' .
245         ($opts{'break'} / 100) . '] (break)>';
246 }
247 $draw = $draw . " (<MF:$to><B:break-v>)"';
248 if (exists $opts{'break'}) {
249     print tailor($draw, 'F') . ";\n";
250     print ' \node[outer sep=.1cm,inner sep=0cm] ' .
251         'at (break) (break-v) {$' . vertex($to) .
252         '$};' . "\n";
253     print ' ' . tailor($draw, 'B');
254 } else {
255     print tailor($draw, 'M');
256 }
257 } elsif (index($head, '>') >= 0) {
258     my ($from, $to) = split (/>+/, $head);
259     my $size = () = $head =~ /=/g;
260     if ($from eq '') {
261         print '\node [phi-arrow, left=' . ($size * 0.6) . 'cm of ' .
262             $to . '.center]';
263     } elsif ($to eq '') {
264         print '\node [phi-arrow, right=' . ($size * 0.6) . 'cm of ' .
265             $from . '.center]';
266     } else {
267         print '\node [phi-arrow] at ($(' .
268             $from . ')!0.5!(' . $to . ')$)';

```

```

269 }
270 print '{}';
271 } elseif (index($head, '!') >= 0) {
272   my ($v, $marker) = split (/!/, $head);
273   my $size = () = $head =~ !/g;
274   print '\node [phi-marker, left=' .
275     ($size * 0.6) . 'cm of ' .
276     $v . '.center]{' . fmt($marker) . '}';
277 } elseif (index($head, '+') >= 0) {
278   my ($v, $suffix) = split (/+/, $head);
279   my @friends = ($v);
280   foreach my $c (@cmds) {
281     $e = $c;
282     $e =~ s/^\s+//g;
283     my $h = $e;
284     $h = substr($e, 0, index($e, ' ')) if index($e, ' ') >= 0;
285     foreach my $f (@friends) {
286       my $add = '';
287       if (index($h, $f . '->') >= 0) {
288         $add = substr($h, index($h, '->') + 2);
289       }
290       if ($h =~ /->\Q${f}\E/) {
291         $add = substr($h, 0, index($h, '->'));
292       }
293       if (index($e, ' xy:' . $f . ',') >= 0) {
294         $add = $h;
295       }
296       if (index($add, '+') == -1
297         and $add ne ''
298         and not(grep(/^Q${add}\E/, @friends))) {
299         push(@friends, $add);
300       }
301     }
302   }
303   my @extra = ();
304   foreach my $e (@cmds) {
305     $m = $e;
306     if ($m =~ /\s*\Q${v}\E\s/) {
307       next;
308     }
309     if ($m =~ /\s*[\s]+\s/ and not($m =~ /\s*\Q${head}\E\s/)) {
310       next;
311     }
312     foreach my $f (@friends) {
313       my $h = $f;
314       $h =~ s/[a-z]//g;
315       if ($m =~ s/^(\\s*)\Q${f}\E+\Q${suffix}\E\s?/\1${h}${suffix} /g) {
316         last;
317       }
318       $m =~ s/^(\\s*)\Q${f}\E\s/\1${h}${suffix} /g;
319       $m =~ s/^(\\s*)\Q${f}\E->/\1${h}${suffix}->/g;
320       $m =~ s/\\sxy:\Q${f}\E,/ xy:${h}${suffix},/g;
321       $m =~ s/->\Q${f}\E\s/->${h}${suffix} /g;
322     }

```

```

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

```

```

377     }
378   } else {
379     print ',phi-object';
380   }
381   print ']';
382   print '(', $head, ')';
383   print '{ $' . vertex($head) . '$ }';
384   if (exists $opts{'box'}) {
385     print ' node[phi-box] at (';
386     print $head, '.south east) {';
387     print $opts{'box'}, ')';
388   }
389 } else {
390   print $cmd;
391 }
392 print ";\n";
393 }
394 print '\end{picture}%', "\n";
395 print "% --- after processing:\n%";
396 foreach my $c (@cmds) {
397   print '% ', $c, "\n";
398 }
399 print '% --- (' . (0+@cmds) . " lines)\n";
400 print '\endinput%';
401 \end{VerbatimOut}
402 \message{eolang: File with Perl script
403   '\eolang@tmpdir/eolang-sodg.pl' saved^^J}%
404 \iexec[trace,null]{perl -pi -e 's/(\\[a-zA-Z])\\s+\\/\\1/g'
405   "\eolang@tmpdir/eolang-sodg.pl"}
406 \makeatother

```

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

```

407 \setcounter{FancyVerbLine}{0}

```

tikz Then, we include tikz package and its libraries:

```

408 \RequirePackage{tikz}
409 \usetikzlibrary{arrows}
410 \usetikzlibrary{shapes}
411 \usetikzlibrary{snakes}
412 \usetikzlibrary{decorations}
413 \usetikzlibrary{decorations.pathmorphing}
414 \usetikzlibrary{positioning}
415 \usetikzlibrary{calc}

```

picture Then, we define internal environment picture:

```

416 \newenvironment{picture}%
417   {\noindent\begin{tikzpicture}[
418     ->,>=stealth',node distance=0,thick,
419     pics/parallel arrow/.style={
420       code={\draw[-latex,phi-rho] (##1) -- (-##1);}}}%
421   {\end{tikzpicture}}
422 \tikzstyle{phi-arrow} = [fill=white!80!black, single arrow,
423   minimum height=0.5cm, minimum width=0.5cm,

```

```

424 single arrow head extend=2mm]
425 \tikzstyle{phi-marker} = [inner sep=0pt, minimum height=1.4em,
426 minimum width=1.4em, font={\small\color{white}\ttfamily},
427 fill=gray]
428 \tikzstyle{phi-thing} = [thick,inner sep=0pt,minimum height=2.4em,
429 draw,font={\small}]
430 \tikzstyle{phi-object} = [phi-thing,circle]
431 \tikzstyle{phi-data} = [phi-thing,regular polygon,
432 regular polygon sides=8]
433 \tikzstyle{phi-empty} = [phi-object]
434 \tikzstyle{phi-rho} = [draw,decorate,decoration={
435 snake,amplitude=.4mm,segment length=2mm,post length=1mm}]
436 \tikzstyle{phi-pi} = [draw,dotted]
437 \tikzstyle{phi-atom} = [phi-object,double]
438 \tikzstyle{phi-box} = [xshift=-5pt,yshift=3pt,draw,fill=white,
439 rectangle,thin,minimum width=1.2em,anchor=north west,
440 font={\scriptsize}]
441 \tikzstyle{phi-attr} = [midway,sloped,inner sep=0pt,
442 above=2pt,sloped/.append style={transform shape},
443 font={\scriptsize},color=black]

```

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

```

444 \makeatletter\newenvironment{sodg}%
445 {\VerbatimEnvironment%
446 \setcounter{eolang@lineno}{\value{FancyVerbLine}}%
447 \begin{VerbatimOut}
448 {\eolang@tmpdir/\jobname/sodg.tex}}
449 {\end{VerbatimOut}}%
450 \def\hash{\eolang@mdfive
451 {\eolang@tmpdir/\jobname/sodg.tex}}%
452 \iexec[null]{cp "\eolang@tmpdir/\jobname/sodg.tex"
453 "\eolang@tmpdir/\jobname/\hash.tex"}}%
454 \iexec[trace,stdout=\eolang@tmpdir/\jobname/\hash-post.tex]{
455 perl "\eolang@tmpdir/eolang-sodg.pl"
456 "\eolang@tmpdir/\jobname/\hash.tex"}}%
457 \setcounter{FancyVerbLine}{\value{eolang@lineno}}%
458 }\makeatother

```

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

```

459 \newcommand\eolang{%
460 \ifdefined\anon%
461 \anon[XYZ]{\sffamily EO}}%
462 \else%
463 {\sffamily EO}}%
464 \fi%
465 }

```

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

```

466 \RequirePackage{hyperref}
467 \newcommand\phic{%
468 \ifdefined\anon%
469 \anon[\texorpdfstring{$\alpha$}{a}-calculus]

```



```

470      {\texorpdfstring{\varphi}{phi}-calculus}%
471 \else%
472   \texorpdfstring{\varphi}{phi}-calculus%
473 \fi%
474 }

```

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

```

475 \newcommand\xmir{%
476   \ifdefined\anon%
477     \anon[XML$~+$]{XMIR}%
478   \else%
479     XMIR%
480   \fi%
481 }

```

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

```

482 \newcommand\phiConst{%
483   \mathrel{\hspace{.15em}}%
484   \mapstochar\mathrel{\hspace{-.15em}}\mapsto

```

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

```

485 \newcommand\phiWave{%
486   \mapstochar\mathrel{\mspace{0.45mu}}\leadsto

```

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

```

487 \newcommand\phiSlot[1]{%
488   \xrightarrow{\text{\sffamily\scshape #1}}

```

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

```

489 \newcommand\phiMany[3]{%
490   \overunderset{\scriptscriptstyle #3}{\scriptscriptstyle #2}{#1}

```

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

```

491 \RequirePackage{trimclip}
492 \RequirePackage{amsfonts}
493 \makeatletter
494 \newcommand\phiDotted{%
495   \mapstochar\mathrel{\mathpalette\phiDotted@\relax}}
496 \newcommand\phiDotted@[2]{%
497   \begingroup
498   \settowidth{\dimen\z@}{\m@th#1\rightarrow}%
499   \settoheight{\dimen\tw@}{\m@th#1\rightarrow}%
500   \sbox\z@{%
501     \makebox[\dimen\z@][s]{%
502       \clipbox{0 0 {0.4\width} 0}%
503       {\resizebox{\dimen\z@}{\height}%
504         {\m@th#1\dashrightarrow}}}%
505     \hss%
506     \clipbox{{0.69\width} {-0.1\height} 0
507       {-\height}}{\m@th#1\rightarrow}%

```

```

508     }%
509 }%
510 \ht\z@=\dimen\tw@ \dp\z@=\z@%
511 \box\z@%
512 \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	
0.0.2		
sodg: The environment <code>phigure</code>		
renamed to <code>sodg</code> for the sake of		
better semantic. The graph in the		
picture is solely a SODG graph,		
that's why the name <code>sodg</code> is better.	16	
eolang-phi.pl: New symbol added		
for basket slots	8	
Parsing of symbols “@”, “^”, and “&”		
enabled (<code>\varphi</code> , <code>\rho</code> , and		
<code>\sigma</code>)	8	
The symbols “[” and “]” replaced		
with “[” and “]” for abstract		
object brackets, because they		
conflicted with normal square		
brackets	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.	10	
<code>\phi</code> : Parsing of additional symbols		
enabled.	10	
0.1.0		
General: Parsing of package options		
introduced.	7	
<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>	16	
<code>\eolang@mdfive</code> : New		
supplementary command added to		
calculate MD5 sum of a file.	8	
eolang-phi.pl: A new Perl script		
“eolang-phi.pl” added for		
parsing of ϕ expressions.	8	
eolang-sodg.pl: There are two Perl		
scripts now: one for ϕ ication,		
another one for <code>sodg</code>	10	
<code>\phi</code> : New command <code>\phi</code> prints		
the name of ϕ -calculus in both		
normal and anonymous mode of		
<code>acmart</code>	16	
<code>\phiConst</code> : New command		
<code>\phiConst</code> added to denote a link		
to a constant attribute.	17	
		<code>\phiDotted</code> : New command
		<code>\phiDotted</code> added to denote a link
		to a special attribute.
		17
	0.2.0	
	eolang-phi.pl: Numbers	
	automatically render as <code>\texttt</code> .	
	No need to use vertical bars around	
	them anymore.	8
	eolang-sodg.pl: The content of	
	<code>atom</code> and <code>data</code> boxes is parsed	
	automatically as formulas and	
	numbers, respectively.	10
	<code>\xm</code> : New command <code>\xm</code> prints	
	XMIR in both normal and	
	anonymous mode of <code>acmart</code>	17
	0.3.0	
	<code>\eolang@lineno</code> : New counter for	
	protecting <code>lineno</code>	8
	eolang-phi.pl: New arrow added,	
	that looks like <code>\leadsto</code>	8
	<code>\phiWave</code> : New command <code>\phiWave</code>	
	added to denote a link to a	
	multi-layer attribute.	17
	0.4.0	
	eolang-sodg.pl: Labels on the edges	
	are automatically printed as math	
	formulas. Also, boxes are prefixed	
	with <code>\Delta</code> and <code>\lambda</code>	
	commands.	10
	Relative positioning of vertices fixed.	10
	0.5.0	
	eolang-phi.pl: Automated	
	formatting of <code>TRUE</code> and <code>FALSE</code>	
	added.	8
	eolang-sodg.pl: It is possible to use	
	<code>tikz</code> commands inside <code>sodg</code>	
	environment.	10
	New syntax introduced that allows	
	to make clones of vertices and all	
	their dependants.	10
	Now edges may have <code>break</code>	
	attribute, to make them shorter.	10
	<code>\phiMany</code> : New command <code>\phiMany</code>	
	enables iterating over an arrow.	17
	<code>\phiSlot</code> : New command <code>\phiSlot</code>	
	added to denote a link to a slot in a	
	basket.	17

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		M	
$\backslash \$$	65	$\backslash m@th$. . .	498, 499, 504, 507
$\backslash ($	69	$\backslash makeatletter$	17, 19, 22,
$\backslash +$	153, 278, 309, 315		114, 138, 149, 444, 493
$\backslash .$	49, 69, 79, 153	$\backslash makeatother$	17, 21, 113,
$\backslash ?$	72		137, 148, 406, 458, 512
$\backslash [$	67, 93	$\backslash makebox$	501
$\backslash \{$	46, 51, 52, 66	$\backslash mapsto$	484
$\backslash \}$	46, 51, 52, 61	$\backslash mapstochar$.	484, 486, 495
$\backslash]$	68, 93	$\backslash mathpalette$	495
$\backslash ^$	66	$\backslash mathrel$	483, 484, 486, 495
$\backslash $	80, 81, 158	$\backslash message$	109, 402
Numbers		$\backslash mspace$	486
$\backslash 2$	49, 55, 69, 173	N	
$\backslash 3$	49	$\backslash newcommand$	20, 114, 138,
A			459, 467, 475, 482,
$\backslash alpha$	469		485, 487, 489, 494, 496
$\backslash anon$	460,	$\backslash newcounter$	17
	461, 468, 469, 476, 477	$\backslash newenvironment$. . .
B			125, 131, 416, 444
$\backslash Bbbk$	3	$\backslash node$	250,
$\backslash begin$	23, 85, 87, 89, 128,		261, 264, 267, 274, 337
	134, 150, 201, 417, 447	$\backslash noindent$	417
$\backslash box$	511	O	
C		$\backslash overunderset$	490
$\backslash clipbox$	502, 506	P	
$\backslash color$	426	$\backslash pdf@filemdfivesum$. .	20
D		$\backslash pgfkeys$	9
$\backslash dashrightarrow$. . .	504	$\backslash Phi$	164
$\backslash def$	115, 141, 450	$\backslash phic$	466
$\backslash Delta$	360	$\backslash phiConst$	482
$\backslash detokenize$	140	$\backslash phicture$	416
$\backslash dimen$	498, 499, 501, 503, 510	$\backslash phiDotted$. .	360, 372, 491
$\backslash dp$	510	$\backslash phiDotted@$. . .	495, 496
$\backslash draw$	213, 420	$\backslash phiMany$	489
E		$\backslash phiq$	138
$\backslash E$. . .	290, 298, 306, 309,	$\backslash phiquation$	114
	315, 318, 319, 320, 321	$\backslash phiSlot$	487
$\backslash end$	101,	$\backslash phiWave$	485
	103, 105, 108, 130,	$\backslash pi$	217
	136, 394, 401, 421, 449	$\backslash ProcessPgfoptions$. .	15
$\backslash endinput$	107, 400	Q	
F		$\backslash Q$. . .	290, 298, 306, 309,
$\backslash FancyVerbLine$	407		315, 318, 319, 320, 321
H		L	
$\backslash hash$. . .	115, 118, 119,	$\backslash lambda$	372
	122, 141, 144, 145,	$\backslash leadsto$	486
	147, 450, 453, 454, 456	J	
$\backslash height$	503, 506, 507	$\backslash jobname$.	16, 116, 117,
$\backslash hspace$	483, 484		118, 119, 122, 129,
$\backslash hss$	505		135, 139, 142, 143,
$\backslash ht$	510		144, 145, 147, 448,
I			451, 452, 453, 454, 456
$\backslash iexec$	16,	K	
	111, 117, 119, 139,	M	
	143, 145, 404, 452, 454	N	
$\backslash ifdefined$. .	460, 468, 476	O	
$\backslash ifluatex$	12	P	
$\backslash ifxetex$	12	Q	
J		R	
$\backslash jobname$.	16, 116, 117,	S	
	118, 119, 122, 129,	T	
	135, 139, 142, 143,	U	
	144, 145, 147, 448,	V	
	451, 452, 453, 454, 456	W	
L		X	
$\backslash lambda$	372	Y	
$\backslash leadsto$	486	Z	

R		
<code>\relax</code>	3, 495	
<code>\RequirePackage</code> ..	1,	
	2, 3, 4, 5, 6, 7, 8,	
	18, 408, 466, 491, 492	
<code>\resizebox</code>	503	
<code>\rightarrow</code> .	498, 499, 507	
S		
<code>\sbox</code>	500	
<code>\scriptscriptstyle</code> .	490	
<code>\scriptsize</code>	440, 443	
<code>\scshape</code>	488	
<code>\setcounter</code>	123,	
	127, 133, 407, 446, 457	
<code>\settoheight</code>	499	
<code>\settowidth</code>	498	
<code>\sffamily</code> ..	461, 463, 488	
<code>\small</code>	426, 429	
<code>\sodg</code>	444	
<code>\sxy</code>	320	409, 410,
		411, 412, 413, 414, 415
T		
<code>\t</code>	31, 179	
<code>\texorpdfstring</code> ...		
	469, 470, 472	
<code>\text</code>	360, 488	
<code>\textnormal</code>	361	
<code>\texttt</code>	361	
<code>\tikz</code>	408	
<code>\tikzstyle</code> ..	422, 425,	
	428, 430, 431, 433,	
	434, 436, 437, 438, 441	
<code>\ttfamily</code>	426	
<code>\tw@</code>	499, 510	
U		
<code>\usetikzlibrary</code> ...		
V		
<code>\value</code> 123, 127, 133, 446, 457		
<code>\varphi</code>	470, 472	
<code>\VerbatimEnvironment</code>		
	126, 132, 445	
W		
<code>\width</code>	502, 506	
X		
<code>\xmir</code>	475	
<code>\xrightarrow</code>	488	
Z		
<code>\z@</code> 498, 500, 501, 503, 510, 511		