

# Implementation of FOREST, a PGF/TikZ-based package for drawing linguistic trees

v2.0.2

Sašo Živanović\*

March 4, 2016

This file contains the documented source of FOREST. If you are searching for the manual, follow this link to [forest-doc.pdf](#).

The latest release of the package, including the sources, can be found on [CTAN](#). For all versions of the package, including any non-yet-released work in progress, visit [FOREST's GitHub repo](#). Contributions are welcome.

A disclaimer: the code could've been much cleaner and better-documented ...

## Contents

1	Identification	2
2	Package options	2
3	Patches	3
4	Utilities	3
4.1	Sorting	9
5	The bracket representation parser	13
5.1	The user interface macros	13
5.2	Parsing	14
5.3	The tree-structure interface	18
6	Nodes	19
6.1	Option setting and retrieval	19
6.2	Tree structure	22
6.3	Node options	30
6.3.1	Option-declaration mechanism	30
6.3.2	Registers	37
6.3.3	Declaring options	43
6.3.4	Option propagation	49
6.4	Aggregate functions	51
6.4.1	pgfmath extensions	53
6.5	Nodewalk	55
6.6	Dynamic tree	73
7	Stages	77
7.1	Typesetting nodes	80
7.2	Packing	82
7.2.1	Tiers	93
7.2.2	Node boundary	96
7.3	Compute absolute positions	101
7.4	Drawing the tree	102

---

\*e-mail: [saso.zivanovic@guest.arnes.si](mailto:saso.zivanovic@guest.arnes.si); web: <http://spj.ff.uni-lj.si/zivanovic/>

8	Geometry	104
8.1	Projections	104
8.2	Break path	107
8.3	Get tight edge of path	109
8.4	Get rectangle/band edge	115
8.5	Distance between paths	116
8.6	Utilities	118
9	The outer UI	120
9.1	Externalization	120
9.2	The <code>forest</code> environment	121
9.3	Standard node	124
9.4	1s coordinate system	126
9.5	Relative node names in <code>TikZ</code>	127
9.6	Anchors	128
10	Compatibility with previous versions	133

## 1 Identification

```

1 \ProvidesPackage{forest}[2016/03/04 v2.0.2 Drawing (linguistic) trees]
2
3 \RequirePackage{tikz}[2013/12/13]
4 \usetikzlibrary{shapes}
5 \usetikzlibrary{fit}
6 \usetikzlibrary{calc}
7 \usepgflibrary{intersections}
8
9 \RequirePackage{pgfopts}
10 \RequirePackage{etoolbox}[2010/08/21]
11 \RequirePackage{elocalloc}% for \locbox
12 \RequirePackage{environ}
13 \RequirePackage{xparse}
14
15 % \usepackage[trace]{trace-pgfkeys}
16
17 /forest is the root of the key hierarchy.
18 \pgfkeys{/forest/.is family}
19 \def\forestset#1{\pgfkeys{/forest}{#1}}

```

## 2 Package options

```

19 \newif\ifforest@external@
20 \newif\ifforesttikzcshack
21 \newif\ifforest@install@keys@to@tikz@path@
22 \newif\ifforestdebug
23 \def\forest@compat{}
24 \forestset{package@options/.cd,
25   external/.is if=forest@external@,
26   tikzcshack/.is if=foresttikzcshack,
27   tikzinstallkeys/.is if=forest@install@keys@to@tikz@path@,
28   debug/.is if=forestdebug,
29   compat/.store in=\forest@compat,
30   compat/.default=most,
31   .unknown/.code={% load library
32     \eappto\forest@loadlibrarieslater{%
33       \noexpand\useforestlibrary{\pgfkeyscurrentname}%
34       \noexpand\forestapplylibrarydefaults{\pgfkeyscurrentname}%
35     }%
36   },

```

```

37 }
38 \forest@install@keys@to@tikz@path@true
39 \foresttikz@cs@hack@true
40 \def\forest@load@libraries@later{}
41 \AtEndOfPackage{\forest@load@libraries@later}
42 \NewDocumentCommand\useforestlibrary{s 0{} m}{%
43   \def\useforestlibrary@##1{\useforestlibrary@{#2}{##1}}%
44   \forcsvlist\useforestlibrary@{#3}%
45   \IfBooleanT{#1}{\forestapplylibrarydefaults{#3}}%
46 }
47 \def\useforestlibrary@#1#2{\RequirePackage[#1]{forest-lib-#2}}
48 \def\forestapplylibrarydefaults#1{\forcsvlist\forestapplylibrarydefaults@{#1}}
49 \def\forestapplylibrarydefaults@#1{\forestset{libraries/#1/defaults/.try}}
50 \NewDocumentCommand\ProvidesForestLibrary{m 0{}}{\ProvidesPackage{forest-lib-#1}[#2]}
51 \ProcessPgfPackageOptions{/forest/package@options}

```

### 3 Patches

This macro implements a fairly safe patching mechanism: the code is only patched if the original hasn't changed. If it did change, a warning message is printed. (This produces a spurious warning when the new version of the code fixes something else too, but what the heck.)

```

52 \def\forest@patch#1#2#3#4#5{%
53   % #1 = cs to be patched
54   % #2 = purpose of the patch
55   % #3 = macro arguments
56   % #4 = original code
57   % #5 = patched code
58   \csdef{forest@original@#1}#3{#4}%
59   \csdef{forest@patched@#1}#3{#5}%
60   \ifcsequal{#1}{forest@original@#1}{%
61     \csletcs{#1}{forest@patched@#1}%
62   }{%
63     \ifcsequal{#1}{forest@patched@#1}{% all is good, the patch is in!
64     }{%
65       \PackageWarning{forest}{Failed patching '\expandafter\string\cname #1\endcsname'. Purpose of the patch
66     }%
67   }%
68 }

```

Patches for PGF 3.0.0 — required version is [2013/12/13].

```

69 \forest@patch{pgf@get@transform}{fix a leaking space}{#1}{%
70   \edef#1{\pgf@pt@aa}{\pgf@pt@ab}{\pgf@pt@ba}{\pgf@pt@bb}{\the\pgf@pt@x}{\the\pgf@pt@y}}
71 }{%
72   \edef#1{\pgf@pt@aa}{\pgf@pt@ab}{\pgf@pt@ba}{\pgf@pt@bb}{\the\pgf@pt@x}{\the\pgf@pt@y}}%
73 }

```

### 4 Utilities

Escaping \ifs.

```

74 \long\def\@escapeif#1#2\fi{\fi#1}
75 \long\def\@escapeifif#1#2\fi#3\fi{\fi\fi#1}
76 \long\def\@escapeififif#1#2\fi#3\fi#4\fi{\fi\fi\fi#1}

```

A factory for creating \dots loop\dots macros.

```

77 \def\newloop#1{%
78   \count@=\escapechar
79   \escapechar=-1
80   \expandafter\newloop@parse@loopname\string#1\newloop@end
81   \escapechar=\count@

```

```

82 }%
83 {\lccode'7='1 \lccode'8='o \lccode'9='p
84 \lowercase{\gdef\newloop@parse@loopname#17889#2\newloop@end{%
85   \edef\newloop@marshal{%
86     \noexpand\csdef{#1loop#2}###1\expandafter\noexpand\csname #1repeat#2\endcsname{%
87       \noexpand\csdef{#1iterate#2}{###1\relax\noexpand\expandafter\expandafter\noexpand\csname#1iterate#
88       \expandafter\noexpand\csname#1iterate#2\endcsname
89       \let\expandafter\noexpand\csname#1iterate#2\endcsname\relax
90     }%
91   }%
92   \newloop@marshal
93 }%
94 }%
95 }%

```

Loop that can be arbitrarily nested. (Not in the same macro, however: use another macro for the inner loop.) Usage: `\safeloop_code_\if..._code_\saferepeat`. `\safeloopn` expands to the current repetition number of the innermost group.

```

96 \def\newsafeloop#1{%
97   \csdef{safeloop@#1}###1\saferepeat{%
98     \edef\safeloop@marshal{%
99       \noexpand\csdef{safeiterate@#1}{%
100         \unexpanded{##1}\relax
101         \noexpand\expandafter
102         \expandonce{\csname safeiterate@#1\endcsname}%
103         \noexpand\fi
104       }%
105     }\safeloop@marshal
106     \csuse{safeiterate@#1}%
107     \advance\noexpand\safeloop@depth-1\relax
108     \cslet{safeiterate@#1}\relax
109   }%
110 }%
111 \newcount\safeloop@depth
112 \def\safeloop{%
113   \advance\safeloop@depth1
114   \ifcsdef{safeloop@the\safeloop@depth}{-}{\expandafter\newsafeloop\expandafter{\the\safeloop@depth}}%
115   \csdef{safeloopn@the\safeloop@depth}{0}%
116   \csuse{safeloop@the\safeloop@depth}%
117   \csedef{safeloopn@the\safeloop@depth}{\number\numexpr\csuse{safeloopn@the\safeloop@depth}+1}%
118 }
119 \let\saferepeat\fi
120 \def\safeloopn{\csuse{safeloopn@the\safeloop@depth}}%

```

Another safeloop for usage with “repeat” / “while” // “until” keys, so that the user can refer to loop *ns* for outer loops.

```

121 \def\newsafeRKloop#1{%
122   \csdef{safeRKloop@#1}###1\safeRKrepeat{%
123     \edef\safeRKloop@marshal{%
124       \noexpand\csdef{safeRKiterate@#1}{%
125         \unexpanded{##1}\relax
126         \noexpand\expandafter
127         \expandonce{\csname safeRKiterate@#1\endcsname}%
128         \noexpand\fi
129       }%
130     }\safeRKloop@marshal
131     \csuse{safeRKiterate@#1}%
132     \advance\noexpand\safeRKloop@depth-1\relax
133     \cslet{safeRKiterate@#1}\relax
134   }%
135   \expandafter\newif\csname ifsafeRKbreak@the\safeRKloop@depth\endcsname

```

```

136 }%
137 \newcount\safeRkloop@depth
138 \def\safeRkloop{%
139   \advance\safeRkloop@depth1
140   \ifcsdef{safeRkloop@the\safeRkloop@depth}{\expandafter\newsafeRkloop\expandafter{\the\safeRkloop@depth}}
141   \csdef{safeRkloopn@the\safeRkloop@depth}{0}%
142   \csuse{safeRkbreak@the\safeRkloop@depth false}%
143   \csuse{safeRkloop@the\safeRkloop@depth}%
144   \csedef{safeRkloopn@the\safeRkloop@depth}{\number\numexpr\csuse{safeRkloopn@the\safeRkloop@depth}+1}%
145 }
146 \let\safeRkrepeat\fi
147 \def\safeRkloopn{\csuse{safeRkloopn@the\safeRkloop@depth}}%
  Additional loops (for embedding).
148 \newloop\forest@loop
  New counters, dimens, ifs.
149 \newdimen\forest@temp@dimen
150 \newcount\forest@temp@count
151 \newcount\forest@n
152 \newif\ifforest@temp
153 \newcount\forest@temp@global@count
154 \newtoks\forest@temp@toks
  Appending and prepending to token lists.
155 \def\etotoks#1#2{\edef\pot@temp{#2}\expandafter#1\expandafter{\pot@temp}}
156 \def\apptotoks#1#2{\expandafter#1\expandafter{\the#1#2}}
157 \long\def\lapptotoks#1#2{\expandafter#1\expandafter{\the#1#2}}
158 \def\eaaptotoks#1#2{\edef\pot@temp{#2}\expandafter\expandafter\expandafter#1\expandafter\expandafter\expandafter
159 \def\pretotoks#1#2{\toks@=#2}\expandafter\expandafter\expandafter#1\expandafter\expandafter\expandafter\expandafter
160 \def\epretotoks#1#2{\edef\pot@temp{#2}\expandafter\expandafter\expandafter#1\expandafter\expandafter\expandafter\expandafter
161 \def\gapptotoks#1#2{\expandafter\global\expandafter#1\expandafter{\the#1#2}}
162 \def\xapptotoks#1#2{\edef\pot@temp{#2}\expandafter\expandafter\expandafter\global\expandafter\expandafter\expandafter
163 \def\gpretotoks#1#2{\toks@=#2}\expandafter\expandafter\expandafter\global\expandafter\expandafter\expandafter
164 \def\xpretotoks#1#2{\edef\pot@temp{#2}\expandafter\expandafter\expandafter\global\expandafter\expandafter\expandafter
  Expanding number arguments.
165 \def\expandnumberarg#1#2{\expandafter#1\expandafter{\number#2}}
166 \def\expandtwonumberargs#1#2#3{%
167   \expandafter\expandtwonumberargs@\expandafter#1\expandafter{\number#3}{#2}}
168 \def\expandtwonumberargs@#1#2#3{%
169   \expandafter#1\expandafter{\number#3}{#2}}
170 \def\expandthreenumberargs#1#2#3#4{%
171   \expandafter\expandthreenumberargs@\expandafter#1\expandafter{\number#4}{#2}{#3}}
172 \def\expandthreenumberargs@#1#2#3#4{%
173   \expandafter\expandthreenumberargs@@\expandafter#1\expandafter{\number#4}{#2}{#3}}
174 \def\expandthreenumberargs@@#1#2#3#4{%
175   \expandafter#1\expandafter{\number#4}{#2}{#3}}
  A macro converting all non-alphanumerics (and an initial number) in a string to __. #1 = string, #2
  = receiving macro. Used for declaring pgfmath functions.
176 \def\forest@convert@others@to@underscores#1#2{%
177   \def\forest@cotu@result{}}%
178   \forest@cotu@first#1\forest@end
179   \let#2\forest@cotu@result
180 }
181 \def\forest@cotu{%
182   \let\forest@cotu@have@num\forest@cotu@have@alpha
183   \futurelet\forest@cotu@nextchar\forest@cotu@checkforspace
184 }
185 \def\forest@cotu@first{%
186   \let\forest@cotu@have@num\forest@cotu@have@other
187   \futurelet\forest@cotu@nextchar\forest@cotu@checkforspace

```

```

188 }
189 \def\forest@cotu@checkforspace{%
190   \expandafter\ifx\space\forest@cotu@nextchar
191   \let\forest@cotu@next\forest@cotu@havespace
192   \else
193   \let\forest@cotu@next\forest@cotu@nospace
194   \fi
195   \forest@cotu@next
196 }
197 \def\forest@cotu@havespace#1{%
198   \appto\forest@cotu@result{_}%
199   \forest@cotu#1%
200 }
201 \def\forest@cotu@nospace{%
202   \ifx\forest@cotu@nextchar\forest@end
203   \@escapeif@gobble
204   \else
205   \@escapeif\forest@cotu@nospaceB
206   \fi
207 }
208 \def\forest@cotu@nospaceB#1{%
209   \ifcat#1a%
210   \let\forest@cotu@next\forest@cotu@have@alpha
211   \else
212   \if!\ifnum9<1#1!\fi
213   \let\forest@cotu@next\forest@cotu@have@num
214   \else
215   \let\forest@cotu@next\forest@cotu@haveother
216   \fi
217   \fi
218   \forest@cotu@next#1%
219 }
220 \def\forest@cotu@have@alpha#1{%
221   \appto\forest@cotu@result{#1}%
222   \forest@cotu
223 }
224 \def\forest@cotu@haveother#1{%
225   \appto\forest@cotu@result{_}%
226   \forest@cotu
227 }

Additional list macros.

228 \def\forest@listedel#1#2{% #1 = list, #2 = item
229   \edef\forest@marshal{\noexpand\forest@listdel\noexpand#1{#2}}%
230   \forest@marshal
231 }
232 \def\forest@listcsdel#1#2{%
233   \expandafter\forest@listdel\csname #1\endcsname{#2}%
234 }
235 \def\forest@listcsedel#1#2{%
236   \expandafter\forest@listedel\csname #1\endcsname{#2}%
237 }
238 \edef\forest@restorelistsepcatcode{\noexpand\catcode'\the\catcode'\relax}%
239 \catcode'\|=3
240 \gdef\forest@listdel#1#2{%
241   \def\forest@listedel@A##1|#2|##2\forest@END{%
242     \forest@listedel@B##1|##2\forest@END%|
243   }%
244   \def\forest@listedel@B|##1\forest@END{%|
245     \def#1{##1}%
246   }%

```

```

247 \expandafter\forest@listedel@A\expandafter|#1\forest@END%|
248 }
249 \forest@restorelistsepcatcode
    Strip (the first level of) braces from all the tokens in the argument.
250 \def\forest@strip@braces#1{%
251   \forest@strip@braces@A#1\forest@strip@braces@preend\forest@strip@braces@end
252 }
253 \def\forest@strip@braces@A#1#2\forest@strip@braces@end{%
254   #1\ifx\forest@strip@braces@preend#2\else\@escapeif{\forest@strip@braces@A#2\forest@strip@braces@end}\fi
255 }

```

Utilities dealing with pgfkeys.

```

256 \def\forest@copycommandkey#1#2{% copies command of #1 into #2
257   \pgfkeysifdefined{#1/.@cmd}{-}%
258   \PackageError{forest}{Key #1 is not a command key}{-}%
259 }%
260 \pgfkeysgetvalue{#1/.@cmd}\forest@temp
261 \pgfkeyslet{#2/.@cmd}\forest@temp
262 \pgfkeysifdefined{#1/.@args}{-}%
263   \pgfkeysgetvalue{#1/.@args}\forest@temp
264   \pgfkeyslet{#2/.@args}\forest@temp
265 }{-}%
266 \pgfkeysifdefined{#1/.@body}{-}%
267   \pgfkeysgetvalue{#1/.@body}\forest@temp
268   \pgfkeyslet{#2/.@body}\forest@temp
269 }{-}%
270 \pgfkeysifdefined{#1/.@@body}{-}%
271   \pgfkeysgetvalue{#1/.@@body}\forest@temp
272   \pgfkeyslet{#2/.@@body}\forest@temp
273 }{-}%
274 \pgfkeysifdefined{#1/.@def}{-}%
275   \pgfkeysgetvalue{#1/.@def}\forest@temp
276   \pgfkeyslet{#2/.@def}\forest@temp
277 }{-}%
278 }
279 \forestset{
280   copy command key/.code 2 args={\forest@copycommandkey{#1}{#2}},
281   autoforward/.code 2 args={\forest@autoforward{#1}{#2={#1={##1}}}{true}},
282   autoforward'/ .code 2 args={\forest@autoforward{#1}{#2=#1,#2={#1={##1}}}{true}},
283   Autoforward/.code 2 args={\forest@autoforward{#1}{#2}{true}},
284   autoforward register/.code 2 args={\forest@autoforward{#1}{#2={#1={##1}}}{false}},
285   autoforward register'/ .code 2 args={\forest@autoforward{#1}{#2=#1,#2={#1={##1}}}{false}},
286   Autoforward register/.code 2 args={\forest@autoforward{#1}{#2}{false}},
287   copy command key@if it exists/.code 2 args={%
288     \pgfkeysifdefined{#1/.@cmd}{-}%
289     \forest@copycommandkey{#1}{#2}%
290   }{-}%
291 },
292   unautoforward/.style={
293     typeout={unautoforwarding #1},
294     copy command key@if it exists={/forest/autoforwarded #1}{/forest/#1},
295     copy command key@if it exists={/forest/autoforwarded #1+}{/forest/#1+},
296     copy command key@if it exists={/forest/autoforwarded #1-}{/forest/#1-},
297     copy command key@if it exists={/forest/autoforwarded #1*}{/forest/#1*},
298     copy command key@if it exists={/forest/autoforwarded #1:}{/forest/#1:},
299     copy command key@if it exists={/forest/autoforwarded #1'}{/forest/#1'},
300     copy command key@if it exists={/forest/autoforwarded #1+'}{/forest/#1+'},
301     copy command key@if it exists={/forest/autoforwarded #1-'}{/forest/#1-'},
302     copy command key@if it exists={/forest/autoforwarded #1*'}{/forest/#1*'},
303     copy command key@if it exists={/forest/autoforwarded #1:'}{/forest/#1:'},
304     copy command key@if it exists={/forest/autoforwarded +#1}{/forest/+#1},

```





```

366 \def\forestnovalue{\forestnovalue}%
367 \pgfqkeys{/forest/remove@key@installer}{#1}%
368 \let\forestnovalue\pgfkeysnovaluetext
369 \pgfqkeys{/forest/remove@key}{#2}%
370 \pgfqkeys{/forest/remove@key@uninstaller}{#1}%
371 #3\forest@temp@toks
372 }
373 \def\forest@remove@key@novalue{\forest@remove@key@novalue}%
374 \forestset{
375   remove@key@installer/.unknown/.code={% #1 = (outer) value
376     \def\forest@temp{#1}%
377     \ifx\forest@temp\pgfkeysnovalue@text
378       \pgfkeysdef{/forest/remove@key/\pgfkeyscurrentname}{}%
379     \else
380       \ifx\forest@temp\forestnovalue
381         \expandafter\forest@remove@key@installer@defwithvalue\expandafter{\pgfkeyscurrentname}{\pgfkeysnovalu
382       \else
383         \expandafter\forest@remove@key@installer@defwithvalue\expandafter{\pgfkeyscurrentname}{#1}%
384       \fi
385     \fi
386   },
387   remove@key/.unknown/.code={% #1 = (inner) value
388     \expandafter\apptotoks\expandafter\forest@temp@toks\expandafter{\pgfkeyscurrentname={#1},}%
389   },
390   remove@key@uninstaller/.unknown/.code={%
391     \pgfkeyslet{/forest/remove@key/\pgfkeyscurrentname/.@cmd}\@undefined},
392 }
393 \def\forest@remove@key@installer@defwithvalue#1#2{% #1=key name, #2 = outer value
394   \pgfkeysdef{/forest/remove@key/#1}{% ##1 = inner value
395     \def\forest@temp@outer{#2}%
396     \def\forest@temp@inner{##1}%
397     \ifx\forest@temp@outer\forest@temp@inner
398     \else
399       \apptotoks\forest@temp@toks{#1={##1},}%
400     \fi
401   }%
402 }

```

## 4.1 Sorting

Macro `\forest@sort` is the user interface to sorting.

The user should prepare the data in an arbitrarily encoded array,<sup>1</sup> and provide the sorting macro (given in #1) and the array let macro (given in #2): these are the only ways in which sorting algorithms access the data. Both user-given macros should take two parameters, which expand to array indices. The comparison macro should compare the given array items and call `\forest@sort@cmp@gt`, `\forest@sort@cmp@lt` or `\forest@sort@cmp@eq` to signal that the first item is greater than, less than, or equal to the second item. The let macro should “copy” the contents of the second item onto the first item.

The sorting direction is given in #3: it can be one of `\forest@sort@ascending` and `\forest@sort@descending`. #4 and #5 must expand to the lower and upper (both inclusive) indices of the array to be sorted.

`\forest@sort` is just a wrapper for the central sorting macro `\forest@@sort`, storing the comparison macro, the array let macro and the direction. The central sorting macro and the algorithm-specific macros take only two arguments: the array bounds.

```

403 \def\forest@sort#1#2#3#4#5{%
404   \let\forest@sort@cmp#1\relax
405   \let\forest@sort@let#2\relax

```

<sup>1</sup>In forest, arrays are encoded as families of macros. An array-macro name consists of the (optional, but recommended) prefix, the index, and the (optional) suffix (e.g. `\forest@42x`). Prefix establishes the “namespace”, while using more than one suffix simulates an array of named tuples. The length of the array is stored in macro `\<prefix>n`.

```

406 \let\forest@sort@direction#3\relax
407 \forest@@@sort{#4}{#5}%
408 }

```

The central sorting macro. Here it is decided which sorting algorithm will be used: for arrays at least `\forest@quicksort@minarraylength` long, quicksort is used; otherwise, insertion sort.

```

409 \def\forest@quicksort@minarraylength{10000}
410 \def\forest@@@sort#1#2{%
411   \ifnum#1<#2\relax\@escapeif{%
412     \forest@sort@m=#2
413     \advance\forest@sort@m -#1
414     \ifnum\forest@sort@m>\forest@quicksort@minarraylength\relax\@escapeif{%
415       \forest@quicksort{#1}{#2}%
416     }\else\@escapeif{%
417       \forest@insertionsort{#1}{#2}%
418     }\fi
419   }\fi
420 }

```

Various counters and macros needed by the sorting algorithms.

```

421 \newcount\forest@sort@m\newcount\forest@sort@k\newcount\forest@sort@p
422 \def\forest@sort@ascending{>}
423 \def\forest@sort@descending{<}
424 \def\forest@sort@cmp{%
425   \PackageError{sort}{You must define forest@sort@cmp function before calling
426     sort}{The macro must take two arguments, indices of the array
427     elements to be compared, and return '=' if the elements are equal
428     and '>'/ '<' if the first is greater /less than the secong element.}%
429 }
430 \def\forest@sort@cmp@gt{\def\forest@sort@cmp@result{>}}
431 \def\forest@sort@cmp@lt{\def\forest@sort@cmp@result{<}}
432 \def\forest@sort@cmp@eq{\def\forest@sort@cmp@result{=}}
433 \def\forest@sort@let{%
434   \PackageError{sort}{You must define forest@sort@let function before calling
435     sort}{The macro must take two arguments, indices of the array:
436     element 2 must be copied onto element 1.}%
437 }

```

Quick sort macro (adapted from [laansort](#)).

```

438 \newloop\forest@sort@loop
439 \newloop\forest@sort@loopA
440 \def\forest@quicksort#1#2{%

```

Compute the index of the middle element (`\forest@sort@m`).

```

441   \forest@sort@m=#2
442   \advance\forest@sort@m -#1
443   \ifodd\forest@sort@m\relax\advance\forest@sort@m1 \fi
444   \divide\forest@sort@m 2
445   \advance\forest@sort@m #1

```

The pivot element is the median of the first, the middle and the last element.

```

446   \forest@sort@cmp{#1}{#2}%
447   \if\forest@sort@cmp@result=%
448     \forest@sort@p=#1
449   \else
450     \if\forest@sort@cmp@result>%
451       \forest@sort@p=#1\relax
452     \else
453       \forest@sort@p=#2\relax
454     \fi
455   \forest@sort@cmp{\the\forest@sort@p}{\the\forest@sort@m}%
456   \if\forest@sort@cmp@result<%
457     \else

```

```

458     \forest@sort@p=\the\forest@sort@m
459     \fi
460     \fi
    Exchange the pivot and the first element.
461     \forest@sort@xch{#1}{\the\forest@sort@p}%
    Counter \forest@sort@m will hold the final location of the pivot element.
462     \forest@sort@m=#1\relax
    Loop through the list.
463     \forest@sort@k=#1\relax
464     \forest@sort@loop
465     \ifnum\forest@sort@k<#2\relax
466     \advance\forest@sort@k 1
    Compare the pivot and the current element.
467     \forest@sort@cmp{#1}{\the\forest@sort@k}%
    If the current element is smaller (ascending) or greater (descending) than the pivot element, move it into
    the first part of the list, and adjust the final location of the pivot.
468     \ifx\forest@sort@direction\forest@sort@cmp@result
469     \advance\forest@sort@m 1
470     \forest@sort@xch{\the\forest@sort@m}{\the\forest@sort@k}
471     \fi
472     \forest@sort@repeat
    Move the pivot element into its final position.
473     \forest@sort@xch{#1}{\the\forest@sort@m}%
    Recursively call sort on the two parts of the list: elements before the pivot are smaller (ascending order)
    / greater (descending order) than the pivot; elements after the pivot are greater (ascending order) /
    smaller (descending order) than the pivot.
474     \forest@sort@k=\forest@sort@m
475     \advance\forest@sort@k -1
476     \advance\forest@sort@m 1
477     \edef\forest@sort@marshal{%
478     \noexpand\forest@sort@{#1}{\the\forest@sort@k}%
479     \noexpand\forest@sort@{\the\forest@sort@m}{#2}%
480     }%
481     \forest@sort@marshal
482 }
483 % We defines the item-exchange macro in terms of the (user-provided)
484 % array let macro.
485 % \begin{macrocode}
486 \def\forest@sort@aux{aux}
487 \def\forest@sort@xch#1#2{%
488     \forest@sort@let{\forest@sort@aux}{#1}%
489     \forest@sort@let{#1}{#2}%
490     \forest@sort@let{#2}{\forest@sort@aux}%
491 }
    Insertion sort.
492 \def\forest@insertionsort#1#2{%
493     \forest@sort@m=#1
494     \edef\forest@insertionsort@low{#1}%
495     \forest@sort@loopA
496     \ifnum\forest@sort@m<#2
497     \advance\forest@sort@m 1
498     \forest@insertionsort@Qbody
499     \forest@sort@repeatA
500 }
501 \newif\ifforest@insertionsort@loop
502 \def\forest@insertionsort@Qbody{%

```

```

503 \forest@sort@let{\forest@sort@aux}{\the\forest@sort@m}%
504 \forest@sort@k\forest@sort@m
505 \advance\forest@sort@k -1
506 \forest@insertionsort@looptrue
507 \forest@sort@loop
508 \ifforest@insertionsort@loop
509 \forest@insertionsort@qbody
510 \forest@sort@repeat
511 \advance\forest@sort@k 1
512 \forest@sort@let{\the\forest@sort@k}{\forest@sort@aux}%
513 }
514 \def\forest@insertionsort@qbody{%
515 \forest@sort@cmp{\the\forest@sort@k}{\forest@sort@aux}%
516 \ifx\forest@sort@direction\forest@sort@cmp@result\relax
517 \forest@sort@p=\forest@sort@k
518 \advance\forest@sort@p 1
519 \forest@sort@let{\the\forest@sort@p}{\the\forest@sort@k}%
520 \advance\forest@sort@k -1
521 \ifnum\forest@sort@k<\forest@insertionsort@low\relax
522 \forest@insertionsort@loopfalse
523 \fi
524 \else
525 \forest@insertionsort@loopfalse
526 \fi
527 }

```

Below, several helpers for writing comparison macros are provided. They take two (pairs of) control sequence names and compare their contents.

Compare numbers.

```

528 \def\forest@sort@cmpnumcs#1#2{%
529 \ifnum\csname#1\endcsname>\csname#2\endcsname\relax
530 \forest@sort@cmp@gt
531 \else
532 \ifnum\csname#1\endcsname<\csname#2\endcsname\relax
533 \forest@sort@cmp@lt
534 \else
535 \forest@sort@cmp@eq
536 \fi
537 \fi
538 }

```

Compare dimensions.

```

539 \def\forest@sort@cmpdimcs#1#2{%
540 \ifdim\csname#1\endcsname>\csname#2\endcsname\relax
541 \forest@sort@cmp@gt
542 \else
543 \ifdim\csname#1\endcsname<\csname#2\endcsname\relax
544 \forest@sort@cmp@lt
545 \else
546 \forest@sort@cmp@eq
547 \fi
548 \fi
549 }

```

Compare points (pairs of dimension) (#1,#2) and (#3,#4).

```

550 \def\forest@sort@cmptwodimcs#1#2#3#4{%
551 \ifdim\csname#1\endcsname>\csname#3\endcsname\relax
552 \forest@sort@cmp@gt
553 \else
554 \ifdim\csname#1\endcsname<\csname#3\endcsname\relax
555 \forest@sort@cmp@lt
556 \else

```

```

557 \ifdim\csname#2\endcsname>\csname#4\endcsname\relax
558 \forest@sort@cmp@gt
559 \else
560 \ifdim\csname#2\endcsname<\csname#4\endcsname\relax
561 \forest@sort@cmp@lt
562 \else
563 \forest@sort@cmp@eq
564 \fi
565 \fi
566 \fi
567 \fi
568 }

```

The following macro reverses an array. The arguments: #1 is the array let macro; #2 is the start index (inclusive), and #3 is the end index (exclusive).

```

569 \def\forest@reversearray#1#2#3{%
570 \let\forest@sort@let#1%
571 \c@pgf@countc=#2
572 \c@pgf@countd=#3
573 \advance\c@pgf@countd -1
574 \safeloop
575 \ifnum\c@pgf@countc<\c@pgf@countd\relax
576 \forest@sort@xch{\the\c@pgf@countc}{\the\c@pgf@countd}%
577 \advance\c@pgf@countc 1
578 \advance\c@pgf@countd -1
579 \saferepeat
580 }

```

## 5 The bracket representation parser

### 5.1 The user interface macros

Settings.

```

581 \def\bracketset#1{\pgfqkeys{/bracket}{#1}}%
582 \bracketset{%
583 /bracket/.is family,
584 /handlers/.let/.style={\pgfkeyscurrentpath/.code={\let#1##1}},
585 opening bracket/.let=\bracket@openingBracket,
586 closing bracket/.let=\bracket@closingBracket,
587 action character/.let=\bracket@actionCharacter,
588 opening bracket=[,
589 closing bracket=],
590 action character,
591 new node/.code n args={3}{% #1=preamble, #2=node spec, #3=cs receiving the id
592 \forest@node@new#3%
593 \forest@set#3{given options}{\forest@contentto=\unexpanded{#2}}%
594 \ifblank{#1}{}{%
595 \forestrset{preamble}{#1}%
596 }%
597 },
598 set afterthought/.code 2 args={% #1=node id, #2=afterthought
599 \ifblank{#2}{}{\forest@appto{#1}{given options}{,afterthought={#2}}}%
600 }
601 }

```

`\bracketParse` is the macro that should be called to parse a balanced bracket representation. It takes five parameters: #1 is the code that will be run after parsing the bracket; #2 is a control sequence that will receive the id of the root of the created tree structure. (The bracket representation should follow (after optional spaces), but is not a formal parameter of the macro.)

```
602 \newtoks\bracket@content
```

```

603 \newtoks\bracket@afterthought
604 \def\bracketParse#1#2={%
605   \def\bracketEndParsingHook{#1}%
606   \def\bracket@saveRootNodeTo{#2}%

```

Content and afterthought will be appended to these macros. (The `\bracket@afterthought` toks register is abused for storing the preamble as well — that’s ok, the preamble comes before any afterthoughts.)

```

607 \bracket@content={}%
608 \bracket@afterthought={}%

```

The parser can be in three states: in content (0), in afterthought (1), or starting (2). While in the content/afterthought state, the parser appends all non-control tokens to the content/afterthought macro.

```

609 \let\bracket@state\bracket@state@starting
610 \bracket@ignorespacestrue

```

By default, don’t expand anything.

```

611 \bracket@expandtokensfalse

```

We initialize several control sequences that are used to store some nodes while parsing.

```

612 \def\bracket@parentNode{0}%
613 \def\bracket@rootNode{0}%
614 \def\bracket@newNode{0}%
615 \def\bracket@afterthoughtNode{0}%

```

Finally, we start the parser.

```

616 \bracket@Parse
617 }

```

The other macro that an end user (actually a power user) can use, is actually just a synonym for `\bracket@Parse`. It should be used to resume parsing when the action code has finished its work.

```

618 \def\bracketResume{\bracket@Parse}%

```

## 5.2 Parsing

We first check if the next token is a space. Spaces need special treatment because they are eaten by both the `\romannumeral` trick and T<sub>E</sub>Xs (undelimited) argument parsing algorithm. If a space is found, remember that, eat it up, and restart the parsing.

```

619 \def\bracket@Parse{%
620   \futurelet\bracket@next@token\bracket@Parse@checkForSpace
621 }
622 \def\bracket@Parse@checkForSpace{%
623   \expandafter\ifx\space\bracket@next@token\@escapeif{%
624     \ifbracket@ignorespaces\else
625       \bracket@haveSpacetrue
626       \fi
627     \expandafter\bracket@Parse\romannumeral-‘0%
628   }\else\@escapeif{%
629     \bracket@Parse@maybeexpand
630   }\fi
631 }

```

We either fully expand the next token (using a popular T<sub>E</sub>Xnical trick ...) or don’t expand it at all, depending on the state of `\ifbracket@expandtokens`.

```

632 \newif\ifbracket@expandtokens
633 \def\bracket@Parse@maybeexpand{%
634   \ifbracket@expandtokens\@escapeif{%
635     \expandafter\bracket@Parse@peekAhead\romannumeral-‘0%
636   }\else\@escapeif{%
637     \bracket@Parse@peekAhead
638   }\fi
639 }

```

We then look ahead to see what's coming.

```
640 \def\bracket@Parse@peekAhead{%
641   \futurelet\bracket@next@token\bracket@Parse@checkForTeXGroup
642 }
```

If the next token is a begin-group token, we append the whole group to the content or afterthought macro, depending on the state.

```
643 \def\bracket@Parse@checkForTeXGroup{%
644   \ifx\bracket@next@token\bgroup%
645     \@escapeif{\bracket@Parse@appendGroup}%
646   \else
647     \@escapeif{\bracket@Parse@token}%
648   \fi
649 }
```

This is easy: if a control token is found, run the appropriate macro; otherwise, append the token to the content or afterthought macro, depending on the state.

```
650 \long\def\bracket@Parse@token#1{%
651   \ifx#1\bracket@openingBracket
652     \@escapeif{\bracket@Parse@openingBracketFound}%
653   \else
654     \@escapeif{%
655       \ifx#1\bracket@closingBracket
656         \@escapeif{\bracket@Parse@closingBracketFound}%
657       \else
658         \@escapeif{%
659           \ifx#1\bracket@actionCharacter
660             \@escapeif{\futurelet\bracket@next@token\bracket@Parse@actionCharacterFound}%
661           \else
662             \@escapeif{\bracket@Parse@appendToken#1}%
663           \fi
664         }%
665       \fi
666     }%
667   \fi
668 }
```

Append the token or group to the content or afterthought macro. If a space was found previously, append it as well.

```
669 \newif\ifbracket@haveSpace
670 \newif\ifbracket@ignorespaces
671 \def\bracket@Parse@appendSpace{%
672   \ifbracket@haveSpace
673     \ifcase\bracket@state\relax
674       \eapptotoks\bracket@content\space
675     \or
676       \eapptotoks\bracket@afterthought\space
677     \or
678       \eapptotoks\bracket@afterthought\space
679     \fi
680   \bracket@haveSpacefalse
681 \fi
682 }
683 \long\def\bracket@Parse@appendToken#1{%
684   \bracket@Parse@appendSpace
685   \ifcase\bracket@state\relax
686     \lapptotoks\bracket@content{#1}%
687   \or
688     \lapptotoks\bracket@afterthought{#1}%
689   \or
690     \lapptotoks\bracket@afterthought{#1}%

```

```

691 \fi
692 \bracket@ignorespacesfalse
693 \bracket@Parse
694 }
695 \def\bracket@Parse@appendGroup#1{%
696 \bracket@Parse@appendSpace
697 \ifcase\bracket@state\relax
698 \apptotoks\bracket@content{#{1}}%
699 \or
700 \apptotoks\bracket@afterthought{#{1}}%
701 \or
702 \apptotoks\bracket@afterthought{#{1}}%
703 \fi
704 \bracket@ignorespacesfalse
705 \bracket@Parse
706 }

```

Declare states.

```

707 \def\bracket@state@inContent{0}
708 \def\bracket@state@inAfterthought{1}
709 \def\bracket@state@starting{2}

```

Welcome to the jungle. In the following two macros, new nodes are created, content and afterthought are sent to them, parents and states are changed. . . Altogether, we distinguish six cases, as shown below: in the schemas, we have just crossed the symbol after the dots. (In all cases, we reset the `\if` for spaces.)

```

710 \def\bracket@Parse@openingBracketFound{%
711 \bracket@haveSpacefalse
712 \ifcase\bracket@state\relax% in content [ ... [

```

[...[: we have just finished gathering the content and are about to begin gathering the content of another node. We create a new node (and put the content (...) into it). Then, if there is a parent node, we append the new node to the list of its children. Next, since we have just crossed an opening bracket, we declare the newly created node to be the parent of the coming node. The state does not change. Finally, we continue parsing.

```

713 \escapeif{%
714 \bracket@createNode
715 \ifnum\bracket@parentNode=0 \else
716 \forest@node@Append{\bracket@parentNode}{\bracket@newNode}%
717 \fi
718 \let\bracket@parentNode\bracket@newNode
719 \bracket@Parse
720 }%
721 \or % in afterthought ] ... [

```

]...[: we have just finished gathering the afterthought and are about to begin gathering the content of another node. We add the afterthought (...) to the “afterthought node” and change into the content state. The parent does not change. Finally, we continue parsing.

```

722 \escapeif{%
723 \bracket@addAfterthought
724 \let\bracket@state\bracket@state@inContent
725 \bracket@Parse
726 }%
727 \else % starting

```

{start}...[: we have just started. Nothing to do yet (we couldn’t have collected any content yet), just get into the content state and continue parsing.

```

728 \escapeif{%
729 \let\bracket@state\bracket@state@inContent
730 \bracket@Parse
731 }%
732 \fi
733 }

```



```

734 \def\bracket@Parse@closingBracketFound{%
735   \bracket@haveSpacefalse
736   \ifcase\bracket@state\relax % in content [ ... ]

```

[...]: we have just finished gathering the content of a node and are about to begin gathering its afterthought. We create a new node (and put the content (...) into it). If there is no parent node, we're done with parsing. Otherwise, we set the newly created node to be the "afterthought node", i.e. the node that will receive the next afterthought, change into the afterthought mode, and continue parsing.

```

737   \@escapeif{%
738     \bracket@createNode
739     \ifnum\bracket@parentNode=0
740       \@escapeif\bracket@EndParsingHook
741     \else
742       \@escapeif{%
743         \let\bracket@afterthoughtNode\bracket@newNode
744         \let\bracket@state\bracket@state@inAfterthought
745         \forest@node@Append{\bracket@parentNode}{\bracket@newNode}%
746         \bracket@Parse
747       }%
748     \fi
749   }%
750 \or % in afterthought ] ... ]

```

]...]: we have finished gathering an afterthought of some node and will begin gathering the afterthought of its parent. We first add the afterthought to the afterthought node and set the current parent to be the next afterthought node. We change the parent to the current parent's parent and check if that node is null. If it is, we're done with parsing (ignore the trailing spaces), otherwise we continue.

```

751   \@escapeif{%
752     \bracket@addAfterthought
753     \let\bracket@afterthoughtNode\bracket@parentNode
754     \edef\bracket@parentNode{\forestOve{\bracket@parentNode}{@parent}}%
755     \ifnum\bracket@parentNode=0
756       \expandafter\bracket@EndParsingHook
757     \else
758       \expandafter\bracket@Parse
759     \fi
760   }%
761 \else % starting

```

{start}...]: something's obviously wrong with the input here...

```

762   \PackageError{forest}{You're attempting to start a bracket representation
763     with a closing bracket}{}%
764 \fi
765 }

```

The action character code. What happens is determined by the next token.

```

766 \def\bracket@Parse@actionCharacterFound{%
  If a braced expression follows, its contents will be fully expanded.
767   \ifx\bracket@next@token\bgroup\@escapeif{%
768     \bracket@Parse@action@expandgroup
769   }\else\@escapeif{%
770     \bracket@Parse@action@notagroup
771   }\fi
772 }
773 \def\bracket@Parse@action@expandgroup#1{%
774   \edef\bracket@Parse@action@expandgroup@macro{#1}%
775   \expandafter\bracket@Parse\bracket@Parse@action@expandgroup@macro
776 }
777 \let\bracket@action@fullyexpandCharacter+
778 \let\bracket@action@dontexpandCharacter-
779 \let\bracket@action@executeCharacter!
780 \def\bracket@Parse@action@notagroup#1{%

```

If + follows, tokens will be fully expanded from this point on.

```
781 \ifx#1\bracket@action@fullyexpandCharacter\@escapeif%
782   \bracket@expandtokenstrue\bracket@Parse
783 }\else\@escapeif%
```

If - follows, tokens will not be expanded from this point on. (This is the default behaviour.)

```
784 \ifx#1\bracket@action@dontexpandCharacter\@escapeif%
785   \bracket@expandtokensfalse\bracket@Parse
786 }\else\@escapeif%
```

Inhibit expansion of the next token.

```
787 \ifx#10\@escapeif%
788   \bracket@Parse@appendToken
789 }\else\@escapeif%
```

If another action characted follows, we yield the control. The user is expected to resume the parser manually, using `\bracketResume`.

```
790 \ifx#1\bracket@actionCharacter
791 \else\@escapeif%
```

Anything else will be expanded once.

```
792 \expandafter\bracket@Parse#1%
793 }\fi
794 }\fi
795 }\fi
796 }\fi
797 }
```

### 5.3 The tree-structure interface

This macro creates a new node and sets its content (and preamble, if it's a root node). Bracket user must define a 3-arg key `/bracket/new node=<preamble><node specification><node cs>`. User's key must define `<node cs>` to be a macro holding the node's id.

```
798 \def\bracket@createNode{%
799   \ifnum\bracket@rootNode=0
800     % root node
801     \bracketset{new node/.expanded=%
802       {\the\bracket@afterthought}%
803       {\the\bracket@content}%
804       \noexpand\bracket@newNode
805     }%
806     \bracket@afterthought={}%
807     \let\bracket@rootNode\bracket@newNode
808     \expandafter\let\bracket@saveRootNodeTo\bracket@newNode
809   \else
810     % other nodes
811     \bracketset{new node/.expanded=%
812       }%
813       {\the\bracket@content}%
814       \noexpand\bracket@newNode
815     }%
816   \fi
817   \bracket@content={}%
818 }
```

This macro sets the afterthought. Bracket user must define a 2-arg key `/bracket/set_afterthought=<node id><afterthought>`.

```
819 \def\bracket@addAfterthought{%
820   \bracketset{%
821     set afterthought/.expanded={\bracket@afterthoughtNode}{\the\bracket@afterthought}%
822   }%
823   \bracket@afterthought={}%

```

## 6 Nodes

Nodes have numeric ids. The node option values of node  $n$  are saved in the `\pgfkeys` tree in path `/forest/@node/ $n$` .

### 6.1 Option setting and retrieval

Macros for retrieving/setting node options of the current node.

```

825 % full expansion expands precisely to the value
826 \def\forestov#1{\expandafter\expandafter\expandafter\expandonce
827   \pgfkeysvalueof{/forest/@node/\forest@cn/#1}}
828 % full expansion expands all the way
829 \def\forestove#1{\pgfkeysvalueof{/forest/@node/\forest@cn/#1}}
830 % full expansion expands to the cs holding the value
831 \def\forestom#1{\expandafter\expandonce\expandafter{\pgfkeysvalueof{/forest/@node/\forest@cn/#1}}}
832 \def\forestoget#1#2{\pgfkeysgetvalue{/forest/@node/\forest@cn/#1}{#2}}
833 \def\forestolet#1#2{\pgfkeyslet{/forest/@node/\forest@cn/#1}{#2}}
834 \def\forestocslet#1#2{%
835   \edef\forest@marshal{%
836     \noexpand\pgfkeyslet{/forest/@node/\forest@cn/#1}{\expandonce{\csname#2\endcsname}}}%
837   }\forest@marshal
838 }
839 \def\forestoset#1#2{\pgfkeyssetvalue{/forest/@node/\forest@cn/#1}{#2}}
840 \def\forestoeset#1#2{%
841   \edef\forest@option@temp{%
842     \noexpand\pgfkeyssetvalue{/forest/@node/\forest@cn/#1}{#2}%
843   }\forest@option@temp
844 }
845 \def\forestooppto#1#2{%
846   \forestoeset{#1}{\forestov{#1}\unexpanded{#2}}%
847 }
848 \def\forestoiifdefined#1#2#3{%
849   \pgfkeysifdefined{/forest/@node/\forest@cn/#1}{#2}{#3}%
850 }

```

User macros for retrieving node options of the current node.

```

851 \let\forestoption\forestov
852 \let\foresteooption\forestove

```

Macros for retrieving node options of a node given by its id.

```

853 \def\forest0v#1#2{\expandafter\expandafter\expandafter\expandonce
854   \pgfkeysvalueof{/forest/@node/#1/#2}}
855 \def\forest0ve#1#2{\pgfkeysvalueof{/forest/@node/#1/#2}}
856 % full expansion expands to the cs holding the value
857 \def\forest0m#1#2{\expandafter\expandonce\expandafter{\pgfkeysvalueof{/forest/@node/#1/#2}}}
858 \def\forest0get#1#2#3{\pgfkeysgetvalue{/forest/@node/#1/#2}{#3}}
859 \def\forest0get#1#2#3{\pgfkeysgetvalue{/forest/@node/#1/#2}{#3}}
860 \def\forest0let#1#2#3{\pgfkeyslet{/forest/@node/#1/#2}{#3}}
861 \def\forest0cslet#1#2#3{%
862   \edef\forest@marshal{%
863     \noexpand\pgfkeyslet{/forest/@node/#1/#2}{\expandonce{\csname#3\endcsname}}}%
864   }\forest@marshal
865 }
866 \def\forest0set#1#2#3{\pgfkeyssetvalue{/forest/@node/#1/#2}{#3}}
867 \def\forest0eset#1#2#3{%
868   \edef\forestoption@temp{%
869     \noexpand\pgfkeyssetvalue{/forest/@node/#1/#2}{#3}%
870   }\forestoption@temp

```

```

871 }
872 \def\forest0appto#1#2#3{%
873   \forest0eset{#1}{#2}{\forest0v{#1}{#2}\unexpanded{#3}}%
874 }
875 \def\forest0eappto#1#2#3{%
876   \forest0eset{#1}{#2}{\forest0v{#1}{#2}#3}%
877 }
878 \def\forest0preto#1#2#3{%
879   \forest0eset{#1}{#2}{\unexpanded{#3}\forest0v{#1}{#2}}%
880 }
881 \def\forest0epreto#1#2#3{%
882   \forest0eset{#1}{#2}{#3\forest0v{#1}{#2}}%
883 }
884 \def\forest0ifdefined#1#2#3#4{%
885   \pgfkeysifdefined{/forest/@node/#1/#2}{#3}{#4}%
886 }
887 \def\forest0let0#1#2#3#4{% option #2 of node #1 <-- option #4 of node #3
888   \forest0get{#3}{#4}\forestoption@temp
889   \forest0let{#1}{#2}\forestoption@temp}
890 \def\forest0leto#1#2#3{%
891   \forest0get{#3}\forestoption@temp
892   \forest0let{#1}{#2}\forestoption@temp}
893 \def\forest0let0#1#2#3{%
894   \forest0get{#2}{#3}\forestoption@temp
895   \forest0let{#1}\forestoption@temp}
896 \def\forest0leto#1#2{%
897   \forest0get{#2}\forestoption@temp
898   \forest0let{#1}\forestoption@temp}

```

Macros for retrieving/setting registers.

```

899 % full expansion expands precisely to the value
900 \def\forestrv#1{\expandafter\expandafter\expandafter\expandonce
901   \pgfkeysvalueof{/forest/@node/register/#1}}
902 % full expansion expands all the way
903 \def\forestrve#1{\pgfkeysvalueof{/forest/@node/register/#1}}
904 % full expansion expands to the cs holding the value
905 \def\forestrm#1{\expandafter\expandonce\expandafter{\pgfkeysvalueof{/forest/@node/register/#1}}}
906 \def\forestrget#1#2{\pgfkeysgetvalue{/forest/@node/register/#1}{#2}}
907 \def\forestrlet#1#2{\pgfkeyslet{/forest/@node/register/#1}{#2}}
908 \def\forestrcslet#1#2{%
909   \edef\forest@marshal{%
910     \noexpand\pgfkeyslet{/forest/@node/register/#1}{\expandonce{\csname#2\endcsname}}%
911   }\forest@marshal
912 }
913 \def\forestrset#1#2{\pgfkeyssetvalue{/forest/@node/register/#1}{#2}}
914 \def\forestreset#1#2{%
915   \edef\forest@option@temp{%
916     \noexpand\pgfkeyssetvalue{/forest/@node/register/#1}{#2}%
917   }\forest@option@temp
918 }
919 \def\forestrappto#1#2{%
920   \forestreset{#1}{\forestrv{#1}\unexpanded{#2}}%
921 }
922 \def\forestrpreto#1#2{%
923   \forestreset{#1}{\unexpanded{#2}\forestrv{#1}}%
924 }
925 \def\forestrifdefined#1#2#3{%
926   \pgfkeysifdefined{/forest/@node/register/#1}{#2}{#3}%
927 }

```

User macros for retrieving node options of the current node.

```

928 \def\forestregister#1{\forestrv{#1}}

```

```

929 \def\forestregister#1{\forestrve{#1}}
Node initialization. Node option declarations append to \forest@node@init.
930 \def\forest@node@init{%
931   \forestoset{@parent}{0}%
932   \forestoset{@previous}{0}% previous sibling
933   \forestoset{@next}{0}% next sibling
934   \forestoset{@first}{0}% primary child
935   \forestoset{@last}{0}% last child
936 }
937 \def\forestoinit#1{%
938   \pgfkeysgetvalue{/forest/#1}\forestoinit@temp
939   \forestolet{#1}\forestoinit@temp
940 }
941 \newcount\forest@node@maxid
942 \def\forest@node@new#1{% #1 = cs receiving the new node id
943   \advance\forest@node@maxid1
944   \forest@fornode{\the\forest@node@maxid}{%
945     \forest@node@init
946     \forest@node@setname{node@\forest@cn}%
947     \forest@initializefromstandardnode
948     \edef#1{\forest@cn}%
949   }%
950 }
951 \let\forestoinit@orig\forestoinit
952 \def\forest@node@copy#1#2{% #1=from node id, cs receiving the new node id
953   \advance\forest@node@maxid1
954   \def\forestoinit##1{\ifstrequal{##1}{name}{\forestoset{name}{node@\forest@cn}}{\forestolet0{##1}{#1}{##1}}}
955   \forest@fornode{\the\forest@node@maxid}{%
956     \forest@node@init
957     \forest@node@setname{\forest@copy@name@template{\forest@ve{#1}{name}}}%
958     \edef#2{\forest@cn}%
959   }%
960   \let\forestoinit\forestoinit@orig
961 }
962 \forestset{
963   copy name template/.code={\def\forest@copy@name@template##1{#1}},
964   copy name template/.default={node@\the\forest@node@maxid},
965   copy name template
966 }
967 \def\forest@tree@copy#1#2{% #1=from node id, #2=cs receiving the new node id
968   \forest@node@copy{#1}\forest@node@copy@temp@id
969   \forest@fornode{\forest@node@copy@temp@id}{%
970     \expandafter\forest@tree@copy\expandafter{\forest@node@copy@temp@id}{#1}%
971     \edef#2{\forest@cn}%
972   }%
973 }
974 \def\forest@tree@copy@#1#2{%
975   \forest@node@Foreachchild{#2}{%
976     \expandafter\forest@tree@copy\expandafter{\forest@cn}\forest@node@copy@temp@childid
977     \forest@node@Append{#1}{\forest@node@copy@temp@childid}%
978   }%
979 }
Macro \forest@cn holds the current node id (a number). Node 0 is a special “null” node which is used
to signal the absence of a node.
980 \def\forest@cn{0}
981 \forest@node@init

```

## 6.2 Tree structure

Node insertion/removal.

For the lowercase variants, `\forest@cn` is the parent/removed node. For the uppercase variants, `#1` is the parent/removed node. For efficiency, the public macros all expand the arguments before calling the internal macros.

```

982 \def\forest@node@append#1{\expandtwonumberargs\forest@node@Append{\forest@cn}{#1}}
983 \def\forest@node@prepend#1{\expandtwonumberargs\forest@node@Insertafter{\forest@cn}{#1}{0}}
984 \def\forest@node@insertafter#1#2{%
985   \expandthreenumberargs\forest@node@Insertafter{\forest@cn}{#1}{#2}}
986 \def\forest@node@insertbefore#1#2{%
987   \expandthreenumberargs\forest@node@Insertafter{\forest@cn}{#1}{\forestOve{#2}{@previous}}%
988 }
989 \def\forest@node@remove{\expandnumberarg\forest@node@Remove{\forest@cn}}
990 \def\forest@node@Append#1#2{\expandtwonumberargs\forest@node@Append@{#1}{#2}}
991 \def\forest@node@Prepend#1#2{\expandtwonumberargs\forest@node@Insertafter{#1}{#2}{0}}
992 \def\forest@node@Insertafter#1#2#3{% #2 is inserted after #3
993   \expandthreenumberargs\forest@node@Insertafter@{#1}{#2}{#3}%
994 }
995 \def\forest@node@Insertbefore#1#2#3{% #2 is inserted before #3
996   \expandthreenumberargs\forest@node@Insertafter{#1}{#2}{\forestOve{#3}{@previous}}%
997 }
998 \def\forest@node@Remove#1{\expandnumberarg\forest@node@Remove@{#1}}
999 \def\forest@node@Insertafter@#1#2#3{%
1000   \ifnum\forestOve{#2}{@parent}=0
1001   \else
1002     \PackageError{forest}{Insertafter(#1,#2,#3):
1003       node #2 already has a parent (\forestOve{#2}{@parent})}{}%
1004   \fi
1005   \ifnum#3=0
1006   \else
1007     \ifnum#1=\forestOve{#3}{@parent}
1008     \else
1009       \PackageError{forest}{Insertafter(#1,#2,#3): node #1 is not the parent of the
1010         intended sibling #3 (with parent \forestOve{#3}{@parent})}{}%
1011     \fi
1012     \fi
1013     \forestOset{#2}{@parent}{#1}%
1014     \forestOset{#2}{@previous}{#3}%
1015     \ifnum#3=0
1016       \forestOget{#1}{@first}\forest@node@temp
1017       \forestOset{#1}{@first}{#2}%
1018     \else
1019       \forestOget{#3}{@next}\forest@node@temp
1020       \forestOset{#3}{@next}{#2}%
1021     \fi
1022     \forestOset{#2}{@next}{\forest@node@temp}%
1023     \ifnum\forest@node@temp=0
1024       \forestOset{#1}{@last}{#2}%
1025     \else
1026       \forestOset{\forest@node@temp}{@previous}{#2}%
1027     \fi
1028 }
1029 \def\forest@node@Append@#1#2{%
1030   \ifnum\forestOve{#2}{@parent}=0
1031   \else
1032     \PackageError{forest}{Append(#1,#2):
1033       node #2 already has a parent (\forestOve{#2}{@parent})}{}%
1034   \fi
1035   \forestOset{#2}{@parent}{#1}%

```

```

1036 \forestOget{#1}{@last}\forest@node@temp
1037 \forestOeset{#1}{@last}{#2}%
1038 \forestOeset{#2}{@previous}\forest@node@temp}%
1039 \ifnum\forest@node@temp=0
1040 \forestOeset{#1}{@first}{#2}%
1041 \else
1042 \forestOeset{\forest@node@temp}{@next}{#2}%
1043 \fi
1044 }
1045 \def\forest@node@Remove@#1{%
1046 \forestOget{#1}{@parent}\forest@node@temp@parent
1047 \ifnum\forest@node@temp@parent=0
1048 \else
1049 \forestOget{#1}{@previous}\forest@node@temp@previous
1050 \forestOget{#1}{@next}\forest@node@temp@next
1051 \ifnum\forest@node@temp@previous=0
1052 \forestOeset{\forest@node@temp@parent}{@first}{\forest@node@temp@next}%
1053 \else
1054 \forestOeset{\forest@node@temp@previous}{@next}{\forest@node@temp@next}%
1055 \fi
1056 \ifnum\forest@node@temp@next=0
1057 \forestOeset{\forest@node@temp@parent}{@last}{\forest@node@temp@previous}%
1058 \else
1059 \forestOeset{\forest@node@temp@next}{@previous}{\forest@node@temp@previous}%
1060 \fi
1061 \forestOset{#1}{@parent}{0}%
1062 \forestOset{#1}{@previous}{0}%
1063 \forestOset{#1}{@next}{0}%
1064 \fi
1065 }

```

Do some stuff and return to the current node.

```

1066 \def\forest@forthis#1{%
1067 \edef\forest@node@marshal{\unexpanded{#1}\def\noexpand\forest@cn}%
1068 \expandafter\forest@node@marshal\expandafter{\forest@cn}%
1069 }
1070 \def\forest@fornode#1#2{%
1071 \edef\forest@node@marshal{\edef\noexpand\forest@cn{#1}\unexpanded{#2}\def\noexpand\forest@cn}%
1072 \expandafter\forest@node@marshal\expandafter{\forest@cn}%
1073 }

```

Looping methods: children.

```

1074 \def\forest@node@foreachchild#1{\forest@node@Foreachchild{\forest@cn}{#1}}
1075 \def\forest@node@Foreachchild#1#2{%
1076 \forest@fornode{\forestOve{#1}{@first}}{\forest@node@@forselfandfollowingsiblings{#2}}%
1077 }
1078 \def\forest@node@@forselfandfollowingsiblings#1{%
1079 \ifnum\forest@cn=0
1080 \else
1081 \forest@forthis{#1}%
1082 \@escapeif{%
1083 \edef\forest@cn{\forestove{@next}}%
1084 \forest@node@@forselfandfollowingsiblings{#1}%
1085 }%
1086 \fi
1087 }
1088 \def\forest@node@@forselfandfollowingsiblings@reversed#1{%
1089 \ifnum\forest@cn=0
1090 \else
1091 \@escapeif{%
1092 \edef\forest@marshal{%
1093 \noexpand\def\noexpand\forest@cn{\forestove{@next}}%

```

```

1094     \noexpand\forest@node@@forselfandfollowingsiblings@reversed{\unexpanded{#1}}%
1095     \noexpand\forest@fornode{\forest@cn}{\unexpanded{#1}}%
1096   }\forest@marshal
1097 }%
1098 \fi
1099 }
1100 \def\forest@node@foreachchild@reversed#1{\forest@node@Foreachchild@reversed{\forest@cn}{#1}}
1101 \def\forest@node@Foreachchild@reversed#1#2{%
1102   \forest@fornode{\forest@ve{#1}{@last}}{\forest@node@@forselfandprecedingsiblings@reversed{#2}}%
1103 }
1104 \def\forest@node@@forselfandprecedingsiblings@reversed#1{%
1105   \ifnum\forest@cn=0
1106   \else
1107     \forest@forthis{#1}%
1108     \@escapeif{%
1109       \edef\forest@cn{\forest@ve{@previous}}%
1110       \forest@node@@forselfandprecedingsiblings@reversed{#1}%
1111     }%
1112   \fi
1113 }
1114 \def\forest@node@@forselfandprecedingsiblings#1{%
1115   \ifnum\forest@cn=0
1116   \else
1117     \@escapeif{%
1118       \edef\forest@marshal{%
1119         \noexpand\def\noexpand\forest@cn{\forest@ve{@previous}}%
1120         \noexpand\forest@node@@forselfandprecedingsiblings{\unexpanded{#1}}%
1121         \noexpand\forest@fornode{\forest@cn}{\unexpanded{#1}}%
1122       }\forest@marshal
1123     }%
1124   \fi
1125 }

```

Looping methods: (sub)tree and descendants.

```

1126 \def\forest@node@@foreach#1#2#3#4{%
1127   % #1 = do what
1128   % #2 = do that -1=before,1=after processing children
1129   % #3 & #4: normal or reversed order of children?
1130   % #3 = @first/@last
1131   % #4 = \forest@node@@forselfandfollowingsiblings / \forest@node@@forselfandprecedingsiblings@reversed
1132   \ifnum#2<0 \forest@forthis{#1}\fi
1133   \ifnum\forest@ve{#3}=0
1134   \else\@escapeif{%
1135     \forest@forthis{%
1136       \edef\forest@cn{\forest@ve{#3}}%
1137       #4{\forest@node@@foreach{#1}{#2}{#3}{#4}}%
1138     }%
1139   }\fi
1140   \ifnum#2>0 \forest@forthis{#1}\fi
1141 }
1142 \def\forest@node@foreach#1{%
1143   \forest@node@@foreach{#1}{-1}{@first}{\forest@node@@forselfandfollowingsiblings}}
1144 \def\forest@node@Foreach#1#2{%
1145   \forest@fornode{#1}{\forest@node@@foreach{#2}{-1}{@first}{\forest@node@@forselfandfollowingsiblings}}}
1146 \def\forest@node@foreach@reversed#1{%
1147   \forest@node@@foreach{#1}{-1}{@last}{\forest@node@@forselfandprecedingsiblings@reversed}}
1148 \def\forest@node@Foreach@reversed#1#2{%
1149   \forest@fornode{#1}{\forest@node@@foreach{#2}{-1}{@last}{\forest@node@@forselfandprecedingsiblings@reversed}}}
1150 \def\forest@node@foreach@childrenfirst#1{%
1151   \forest@node@@foreach{#1}{1}{@first}{\forest@node@@forselfandfollowingsiblings}}
1152 \def\forest@node@Foreach@childrenfirst#1#2{%

```



```

1153 \forest@fornode{#1}{\forest@node@@foreach{#2}{1}{@first}{\forest@node@@forselfandfollowingsiblings}}
1154 \def\forest@node@foreach@childrenfirst@reversed#1{%
1155 \forest@node@@foreach{#1}{1}{@last}{\forest@node@@forselfandprecedingsiblings@reversed}}
1156 \def\forest@node@Foreach@childrenfirst@reversed#1#2{%
1157 \forest@fornode{#1}{\forest@node@@foreach{#2}{1}{@last}{\forest@node@@forselfandprecedingsiblings@reversed}}
1158 \def\forest@node@foreachdescendant#1{%
1159 \forest@node@foreachchild{\forest@node@@foreach{#1}{-1}{@first}{\forest@node@@forselfandfollowingsiblings}}
1160 \def\forest@node@Foreachdescendant#1#2{%
1161 \forest@node@Foreachchild{#1}{\forest@node@@foreach{#2}{-1}{@first}{\forest@node@@forselfandfollowingsiblings}}
1162 \def\forest@node@foreachdescendant@reversed#1{%
1163 \forest@node@foreachchild@reversed{\forest@node@@foreach{#1}{-1}{@last}{\forest@node@@forselfandprecedingsiblings}}
1164 \def\forest@node@Foreachdescendant@reversed#1#2{%
1165 \forest@node@Foreachchild@reversed{#1}{\forest@node@@foreach{#2}{-1}{@last}{\forest@node@@forselfandprecedingsiblings}}
1166 \def\forest@node@foreachdescendant@childrenfirst#1{%
1167 \forest@node@foreachchild{\forest@node@@foreach{#1}{1}{@first}{\forest@node@@forselfandfollowingsiblings}}
1168 \def\forest@node@Foreachdescendant@childrenfirst#1#2{%
1169 \forest@node@Foreachchild{#1}{\forest@node@@foreach{#2}{1}{@first}{\forest@node@@forselfandfollowingsiblings}}
1170 \def\forest@node@foreachdescendant@childrenfirst@reversed#1{%
1171 \forest@node@foreachchild@reversed{\forest@node@@foreach{#1}{1}{@last}{\forest@node@@forselfandprecedingsiblings}}
1172 \def\forest@node@Foreachdescendant@childrenfirst@reversed#1#2{%
1173 \forest@node@Foreachchild@reversed{#1}{\forest@node@@foreach{#2}{1}{@last}{\forest@node@@forselfandprecedingsiblings}}

```

Looping methods: breadth-first.

```

1174 \def\forest@node@foreach@breadthfirst#1#2{% #1 = max level, #2 = code
1175 \forest@node@Foreach@breadthfirst@{\forest@cn}{@first}{@next}{#1}{#2}}
1176 \def\forest@node@foreach@breadthfirst@reversed#1#2{% #1 = max level, #2 = code
1177 \forest@node@Foreach@breadthfirst@{\forest@cn}{@last}{@previous}{#1}{#2}}
1178 \def\forest@node@Foreach@breadthfirst#1#2#3{% #1 = node id, #2 = max level, #3 = code
1179 \forest@node@Foreach@breadthfirst@{#1}{@first}{@next}{#2}{#3}}
1180 \def\forest@node@Foreach@breadthfirst@reversed#1#2#3{% #1 = node id, #2 = max level, #3 = code
1181 \forest@node@Foreach@breadthfirst@{#1}{@last}{@previous}{#2}{#3}}
1182 \def\forest@node@Foreach@breadthfirst@#1#2#3#4#5{%
1183 % #1 = root node,
1184 % #2 = @first/@last, #3 = @next/@previous (must be in sync with #2),
1185 % #4 = max level (< 0 means infinite)
1186 % #5 = code to execute at each node
1187 \forest@node@Foreach@breadthfirst@processqueue{#1,}{#2}{#3}{#4}{#5}%
1188 }
1189 \def\forest@node@Foreach@breadthfirst@processqueue#1#2#3#4#5{%
1190 % #1 = queue,
1191 % #2 = @first/@last, #3 = @next/@previous (must be in sync with #2),
1192 % #4 = max level (< 0 means infinite)
1193 % #5 = code to execute at each node
1194 \ifstrempy{#1}{-}{%
1195 \forest@node@Foreach@breadthfirst@processqueue@#1\forest@node@Foreach@breadthfirst@processqueue@
1196 {#2}{#3}{#4}{#5}%
1197 }%
1198 }
1199 \def\forest@node@Foreach@breadthfirst@processqueue@#1,#2\forest@node@Foreach@breadthfirst@processqueue@#3#4#5{%
1200 % #1 = first,
1201 % #2 = rest,
1202 % #3 = @first/@last, #4 = next/previous (must be in sync with #2),
1203 % #5 = max level (< 0 means infinite)
1204 % #6 = code to execute at each node
1205 \forest@fornode{#1}{%
1206 #6%
1207 \ifnum#5<0
1208 \forest@node@getlistofchildren\forest@temp{#3}{#4}%
1209 \else
1210 \ifnum\forest@tove{level}>#5\relax
1211 \def\forest@temp{#4}%

```

```

1212     \else
1213     \forest@node@getlistofchildren\forest@temp{#3}{#4}%
1214     \fi
1215     \fi
1216     \edef\forest@marshal{%
1217     \noexpand\forest@node@Foreach@breadthfirst@processqueue{\unexpanded{#2}\forest@temp}%
1218     {#3}{#4}{#5}{\unexpanded{#6}}%
1219     }\forest@marshal
1220 }%
1221 }
1222 \def\forest@node@getlistofchildren#1#2#3{% #1 = list cs, #2 = @first/@last, #3 = @next/@previous
1223 \forest@node@Getlistofchildren{\forest@cn}{#1}{#2}{#3}%
1224 }
1225 \def\forest@node@Getlistofchildren#1#2#3#4{% #1 = node, #2 = list cs, #3 = @first/@last, #4 = @next/@previous
1226 \def#2{}%
1227 \ifnum\forestove{#3}=0
1228 \else
1229 \eappto#2{\forestOve{#1}{#3},}%
1230 \@escapeif{%
1231 \edef\forest@marshal{%
1232 \noexpand\forest@node@Getlistofchildren@{\forestOve{#1}{#3}}\noexpand#2{#4}%
1233 }\forest@marshal
1234 }%
1235 \fi
1236 }
1237 \def\forest@node@Getlistofchildren@#1#2#3{% #1 = node, #2 = list cs, #3 = @next/@previous
1238 \ifnum\forestOve{#1}{#3}=0
1239 \else
1240 \eappto#2{\forestOve{#1}{#3},}%
1241 \@escapeif{%
1242 \edef\forest@marshal{%
1243 \noexpand\forest@node@Getlistofchildren@{\forestOve{#1}{#3}}\noexpand#2{#3}%
1244 }\forest@marshal
1245 }%
1246 \fi
1247 }

Compute n, n', n children and level.
1248 \def\forest@node@Compute@numeric@ts@info@#1{%
1249 \forest@node@Foreach{#1}{\forest@node@@compute@numeric@ts@info}%
1250 \ifnum\forestOve{#1}{@parent}=0
1251 \else
1252 \forest@fornode{#1}{\forest@node@@compute@numeric@ts@info@nbar}%
1253 % hack: the parent of the node we called the update for gets +1 for n_children
1254 \edef\forest@node@temp{\forestOve{#1}{@parent}}%
1255 \forestOset{\forest@node@temp}{n children}%
1256 \number\numexpr\forestOve{\forest@node@temp}{n children}-1%
1257 }%
1258 \fi
1259 \forest@node@Foreachdescendant{#1}{\forest@node@@compute@numeric@ts@info@nbar}%
1260 }
1261 \def\forest@node@@compute@numeric@ts@info{%
1262 \forestoset{n children}{0}%
1263 %
1264 \edef\forest@node@temp{\forestove{@previous}}%
1265 \ifnum\forest@node@temp=0
1266 \forestoset{n}{1}%
1267 \else
1268 \forestoeset{n}{\number\numexpr\forestOve{\forest@node@temp}{n}+1}%
1269 \fi
1270 %

```

```

1271 \edef\forest@node@temp{\forestove{@parent}}%
1272 \ifnum\forest@node@temp=0
1273   \forestoset{n}{0}%
1274   \forestoset{n'}{0}%
1275   \forestoset{level}{0}%
1276 \else
1277   \forestOset{\forest@node@temp}{n children}{%
1278     \number\numexpr\forestOve{\forest@node@temp}{n children}+1%
1279   }%
1280   \forestoeset{level}{%
1281     \number\numexpr\forestOve{\forest@node@temp}{level}+1%
1282   }%
1283 \fi
1284 }
1285 \def\forest@node@@compute@numeric@ts@info@nbar{%
1286   \forestoeset{n'}{\number\numexpr\forestOve{\forestove{@parent}}{n children}-\forestove{n}+1}%
1287 }
1288 \def\forest@node@compute@numeric@ts@info#1{%
1289   \expandnumberarg\forest@node@Compute@numeric@ts@info@{\forest@cn}%
1290 }
1291 \def\forest@node@Compute@numeric@ts@info#1{%
1292   \expandnumberarg\forest@node@Compute@numeric@ts@info@{#1}%
1293 }

    Tree structure queries.
1294 \def\forest@node@rootid{%
1295   \expandnumberarg\forest@node@Rootid{\forest@cn}%
1296 }
1297 \def\forest@node@Rootid#1{% #1=node
1298   \ifnum\forestOve{#1}{@parent}=0
1299     #1%
1300   \else
1301     \@escapeif{\expandnumberarg\forest@node@Rootid{\forestOve{#1}{@parent}}}%
1302   \fi
1303 }
1304 \def\forest@node@nthchildid#1{% #1=n
1305   \ifnum#1<1
1306     0%
1307   \else
1308     \expandnumberarg\forest@node@nthchildid@{\number\forestove{@first}}{#1}%
1309   \fi
1310 }
1311 \def\forest@node@nthchildid@#1#2{%
1312   \ifnum#1=0
1313     0%
1314   \else
1315     \ifnum#2>1
1316       \@escapeifif{\expandtwonumberargs
1317         \forest@node@nthchildid@{\forestOve{#1}{@next}}{\numexpr#2-1}}%
1318     \else
1319       #1%
1320     \fi
1321   \fi
1322 }
1323 \def\forest@node@nbarthchildid#1{% #1=n
1324   \expandnumberarg\forest@node@nbarthchildid@{\number\forestove{@last}}{#1}%
1325 }
1326 \def\forest@node@nbarthchildid@#1#2{%
1327   \ifnum#1=0
1328     0%
1329   \else

```

```

1330 \ifnum#2>1
1331 \escapeiffif{\expandtwonumberargs
1332 \forest@node@nbarthchildid@{\forestOve{#1}{@previous}}{\numexpr#2-1}}%
1333 \else
1334 #1%
1335 \fi
1336 \fi
1337 }
1338 \def\forest@node@nornbarthchildid#1{%
1339 \ifnum#1>0
1340 \forest@node@nthchildid{#1}%
1341 \else
1342 \ifnum#1<0
1343 \forest@node@nbarthchildid{-#1}%
1344 \else
1345 \forest@node@nornbarthchildid@error
1346 \fi
1347 \fi
1348 }
1349 \def\forest@node@nornbarthchildid@error{%
1350 \PackageError{forest}{In \string\forest@node@nornbarthchildid, n should !=0}{}%
1351 }
1352 \def\forest@node@previousleafid{%
1353 \expandnumberarg\forest@node@Previousleafid{\forest@cn}%
1354 }
1355 \def\forest@node@Previousleafid#1{%
1356 \ifnum\forestOve{#1}{@previous}=0
1357 \escapeiff{\expandnumberarg\forest@node@previousleafid@Goup{#1}}%
1358 \else
1359 \expandnumberarg\forest@node@previousleafid@Godown{\forestOve{#1}{@previous}}%
1360 \fi
1361 }
1362 \def\forest@node@previousleafid@Goup#1{%
1363 \ifnum\forestOve{#1}{@parent}=0
1364 \PackageError{forest}{get previous leaf: this is the first leaf}{}%
1365 \else
1366 \escapeiff{\expandnumberarg\forest@node@Previousleafid{\forestOve{#1}{@parent}}}%
1367 \fi
1368 }
1369 \def\forest@node@previousleafid@Godown#1{%
1370 \ifnum\forestOve{#1}{@last}=0
1371 #1%
1372 \else
1373 \escapeiff{\expandnumberarg\forest@node@previousleafid@Godown{\forestOve{#1}{@last}}}%
1374 \fi
1375 }
1376 \def\forest@node@nextleafid{%
1377 \expandnumberarg\forest@node@Nextleafid{\forest@cn}%
1378 }
1379 \def\forest@node@Nextleafid#1{%
1380 \ifnum\forestOve{#1}{@next}=0
1381 \escapeiff{\expandnumberarg\forest@node@nextleafid@Goup{#1}}%
1382 \else
1383 \expandnumberarg\forest@node@nextleafid@Godown{\forestOve{#1}{@next}}%
1384 \fi
1385 }
1386 \def\forest@node@nextleafid@Goup#1{%
1387 \ifnum\forestOve{#1}{@parent}=0
1388 \PackageError{forest}{get next leaf: this is the last leaf}{}%
1389 \else
1390 \escapeiff{\expandnumberarg\forest@node@Nextleafid{\forestOve{#1}{@parent}}}%

```

```

1391 \fi
1392 }
1393 \def\forest@node@nextleafid@Gdown#1{%
1394 \ifnum\forestOve{#1}{@first}=0
1395 #1%
1396 \else
1397 \@escapeif{\expandnumberarg\forest@node@nextleafid@Gdown{\forestOve{#1}{@first}}}%
1398 \fi
1399 }
1400
1401
1402
1403 \def\forest@node@linearnextid{%
1404 \ifnum\forestove{@first}=0
1405 \expandafter\forest@node@linearnextnotdescendantid
1406 \else
1407 \forestove{@first}%
1408 \fi
1409 }
1410 \def\forest@node@linearnextnotdescendantid{%
1411 \expandnumberarg\forest@node@Linearnextnotdescendantid{\forest@cn}%
1412 }
1413 \def\forest@node@Linearnextnotdescendantid#1{%
1414 \ifnum\forestOve{#1}{@next}=0
1415 \ifnum\forestOve{#1}{@parent}=0
1416 0%
1417 \else
1418 \@escapeif{\expandnumberarg\forest@node@Linearnextnotdescendantid{\forestOve{#1}{@parent}}}%
1419 \fi
1420 \else
1421 \forestOve{#1}{@next}%
1422 \fi
1423 }
1424
1425
1426 \def\forest@node@linearpreviousid{%
1427 \ifnum\forestove{@previous}=0
1428 \forestove{@parent}%
1429 \else
1430 \forest@node@previousleafid
1431 \fi
1432 }
1433 \def\forest@ifancestorof#1{% is the current node an ancestor of #1? Yes: #2, no: #3
1434 \expandnumberarg\forest@ifancestorof@{\forestOve{#1}{@parent}}%
1435 }
1436 \def\forest@ifancestorof@#1#2#3{%
1437 \ifnum#1=0
1438 \def\forest@ifancestorof@next{\@secondoftwo}%
1439 \else
1440 \ifnum\forest@cn=#1
1441 \def\forest@ifancestorof@next{\@firstoftwo}%
1442 \else
1443 \def\forest@ifancestorof@next{\expandnumberarg\forest@ifancestorof@{\forestOve{#1}{@parent}}}%
1444 \fi
1445 \fi
1446 \forest@ifancestorof@next{#2}{#3}%
1447 }

```

## 6.3 Node options

### 6.3.1 Option-declaration mechanism

Common code for declaring options.

```
1448 \def\forest@declarehandler#1#2#3{%#1=handler for specific type,#2=option name,#3=default value
1449   \pgfkeyssetvalue{/forest/#2}{#3}%
1450   \appto\forest@node@init{\forest@init{#2}}%
1451   \pgfkeyssetvalue{/forest/#2/node@or@reg}{\forest@cn}%
1452   \forest@convert@others@to@underscores{#2}\forest@pgfmathoptionname
1453   \edef\forest@marshal{%
1454     \noexpand#1{/forest/#2}{/forest}{#2}{\forest@pgfmathoptionname}%
1455   }\forest@marshal
1456 }
1457 \def\forest@def@with@pgfeov#1#2{% \pgfeov mustn't occur in the arg of the .code handler!!!
1458   \long\def#1##1\pgfeov{#2}%
1459 }
```

Option-declaration handlers.

```
1460 \def\forest@declaretokens@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
1461   \forest@declaretokens@handler@A{#1}{#2}{#3}{#4}{}%
1462 }
1463 \def\forest@declarekeylist@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
1464   \forest@declaretokens@handler@A{#1}{#2}{#3}{#4}{,}%
1465   \forest@copycommandkey{#1}{#1}'%
1466   \pgfkeyssetvalue{#1'/option@name}{#3}%
1467   \forest@copycommandkey{#1+}{#1}'%
1468   \pgfkeysalso{#1-/.code={%
1469     \forest@fornode{\forest@setter@node}{%
1470       \forest@node@removekeysfromkeylist{##1}{#3}%
1471     }}%
1472   \pgfkeyssetvalue{#1-/.option@name}{#3}%
1473 }
1474 \def\forest@declaretokens@handler@A#1#2#3#4#5{% #1=key,#2=path,#3=name,#4=pgfmathname,#5=infix
1475   \pgfkeysalso{%
1476     #1/.code={\forest@set{\forest@setter@node}{#3}{##1}},
1477     #1+/.code={\forest@appto{\forest@setter@node}{#3}{#5##1}},
1478     #2/+#3/.code={\forest@preto{\forest@setter@node}{#3}{##1#5}},
1479     #2/if #3/.code n args={3}{%
1480       \forest@toget{#3}\forest@temp@option@value
1481       \edef\forest@temp@compared@value{\unexpanded{##1}}%
1482       \ifx\forest@temp@option@value\forest@temp@compared@value
1483         \pgfkeysalso{##2}%
1484       \else
1485         \pgfkeysalso{##3}%
1486       \fi
1487     },
1488     #2/if in #3/.code n args={3}{%
1489       \forest@toget{#3}\forest@temp@option@value
1490       \edef\forest@temp@compared@value{\unexpanded{##1}}%
1491       \expandafter\expandafter\expandafter\pgfutil@in@\expandafter\expandafter\expandafter{\expandafter\forest@
1492       \ifpgfutil@in@
1493         \pgfkeysalso{##2}%
1494       \else
1495         \pgfkeysalso{##3}%
1496       \fi
1497     },
1498     #2/where #3/.style n args={3}{for tree={#2/if #3={##1}{##2}{##3}}},
1499     #2/where in #3/.style n args={3}{for tree={#2/if in #3={##1}{##2}{##3}}}
1500   }%
1501   \pgfkeyssetvalue{#1/option@name}{#3}%
1502   \pgfkeyssetvalue{#1+/.option@name}{#3}%

```

```

1503 \pgfkeyssetvalue{#2/+#3/option@name}{#3}%
1504 \pgfmathdeclarefunction{#4}{1}{\forest@pgfmathhelper@attribute@toks{##1}{#3}}%
1505 }
1506 \def\forest@declareautowrappedtoks@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname,#5=infix
1507 \forest@declaretoks@handler{#1}{#2}{#3}{#4}%
1508 \forest@copycommandkey{#1}{#1'}%
1509 \pgfkeysalso{#1/.style={#1'/.wrap value={##1}}}%
1510 \pgfkeyssetvalue{#1'/option@name}{#3}%
1511 \forest@copycommandkey{#1+}{#1+'}%
1512 \pgfkeysalso{#1+/.style={#1+'/.wrap value={##1}}}%
1513 \pgfkeyssetvalue{#1+'/option@name}{#3}%
1514 \forest@copycommandkey{#2/+#3}{#2/+#3'}%
1515 \pgfkeysalso{#2/+#3/.style={#2/+#3'/.wrap value={##1}}}%
1516 \pgfkeyssetvalue{#2/+#3'/option@name}{#3}%
1517 }
1518 \def\forest@declarereadonlydimen@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
1519 \pgfkeysalso{%
1520 #2/if #3/.code n args={3}{%
1521 \forestoget{#3}\forest@temp@option@value
1522 \ifdim\forest@temp@option@value=##1\relax
1523 \pgfkeysalso{##2}%
1524 \else
1525 \pgfkeysalso{##3}%
1526 \fi
1527 },
1528 #2/where #3/.style n args={3}{for tree={#2/if #3={##1}{##2}{##3}}},
1529 }%
1530 \pgfmathdeclarefunction{#4}{1}{\forest@pgfmathhelper@attribute@dimen{##1}{#3}}%
1531 }
1532 \def\forest@declaredimensions@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
1533 \forest@declarereadonlydimen@handler{#1}{#2}{#3}{#4}%
1534 \pgfkeysalso{%
1535 #1/.code={%
1536 \pgfmathsetlengthmacro\forest@temp{##1}%
1537 \forest@let{\forest@setter@node}{#3}\forest@temp
1538 },
1539 #1+/.code={%
1540 \pgfmathsetlengthmacro\forest@temp{##1}%
1541 \pgfutil@tempdima=\forestove{#3}
1542 \advance\pgfutil@tempdima\forest@temp\relax
1543 \forest@set{\forest@setter@node}{#3}{\the\pgfutil@tempdima}%
1544 },
1545 #1-/.code={%
1546 \pgfmathsetlengthmacro\forest@temp{##1}%
1547 \pgfutil@tempdima=\forestove{#3}
1548 \advance\pgfutil@tempdima-\forest@temp\relax
1549 \forest@set{\forest@setter@node}{#3}{\the\pgfutil@tempdima}%
1550 },
1551 #1*/.style={%
1552 #1={#4()}*(##1)}%
1553 },
1554 #1:/ .style={%
1555 #1={#4()}/(##1)}%
1556 },
1557 #1'/.code={%
1558 \pgfutil@tempdima=##1\relax
1559 \forest@set{\forest@setter@node}{#3}{\the\pgfutil@tempdima}%
1560 },
1561 #1'+/.code={%
1562 \pgfutil@tempdima=\forestove{#3}\relax
1563 \advance\pgfutil@tempdima##1\relax

```

```

1564     \forestOeset{\forest@setter@node}{#3}{\the\pgfutil@tempdima}%
1565 },
1566 #1'-.code={%
1567     \pgfutil@tempdima=\forestove{#3}\relax
1568     \advance\pgfutil@tempdima-##1\relax
1569     \forestOeset{\forest@setter@node}{#3}{\the\pgfutil@tempdima}%
1570 },
1571 #1'*/.style={%
1572     \pgfutil@tempdima=\forestove{#3}\relax
1573     \multiply\pgfutil@tempdima##1\relax
1574     \forestOeset{\forest@setter@node}{#3}{\the\pgfutil@tempdima}%
1575 },
1576 #1':/.style={%
1577     \pgfutil@tempdima=\forestove{#3}\relax
1578     \divide\pgfutil@tempdima##1\relax
1579     \forestOeset{\forest@setter@node}{#3}{\the\pgfutil@tempdima}%
1580 },
1581 }%
1582 \pgfkeyssetvalue{#1/option@name}{#3}%
1583 \pgfkeyssetvalue{#1+/option@name}{#3}%
1584 \pgfkeyssetvalue{#1-/option@name}{#3}%
1585 \pgfkeyssetvalue{#1*/option@name}{#3}%
1586 \pgfkeyssetvalue{#1:/option@name}{#3}%
1587 \pgfkeyssetvalue{#1'/option@name}{#3}%
1588 \pgfkeyssetvalue{#1'+/option@name}{#3}%
1589 \pgfkeyssetvalue{#1'-/option@name}{#3}%
1590 \pgfkeyssetvalue{#1'*/option@name}{#3}%
1591 \pgfkeyssetvalue{#1':/option@name}{#3}%
1592 }
1593 \def\forest@declarereadonlycount@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
1594     \pgfkeysalso{
1595         #2/if #3/.code n args={3}{%
1596             \forestoget{#3}\forest@temp@option@value
1597             \ifnum\forest@temp@option@value=##1\relax
1598                 \pgfkeysalso{##2}%
1599             \else
1600                 \pgfkeysalso{##3}%
1601             \fi
1602         },
1603         #2/where #3/.style n args={3}{for tree={#2/if #3={##1}{##2}{##3}}},
1604     }%
1605     \pgfmathdeclarefunction{#4}{1}{\forest@pgfmathhelper@attribute@count{##1}{#3}%
1606 }
1607 \def\forest@declarecount@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
1608     \forest@declarereadonlycount@handler{#1}{#2}{#3}{#4}%
1609     \pgfkeysalso{
1610         #1/.code={%
1611             \pgfmathtruncatemacro\forest@temp{##1}%
1612             \forestOlet{\forest@setter@node}{#3}\forest@temp
1613         },
1614         #1+/.code={%
1615             \pgfmathtruncatemacro\forest@temp{##1}%
1616             \c@pgf@counta=\forestove{#3}\relax
1617             \advance\c@pgf@counta\forest@temp\relax
1618             \forestOeset{\forest@setter@node}{#3}{\the\c@pgf@counta}%
1619         },
1620         #1-/.code={%
1621             \pgfmathtruncatemacro\forest@temp{##1}%
1622             \c@pgf@counta=\forestove{#3}\relax
1623             \advance\c@pgf@counta-\forest@temp\relax
1624             \forestOeset{\forest@setter@node}{#3}{\the\c@pgf@counta}%

```



```

1625 },
1626 #1*/.code={%
1627   \pgfmathtruncatemacro\forest@temp{##1}%
1628   \c@pgf@counta=\forestove{#3}\relax
1629   \multiply\c@pgf@counta\forest@temp\relax
1630   \forestOset{\forest@setter@node}{#3}{\the\c@pgf@counta}%
1631 },
1632 #1:/.code={%
1633   \pgfmathtruncatemacro\forest@temp{##1}%
1634   \c@pgf@counta=\forestove{#3}\relax
1635   \divide\c@pgf@counta\forest@temp\relax
1636   \forestOset{\forest@setter@node}{#3}{\the\c@pgf@counta}%
1637 },
1638 #1'/.code={%
1639   \c@pgf@counta=##1\relax
1640   \forestOset{\forest@setter@node}{#3}{\the\c@pgf@counta}%
1641 },
1642 #1'+/.code={%
1643   \c@pgf@counta=\forestove{#3}\relax
1644   \advance\c@pgf@counta##1\relax
1645   \forestOset{\forest@setter@node}{#3}{\the\c@pgf@counta}%
1646 },
1647 #1'-/.code={%
1648   \c@pgf@counta=\forestove{#3}\relax
1649   \advance\c@pgf@counta-##1\relax
1650   \forestOset{\forest@setter@node}{#3}{\the\c@pgf@counta}%
1651 },
1652 #1'*/.style={%
1653   \c@pgf@counta=\forestove{#3}\relax
1654   \multiply\c@pgf@counta##1\relax
1655   \forestOset{\forest@setter@node}{#3}{\the\c@pgf@counta}%
1656 },
1657 #1':/.style={%
1658   \c@pgf@counta=\forestove{#3}\relax
1659   \divide\c@pgf@counta##1\relax
1660   \forestOset{\forest@setter@node}{#3}{\the\c@pgf@counta}%
1661 },
1662 }%
1663 \pgfkeyssetvalue{#1/option@name}{#3}%
1664 \pgfkeyssetvalue{#1+/option@name}{#3}%
1665 \pgfkeyssetvalue{#1-/option@name}{#3}%
1666 \pgfkeyssetvalue{#1*/option@name}{#3}%
1667 \pgfkeyssetvalue{#1:/option@name}{#3}%
1668 \pgfkeyssetvalue{#1'/option@name}{#3}%
1669 \pgfkeyssetvalue{#1'+/option@name}{#3}%
1670 \pgfkeyssetvalue{#1'-/option@name}{#3}%
1671 \pgfkeyssetvalue{#1'* /option@name}{#3}%
1672 \pgfkeyssetvalue{#1': /option@name}{#3}%
1673 }
1674 \def\forest@declareboolean@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
1675   \pgfkeysalso{%
1676     #1/.code={%
1677       \ifstrequal{##1}{1}{%
1678         \forestOset{\forest@setter@node}{#3}{1}%
1679       }{%
1680         \ifstrequal{##1}{0}{%
1681           \forestOset{\forest@setter@node}{#3}{0}%
1682         }{%
1683           \pgfmathifthenelse{##1}{1}{0}%
1684           \forestOlet{\forest@setter@node}{#3}\pgfmathresult
1685         }%

```

```

1686     }%
1687 },
1688 #1/.default=1,
1689 #2/not #3/.code={\forestOset{\forest@setter@node}{#3}{0}},
1690 #2/if #3/.code 2 args={%
1691     \forestoget{#3}\forest@temp@option@value
1692     \ifnum\forest@temp@option@value=1
1693         \pgfkeysalso{##1}%
1694     \else
1695         \pgfkeysalso{##2}%
1696     \fi
1697 },
1698 #2/where #3/.style 2 args={for tree={#2/if #3={##1}{##2}}}
1699 }%
1700 \pgfkeyssetvalue{#1/option@name}{#3}%
1701 \pgfmathdeclarefunction{#4}{1}{\forest@pgfmathhelper@attribute@count{##1}{#3}}%
1702 }
1703 \forestset{
1704   declare toks/.code 2 args={%
1705     \forest@declarehandler\forest@declaretoks@handler{#1}{#2}%
1706   },
1707   declare autowrapped toks/.code 2 args={%
1708     \forest@declarehandler\forest@declareautowrappedtoks@handler{#1}{#2}%
1709   },
1710   declare keylist/.code 2 args={%
1711     \forest@declarehandler\forest@declarekeylist@handler{#1}{#2}%
1712   },
1713   declare readonly dimen/.code={%
1714     \forest@declarehandler\forest@declarereadonlydimen@handler{#1}{}%
1715   },
1716   declare dimen/.code 2 args={%
1717     \forest@declarehandler\forest@declaredimen@handler{#1}{#2}%
1718   },
1719   declare readonly count/.code={%
1720     \forest@declarehandler\forest@declarereadonlycount@handler{#1}{}%
1721   },
1722   declare count/.code 2 args={%
1723     \forest@declarehandler\forest@declarecount@handler{#1}{#2}%
1724   },
1725   declare boolean/.code 2 args={%
1726     \forest@declarehandler\forest@declareboolean@handler{#1}{#2}%
1727   },
1728   /handlers/.restore default value/.code={%
1729     \edef\forest@handlers@currentpath{\pgfkeyscurrentpath}%
1730     \pgfkeysgetvalue{\pgfkeyscurrentpath/option@name}\forest@currentoptionname
1731     \pgfkeysgetvalue{/forest/\forest@currentoptionname}\forest@temp
1732     \expandafter\pgfkeysalso\expandafter{\forest@handlers@currentpath/.expand once=\forest@temp}%
1733   },
1734   /handlers/.pgfmath/.code={%
1735     \pgfmathparse{#1}%
1736     \pgfkeysalso{\pgfkeyscurrentpath/.expand once=\pgfmathresult}%
1737   },
1738   /handlers/.wrap value/.code={%
1739     \edef\forest@handlers@wrap@currentpath{\pgfkeyscurrentpath}%
1740     \pgfkeysgetvalue{\forest@handlers@wrap@currentpath/option@name}\forest@currentoptionname
1741     \forestOget{\pgfkeysvalueof{/forest/\forest@currentoptionname/node@or@reg}}{\forest@currentoptionname}\fo
1742     \forest@def@with@pgfeov\forest@wrap@code{#1}%
1743     \expandafter\edef\expandafter\forest@wrapped@value\expandafter{\expandafter\expandonce\expandafter{\expan
1744     \pgfkeysalso{\forest@handlers@wrap@currentpath/.expand once=\forest@wrapped@value}%
1745   },
1746   /handlers/.option/.code={%

```

```

1747 \edef\forest@marshal{%
1748 \noexpand\pgfkeysalso{\pgfkeyscurrentpath={\forestoption{#1}}}%
1749 }\forest@marshal
1750 },
1751 /handlers/.register/.code={%
1752 \edef\forest@marshal{%
1753 \noexpand\pgfkeysalso{\pgfkeyscurrentpath={\forestregister{#1}}}%
1754 }\forest@marshal
1755 },
1756 /handlers/.wrap pgfmath arg/.code 2 args={%
1757 \pgfmathparse{#2}\let\forest@wrap@arg@i\pgfmathresult
1758 \edef\forest@wrap@args{{\expandonce\forest@wrap@arg@i}}%
1759 \def\forest@wrap@code##1{#1}%
1760 \expandafter\expandafter\expandafter\forest@temp@toks\expandafter\expandafter\expandafter{\expandafter\fo
1761 \pgfkeysalso{\pgfkeyscurrentpath/.expand once=\the\forest@temp@toks}%
1762 },
1763 /handlers/.wrap 2 pgfmath args/.code n args={3}{-%
1764 \pgfmathparse{#2}\let\forest@wrap@arg@i\pgfmathresult
1765 \pgfmathparse{#3}\let\forest@wrap@arg@ii\pgfmathresult
1766 \edef\forest@wrap@args{{\expandonce\forest@wrap@arg@i}{\expandonce\forest@wrap@arg@ii}}%
1767 \def\forest@wrap@code##1##2{#1}%
1768 \expandafter\expandafter\expandafter\def\expandafter\expandafter\expandafter\forest@wrapped\expandafter\ve
1769 \pgfkeysalso{\pgfkeyscurrentpath/.expand once=\forest@wrapped}%
1770 },
1771 /handlers/.wrap 3 pgfmath args/.code n args={4}{-%
1772 \forest@wrap@n@pgfmath@args{#2}{#3}{#4}{-}{-}{-}{-}{3}%
1773 \forest@wrap@n@pgfmath@do{#1}{3}},
1774 /handlers/.wrap 4 pgfmath args/.code n args={5}{-%
1775 \forest@wrap@n@pgfmath@args{#2}{#3}{#4}{#5}{-}{-}{-}{-}{4}%
1776 \forest@wrap@n@pgfmath@do{#1}{4}},
1777 /handlers/.wrap 5 pgfmath args/.code n args={6}{-%
1778 \forest@wrap@n@pgfmath@args{#2}{#3}{#4}{#5}{#6}{-}{-}{-}{5}%
1779 \forest@wrap@n@pgfmath@do{#1}{5}},
1780 /handlers/.wrap 6 pgfmath args/.code n args={7}{-%
1781 \forest@wrap@n@pgfmath@args{#2}{#3}{#4}{#5}{#6}{#7}{-}{-}{6}%
1782 \forest@wrap@n@pgfmath@do{#1}{6}},
1783 /handlers/.wrap 7 pgfmath args/.code n args={8}{-%
1784 \forest@wrap@n@pgfmath@args{#2}{#3}{#4}{#5}{#6}{#7}{#8}{-}{7}%
1785 \forest@wrap@n@pgfmath@do{#1}{7}},
1786 /handlers/.wrap 8 pgfmath args/.code n args={9}{-%
1787 \forest@wrap@n@pgfmath@args{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}{8}%
1788 \forest@wrap@n@pgfmath@do{#1}{8}},
1789 /handlers/.process args/.code={%
1790 \forest@processargs#1\forest@eov\forest@END
1791 },
1792 }
1793 \def\forest@wrap@n@pgfmath@args#1#2#3#4#5#6#7#8#9{%
1794 \pgfmathparse{#1}\let\forest@wrap@arg@i\pgfmathresult
1795 \ifnum#9>1 \pgfmathparse{#2}\let\forest@wrap@arg@ii\pgfmathresult\fi
1796 \ifnum#9>2 \pgfmathparse{#3}\let\forest@wrap@arg@iii\pgfmathresult\fi
1797 \ifnum#9>3 \pgfmathparse{#4}\let\forest@wrap@arg@iv\pgfmathresult\fi
1798 \ifnum#9>4 \pgfmathparse{#5}\let\forest@wrap@arg@v\pgfmathresult\fi
1799 \ifnum#9>5 \pgfmathparse{#6}\let\forest@wrap@arg@vi\pgfmathresult\fi
1800 \ifnum#9>6 \pgfmathparse{#7}\let\forest@wrap@arg@vii\pgfmathresult\fi
1801 \ifnum#9>7 \pgfmathparse{#8}\let\forest@wrap@arg@viii\pgfmathresult\fi
1802 \edef\forest@wrap@args{%
1803 {\expandonce\forest@wrap@arg@i}
1804 \ifnum#9>1 {\expandonce\forest@wrap@arg@ii}\fi
1805 \ifnum#9>2 {\expandonce\forest@wrap@arg@iii}\fi
1806 \ifnum#9>3 {\expandonce\forest@wrap@arg@iv}\fi
1807 \ifnum#9>4 {\expandonce\forest@wrap@arg@v}\fi

```

```

1808 \ifnum#9>5 {\expandonce\forest@wrap@arg@vi}\fi
1809 \ifnum#9>6 {\expandonce\forest@wrap@arg@vii}\fi
1810 \ifnum#9>7 {\expandonce\forest@wrap@arg@viii}\fi
1811 }%
1812 }
1813 \def\forest@wrap@n@pgfmath@do#1#2{%
1814 \ifcase#2\relax
1815 \or\def\forest@wrap@code##1{#1}%
1816 \or\def\forest@wrap@code##1##2{#1}%
1817 \or\def\forest@wrap@code##1##2##3{#1}%
1818 \or\def\forest@wrap@code##1##2##3##4{#1}%
1819 \or\def\forest@wrap@code##1##2##3##4##5{#1}%
1820 \or\def\forest@wrap@code##1##2##3##4##5##6{#1}%
1821 \or\def\forest@wrap@code##1##2##3##4##5##6##7{#1}%
1822 \or\def\forest@wrap@code##1##2##3##4##5##6##7##8{#1}%
1823 \fi
1824 \expandafter\expandafter\expandafter\def\expandafter\expandafter\expandafter\forest@wrapped\expandafter\exp
1825 \pgfkeysalso{\pgfkeyscurrentpath/.expand once=\forest@wrapped}%
1826 }
1827 \newtoks\forest@processargs@result
1828 \newtoks\forest@processargs@current
1829 \newif\ifforest@processargs@append
1830 \def\forest@eov{\forest@eov}
1831 \def\forest@processargs#1#2\forest@END{% #1 = processing instructions, #2 = args
1832 \forest@processargs@result{}}%
1833 \forest@processargs@current{}}%
1834 \forest@processargs@appendfalse
1835 \forest@processargs@getins#1.\forest@END#2\forest@END
1836 }
1837 \def\forest@processargs@maybeappend{%
1838 \ifforest@processargs@append
1839 \eapptotoks\forest@processargs@result{{\the\forest@processargs@current}}}%
1840 \forest@processargs@current{}}%
1841 \fi
1842 }
1843 \def\forest@processargs@getins#1#2\forest@END#3\forest@END{% #1 = first instruction, #2 = rest of instruction
1844 \csname forest@processargs@ins@#1\endcsname#2\forest@END#3\forest@END
1845 }
1846 \csdef{forest@processargs@ins@.}\forest@END#1\forest@END{%
1847 \forest@processargs@maybeappend
1848 \forest@processargs@appremainingargs#1\forest@END
1849 }
1850 \def\forest@processargs@appremainingargs#1#2\forest@END{%
1851 \edef\forest@temp{\unexpanded{#1}}}%
1852 \ifx\forest@temp\forest@eov
1853 \let\forest@processargs@next\forest@processargs@done
1854 \else
1855 \apptotoks\forest@processargs@result{{#1}}}%
1856 \let\forest@processargs@next\forest@processargs@appremainingargs
1857 \fi
1858 \forest@processargs@next#2\forest@END
1859 }
1860 \def\forest@processargs@done#1\forest@END{%
1861 \pgfkeysalso{\pgfkeyscurrentpath/.expanded=\the\forest@processargs@result}}%
1862 }
1863 \csdef{forest@processargs@ins@_}\forest@END#2#3\forest@END{% no processing
1864 % #1 = rest of ins, #2 = first arg, #3 = rest of args
1865 \forest@processargs@maybeappend
1866 \forest@processargs@current{#2}}%
1867 \forest@processargs@appendtrue
1868 \forest@processargs@getins#1\forest@END#3\forest@END

```

```

1869 }
1870 \csdef{forest@processargs@ins@x}#1\forest@END#2#3\forest@END{% expand
1871 % #1 = rest of ins, #2 = first arg, #3 = rest of args
1872 \forest@processargs@maybeappend
1873 \etotoks\forest@processargs@current{#2}%
1874 \forest@processargs@appendtrue
1875 \forest@processargs@getins#1\forest@END#3\forest@END
1876 }
1877 \csdef{forest@processargs@ins@o}#1\forest@END#2#3\forest@END{% expand once
1878 % #1 = rest of ins, #2 = first arg, #3 = rest of args
1879 \forest@processargs@maybeappend
1880 \expandafter\forest@processargs@current\expandafter{#2}%
1881 \forest@processargs@appendtrue
1882 \forest@processargs@getins#1\forest@END#3\forest@END
1883 }
1884 \csdef{forest@processargs@ins@O}#1\forest@END#2#3\forest@END{% option
1885 % #1 = rest of ins, #2 = first arg, #3 = rest of args
1886 \forest@processargs@maybeappend
1887 \etotoks\forest@processargs@current{\forestoption{#2}}%
1888 \forest@processargs@appendtrue
1889 \forest@processargs@getins#1\forest@END#3\forest@END
1890 }
1891 \csdef{forest@processargs@ins@R}#1\forest@END#2#3\forest@END{% register
1892 % #1 = rest of ins, #2 = first arg, #3 = rest of args
1893 \forest@processargs@maybeappend
1894 \etotoks\forest@processargs@current{\forestregister{#2}}%
1895 \forest@processargs@appendtrue
1896 \forest@processargs@getins#1\forest@END#3\forest@END
1897 }
1898 \csdef{forest@processargs@ins@P}#1\forest@END#2#3\forest@END{% pgfmath expression
1899 % #1 = rest of ins, #2 = first arg, #3 = rest of args
1900 \forest@processargs@maybeappend
1901 \pgfmathparse{#2}%
1902 \expandafter\forest@processargs@current\expandafter{\pgfmathresult}%
1903 \forest@processargs@appendtrue
1904 \forest@processargs@getins#1\forest@END#3\forest@END
1905 }
1906 \csdef{forest@processargs@ins@+}#1\forest@END#2\forest@END{% join processors
1907 \forest@processargs@appendfalse
1908 \edef\forest@marshal{%
1909 \unexpanded{\forest@processargs@getins#1\forest@END}{\the\forest@processargs@current}\unexpanded{#2\forest@
1910 }\forest@marshal
1911 }
1912 \csdef{forest@processargs@ins@r}#1\forest@END#2#3\forest@END{% reverse keylist
1913 % #1 = rest of ins, #2 = first arg, #3 = rest of args
1914 \forest@processargs@maybeappend
1915 \forest@processargs@current{}%
1916 \pgfqkeys{/forest}{split={#2}{,}{reverse@keylist}}%
1917 \forest@processargs@appendtrue
1918 \forest@processargs@getins#1\forest@END#3\forest@END
1919 }
1920 \forestset{%
1921 reverse@keylist/.code={%
1922 \epretotoks\forest@processargs@current{#1,}%
1923 },
1924 }

```

### 6.3.2 Registers

Register declaration mechanism is an adjusted copy-paste of the option declaration mechanism.

```

1925 \def\forest@pgfmathhelper@register@toks#1#2{% #1 is discarded: it is present only for analogy with options
1926   \forestrget{#2}\pgfmathresult
1927 }
1928 \def\forest@pgfmathhelper@register@dimen#1#2{%
1929   \forestrget{#2}\forest@temp
1930   \pgfmathparse{+\forest@temp}%
1931 }
1932 \def\forest@pgfmathhelper@register@count#1#2{%
1933   \forestrget{#2}\forest@temp
1934   \pgfmathtruncatemacro\pgfmathresult{\forest@temp}%
1935 }
1936 \def\forest@declareregisterhandler#1#2{%#1=handler for specific type,#2=option name
1937   \pgfkeyssetvalue{/forest/#2/node@or@reg}{register}%
1938   \forest@convert@others@to@underscores{#2}\forest@pgfmathoptionname
1939   \edef\forest@marshal{%
1940     \noexpand#1{/forest/#2}{/forest}{#2}{\forest@pgfmathoptionname}%
1941   }\forest@marshal
1942 }
1943 \def\forest@declaretoksregister@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
1944   \forest@declaretoksregister@handler@A{#1}{#2}{#3}{#4}{}%
1945 }
1946 \def\forest@declarekeylistregister@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
1947   \forest@declaretoksregister@handler@A{#1}{#2}{#3}{#4}{,}%
1948   \forest@copycommandkey{#1}{#1}'%
1949   \pgfkeyssetvalue{#1'/option@name}{#3}%
1950   \forest@copycommandkey{#1+}{#1}%
1951   \pgfkeysalso{#1-/.code={%
1952     \forest@fornode{register}{%
1953       \forest@node@removekeysfromkeylist{##1}{#3}%
1954     }}%
1955   \pgfkeyssetvalue{#1-/option@name}{#3}%
1956 }
1957 \def\forest@declaretoksregister@handler@A#1#2#3#4#5{% #1=key,#2=path,#3=name,#4=pgfmathname,#5=infix
1958   \pgfkeysalso{%
1959     #1/.code={\forestrset{#3}{##1}},
1960     #1+/.code={\forestrappto{#3}{#5##1}},
1961     #2/+#3/.code={\forestrpreto{#3}{##1#5}},
1962     #2/if #3/.code n args={3}{%
1963       \forestrget{#3}\forest@temp@option@value
1964       \edef\forest@temp@compared@value{\unexpanded{##1}}%
1965       \ifx\forest@temp@option@value\forest@temp@compared@value
1966         \pgfkeysalso{##2}%
1967       \else
1968         \pgfkeysalso{##3}%
1969       \fi
1970     },
1971     #2/if in #3/.code n args={3}{%
1972       \forestrget{#3}\forest@temp@option@value
1973       \edef\forest@temp@compared@value{\unexpanded{##1}}%
1974       \expandafter\expandafter\expandafter\pgfutil@in@\expandafter\expandafter\expandafter{\expandafter\forest@temp@option@value}
1975       \ifpgfutil@in@
1976         \pgfkeysalso{##2}%
1977       \else
1978         \pgfkeysalso{##3}%
1979       \fi
1980     },
1981   }%
1982   \pgfkeyssetvalue{#1/option@name}{#3}%
1983   \pgfkeyssetvalue{#1+/option@name}{#3}%
1984   \pgfkeyssetvalue{#2/+#3/option@name}{#3}%
1985   \pgfmathdeclarefunction{#4}{1}{\forest@pgfmathhelper@register@toks{##1}{#3}}%

```

```

1986 }
1987 \def\forest@declareautowrappedtoksregister@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname,#5=infix
1988 \forest@declaretoksregister@handler{#1}{#2}{#3}{#4}%
1989 \forest@copycommandkey{#1}{#1'}%
1990 \pgfkeysalso{#1/.style={#1'/.wrap value={##1}}}%
1991 \pgfkeyssetvalue{#1'/option@name}{#3}%
1992 \forest@copycommandkey{#1+}{#1+'}%
1993 \pgfkeysalso{#1+/.style={#1+'/.wrap value={##1}}}%
1994 \pgfkeyssetvalue{#1+'/option@name}{#3}%
1995 \forest@copycommandkey{#2/+3}{#2/+3'}%
1996 \pgfkeysalso{#2/+3/.style={#2/+3'/.wrap value={##1}}}%
1997 \pgfkeyssetvalue{#2/+3'/option@name}{#3}%
1998 }
1999 \def\forest@declarereadonlydimenregister@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
2000 \pgfkeysalso{%
2001 #2/if #3/.code n args={3}{%
2002 \forestrget{#3}\forest@temp@option@value
2003 \ifdim\forest@temp@option@value=##1\relax
2004 \pgfkeysalso{##2}%
2005 \else
2006 \pgfkeysalso{##3}%
2007 \fi
2008 },
2009 }%
2010 \pgfmathdeclarefunction{#4}{1}{\forest@pgfmathhelper@register@dimen{##1}{#3}}%
2011 }
2012 \def\forest@declaredimenregister@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
2013 \forest@declarereadonlydimenregister@handler{#1}{#2}{#3}{#4}%
2014 \pgfkeysalso{%
2015 #1/.code={%
2016 \pgfmathsetlengthmacro\forest@temp{##1}%
2017 \forestrlet{#3}\forest@temp
2018 },
2019 #1+/.code={%
2020 \pgfmathsetlengthmacro\forest@temp{##1}%
2021 \pgfutil@tempdima=\forestrve{#3}
2022 \advance\pgfutil@tempdima\forest@temp\relax
2023 \forestreset{#3}{\the\pgfutil@tempdima}%
2024 },
2025 #1-/.code={%
2026 \pgfmathsetlengthmacro\forest@temp{##1}%
2027 \pgfutil@tempdima=\forestrve{#3}
2028 \advance\pgfutil@tempdima-\forest@temp\relax
2029 \forestreset{#3}{\the\pgfutil@tempdima}%
2030 },
2031 #1*/.style={%
2032 #1={#4()* (##1)}%
2033 },
2034 #1:/ .style={%
2035 #1={#4()/ (##1)}%
2036 },
2037 #1'/.code={%
2038 \pgfutil@tempdima=##1\relax
2039 \forestreset{#3}{\the\pgfutil@tempdima}%
2040 },
2041 #1'+/.code={%
2042 \pgfutil@tempdima=\forestrve{#3}\relax
2043 \advance\pgfutil@tempdima##1\relax
2044 \forestreset{#3}{\the\pgfutil@tempdima}%
2045 },
2046 #1'-/.code={%

```

```

2047     \pgfutil@tempdima=\forestrve{#3}\relax
2048     \advance\pgfutil@tempdima-##1\relax
2049     \forestreset{#3}{\the\pgfutil@tempdima}%
2050   },
2051   #1'*/.style={%
2052     \pgfutil@tempdima=\forestrve{#3}\relax
2053     \multiply\pgfutil@tempdima##1\relax
2054     \forestreset{#3}{\the\pgfutil@tempdima}%
2055   },
2056   #1':/.style={%
2057     \pgfutil@tempdima=\forestrve{#3}\relax
2058     \divide\pgfutil@tempdima##1\relax
2059     \forestreset{#3}{\the\pgfutil@tempdima}%
2060   },
2061   }%
2062   \pgfkeyssetvalue{#1/option@name}{#3}%
2063   \pgfkeyssetvalue{#1+/option@name}{#3}%
2064   \pgfkeyssetvalue{#1-/option@name}{#3}%
2065   \pgfkeyssetvalue{#1*/option@name}{#3}%
2066   \pgfkeyssetvalue{#1:/option@name}{#3}%
2067   \pgfkeyssetvalue{#1'/option@name}{#3}%
2068   \pgfkeyssetvalue{#1'+/option@name}{#3}%
2069   \pgfkeyssetvalue{#1'-/option@name}{#3}%
2070   \pgfkeyssetvalue{#1'*/*option@name}{#3}%
2071   \pgfkeyssetvalue{#1':/*option@name}{#3}%
2072 }
2073 \def\forest@declarereadonlycountregister@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
2074   \pgfkeysalso{
2075     #2/if #3/.code n args={3}{%
2076       \forestrget{#3}\forest@temp@option@value
2077       \ifnum\forest@temp@option@value=##1\relax
2078         \pgfkeysalso{##2}%
2079       \else
2080         \pgfkeysalso{##3}%
2081       \fi
2082     },
2083   }%
2084   \pgfmathdeclarefunction{#4}{1}{\forest@pgfmathhelper@register@count{##1}{#3}}%
2085 }
2086 \def\forest@declarecountregister@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
2087   \forest@declarereadonlycountregister@handler{#1}{#2}{#3}{#4}%
2088   \pgfkeysalso{
2089     #1/.code={%
2090       \pgfmathtruncatemacro\forest@temp{##1}%
2091       \forestrlet{#3}\forest@temp
2092     },
2093     #1+/.code={%
2094       \pgfmathtruncatemacro\forest@temp{##1}%
2095       \c@pgf@counta=\forestrve{#3}\relax
2096       \advance\c@pgf@counta\forest@temp\relax
2097       \forestreset{#3}{\the\c@pgf@counta}%
2098     },
2099     #1-/.code={%
2100       \pgfmathtruncatemacro\forest@temp{##1}%
2101       \c@pgf@counta=\forestrve{#3}\relax
2102       \advance\c@pgf@counta-\forest@temp\relax
2103       \forestreset{#3}{\the\c@pgf@counta}%
2104     },
2105     #1*/.code={%
2106       \pgfmathtruncatemacro\forest@temp{##1}%
2107       \c@pgf@counta=\forestrve{#3}\relax

```



```

2108     \multiply\c@pgf@counta\forest@temp\relax
2109     \forestreset{#3}{\the\c@pgf@counta}%
2110 },
2111 #1/.code={%
2112     \pgfmathtruncatemacro\forest@temp{##1}%
2113     \c@pgf@counta=\forestrve{#3}\relax
2114     \divide\c@pgf@counta\forest@temp\relax
2115     \forestreset{#3}{\the\c@pgf@counta}%
2116 },
2117 #1'/.code={%
2118     \c@pgf@counta=##1\relax
2119     \forestreset{#3}{\the\c@pgf@counta}%
2120 },
2121 #1'+/.code={%
2122     \c@pgf@counta=\forestrve{#3}\relax
2123     \advance\c@pgf@counta##1\relax
2124     \forestreset{#3}{\the\c@pgf@counta}%
2125 },
2126 #1'-/.code={%
2127     \c@pgf@counta=\forestrve{#3}\relax
2128     \advance\c@pgf@counta-##1\relax
2129     \forestreset{#3}{\the\c@pgf@counta}%
2130 },
2131 #1'*/.style={%
2132     \c@pgf@counta=\forestrve{#3}\relax
2133     \multiply\c@pgf@counta##1\relax
2134     \forestreset{#3}{\the\c@pgf@counta}%
2135 },
2136 #1':/.style={%
2137     \c@pgf@counta=\forestrve{#3}\relax
2138     \divide\c@pgf@counta##1\relax
2139     \forestreset{#3}{\the\c@pgf@counta}%
2140 },
2141 }%
2142 \pgfkeyssetvalue{#1/option@name}{#3}%
2143 \pgfkeyssetvalue{#1+/option@name}{#3}%
2144 \pgfkeyssetvalue{#1-/option@name}{#3}%
2145 \pgfkeyssetvalue{#1*/option@name}{#3}%
2146 \pgfkeyssetvalue{#1:/option@name}{#3}%
2147 \pgfkeyssetvalue{#1'/option@name}{#3}%
2148 \pgfkeyssetvalue{#1'+/option@name}{#3}%
2149 \pgfkeyssetvalue{#1'-/option@name}{#3}%
2150 \pgfkeyssetvalue{#1'*/option@name}{#3}%
2151 \pgfkeyssetvalue{#1':/option@name}{#3}%
2152 }
2153 \def\forest@declarebooleanregister@handler#1#2#3#4{% #1=key,#2=path,#3=name,#4=pgfmathname
2154 \pgfkeysalso{%
2155     #1/.code={%
2156         \ifstrequal{##1}{1}{%
2157             \forestrset{#3}{1}%
2158         }{%
2159             \ifstrequal{##1}{0}{%
2160                 \forestrset{#3}{0}%
2161             }{%
2162                 \pgfmathifthenelse{##1}{1}{0}%
2163                 \forestrlet{#3}\pgfmathresult
2164             }%
2165         }%
2166     },
2167     #1/.default=1,
2168     #2/not #3/.code={\forestrset{#3}{0}},

```

```

2169 #2/if #3/.code 2 args={%
2170   \forestrget{#3}\forest@temp@option@value
2171   \ifnum\forest@temp@option@value=1
2172     \pgfkeysalso{##1}%
2173   \else
2174     \pgfkeysalso{##2}%
2175   \fi
2176 },
2177 }%
2178 \pgfkeyssetvalue{#1/option@name}{#3}%
2179 \pgfmathdeclarefunction{#4}{1}{\forest@pgfmathhelper@register@count{##1}{#3}}%
2180 }
2181 \forestset{
2182   declare toks register/.code={%
2183     \forest@declareregisterhandler\forest@declaretoksregister@handler{#1}%
2184     \forestset{#1={}}%
2185   },
2186   declare autowrapped toks register/.code={%
2187     \forest@declareregisterhandler\forest@declareautowrappedtoksregister@handler{#1}%
2188     \forestset{#1={}}%
2189   },
2190   declare keylist register/.code={%
2191     \forest@declareregisterhandler\forest@declarekeylistregister@handler{#1}%
2192     \forestset{#1'={}}%
2193   },
2194   declare dimen register/.code={%
2195     \forest@declareregisterhandler\forest@declaredimenregister@handler{#1}%
2196     \forestset{#1'={0pt}}%
2197   },
2198   declare count register/.code={%
2199     \forest@declareregisterhandler\forest@declarecountregister@handler{#1}%
2200     \forestset{#1'={0}}%
2201   },
2202   declare boolean register/.code={%
2203     \forest@declareregisterhandler\forest@declarebooleanregister@handler{#1}%
2204     \forestset{#1={0}}%
2205   },
2206 }

```

Declare some temporary registers.

```

2207 \forestset{
2208   declare toks register=temptoksa,temptoksa={},
2209   declare toks register=temptoksb,temptoksb={},
2210   declare toks register=temptoksc,temptoksc={},
2211   declare toks register=temptoksd,temptoksd={},
2212   declare keylist register=tempkeylista,tempkeylista'={},
2213   declare keylist register=tempkeylistb,tempkeylistb'={},
2214   declare keylist register=tempkeylistc,tempkeylistc'={},
2215   declare keylist register=tempkeylistd,tempkeylistd'={},
2216   declare dimen register=tempdima,tempdima'={0pt},
2217   declare dimen register=tempdimb,tempdimb'={0pt},
2218   declare dimen register=tempdimc,tempdimc'={0pt},
2219   declare dimen register=tempdimd,tempdimd'={0pt},
2220   declare dimen register=tempdimx,tempdimx'={0pt},
2221   declare dimen register=tempdimy,tempdimy'={0pt},
2222   declare dimen register=tempdiml,tempdiml'={0pt},
2223   declare dimen register=tempdims,tempdims'={0pt},
2224   declare count register=tempcounta,tempcounta'={0},
2225   declare count register=tempcountb,tempcountb'={0},
2226   declare count register=tempcountc,tempcountc'={0},
2227   declare count register=tempcountd,tempcountd'={0},

```

```

2228 declare boolean register=tempboola,tempboola={0},
2229 declare boolean register=tempboolb,tempboolb={0},
2230 declare boolean register=tempboolc,tempboolc={0},
2231 declare boolean register=tempboold,tempboold={0},
2232 }

```

### 6.3.3 Declaring options

```

2233 \def\forest@node@Nametoid#1{% #1 = name
2234 \csname forest@id@of@#1\endcsname
2235 }
2236 \def\forest@node@ifnamedefined#1#2#3{% #1 = name, #2=true,#3=false
2237 \ifcvoid{forest@id@of@#1}{#3}{#2}%
2238 }
2239 \def\forest@node@setname#1{%
2240 \def\forest@temp@setname{y}%
2241 \def\forest@temp@silent{n}%
2242 \def\forest@temp@propagating{n}%
2243 \forest@node@setnameoralias{#1}%
2244 }
2245 \def\forest@node@setname@silent#1{%
2246 \def\forest@temp@setname{y}%
2247 \def\forest@temp@silent{y}%
2248 \def\forest@temp@propagating{n}%
2249 \forest@node@setnameoralias{#1}%
2250 }
2251 \def\forest@node@setalias#1{%
2252 \def\forest@temp@setname{n}%
2253 \def\forest@temp@silent{n}%
2254 \def\forest@temp@propagating{n}%
2255 \forest@node@setnameoralias{#1}%
2256 }
2257 \def\forest@node@setalias@silent#1{%
2258 \def\forest@temp@setname{n}%
2259 \def\forest@temp@silent{y}%
2260 \def\forest@temp@propagating{n}%
2261 \forest@node@setnameoralias{#1}%
2262 }
2263 \def\forest@node@setnameoralias#1{%
2264 \ifstrempy{#1}{%
2265 \forest@node@setnameoralias{node@\forest@cn}%
2266 }{%
2267 \forest@node@ifnamedefined{#1}{%
2268 \if y\forest@temp@propagating
2269 % this will find a unique name, eventually:
2270 \@escapeif{\forest@node@setnameoralias{#1@\forest@cn}}%
2271 \else\@escapeif{%
2272 \if y\forest@temp@setname
2273 \@escapeif{\forest@node@setnameoralias@nameclash{#1}}%
2274 \else\@escapeif{% setting an alias: clashing with alias is not a problem
2275 \forest@get{\forest@node@Nametoid{#1}}{name}\forest@temp
2276 \expandafter\ifstrequal\expandafter{\forest@temp}{#1}{%
2277 \forest@node@setnameoralias@nameclash{#1}%
2278 }{%
2279 \forest@node@setnameoralias@do{#1}%
2280 }%
2281 }\fi
2282 }\fi
2283 }{%
2284 \forest@node@setnameoralias@do{#1}%
2285 }%

```

```

2286 }%
2287 }
2288 \def\forest@node@setnameoralias@nameclash#1{%
2289   \if y\forest@temp@silent
2290     \forest@fornode{\forest@node@Nametoid{#1}}{%
2291       \def\forest@temp@propagating{y}%
2292       \forest@node@setnameoralias{}%
2293     }%
2294     \forest@node@setnameoralias@do{#1}%
2295   \else
2296     \PackageError{forest}{Node name "#1" is already used}{}%
2297   \fi
2298 }
2299 \def\forest@node@setnameoralias@do#1{%
2300   \if y\forest@temp@setname
2301     \csdef{forest@id@of@\forest@name}{}%
2302     \forest@set{#1}%
2303   \fi
2304   \csedef{forest@id@of@#1}{\forest@cn}%
2305 }
2306 \forestset{
2307   TeX/.code={#1},
2308   TeX'/.code={\appto\forest@externalize@loadimages{#1}#1},
2309   TeX'/.code={\appto\forest@externalize@loadimages{#1}},
2310   options/.code={\forestset{#1}},
2311   typeout/.style={TeX={\typeout{#1}}},
2312   declare toks={name}{},
2313   name/.code={% override the default setter
2314     \forest@fornode{\forest@setter@node}{\forest@node@setname{#1}}%
2315   },
2316   name/.default={},
2317   name'/.code={% override the default setter
2318     \forest@fornode{\forest@setter@node}{\forest@node@setname@silent{#1}}%
2319   },
2320   name'/.default={},
2321   alias/.code={\forest@fornode{\forest@setter@node}{\forest@node@setalias{#1}}},
2322   alias'/.code={\forest@fornode{\forest@setter@node}{\forest@node@setalias@silent{#1}}},
2323   begin draw/.code={\begin{tikzpicture}},
2324   end draw/.code={\end{tikzpicture}},
2325   declare keylist register=default preamble,
2326   default preamble'={},
2327   declare keylist register=preamble,
2328   preamble'={},
2329   declare autowrapped toks={content}{},
2330   % #1 = which option to split, #2 = separator (one char!), #3 = receiving options
2331   %%% begin listing region: split_option
2332   split option/.style n args=3{split/.process args={0}{#1}{#2}{#3}}
2333   %%% end listing region: split_option
2334   ,
2335   split register/.style n args=3{% #1 = which register to split, #2 = separator (one char!), #3 = receiving o
2336     split/.process args={R}{#1}{#2}{#3},
2337   },
2338   TeX={%
2339     \def\forest@split@sourcevalues{%
2340       \def\forest@split@sourcevalue{%
2341         \def\forest@split@receivingoptions{%
2342           \def\forest@split@receivingoption{}%
2343         }%
2344       split/.code n args=3{% #1 = string to split, #2 = separator (one char!), #3 = receiving options
2345         \forest@saveandrestoremacro\forest@split@sourcevalues{%
2346           \forest@saveandrestoremacro\forest@split@sourcevalue{%

```

```

2347 \forest@saveandrestoremacro\forest@split@receivingoptions{%
2348 \forest@saveandrestoremacro\forest@split@receivingoption{%
2349 \def\forest@split@sourcevalues{#1#2}%
2350 \edef\forest@split@receivingoptions{#3,}%
2351 \def\forest@split@receivingoption{%
2352 \safeloop
2353 \expandafter\forest@split\expandafter{\forest@split@sourcevalues}{#2}\forest@split@sourcevalue
2354 \ifdefempty\forest@split@receivingoptions{}{%
2355 \expandafter\forest@split\expandafter{\forest@split@receivingoptions}{,}\forest@temp\forest@split@receivingoption
2356 \ifdefempty\forest@temp{}{\let\forest@split@receivingoption\forest@temp\def\forest@temp{}}%
2357 }%
2358 \edef\forest@marshal{%
2359 \noexpand\pgfkeysalso{\forest@split@receivingoption=\forest@split@sourcevalue}}%
2360 }\forest@marshal
2361 \ifdefempty\forest@split@sourcevalues{\forest@tempfalse}{\forest@temptrue}%
2362 \ifforest@temp
2363 \saferepeat
2364 }}}}%,
2365 },
2366 declare count={grow}{270},
2367 TeX={% a hack for grow-reversed connection, and compass-based grow specification
2368 \forest@copycommandkey{/forest/grow}{/forest/grow@0}%
2369 %\pgfkeysgetvalue{/forest/grow/.@cmd}\forest@temp
2370 %\pgfkeyslet{/forest/grow@0/.@cmd}\forest@temp
2371 },
2372 grow/.style={grow@=#1,reversed=0},
2373 grow'/.style={grow@=#1,reversed=1},
2374 grow''/.style={grow@=#1}},
2375 grow@/.is choice,
2376 grow@/east/.style={/forest/grow@0=0},
2377 grow@/north east/.style={/forest/grow@0=45},
2378 grow@/north/.style={/forest/grow@0=90},
2379 grow@/north west/.style={/forest/grow@0=135},
2380 grow@/west/.style={/forest/grow@0=180},
2381 grow@/south west/.style={/forest/grow@0=225},
2382 grow@/south/.style={/forest/grow@0=270},
2383 grow@/south east/.style={/forest/grow@0=315},
2384 grow@/.unknown/.code={\let\forest@temp@grow\pgfkeyscurrentname
2385 \pgfkeysalso{/forest/grow@0/.expand once=\forest@temp@grow}},
2386 declare boolean={reversed}{0},
2387 declare toks={parent anchor}{},
2388 declare toks={child anchor}{},
2389 declare toks={anchor}{base},
2390 Autoforward={anchor}{
2391 node options-=anchor,
2392 node options+={anchor=#1}}
2393 },
2394 anchor'/.style={anchor@no@compass=true,anchor=#1},
2395 anchor+'/.style={anchor@no@compass=true,anchor+=#1},
2396 anchor-'/.style={anchor@no@compass=true,anchor-=#1},
2397 anchor*'/.style={anchor@no@compass=true,anchor*=#1},
2398 anchor:'.style={anchor@no@compass=true,anchor:=#1},
2399 anchor'+'.style={anchor@no@compass=true,anchor'+=#1},
2400 anchor'-'.style={anchor@no@compass=true,anchor'-=#1},
2401 anchor*''.style={anchor@no@compass=true,anchor*'=#1},
2402 anchor:'.style={anchor@no@compass=true,anchor':=#1},
2403 % /tikz/forest anchor/.style={
2404 % /forest/TeX={\forestanchor totikzanchor{#1}\forest@temp@anchor},
2405 % anchor/.expand once=\forest@temp@anchor
2406 % },
2407 declare toks={calign}{midpoint},

```

```

2408 TeX={%
2409   \forest@copycommandkey{/forest/calign}{/forest/calign'}%
2410 },
2411 calign/.is choice,
2412 calign/child/.style={calign'=child},
2413 calign/first/.style={calign'=child,calign primary child=1},
2414 calign/last/.style={calign'=child,calign primary child=-1},
2415 calign with current/.style={for parent/.wrap pgfmath arg={calign=child,calign primary child=##1}{n}},
2416 calign with current edge/.style={for parent/.wrap pgfmath arg={calign=child edge,calign primary child=##1}{n}},
2417 calign/child edge/.style={calign'=child edge},
2418 calign/midpoint/.style={calign'=midpoint},
2419 calign/center/.style={calign'=midpoint,calign primary child=1,calign secondary child=-1},
2420 calign/edge midpoint/.style={calign'=edge midpoint},
2421 calign/fixed angles/.style={calign'=fixed angles},
2422 calign/fixed edge angles/.style={calign'=fixed edge angles},
2423 calign/.unknown/.code={\PackageError{forest}{unknown calign '\pgfkeyscurrentname'}{}}},
2424 declare count={calign primary child}{1},
2425 declare count={calign secondary child}{-1},
2426 declare count={calign primary angle}{-35},
2427 declare count={calign secondary angle}{35},
2428 calign child/.style={calign primary child={#1}},
2429 calign angle/.style={calign primary angle={-#1},calign secondary angle={#1}},
2430 declare toks={tier}{},
2431 declare toks={fit}{tight},
2432 declare boolean={ignore}{0},
2433 declare boolean={ignore edge}{0},
2434 no edge/.style={edge'={},ignore edge},
2435 declare keylist={edge}{draw},
2436 declare toks={edge path}{%
2437   \noexpand\path[\forestoption{edge}]%
2438   (\forestOve{\forestove{@parent}}{name}.parent anchor)--(\forestove{name}.child anchor)
2439   % =
2440   % (!u.parent anchor)--(.child anchor)\forestoption{edge label};
2441   \forestoption{edge label};%
2442 },
2443 edge path'/.style={
2444   edge path={%
2445     \noexpand\path[\forestoption{edge}]%
2446     #1%
2447     \forestoption{edge label};
2448   }
2449 },
2450 declare toks={edge label}{},
2451 declare boolean={phantom}{0},
2452 baseline/.style={alias={forest@baseline@node}},
2453 declare readonly count={n},
2454 declare readonly count={n'},
2455 declare readonly count={n children},
2456 declare readonly count={level},
2457 declare dimen=x{0pt},
2458 declare dimen=y{0pt},
2459 declare dimen={s}{0pt},
2460 declare dimen={l}{6ex}, % just in case: should be set by the calibration
2461 declare dimen={s sep}{0.6666em},
2462 declare dimen={l sep}{1ex}, % just in case: calibration!
2463 declare keylist={node options}{anchor=base},
2464 declare toks={tikz}{},
2465 afterthought/.style={tikz+={#1}},
2466 label/.style={tikz={\path[late options={%
2467   name=\forestoption{name},label={#1}}];}},
2468 pin/.style={tikz={\path[late options={%

```

```

2469     name=\forestoption{name},pin={#1}};});
2470 declare toks={content format}{\forestoption{content}},
2471 plain content/.style={content format={\forestoption{content}}},
2472 math content/.style={content format={\noexpand\ensuremath{\forestoption{content}}}},
2473 declare toks={node format}{%
2474   \noexpand\node
2475   (\forestoption{name})%
2476   [\forestoption{node options}]%
2477   {\forestoption{content format}};%
2478 },
2479 node format'/.style={
2480   node format={\noexpand\node(\forestoption{name})#1;}
2481 },
2482 tabular@environment/.style={content format={%
2483   \noexpand\begin{tabular}[\forestoption{base}]{\forestoption{align}}%
2484     \forestoption{content}%
2485   \noexpand\end{tabular}%
2486 }},
2487 declare toks={align}{},
2488 TeX={%
2489   \forest@copycommandkey{/forest/align}{/forest/align'}%
2490   %\pgfkeysgetvalue{/forest/align/.@cmd}\forest@temp
2491   %\pgfkeyslet{/forest/align'/.@cmd}\forest@temp
2492 },
2493 align/.is choice,
2494 align/.unknown/.code={%
2495   \edef\forest@marshal{%
2496     \noexpand\pgfkeysalso{%
2497       align'={\pgfkeyscurrentname},%
2498       tabular@environment
2499     }%
2500   }\forest@marshal
2501 },
2502 align/center/.style={align'={@{}c@{}},tabular@environment},
2503 align/left/.style={align'={@{}l@{}},tabular@environment},
2504 align/right/.style={align'={@{}r@{}},tabular@environment},
2505 declare toks={base}{t},
2506 TeX={%
2507   \forest@copycommandkey{/forest/base}{/forest/base'}%
2508   %\pgfkeysgetvalue{/forest/base/.@cmd}\forest@temp
2509   %\pgfkeyslet{/forest/base'/.@cmd}\forest@temp
2510 },
2511 base/.is choice,
2512 base/top/.style={base'=t},
2513 base/bottom/.style={base'=b},
2514 base/.unknown/.style={base'/.expand once=\pgfkeyscurrentname},
2515 unknown to/.store in=\forest@unknownto,
2516 unknown to=node options,
2517 unknown key error/.code={\PackageError{forest}{Unknown keyval: \detokenize{#1}}{}},
2518 content to/.store in=\forest@contentto,
2519 content to=content,
2520 .unknown/.code={%
2521   \expandafter\pgfutil@in@\expandafter.\expandafter{\pgfkeyscurrentname}%
2522   \ifpgfutil@in@
2523     \expandafter\forest@relatednode@option@setter\pgfkeyscurrentname=#1\forest@END
2524   \else
2525     \edef\forest@marshal{%
2526       \noexpand\pgfkeysalso{\forest@unknownto={\pgfkeyscurrentname=\unexpanded{#1}}}%
2527     }\forest@marshal
2528   \fi
2529 },

```

```

2530 get node boundary/.code={%
2531   \forest@get{@boundary}\forest@node@boundary
2532   \def#1{}%
2533   \forest@extendpath#1\forest@node@boundary{\pgfpoint{\forestove{x}}{\forestove{y}}}%
2534 },
2535 % get min l tree boundary/.code={%
2536 %   \forest@get@tree@boundary{negative}{\the\numexpr\forestove{grow}-90\relax}#1},
2537 % get max l tree boundary/.code={%
2538 %   \forest@get@tree@boundary{positive}{\the\numexpr\forestove{grow}-90\relax}#1},
2539 get min s tree boundary/.code={%
2540   \forest@get@tree@boundary{negative}{\forestove{grow}}#1},
2541 get max s tree boundary/.code={%
2542   \forest@get@tree@boundary{positive}{\forestove{grow}}#1},
2543 use as bounding box/.style={%
2544   before drawing tree={
2545     tikz+/.expanded={%
2546       \noexpand\pgfresetboundingbox
2547       \noexpand\useasboundingbox
2548       ($(.anchor)+(\forestoption{min x},\forestoption{min y})$)
2549       rectangle
2550       ($(.anchor)+(\forestoption{max x},\forestoption{max y})$)
2551     };
2552   }
2553 }
2554 },
2555 use as bounding box'/.style={%
2556   before drawing tree={
2557     tikz+/.expanded={%
2558       \noexpand\pgfresetboundingbox
2559       \noexpand\useasboundingbox
2560       ($(.anchor)+(\forestoption{min x}+\pgfkeysvalueof{/pgf/outer xsep}/2+\pgfkeysvalueof{/pgf/inner xsep})
2561       rectangle
2562       ($(.anchor)+(\forestoption{max x}-\pgfkeysvalueof{/pgf/outer xsep}/2-\pgfkeysvalueof{/pgf/inner xsep})
2563       );
2564     }
2565   }
2566 },
2567 }%
2568 \def\forest@iftikzkey#1#2#3{% #1 = key name, #2 = true code, #3 = false code
2569   \forest@temptrue
2570   \pgfkeysifdefined{/tikz/\pgfkeyscurrentname}{-}{%
2571     \pgfkeysifdefined{/tikz/\pgfkeyscurrentname/.cmd}{-}{%
2572       \pgfkeysifdefined{/pgf/\pgfkeyscurrentname}{-}{%
2573         \pgfkeysifdefined{/pgf/\pgfkeyscurrentname/.cmd}{-}{%
2574           \forest@tempfalse
2575         }}}}%
2576   \ifforest@temp\@escapeif{#2}\else\@escapeif{#3}\fi
2577 }
2578 \def\forest@ifoptionortikzkey#1#2#3{% #1 = key name, #2 = true code, #3 = false code
2579   \forest@temptrue
2580   \pgfkeysifdefined{/forest/\pgfkeyscurrentname}{-}{%
2581     \pgfkeysifdefined{/forest/\pgfkeyscurrentname/.cmd}{-}{%
2582       \forest@iftikzkey{#1}{-}{-}{%
2583         }}}}%
2584   \ifforest@temp\@escapeif{#2}\else\@escapeif{#3}\fi
2585 }
2586 \def\forest@get@tree@boundary#1#2#3{%#1=pos/neg,#2=grow,#3=receiving cs
2587   \def#3{}%
2588   \forest@node@getedge{#1}{#2}\forest@temp@boundary
2589   \forest@extendpath#3\forest@temp@boundary{\pgfpoint{\forestove{x}}{\forestove{y}}}%
2590 }

```



```

2591 \def\forest@setter@node{\forest@cn}%
2592 \def\forest@relatednode@option@compat@ignoreinvalidsteps#1{#1}
2593 \def\forest@relatednode@option@setter#1.#2=#3\forest@END{%
2594   \forest@forthis{%
2595     \forest@relatednode@option@compat@ignoreinvalidsteps{%
2596       \forest@nameandgo{#1}%
2597       \let\forest@setter@node\forest@cn
2598     }%
2599   }%
2600 \ifnum\forest@setter@node=0
2601 \else
2602   \forestset{#2={#3}}%
2603 \fi
2604 \def\forest@setter@node{\forest@cn}%
2605 }%
2606 \def\forest@split#1#2#3#4{% #1=list (assuming that the list is nonempty and finishes with the separator), #2
2607   \def\forest@split@@##1#2##2\forest@split@@##3##4{\def##3{##1}\def##4{##2}}%
2608   \forest@split@@#1\forest@split@@{#3}{#4}}

```

### 6.3.4 Option propagation

The propagators targeting single nodes are automatically defined by nodewalk steps definitions.

```

2609 \forestset{
2610   for tree'/.style 2 args={#1,for children={for tree'={#1}{#2}},#2},
2611   if/.code n args={3}{%
2612     \pgfmathparse{#1}%
2613     \ifnum\pgfmathresult=0
2614       \@escapeif{\pgfkeysalso{#3}}%
2615     \else
2616       \@escapeif{\pgfkeysalso{#2}}%
2617     \fi
2618   },
2619   if nodewalk valid/.code n args={3}{%
2620     \edef\forest@marshal{%
2621       \unexpanded{\forest@forthis{%
2622         \forest@nodewalk{%
2623           on invalid={fake}{#1},
2624           TeX={\global\let\forest@global@temp\forest@cn}
2625         }{}}%
2626       }%
2627       \def\forest@cn{\forest@cn}\unexpanded{%
2628         \ifnum\forest@global@temp=0
2629           \@escapeif{\pgfkeysalso{#3}}%
2630         \else
2631           \@escapeif{\pgfkeysalso{#2}}%
2632         \fi}%
2633       }\forest@marshal
2634   },
2635   if nodewalk empty/.code n args={3}{%
2636     \forest@forthis{%
2637       \forest@nodewalk{%
2638         on invalid={fake}{#1},
2639         TeX={\global\let\forest@global@temp\forest@nodewalk@cn},
2640       }{}}%
2641     }%
2642     \ifnum\forest@global@temp=0
2643       \@escapeif{\pgfkeysalso{#2}}%
2644     \else
2645       \@escapeif{\pgfkeysalso{#3}}%
2646     \fi
2647   },

```

```

2648 where/.style n args={3}{for tree={if={#1}{#2}{#3}}},
2649 where nodewalk valid/.style n args={3}{for tree={if nodewalk valid={#1}{#2}{#3}}},
2650 where nodewalk empty/.style n args={3}{for tree={if nodewalk empty={#1}{#2}{#3}}},
2651 repeat/.code 2 args={%
2652   \pgfmathtruncatemacro\forest@temp{#1}%
2653   \expandafter\forest@repeatkey\expandafter{\forest@temp}{#2}%
2654 },
2655 until/.code 2 args={%
2656   \ifstrempy{#1}{%
2657     \forest@untilkey{\ifnum\forest@cn=0\else\relax\forestloopbreak\fi}{on invalid={fake}{#2}}%
2658   }{%
2659     \forest@untilkey{\pgfmathifthenelse{#1}{"\noexpand\forestloopbreak"}{""}\pgfmathresult}{#2}%
2660   }%
2661 },
2662 while/.code 2 args={%
2663   \ifstrempy{#1}{%
2664     \forest@untilkey{\ifnum\forest@cn=0\relax\forestloopbreak\fi}{on invalid={fake}{#2}}%
2665   }{%
2666     \forest@untilkey{\pgfmathifthenelse{not(#1)}{"\noexpand\forestloopbreak"}{""}\pgfmathresult}{#2}%
2667   }%
2668 },
2669 do until/.code 2 args={%
2670   \ifstrempy{#1}{%
2671     \forest@dountilkey{\ifnum\forest@cn=0\else\relax\forestloopbreak\fi}{on invalid={fake}{#2}}%
2672   }{%
2673     \forest@dountilkey{\pgfmathifthenelse{#1}{"\noexpand\forestloopbreak"}{""}\pgfmathresult}{#2}%
2674   }%
2675 },
2676 do while/.code 2 args={%
2677   \ifstrempy{#1}{%
2678     \forest@dountilkey{\ifnum\forest@cn=0\relax\forestloopbreak\fi}{on invalid={fake}{#2}}%
2679   }{%
2680     \forest@dountilkey{\pgfmathifthenelse{not(#1)}{"\noexpand\forestloopbreak"}{""}\pgfmathresult}{#2}%
2681   }%
2682 },
2683 until nodewalk valid/.code 2 args={%
2684   \forest@untilkey{\forest@forthis{%
2685     \forest@nodewalk{on invalid={fake}{#1},TeX={\ifnum\forest@cn=0\relax\else\forestloopbreak\fi}}{}}}{#2}%
2686 },
2687 while nodewalk valid/.code 2 args={%
2688   \forest@untilkey{\forest@forthis{%
2689     \forest@nodewalk{on invalid={fake}{#1},TeX={\ifnum\forest@cn=0\relax\forestloopbreak\fi}}{}}}{#2}%
2690 },
2691 do until nodewalk valid/.code 2 args={%
2692   \forest@dountilkey{\forest@forthis{%
2693     \forest@nodewalk{on invalid={fake}{#1},TeX={\ifnum\forest@cn=0\relax\else\forestloopbreak\fi}}{}}}{#2}%
2694 },
2695 do while nodewalk valid/.code 2 args={%
2696   \forest@dountilkey{\forest@forthis{%
2697     \forest@nodewalk{on invalid={fake}{#1},TeX={\ifnum\forest@cn=0\relax\forestloopbreak\fi}}{}}}{#2}%
2698 },
2699 until nodewalk empty/.code 2 args={%
2700   \forest@untilkey{\forest@forthis{%
2701     \forest@nodewalk{on invalid={fake}{#1},TeX={\ifnum\forest@nodewalk@n=0\relax\forestloopbreak\fi}}{}}}{#2}%
2702 },
2703 while nodewalk empty/.code 2 args={%
2704   \forest@untilkey{\forest@forthis{%
2705     \forest@nodewalk{on invalid={fake}{#1},TeX={\ifnum\forest@nodewalk@n=0\relax\else\forestloopbreak\fi}}{}}}{#2}%
2706 },
2707 do until nodewalk empty/.code 2 args={%
2708   \forest@dountilkey{\forest@forthis{%

```

```

2709     \forest@nodewalk{on invalid={fake}{#1},TeX={\ifnum\forest@nodewalk@n=0\relax\forestloopbreak\fi}}{}}
2710 },
2711 do while nodewalk empty/.code 2 args={%
2712   \forest@dountilkey{\forest@forthis{%
2713     \forest@nodewalk{on invalid={fake}{#1},TeX={\ifnum\forest@nodewalk@n=0\relax\else\forestloopbreak\fi}}
2714   },
2715   break/.code={\forestloopBreak{#1}},
2716   break/.default=0,
2717 }
2718 \def\forest@repeatkey#1#2{%
2719   \safeRKloop
2720   \ifnum\safeRKloopn>#1\relax
2721   \csuse{safeRKbreak@\the\safeRKloop@depth true}%
2722   \fi
2723   \expandafter\unless\csname ifsafeRKbreak@\the\safeRKloop@depth\endcsname
2724   \pgfkeysalso{#2}%
2725   \safeRKrepeat
2726 }
2727 \def\forest@untilkey#1#2{% #1 = condition, #2 = keys
2728   \safeRKloop
2729   #1%
2730   \expandafter\unless\csname ifsafeRKbreak@\the\safeRKloop@depth\endcsname
2731   \pgfkeysalso{#2}%
2732   \safeRKrepeat
2733 }
2734 \def\forest@dountilkey#1#2{% #1 = condition, #2 = keys
2735   \safeRKloop
2736   \pgfkeysalso{#2}%
2737   #1%
2738   \expandafter\unless\csname ifsafeRKbreak@\the\safeRKloop@depth\endcsname
2739   \safeRKrepeat
2740 }
2741 \def\forestloopbreak{%
2742   \csname safeRKbreak@\the\safeRKloop@depth true\endcsname
2743 }
2744 \def\forestloopBreak#1{%
2745   \csname safeRKbreak@\number\numexpr\the\safeRKloop@depth-#1\relax true\endcsname
2746 }
2747 \def\forestloopcount{%
2748   \csname safeRKloopn@\number\numexpr\the\safeRKloop@depth\endcsname
2749 }
2750 \def\forestloopCount#1{%
2751   \csname safeRKloopn@\number\numexpr\the\safeRKloop@depth-#1\endcsname
2752 }
2753 \pgfmathdeclarefunction{forestloopcount}{1}{%
2754   \edef\pgfmathresult{\forestloopCount{\ifstrempy{#1}{0}{#1}}}%
2755 }
2756 \forest@copycommandkey{/forest/repeat}{/forest/nodewalk/repeat}
2757 \forest@copycommandkey{/forest/while}{/forest/nodewalk/while}
2758 \forest@copycommandkey{/forest/do while}{/forest/nodewalk/do while}
2759 \forest@copycommandkey{/forest/until}{/forest/nodewalk/until}
2760 \forest@copycommandkey{/forest/do until}{/forest/nodewalk/do until}
2761 \forest@copycommandkey{/forest/if}{/forest/nodewalk/if}
2762 \forest@copycommandkey{/forest/if nodewalk valid}{/forest/nodewalk/if nodewalk valid}
2763 %

```

## 6.4 Aggregate functions

```

2764 \forestset{
2765   aggregate postparse/.is choice,

```

```

2766 aggregate postparse/int/.code={%
2767   \let\forest@aggregate@pgfmathpostparse\forest@aggregate@pgfmathpostparse@toint},
2768 aggregate postparse/none/.code={%
2769   \let\forest@aggregate@pgfmathpostparse\relax},
2770 aggregate postparse/print/.code={%
2771   \let\forest@aggregate@pgfmathpostparse\forest@aggregate@pgfmathpostparse@print},
2772 aggregate postparse/macro/.code={%
2773   \let\forest@aggregate@pgfmathpostparse\forest@aggregate@pgfmathpostparse@usemacro},
2774 aggregate postparse macro/.store in=\forest@aggregate@pgfmathpostparse@macro,
2775 }
2776 \def\forest@aggregate@pgfmathpostparse@print{%
2777   \pgfmathprintnumberto{\pgfmathresult}{\pgfmathresult}%
2778 }
2779 \def\forest@aggregate@pgfmathpostparse@toint{%
2780   \expandafter\forest@split\expandafter{\pgfmathresult.}{.}\pgfmathresult\forest@temp
2781 }
2782 \def\forest@aggregate@pgfmathpostparse@usemacro{%
2783   \forest@aggregate@pgfmathpostparse@macro
2784 }
2785 \let\forest@aggregate@pgfmathpostparse\relax
2786 \pgfkeys{
2787   /handlers/.aggregate/.code n args=4{%
2788     % #1 = start value
2789     % #2 = pgfmath expression for every step
2790     % #3 = pgfmath expression for after walk
2791     % #4 = nodewalk
2792     \edef\forest@marshal{%
2793       \unexpanded{\forest@aggregate{#1}{#2}{#3}{#4}}{\pgfkeyscurrentpath}%
2794     }\forest@marshal
2795   },
2796   /handlers/.sum/.style 2 args={/handlers/.aggregate={0}{(##1)+(##2)}{##1}{##2}},
2797   /handlers/.count/.style={/handlers/.aggregate={0}{1+(##1)}{##1}{##1}},
2798   /handlers/.average/.style 2 args={/handlers/.aggregate={0}{(##1)+(##2)}{##1/\forestregister{aggregate n}}{##2}},
2799   /handlers/.product/.style 2 args={/handlers/.aggregate={1}{(##1)*(##2)}{##1}{##2}},
2800   /handlers/.min/.style 2 args={/handlers/.aggregate={}\{min(##1,##2)\}{##1}{##2}},
2801   /handlers/.max/.style 2 args={/handlers/.aggregate={}\{max(##1,##2)\}{##1}{##2}},
2802   /forest/declare count register={aggregate n},
2803 }
2804
2805 \def\forest@aggregate#1#2#3#4#5{%
2806   % #5 = current path
2807   \def\forest@aggregate@result{#1}%
2808   \forest@forthis{%
2809     \foreststrset{aggregate n}{0}%
2810     \forest@nodewalk{#4}{%
2811       TeX={%
2812         \forestreset{aggregate n}{\number\numexpr\forestrv{aggregate n}+1}%
2813         \def\forest@marshal##1{\pgfmathparse{#2}}%
2814         \expandafter\forest@marshal\expandafter{\forest@aggregate@result}%
2815         \let\forest@aggregate@result\pgfmathresult
2816       }%
2817     }%
2818   }%
2819   \def\forest@marshal##1{\pgfmathparse{#3}}%
2820   \expandafter\forest@marshal\expandafter{\forest@aggregate@result}%
2821   \let\forest@aggregate@result\pgfmathresult
2822   \let\forest@temp@pgfmathpostparse\pgfmathpostparse
2823   \let\pgfmathpostparse\forest@aggregate@pgfmathpostparse
2824   \pgfmathqparse{\forest@aggregate@result pt}%
2825   \let\pgfmathpostparse\forest@temp@pgfmathpostparse
2826   \let\forest@aggregate@result\pgfmathresult

```

```

2827 \pgfkeysalso{/#5/.expand once=\forest@aggregate@result}%
2828 }

```

#### 6.4.1 pgfmath extensions

```

2829 \pgfmathdeclarefunction{strequal}{2}{%
2830 \ifstrequal{#1}{#2}{\def\pgfmathresult{1}}{\def\pgfmathresult{0}}%
2831 }
2832 \pgfmathdeclarefunction{instr}{2}{%
2833 \pgfutil@in@{#1}{#2}%
2834 \ifpgfutil@in@\def\pgfmathresult{1}\else\def\pgfmathresult{0}\fi
2835 }
2836 \pgfmathdeclarefunction{strcat}{...}{%
2837 \edef\pgfmathresult{\forest@strip@braces{#1}}%
2838 }
2839 \pgfmathdeclarefunction{min_s}{2}{% #1 = node, #2 = context node (for growth rotation)
2840 \forest@forthis{%
2841 \forest@nameandgo{#1}%
2842 \forest@compute@minmax@ls{#2}%
2843 \pgfmathsetmacro\pgfmathresult{\forestove{min@s}}%
2844 }%
2845 }
2846 \pgfmathdeclarefunction{min_l}{2}{% #1 = node, #2 = context node (for growth rotation)
2847 \forest@forthis{%
2848 \forest@nameandgo{#1}%
2849 \forest@compute@minmax@ls{#2}%
2850 \pgfmathsetmacro\pgfmathresult{\forestove{min@l}}%
2851 }%
2852 }
2853 \pgfmathdeclarefunction{max_s}{2}{% #1 = node, #2 = context node (for growth rotation)
2854 \forest@forthis{%
2855 \forest@nameandgo{#1}%
2856 \forest@compute@minmax@ls{#2}%
2857 \pgfmathsetmacro\pgfmathresult{\forestove{max@s}}%
2858 }%
2859 }
2860 \pgfmathdeclarefunction{max_l}{2}{% #1 = node, #2 = context node (for growth rotation)
2861 \forest@forthis{%
2862 \forest@nameandgo{#1}%
2863 \forest@compute@minmax@ls{#2}%
2864 \pgfmathsetmacro\pgfmathresult{\forestove{max@l}}%
2865 }%
2866 }
2867 \def\forest@compute@minmax@ls#1{% #1 = nodewalk; in the context of which node?
2868 {%
2869 \pgftransformreset
2870 \forest@forthis{%
2871 \forest@nameandgo{#1}%
2872 \pgftransformrotate{-\forestove{grow}}%
2873 }%
2874 \forestoget{min x}\forest@temp@minx
2875 \forestoget{min y}\forest@temp@miny
2876 \forestoget{max x}\forest@temp@maxx
2877 \forestoget{max y}\forest@temp@maxy
2878 \pgfpointransformed{\pgfqpoint{\forest@temp@minx}{\forest@temp@miny}}%
2879 \forestoaset{min@l}{\the\pgf@x}%
2880 \forestoaset{min@s}{\the\pgf@y}%
2881 \forestoaset{max@l}{\the\pgf@x}%
2882 \forestoaset{max@s}{\the\pgf@y}%
2883 \pgfpointransformed{\pgfqpoint{\forest@temp@minx}{\forest@temp@maxy}}%
2884 \ifdim\pgf@x<\forestove{min@l}\relax\forestoaset{min@l}{\the\pgf@x}\fi
2885 \ifdim\pgf@y<\forestove{min@s}\relax\forestoaset{min@s}{\the\pgf@y}\fi

```

```

2886 \ifdim\pgf@x>\forestove{max@1}\relax\forestoeset{max@1}{\the\pgf@x}\fi
2887 \ifdim\pgf@y>\forestove{max@s}\relax\forestoeset{max@s}{\the\pgf@y}\fi
2888 \pgfpointransformed{\pgfqpoint{\forest@temp@maxx}{\forest@temp@miny}}%
2889 \ifdim\pgf@x<\forestove{min@1}\relax\forestoeset{min@1}{\the\pgf@x}\fi
2890 \ifdim\pgf@y<\forestove{min@s}\relax\forestoeset{min@s}{\the\pgf@y}\fi
2891 \ifdim\pgf@x>\forestove{max@1}\relax\forestoeset{max@1}{\the\pgf@x}\fi
2892 \ifdim\pgf@y>\forestove{max@s}\relax\forestoeset{max@s}{\the\pgf@y}\fi
2893 \pgfpointransformed{\pgfqpoint{\forest@temp@maxx}{\forest@temp@maxy}}%
2894 \ifdim\pgf@x<\forestove{min@1}\relax\forestoeset{min@1}{\the\pgf@x}\fi
2895 \ifdim\pgf@y<\forestove{min@s}\relax\forestoeset{min@s}{\the\pgf@y}\fi
2896 \ifdim\pgf@x>\forestove{max@1}\relax\forestoeset{max@1}{\the\pgf@x}\fi
2897 \ifdim\pgf@y>\forestove{max@s}\relax\forestoeset{max@s}{\the\pgf@y}\fi
2898 % smuggle out
2899 \edef\forest@marshal{%
2900   \noexpand\forestoeset{min@1}{\forestove{min@1}}%
2901   \noexpand\forestoeset{min@s}{\forestove{min@s}}%
2902   \noexpand\forestoeset{max@1}{\forestove{max@1}}%
2903   \noexpand\forestoeset{max@s}{\forestove{max@s}}%
2904 } \expandafter
2905 } \forest@marshal
2906 }
2907 \def\forest@pgfmathhelper@attribute@toks#1#2{%
2908   \forest@forthis{%
2909     \forest@nameandgo{#1}%
2910     \ifnum\forest@cn=0
2911       \def\pgfmathresult{}%
2912     \else
2913       \forestoget{#2}\pgfmathresult
2914     \fi
2915   }%
2916 }
2917 \def\forest@pgfmathhelper@attribute@dimen#1#2{%
2918   \forest@forthis{%
2919     \forest@nameandgo{#1}%
2920     \ifnum\forest@cn=0
2921       \def\pgfmathresult{Opt}%
2922     \else
2923       \forestoget{#2}\forest@temp
2924       \pgfmathparse{+\forest@temp}%
2925     \fi
2926   }%
2927 }
2928 \def\forest@pgfmathhelper@attribute@count#1#2{%
2929   \forest@forthis{%
2930     \forest@nameandgo{#1}%
2931     \ifnum\forest@cn=0
2932       \def\pgfmathresult{0}%
2933     \else
2934       \forestoget{#2}\forest@temp
2935       \pgfmathtruncatemacro\pgfmathresult{\forest@temp}%
2936     \fi
2937   }%
2938 }
2939 \pgfmathdeclarefunction{id}{1}{%
2940   \forest@forthis{%
2941     \forest@nameandgo{#1}%
2942     \let\pgfmathresult\forest@cn
2943   }%
2944 }
2945 \forestset{%
2946   if id/.code n args={3}{%

```

```

2947 \ifnum#1=\forest@cn\relax
2948 \pgfkeysalso{#2}%
2949 \else
2950 \pgfkeysalso{#3}%
2951 \fi
2952 },
2953 where id/.style n args={3}{for tree={if id={#1}{#2}{#3}}}
2954 }

```

## 6.5 Nodewalk

Setup machinery.

```

2955 \def\forest@nodewalk@n{0}
2956 \def\forest@nodewalk@historyback{0,}
2957 \def\forest@nodewalk@historyforward{0,}
2958 \def\forest@nodewalk@origin{0}
2959 \def\forest@nodewalk@config@everystep@independent@before#1{% #1 = every step keylist
2960 \forestset{every step}{#1}%
2961 }
2962 \def\forest@nodewalk@config@everystep@independent@after{%
2963 \noexpand\forestset{every step}{\forestrv{every step}}}%
2964 }
2965 \def\forest@nodewalk@config@history@independent@before{%
2966 \def\forest@nodewalk@n{0}%
2967 \edef\forest@nodewalk@origin{\forest@cn}%
2968 \def\forest@nodewalk@historyback{0,}%
2969 \def\forest@nodewalk@historyforward{0,}%
2970 }
2971 \def\forest@nodewalk@config@history@independent@after{%
2972 \edef\noexpand\forest@nodewalk@n{\expandonce{\forest@nodewalk@n}}%
2973 \edef\noexpand\forest@nodewalk@origin{\expandonce{\forest@nodewalk@origin}}%
2974 \edef\noexpand\forest@nodewalk@historyback{\expandonce{\forest@nodewalk@historyback}}%
2975 \edef\noexpand\forest@nodewalk@historyforward{\expandonce{\forest@nodewalk@historyforward}}%
2976 }
2977 \def\forest@nodewalk@config@everystep@shared@before#1{% #1 = every step keylist
2978 \def\forest@nodewalk@config@everystep@shared@after{}
2979 \def\forest@nodewalk@config@history@shared@before{}
2980 \def\forest@nodewalk@config@history@shared@after{}
2981 \def\forest@nodewalk@config@everystep@inherited@before#1{} #1 = every step keylist
2982 \let\forest@nodewalk@config@everystep@inherited@after\forest@nodewalk@config@everystep@independent@after
2983 \def\forest@nodewalk@config@history@inherited@before{}
2984 \let\forest@nodewalk@config@history@inherited@after\forest@nodewalk@config@history@independent@after
2985 \def\forest@nodewalk#1#2{% #1 = nodewalk, #2 = every step keylist
2986 \def\forest@nodewalk@config@everystep@method{independent}%
2987 \def\forest@nodewalk@config@history@method{independent}%
2988 \def\forest@nodewalk@config@oninvalid{inherited}%
2989 \forest@saveandrestoremacro\forest@nodewalk@oninvalid{%
2990 \forest@Nodewalk{#1}{#2}%
2991 }%
2992 }
2993 \def\forest@Nodewalk#1#2{% #1 = nodewalk, #2 = every step keylist
2994 \edef\forest@marshal{%
2995 \noexpand\pgfqkeys{/forest/nodewalk}{\unexpanded{#1}}%
2996 \csname forest@nodewalk@config@everystep@\forest@nodewalk@config@everystep@method @after\endcsname
2997 \csname forest@nodewalk@config@history@\forest@nodewalk@config@history@method @after\endcsname
2998 }%
2999 \csname forest@nodewalk@config@everystep@\forest@nodewalk@config@everystep@method @before\endcsname{#2}%
3000 \csname forest@nodewalk@config@history@\forest@nodewalk@config@history@method @before\endcsname
3001 \forest@nodewalk@fakefalse
3002 \forest@marshal
3003 }

```

```

3004 \pgfmathdeclarefunction{valid}{1}{%
3005   \forest@forthis{%
3006     \forest@nameandgo{#1}%
3007     \edef\pgfmathresult{\ifnum\forest@cn=0 0\else 1\fi}%
3008   }%
3009 }
3010 \pgfmathdeclarefunction{invalid}{1}{%
3011   \forest@forthis{%
3012     \forest@nameandgo{#1}%
3013     \edef\pgfmathresult{\ifnum\forest@cn=0 1\else 0\fi}%
3014   }%
3015 }
3016 \newif\ifforest@nodewalk@fake
3017 \def\forest@nodewalk@oninvalid{error}
3018 \def\forest@nodewalk@makestep{%
3019   \ifnum\forest@cn=0
3020     \csname forest@nodewalk@makestep@oninvalid@\forest@nodewalk@config@oninvalid\endcsname
3021   \else
3022     \forest@nodewalk@makestep@
3023   \fi
3024 }
3025 \def\forest@nodewalk@makestep@oninvalid@step{\forest@nodewalk@makestep@}
3026 \def\forest@nodewalk@makestep@oninvalid@error{\PackageError{forest}{nodewalk stepped to the invalid node; sta
3027 \let\forest@nodewalk@makestep@oninvalid@fake\relax
3028 \def\forest@nodewalk@makestep@oninvalid@compatfake{%
3029   \forest@deprecated{last step in stack "\forest@nodewalk@currentstepname", which stepped on an invalid node;
3030 }%
3031 \def\forest@nodewalk@makestep@oninvalid@inherited{\csname forest@nodewalk@makestep@oninvalid@\forest@nodewalk
3032 \def\forest@nodewalk@makestep@{%
3033   \ifforest@nodewalk@fake
3034   \else
3035     \edef\forest@nodewalk@n{\number\numexpr\forest@nodewalk@n+1}%
3036     \epreto\forest@nodewalk@historyback{\forest@cn,}%
3037     \def\forest@nodewalk@historyforward{0,}%
3038     \ifforestdebug\forest@nodewalk@makestep@debug\fi
3039     \forest@process@keylist@register{every step}%
3040   \fi
3041 }
3042 \def\forest@nodewalk@makestep@debug{%
3043   \edef\forest@marshal{%
3044     \noexpand\typeout{\ifforest@nodewalk@fake fake \fi "\forest@nodewalk@currentstepname" step to node id=\fo
3045   }\forest@marshal
3046 }%
3047 \forestset{
3048   debug/.is if=forestdebug,
3049 }
3050 \def\forest@handlers@savecurrentpath{%
3051   \edef\pgfkeyscurrentkey{\pgfkeyscurrentpath}%
3052   \let\forest@currentkey\pgfkeyscurrentkey
3053   \pgfkeys@split@path
3054   \edef\forest@currentpath{\pgfkeyscurrentpath}%
3055   \let\forest@currentname\pgfkeyscurrentname
3056 }
3057 \pgfkeys{/handlers/save current path/.code={\forest@handlers@savecurrentpath}}
3058 \newif\ifforest@nodewalkstephandler@style
3059 \newif\ifforest@nodewalkstephandler@autostep
3060 \newif\ifforest@nodewalkstephandler@stripfakesteps
3061 \newif\ifforest@nodewalkstephandler@muststartatvalidnode
3062 \newif\ifforest@nodewalkstephandler@makefor
3063 \let\forest@nodewalkstephandler@styletrueorfalse\forest@nodewalkstephandler@stylefalse
3064 \def\forest@nodewalk@currentstepname{}

```



```

3065 \forestset{
3066 /forest/define@step/style/.is if=forest@nodewalkstephandler@style,
3067 /forest/define@step/autostep/.is if=forest@nodewalkstephandler@autostep,
3068 % the following is useful because some macros use grouping (by \forest@forthis or similar) and therefore, a
3069 /forest/define@step/strip fake steps/.is if=forest@nodewalkstephandler@stripfakesteps,
3070 % this can never happen with autosteps ...
3071 /forest/define@step/autostep/.append code={%
3072   \ifforest@nodewalkstephandler@autostep
3073     \forest@nodewalkstephandler@stripfakestepsfalse
3074   \fi
3075 },
3076 /forest/define@step/must start at valid node/.is if=forest@nodewalkstephandler@muststartatvalidnode,
3077 /forest/define@step/n args/.store in=\forest@nodewalkstephandler@nargs,
3078 /forest/define@step/make for/.is if=forest@nodewalkstephandler@makefor,
3079 /forest/define@step/@bare/.style={strip fake steps=false,must start at valid node=false,make for=false},
3080 define long step/.code n args=3{%
3081   \forest@nodewalkstephandler@styletrueorfalse % true for end users; but in the package, most of steps are
3082   \forest@nodewalkstephandler@autostepfalse
3083   \forest@nodewalkstephandler@stripfakestepstrue
3084   \forest@nodewalkstephandler@muststartatvalidnodetrue % most steps can only start at a valid node
3085   \forest@nodewalkstephandler@makefortrue % make for prefix?
3086   \def\forest@nodewalkstephandler@nargs{0}%
3087   \pgfqkeys{/forest/define@step}{#2}%
3088   \forest@temp@toks{#3}% handler code
3089   \ifforest@nodewalkstephandler@style
3090     \expandafter\forest@temp@toks\expandafter{%
3091       \expandafter\pgfqkeysalso\expandafter{\the\forest@temp@toks}%
3092     }%
3093   \fi
3094   \ifforest@nodewalkstephandler@autostep
3095     \apptotoks\forest@temp@toks{\forest@nodewalk@makestep}%
3096   \fi
3097   \ifforest@nodewalkstephandler@stripfakesteps
3098     \expandafter\forest@temp@toks\expandafter{\expandafter\forest@nodewalk@stripfakesteps\expandafter{\the\
3099   \fi
3100   \ifforest@nodewalkstephandler@muststartatvalidnode
3101     \edef\forest@marshal{%
3102       \noexpand\forest@temp@toks{%
3103         \unexpanded{%
3104           \ifnum\forest@cn=0
3105             \csname forest@nodewalk@start@oninvalid\forest@nodewalk@oninvalid\endcsname{#1}%
3106           \else
3107             }%
3108           \noexpand\@escapeif{\the\forest@temp@toks}%
3109           \noexpand\fi
3110         }%
3111       }\forest@marshal
3112     \fi
3113     \pretotoks\forest@temp@toks{\appto\forest@nodewalk@currentstepname{,#1}}%
3114     \expandafter\forest@temp@toks\expandafter{\expandafter\forest@saveandrestoremacro\expandafter\forest@node
3115   \ifforestdebug
3116     \epretotoks\forest@temp@toks{\noexpand\typeout{Starting step "#1" from id=\forest@cn
3117       \ifnum\forest@nodewalkstephandler@nargs>0 with args #####1\fi
3118       \ifnum\forest@nodewalkstephandler@nargs>1 ,####2\fi
3119       \ifnum\forest@nodewalkstephandler@nargs>2 ,####3\fi
3120       \ifnum\forest@nodewalkstephandler@nargs>3 ,####4\fi
3121       \ifnum\forest@nodewalkstephandler@nargs>4 ,####5\fi
3122       \ifnum\forest@nodewalkstephandler@nargs>5 ,####6\fi
3123       \ifnum\forest@nodewalkstephandler@nargs>6 ,####7\fi
3124       \ifnum\forest@nodewalkstephandler@nargs>7 ,####8\fi
3125       \ifnum\forest@nodewalkstephandler@nargs>8 ,####9\fi

```

```

3126     }}%
3127 \fi
3128 \def\forest@temp{/forest/nodewalk/#1/.code}%
3129 \ifnum\forest@nodewalkstephandler@nargs<2
3130   \eappto\forest@temp{=}
3131 \else\ifnum\forest@nodewalkstephandler@nargs=2
3132   \eappto\forest@temp{ 2 args=}
3133 \else
3134   \eappto\forest@temp{ n args={\forest@nodewalkstephandler@nargs}}
3135 \fi\fi
3136 \eappto\forest@temp{{\the\forest@temp@toks}}
3137 \expandafter\pgfkeysalso\expandafter{\forest@temp}%
3138 \ifforest@nodewalkstephandler@makefor
3139   \ifnum\forest@nodewalkstephandler@nargs=0
3140     \forestset{%
3141       for #1/.code={\forest@forstepwrapper{#1}{##1}},
3142     }%
3143 \else\ifnum\forest@nodewalkstephandler@nargs=1
3144   \forestset{%
3145     for #1/.code 2 args={\forest@forstepwrapper{#1={##1}}{##2}},
3146   }%
3147 \else
3148   \forestset{%
3149     for #1/.code n args/.expanded=%
3150       {\number\numexpr\forest@nodewalkstephandler@nargs+1}%
3151       {\noexpand\forest@forstepwrapper{#1\ifnum\forest@nodewalkstephandler@nargs>0=\fi\forest@util@narg
3152     }}%
3153 \fi\fi
3154 \fi
3155 },
3156 }
3157 \pgfkeys{/handlers}{
3158   .nodewalk style/.code={\forest@handlers@savecurrentpath\pgfkeysalso{%
3159     \forest@currentpath/nodewalk/\forest@currentname/.style={#1}%
3160   }},
3161 }

```

\forest@forstepwrapper is defined so that it can be changed by compat to create unfailable spatial propagators from v1.0.

```

3162 \def\forest@forstepwrapper#1#2{\forest@forthis{\forest@nodewalk{#1}{#2}}}
3163 \def\forest@util@nargs#1#2#3{% #1 = prefix (#, ##, ...), #2 = n args, #3=start; returns {#start}{#start+1}...
3164   \ifnum#2>0 {#1\number\numexpr#3+1}\fi
3165   \ifnum#2>1 {#1\number\numexpr#3+2}\fi
3166   \ifnum#2>2 {#1\number\numexpr#3+3}\fi
3167   \ifnum#2>3 {#1\number\numexpr#3+4}\fi
3168   \ifnum#2>4 {#1\number\numexpr#3+5}\fi
3169   \ifnum#2>5 {#1\number\numexpr#3+6}\fi
3170   \ifnum#2>6 {#1\number\numexpr#3+7}\fi
3171   \ifnum#2>7 {#1\number\numexpr#3+8}\fi
3172   \ifnum#2>8 {#1\number\numexpr#3+9}\fi
3173 }
3174 \def\forest@nodewalk@start@oninvalid@fake#1{}
3175 \def\forest@nodewalk@start@oninvalid@real#1{}
3176 \def\forest@nodewalk@start@oninvalid@error#1{\PackageError{forest}{nodewalk step "#1" cannot start at the inv

```

Define long-form single-step walks.

```

3177 \forestset{
3178   define long step={current}{autostep}{},
3179   define long step={next}{autostep}{\edef\forest@cn{\forestove{@next}}},
3180   define long step={previous}{autostep}{\edef\forest@cn{\forestove{@previous}}},
3181   define long step={parent}{autostep}{\edef\forest@cn{\forestove{@parent}}},

```

```

3182 define long step={first}{autostep}{\edef\forest@cn{\forestove{@first}}},
3183 define long step={last}{autostep}{\edef\forest@cn{\forestove{@last}}},
3184 define long step={sibling}{autostep}{%
3185   \edef\forest@cn{%
3186     \ifnum\forestove{@previous}=0
3187     \forestove{@next}%
3188     \else
3189     \forestove{@previous}%
3190     \fi
3191   }%
3192 },
3193 define long step={next node}{autostep}{\edef\forest@cn{\forest@node@linearnextid}},
3194 define long step={previous node}{autostep}{\edef\forest@cn{\forest@node@linearpreviousid}},
3195 define long step={first leaf}{autostep}{%
3196   \safeloop
3197   \edef\forest@cn{\forestove{@first}}%
3198   \unless\ifnum\forestove{@first}=0
3199   \saferepeat
3200 },
3201 define long step={first leaf'}{autostep}{%
3202   \safeloop
3203   \unless\ifnum\forestove{@first}=0
3204   \edef\forest@cn{\forestove{@first}}%
3205   \saferepeat
3206 },
3207 define long step={last leaf}{autostep}{%
3208   \safeloop
3209   \edef\forest@cn{\forestove{@last}}%
3210   \unless\ifnum\forestove{@last}=0
3211   \saferepeat
3212 },
3213 define long step={last leaf'}{autostep}{%
3214   \safeloop
3215   \unless\ifnum\forestove{@last}=0
3216   \edef\forest@cn{\forestove{@last}}%
3217   \saferepeat
3218 },
3219 define long step={next leaf}{style,strip fake steps=false}{group={do until={n_children()==0}{next node}}},
3220 define long step={previous leaf}{style,strip fake steps=false}{group={do until={n_children()==0}{previous node}}},
3221 define long step={next on tier}{autostep}{%
3222   \def\forest@temp{#1}%
3223   \ifx\forest@temp\pgfkeysnovalue@text
3224     \foresttoget{tier}\forest@nodewalk@giventier
3225   \else
3226     \def\forest@nodewalk@giventier{#1}%
3227   \fi
3228   \edef\forest@cn{\forest@node@linearnextnotdescendantid}%
3229   \safeloop
3230     \foresttoget{tier}\forest@nodewalk@tier
3231   \unless\ifx\forest@nodewalk@tier\forest@nodewalk@giventier
3232     \edef\forest@cn{\forest@node@linearnextid}%
3233   \saferepeat
3234 },
3235 define long step={previous on tier}{autostep}{%
3236   \def\forest@temp{#1}%
3237   \ifx\forest@temp\pgfkeysnovalue@text
3238     \foresttoget{tier}\forest@nodewalk@giventier
3239   \else
3240     \def\forest@nodewalk@giventier{#1}%
3241   \fi
3242   \safeloop

```

```

3243     \edef\forest@cn{\forest@node@linearpreviousid}%
3244     \forestoget{tier}\forest@nodewalk@tier
3245     \unless\ifx\forest@nodewalk@tier\forest@nodewalk@giventier
3246     \saferepeat
3247 },
3248 %
3249 define long step={root}{autostep,must start at valid node=false}{%
3250     \edef\forest@cn{\forest@node@rootid}},
3251 define long step={root'}{autostep,must start at valid node=false}{%
3252     \forest0ifdefined{\forest@root}{@parent}{\edef\forest@cn{\forest@root}}{\edef\forest@cn{0}}%
3253 },
3254 define long step={origin}{autostep,must start at valid node=false}{\edef\forest@cn{\forest@nodewalk@origin}}
3255 %
3256 define long step={n}{autostep,n args=1}{%
3257     \pgfmathtruncatemacro\forest@temp@n{#1}%
3258     \edef\forest@cn{\forest@node@nthchildid{\forest@temp@n}}%
3259 },
3260 define long step={n}{autostep,make for=false,n args=1}{%
3261     % Yes, twice. ;- )
3262     % n=1 and n(ext)
3263     \def\forest@nodewalk@temp{#1}%
3264     \ifx\forest@nodewalk@temp\pgfkeysnovalue@text
3265         \edef\forest@cn{\forestove{@next}}%
3266     \else
3267         \pgfmathtruncatemacro\forest@temp@n{#1}%
3268         \edef\forest@cn{\forest@node@nthchildid{\forest@temp@n}}%
3269     \fi
3270 },
3271 define long step={n'}{autostep,n args=1}{%
3272     \pgfmathtruncatemacro\forest@temp@n{#1}%
3273     \edef\forest@cn{\forest@node@nbarthchildid{\forest@temp@n}}%
3274 },
3275 define long step={to tier}{autostep,n args=1}{%
3276     \def\forest@nodewalk@giventier{#1}%
3277     \safeloop
3278     \forestoget{tier}\forest@nodewalk@tier
3279     \unless\ifx\forest@nodewalk@tier\forest@nodewalk@giventier
3280     \forestoget{@parent}\forest@cn
3281     \saferepeat
3282 },
3283 %
3284 define long step={name}{autostep,n args=1,must start at valid node=false}{%
3285     \edef\forest@cn{%
3286         \forest@node@ifnamedefined{#1}{\forest@node@Nametoid{#1}}{0}}%
3287     }%
3288 },
3289 define long step={id}{autostep,n args=1,must start at valid node=false}{%
3290     \forest0ifdefined{#1}{@parent}{\edef\forest@cn{#1}}{\edef\forest@cn{0}}%
3291 },
3292 define long step={Nodewalk}{n args=3,@bare}{% #1 = config, #2 = nodewalk
3293     \def\forest@nodewalk@config@everystep@method{independent}%
3294     \def\forest@nodewalk@config@history@method{shared}%
3295     \def\forest@nodewalk@config@oninvalid{inherited}%
3296     \forest@saveandrestoremacro\forest@nodewalk@oninvalid{%
3297         \pgfqkeys{/forest/nodewalk@config}{#1}%
3298         \forest@Nodewalk{#2}{#3}%
3299     }%
3300 },
3301 define long step={nodewalk}{n args=2,@bare}{% #1 = nodewalk, #2 = every step
3302     \forest@nodewalk{#1}{#2}%
3303 },

```

```

3304 define long step={nodewalk'}{n args=1,@bare}{% #1 = nodewalk, #2 = every step
3305   \def\forest@nodewalk@config@everystep@method{inherited}%
3306   \def\forest@nodewalk@config@history@method{independent}%
3307   \def\forest@nodewalk@config@oninvalid{inherited}%
3308   \forest@saveandrestoremacro\forest@nodewalk@oninvalid{%
3309     \forest@Nodewalk{#1}{}%
3310   }%
3311 },
3312 % these must be defined explicitly, as prefix "for" normally introduces the every-step keylist
3313 for nodewalk/.code 2 args={%
3314   \forest@forthis{\forest@nodewalk{#1}{#2}},
3315 for nodewalk'/.code={%
3316   \def\forest@nodewalk@config@everystep@method{inherited}%
3317   \def\forest@nodewalk@config@history@method{independent}%
3318   \def\forest@nodewalk@config@oninvalid{inherited}%
3319   \forest@saveandrestoremacro\forest@nodewalk@oninvalid{%
3320     \forest@forthis{\forest@Nodewalk{#1}{}}%
3321   }%
3322 },
3323 for Nodewalk/.code n args=3{% #1 = config, #2 = nodewalk, #3 = every-step
3324   \def\forest@nodewalk@config@everystep@method{inherited}%
3325   \def\forest@nodewalk@config@history@method{independent}%
3326   \def\forest@nodewalk@config@oninvalid{inherited}%
3327   \forest@saveandrestoremacro\forest@nodewalk@oninvalid{%
3328     \pgfqkeys{/forest/nodewalk@config}{#1}%
3329     \forest@forthis{\forest@Nodewalk{#2}{#3}}%
3330   }%
3331 },
3332 copy command key={/forest/for nodewalk}{/forest/nodewalk/for nodewalk},
3333 copy command key={/forest/for nodewalk'}{/forest/nodewalk/for nodewalk'},
3334 copy command key={/forest/for Nodewalk}{/forest/nodewalk/for Nodewalk},
3335 declare keylist register=every step,
3336 every step'={},
3337 %%% begin nodewalk config
3338 nodewalk@config/.cd,
3339 every@step/.is choice,
3340 every@step/independent/.code={},
3341 every@step/inherited/.code={},
3342 every@step/shared/.code={},
3343 every step/.store in=\forest@nodewalk@config@everystep@method,
3344 every step/.prefix style={every@step=#1},
3345 @history/.is choice,
3346 @history/independent/.code={},
3347 @history/inherited/.code={},
3348 @history/shared/.code={},
3349 history/.store in=\forest@nodewalk@config@history@method,
3350 history/.prefix style={@history=#1},
3351 on@invalid/.is choice,
3352 on@invalid/error/.code={},
3353 on@invalid/fake/.code={},
3354 on@invalid/step/.code={},
3355 on@invalid/inherited/.code={},
3356 on invalid/.store in=\forest@nodewalk@config@oninvalid,
3357 on invalid/.prefix style={on@invalid=#1},
3358 %%% end nodewalk config
3359 }
3360 \forestset{
3361   define long step={branch}{n args=1,@bare,style}{%
3362     branch@@build/.style={branch@build={##1}{}},
3363     @branch={#1},
3364   },

```

```

3365 define long step={branch'}{n args=1,@bare,style}{%
3366   branch@@build/.style/.expanded={branch@build={###1}}{\forestregister{every step}},
3367   @branch={#1},
3368 },
3369 nodewalk/@branch/.style={
3370   TeX={\forest@temp@toks{}},
3371   split/.process args={r}{#1}{,}{branch@@build},
3372   branch@do,
3373 },
3374 nodewalk/branch@build/.code 2 args={% #1 = nodewalk for this branch, #2 = perhaps every step keylist
3375   \ifstrempy{#1}{}{%
3376     \expandafter\ifstrempy\expandafter{\the\forest@temp@toks}{%
3377       \edef\forest@marshal{%
3378         \noexpand\forest@temp@toks{for nodewalk={\unexpanded{#1}}{\forestregister{every step}}}%
3379       }\forest@marshal
3380     }{%
3381       \edef\forest@marshal{%
3382         \noexpand\forest@temp@toks{for nodewalk={\unexpanded{#1}}{#2\the\forest@temp@toks}}%
3383       }\forest@marshal
3384     }%
3385   }%
3386 },
3387 nodewalk/branch@do/.code={%
3388   \edef\forest@marshal{%
3389     \noexpand\pgfkeysalso{fake={\the\forest@temp@toks}}%
3390   }\forest@marshal
3391 },
3392 define long step={group}{autostep}{\forest@go{#1}},
3393 nodewalk/fake/.code={%
3394   \forest@saveandrestoreifcs{forest@nodewalk@fake}{%
3395     \forest@nodewalk@faketrue
3396     \pgfkeysalso{#1}%
3397   }%
3398 },
3399 nodewalk/real/.code={%
3400   \forest@saveandrestoreifcs{forest@nodewalk@fake}{%
3401     \forest@nodewalk@fakefalse
3402     \pgfkeysalso{#1}%
3403   }%
3404 },
3405 define long step={filter}{n args=2,@bare,style}{% #1 = nodewalk, #2 = condition
3406   nodewalk/.expanded={\unexpanded{#1}}{if={\unexpanded{#2}}{\forestregister{every step}}{}}{}}
3407 },
3408 on@invalid/.is choice,
3409 on@invalid/error/.code={},
3410 on@invalid/fake/.code={},
3411 on@invalid/step/.code={},
3412 on invalid/.code 2 args={%
3413   \pgfkeysalso{/forest/on@invalid={#1}}%
3414   \forest@saveandrestoremacro\forest@nodewalk@oninvalid{%
3415     \def\forest@nodewalk@oninvalid{#1}%
3416     \pgfkeysalso{#2}%
3417   }%
3418 },
3419 define long step={strip fake steps}{n args=1,@bare}{%
3420   \forest@nodewalk@stripfakesteps{\pgfkeysalso{#1}}},
3421 define long step={walk back}{n args=1,@bare}{%
3422   \pgfmathtruncatemacro\forest@temp@n{#1}%
3423   \forest@nodewalk@walklist{\forest@nodewalk@historyforward}{\forest@nodewalk@historyback}{\ifnum\forest@cn
3424   \forest@nodewalk@back@updatehistory
3425 },

```

```

3426 nodewalk/walk back/.default=1,
3427 define long step={jump back}{n args=1,@bare}{%
3428   \pgfmathtruncatemacro\forest@temp@n{(#1)+\ifnum\forest@cn=0 0\else1\fi}%
3429   \forest@nodewalk@walklist{\forest@nodewalk@historyforward}{\forest@nodewalk@historyback}{\forest@temp@n-1}
3430   \forest@nodewalk@back@updatehistory
3431 },
3432 nodewalk/jump back/.default=1,
3433 define long step={back}{n args=1,@bare}{%
3434   \pgfmathtruncatemacro\forest@temp@n{#1}%
3435   \forest@nodewalk@walklist{\forest@nodewalk@historyforward}{\forest@nodewalk@historyback}{\ifnum\forest@cn=0 0\else1\fi}
3436   \forest@nodewalk@back@updatehistory
3437 },
3438 nodewalk/back/.default=1,
3439 define long step={walk forward}{n args=1,@bare}{%
3440   \pgfmathtruncatemacro\forest@temp@n{#1}%
3441   \forest@nodewalk@walklist{\forest@nodewalk@historyback}{\forest@nodewalk@historyforward}{0}{\forest@temp@n-1}
3442   \forest@nodewalk@forward@updatehistory
3443 },
3444 nodewalk/walk forward/.default=1,
3445 define long step={jump forward}{n args=1,@bare}{%
3446   \pgfmathtruncatemacro\forest@temp@n{#1}%
3447   \forest@nodewalk@walklist{\forest@nodewalk@historyback}{\forest@nodewalk@historyforward}{\forest@temp@n-1}
3448   \forest@nodewalk@forward@updatehistory
3449 },
3450 nodewalk/jump forward/.default=1,
3451 define long step={forward}{n args=1,@bare}{%
3452   \pgfmathtruncatemacro\forest@temp@n{#1}%
3453   \forest@nodewalk@walklist{\forest@nodewalk@historyback}{\forest@nodewalk@historyforward}{0}{\forest@temp@n-1}
3454   \forest@nodewalk@forward@updatehistory
3455 },
3456 nodewalk/forward/.default=1,
3457 define long step={last valid'}{@bare}{%
3458   \ifnum\forest@cn=0
3459     \forest@nodewalk@tolastvalid
3460     \forest@nodewalk@makestep
3461   \fi
3462 },
3463 define long step={last valid}{@bare}{%
3464   \forest@nodewalk@tolastvalid
3465 },
3466 define long step={reverse}{n args=1,@bare,make for}{%
3467   \forest@nodewalk{#1,TeX={%
3468     \global\let\forest@global@temp\forest@nodewalk@historyback
3469     \global\let\forest@global@tempn\forest@nodewalk@n
3470     }}}%
3471   \forest@nodewalk@walklist{\forest@global@temp}{0}{\forest@global@tempn}{\let\forest@cn\forest@nodewalk@n}
3472 },
3473 define long step={walk and reverse}{n args=1,@bare,make for}{%
3474   \edef\forest@marshal{%
3475     \noexpand\pgfkeysalso{\unexpanded{#1}}%
3476     \noexpand\forest@nodewalk@walklist{\forest@nodewalk@historyback}{0}{\noexpand\forest@nodewalk@n}
3477   }\forest@marshal
3478 },
3479 define long step={sort}{n args=1,@bare,make for}{%
3480   \forest@nodewalk{#1,TeX={%
3481     \global\let\forest@global@temp\forest@nodewalk@historyback
3482     \global\let\forest@global@tempn\forest@nodewalk@n
3483     }}}%
3484   \forest@nodewalk@sortlist{\forest@global@temp}{\forest@global@tempn}\forest@sort@ascending
3485 },
3486 define long step={sort'}{n args=1,@bare,make for}{%

```

```

3487 \forest@nodewalk{#1,TeX={%
3488 \global\let\forest@global@temp\forest@nodewalk@historyback
3489 \global\let\forest@global@tempn\forest@nodewalk@n
3490 }}{}%
3491 \forest@nodewalk@sortlist{\forest@global@temp}{\forest@global@tempn}\forest@sort@descending
3492 },
3493 define long step={walk and sort}{n args=1,@bare,make for}{% walk as given, then walk sorted
3494 \edef\forest@marshal{%
3495 \noexpand\pgfkeysalso{\unexpanded{#1}}}%
3496 \noexpand\forest@nodewalk@sortlist{\noexpand\forest@nodewalk@historyback}{\noexpand\forest@nodewalk@n-
3497 }\forest@marshal
3498 },
3499 define long step={walk and sort'}{n args=1,@bare,make for}{%
3500 \edef\forest@marshal{%
3501 \noexpand\pgfkeysalso{\unexpanded{#1}}}%
3502 \noexpand\forest@nodewalk@sortlist{\noexpand\forest@nodewalk@historyback}{\noexpand\forest@nodewalk@n-
3503 }\forest@marshal
3504 },
3505 sort by/.store in=\forest@nodesort@by,
3506 define long step={save}{n args=2,@bare,make for}{% #1 = name, #2 = nodewalk
3507 \forest@forthis{%
3508 \forest@nodewalk{#2,TeX={%
3509 \global\let\forest@global@temp\forest@nodewalk@historyback
3510 \global\let\forest@global@tempn\forest@nodewalk@n
3511 }}{}%
3512 }%
3513 \forest@nodewalk@walklist{}{\forest@global@temp}{0}{\forest@global@tempn}\relax
3514 \csedef{forest@nodewalk@saved@#1}{\forest@nodewalk@walklist@walked}%
3515 },
3516 define long step={walk and save}{n args=2,@bare,make for}{% #1 = name, #2 = nodewalk
3517 \edef\forest@marshal{%
3518 \noexpand\pgfkeysalso{\unexpanded{#2}}}%
3519 \noexpand\forest@nodewalk@walklist{}{\noexpand\forest@nodewalk@historyback}{0}{\noexpand\forest@nodewalk@
3520 }\forest@marshal
3521 \csedef{forest@nodewalk@saved@#1}{\forest@nodewalk@walklist@walked}%
3522 },
3523 nodewalk/save history/.code 2 args={% #1 = back, forward
3524 \csedef{forest@nodewalk@saved@#1}{\forest@nodewalk@historyback}%
3525 \csedef{forest@nodewalk@saved@#2}{\forest@nodewalk@historyforward}%
3526 },
3527 define long step={load}{n args=1,@bare,make for}{%
3528 \forest@nodewalk@walklist{}{\csuse{forest@nodewalk@saved@#1}0,}{0}{-1}{\ifnum\forest@nodewalk@cn=0 \else\
3529 },
3530 if in saved nodewalk/.code n args=4{% is node #1 in nodewalk #2; yes: #3, no: #4
3531 \forest@forthis{%
3532 \forest@go{#1}%
3533 \edef\forest@marshal{%
3534 \noexpand\pgfutil@in@{\forest@cn,}{\csuse{forest@nodewalk@saved@#2},}%
3535 }\forest@marshal
3536 }%
3537 \ifpgfutil@in@
3538 \@escapeif{\pgfkeysalso{#3}}%
3539 \else
3540 \@escapeif{\pgfkeysalso{#4}}%
3541 \fi
3542 },
3543 where in saved nodewalk/.style n args=4{
3544 for tree={if in saved nodewalk={#1}{#2}{#3}{#4}}
3545 },
3546 nodewalk/options/.code={\forestset{#1}},
3547 nodewalk/TeX/.code={#1},

```



```

3548 nodewalk/TeX'/.code={\appto\forest@externalize@loadimages{#1}#1},
3549 nodewalk/TeX'/.code={\appto\forest@externalize@loadimages{#1}},
3550 nodewalk/typeout/.style={TeX={\typeout{#1}}},
3551 % repeat is taken later from /forest/repeat
3552 }
3553 \def\forest@nodewalk@walklist#1#2#3#4#5{%
3554 % #1 = list of preceding, #2 = list to walk
3555 % #3 = from, #4 = to
3556 % #5 = every step code
3557 \let\forest@nodewalk@cn\forest@cn
3558 \edef\forest@marshal{%
3559 \noexpand\forest@nodewalk@walklist@{#1}{#2}{\number\numexpr#3}{\number\numexpr#4}{1}{0}{\unexpanded{#5}}%
3560 }\forest@marshal
3561 }
3562 \def\forest@nodewalk@walklist@#1#2#3#4#5#6#7{%
3563 % #1 = list of walked, #2 = list to walk
3564 % #3 = from, #4 = to
3565 % #5 = current step n, #6 = steps made
3566 % #7 = every step code
3567 \def\forest@nodewalk@walklist@walked{#1}%
3568 \def\forest@nodewalk@walklist@rest{#2}%
3569 \edef\forest@nodewalk@walklist@stepsmade{#6}%
3570 \ifnum#4<0
3571 \forest@temptrue
3572 \else
3573 \ifnum#5>#4\relax
3574 \forest@tempfalse
3575 \else
3576 \forest@temptrue
3577 \fi
3578 \fi
3579 \ifforest@temp
3580 \edef\forest@nodewalk@cn{\forest@csvlist@getfirst@{#2}}%
3581 \ifnum\forest@nodewalk@cn=0
3582 #7%
3583 \else
3584 \ifnum#5>#3\relax
3585 #7%
3586 \edef\forest@nodewalk@walklist@stepsmade{\number\numexpr#6+1}%
3587 \fi
3588 \forest@csvlist@getfirstrest@{#2}\forest@nodewalk@cn\forest@nodewalk@walklist@rest
3589 \@escapeiffif{%
3590 \edef\forest@marshal{%
3591 \noexpand\forest@nodewalk@walklist@
3592 {\forest@nodewalk@cn,#1}{\forest@nodewalk@walklist@rest}{#3}{#4}{\number\numexpr#5+1}{\forest@nod
3593 }\forest@marshal
3594 }%
3595 \fi
3596 \fi
3597 }
3598
3599 \def\forest@nodewalk@back@updatehistory{%
3600 \ifnum\forest@cn=0
3601 \let\forest@nodewalk@historyback\forest@nodewalk@walklist@rest
3602 \let\forest@nodewalk@historyforward\forest@nodewalk@walklist@walked
3603 \else
3604 \expandafter\forest@csvlist@getfirstrest@\expandafter{\forest@nodewalk@walklist@walked}\forest@temp\fore
3605 \edef\forest@nodewalk@historyback{\forest@temp,\forest@nodewalk@walklist@rest}%
3606 \fi
3607 }
3608 \def\forest@nodewalk@forward@updatehistory{%

```

```

3609 \let\forest@nodewalk@historyforward\forest@nodewalk@walklist@rest
3610 \let\forest@nodewalk@historyback\forest@nodewalk@walklist@walked
3611 }
3612 \def\forest@go#1{%
3613 \def\forest@nodewalk@config@everystep@method{independent}%
3614 \def\forest@nodewalk@config@history@method{inherited}%
3615 \def\forest@nodewalk@config@oninvalid{inherited}%
3616 \forest@saveandrestoremacro\forest@nodewalk@oninvalid{%
3617 \forest@Nodewalk{#1}{}%
3618 }%
3619 }
3620 \def\forest@csvlist@getfirst@#1{% assuming that the list is nonempty and finishes with a comma
3621 \forest@csvlist@getfirst@@#1\forest@csvlist@getfirst@@}
3622 \def\forest@csvlist@getfirst@@#1,#2\forest@csvlist@getfirst@@{#1}
3623 \def\forest@csvlist@getrest@#1{% assuming that the list is nonempty and finishes with a comma
3624 \forest@csvlist@getrest@@#1\forest@csvlist@getrest@@}
3625 \def\forest@csvlist@getrest@@#1,#2\forest@csvlist@getrest@@{#2}
3626 \def\forest@csvlist@getfirstrest@#1#2#3{% assuming that the list is nonempty and finishes with a comma
3627 % #1 = list, #2 = cs receiving first, #3 = cs receiving rest
3628 \forest@csvlist@getfirstrest@@#1\forest@csvlist@getfirstrest@@{#2}{#3}}
3629 \def\forest@csvlist@getfirstrest@@#1,#2\forest@csvlist@getfirstrest@@@#3#4{%
3630 \def#3{#1}%
3631 \def#4{#2}%
3632 }
3633 \def\forest@nodewalk@stripfakesteps#1{%
3634 % go to the last valid node if the walk contained any nodes, otherwise restore the current node
3635 \edef\forest@marshal{%
3636 \unexpanded{#1}%
3637 \noexpand\ifnum\noexpand\forest@nodewalk@n=\forest@nodewalk@n\relax
3638 \def\noexpand\forest@cn{\forest@cn}%
3639 \noexpand\else
3640 \unexpanded{%
3641 \edef\forest@cn{%
3642 \expandafter\forest@csvlist@getfirst@\expandafter{\forest@nodewalk@historyback}}%
3643 }%
3644 }%
3645 \noexpand\fi
3646 }\forest@marshal
3647 }
3648 \def\forest@nodewalk@tolastvalid{%
3649 \ifnum\forest@cn=0
3650 \edef\forest@cn{\expandafter\forest@csvlist@getfirst@\expandafter{\forest@nodewalk@historyback}}%
3651 \ifnum\forest@cn=0
3652 \let\forest@cn\forest@nodewalk@origin
3653 \fi
3654 \fi
3655 }
3656 \def\forest@nodewalk@sortlist#1#2#3{%#1=list,#2=to,#3=asc/desc
3657 \edef\forest@nodewalksort@list{#1}%
3658 \expandafter\forest@nodewalk@sortlist@\expandafter{\number\numexpr#2}{#3}%
3659 }
3660 \def\forest@nodewalk@sortlist@#1#2{%#1=to,#2=asc/desc
3661 \safeloop
3662 \unless\ifnum\safeloopn>#1\relax
3663 \expandafter\forest@csvlist@getfirstrest@\expandafter{\forest@nodewalksort@list}\forest@nodewalksort@cn\fi
3664 \csedef{forest@nodesort@\safeloopn}{\forest@nodewalksort@cn}%
3665 \saferepeat
3666 \edef\forest@nodesort@sortkey{\forest@nodesort@by}%
3667 \forest@sort\forest@nodesort@cmpnodes\forest@nodesort@let#2{1}{#1}%
3668 \def\forest@nodewalksort@sorted{}%
3669 \safeloop

```

```

3670 \unless\ifnum\safeloopn>#1\relax
3671 \edef\forest@cn{\csname forest@nodesort@\safeloopn\endcsname}%
3672 \forest@nodewalk@makestep
3673 \saferepeat
3674 }

Find minimal/maximal node in a walk.

3675 \forestset{
3676 define long step={min}{n args=1,@bare,make for}{% the first min in the argument nodewalk
3677 \forest@nodewalk{#1,TeX={%
3678 \global\let\forest@global@temp\forest@nodewalk@historyback
3679 }}{}}%
3680 \forest@nodewalk@minmax{\forest@global@temp}{-1}{<}{\forest@nodewalk@minmax@node,}%
3681 },
3682 define long step={mins}{n args=1,@bare,make for}{% all mins in the argument nodewalk
3683 \forest@nodewalk{#1,TeX={%
3684 \global\let\forest@global@temp\forest@nodewalk@historyback
3685 }}{}}%
3686 \forest@nodewalk@minmax{\forest@global@temp}{-1}{<}{\forest@nodewalk@minmax@nodes}%
3687 },
3688 define long step={walk and min}{n args=1,@bare}{%
3689 \edef\forest@marshal{%
3690 \noexpand\pgfkeysalso{\unexpanded{#1}}%
3691 \noexpand\forest@nodewalk@minmax{\noexpand\forest@nodewalk@historyback}{\noexpand\forest@nodewalk@n-\fo
3692 }\forest@marshal
3693 },
3694 define long step={walk and mins}{n args=1,@bare}{%
3695 \edef\forest@marshal{%
3696 \noexpand\pgfkeysalso{\unexpanded{#1}}%
3697 \noexpand\forest@nodewalk@minmax{\noexpand\forest@nodewalk@historyback}{\noexpand\forest@nodewalk@n-\fo
3698 }\forest@marshal
3699 },
3700 define long step={min in nodewalk}{@bare}{% find the first min in the preceding nodewalk, step to it
3701 \forest@nodewalk@minmax{\forest@nodewalk@historyback}{-1}{<}{\forest@nodewalk@minmax@node,}%
3702 },
3703 define long step={mins in nodewalk}{@bare}{% find mins in the preceding nodewalk, step to mins
3704 \forest@nodewalk@minmax{\forest@nodewalk@historyback}{-1}{<}{\forest@nodewalk@minmax@nodes}%
3705 },
3706 define long step={min in nodewalk'}{@bare}{% find the first min in the preceding nodewalk, step to min in h
3707 \forest@nodewalk@minmax{\forest@nodewalk@historyback}{-1}{<}{}}%
3708 },
3709 %
3710 define long step={max}{n args=1,@bare,make for}{% the first max in the argument nodewalk
3711 \forest@nodewalk{#1,TeX={%
3712 \global\let\forest@global@temp\forest@nodewalk@historyback
3713 }}{}}%
3714 \forest@nodewalk@minmax{\forest@global@temp}{-1}{>}{\forest@nodewalk@minmax@node,}%
3715 },
3716 define long step={maxs}{n args=1,@bare,make for}{% all maxs in the argument nodewalk
3717 \forest@nodewalk{#1,TeX={%
3718 \global\let\forest@global@temp\forest@nodewalk@historyback
3719 }}{}}%
3720 \forest@nodewalk@minmax{\forest@global@temp}{-1}{>}{\forest@nodewalk@minmax@nodes}%
3721 },
3722 define long step={walk and max}{n args=1,@bare}{%
3723 \edef\forest@marshal{%
3724 \noexpand\pgfkeysalso{\unexpanded{#1}}%
3725 \noexpand\forest@nodewalk@minmax{\noexpand\forest@nodewalk@historyback}{\noexpand\forest@nodewalk@n-\fo
3726 }\forest@marshal
3727 },
3728 define long step={walk and maxs}{n args=1,@bare}{%

```

```

3729 \edef\forest@marshal{%
3730 \noexpand\pgfkeysalso{\unexpanded{#1}}%
3731 \noexpand\forest@nodewalk@minmax{\noexpand\forest@nodewalk@historyback}{\noexpand\forest@nodewalk@n-\fo
3732 }\forest@marshal
3733 },
3734 define long step={max in nodewalk}{@bare}{% find the first max in the preceding nodewalk, step to it
3735 \forest@nodewalk@minmax{\forest@nodewalk@historyback}{-1}{>}{\forest@nodewalk@minmax@node,}%
3736 },
3737 define long step={maxs in nodewalk}{@bare}{% find maxs in the preceding nodewalk, step to maxs
3738 \forest@nodewalk@minmax{\forest@nodewalk@historyback}{-1}{>}{\forest@nodewalk@minmax@nodes}%
3739 },
3740 define long step={max in nodewalk'}{@bare}{% find the first max in the preceding nodewalk, step to max in h
3741 \forest@nodewalk@minmax{\forest@nodewalk@historyback}{-1}{>}{}}%
3742 },
3743 }
3744
3745 \def\forest@nodewalk@minmax#1#2#3#4{%
3746 % #1 = list of nodes
3747 % #2 = max index in list (start with 1)
3748 % #3 = min/max = ascending/descending = </>
3749 % #4 = how many results? 1 = {\forest@nodewalk@minmax@node,}, all={\forest@nodewalk@minmax@nodes}, walk in
3750 \edef\forest@nodesort@sortkey{\forest@nodesort@by}%
3751 \edef\forest@nodewalk@minmax@N{\number\numexpr#2}%
3752 \edef\forest@nodewalk@minmax@n{#1}%
3753 \edef\forest@nodewalk@minmax@list{#1}%
3754 \def\forest@nodewalk@minmax@nodes{#1}%
3755 \def\forest@nodewalk@minmax@node{#1}%
3756 \ifdefempty{\forest@nodewalk@minmax@list}{%
3757 }{%
3758 \safeloop
3759 \expandafter\forest@cvslist@getfirstrest\expandafter{\forest@nodewalk@minmax@list}\forest@nodewalk@min
3760 \ifnum\forest@nodewalk@minmax@cn=0 \else
3761 \ifdefempty{\forest@nodewalk@minmax@node}{%
3762 \edef\forest@nodewalk@minmax@node{\forest@nodewalk@minmax@cn}%
3763 \edef\forest@nodewalk@minmax@nodes{\forest@nodewalk@minmax@cn,}%
3764 \edef\forest@nodewalk@minmax@n{\safeloopn}%
3765 }{%
3766 \csedef{forest@nodesort@1}{\forest@nodewalk@minmax@node}%
3767 \csedef{forest@nodesort@2}{\forest@nodewalk@minmax@cn}%
3768 \forest@nodesort@cmpnodes{2}{1}%
3769 \if=\forest@sort@cmp@result
3770 \edef\forest@nodewalk@minmax@node{\forest@nodewalk@minmax@cn}%
3771 \epreto\forest@nodewalk@minmax@nodes{\forest@nodewalk@minmax@cn,}%
3772 \edef\forest@nodewalk@minmax@n{\safeloopn}%
3773 \else
3774 \if#3\forest@sort@cmp@result
3775 \edef\forest@nodewalk@minmax@node{\forest@nodewalk@minmax@cn}%
3776 \edef\forest@nodewalk@minmax@nodes{\forest@nodewalk@minmax@cn,}%
3777 \edef\forest@nodewalk@minmax@n{\safeloopn}%
3778 \fi
3779 \fi
3780 }%
3781 \fi
3782 \ifdefempty{\forest@nodewalk@minmax@list}{\forest@tempfalse}{\forest@temptrue}%
3783 \ifnum\safeloopn=\forest@nodewalk@minmax@N\relax\forest@temptrue\fi
3784 \ifforest@temp
3785 \saferepeat
3786 \edef\forest@nodewalk@minmax@list{#4}%
3787 \ifdefempty\forest@nodewalk@minmax@list{%
3788 \forestset{nodewalk/jump back=\forest@nodewalk@minmax@n-1}% CHECK
3789 }{%

```

```

3790     \safeloop
3791     \expandafter\forest@csvlist@getfirstrest@\expandafter{\forest@nodewalk@minmax@list}\forest@cn\forest@
3792     \forest@nodewalk@makestep
3793     \ifdefempty{\forest@nodewalk@minmax@list}{\forest@tempfalse}{\forest@temptrue}%
3794     \ifforest@temp
3795     \saferepeat
3796   }%
3797 }%
3798 }

```

The short-form step mechanism. The complication is that we want to be able to collect tikz and pgf options here, and it is impossible(?) to know in advance what keys are valid there. So we rather check whether the given keyname is a sequence of short steps; if not, we pass the key on.

```

3799 \newtoks\forest@nodewalk@shortsteps@resolution
3800 \newif\ifforest@nodewalk@aeshortsteps
3801 \pgfqkeys{/forest/nodewalk}{
3802   .unknown/.code={%
3803     \forest@nodewalk@aeshortstepsfalse
3804     \pgfkeysifdefined{/forest/\pgfkeyscurrentname/.cmd}{%
3805       }{%
3806         \ifx\pgfkeyscurrentvalue\pgfkeysnovalue@text % no value, so possibly short steps
3807         \forest@nodewalk@shortsteps@resolution{}%
3808         \forest@nodewalk@aeshortstepstrue
3809         \expandafter\forest@nodewalk@shortsteps\pgfkeyscurrentname=====,% "=" and "," cannot be short st
3810       \fi
3811     }%
3812   \ifforest@nodewalk@aeshortsteps
3813     \@escapeif{\expandafter\pgfkeysalso\expandafter{\the\forest@nodewalk@shortsteps@resolution}}%
3814   \else
3815     \@escapeif{\pgfkeysalso{/forest/\pgfkeyscurrentname=#1}}%
3816   \fi
3817 },
3818 }
3819 \def\forest@nodewalk@shortsteps{%
3820   \futurelet\forest@nodewalk@nexttoken\forest@nodewalk@shortsteps@
3821 }
3822 \def\forest@nodewalk@shortsteps@{%
3823   \ifx\forest@nodewalk@nexttoken=%
3824     \let\forest@nodewalk@nextop\forest@nodewalk@shortsteps@end
3825   \else
3826     \ifx\forest@nodewalk@nexttoken\bgroup
3827       \letcs\forest@nodewalk@nextop{forest@shortstep@group}%
3828     \else
3829       \let\forest@nodewalk@nextop\forest@nodewalk@shortsteps@@
3830     \fi
3831   \fi
3832   \forest@nodewalk@nextop
3833 }
3834 \def\forest@nodewalk@shortsteps@@#1{%
3835   \ifcsdef{forest@shortstep@#1}{%
3836     \csname forest@shortstep@#1\endcsname
3837   }{%
3838     \forest@nodewalk@aeshortstepsfalse
3839     \forest@nodewalk@shortsteps@end
3840   }%
3841 }
3842 % in the following definitions:
3843 % #1 = short step
3844 % #2 = (long) step, or a style in /forest/nodewalk (taking n args)
3845 \csdef{forest@nodewalk@defshortstep@0@args}#1#2{%
3846   \csdef{forest@shortstep@#1}{%

```

```

3847 \apptotoks\forest@nodewalk@shortsteps@resolution{,#2}%
3848 \forest@nodewalk@shortsteps}}
3849 \csdef{forest@nodewalk@defshortstep@1@args}#1#2{%
3850 \csdef{forest@shortstep@#1}##1{%
3851 \edef\forest@marshal####1{#2}%
3852 \eapptotoks\forest@nodewalk@shortsteps@resolution{,\forest@marshal{##1}}%
3853 \forest@nodewalk@shortsteps}}
3854 \csdef{forest@nodewalk@defshortstep@2@args}#1#2{%
3855 \csdef{forest@shortstep@#1}##1##2{%
3856 \edef\forest@marshal####1####2{#2}%
3857 \eapptotoks\forest@nodewalk@shortsteps@resolution{,\forest@marshal{##1}{##2}}%
3858 \forest@nodewalk@shortsteps}}
3859 \csdef{forest@nodewalk@defshortstep@3@args}#1#2{%
3860 \csdef{forest@shortstep@#1}##1##2##3{%
3861 \edef\forest@marshal####1####2####3{#2}%
3862 \eapptotoks\forest@nodewalk@shortsteps@resolution{,\forest@marshal{##1}{##2}{##3}}%
3863 \forest@nodewalk@shortsteps}}
3864 \csdef{forest@nodewalk@defshortstep@4@args}#1#2{%
3865 \csdef{forest@shortstep@#1}##1##2##3##4{%
3866 \edef\forest@marshal####1####2####3####4{#2}%
3867 \eapptotoks\forest@nodewalk@shortsteps@resolution{,\forest@marshal{##1}{##2}{##3}{##4}}%
3868 \forest@nodewalk@shortsteps}}
3869 \csdef{forest@nodewalk@defshortstep@5@args}#1#2{%
3870 \csdef{forest@shortstep@#1}##1##2##3##4##5{%
3871 \edef\forest@marshal####1####2####3####4####5{#2}%
3872 \eapptotoks\forest@nodewalk@shortsteps@resolution{,\forest@marshal{##1}{##2}{##3}{##4}{##5}}%
3873 \forest@nodewalk@shortsteps}}
3874 \csdef{forest@nodewalk@defshortstep@6@args}#1#2{%
3875 \csdef{forest@shortstep@#1}##1##2##3##4##5##6{%
3876 \edef\forest@marshal####1####2####3####4####5####6{#2}%
3877 \eapptotoks\forest@nodewalk@shortsteps@resolution{,\forest@marshal{##1}{##2}{##3}{##4}{##5}{##6}}%
3878 \forest@nodewalk@shortsteps}}
3879 \csdef{forest@nodewalk@defshortstep@7@args}#1#2{%
3880 \csdef{forest@shortstep@#1}##1##2##3##4##5##6##7{%
3881 \edef\forest@marshal####1####2####3####4####5####6####7{#2}%
3882 \eapptotoks\forest@nodewalk@shortsteps@resolution{,\forest@marshal{##1}{##2}{##3}{##4}{##5}{##6}{##7}}%
3883 \forest@nodewalk@shortsteps}}
3884 \csdef{forest@nodewalk@defshortstep@8@args}#1#2{%
3885 \csdef{forest@shortstep@#1}##1##2##3##4##5##6##7##8{%
3886 \edef\forest@marshal####1####2####3####4####5####6####7####8{#2}%
3887 \eapptotoks\forest@nodewalk@shortsteps@resolution{,\forest@marshal{##1}{##2}{##3}{##4}{##5}{##6}{##7}{##8}}%
3888 \forest@nodewalk@shortsteps}}
3889 \csdef{forest@nodewalk@defshortstep@9@args}#1#2{%
3890 \csdef{forest@shortstep@#1}##1##2##3##4##5##6##7##8##9{%
3891 \edef\forest@marshal####1####2####3####4####5####6####7####8####9{#2}%
3892 \eapptotoks\forest@nodewalk@shortsteps@resolution{,\forest@marshal{##1}{##2}{##3}{##4}{##5}{##6}{##7}{##8}{##9}}%
3893 \forest@nodewalk@shortsteps}}
3894 \forestset{
3895 define short step/.code n args=3{% #1 = short step, #2 = n args, #3 = long step
3896 \csname forest@nodewalk@defshortstep@#2@args\endcsname{##1}{##3}%
3897 },
3898 }
3899 \def\forest@nodewalk@shortsteps@end#1,{
    Define short-form steps.
3900 \forestset{
3901 define short step={group}{1}{group={#1}}, % {braces} are special
3902 define short step={p}{0}{previous},
3903 define short step={n}{0}{next},
3904 define short step={u}{0}{parent},
3905 define short step={s}{0}{sibling},

```

```

3906 define short step={c}{0}{current},
3907 define short step={o}{0}{origin},
3908 define short step={r}{0}{root},
3909 define short step={R}{0}{root'},
3910 define short step={P}{0}{previous leaf},
3911 define short step={N}{0}{next leaf},
3912 define short step={F}{0}{first leaf},
3913 define short step={L}{0}{last leaf},
3914 define short step={>}{0}{next on tier},
3915 define short step={<}{0}{previous on tier},
3916 define short step={1}{0}{n=1},
3917 define short step={2}{0}{n=2},
3918 define short step={3}{0}{n=3},
3919 define short step={4}{0}{n=4},
3920 define short step={5}{0}{n=5},
3921 define short step={6}{0}{n=6},
3922 define short step={7}{0}{n=7},
3923 define short step={8}{0}{n=8},
3924 define short step={9}{0}{n=9},
3925 define short step={l}{0}{last},
3926 define short step={b}{0}{back},
3927 define short step={f}{0}{forward},
3928 define short step={v}{0}{last valid},
3929 define short step={*}{2}{repeat={#1}{#2}},
3930 for 1/.style={for nodewalk={n=1}{#1}},
3931 for 2/.style={for nodewalk={n=2}{#1}},
3932 for 3/.style={for nodewalk={n=3}{#1}},
3933 for 4/.style={for nodewalk={n=4}{#1}},
3934 for 5/.style={for nodewalk={n=5}{#1}},
3935 for 6/.style={for nodewalk={n=6}{#1}},
3936 for 7/.style={for nodewalk={n=7}{#1}},
3937 for 8/.style={for nodewalk={n=8}{#1}},
3938 for 9/.style={for nodewalk={n=9}{#1}},
3939 for -1/.style={for nodewalk={n'=1}{#1}},
3940 for -2/.style={for nodewalk={n'=2}{#1}},
3941 for -3/.style={for nodewalk={n'=3}{#1}},
3942 for -4/.style={for nodewalk={n'=4}{#1}},
3943 for -5/.style={for nodewalk={n'=5}{#1}},
3944 for -6/.style={for nodewalk={n'=6}{#1}},
3945 for -7/.style={for nodewalk={n'=7}{#1}},
3946 for -8/.style={for nodewalk={n'=8}{#1}},
3947 for -9/.style={for nodewalk={n'=9}{#1}},
3948 }

```

Define multiple-step walks.

```

3949 \forestset{
3950 define long step={tree}{-}{\forest@node@foreach{\forest@nodewalk@makestep}},
3951 define long step={tree reversed}{-}{\forest@node@foreach@reversed{\forest@nodewalk@makestep}},
3952 define long step={tree children-first}{-}{\forest@node@foreach@childrenfirst{\forest@nodewalk@makestep}},
3953 define long step={tree children-first reversed}{-}{\forest@node@foreach@childrenfirst@reversed{\forest@nodewalk@makestep}},
3954 define long step={tree breadth-first}{-}{\forest@node@foreach@breadthfirst{-1}{\forest@nodewalk@makestep}},
3955 define long step={tree breadth-first reversed}{-}{\forest@node@foreach@breadthfirst@reversed{-1}{\forest@nodewalk@makestep}},
3956 define long step={descendants}{-}{\forest@node@foreachdescendant{\forest@nodewalk@makestep}},
3957 define long step={descendants reversed}{-}{\forest@node@foreachdescendant@reversed{\forest@nodewalk@makestep}},
3958 define long step={descendants children-first}{-}{\forest@node@foreachdescendant@childrenfirst{\forest@nodewalk@makestep}},
3959 define long step={descendants children-first reversed}{-}{\forest@node@foreachdescendant@childrenfirst@reversed{\forest@nodewalk@makestep}},
3960 define long step={descendants breadth-first}{-}{\forest@node@foreach@breadthfirst{0}{\forest@nodewalk@makestep}},
3961 define long step={descendants breadth-first reversed}{-}{\forest@node@foreach@breadthfirst@reversed{0}{\forest@nodewalk@makestep}},
3962 define long step={level}{n args=1}{%
3963   \pgfmathtruncatemacro\forest@temp{#1}%
3964   \edef\forest@marshal{%

```

```

3965     \noexpand\forest@node@foreach@breadthfirst
3966     {\forest@temp}%
3967     {\noexpand\ifnum\noexpand\forestove{level}=\forest@temp\relax\noexpand\forest@nodewalk@makestep\noexp
3968     }\forest@marshal
3969 },
3970 define long step={level>}{n args=1}{%
3971     \pgfmathtruncatemacro\forest@temp{#1}%
3972     \edef\forest@marshal{%
3973         \noexpand\forest@node@foreach@breadthfirst
3974         {-1}%
3975         {\noexpand\ifnum\noexpand\forestove{level}<\forest@temp\relax\noexpand\else\noexpand\forest@nodewalk@
3976         }\forest@marshal
3977     },
3978 define long step={level<}{n args=1}{%
3979     \pgfmathtruncatemacro\forest@temp{(#1)-1}%
3980     \edef\forest@marshal{%
3981         \noexpand\forest@node@foreach@breadthfirst
3982         {\forest@temp}%
3983         {\noexpand\forest@nodewalk@makestep}%
3984     }\forest@marshal
3985 },
3986 define long step={level reversed}{n args=1}{%
3987     \pgfmathtruncatemacro\forest@temp{#1}%
3988     \edef\forest@marshal{%
3989         \noexpand\forest@node@foreach@breadthfirst@reversed
3990         {\forest@temp}%
3991         {\noexpand\ifnum\noexpand\forestove{level}=\forest@temp\relax\noexpand\forest@nodewalk@makestep\noexp
3992         }\forest@marshal
3993     },
3994 define long step={level reversed>}{n args=1}{%
3995     \pgfmathtruncatemacro\forest@temp{#1}%
3996     \edef\forest@marshal{%
3997         \noexpand\forest@node@foreach@breadthfirst@reversed
3998         {-1}%
3999         {\noexpand\ifnum\noexpand\forestove{level}<\forest@temp\relax\noexpand\else\noexpand\forest@nodewalk@
4000         }\forest@marshal
4001     },
4002 define long step={level reversed<}{n args=1}{%
4003     \pgfmathtruncatemacro\forest@temp{(#1)-1}%
4004     \edef\forest@marshal{%
4005         \noexpand\forest@node@foreach@breadthfirst@reversed
4006         {\forest@temp}%
4007         {\noexpand\forest@nodewalk@makestep}%
4008     }\forest@marshal
4009 },
4010 %
4011 define long step={relative level}{n args=1}{%
4012     \pgfmathtruncatemacro\forest@temp{(#1)+\forestove{level}}%
4013     \edef\forest@marshal{%
4014         \noexpand\forest@node@foreach@breadthfirst
4015         {\forest@temp}%
4016         {\noexpand\ifnum\noexpand\forestove{level}=\forest@temp\relax\noexpand\forest@nodewalk@makestep\noexp
4017     }\forest@marshal
4018 },
4019 define long step={relative level>}{n args=1}{%
4020     \pgfmathtruncatemacro\forest@temp{(#1)+\forestove{level}}%
4021     \edef\forest@marshal{%
4022         \noexpand\forest@node@foreach@breadthfirst
4023         {-1}%
4024         {\noexpand\ifnum\noexpand\forestove{level}<\forest@temp\relax\noexpand\else\noexpand\forest@nodewalk@
4025     }\forest@marshal

```



```

4026 },
4027 define long step={relative level<}{n args=1}{%
4028   \pgfmathtruncatemacro\forest@temp{(#1)+\forestove{level}-1}%
4029   \edef\forest@marshal{%
4030     \noexpand\forest@node@foreach@breadthfirst
4031     {\forest@temp}%
4032     {\noexpand\forest@nodewalk@makestep}%
4033   }\forest@marshal
4034 },
4035 define long step={relative level reversed}{n args=1}{%
4036   \pgfmathtruncatemacro\forest@temp{(#1)+\forestove{level}}%
4037   \edef\forest@marshal{%
4038     \noexpand\forest@node@foreach@breadthfirst@reversed
4039     {\forest@temp}%
4040     {\noexpand\ifnum\noexpand\forestove{level}=\forest@temp\relax\noexpand\forest@nodewalk@makestep\noexp
4041   }\forest@marshal
4042 },
4043 define long step={relative level reversed>}{n args=1}{%
4044   \pgfmathtruncatemacro\forest@temp{(#1)+\forestove{level}}%
4045   \edef\forest@marshal{%
4046     \noexpand\forest@node@foreach@breadthfirst@reversed
4047     {-1}%
4048     {\noexpand\ifnum\noexpand\forestove{level}<\forest@temp\relax\noexpand\else\noexpand\forest@nodewalk@
4049   }\forest@marshal
4050 },
4051 define long step={relative level reversed<}{n args=1}{%
4052   \pgfmathtruncatemacro\forest@temp{(#1)+\forestove{level}-1}%
4053   \edef\forest@marshal{%
4054     \noexpand\forest@node@foreach@breadthfirst@reversed
4055     {\forest@temp}%
4056     {\noexpand\forest@nodewalk@makestep}%
4057   }\forest@marshal
4058 },
4059 define long step={children}{\forest@node@foreachchild{\forest@nodewalk@makestep}},
4060 define long step={children reversed}{\forest@node@foreachchild@reversed{\forest@nodewalk@makestep}},
4061 define long step={current and following siblings}{\forest@node@@forselfandfollowingsiblings{\forest@nodew
4062 define long step={following siblings}{style}{if nodewalk valid={next}{fake=next,current and following sibli
4063 define long step={current and preceding siblings}{\forest@node@@forselfandprecedingsiblings{\forest@nodew
4064 define long step={preceding siblings}{style}{if nodewalk valid={previous}{fake=previous,current and precedi
4065 define long step={current and following siblings reversed}{\forest@node@@forselfandfollowingsiblings@reve
4066 define long step={following siblings reversed}{style}{fake=next,current and following siblings reversed},
4067 define long step={current and preceding siblings reversed}{\forest@node@@forselfandprecedingsiblings@reve
4068 define long step={preceding siblings reversed}{style}{fake=previous,current and preceding siblings reversed
4069 define long step={siblings}{style}{for nodewalk'={preceding siblings},following siblings},
4070 define long step={siblings reversed}{style}{for nodewalk'={following siblings reversed},preceding siblings
4071 define long step={current and siblings}{style}{for nodewalk'={preceding siblings},current and following sib
4072 define long step={current and siblings reversed}{style}{for nodewalk'={current and following siblings rever
4073 define long step={ancestors}{style}{while={}{parent},last valid},
4074 define long step={current and ancestors}{style}{current,ancestors},
4075 define long step={following nodes}{style}{while={}{next node},last valid},
4076 define long step={preceding nodes}{style}{while={}{previous node},last valid},
4077 define long step={current and following nodes}{style}{current,following nodes},
4078 define long step={current and preceding nodes}{style}{current,preceding nodes},
4079 }
4080 \let\forest@nodewalkstephandler@styletrueorfalse\forest@nodewalkstephandler@styletrue

```

## 6.6 Dynamic tree

```

4081 \def\forest@last@node{0}
4082 \csdef{forest@nodewalk@saved@dynamic nodes}{}

```

```

4083 \def\forest@nodehandleby@name@nodewalk@or@bracket#1{%
4084   \ifx\pgfkeysnovalue#1%
4085     \edef\forest@last@node{\forest@node@Nametoid{forest@last@node}}%
4086   \else
4087     \forest@nodehandleby@nnb@checkfirst#1\forest@END
4088   \fi
4089 }
4090 \def\forest@nodehandleby@nnb@checkfirst#1#2\forest@END{%
4091   \ifx[#1]
4092     \forest@create@node{#1#2}%
4093     \cseappto{\forest@nodewalk@saved@dynamic nodes}{\forest@last@node,}%
4094   \else
4095     \forest@forthis{%
4096       \forest@nameandgo{#1#2}%
4097       \ifnum\forest@cn=0
4098         \PackageError{forest}{Cannot use a dynamic key on the invalid node}{}%
4099       \fi
4100       \let\forest@last@node\forest@cn
4101     }%
4102   \fi
4103 }
4104 \def\forest@create@node#1{% #1=bracket representation
4105   \bracketParse{\forest@create@collectafterthought}%
4106     \forest@last@node=#1\forest@end@create@node
4107 }
4108 \def\forest@create@collectafterthought#1\forest@end@create@node{%
4109   \forest@node@Foreach{\forest@last@node}{%
4110     \forest@toletto{delay}{given options}%
4111     \forest@setto{given options}{}%
4112   }%
4113   \forest@eappto{\forest@last@node}{delay}{,\unexpanded{#1}}%
4114   \forest@setto{\forest@last@node}{given options}{delay={}}%
4115 }
4116 \def\forest@create@node@and@process@given@options#1{% #1=bracket representation
4117   \bracketParse{\forest@createandprocess@collectafterthought}%
4118     \forest@last@node=#1\forest@end@create@node
4119 }
4120 \def\forest@createandprocess@collectafterthought#1\forest@end@create@node{%
4121   \forest@node@Compute@numeric@ts@info{\forest@last@node}%
4122   \forest@saveandrestoremacro\forest@root{%
4123     \let\forest@root\forest@last@node
4124     \forest@setto{process keylist=given options}%
4125   }%
4126 }
4127 \def\forest@saveandrestoremacro#1#2{% #1 = the (zero-arg) macro to save before and restore after processing c
4128   \edef\forest@marshal{%
4129     \unexpanded{#2}%
4130     \noexpand\def\noexpand#1{\expandonce{#1}}%
4131   }\forest@marshal
4132 }
4133 \def\forest@saveandrestoreifcs#1#2{% #1 = the if cs to save before and restore after processing code in #2
4134   \edef\forest@marshal{%
4135     \unexpanded{#2}%
4136     \ifbool{#1}{\noexpand\setbool{#1}{true}}{\noexpand\setbool{#1}{false}}
4137   }\forest@marshal
4138 }
4139 \def\forest@saveandrestoretoks#1#2{% #1 = the toks to save before and restore after processing code in #2
4140   \edef\forest@marshal{%
4141     \unexpanded{#2}%
4142     \noexpand#1{\the#1}%
4143   }\forest@marshal

```

```

4144 }
4145 \def\forest@saveandrestorereregister#1#2{% #1 = the register to save before and restore after processing code i
4146   \edef\forest@marshal{%
4147     \unexpanded{#2}%
4148     \noexpand\forestregister{#1}{\forestregister{#1}}%
4149   }\forest@marshal
4150 }
4151 \def\forest@remove@node#1{%
4152   \forest@node@Remove{#1}%
4153 }
4154 \def\forest@append@node#1#2{%
4155   \forest@node@Remove{#2}%
4156   \forest@node@Append{#1}{#2}%
4157 }
4158 \def\forest@prepend@node#1#2{%
4159   \forest@node@Remove{#2}%
4160   \forest@node@Prepend{#1}{#2}%
4161 }
4162 \def\forest@insertafter@node#1#2{%
4163   \forest@node@Remove{#2}%
4164   \forest@node@Insertafter{\forestOve{#1}{@parent}}{#2}{#1}%
4165 }
4166 \def\forest@insertbefore@node#1#2{%
4167   \forest@node@Remove{#2}%
4168   \forest@node@Insertbefore{\forestOve{#1}{@parent}}{#2}{#1}%
4169 }
4170 \def\forest@set@root#1#2{%
4171   \def\forest@root{#2}%
4172 }
4173 \def\forest@appto@do@ynamics#1#2{%
4174   \forest@nodehandleby@name@nodewalk@or@bracket{#2}%
4175   \ifcase\forest@dynamics@copyhow\relax\or
4176     \forest@tree@copy{\forest@last@node}\forest@last@node
4177   \or
4178     \forest@node@copy{\forest@last@node}\forest@last@node
4179   \fi
4180   \forest@node@ifnamedefined{forest@last@node}{%
4181     \forest@preto{\forest@last@node}{delay}
4182     {for id={\forest@node@Nametoid{forest@last@node}}{alias=forest@last@node},}%
4183   }{%
4184   \edef\forest@marshal{%
4185     \noexpand\apptotoks\noexpand\forest@do@ynamics{%
4186       \noexpand#1{\forest@cn}{\forest@last@node}}%
4187   }\forest@marshal
4188 }
4189 \forestset{%
4190   create/.code={%
4191     \forest@create@node{#1}%
4192     \forest@fornode{\forest@last@node}{%
4193       \forest@node@setalias{forest@last@node}%
4194       \cseappto{forest@nodewalk@saved@dynamic nodes}{\forest@last@node,}%
4195     }%
4196   },
4197   create'/.code={%
4198     \forest@create@node@and@process@given@options{#1}%
4199     \forest@fornode{\forest@last@node}{%
4200       \forest@node@setalias{forest@last@node}%
4201       \cseappto{forest@nodewalk@saved@dynamic nodes}{\forest@last@node,}%
4202     }%
4203   },
4204   append/.code={\def\forest@dynamics@copyhow{0}\forest@appto@do@ynamics\forest@append@node{#1}},

```

```

4205 prepend/.code={\def\forest@dynamics@copyhow{0}\forest@appto@do@dynamics\forest@prepend@node{#1}},
4206 insert after/.code={\def\forest@dynamics@copyhow{0}\forest@appto@do@dynamics\forest@insertafter@node{#1}},
4207 insert before/.code={\def\forest@dynamics@copyhow{0}\forest@appto@do@dynamics\forest@insertbefore@node{#1}},
4208 append'/.code={\def\forest@dynamics@copyhow{1}\forest@appto@do@dynamics\forest@append@node{#1}},
4209 prepend'/.code={\def\forest@dynamics@copyhow{1}\forest@appto@do@dynamics\forest@prepend@node{#1}},
4210 insert after'/.code={\def\forest@dynamics@copyhow{1}\forest@appto@do@dynamics\forest@insertafter@node{#1}},
4211 insert before'/.code={\def\forest@dynamics@copyhow{1}\forest@appto@do@dynamics\forest@insertbefore@node{#1}},
4212 append''/.code={\def\forest@dynamics@copyhow{2}\forest@appto@do@dynamics\forest@append@node{#1}},
4213 prepend''/.code={\def\forest@dynamics@copyhow{2}\forest@appto@do@dynamics\forest@prepend@node{#1}},
4214 insert after''/.code={\def\forest@dynamics@copyhow{2}\forest@appto@do@dynamics\forest@insertafter@node{#1}},
4215 insert before''/.code={\def\forest@dynamics@copyhow{2}\forest@appto@do@dynamics\forest@insertbefore@node{#1}},
4216 remove/.code={%
4217   \pgfkeysalso{alias=forest@last@node}%
4218   \cseappto{forest@nodewalk@saved@dynamic nodes}{\forest@cn,}%
4219   \expandafter\apptotoks\expandafter\forest@do@dynamics\expandafter{%
4220     \expandafter\forest@remove@node\expandafter{\forest@cn}}%
4221 },
4222 set root/.code={%
4223   \def\forest@dynamics@copyhow{0}%
4224   \forest@appto@do@dynamics\forest@set@root{#1}%
4225 },
4226 replace by/.code={\forest@replaceby@code{#1}{insert after}},
4227 replace by'/.code={\forest@replaceby@code{#1}{insert after'}},
4228 replace by''/.code={\forest@replaceby@code{#1}{insert after''}},
4229 sort/.code={%
4230   \eapptotoks\forest@do@dynamics{%
4231     \noexpand\forest@nodesort
4232     \noexpand\forest@sort@ascending
4233     {\forest@cn}%
4234     {\expandonce{\forest@nodesort@by}}%
4235   }%
4236 },
4237 sort'/.code={%
4238   \eapptotoks\forest@do@dynamics{%
4239     \noexpand\forest@nodesort
4240     \noexpand\forest@sort@descending
4241     {\forest@cn}%
4242     {\expandonce{\forest@nodesort@by}}%
4243   }%
4244 },
4245 sort by/.store in=\forest@nodesort@by,
4246 }
4247 \def\forest@replaceby@code#1#2{%#1=node spec,#2=insert after['][']
4248   \ifnum\forestove{@parent}=0
4249     \cseappto{forest@nodewalk@saved@dynamic nodes}{\forest@cn,}%
4250     \pgfkeysalso{alias=forest@last@node,set root={#1}}%
4251   \else
4252     \cseappto{forest@nodewalk@saved@dynamic nodes}{\forest@cn,}%
4253     \pgfkeysalso{alias=forest@last@node,#2={#1}}%
4254     \eapptotoks\forest@do@dynamics{%
4255       \noexpand\ifnum\noexpand\forestOve{\forest@cn}{@parent}=\forestove{@parent}
4256       \noexpand\forest@remove@node{\forest@cn}%
4257       \noexpand\fi
4258     }%
4259   \fi
4260 }
4261 \def\forest@nodesort#1#2#3{% #1 = direction, #2 = parent node, #3 = sort key
4262   \def\forest@nodesort@sortkey{#3}%
4263   \forest@fornode{#2}{\forest@nodesort@#1}%
4264 }
4265 \def\forest@nodesort@#1{%

```

```

4266 % prepare the array of child ids
4267 \c@pgf@counta=0
4268 \foresttoget{@first}\forest@nodesort@id
4269 \forest@loop
4270 \ifnum\forest@nodesort@id>0
4271   \advance\c@pgf@counta 1
4272   \csedef{forest@nodesort@the\c@pgf@counta}{\forest@nodesort@id}%
4273   \forestOget{\forest@nodesort@id}{@next}\forest@nodesort@id
4274 \forest@repeat
4275 % sort
4276 \foresttoget{n children}\forest@nodesort@n
4277 \forest@sort\forest@nodesort@cmpnodes\forest@nodesort@let#1{1}{\forest@nodesort@n}%
4278 % remove all children
4279 \foresttoget{@first}\forest@nodesort@id
4280 \forest@loop
4281 \ifnum\forest@nodesort@id>0
4282   \forest@node@Remove{\forest@nodesort@id}%
4283   \foresttoget{@first}\forest@nodesort@id
4284 \forest@repeat
4285 % insert the children in new order
4286 \c@pgf@counta=0
4287 \forest@loop
4288 \ifnum\c@pgf@counta<\forest@nodesort@n\relax
4289   \advance\c@pgf@counta 1
4290   \edef\temp{\csname forest@nodesort@the\c@pgf@counta\endcsname}%
4291   \forest@node@append{\csname forest@nodesort@the\c@pgf@counta\endcsname}%
4292 \forest@repeat
4293 }
4294 \def\forest@nodesort@cmpnodes#1#2{%
4295   \global\let\forest@nodesort@cmpresult\forest@sort@cmp@eq
4296   \foreach \forest@temp@pgfmath in \forest@nodesort@sortkey {%
4297     \forest@fornode{\csname forest@nodesort@#1\endcsname}{%
4298       \pgfmathparse{\forest@temp@pgfmath}\global\let\forest@global@tempa\pgfmathresult}%
4299     \forest@fornode{\csname forest@nodesort@#2\endcsname}{%
4300       \pgfmathparse{\forest@temp@pgfmath}\global\let\forest@global@tempb\pgfmathresult}%
4301     \ifdim\forest@global@tempa pt<\forest@global@tempb pt
4302       \global\let\forest@nodesort@cmpresult\forest@sort@cmp@lt
4303       \breakforeach
4304     \else
4305       \ifdim\forest@global@tempa pt>\forest@global@tempb pt
4306         \global\let\forest@nodesort@cmpresult\forest@sort@cmp@gt
4307         \breakforeach
4308       \fi
4309     \fi
4310   }%
4311   \forest@nodesort@cmpresult
4312 }
4313 \def\forest@nodesort@let#1#2{%
4314   \csletcs{forest@nodesort@#1}{forest@nodesort@#2}%
4315 }
4316 \forestset{
4317   define long step={last dynamic node}{style,must start at valid node=false}{%
4318     name=forest@last@node
4319   }
4320 }

```

## 7 Stages

```

4321 \def\forest@root{0}
4322 %%% begin listing region: stages
4323 \forestset{

```

```

4324 stages/.style={
4325   for root'={
4326     process keylist register=default preamble,
4327     process keylist register=preamble
4328   },
4329   process keylist=given options,
4330   process keylist=before typesetting nodes,
4331   typeset nodes stage,
4332   process keylist=before packing,
4333   pack stage,
4334   process keylist=before computing xy,
4335   compute xy stage,
4336   process keylist=before drawing tree,
4337   draw tree stage
4338 },
4339 typeset nodes stage/.style={for root'=typeset nodes},
4340 pack stage/.style={for root'=pack},
4341 compute xy stage/.style={for root'=compute xy},
4342 draw tree stage/.style={for root'=draw tree},
4343 }
4344 %%% end listing region: stages
4345 \forestset{
4346   process keylist/.code={%
4347     \forest@process@hook@keylist{#1}{#1 processing order/.try,processing order/.lastretry}},
4348   process keylist'/.code 2 args={\forest@process@hook@keylist@nodynamics{#1}{#2}},
4349   process keylist''/.code 2 args={\forest@process@hook@keylist@{#1}{#2}},
4350   process keylist register/.code={\forest@process@keylist@register{#1}},
4351   process delayed/.code={%
4352     \forest@havedelayedoptions{@delay}{#1}%
4353     \ifforest@havedelayedoptions
4354     \forest@process@hook@keylist@nodynamics{@delay}{#1}%
4355     \fi
4356   },
4357   do dynamics/.code={%
4358     \the\forest@do@dynamics
4359     \forest@do@dynamics{}%
4360     \forest@node@Compute@numeric@ts@info{\forest@root}%
4361   },
4362   declare keylist={given options}{},
4363   declare keylist={before typesetting nodes}{},
4364   declare keylist={before packing}{},
4365   declare keylist={before packing node}{},
4366   declare keylist={after packing node}{},
4367   declare keylist={before computing xy}{},
4368   declare keylist={before drawing tree}{},
4369   declare keylist={delay}{},
4370   delay n/.style 2 args={if=#1==0}{#2}{delay@n={#1}{#2}},
4371   delay@n/.style 2 args={
4372     if=#1==1}{delay={#2}}{delay={delay@n/.wrap pgfmath arg={{##1}{#2}}{#1-1}}}
4373   },
4374   if have delayed/.style 2 args={if have delayed'={processing order}{#1}{#2}},
4375   if have delayed'/.code n args=3{%
4376     \forest@havedelayedoptionsfalse
4377     \forest@forthis{%
4378       \forest@nodewalk{#1}{%
4379         TeX={%
4380           \foresttoget{delay}\forest@temp@delayed
4381           \ifdefempty\forest@temp@delayed}{\forest@havedelayedoptionstrue}%
4382         }%
4383       }%
4384     }%

```

```

4385 \ifforest@havedelayedoptions\pgfkeysalso{#2}\else\pgfkeysalso{#3}\fi
4386 },
4387 typeset nodes/.code={%
4388 \forest@drawtree@preservenodeboxes@false
4389 \forest@nodewalk
4390 {typeset nodes processing order/.try,processing order/.lastretry}%
4391 {TeX={\forest@node@typeset}}%
4392 },
4393 typeset nodes'/.code={%
4394 \forest@drawtree@preservenodeboxes@true
4395 \forest@nodewalk
4396 {typeset nodes processing order/.try,processing order/.lastretry}%
4397 {TeX={\forest@node@typeset}}%
4398 },
4399 typeset node/.code={%
4400 \forest@drawtree@preservenodeboxes@false
4401 \forest@node@typeset
4402 },
4403 pack/.code={\forest@pack},
4404 pack'/.code={\forest@pack@onlythisnode},
4405 compute xy/.code={\forest@node@computeabsolutepositions},
4406 draw tree box/.store in=\forest@drawtreebox,
4407 draw tree box,
4408 draw tree/.code={%
4409 \forest@drawtree@preservenodeboxes@false
4410 \forest@node@drawtree
4411 },
4412 draw tree'/.code={%
4413 \forest@drawtree@preservenodeboxes@true
4414 \forest@node@drawtree
4415 },
4416 %%% begin listing region: draw_tree_method
4417 draw tree method/.style={
4418 for nodewalk={
4419 draw tree nodes processing order/.try,
4420 draw tree processing order/.retry,
4421 processing order/.lastretry
4422 }{draw tree node},
4423 for nodewalk={
4424 draw tree edges processing order/.try,
4425 draw tree processing order/.retry,
4426 processing order/.lastretry
4427 }{draw tree edge},
4428 for nodewalk={
4429 draw tree tikz processing order/.try,
4430 draw tree processing order/.retry,
4431 processing order/.lastretry
4432 }{draw tree tikz}
4433 },
4434 %%% end listing region: draw_tree_method
4435 draw tree node/.code={\forest@draw@node},
4436 draw tree edge/.code={\forest@draw@edge},
4437 draw tree tikz/.code={\forest@draw@tikz},
4438 draw tree node'/.code={\forest@draw@node@},
4439 draw tree edge'/.code={\forest@draw@edge@},
4440 draw tree tikz'/.code={\forest@draw@tikz@},
4441 processing order/.nodewalk style={tree},
4442 %given options processing order/.style={processing order},
4443 %before typesetting nodes processing order/.style={processing order},
4444 %before packing processing order/.style={processing order},
4445 %before computing xy processing order/.style={processing order},

```

```

4446 %before drawing tree processing order/.style={processing order},
4447 }
4448 \newtoks\forest@do@dynamics
4449 \newif\ifforest@havedelayedoptions
4450 \def\forest@process@hook@keylist#1#2{%,#1=keylist,#2=processing order nodewalk
4451   \safeloop
4452     \forest@fornode{\forest@root}{\forest@process@hook@keylist@{#1}{#2}}%
4453     \expandafter\ifstremp\expandafter{\the\forest@do@dynamics}{-}{%
4454       \the\forest@do@dynamics
4455       \forest@do@dynamics={}%
4456       \forest@node@Compute@numeric@ts@info{\forest@root}%
4457     }%
4458   \forest@fornode{\forest@root}{\forest@havedelayedoptions{#1}{#2}}%
4459   \ifforest@havedelayedoptions
4460   \saferepeat
4461 }
4462 \def\forest@process@hook@keylist@nodynamics#1#2{%,#1=keylist,#2=processing order nodewalk
4463 % note: this macro works on (nodewalk starting at) the current node
4464   \safeloop
4465     \forest@forthis{\forest@process@hook@keylist@{#1}{#2}}%
4466     \forest@havedelayedoptions{#1}{#2}%
4467     \ifforest@havedelayedoptions
4468     \saferepeat
4469 }
4470 \def\forest@process@hook@keylist@#1#2{%,#1=keylist,#2=processing order nodewalk
4471   \forest@nodewalk{#2}{%
4472     TeX={%
4473       \forest@toget{#1}\forest@temp@keys
4474       \ifdefvoid\forest@temp@keys{-}{%
4475         \forest@set{#1}{-}%
4476         \expandafter\forest@set\expandafter{\forest@temp@keys}%
4477       }%
4478     }%
4479   }%
4480 }
4481 \def\forest@process@keylist@register#1{%,
4482   \edef\forest@marshal{%
4483     \noexpand\forest@set{\forest@register{#1}}%
4484   }\forest@marshal
4485 }

```

Clear the keylist, transfer delayed into it, and set `\ifforest@havedelayedoptions`.

```

4486 \def\forest@havedelayedoptions#1#2{%,#1 = keylist, #2=nodewalk
4487   \forest@havedelayedoptionsfalse
4488   \forest@forthis{%
4489     \forest@nodewalk{#2}{%
4490       TeX={%
4491         \forest@toget{delay}\forest@temp@delayed
4492         \ifdefempty\forest@temp@delayed{-}{\forest@havedelayedoptionstrue}%
4493         \forest@let{#1}\forest@temp@delayed
4494         \forest@set{delay}{-}%
4495       }%
4496     }%
4497   }%
4498 }

```

## 7.1 Typesetting nodes

```

4499 \def\forest@node@typeset{%
4500   \let\forest@next\forest@node@typeset@
4501   \forest@ifdefined{@box}{%
4502     \forest@toget{@box}\forest@temp

```



```

4503 \ifdefempty\forest@temp{%
4504   \locbox\forest@temp@box
4505   \forestolet{@box}\forest@temp@box
4506 }{%
4507   \ifforest@drawtree@preservenodeboxes@
4508   \let\forest@next\relax
4509   \fi
4510 }%
4511 }{%
4512   \locbox\forest@temp@box
4513   \forestolet{@box}\forest@temp@box
4514 }%
4515 \def\forest@node@typeset@restore{%
4516 \ifdefined\ifsa@tikz\forest@standalone@hack\fi
4517 \forest@next
4518 \forest@node@typeset@restore
4519 }
4520 \def\forest@standalone@hack{%
4521 \ifsa@tikz
4522   \let\forest@standalone@tikzpicture\tikzpicture
4523   \let\forest@standalone@endtikzpicture\endtikzpicture
4524   \let\tikzpicture\sa@orig@tikzpicture
4525   \let\endtikzpicture\sa@orig@endtikzpicture
4526   \def\forest@node@typeset@restore{%
4527     \let\tikzpicture\forest@standalone@tikzpicture
4528     \let\endtikzpicture\forest@standalone@endtikzpicture
4529   }%
4530 \fi
4531 }
4532 \newbox\forest@box
4533 \def\forest@pgf@notyetpositioned{not yet positionedPGFINTERNAL}
4534 \def\forest@node@typeset@{%
4535 \forestanchortotikzanchor{anchor}\forest@temp
4536 \edef\forest@marshal{%
4537 \noexpand\forestolet{anchor}\noexpand\forest@temp
4538 \noexpand\forest@node@typeset@@
4539 \noexpand\forestoset{anchor}{\forestov{anchor}}%
4540 }\forest@marshal
4541 }
4542 \def\forest@node@typeset@@{%
4543 \forestoget{name}\forest@nodename
4544 \edef\forest@temp@nodeformat{\forestove{node format}}%
4545 \gdef\forest@smuggle{%
4546 \setbox0=\hbox{%
4547 \begin{tikzpicture}%
4548 /forest/copy command key={/tikz/anchor}{/tikz/forest@orig@anchor},
4549 anchor/.style={%
4550 /forest/TeX={\forestanchortotikzanchor{##1}\forest@temp@anchor},
4551 forest@orig@anchor/.expand once=\forest@temp@anchor
4552 }}
4553 \pgfpositionnodelater{\forest@positionnodelater@save}%
4554 \forest@temp@nodeformat
4555 \pgfinterruptpath
4556 \pgfpointanchor{\forest@pgf@notyetpositioned\forest@nodename}{forestcomputenodeboundary}%
4557 \endpgfinterruptpath
4558 \end{tikzpicture}%
4559 }%
4560 \setbox\forestove{@box}=\box\forest@box % smuggle the box
4561 \forestolet{@boundary}\forest@global@boundary
4562 \forest@smuggle % ... and the rest
4563 }

```

```

4564
4565
4566 \forestset{
4567   declare readonly dimen={min x},
4568   declare readonly dimen={min y},
4569   declare readonly dimen={max x},
4570   declare readonly dimen={max y},
4571 }
4572 \def\forest@patch@enormouscoordinateboxbounds@plus#1{%
4573   \expandafter\ifstrequal\expandafter{#1}{16000.0pt}{\def#1{0.0pt}}{}%
4574 }
4575 \def\forest@patch@enormouscoordinateboxbounds@minus#1{%
4576   \expandafter\ifstrequal\expandafter{#1}{-16000.0pt}{\def#1{0.0pt}}{}%
4577 }
4578 \def\forest@positionnodelater@save{%
4579   \global\setbox\forest@box=\box\pgfpositionnodelaterbox
4580   \xappto\forest@smuggle{\noexpand\forestoset{later@name}{\pgfpositionnodelatername}}%
4581   % a bug in pgf? ---well, here's a patch
4582   \forest@patch@enormouscoordinateboxbounds@plus\pgfpositionnodelaterminx
4583   \forest@patch@enormouscoordinateboxbounds@plus\pgfpositionnodelaterminy
4584   \forest@patch@enormouscoordinateboxbounds@minus\pgfpositionnodelatermaxx
4585   \forest@patch@enormouscoordinateboxbounds@minus\pgfpositionnodelatermaxy
4586   % end of patch
4587   \xappto\forest@smuggle{\noexpand\forestoset{min x}{\pgfpositionnodelaterminx}}%
4588   \xappto\forest@smuggle{\noexpand\forestoset{min y}{\pgfpositionnodelaterminy}}%
4589   \xappto\forest@smuggle{\noexpand\forestoset{max x}{\pgfpositionnodelatermaxx}}%
4590   \xappto\forest@smuggle{\noexpand\forestoset{max y}{\pgfpositionnodelatermaxy}}%
4591 }
4592 \def\forest@node@forest@positionnodelater@restore{%
4593   \ifforest@drawtree@preservenodeboxes@
4594     \let\forest@boxorcopy\copy
4595   \else
4596     \let\forest@boxorcopy\box
4597   \fi
4598   \forestoget{@box}\forest@temp
4599   \setbox\pgfpositionnodelaterbox=\forest@boxorcopy\forest@temp
4600   \edef\pgfpositionnodelatername{\forestove{later@name}}%
4601   \edef\pgfpositionnodelaterminx{\forestove{min x}}%
4602   \edef\pgfpositionnodelaterminy{\forestove{min y}}%
4603   \edef\pgfpositionnodelatermaxx{\forestove{max x}}%
4604   \edef\pgfpositionnodelatermaxy{\forestove{max y}}%
4605   \ifforest@drawtree@preservenodeboxes@
4606   \else
4607     \forestoset{@box}{}%
4608   \fi
4609 }

```

## 7.2 Packing

Method `pack` should be called to calculate the positions of descendant nodes; the positions are stored in attributes `l` and `s` of these nodes, in a level/sibling coordinate system with origin at the parent's anchor.

```

4610 \def\forest@pack{%
4611   \pgfsyssoftpath@getcurrentpath\forest@pack@original@path
4612   \forest@pack@computetiers
4613   \forest@pack@computegrowthuniformity
4614   \forest@@pack
4615   \pgfsyssoftpath@setcurrentpath\forest@pack@original@path
4616 }
4617 \def\forest@@pack{%
4618   \ifnum\forestove{n children}>0
4619     \ifnum\forestove{uniform growth}>0

```

```

4620     \forest@pack@level@uniform
4621     \forest@pack@aligntiers@ofsubtree
4622     \forest@pack@sibling@uniform@recursive
4623 \else
4624     \forest@node@foreachchild{\forest@pack}%
4625     \forest@process@hook@keylist@nodynamics{before packing node}{current}%
4626     \forest@pack@level@nonuniform
4627     \forest@pack@aligntiers
4628     \forest@pack@sibling@uniform@applyreversed
4629 \fi
4630 \fi
4631 \forest@temp@keys
4632 \forest@process@hook@keylist@nodynamics{after packing node}{current}%
4633 }
4634 % \forest@set{recalculate tree boundary/.code={\forest@node@recalculate@edges}}
4635 % \def\forest@node@recalculate@edges{%
4636 %   \edef\forest@marshal{%
4637 %     \noexpand\forest@forthis{\noexpand\forest@node@getedges{\forest@ve{grow}}}%
4638 %   }\forest@marshal
4639 % }
4640 \def\forest@pack@onlythisnode{%
4641   \ifnum\forest@ve{n children}>0
4642     \forest@pack@computetiers
4643     \forest@pack@level@nonuniform
4644     \forest@pack@aligntiers
4645     \forest@node@foreachchild{\forest@set{s}{0pt}}%
4646     \forest@pack@sibling@uniform@applyreversed
4647   \fi
4648 }

```

Compute growth uniformity for the subtree. A tree grows uniformly if all its branching nodes have the same grow.

```

4649 \def\forest@pack@computegrowthuniformity{%
4650   \forest@node@foreachchild{\forest@pack@computegrowthuniformity}%
4651   \edef\forest@pack@cgu@uniformity{%
4652     \ifnum\forest@ve{n children}=0
4653       2\else 1\fi
4654   }%
4655   \forest@temp@grow\forest@pack@cgu@parentgrow
4656   \forest@node@foreachchild{%
4657     \ifnum\forest@ve{uniform growth}=0
4658       \def\forest@pack@cgu@uniformity{0}%
4659     \else
4660       \ifnum\forest@ve{uniform growth}=1
4661         \ifnum\forest@ve{grow}=\forest@pack@cgu@parentgrow\relax\else
4662           \def\forest@pack@cgu@uniformity{0}%
4663         \fi
4664       \fi
4665     \fi
4666   }%
4667   \forest@temp@a
4668   \forest@temp@b
4669   \expandafter\expandafter\expandafter\ifstrempy\expandafter\expandafter\expandafter{\expandafter\forest@temp@
4670     \forest@let{uniform growth}\forest@pack@cgu@uniformity
4671   }{%
4672     \forest@set{uniform growth}{0}%
4673   }%
4674 }

```

Pack children in the level dimension in a uniform tree.

```

4675 \def\forest@pack@level@uniform{%

```

```

4676 \let\forest@plu@minchildl\relax
4677 \forestoget{grow}\forest@plu@grow
4678 \forest@node@foreachchild{%
4679   \forest@node@getboundingrectangle@ls{\forest@plu@grow}%
4680   \advance\pgf@xa\forestove{1}\relax
4681   \ifx\forest@plu@minchildl\relax
4682     \edef\forest@plu@minchildl{\the\pgf@xa}%
4683   \else
4684     \ifdim\pgf@xa<\forest@plu@minchildl\relax
4685       \edef\forest@plu@minchildl{\the\pgf@xa}%
4686     \fi
4687   \fi
4688 }%
4689 \forest@node@getboundingrectangle@ls{\forest@plu@grow}%
4690 \pgfutil@tempdima=\pgf@xb\relax
4691 \advance\pgfutil@tempdima -\forest@plu@minchildl\relax
4692 \advance\pgfutil@tempdima \forestove{1 sep}\relax
4693 \ifdim\pgfutil@tempdima>0pt
4694   \forest@node@foreachchild{%
4695     \forestoerset{1}{\the\dimexpr\forestove{1}+\the\pgfutil@tempdima}%
4696   }%
4697 \fi
4698 \forest@node@foreachchild{%
4699   \ifnum\forestove{n children}>0
4700     \forest@pack@level@uniform
4701   \fi
4702 }%
4703 }

```

Pack children in the level dimension in a non-uniform tree. (Expects the children to be fully packed.)

```

4704 \def\forest@pack@level@nonuniform{%
4705   \let\forest@plu@minchildl\relax
4706   \forestoget{grow}\forest@plu@grow
4707   \forest@node@foreachchild{%
4708     \forest@node@getedge{negative}{\forest@plu@grow}{\forest@plnu@negativechildedge}%
4709     \forest@node@getedge{positive}{\forest@plu@grow}{\forest@plnu@positivechildedge}%
4710     \def\forest@plnu@childedge{\forest@plnu@negativechildedge\forest@plnu@positivechildedge}%
4711     \forest@path@getboundingrectangle@ls\forest@plnu@childedge{\forest@plu@grow}%
4712     \advance\pgf@xa\forestove{1}\relax
4713     \ifx\forest@plu@minchildl\relax
4714       \edef\forest@plu@minchildl{\the\pgf@xa}%
4715     \else
4716       \ifdim\pgf@xa<\forest@plu@minchildl\relax
4717         \edef\forest@plu@minchildl{\the\pgf@xa}%
4718       \fi
4719     \fi
4720   }%
4721   \forest@node@getboundingrectangle@ls{\forest@plu@grow}%
4722   \pgfutil@tempdima=\pgf@xb\relax
4723   \advance\pgfutil@tempdima -\forest@plu@minchildl\relax
4724   \advance\pgfutil@tempdima \forestove{1 sep}\relax
4725   \ifdim\pgfutil@tempdima>0pt
4726     \forest@node@foreachchild{%
4727       \forestoerset{1}{\the\dimexpr\the\pgfutil@tempdima+\forestove{1}}%
4728     }%
4729   \fi
4730 }

```

Align tiers.

```

4731 \def\forest@pack@aligntiers{%
4732   \forestoget{grow}\forest@temp@parentgrow
4733   \forestoget{@tiers}\forest@temp@tiers

```

```

4734 \forlistloop\forest@pack@aligntier@\forest@temp@tiers
4735 }
4736 \def\forest@pack@aligntiers@ofsubtree{%
4737 \forest@node@foreach{\forest@pack@aligntiers}%
4738 }
4739 \def\forest@pack@aligntiers@computeabsl{%
4740 \forestoleto{abs@1}{1}%
4741 \forest@node@foreachdescendant{\forest@pack@aligntiers@computeabsl@}%
4742 }
4743 \def\forest@pack@aligntiers@computeabsl@{%
4744 \forestoeset{abs@1}{\the\dimexpr\forestove{1}+\forestove{\forestove{\parent}}{abs@1}}%
4745 }
4746 \def\forest@pack@aligntier@#1{%
4747 \forest@pack@aligntiers@computeabsl
4748 \pgfutil@tempdima=-\maxdimen\relax
4749 \def\forest@temp@currenttier{#1}%
4750 \forest@node@foreach{%
4751 \forestoget{tier}\forest@temp@tier
4752 \ifx\forest@temp@currenttier\forest@temp@tier
4753 \ifdim\pgfutil@tempdima<\forestove{abs@1}\relax
4754 \pgfutil@tempdima=\forestove{abs@1}\relax
4755 \fi
4756 \fi
4757 }%
4758 \ifdim\pgfutil@tempdima=-\maxdimen\relax\else
4759 \forest@node@foreach{%
4760 \forestoget{tier}\forest@temp@tier
4761 \ifx\forest@temp@currenttier\forest@temp@tier
4762 \forestoeset{1}{\the\dimexpr\pgfutil@tempdima-\forestove{abs@1}+\forestove{1}}%
4763 \fi
4764 }%
4765 \fi
4766 }

```

Pack children in the sibling dimension in a uniform tree: recursion.

```

4767 \def\forest@pack@sibling@uniform@recursive{%
4768 \forest@node@foreachchild{\forest@pack@sibling@uniform@recursive}%
4769 \forest@pack@sibling@uniform@applyreversed
4770 }

```

Pack children in the sibling dimension in a uniform tree: applyreversed.

```

4771 \def\forest@pack@sibling@uniform@applyreversed{%
4772 \ifnum\forestove{n children}>1
4773 \ifnum\forestove{reversed}=0
4774 \forest@pack@sibling@uniform@main{first}{last}{next}{previous}%
4775 \else
4776 \forest@pack@sibling@uniform@main{last}{first}{previous}{next}%
4777 \fi
4778 \else
4779 \ifnum\forestove{n children}=1

```

No need to run packing, but we still need to align the children.

```

4780 \csname forest@calign@\forestove{calign}\endcsname
4781 \fi
4782 \fi
4783 }

```

Pack children in the sibling dimension in a uniform tree: the main routine.

```

4784 \def\forest@pack@sibling@uniform@main#1#2#3#4{%

```

Loop through the children. At each iteration, we compute the distance between the negative edge of the current child and the positive edge of the block of the previous children, and then set the `s` attribute of the current child accordingly.

We start the loop with the second (to last) child, having initialized the positive edge of the previous children to the positive edge of the first child.

```

4785 \forestoget{#@#1}\forest@child
4786 \edef\forest@marshal{%
4787   \noexpand\forest@fornode{\forestove{#@#1}}{%
4788     \noexpand\forest@node@getedge
4789       {positive}%
4790       {\forestove{grow}}}%
4791   \noexpand\forest@temp@edge
4792 }%
4793 }\forest@marshal
4794 \forest@pack@pgfpoint@child@position\forest@child
4795 \let\forest@previous@positive@edge\pgfutil@empty
4796 \forest@extendpath\forest@previous@positive@edge\forest@temp@edge{}%
4797 \forest@get{\forest@child}{#@#3}\forest@child

```

Loop until the current child is the null node.

```

4798 \edef\forest@previous@child@s{0pt}%
4799 \safeloop
4800 \unless\ifnum\forest@child=0

```

Get the negative edge of the child.

```

4801 \edef\forest@temp{%
4802   \noexpand\forest@fornode{\forest@child}{%
4803     \noexpand\forest@node@getedge
4804       {negative}%
4805       {\forestove{grow}}}%
4806   \noexpand\forest@temp@edge
4807 }%
4808 }\forest@temp

```

Set  $\pgf@x$  and  $\pgf@y$  to the position of the child (in the coordinate system of this node).

```

4809 \forest@pack@pgfpoint@child@position\forest@child

```

Translate the edge of the child by the child's position.

```

4810 \let\forest@child@negative@edge\pgfutil@empty
4811 \forest@extendpath\forest@child@negative@edge\forest@temp@edge{}%

```

Setup the grow line: the angle is given by this node's `grow` attribute.

```

4812 \forest@setupgrowline{\forestove{grow}}%

```

Get the distance (wrt the grow line) between the positive edge of the previous children and the negative edge of the current child. (The distance can be negative!)

```

4813 \forest@distance@between@edge@paths\forest@previous@positive@edge\forest@child@negative@edge\forest@csdis

```

If the distance is `\relax`, the projections of the edges onto the grow line don't overlap: do nothing.

Otherwise, shift the current child so that its distance to the block of previous children is `s_sep`.

```

4814 \ifx\forest@csdistance\relax
4815   %\forest@set{\forest@child}{s}{\forest@previous@child@s}%
4816 \else
4817   \advance\pgfutil@tempdimb-\forest@csdistance\relax
4818   \advance\pgfutil@tempdimb\forestove{s sep}\relax
4819   \forest@set{\forest@child}{s}{\the\dimexpr\forest@ove{\forest@child}{s}-\forest@csdistance+\forestove{s
4820 \fi

```

Retain monotonicity (is this ok?). (This problem arises when the adjacent children's 1 are too far apart.)

```

4821 \ifdim\forest@ove{\forest@child}{s}<\forest@previous@child@s\relax
4822   \forest@set{\forest@child}{s}{\forest@previous@child@s}%
4823 \fi

```

Prepare for the next iteration: add the current child's positive edge to the positive edge of the previous children, and set up the next current child.

```

4824 \forest@get{\forest@child}{s}\forest@child@s
4825 \edef\forest@previous@child@s{\forest@child@s}%

```

```

4826 \edef\forest@temp{%
4827   \noexpand\forest@fornode{\forest@child}{%
4828     \noexpand\forest@node@getedge
4829       {positive}}%
4830     {\forestove{grow}}}%
4831   \noexpand\forest@temp@edge
4832 }%
4833 }\forest@temp
4834 \forest@pack@pgfpoint@childposition\forest@child
4835 \forest@extendpath\forest@previous@positive@edge\forest@temp@edge{%
4836 \forest@getpositivetightedgeofpath\forest@previous@positive@edge\forest@previous@positive@edge
4837 \forest@get{\forest@child}{@#3}\forest@child
4838 \saferepeat

```

Shift the position of all children to achieve the desired alignment of the parent and its children.

```

4839 \csname forest@calign@\forestove{calign}\endcsname
4840 }

```

Get the position of child #1 in the current node, in node's l-s coordinate system.

```

4841 \def\forest@pack@pgfpoint@childposition#1{%
4842   {%
4843     \pgftransformreset
4844     \pgftransformrotate{\forestove{grow}}%
4845     \forest@fornode{#1}{%
4846       \pgfpointtransformed{\pgfpoint{\forestove{1}}{\forestove{s}}}%
4847     }%
4848   }%
4849 }

```

Get the position of the node in the grow (#1)-rotated coordinate system.

```

4850 \def\forest@pack@pgfpoint@positioningrow#1{%
4851   {%
4852     \pgftransformreset
4853     \pgftransformrotate{#1}%
4854     \pgfpointtransformed{\pgfpoint{\forestove{1}}{\forestove{s}}}%
4855   }%
4856 }

```

Child alignment.

```

4857 \def\forest@calign@s@shift#1{%
4858   \pgfutil@tempdima=#1\relax
4859   \forest@node@foreachchild{%
4860     \forestoeset{s}{\the\dimexpr\forestove{s}+\pgfutil@tempdima}%
4861   }%
4862 }
4863 \def\forest@calign@child{%
4864   \forest@calign@s@shift{-\forestove{\forest@node@normbarthchildid{\forestove{calign primary child}}}{s}}%
4865 }
4866 \csdef{forest@calign@child edge}{%
4867   {%
4868     \edef\forest@temp@child{\forest@node@normbarthchildid{\forestove{calign primary child}}}%
4869     \pgftransformreset
4870     \pgftransformrotate{\forestove{grow}}%
4871     \pgfpointtransformed{\pgfpoint{\forestove{\forest@temp@child}{1}}{\forestove{\forest@temp@child}{s}}}%
4872     \pgf@xa=\pgf@x\relax\pgf@ya=\pgf@y\relax
4873     \forest@Pointanchor{\forest@temp@child}{child anchor}%
4874     \advance\pgf@xa\pgf@x\relax\advance\pgf@ya\pgf@y\relax
4875     \forest@pointanchor{parent anchor}%
4876     \advance\pgf@xa-\pgf@x\relax\advance\pgf@ya-\pgf@y\relax
4877     \edef\forest@marshal{%
4878       \noexpand\pgftransformreset
4879       \noexpand\pgftransformrotate{-\forestove{grow}}%

```

```

4880     \noexpand\pgfpointransformed{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}%
4881   }\forest@marshal
4882 }%
4883 \forest@calign@s@shift{\the\dimexpr-\the\pgf@y}%
4884 }
4885 \csdef{forest@calign@midpoint}{%
4886   \forest@calign@s@shift{\the\dimexpr Opt -%
4887     (\forestOve{\forest@node@nornbarthchildid{\forestove{calign primary child}}}{s}%
4888     +\forestOve{\forest@node@nornbarthchildid{\forestove{calign secondary child}}}{s}%
4889     )/2\relax
4890   }%
4891 }
4892 \csdef{forest@calign@edge midpoint}{%
4893   {%
4894     \edef\forest@temp@firstchild{\forest@node@nornbarthchildid{\forestove{calign primary child}}}%
4895     \edef\forest@temp@secondchild{\forest@node@nornbarthchildid{\forestove{calign secondary child}}}%
4896     \pgftransformreset
4897     \pgftransformrotate{\forestove{grow}}%
4898     \pgfpointransformed{\pgfqpoint{\forestOve{\forest@temp@firstchild}{1}}{\forestOve{\forest@temp@firstchild}{1}}}%
4899     \pgf@xa=\pgf@x\relax\pgf@ya=\pgf@y\relax
4900     \forest@Pointanchor{\forest@temp@firstchild}{child anchor}%
4901     \advance\pgf@xa\pgf@x\relax\advance\pgf@ya\pgf@y\relax
4902     \edef\forest@marshal{%
4903       \noexpand\pgfpointransformed{\noexpand\pgfqpoint{\forestOve{\forest@temp@secondchild}{1}}{\forestOve{\forest@temp@secondchild}{1}}}%
4904     }\forest@marshal
4905     \advance\pgf@xa\pgf@x\relax\advance\pgf@ya\pgf@y\relax
4906     \forest@Pointanchor{\forest@temp@secondchild}{child anchor}%
4907     \advance\pgf@xa\pgf@x\relax\advance\pgf@ya\pgf@y\relax
4908     \divide\pgf@xa2 \divide\pgf@ya2
4909     \forest@pointanchor{parent anchor}%
4910     \advance\pgf@xa-\pgf@x\relax\advance\pgf@ya-\pgf@y\relax
4911     \edef\forest@marshal{%
4912       \noexpand\pgftransformreset
4913       \noexpand\pgftransformrotate{-\forestove{grow}}%
4914       \noexpand\pgfpointransformed{\noexpand\pgfqpoint{\the\pgf@xa}{\the\pgf@ya}}%
4915     }\forest@marshal
4916   }%
4917 \forest@calign@s@shift{\the\dimexpr-\the\pgf@y}%
4918 }

```

Aligns the children to the center of the angles given by the options `calign_first_angle` and `calign_second_angle` and spreads them additionally if needed to fill the whole space determined by the option. The version `fixed_angles` calculates the angles between node anchors; the version `fixes_edge_angles` calculates the angles between the node edges.

```

4919 \csdef{forest@calign@fixed angles}{%
4920   \ifnum\forestove{n children}>1
4921     \edef\forest@ca@first@child{\forest@node@nornbarthchildid{\forestove{calign primary child}}}%
4922     \edef\forest@ca@second@child{\forest@node@nornbarthchildid{\forestove{calign secondary child}}}%
4923     \ifnum\forestove{reversed}=1
4924       \let\forest@temp\forest@ca@first@child
4925       \let\forest@ca@first@child\forest@ca@second@child
4926       \let\forest@ca@second@child\forest@temp
4927     \fi
4928     \forestOget{\forest@ca@first@child}{1}\forest@ca@first@l
4929     \forestOget{\forest@ca@second@child}{1}\forest@ca@second@l
4930     \pgfmathsetlengthmacro\forest@ca@desired@s@distance{%
4931       tan(\forestove{calign secondary angle})*\forest@ca@second@l
4932       -tan(\forestove{calign primary angle})*\forest@ca@first@l
4933     }%
4934     \forestOget{\forest@ca@first@child}{s}\forest@ca@first@s
4935     \forestOget{\forest@ca@second@child}{s}\forest@ca@second@s

```



```

4936 \pgfmathsetlengthmacro\forest@ca@actual@s@distance{%
4937   \forest@ca@second@s-\forest@ca@first@s}%
4938 \ifdim\forest@ca@desired@s@distance>\forest@ca@actual@s@distance\relax
4939   \ifdim\forest@ca@actual@s@distance=Opt
4940     \pgfmathsetlength\pgfutil@tempdima{\tan(\forestove{calign primary angle})*\forest@ca@second@l}%
4941     \pgfmathsetlength\pgfutil@tempdimb{\forest@ca@desired@s@distance/(\forestove{n children}-1)}%
4942     \forest@node@foreachchild{%
4943       \forestoeset{s}{\the\pgfutil@tempdima}%
4944       \advance\pgfutil@tempdima\pgfutil@tempdimb
4945     }%
4946     \def\forest@calign@anchor{Opt}%
4947   \else
4948     \pgfmathsetmacro\forest@ca@ratio{%
4949       \forest@ca@desired@s@distance/\forest@ca@actual@s@distance}%
4950     \forest@node@foreachchild{%
4951       \pgfmathsetlengthmacro\forest@temp{\forest@ca@ratio*\forestove{s}}%
4952       \forestolet{s}\forest@temp
4953     }%
4954     \pgfmathsetlengthmacro\forest@calign@anchor{%
4955       -tan(\forestove{calign primary angle})*\forest@ca@first@l}%
4956   \fi
4957 \else
4958   \ifdim\forest@ca@desired@s@distance<\forest@ca@actual@s@distance\relax
4959     \pgfmathsetlengthmacro\forest@ca@ratio{%
4960       \forest@ca@actual@s@distance/\forest@ca@desired@s@distance}%
4961     \forest@node@foreachchild{%
4962       \pgfmathsetlengthmacro\forest@temp{\forest@ca@ratio*\forestove{l}}%
4963       \forestolet{l}\forest@temp
4964     }%
4965     \forest0get{\forest@ca@first@child}{l}\forest@ca@first@l
4966     \pgfmathsetlengthmacro\forest@calign@anchor{%
4967       -tan(\forestove{calign primary angle})*\forest@ca@first@l}%
4968   \fi
4969 \fi
4970 \forest@calign@s@shift{-\forest@calign@anchor}%
4971 \fi
4972 }
4973 \csdef{forest@calign@fixed edge angles}{%
4974   \ifnum\forestove{n children}>1
4975     \edef\forest@ca@first@child{\forest@node@nornbarthchildid{\forestove{calign primary child}}}%
4976     \edef\forest@ca@second@child{\forest@node@nornbarthchildid{\forestove{calign secondary child}}}%
4977     \ifnum\forestove{reversed}=1
4978       \let\forest@temp\forest@ca@first@child
4979       \let\forest@ca@first@child\forest@ca@second@child
4980       \let\forest@ca@second@child\forest@temp
4981     \fi
4982     \forest0get{\forest@ca@first@child}{l}\forest@ca@first@l
4983     \forest0get{\forest@ca@second@child}{l}\forest@ca@second@l
4984     \forest@pointanchor{parent anchor}%
4985     \edef\forest@ca@parent@anchor@s{\the\pgf@x}%
4986     \edef\forest@ca@parent@anchor@l{\the\pgf@y}%
4987     \forest@Pointanchor{\forest@ca@first@child}{child anchor}%
4988     \edef\forest@ca@first@child@anchor@s{\the\pgf@x}%
4989     \edef\forest@ca@first@child@anchor@l{\the\pgf@y}%
4990     \forest@Pointanchor{\forest@ca@second@child}{child anchor}%
4991     \edef\forest@ca@second@child@anchor@s{\the\pgf@x}%
4992     \edef\forest@ca@second@child@anchor@l{\the\pgf@y}%
4993     \pgfmathsetlengthmacro\forest@ca@desired@second@edge@s{\tan(\forestove{calign secondary angle})*
4994       (\forest@ca@second@l-\forest@ca@second@child@anchor@l+\forest@ca@parent@anchor@l)}%
4995     \pgfmathsetlengthmacro\forest@ca@desired@first@edge@s{\tan(\forestove{calign primary angle})*
4996       (\forest@ca@first@l-\forest@ca@first@child@anchor@l+\forest@ca@parent@anchor@l)}

```

```

4997 \pgfmathsetlengthmacro\forest@ca@desired@s@distance{\forest@ca@desired@second@edge@s-\forest@ca@desired@f
4998 \forest@get{\forest@ca@first@child}{s}\forest@ca@first@s
4999 \forest@get{\forest@ca@second@child}{s}\forest@ca@second@s
5000 \pgfmathsetlengthmacro\forest@ca@actual@s@distance{%
5001 \forest@ca@second@s+\forest@ca@second@child@anchor@s
5002 -\forest@ca@first@s-\forest@ca@first@child@anchor@s}%
5003 \ifdim\forest@ca@desired@s@distance>\forest@ca@actual@s@distance\relax
5004 \ifdim\forest@ca@actual@s@distance=0pt
5005 \forest@toget{n children}\forest@temp@n@children
5006 \forest@node@foreachchild{%
5007 \forest@pointanchor{child anchor}%
5008 \edef\forest@temp@child@anchor@s{\the\pgf{x}%
5009 \pgfmathsetlengthmacro\forest@temp{%
5010 \forest@ca@desired@first@edge@s+(\forest@ve{n}-1)*\forest@ca@desired@s@distance/(\forest@temp@n@
5011 \forest@let{s}\forest@temp
5012 }%
5013 \def\forest@calign@anchor{0pt}%
5014 \else
5015 \pgfmathsetmacro\forest@ca@ratio{%
5016 \forest@ca@desired@s@distance/\forest@ca@actual@s@distance}%
5017 \forest@node@foreachchild{%
5018 \forest@pointanchor{child anchor}%
5019 \edef\forest@temp@child@anchor@s{\the\pgf{x}%
5020 \pgfmathsetlengthmacro\forest@temp{%
5021 \forest@ca@ratio*(%
5022 \forest@ve{s}-\forest@ca@first@s
5023 +\forest@temp@child@anchor@s-\forest@ca@first@child@anchor@s)%
5024 +\forest@ca@first@s
5025 +\forest@ca@first@child@anchor@s-\forest@temp@child@anchor@s}%
5026 \forest@let{s}\forest@temp
5027 }%
5028 \pgfmathsetlengthmacro\forest@calign@anchor{%
5029 -tan(\forest@ve{calign primary angle})*(\forest@ca@first@l-\forest@ca@first@child@anchor@l+\forest@
5030 +\forest@ca@first@child@anchor@s-\forest@ca@parent@anchor@s
5031 }%
5032 \fi
5033 \else
5034 \ifdim\forest@ca@desired@s@distance<\forest@ca@actual@s@distance\relax
5035 \pgfmathsetlengthmacro\forest@ca@ratio{%
5036 \forest@ca@actual@s@distance/\forest@ca@desired@s@distance}%
5037 \forest@node@foreachchild{%
5038 \forest@pointanchor{child anchor}%
5039 \edef\forest@temp@child@anchor@l{\the\pgf{y}%
5040 \pgfmathsetlengthmacro\forest@temp{%
5041 \forest@ca@ratio*(%
5042 \forest@ve{l}+\forest@ca@parent@anchor@l-\forest@temp@child@anchor@l)
5043 -\forest@ca@parent@anchor@l+\forest@temp@child@anchor@l}%
5044 \forest@let{l}\forest@temp
5045 }%
5046 \forest@get{\forest@ca@first@child}{l}\forest@ca@first@l
5047 \pgfmathsetlengthmacro\forest@calign@anchor{%
5048 -tan(\forest@ve{calign primary angle})*(\forest@ca@first@l+\forest@ca@parent@anchor@l-\forest@temp@
5049 +\forest@ca@first@child@anchor@s-\forest@ca@parent@anchor@s
5050 }%
5051 \fi
5052 \fi
5053 \forest@calign@s@shift{-\forest@calign@anchor}%
5054 \fi
5055 }

```

Get edge: #1 = positive/negative, #2 = grow (in degrees), #3 = the control sequence receiving

the resulting path. The edge is taken from the cache (attribute #1@edge@#2) if possible; otherwise, both positive and negative edge are computed and stored in the cache.

```

5056 \def\forest@node@getedge#1#2#3{%
5057   \forestoget{#1@edge@#2}#3%
5058   \ifx#3\relax
5059     \forest@node@foreachchild{%
5060       \forest@node@getedge{#1}#2}{\forest@temp@edge}%
5061     }%
5062     \forest@forthis{\forest@node@getedges{#2}}%
5063     \forestoget{#1@edge@#2}#3%
5064   \fi
5065 }

```

Get edges. #1 = grow (in degrees). The result is stored in attributes negative@edge@#1 and positive@edge@#1. This method expects that the children's edges are already cached.

```

5066 \def\forest@node@getedges#1{%
  Run the computation in a TEX group.
5067   %{}
  Setup the grow line.
5068   \forest@setupgrowline{#1}%
  Get the edge of the node itself.
5069   \ifnum\forestove{ignore}=0
5070     \forestoget{@boundary}\forest@node@boundary
5071   \else
5072     \def\forest@node@boundary{}%
5073   \fi
5074   \csname forest@getboth\forestove{fit}edgesofpath\endcsname
5075     \forest@node@boundary\forest@negative@node@edge\forest@positive@node@edge
5076   \forestolet{negative@edge@#1}\forest@negative@node@edge
5077   \forestolet{positive@edge@#1}\forest@positive@node@edge

```

Add the edges of the children.

```

5078   \get@edges@merge{negative}{#1}%
5079   \get@edges@merge{positive}{#1}%
5080   %}%
5081 }

```

Merge the #1 (=negative or positive) edge of the node with #1 edges of the children. #2 = grow angle.

```

5082 \def\get@edges@merge#1#2{%
5083   \ifnum\forestove{n children}>0
5084     \forestoget{#1@edge@#2}\forest@node@edge

```

Remember the node's parent anchor and add it to the path (for breaking).

```

5085   \forest@pointanchor{parent anchor}%
5086   \edef\forest@getedge@pa1{\the\pgf@x}%
5087   \edef\forest@getedge@pa2{\the\pgf@y}%
5088   \eappto\forest@node@edge{\noexpand\pgfsyssoftpath@movetotoken{\forest@getedge@pa1}{\forest@getedge@pa2}}

```

Switch to this node's (1,s) coordinate system (origin at the node's anchor).

```

5089   \pgfgettransform\forest@temp@transform
5090   \pgftransformreset
5091   \pgftransformrotate{\forestove{grow}}%

```

Get the child's (cached) edge, translate it by the child's position, and add it to the path holding all edges. Also add the edge from parent to the child to the path. This gets complicated when the child and/or parent anchor is empty, i.e. automatic border: we can get self-intersecting paths. So we store all the parent-child edges to a safe place first, compute all the possible breaking points (i.e. all the points in node@edge path), and break the parent-child edges on these points.

```

5092   \def\forest@all@edges{}%
5093   \forest@node@foreachchild{%

```

```

5094 \foresttoget{#1@edge@#2}\forest@temp@edge
5095 \pgfpointtransformed{\pgfpoint{\forestove{1}}{\forestove{s}}}%
5096 \forest@extendpath\forest@node@edge\forest@temp@edge{%
5097 \ifnum\forestove{ignore edge}=0
5098 \pgfpointadd
5099 {\pgfpointtransformed{\pgfpoint{\forestove{1}}{\forestove{s}}}}%
5100 {\forest@pointanchor{child anchor}}%
5101 \pgfgetlastxy{\forest@getedge@ca@1}{\forest@getedge@ca@s}%
5102 \eappto\forest@all@edges{%
5103 \noexpand\pgfsyssoftpath@movetotoken{\forest@getedge@pa@1}{\forest@getedge@pa@s}%
5104 \noexpand\pgfsyssoftpath@linetotoken{\forest@getedge@ca@1}{\forest@getedge@ca@s}%
5105 }%
5106 % this deals with potential overlap of the edges:
5107 \eappto\forest@node@edge{\noexpand\pgfsyssoftpath@movetotoken{\forest@getedge@ca@1}{\forest@getedge@ca@s}}%
5108 \fi
5109 }%
5110 \ifdefempty{\forest@all@edges}{}%
5111 \pgfintersectionofpaths{\pgfsetpath\forest@all@edges}{\pgfsetpath\forest@node@edge}%
5112 \def\forest@edgenode@intersections{%
5113 \forest@merge@intersectionloop
5114 \eappto\forest@node@edge{\expandonce{\forest@all@edges}\expandonce{\forest@edgenode@intersections}}%
5115 }%
5116 \pgfsettransform\forest@temp@transform

```

Process the path into an edge and store the edge.

```

5117 \csname forest@get#1\forestove{fit}edgeofpath\endcsname\forest@node@edge\forest@node@edge
5118 \forestolet{#1@edge@#2}\forest@node@edge
5119 \fi
5120 }
5121 %\newloop\forest@merge@loop
5122 \def\forest@merge@intersectionloop{%
5123 \c@pgf@counta=0
5124 \forest@loop
5125 \ifnum\c@pgf@counta<\pgfintersectionsolutions\relax
5126 \advance\c@pgf@counta1
5127 \pgfpointintersectionsolution{\the\c@pgf@counta}%
5128 \eappto\forest@edgenode@intersections{\noexpand\pgfsyssoftpath@movetotoken
5129 {\the\pgf@x}{\the\pgf@y}}%
5130 \forest@repeat
5131 }

```

Get the bounding rectangle of the node (without descendants). #1 = grow.

```

5132 \def\forest@node@getboundingrectangle@ls#1{%
5133 \foresttoget{@boundary}\forest@node@boundary
5134 \forest@path@getboundingrectangle@ls\forest@node@boundary{#1}%
5135 }

```

Applies the current coordinate transformation to the points in the path #1. Returns via the current path (so that the coordinate transformation can be set up as local).

```

5136 \def\forest@pgfpathtransformed#1{%
5137 \forest@save@pgfsyssoftpath@tokendefs
5138 \let\pgfsyssoftpath@movetotoken\forest@pgfpathtransformed@moveto
5139 \let\pgfsyssoftpath@linetotoken\forest@pgfpathtransformed@lineto
5140 \pgfsyssoftpath@setcurrentpath\pgfutil@empty
5141 #1%
5142 \forest@restore@pgfsyssoftpath@tokendefs
5143 }
5144 \def\forest@pgfpathtransformed@moveto#1#2{%
5145 \forest@pgfpathtransformed@op\pgfsyssoftpath@moveto{#1}{#2}%
5146 }
5147 \def\forest@pgfpathtransformed@lineto#1#2{%
5148 \forest@pgfpathtransformed@op\pgfsyssoftpath@lineto{#1}{#2}%

```

```

5149 }
5150 \def\forest@pgfpathtransformed@op#1#2#3{%
5151   \pgfpointtransformed{\pgfpoint{#2}{#3}}%
5152   \edef\forest@temp{%
5153     \noexpand#1{\the\pgf@x}{\the\pgf@y}%
5154   }%
5155   \forest@temp
5156 }

```

### 7.2.1 Tiers

Compute tiers to be aligned at a node. The result is saved in attribute `@tiers`.

```

5157 \def\forest@pack@computetiers{%
5158   {%
5159     \forest@pack@tiers@getalltiersinsubtree
5160     \forest@pack@tiers@computetierhierarchy
5161     \forest@pack@tiers@findcontainers
5162     \forest@pack@tiers@raisecontainers
5163     \forest@pack@tiers@computeprocessingorder
5164     \gdef\forest@smuggle{%
5165       \forest@pack@tiers@write
5166     }%
5167     \forest@node@foreach{\forestoset{@tiers}{}}%
5168     \forest@smuggle
5169 }

```

Puts all tiers contained in the subtree into attribute `tiers`.

```

5170 \def\forest@pack@tiers@getalltiersinsubtree{%
5171   \ifnum\forestove{n children}>0
5172     \forest@node@foreachchild{\forest@pack@tiers@getalltiersinsubtree}%
5173   \fi
5174   \forestoget{tier}\forest@temp@mytier
5175   \def\forest@temp@mytiers{%
5176     \ifdefempty\forest@temp@mytier{}{%
5177       \listead\forest@temp@mytiers\forest@temp@mytier
5178     }%
5179   \ifnum\forestove{n children}>0
5180     \forest@node@foreachchild{%
5181       \forestoget{tiers}\forest@temp@tiers
5182       \forlistloop\forest@pack@tiers@forhandlerA\forest@temp@tiers
5183     }%
5184   \fi
5185   \forestolet{tiers}\forest@temp@mytiers
5186 }
5187 \def\forest@pack@tiers@forhandlerA#1{%
5188   \ifinlist{#1}\forest@temp@mytiers{}{%
5189     \listead\forest@temp@mytiers{#1}%
5190   }%
5191 }

```

Compute a set of higher and lower tiers for each tier. Tier A is higher than tier B iff a node on tier A is an ancestor of a node on tier B.

```

5192 \def\forest@pack@tiers@computetierhierarchy{%
5193   \def\forest@tiers@ancestors{%
5194     \forestoget{tiers}\forest@temp@mytiers
5195     \forlistloop\forest@pack@tiers@cth@init\forest@temp@mytiers
5196   \forest@pack@tiers@computetierhierarchy@
5197 }
5198 \def\forest@pack@tiers@cth@init#1{%
5199   \csdef{forest@tiers@higher@#1}{}%
5200   \csdef{forest@tiers@lower@#1}{}%

```

```

5201 }
5202 \def\forest@pack@tiers@computetierhierarchy@{%
5203   \foresttoget{tier}\forest@temp@mytier
5204   \ifdefempty\forest@temp@mytier{}-%
5205     \forlistloop\forest@pack@tiers@forhandlerB\forest@tiers@ancestors
5206     \listseadd\forest@tiers@ancestors\forest@temp@mytier
5207   }%
5208   \forest@node@foreachchild{%
5209     \forest@pack@tiers@computetierhierarchy@
5210   }%
5211   \foresttoget{tier}\forest@temp@mytier
5212   \ifdefempty\forest@temp@mytier{}-%
5213     \forest@listedel\forest@tiers@ancestors\forest@temp@mytier
5214   }%
5215 }
5216 \def\forest@pack@tiers@forhandlerB#1{%
5217   \def\forest@temp@tier{#1}%
5218   \ifx\forest@temp@tier\forest@temp@mytier
5219     \PackageError{forest}{Circular tier hierarchy (tier \forest@temp@mytier)}{}%
5220   \fi
5221   \ifinlistcs{#1}{\forest@tiers@higher@\forest@temp@mytier}{}-%
5222     \listcsadd{\forest@tiers@higher@\forest@temp@mytier}{#1}%
5223   \xifinlistcs\forest@temp@mytier{\forest@tiers@lower@#1}{}-%
5224     \listcseadd{\forest@tiers@lower@#1}{\forest@temp@mytier}%
5225 }
5226 \def\forest@pack@tiers@findcontainers{%
5227   \foresttoget{tiers}\forest@temp@tiers
5228   \forlistloop\forest@pack@tiers@findcontainer\forest@temp@tiers
5229 }
5230 \def\forest@pack@tiers@findcontainer#1{%
5231   \def\forest@temp@tier{#1}%
5232   \foresttoget{tier}\forest@temp@mytier
5233   \ifx\forest@temp@tier\forest@temp@mytier
5234     \csedef{\forest@tiers@container@#1}{\forest@cn}%
5235   \else\@escapeif{%
5236     \forest@pack@tiers@findcontainerA{#1}%
5237   }\fi%
5238 }
5239 \def\forest@pack@tiers@findcontainerA#1{%
5240   \c@pgf@counta=0
5241   \forest@node@foreachchild{%
5242     \foresttoget{tiers}\forest@temp@tiers
5243     \ifinlist{#1}\forest@temp@tiers{%
5244       \advance\c@pgf@counta 1
5245       \let\forest@temp@child\forest@cn
5246     }{}%
5247   }%
5248   \ifnum\c@pgf@counta>1
5249     \csedef{\forest@tiers@container@#1}{\forest@cn}%
5250   \else\@escapeif{% surely =1
5251     \forest@fornode{\forest@temp@child}{%
5252       \forest@pack@tiers@findcontainer{#1}%
5253     }%
5254   }\fi
5255 }
5256 \def\forest@pack@tiers@raisecontainers{%
5257   \foresttoget{tiers}\forest@temp@mytiers
5258   \forlistloop\forest@pack@tiers@rc@forhandlerA\forest@temp@mytiers
5259 }
5260 \def\forest@pack@tiers@rc@forhandlerA#1{%
5261   \edef\forest@tiers@temptier{#1}%

```

```

5262 \letcs\forest@tiers@containernodeoftier{forest@tiers@container@#1}%
5263 \letcs\forest@temp@lowertiers{forest@tiers@lower@#1}%
5264 \forlistloop\forest@pack@tiers@rc@forhandlerB\forest@temp@lowertiers
5265 }
5266 \def\forest@pack@tiers@rc@forhandlerB#1{%
5267 \letcs\forest@tiers@containernodeoflowertier{forest@tiers@container@#1}%
5268 \forest0get{\forest@tiers@containernodeoflowertier}{content}\lowercontent
5269 \forest0get{\forest@tiers@containernodeoftier}{content}\uppercontent
5270 \forest@fornode{\forest@tiers@containernodeoflowertier}{%
5271 \forest@ifancestorof
5272 {\forest@tiers@containernodeoftier}
5273 {\csletcs{forest@tiers@container@forest@tiers@temptier}{forest@tiers@container@#1}}%
5274 {}}%
5275 }%
5276 }
5277 \def\forest@pack@tiers@computeprocessingorder{%
5278 \def\forest@tiers@processingorder{}%
5279 \forestoget{tiers}\forest@tiers@cpo@tierstodo
5280 \safeloop
5281 \ifdefempty\forest@tiers@cpo@tierstodo{\forest@tempfalse}{\forest@temptrue}%
5282 \ifforest@temp
5283 \def\forest@tiers@cpo@tiersremaining{}%
5284 \def\forest@tiers@cpo@tiersindependent{}%
5285 \forlistloop\forest@pack@tiers@cpo@forhandlerA\forest@tiers@cpo@tierstodo
5286 \ifdefempty\forest@tiers@cpo@tiersindependent{%
5287 \PackageError{forest}{Circular tiers!}{-}{-}%
5288 \forlistloop\forest@pack@tiers@cpo@forhandlerB\forest@tiers@cpo@tiersremaining
5289 \let\forest@tiers@cpo@tierstodo\forest@tiers@cpo@tiersremaining
5290 \saferepeat
5291 }
5292 \def\forest@pack@tiers@cpo@forhandlerA#1{%
5293 \ifcsemtier{forest@tiers@higher@#1}{%
5294 \listadd\forest@tiers@cpo@tiersindependent{#1}%
5295 \listadd\forest@tiers@processingorder{#1}%
5296 }{%
5297 \listadd\forest@tiers@cpo@tiersremaining{#1}%
5298 }%
5299 }
5300 \def\forest@pack@tiers@cpo@forhandlerB#1{%
5301 \def\forest@pack@tiers@cpo@aremainingtier{#1}%
5302 \forlistloop\forest@pack@tiers@cpo@forhandlerC\forest@tiers@cpo@tiersindependent
5303 }
5304 \def\forest@pack@tiers@cpo@forhandlerC#1{%
5305 \ifinlistcs{#1}{forest@tiers@higher@\forest@pack@tiers@cpo@aremainingtier}{%
5306 \forest@listcsdel{forest@tiers@higher@\forest@pack@tiers@cpo@aremainingtier}{#1}%
5307 }{}%
5308 }
5309 \def\forest@pack@tiers@write{%
5310 \forlistloop\forest@pack@tiers@write@forhandler\forest@tiers@processingorder
5311 }
5312 \def\forest@pack@tiers@write@forhandler#1{%
5313 \forest@fornode{\csname forest@tiers@container@#1\endcsname}{%
5314 \forest@pack@tiers@check{#1}%
5315 }%
5316 \xappto\forest@smuggle{%
5317 \noexpand\listadd
5318 \forest0m{\csname forest@tiers@container@#1\endcsname}{@tiers}%
5319 {#1}%
5320 }%
5321 }
5322 % checks if the tier is compatible with growth changes and calign=node/edge angle

```

```

5323 \def\forest@pack@tiers@check#1{%
5324   \def\forest@temp@currenttier{#1}%
5325   \forest@node@foreachdescendant{%
5326     \ifnum\forestove{grow}=\forestOve{\forestove{@parent}}{grow}
5327     \else
5328       \forest@pack@tiers@check@grow
5329     \fi
5330     \ifnum\forestove{n children}>1
5331       \forestoget{calign}\forest@temp
5332       \ifx\forest@temp\forest@pack@tiers@check@nodeangle
5333         \forest@pack@tiers@check@calign
5334       \fi
5335       \ifx\forest@temp\forest@pack@tiers@check@edgeangle
5336         \forest@pack@tiers@check@calign
5337       \fi
5338     \fi
5339   }%
5340 }
5341 \def\forest@pack@tiers@check@nodeangle{node angle}%
5342 \def\forest@pack@tiers@check@edgeangle{edge angle}%
5343 \def\forest@pack@tiers@check@grow{%
5344   \forestoget{content}\forest@temp@content
5345   \let\forest@temp@currentnode\forest@cn
5346   \forest@node@foreachdescendant{%
5347     \forestoget{tier}\forest@temp
5348     \ifx\forest@temp@currenttier\forest@temp
5349       \forest@pack@tiers@check@grow@error
5350     \fi
5351   }%
5352 }
5353 \def\forest@pack@tiers@check@grow@error{%
5354   \PackageError{forest}{Tree growth direction changes in node \forest@temp@currentnode\space
5355     (content: \forest@temp@content), while tier '\forest@temp' is specified for nodes both
5356     out- and inside the subtree rooted in node \forest@temp@currentnode. This will not work.}{}%
5357 }
5358 \def\forest@pack@tiers@check@calign{%
5359   \forest@node@foreachchild{%
5360     \forestoget{tier}\forest@temp
5361     \ifx\forest@temp@currenttier\forest@temp
5362       \forest@pack@tiers@check@calign@warning
5363     \fi
5364   }%
5365 }
5366 \def\forest@pack@tiers@check@calign@warning{%
5367   \PackageWarning{forest}{Potential option conflict: node \forestove{@parent} (content:
5368     '\forestOve{\forestove{@parent}}{content}') was given 'calign=\forestove{calign}', while its
5369     child \forest@cn\space (content: '\forestove{content}') was given 'tier=\forestove{tier}'.
5370     The parent's 'calign' will only work if the child was the lowest node on its tier before the
5371     alignment.}%
5372 }

```

## 7.2.2 Node boundary

Compute the node boundary: it will be put in the pgf's current path. The computation is done within a generic anchor so that the shape's saved anchors and macros are available.

```

5373 \pgfdeclaregenericanchor{forestcomputenodeboundary}{%
5374   \letcs\forest@temp@boundary@macro\forest@compute@node@boundary@#1}%
5375   \ifcsname forest@compute@node@boundary@#1\endcsname
5376     \csname forest@compute@node@boundary@#1\endcsname
5377   \else

```



```

5378 \forest@compute@node@boundary@rectangle
5379 \fi
5380 \pgfsyssoftpath@getcurrentpath\forest@temp
5381 \global\let\forest@global@boundary\forest@temp
5382 }
5383 \def\forest@mt#1{%
5384 \expandafter\pgfpointanchor\expandafter{\pgfreferencednodename}{#1}%
5385 \pgfsyssoftpath@moveto{\the\pgf@x}{\the\pgf@y}%
5386 }%
5387 \def\forest@lt#1{%
5388 \expandafter\pgfpointanchor\expandafter{\pgfreferencednodename}{#1}%
5389 \pgfsyssoftpath@lineto{\the\pgf@x}{\the\pgf@y}%
5390 }%
5391 \def\forest@compute@node@boundary@coordinate{%
5392 \forest@mt{center}%
5393 }
5394 \def\forest@compute@node@boundary@circle{%
5395 \forest@mt{east}%
5396 \forest@lt{north east}%
5397 \forest@lt{north}%
5398 \forest@lt{north west}%
5399 \forest@lt{west}%
5400 \forest@lt{south west}%
5401 \forest@lt{south}%
5402 \forest@lt{south east}%
5403 \forest@lt{east}%
5404 }
5405 \def\forest@compute@node@boundary@rectangle{%
5406 \forest@mt{south west}%
5407 \forest@lt{south east}%
5408 \forest@lt{north east}%
5409 \forest@lt{north west}%
5410 \forest@lt{south west}%
5411 }
5412 \def\forest@compute@node@boundary@diamond{%
5413 \forest@mt{east}%
5414 \forest@lt{north}%
5415 \forest@lt{west}%
5416 \forest@lt{south}%
5417 \forest@lt{east}%
5418 }
5419 \let\forest@compute@node@boundary@ellipse\forest@compute@node@boundary@circle
5420 \def\forest@compute@node@boundary@trapezium{%
5421 \forest@mt{top right corner}%
5422 \forest@lt{top left corner}%
5423 \forest@lt{bottom left corner}%
5424 \forest@lt{bottom right corner}%
5425 \forest@lt{top right corner}%
5426 }
5427 \def\forest@compute@node@boundary@semicircle{%
5428 \forest@mt{arc start}%
5429 \forest@lt{north}%
5430 \forest@lt{east}%
5431 \forest@lt{north east}%
5432 \forest@lt{apex}%
5433 \forest@lt{north west}%
5434 \forest@lt{west}%
5435 \forest@lt{arc end}%
5436 \forest@lt{arc start}%
5437 }
5438 %\newloop\forest@computenodeboundary@loop

```

```

5439 \csdef{forest@compute@node@boundary@regular polygon}{%
5440   \forest@mt{corner 1}%
5441   \c@pgf@counta=\sides\relax
5442   \forest@loop
5443   \ifnum\c@pgf@counta>0
5444     \forest@lt{corner \the\c@pgf@counta}%
5445     \advance\c@pgf@counta-1
5446     \forest@repeat
5447   }%
5448 \def\forest@compute@node@boundary@star{%
5449   \forest@mt{outer point 1}%
5450   \c@pgf@counta=\totalstarpoints\relax
5451   \divide\c@pgf@counta2
5452   \forest@loop
5453   \ifnum\c@pgf@counta>0
5454     \forest@lt{inner point \the\c@pgf@counta}%
5455     \forest@lt{outer point \the\c@pgf@counta}%
5456     \advance\c@pgf@counta-1
5457     \forest@repeat
5458   }%
5459 \csdef{forest@compute@node@boundary@isosceles triangle}{%
5460   \forest@mt{apex}%
5461   \forest@lt{left corner}%
5462   \forest@lt{right corner}%
5463   \forest@lt{apex}%
5464 }
5465 \def\forest@compute@node@boundary@kite{%
5466   \forest@mt{upper vertex}%
5467   \forest@lt{left vertex}%
5468   \forest@lt{lower vertex}%
5469   \forest@lt{right vertex}%
5470   \forest@lt{upper vertex}%
5471 }
5472 \def\forest@compute@node@boundary@dart{%
5473   \forest@mt{tip}%
5474   \forest@lt{left tail}%
5475   \forest@lt{tail center}%
5476   \forest@lt{right tail}%
5477   \forest@lt{tip}%
5478 }
5479 \csdef{forest@compute@node@boundary@circular sector}{%
5480   \forest@mt{sector center}%
5481   \forest@lt{arc start}%
5482   \forest@lt{arc center}%
5483   \forest@lt{arc end}%
5484   \forest@lt{sector center}%
5485 }
5486 \def\forest@compute@node@boundary@cylinder{%
5487   \forest@mt{top}%
5488   \forest@lt{after top}%
5489   \forest@lt{before bottom}%
5490   \forest@lt{bottom}%
5491   \forest@lt{after bottom}%
5492   \forest@lt{before top}%
5493   \forest@lt{top}%
5494 }
5495 \cslet{forest@compute@node@boundary@forbidden sign}\forest@compute@node@boundary@circle
5496 \cslet{forest@compute@node@boundary@magnifying glass}\forest@compute@node@boundary@circle
5497 \def\forest@compute@node@boundary@cloud{%
5498   \getradii
5499   \forest@mt{puff 1}%

```

```

5500 \c@pgf@counta=\puffs\relax
5501 \forest@loop
5502 \ifnum\c@pgf@counta>0
5503   \forest@lt{puff \the\c@pgf@counta}%
5504   \advance\c@pgf@counta-1
5505 \forest@repeat
5506 }
5507 \def\forest@compute@node@boundary@starburst{
5508   \calculatestarburstpoints
5509   \forest@mt{outer point 1}%
5510   \c@pgf@counta=\totalpoints\relax
5511   \divide\c@pgf@counta2
5512   \forest@loop
5513   \ifnum\c@pgf@counta>0
5514     \forest@lt{inner point \the\c@pgf@counta}%
5515     \forest@lt{outer point \the\c@pgf@counta}%
5516     \advance\c@pgf@counta-1
5517   \forest@repeat
5518 }%
5519 \def\forest@compute@node@boundary@signal{%
5520   \forest@mt{east}%
5521   \forest@lt{south east}%
5522   \forest@lt{south west}%
5523   \forest@lt{west}%
5524   \forest@lt{north west}%
5525   \forest@lt{north east}%
5526   \forest@lt{east}%
5527 }
5528 \def\forest@compute@node@boundary@tape{%
5529   \forest@mt{north east}%
5530   \forest@lt{60}%
5531   \forest@lt{north}%
5532   \forest@lt{120}%
5533   \forest@lt{north west}%
5534   \forest@lt{south west}%
5535   \forest@lt{240}%
5536   \forest@lt{south}%
5537   \forest@lt{310}%
5538   \forest@lt{south east}%
5539   \forest@lt{north east}%
5540 }
5541 \csdef{forest@compute@node@boundary@single arrow}{%
5542   \forest@mt{tip}%
5543   \forest@lt{after tip}%
5544   \forest@lt{after head}%
5545   \forest@lt{before tail}%
5546   \forest@lt{after tail}%
5547   \forest@lt{before head}%
5548   \forest@lt{before tip}%
5549   \forest@lt{tip}%
5550 }
5551 \csdef{forest@compute@node@boundary@double arrow}{%
5552   \forest@mt{tip 1}%
5553   \forest@lt{after tip 1}%
5554   \forest@lt{after head 1}%
5555   \forest@lt{before head 2}%
5556   \forest@lt{before tip 2}%
5557   \forest@mt{tip 2}%
5558   \forest@lt{after tip 2}%
5559   \forest@lt{after head 2}%
5560   \forest@lt{before head 1}%

```

```

5561 \forest@lt{before tip 1}%
5562 \forest@lt{tip 1}%
5563 }
5564 \csdef{forest@compute@node@boundary@arrow box}{%
5565 \forest@mt{before north arrow}%
5566 \forest@lt{before north arrow head}%
5567 \forest@lt{before north arrow tip}%
5568 \forest@lt{north arrow tip}%
5569 \forest@lt{after north arrow tip}%
5570 \forest@lt{after north arrow head}%
5571 \forest@lt{after north arrow}%
5572 \forest@lt{north east}%
5573 \forest@lt{before east arrow}%
5574 \forest@lt{before east arrow head}%
5575 \forest@lt{before east arrow tip}%
5576 \forest@lt{east arrow tip}%
5577 \forest@lt{after east arrow tip}%
5578 \forest@lt{after east arrow head}%
5579 \forest@lt{after east arrow}%
5580 \forest@lt{south east}%
5581 \forest@lt{before south arrow}%
5582 \forest@lt{before south arrow head}%
5583 \forest@lt{before south arrow tip}%
5584 \forest@lt{south arrow tip}%
5585 \forest@lt{after south arrow tip}%
5586 \forest@lt{after south arrow head}%
5587 \forest@lt{after south arrow}%
5588 \forest@lt{south west}%
5589 \forest@lt{before west arrow}%
5590 \forest@lt{before west arrow head}%
5591 \forest@lt{before west arrow tip}%
5592 \forest@lt{west arrow tip}%
5593 \forest@lt{after west arrow tip}%
5594 \forest@lt{after west arrow head}%
5595 \forest@lt{after west arrow}%
5596 \forest@lt{north west}%
5597 \forest@lt{before north arrow}%
5598 }
5599 \cslet{forest@compute@node@boundary@circle split}\forest@compute@node@boundary@circle
5600 \cslet{forest@compute@node@boundary@circle solidus}\forest@compute@node@boundary@circle
5601 \cslet{forest@compute@node@boundary@ellipse split}\forest@compute@node@boundary@ellipse
5602 \cslet{forest@compute@node@boundary@rectangle split}\forest@compute@node@boundary@rectangle
5603 \def\forest@compute@node@boundary@@callout{%
5604 \beforecalloutpointer
5605 \pgfsyssoftpath@moveto{\the\pgf@x}{\the\pgf@y}%
5606 \calloutpointeranchor
5607 \pgfsyssoftpath@lineto{\the\pgf@x}{\the\pgf@y}%
5608 \aftercalloutpointer
5609 \pgfsyssoftpath@lineto{\the\pgf@x}{\the\pgf@y}%
5610 }
5611 \csdef{forest@compute@node@boundary@rectangle callout}{%
5612 \forest@compute@node@boundary@rectangle
5613 \rectanglecalloutpoints
5614 \forest@compute@node@boundary@@callout
5615 }
5616 \csdef{forest@compute@node@boundary@ellipse callout}{%
5617 \forest@compute@node@boundary@ellipse
5618 \ellipsecalloutpoints
5619 \forest@compute@node@boundary@@callout
5620 }
5621 \csdef{forest@compute@node@boundary@cloud callout}{%

```

```

5622 \forest@compute@node@boundary@cloud
5623 % at least a first approx...
5624 \forest@mt{center}%
5625 \forest@lt{pointer}%
5626 }%
5627 \csdef{forest@compute@node@boundary@cross out}{%
5628 \forest@mt{south east}%
5629 \forest@lt{north west}%
5630 \forest@mt{south west}%
5631 \forest@lt{north east}%
5632 }%
5633 \csdef{forest@compute@node@boundary@strike out}{%
5634 \forest@mt{north east}%
5635 \forest@lt{south west}%
5636 }%
5637 \csdef{forest@compute@node@boundary@rounded rectangle}{%
5638 \forest@mt{east}%
5639 \forest@lt{north east}%
5640 \forest@lt{north}%
5641 \forest@lt{north west}%
5642 \forest@lt{west}%
5643 \forest@lt{south west}%
5644 \forest@lt{south}%
5645 \forest@lt{south east}%
5646 \forest@lt{east}%
5647 }%
5648 \csdef{forest@compute@node@boundary@chamfered rectangle}{%
5649 \forest@mt{before south west}%
5650 \forest@mt{after south west}%
5651 \forest@lt{before south east}%
5652 \forest@lt{after south east}%
5653 \forest@lt{before north east}%
5654 \forest@lt{after north east}%
5655 \forest@lt{before north west}%
5656 \forest@lt{after north west}%
5657 \forest@lt{before south west}%
5658 }%

```

### 7.3 Compute absolute positions

Computes absolute positions of descendants relative to this node. Stores the results in attributes `x` and `y`.

```

5659 \def\forest@node@computeabsolutepositions{%
5660 \edef\forest@marshal{%
5661 \noexpand\forest@node@foreachchild{%
5662 \noexpand\forest@node@computeabsolutepositions@{\forestove{x}}{\forestove{y}}{\forestove{grow}}%
5663 }%
5664 }\forest@marshal
5665 }
5666 \def\forest@node@computeabsolutepositions@#1#2#3{%
5667 \pgfpointadd{\pgfpoint{#1}{#2}}{%
5668 \pgfpointadd{\pgfpolar{#3}{\forestove{1}}}{\pgfpolar{90 + #3}{\forestove{s}}}}%
5669 \pgfgetlastxy\forest@temp@x\forest@temp@y
5670 \forestolet{x}\forest@temp@x
5671 \forestolet{y}\forest@temp@y
5672 \edef\forest@marshal{%
5673 \noexpand\forest@node@foreachchild{%
5674 \noexpand\forest@node@computeabsolutepositions@{\forest@temp@x}{\forest@temp@y}{\forestove{grow}}%
5675 }%
5676 }\forest@marshal

```

5677 }

## 7.4 Drawing the tree

```
5678 \newif\ifforest@drawtree@preservenodeboxes@
5679 \def\forest@node@drawtree{%
5680   \expandafter\ifstrequal\expandafter{\forest@drawtreebox}{\pgfkeysnovalue}{%
5681     \let\forest@drawtree@beginbox\relax
5682     \let\forest@drawtree@endbox\relax
5683   }{%
5684     \edef\forest@drawtree@beginbox{\global\setbox\forest@drawtreebox=\hbox\bgroup}%
5685     \let\forest@drawtree@endbox\egroup
5686   }%
5687 \ifforest@external@
5688 \ifforest@externalize@tree@
5689   \forest@temptrue
5690 \else
5691   \tikzifexternalizing{%
5692     \ifforest@was@tikzexternalwasenable
5693     \forest@temptrue
5694     \pgfkeys{/tikz/external/optimize=false}%
5695     \let\forest@drawtree@beginbox\relax
5696     \let\forest@drawtree@endbox\relax
5697   \else
5698     \forest@tempfalse
5699   \fi
5700 }{%
5701   \forest@tempfalse
5702 }%
5703 \fi
5704 \ifforest@temp
5705   \advance\forest@externalize@inner@n 1
5706   \edef\forest@externalize@filename{%
5707     \tikzexternalrealjob-forest-\forest@externalize@outer@n
5708     \ifnum\forest@externalize@inner@n=0 \else.\the\forest@externalize@inner@n\fi}%
5709   \expandafter\tikzsetnextfilename\expandafter{\forest@externalize@filename}%
5710   \tikzexternalenable
5711   \pgfkeysalso{/tikz/external/remake next,/tikz/external/export next}%
5712 \fi
5713 \ifforest@externalize@tree@
5714   \typeout{forest: Invoking a recursive call to generate the external picture
5715     '\forest@externalize@filename' for the following context+code:
5716     '\expandafter\detokenize\expandafter{\forest@externalize@id}'}%
5717 \fi
5718 \fi
5719 %
5720 \ifforesttikzcshack
5721   \let\forest@original@tikz@parse@node\tikz@parse@node
5722   \let\tikz@parse@node\forest@tikz@parse@node
5723 \fi
5724 \pgfkeysgetvalue{/forest/begin draw/.@cmd}\forest@temp@begindraw
5725 \pgfkeysgetvalue{/forest/end draw/.@cmd}\forest@temp@enddraw
5726 \edef\forest@marshal{%
5727   \noexpand\forest@drawtree@beginbox
5728   \expandonce{\forest@temp@begindraw\pgfkeysnovalue\pgfeov}%
5729   \noexpand\forest@node@drawtree@
5730   \expandonce{\forest@temp@enddraw\pgfkeysnovalue\pgfeov}%
5731   \noexpand\forest@drawtree@endbox
5732 } \forest@marshal
5733 \ifforesttikzcshack
```

```

5734 \let\tikz@parse@node\forest@original@tikz@parse@node
5735 \fi
5736 %
5737 \ifforest@external@
5738 \ifforest@externalize@tree@
5739 \tikzexternaldisable
5740 \eappto\forest@externalize@checkimages{%
5741 \noexpand\forest@includeexternal@check{\forest@externalize@filename}%
5742 }%
5743 \expandafter\ifstrequal\expandafter{\forest@drawtreebox}{\pgfkeysnovalue}{%
5744 \eappto\forest@externalize@loadimages{%
5745 \noexpand\forest@includeexternal{\forest@externalize@filename}%
5746 }%
5747 }{%
5748 \eappto\forest@externalize@loadimages{%
5749 \noexpand\forest@includeexternal@box\forest@drawtreebox{\forest@externalize@filename}%
5750 }%
5751 }%
5752 \fi
5753 \fi
5754 }
5755 \def\forest@drawtree@root{0}
5756 \def\forest@node@drawtree@{%
5757 \forest@forthis{%
5758 \forest@saveandrestoremacro\forest@drawtree@root{%
5759 \edef\forest@drawtree@root{\forest@cn}%
5760 \forestset{draw tree method}%
5761 }%
5762 }%
5763 \forest@node@ifnamedefined{forest@baseline@node}{%
5764 \edef\forest@temp{%
5765 \noexpand\pgfsetbaselinepointlater{%
5766 \noexpand\pgfpointanchor
5767 {\forest@Ove{\forest@node@Nametoid{forest@baseline@node}}{name}}
5768 {\forest@Ove{\forest@node@Nametoid{forest@baseline@node}}{anchor}}
5769 }%
5770 }\forest@temp
5771 }{}%
5772 }
5773 \def\forest@draw@node{%
5774 \ifnum\forest@ve{phantom}=0
5775 \forest@draw@node@
5776 \fi
5777 }
5778 \def\forest@draw@node@{%
5779 \forest@node@forest@position@node@later@restore
5780 \ifforest@drawtree@preservenodeboxes@
5781 \pgfnodealias{forest@temp}{\forest@ve{later@name}}%
5782 \fi
5783 \pgfpositionnodenow{\pgfqpoint{\forest@ve{x}}{\forest@ve{y}}}%
5784 \ifforest@drawtree@preservenodeboxes@
5785 \pgfnodealias{\forest@ve{later@name}}{forest@temp}%
5786 \fi
5787 }
5788 \def\forest@draw@edge{%
5789 \ifnum\forest@cn=\forest@drawtree@root\relax\else
5790 \ifnum\forest@ve{phantom}=0
5791 \ifnum\forest@Ove{\forest@ve{@parent}}{phantom}=0
5792 \forest@draw@edge@
5793 \fi
5794 \fi

```

```

5795 \fi
5796 }
5797 \def\forest@draw@edge@{%
5798 \edef\forest@temp{\forest@edge path}}\forest@temp
5799 }
5800 \def\forest@draw@tikz{%
5801 \ifnum\forest@ve{phantom}=0
5802 \forest@draw@tikz@
5803 \fi
5804 }
5805 \def\forest@draw@tikz@{%
5806 \forest@ve{tikz}%
5807 }

```

## 8 Geometry

A  $\alpha$  *grow line* is a line through the origin at angle  $\alpha$ . The following macro sets up the grow line, which can then be used by other code (the change is local to the  $\TeX$  group). More precisely, two normalized vectors are set up: one  $(x_g, y_g)$  on the grow line, and one  $(x_s, y_s)$  orthogonal to it—to get  $(x_s, y_s)$ , rotate  $(x_g, y_g)$   $90^\circ$  counter-clockwise.

```

5808 \newdimen\forest@xg
5809 \newdimen\forest@yg
5810 \newdimen\forest@xs
5811 \newdimen\forest@ys
5812 \def\forest@setupgrowline#1{%
5813 \edef\forest@grow{#1}%
5814 \pgfpointpolar\forest@grow{1pt}%
5815 \forest@xg=\pgf@x
5816 \forest@yg=\pgf@y
5817 \forest@xs=-\pgf@y
5818 \forest@ys=\pgf@x
5819 }

```

### 8.1 Projections

The following macro belongs to the `\pgfpoint...` family: it projects point #1 on the grow line. (The result is returned via `\pgf@x` and `\pgf@y`.) The implementation is based on code from `tikzlibrarycalc`, but optimized for projecting on grow lines, and split to optimize serial usage in `\forest@projectpath`.

```

5820 \def\forest@pgfpointprojectiontogrowline#1{%
5821 \pgf@process{#1}%

```

Calculate the scalar product of  $(x, y)$  and  $(x_g, y_g)$ : that's the distance of  $(x, y)$  to the grow line.

```

5822 \pgfutil@tempdima=\pgf@sys@tonumber{\pgf@x}\forest@xg%
5823 \advance\pgfutil@tempdima by\pgf@sys@tonumber{\pgf@y}\forest@yg%

```

The projection is  $(x_g, y_g)$  scaled by the distance.

```

5824 \global\pgf@x=\pgf@sys@tonumber{\pgfutil@tempdima}\forest@xg%
5825 \global\pgf@y=\pgf@sys@tonumber{\pgfutil@tempdima}\forest@yg%
5826 }}

```

The following macro calculates the distance of point #2 to the grow line and stores the result in  $\TeX$ -dimension #1. The distance is the scalar product of the point vector and the normalized vector orthogonal to the grow line.

```

5827 \def\forest@distancetogrowline#1#2{%
5828 \pgf@process{#2}%
5829 #1=\pgf@sys@tonumber{\pgf@x}\forest@xs\relax
5830 \advance#1 by\pgf@sys@tonumber{\pgf@y}\forest@ys\relax
5831 }

```



Note that the distance to the grow line is positive for points on one of its sides and negative for points on the other side. (It is positive on the side which  $(x_s, y_s)$  points to.) We thus say that the grow line partitions the plane into a *positive* and a *negative* side.

The following macro projects all segment edges (“points”) of a simple<sup>2</sup> path #1 onto the grow line. The result is an array of tuples  $(x_o, y_o, x_p, y_p)$ , where  $x_o$  and  $y_o$  stand for the *original* point, and  $x_p$  and  $y_p$  stand for its *projection*. The prefix of the array is given by #2. If the array already exists, the new items are appended to it. The array is not sorted: the order of original points in the array is their order in the path. The computation does not destroy the current path. All result-macros have local scope.

The macro is just a wrapper for `\forest@projectpath@process`.

```
5832 \let\forest@pp@n\relax
5833 \def\forest@projectpath@togrowline#1#2{%
5834   \edef\forest@pp@prefix{#2}%
5835   \forest@save@pgfsyssoftpath@tokendefs
5836   \let\pgfsyssoftpath@movetotoken\forest@projectpath@processpoint
5837   \let\pgfsyssoftpath@linetotoken\forest@projectpath@processpoint
5838   \c@pgf@counta=0
5839   #1%
5840   \csedef{#2n}{\the\c@pgf@counta}%
5841   \forest@restore@pgfsyssoftpath@tokendefs
5842 }
```

For each point, remember the point and its projection to grow line.

```
5843 \def\forest@projectpath@processpoint#1#2{%
5844   \pgfqpoint{#1}{#2}%
5845   \expandafter\edef\c@pgf@counta\forest@pp@prefix\the\c@pgf@counta x\endcsname{\the\pgf@x}%
5846   \expandafter\edef\c@pgf@counta\forest@pp@prefix\the\c@pgf@counta y\endcsname{\the\pgf@y}%
5847   \forest@pgfpointprojection@togrowline{}%
5848   \expandafter\edef\c@pgf@counta\forest@pp@prefix\the\c@pgf@counta xp\endcsname{\the\pgf@x}%
5849   \expandafter\edef\c@pgf@counta\forest@pp@prefix\the\c@pgf@counta yp\endcsname{\the\pgf@y}%
5850   \advance\c@pgf@counta 1\relax
5851 }
```

Sort the array (prefix #1) produced by `\forest@projectpath@togrowline` by  $(x_p, y_p)$ , in the ascending order.

```
5852 \def\forest@sort@projections#1{%
5853   % todo: optimize in cases when we know that the array is actually a
5854   % merger of sorted arrays; when does this happen? in
5855   % distance_between_paths, and when merging the edges of the parent
5856   % and its children in a uniform growth tree
5857   \edef\forest@ppi@inputprefix{#1}%
5858   \c@pgf@counta=\c@pgf@counta#1n\endcsname\relax
5859   \advance\c@pgf@counta -1
5860   \forest@sort\forest@ppiraw@cmp\forest@ppiraw@let\forest@sort@ascending{0}{\the\c@pgf@counta}%
5861 }
```

The following macro processes the data gathered by (possibly more than one invocation of) `\forest@projectpath@togrowline` into array with prefix #1. The resulting data is the following.

- Array of projections (prefix #2)
  - its items are tuples  $(x, y)$  (the array is sorted by  $x$  and  $y$ ), and
  - an inner array of original points (prefix #2N@, where  $N$  is the index of the item in array #2. The items of #2N@ are  $x$ ,  $y$  and  $d$ :  $x$  and  $y$  are the coordinates of the original point;  $d$  is its distance to the grow line. The inner array is not sorted.
- A dictionary #2: keys are the coordinates  $(x, y)$  of the original points; a value is the index of the original point’s projection in array #2.<sup>3</sup>

<sup>2</sup>A path is *simple* if it consists of only move-to and line-to operations.

<sup>3</sup>At first sight, this information could be cached “at the source”: by `forest@pgfpointprojection@togrowline`. However, due to imprecise intersecting (in `breakpath`), we cheat and merge very adjacent projection points, expecting that the points to project to the merged projection point. All this depends on the given path, so a generic cache is not feasible.

```

5862 \def\forest@processprojectioninfo#1#2{%
5863   \edef\forest@ppi@inputprefix{#1}%

```

Loop (counter \c@pgf@counta) through the sorted array of raw data.

```

5864   \c@pgf@counta=0
5865   \c@pgf@countb=-1
5866   \safeloop
5867   \ifnum\c@pgf@counta<\csname#1n\endcsname\relax

```

Check if the projection tuple in the current raw item equals the current projection.

```

5868     \letcs\forest@xo{#1\the\c@pgf@counta xo}%
5869     \letcs\forest@yo{#1\the\c@pgf@counta yo}%
5870     \letcs\forest@xp{#1\the\c@pgf@counta xp}%
5871     \letcs\forest@yp{#1\the\c@pgf@counta yp}%
5872     \ifnum\c@pgf@countb<0
5873       \forest@equaltotolerancefalse
5874     \else
5875       \forest@equaltotolerance
5876       {\pgfqpoint\forest@xp\forest@yp}%
5877       {\pgfqpoint
5878        {\csname#2\the\c@pgf@countb x\endcsname}%
5879        {\csname#2\the\c@pgf@countb y\endcsname}%
5880       }%
5881     \fi
5882     \ifforest@equaltotolerance\else

```

It not, we will append a new item to the outer result array.

```

5883       \advance\c@pgf@countb 1
5884       \cslet{#2\the\c@pgf@countb x}\forest@xp
5885       \cslet{#2\the\c@pgf@countb y}\forest@yp
5886       \csdef{#2\the\c@pgf@countb @n}{0}%
5887     \fi

```

If the projection is actually a projection of one a point in our path:

```

5888     % todo: this is ugly!
5889     \ifdefined\forest@xo\ifx\forest@xo\relax\else
5890     \ifdefined\forest@yo\ifx\forest@yo\relax\else

```

Append the point of the current raw item to the inner array of points projecting to the current projection.

```

5891       \forest@append@point@to@inner@array
5892       \forest@xo\forest@yo
5893       {#2\the\c@pgf@countb @}%

```

Put a new item in the dictionary: key = the original point, value = the projection index.

```

5894       \csedef{#2(\forest@xo,\forest@yo)}{\the\c@pgf@countb}%
5895     \fi\fi
5896   \fi\fi

```

Clean-up the raw array item.

```

5897     \cslet{#1\the\c@pgf@counta xo}\relax
5898     \cslet{#1\the\c@pgf@counta yo}\relax
5899     \cslet{#1\the\c@pgf@counta xp}\relax
5900     \cslet{#1\the\c@pgf@counta yp}\relax
5901     \advance\c@pgf@counta 1
5902   \saferepeat

```

Clean up the raw array length.

```

5903   \cslet{#1n}\relax

```

Store the length of the outer result array.

```

5904   \advance\c@pgf@countb 1
5905   \csdef{#2n}{\the\c@pgf@countb}%
5906 }

```

Item-exchange macro for quicksorting the raw projection data. (#1 is copied into #2.)

```
5907 \def\forest@ppiraw@let#1#2{%
5908   \csletcs{\forest@ppi@inputprefix#1xo}{\forest@ppi@inputprefix#2xo}%
5909   \csletcs{\forest@ppi@inputprefix#1yo}{\forest@ppi@inputprefix#2yo}%
5910   \csletcs{\forest@ppi@inputprefix#1xp}{\forest@ppi@inputprefix#2xp}%
5911   \csletcs{\forest@ppi@inputprefix#1yp}{\forest@ppi@inputprefix#2yp}%
5912 }
```

Item comparison macro for quicksorting the raw projection data.

```
5913 \def\forest@ppiraw@cmp#1#2{%
5914   \forest@sort@cmptwodimcs
5915   {\forest@ppi@inputprefix#1xp}{\forest@ppi@inputprefix#1yp}%
5916   {\forest@ppi@inputprefix#2xp}{\forest@ppi@inputprefix#2yp}%
5917 }
```

Append the point (#1,#2) to the (inner) array of points (prefix #3).

```
5918 \def\forest@append@point@to@inner@array#1#2#3{%
5919   \c@pgf@countc=\csname#3n\endcsname\relax
5920   \csedef{#3\the\c@pgf@countc x}{#1}%
5921   \csedef{#3\the\c@pgf@countc y}{#2}%
5922   \forest@distancetogrowline\pgfutil@tempdima{\pgfqpoint#1#2}%
5923   \csedef{#3\the\c@pgf@countc d}{\the\pgfutil@tempdima}%
5924   \advance\c@pgf@countc 1
5925   \csedef{#3n}{\the\c@pgf@countc}%
5926 }
```

## 8.2 Break path

The following macro computes from the given path (#1) a “broken” path (#3) that contains the same points of the plane, but has potentially more segments, so that, for every point from a given set of points on the grow line, a line through this point perpendicular to the grow line intersects the broken path only at its edge segments (i.e. not between them).

The macro works only for *simple* paths, i.e. paths built using only move-to and line-to operations. Furthermore, `\forest@processprojectioninfo` must be called before calling `\forest@breakpath`: we expect information with prefix #2. The macro updates the information compiled by `\forest@processprojectioninfo` with information about points added by path-breaking.

```
5927 \def\forest@breakpath#1#2#3{%
```

Store the current path in a macro and empty it, then process the stored path. The processing creates a new current path.

```
5928   \edef\forest@bp@prefix{#2}%
5929   \forest@save@pgfsyssoftpath@tokendefs
5930   \let\pgfsyssoftpath@movetotoken\forest@breakpath@processfirstpoint
5931   \let\pgfsyssoftpath@linetotoken\forest@breakpath@processfirstpoint
5932   %\pgfusepath{ }% empty the current path. ok?
5933   #1%
5934   \forest@restore@pgfsyssoftpath@tokendefs
5935   \pgfsyssoftpath@getcurrentpath#3%
5936 }
```

The original and the broken path start in the same way. (This code implicitly “repairs” a path that starts illegally, with a line-to operation.)

```
5937 \def\forest@breakpath@processfirstpoint#1#2{%
5938   \forest@breakpath@processmoveto{#1}{#2}%
5939   \let\pgfsyssoftpath@movetotoken\forest@breakpath@processmoveto
5940   \let\pgfsyssoftpath@linetotoken\forest@breakpath@processlineto
5941 }
```

When a move-to operation is encountered, it is simply copied to the broken path, starting a new subpath. Then we remember the last point, its projection’s index (the point dictionary is used here) and the actual projection point.

```

5942 \def\forest@breakpath@processmoveto#1#2{%
5943   \pgfsyssoftpath@moveto{#1}{#2}%
5944   \def\forest@previous@x{#1}%
5945   \def\forest@previous@y{#2}%
5946   \expandafter\let\expandafter\forest@previous@i
5947   \csname\forest@bp@prefix(#1,#2)\endcsname
5948   \expandafter\let\expandafter\forest@previous@px
5949   \csname\forest@bp@prefix\forest@previous@i x\endcsname
5950   \expandafter\let\expandafter\forest@previous@py
5951   \csname\forest@bp@prefix\forest@previous@i y\endcsname
5952 }

```

This is the heart of the path-breaking procedure.

```

5953 \def\forest@breakpath@processlineto#1#2{%

```

Usually, the broken path will continue with a line-to operation (to the current point (#1,#2)).

```

5954   \let\forest@breakpath@op\pgfsyssoftpath@lineto

```

Get the index of the current point's projection and the projection itself. (The point dictionary is used here.)

```

5955   \expandafter\let\expandafter\forest@i
5956   \csname\forest@bp@prefix(#1,#2)\endcsname
5957   \expandafter\let\expandafter\forest@px
5958   \csname\forest@bp@prefix\forest@i x\endcsname
5959   \expandafter\let\expandafter\forest@py
5960   \csname\forest@bp@prefix\forest@i y\endcsname

```

Test whether the projections of the previous and the current point are the same.

```

5961   \forest@equaltotolerance
5962   {\pgfpoint{\forest@previous@px}{\forest@previous@py}}%
5963   {\pgfpoint{\forest@px}{\forest@py}}%
5964   \ifforest@equaltotolerance

```

If so, we are dealing with a segment, perpendicular to the grow line. This segment must be removed, so we change the operation to move-to.

```

5965   \let\forest@breakpath@op\pgfsyssoftpath@moveto
5966   \else

```

Figure out the "direction" of the segment: in the order of the array of projections, or in the reversed order? Setup the loop step and the test condition.

```

5967   \forest@temp@count=\forest@previous@i\relax
5968   \ifnum\forest@previous@i<\forest@i\relax
5969     \def\forest@breakpath@step{1}%
5970     \def\forest@breakpath@test{\forest@temp@count<\forest@i\relax}%
5971   \else
5972     \def\forest@breakpath@step{-1}%
5973     \def\forest@breakpath@test{\forest@temp@count>\forest@i\relax}%
5974   \fi

```

Loop through all the projections between (in the (possibly reversed) array order) the projections of the previous and the current point (both exclusive).

```

5975   \safeloop
5976   \advance\forest@temp@count\forest@breakpath@step\relax
5977   \expandafter\ifnum\forest@breakpath@test

```

Intersect the current segment with the line through the current (in the loop!) projection perpendicular to the grow line. (There *will* be an intersection.)

```

5978   \pgfpointintersectionoflines
5979   {\pgfpoint
5980     {\csname\forest@bp@prefix\the\forest@temp@count x\endcsname}%
5981     {\csname\forest@bp@prefix\the\forest@temp@count y\endcsname}%
5982   }%
5983   {\pgfpointadd

```

```

5984     {\pgfqpoint
5985       {\csname\forest@bp@prefix\the\forest@temp@count x\endcsname}%
5986       {\csname\forest@bp@prefix\the\forest@temp@count y\endcsname}%
5987     }%
5988     {\pgfqpoint{\forest@xs}{\forest@ys}}%
5989   }%
5990   {\pgfqpoint{\forest@previous@x}{\forest@previous@y}}%
5991   {\pgfqpoint{#1}{#2}}%

Break the segment at the intersection.
5992   \pgfgetlastxy\forest@last@x\forest@last@y
5993   \pgfsyssoftpath@lineto\forest@last@x\forest@last@y

Append the breaking point to the inner array for the projection.
5994   \forest@append@point@to@inner@array
5995   \forest@last@x\forest@last@y
5996   {\forest@bp@prefix\the\forest@temp@count @}%

Cache the projection of the new segment edge.
5997   \csedef{\forest@bp@prefix(\the\pgf@x,\the\pgf@y)}{\the\forest@temp@count}%
5998   \saferepeat
5999   \fi

Add the current point.
6000   \forest@breakpath@op{#1}{#2}%

Setup new “previous” info: the segment edge, its projection’s index, and the projection.
6001   \def\forest@previous@x{#1}%
6002   \def\forest@previous@y{#2}%
6003   \let\forest@previous@i\forest@i
6004   \let\forest@previous@px\forest@px
6005   \let\forest@previous@py\forest@py
6006 }

```

### 8.3 Get tight edge of path

This is one of the central algorithms of the package. Given a simple path and a grow line, this method computes its (negative and positive) “tight edge”, which we (informally) define as follows.

Imagine an infinitely long light source parallel to the grow line, on the grow line’s negative/positive side.<sup>4</sup> Furthermore imagine that the path is opaque. Then the negative/positive tight edge of the path is the part of the path that is illuminated.

This macro takes three arguments: #1 is the path; #2 and #3 are macros which will receive the negative and the positive edge, respectively. The edges are returned in the softpath format. Grow line should be set before calling this macro.

Enclose the computation in a  $\TeX$  group. This is actually quite crucial: if there was no enclosure, the temporary data (the segment dictionary, to be precise) computed by the prior invocations of the macro could corrupt the computation in the current invocation.

```

6007 \def\forest@getnegativetightedgeofpath#1#2{%
6008   \forest@get@onetightedgeofpath#1\forest@sort@ascending#2}
6009 \def\forest@getpositivetightedgeofpath#1#2{%
6010   \forest@get@onetightedgeofpath#1\forest@sort@descending#2}
6011 \def\forest@get@onetightedgeofpath#1#2#3{%
6012   {%
6013     \forest@get@one@tightedgeofpath#1#2\forest@gep@edge
6014     \global\let\forest@gep@global@edge\forest@gep@edge
6015   }%
6016   \let#3\forest@gep@global@edge
6017 }
6018 \def\forest@get@one@tightedgeofpath#1#2#3{%

```

<sup>4</sup>For the definition of negative/positive side, see `forest@distancetogrowline` in §8.1

Project the path to the grow line and compile some useful information.

```
6019 \forest@projectpathtogrowline#1{forest@pp@}%
6020 \forest@sortprojections{forest@pp@}%
6021 \forest@processprojectioninfo{forest@pp@}{forest@pi@}%
```

Break the path.

```
6022 \forest@breakpath#1{forest@pi@}\forest@brokenpath
```

Compile some more useful information.

```
6023 \forest@sort@inner@arrays{forest@pi@}#2%
6024 \forest@pathtodict\forest@brokenpath{forest@pi@}%
```

The auxiliary data is set up: do the work!

```
6025 \forest@gettightedgeofpath@getedge\forest@edge
```

Where possible, merge line segments of the path into a single line segment. This is an important optimization, since the edges of the subtrees are computed recursively. Not simplifying the edge could result in a wild growth of the length of the edge (in the sense of the number of segments).

```
6026 \forest@simplifypath\forest@edge#3%
6027 }
```

Get both negative (stored in #2) and positive (stored in #3) edge of the path #1.

```
6028 \def\forest@getbothtightededgesofpath#1#2#3{%
6029   {%
6030     \forest@get@one@tightedgeofpath#1\forest@sort@ascending\forest@gep@firstedge
```

Reverse the order of items in the inner arrays.

```
6031   \c@pgf@counta=0
6032   \forest@loop
6033   \ifnum\c@pgf@counta<\forest@pi@n\relax
6034     \forest@ppi@deflet{forest@pi@\the\c@pgf@counta @}%
6035     \forest@reversearray\forest@ppi@let
6036     {0}%
6037     {\csname forest@pi@\the\c@pgf@counta @n\endcsname}%
6038     \advance\c@pgf@counta 1
6039   \forest@repeat
```

Calling `\forest@gettightedgeofpath@getedge` now will result in the positive edge.

```
6040   \forest@gettightedgeofpath@getedge\forest@edge
6041   \forest@simplifypath\forest@edge\forest@gep@secondedge
```

Smuggle the results out of the enclosing  $\TeX$  group.

```
6042   \global\let\forest@gep@global@firstedge\forest@gep@firstedge
6043   \global\let\forest@gep@global@secondedge\forest@gep@secondedge
6044   }%
6045   \let#2\forest@gep@global@firstedge
6046   \let#3\forest@gep@global@secondedge
6047 }
```

Sort the inner arrays of original points wrt the distance to the grow line. #2 = `\forest@sort@ascending/\forest@sor`

```
6048 \def\forest@sort@inner@arrays#1#2{%
6049   \c@pgf@counta=0
6050   \safeloop
6051   \ifnum\c@pgf@counta<\csname#1n\endcsname
6052     \c@pgf@countb=\csname#1\the\c@pgf@counta @n\endcsname\relax
6053     \ifnum\c@pgf@countb>1
6054       \advance\c@pgf@countb -1
6055       \forest@ppi@deflet{#1\the\c@pgf@counta @}%
6056       \forest@ppi@defcmp{#1\the\c@pgf@counta @}%
6057       \forest@sort\forest@ppi@cmp\forest@ppi@let#2{0}{\the\c@pgf@countb}%
6058     \fi
6059     \advance\c@pgf@counta 1
6060   \saferepeat
6061 }
```

A macro that will define the item exchange macro for quicksorting the inner arrays of original points.  
It takes one argument: the prefix of the inner array.

```
6062 \def\forest@ppi@deflet#1{%
6063   \edef\forest@ppi@let##1##2{%
6064     \noexpand\csletcs{#1##1x}{#1##2x}%
6065     \noexpand\csletcs{#1##1y}{#1##2y}%
6066     \noexpand\csletcs{#1##1d}{#1##2d}%
6067   }%
6068 }
```

A macro that will define the item-compare macro for quicksorting the embedded arrays of original points.  
It takes one argument: the prefix of the inner array.

```
6069 \def\forest@ppi@defcmp#1{%
6070   \edef\forest@ppi@cmp##1##2{%
6071     \noexpand\forest@sort@cmpdimcs{#1##1d}{#1##2d}%
6072   }%
6073 }
```

Put path segments into a “segment dictionary”: for each segment of the path from  $(x_1, y_1)$  to  $(x_2, y_2)$   
let  $\text{\forest@}(x_1, y_1) \text{--}(x_2, y_2)$  be  $\text{\forest@inpath}$  (which can be anything but  $\text{\relax}$ ).

```
6074 \let\forest@inpath\advance
```

This macro is just a wrapper to process the path.

```
6075 \def\forest@pathtodict#1#2{%
6076   \edef\forest@pathtodict@prefix{#2}%
6077   \forest@save@pgfsyssoftpath@tokendefs
6078   \let\pgfsyssoftpath@movetotoken\forest@pathtodict@movetoop
6079   \let\pgfsyssoftpath@linetotoken\forest@pathtodict@linetoop
6080   \def\forest@pathtodict@subpathstart{}%
6081   #1%
6082   \forest@restore@pgfsyssoftpath@tokendefs
6083 }
```

When a move-to operation is encountered:

```
6084 \def\forest@pathtodict@movetoop#1#2{%
```

If a subpath had just started, it was a degenerate one (a point). No need to store that (i.e. no code would use this information). So, just remember that a new subpath has started.

```
6085   \def\forest@pathtodict@subpathstart{(#1,#2)-}%
6086 }
```

When a line-to operation is encountered:

```
6087 \def\forest@pathtodict@linetoop#1#2{%
```

If the subpath has just started, its start is also the start of the current segment.

```
6088 \if\relax\forest@pathtodict@subpathstart\relax\else
6089   \let\forest@pathtodict@from\forest@pathtodict@subpathstart
6090   \fi
```

Mark the segment as existing.

```
6091   \expandafter\let\csname\forest@pathtodict@prefix\forest@pathtodict@from-(#1,#2)\endcsname\forest@inpath
```

Set the start of the next segment to the current point, and mark that we are in the middle of a subpath.

```
6092   \def\forest@pathtodict@from{(#1,#2)-}%
6093   \def\forest@pathtodict@subpathstart{}%
6094 }
```

In this macro, the edge is actually computed.

```
6095 \def\forest@gettightedgeofpath@getedge#1{% cs to store the edge into
```

Clear the path and the last projection.

```
6096   \pgfsyssoftpath@setcurrentpath\pgfutil@empty
6097   \let\forest@last@x\relax
6098   \let\forest@last@y\relax
```

Loop through the (ordered) array of projections. (Since we will be dealing with the current and the next projection in each iteration of the loop, we loop the counter from the first to the second-to-last projection.)

```

6099 \c@pgf@counta=0
6100 \forest@temp@count=\forest@pi@n\relax
6101 \advance\forest@temp@count -1
6102 \edef\forest@nminusone{\the\forest@temp@count}%
6103 \safeloop
6104 \ifnum\c@pgf@counta<\forest@nminusone\relax
6105   \forest@gettightedgeofpath@getedge@loopa
6106 \saferepeat

```

A special case: the edge ends with a degenerate subpath (a point).

```

6107 \ifnum\forest@nminusone<\forest@n\relax\else
6108   \ifnum\csname forest@pi@\forest@nminusone @n\endcsname>0
6109     \forest@gettightedgeofpath@maybemoveto{\forest@nminusone}{0}%
6110   \fi
6111 \fi
6112 \pgfsyssoftpath@getcurrentpath#1%
6113 \pgfsyssoftpath@setcurrentpath\pgfutil@empty
6114 }

```

The body of a loop containing an embedded loop must be put in a separate macro because it contains the `\if...` of the embedded `\forest@loop...` without the matching `\fi`: `\fi` is “hiding” in the embedded `\forest@loop`, which has not been expanded yet.

```

6115 \def\forest@gettightedgeofpath@getedge@loopa{%
6116   \ifnum\csname forest@pi@\the\c@pgf@counta @n\endcsname>0

```

Degenerate case: a subpath of the edge is a point.

```

6117   \forest@gettightedgeofpath@maybemoveto{\the\c@pgf@counta}{0}%

```

Loop through points projecting to the current projection. The preparations above guarantee that the points are ordered (either in the ascending or the descending order) with respect to their distance to the grow line.

```

6118   \c@pgf@countb=0
6119   \safeloop
6120   \ifnum\c@pgf@countb<\csname forest@pi@\the\c@pgf@counta @n\endcsname\relax
6121     \forest@gettightedgeofpath@getedge@loopb
6122   \saferepeat
6123   \fi
6124   \advance\c@pgf@counta 1
6125 }

```

Loop through points projecting to the next projection. Again, the points are ordered.

```

6126 \def\forest@gettightedgeofpath@getedge@loopb{%
6127   \c@pgf@countc=0
6128   \advance\c@pgf@counta 1
6129   \edef\forest@aplusone{\the\c@pgf@counta}%
6130   \advance\c@pgf@counta -1
6131   \safeloop
6132   \ifnum\c@pgf@countc<\csname forest@pi@\forest@aplusone @n\endcsname\relax

```

Test whether [the current point]–[the next point] or [the next point]–[the current point] is a segment in the (broken) path. The first segment found is the one with the minimal/maximal distance (depending on the sort order of arrays of points projecting to the same projection) to the grow line.

Note that for this to work in all cases, the original path should have been broken on its self-intersections. However, a careful reader will probably remember that `\forest@breakpath` does *not* break the path at its self-intersections. This is omitted for performance reasons. Given the intended use of the algorithm (calculating edges of subtrees), self-intersecting paths cannot arise anyway, if only the node boundaries are non-self-intersecting. So, a warning: if you develop a new shape and write a macro computing its boundary, make sure that the computed boundary path is non-self-intersecting!



```

6133     \forest@tempfalse
6134     \expandafter\ifx\csname forest@pi@(%
6135         \csname forest@pi@\the\c@pgf@counta @\the\c@pgf@countb x\endcsname,%
6136         \csname forest@pi@\the\c@pgf@counta @\the\c@pgf@countb y\endcsname)--(%
6137         \csname forest@pi@\forest@aplusone @\the\c@pgf@countc x\endcsname,%
6138         \csname forest@pi@\forest@aplusone @\the\c@pgf@countc y\endcsname)%
6139     \endcsname\forest@inpath
6140     \forest@temptrue
6141     \else
6142     \expandafter\ifx\csname forest@pi@(%
6143         \csname forest@pi@\forest@aplusone @\the\c@pgf@countc x\endcsname,%
6144         \csname forest@pi@\forest@aplusone @\the\c@pgf@countc y\endcsname)--(%
6145         \csname forest@pi@\the\c@pgf@counta @\the\c@pgf@countb x\endcsname,%
6146         \csname forest@pi@\the\c@pgf@counta @\the\c@pgf@countb y\endcsname)%
6147     \endcsname\forest@inpath
6148     \forest@temptrue
6149     \fi
6150     \fi
6151     \ifforest@temp

```

We have found the segment with the minimal/maximal distance to the grow line. So let's add it to the edge path.

First, deal with the start point of the edge: check if the current point is the last point. If that is the case (this happens if the current point was the end point of the last segment added to the edge), nothing needs to be done; otherwise (this happens if the current point will start a new subpath of the edge), move to the current point, and update the last-point macros.

```

6152     \forest@gettightedgeofpath@maybemoveto{\the\c@pgf@counta}\the\c@pgf@countb}%

```

Second, create a line to the end point.

```

6153     \edef\forest@last@x{%
6154         \csname forest@pi@\forest@aplusone @\the\c@pgf@countc x\endcsname}%
6155     \edef\forest@last@y{%
6156         \csname forest@pi@\forest@aplusone @\the\c@pgf@countc y\endcsname}%
6157     \pgfsyssoftpath@lineto\forest@last@x\forest@last@y

```

Finally, "break" out of the innermost two loops.

```

6158         \c@pgf@countc=\csname forest@pi@\forest@aplusone @n\endcsname
6159         \c@pgf@countb=\csname forest@pi@\the\c@pgf@counta @n\endcsname
6160     \fi
6161     \advance\c@pgf@countc 1
6162     \saferepeat
6163     \advance\c@pgf@countb 1
6164 }

```

`\forest@#1@` is an (ordered) array of points projecting to projection with index #1. Check if #2th point of that array equals the last point added to the edge: if not, add it.

```

6165 \def\forest@gettightedgeofpath@maybemoveto#1#2{%
6166     \forest@temptrue
6167     \ifx\forest@last@x\relax\else
6168         \ifdim\forest@last@x=\csname forest@pi@#1@#2x\endcsname\relax
6169         \ifdim\forest@last@y=\csname forest@pi@#1@#2y\endcsname\relax
6170             \forest@tempfalse
6171         \fi
6172     \fi
6173     \fi
6174     \ifforest@temp
6175         \edef\forest@last@x{\csname forest@pi@#1@#2x\endcsname}%
6176         \edef\forest@last@y{\csname forest@pi@#1@#2y\endcsname}%
6177         \pgfsyssoftpath@moveto\forest@last@x\forest@last@y
6178     \fi
6179 }

```

Simplify the resulting path by “unbreaking” segments where possible. (The macro itself is just a wrapper for path processing macros below.)

```

6180 \def\forest@simplifypath#1#2{%
6181   \pgfsyssoftpath@setcurrentpath\pgfutil@empty
6182   \forest@save@pgfsyssoftpath@tokendefs
6183   \let\pgfsyssoftpath@movetotoken\forest@simplifypath@moveto
6184   \let\pgfsyssoftpath@linetotoken\forest@simplifypath@lineto
6185   \let\forest@last@x\relax
6186   \let\forest@last@y\relax
6187   \let\forest@last@atan\relax
6188   #1%
6189   \ifx\forest@last@x\relax\else
6190     \ifx\forest@last@atan\relax\else
6191       \pgfsyssoftpath@lineto\forest@last@x\forest@last@y
6192     \fi
6193   \fi
6194   \forest@restore@pgfsyssoftpath@tokendefs
6195   \pgfsyssoftpath@getcurrentpath#2%
6196   \pgfsyssoftpath@setcurrentpath\pgfutil@empty
6197 }

```

When a move-to is encountered, we flush whatever segment we were building, make the move, remember the last position, and set the slope to unknown.

```

6198 \def\forest@simplifypath@moveto#1#2{%
6199   \ifx\forest@last@x\relax\else
6200     \pgfsyssoftpath@lineto\forest@last@x\forest@last@y
6201   \fi
6202   \pgfsyssoftpath@moveto{#1}{#2}%
6203   \def\forest@last@x{#1}%
6204   \def\forest@last@y{#2}%
6205   \let\forest@last@atan\relax
6206 }

```

How much may the segment slopes differ that we can still merge them? (Ignore pt, these are degrees.) Also, how good is this number?

```

6207 \def\forest@getedgeofpath@precision{1pt}

```

When a line-to is encountered...

```

6208 \def\forest@simplifypath@lineto#1#2{%
6209   \ifx\forest@last@x\relax

```

If we’re not in the middle of a merger, we need to nothing but start it.

```

6210     \def\forest@last@x{#1}%
6211     \def\forest@last@y{#2}%
6212     \let\forest@last@atan\relax
6213   \else

```

Otherwise, we calculate the slope of the current segment (i.e. the segment between the last and the current point), ...

```

6214     \pgfpointdiff{\pgfpoint{#1}{#2}}{\pgfpoint{\forest@last@x}{\forest@last@y}}%
6215     \ifdim\pgf@x<\pgfintersectiontolerance
6216       \ifdim-\pgf@x<\pgfintersectiontolerance
6217         \pgf@x=0pt
6218       \fi
6219     \fi
6220     \csname pgfmatahan2\endcsname{\pgf@x}{\pgf@y}%
6221     \let\forest@current@atan\pgfmathresult
6222     \ifx\forest@last@atan\relax

```

If this is the first segment in the current merger, simply remember the slope and the last point.

```

6223     \def\forest@last@x{#1}%
6224     \def\forest@last@y{#2}%

```

```

6225     \let\forest@last@atan\forest@current@atan
6226     \else
    Otherwise, compare the first and the current slope.
6227     \pgfutil@tempdima=\forest@current@atan pt
6228     \advance\pgfutil@tempdima -\forest@last@atan pt
6229     \ifdim\pgfutil@tempdima<0pt\relax
6230     \multiply\pgfutil@tempdima -1
6231     \fi
6232     \ifdim\pgfutil@tempdima<\forest@getedgeofpath@precision\relax
6233     \else

```

If the slopes differ too much, flush the path up to the previous segment, and set up a new first slope.

```

6234     \pgfsyssoftpath@lineto\forest@last@x\forest@last@y
6235     \let\forest@last@atan\forest@current@atan
6236     \fi

```

In any event, update the last point.

```

6237     \def\forest@last@x{#1}%
6238     \def\forest@last@y{#2}%
6239     \fi
6240     \fi
6241 }

```

## 8.4 Get rectangle/band edge

```

6242 \def\forest@getnegativerectangleedgeofpath#1#2{%
6243   \forest@getnegativerectangleorbandededgeofpath{#1}{#2}{\the\pgf@xb}}
6244 \def\forest@getpositiverectangleedgeofpath#1#2{%
6245   \forest@getpositiverectangleorbandededgeofpath{#1}{#2}{\the\pgf@xb}}
6246 \def\forest@getbothrectangleedgesofpath#1#2#3{%
6247   \forest@getbothrectangleorbandededgesofpath{#1}{#2}{#3}{\the\pgf@xb}}
6248 \def\forest@bandlength{5000pt} % something large (ca. 180cm), but still manageable for TeX without producing
6249 \def\forest@getnegativebandededgeofpath#1#2{%
6250   \forest@getnegativerectangleorbandededgeofpath{#1}{#2}{\forest@bandlength}}
6251 \def\forest@getpositivebandededgeofpath#1#2{%
6252   \forest@getpositiverectangleorbandededgeofpath{#1}{#2}{\forest@bandlength}}
6253 \def\forest@getbothbandededgesofpath#1#2#3{%
6254   \forest@getbothrectangleorbandededgesofpath{#1}{#2}{#3}{\forest@bandlength}}
6255 \def\forest@getnegativerectangleorbandededgeofpath#1#2#3{%
6256   \forest@path@getboundingrectangle@ls#1{\forest@grow}%
6257   \edef\forest@gre@path{%
6258     \noexpand\pgfsyssoftpath@movetotoken{\the\pgf@xa}{\the\pgf@ya}%
6259     \noexpand\pgfsyssoftpath@linetotoken{#3}{\the\pgf@ya}%
6260   }%
6261   {%
6262     \pgftransformreset
6263     \pgftransformrotate{\forest@grow}%
6264     \forest@pgfpathtransformed\forest@gre@path
6265   }%
6266   \pgfsyssoftpath@getcurrentpath#2%
6267 }
6268 \def\forest@getpositiverectangleorbandededgeofpath#1#2#3{%
6269   \forest@path@getboundingrectangle@ls#1{\forest@grow}%
6270   \edef\forest@gre@path{%
6271     \noexpand\pgfsyssoftpath@movetotoken{\the\pgf@xa}{\the\pgf@yb}%
6272     \noexpand\pgfsyssoftpath@linetotoken{#3}{\the\pgf@yb}%
6273   }%
6274   {%
6275     \pgftransformreset
6276     \pgftransformrotate{\forest@grow}%
6277     \forest@pgfpathtransformed\forest@gre@path

```

```

6278 }%
6279 \pgfsyssoftpath@getcurrentpath#2%
6280 }
6281 \def\forest@getbothrectangleorbandedgesofpath#1#2#3#4{%
6282 \forest@path@getboundingrectangle@ls#1{\forest@grow}%
6283 \edef\forest@gre@negpath{%
6284 \noexpand\pgfsyssoftpath@mometoken{\the\pgf@xa}{\the\pgf@ya}%
6285 \noexpand\pgfsyssoftpath@linetoken{#4}{\the\pgf@ya}%
6286 }%
6287 \edef\forest@gre@pospath{%
6288 \noexpand\pgfsyssoftpath@mometoken{\the\pgf@xa}{\the\pgf@yb}%
6289 \noexpand\pgfsyssoftpath@linetoken{#4}{\the\pgf@yb}%
6290 }%
6291 {%
6292 \pgftransformreset
6293 \pgftransformrotate{\forest@grow}%
6294 \forest@pgfpathtransformed\forest@gre@negpath
6295 }%
6296 \pgfsyssoftpath@getcurrentpath#2%
6297 {%
6298 \pgftransformreset
6299 \pgftransformrotate{\forest@grow}%
6300 \forest@pgfpathtransformed\forest@gre@pospath
6301 }%
6302 \pgfsyssoftpath@getcurrentpath#3%
6303 }

```

## 8.5 Distance between paths

Another crucial part of the package.

```

6304 \def\forest@distance@between@edge@paths#1#2#3{%
6305 % #1, #2 = (edge) paths
6306 %
6307 % project paths
6308 \forest@projectpathtogroupline#1{\forest@p1}%
6309 \forest@projectpathtogroupline#2{\forest@p2}%
6310 % merge projections (the lists are sorted already, because edge
6311 % paths are |sorted|)
6312 \forest@dbep@mergeprojections
6313 {forest@p1}{forest@p2}%
6314 {forest@P1}{forest@P2}%
6315 % process projections
6316 \forest@processprojectioninfo{forest@P1}{forest@PI1}%
6317 \forest@processprojectioninfo{forest@P2}{forest@PI2}%
6318 % break paths
6319 \forest@breakpath#1{\forest@PI1}\forest@broken@one
6320 \forest@breakpath#2{\forest@PI2}\forest@broken@two
6321 % sort inner arrays ---optimize: it's enough to find max and min
6322 \forest@sort@inner@arrays{forest@PI1}\forest@sort@descending
6323 \forest@sort@inner@arrays{forest@PI2}\forest@sort@ascending
6324 % compute the distance
6325 \let\forest@distance\relax
6326 \c@pgf@countc=0
6327 \forest@loop
6328 \ifnum\c@pgf@countc<\csname forest@PI1\endcsname\relax
6329 \ifnum\csname forest@PI1\the\c@pgf@countc @\endcsname=0 \else
6330 \ifnum\csname forest@PI2\the\c@pgf@countc @\endcsname=0 \else
6331 \pgfutil@tempdima=\csname forest@PI2\the\c@pgf@countc @0d\endcsname\relax
6332 \advance\pgfutil@tempdima-\csname forest@PI1\the\c@pgf@countc @0d\endcsname\relax
6333 \ifx\forest@distance\relax
6334 \edef\forest@distance{\the\pgfutil@tempdima}%

```

```

6335     \else
6336     \ifdim\pgfutil@tempdima<\forest@distance\relax
6337     \edef\forest@distance{\the\pgfutil@tempdima}%
6338     \fi
6339     \fi
6340     \fi
6341     \fi
6342     \advance\c@pgf@countc 1
6343     \forest@repeat
6344     \let#3\forest@distance
6345 }
6346 % merge projections: we need two projection arrays, both containing
6347 % projection points from both paths, but each with the original
6348 % points from only one path
6349 \def\forest@dbep@mergeprojections#1#2#3#4{%
6350 % TODO: optimize: v bistvu ni treba sortirat, ker je edge path e sortiran
6351 \forest@sortprojections{#1}%
6352 \forest@sortprojections{#2}%
6353 \c@pgf@counta=0
6354 \c@pgf@countb=0
6355 \c@pgf@countc=0
6356 \edef\forest@input@prefix@one{#1}%
6357 \edef\forest@input@prefix@two{#2}%
6358 \edef\forest@output@prefix@one{#3}%
6359 \edef\forest@output@prefix@two{#4}%
6360 \forest@dbep@mp@iterate
6361 \csedef{#3n}{\the\c@pgf@countc}%
6362 \csedef{#4n}{\the\c@pgf@countc}%
6363 }
6364 \def\forest@dbep@mp@iterate{%
6365 \let\forest@dbep@mp@next\forest@dbep@mp@iterate
6366 \ifnum\c@pgf@counta<\csname\forest@input@prefix@one n\endcsname\relax
6367 \ifnum\c@pgf@countb<\csname\forest@input@prefix@two n\endcsname\relax
6368 \let\forest@dbep@mp@next\forest@dbep@mp@do
6369 \else
6370 \let\forest@dbep@mp@next\forest@dbep@mp@iteratefirst
6371 \fi
6372 \else
6373 \ifnum\c@pgf@countb<\csname\forest@input@prefix@two n\endcsname\relax
6374 \let\forest@dbep@mp@next\forest@dbep@mp@iteratesecond
6375 \else
6376 \let\forest@dbep@mp@next\relax
6377 \fi
6378 \fi
6379 \forest@dbep@mp@next
6380 }
6381 \def\forest@dbep@mp@do{%
6382 \forest@sort@cmptwodimcs%
6383 {\forest@input@prefix@one\the\c@pgf@counta xp}%
6384 {\forest@input@prefix@one\the\c@pgf@counta yp}%
6385 {\forest@input@prefix@two\the\c@pgf@countb xp}%
6386 {\forest@input@prefix@two\the\c@pgf@countb yp}%
6387 \if\forest@sort@cmp@result=%
6388 \forest@dbep@mp@store@p\forest@input@prefix@one\c@pgf@counta
6389 \forest@dbep@mp@store@o\forest@input@prefix@one
6390 \c@pgf@counta\forest@output@prefix@one
6391 \forest@dbep@mp@store@o\forest@input@prefix@two
6392 \c@pgf@countb\forest@output@prefix@two
6393 \advance\c@pgf@counta 1
6394 \advance\c@pgf@countb 1
6395 \else

```

```

6396 \if\forest@sort@cmp@result>%
6397 \forest@dbep@mp@@store@p\forest@input@prefix@two@c@pgf@countb
6398 \forest@dbep@mp@@store@o\forest@input@prefix@two
6399 \c@pgf@countb\forest@output@prefix@two
6400 \advance\c@pgf@countb 1
6401 \else%<
6402 \forest@dbep@mp@@store@p\forest@input@prefix@one@c@pgf@counta
6403 \forest@dbep@mp@@store@o\forest@input@prefix@one
6404 \c@pgf@counta\forest@output@prefix@one
6405 \advance\c@pgf@counta 1
6406 \fi
6407 \fi
6408 \advance\c@pgf@countc 1
6409 \forest@dbep@mp@iterate
6410 }
6411 \def\forest@dbep@mp@@store@p#1#2{%
6412 \csletcs
6413 {\forest@output@prefix@one\the\c@pgf@countc xp}%
6414 {#1\the#2xp}%
6415 \csletcs
6416 {\forest@output@prefix@one\the\c@pgf@countc yp}%
6417 {#1\the#2yp}%
6418 \csletcs
6419 {\forest@output@prefix@two\the\c@pgf@countc xp}%
6420 {#1\the#2xp}%
6421 \csletcs
6422 {\forest@output@prefix@two\the\c@pgf@countc yp}%
6423 {#1\the#2yp}%
6424 }
6425 \def\forest@dbep@mp@@store@o#1#2#3{%
6426 \csletcs{#3\the\c@pgf@countc xo}{#1\the#2xo}%
6427 \csletcs{#3\the\c@pgf@countc yo}{#1\the#2yo}%
6428 }
6429 \def\forest@dbep@mp@iteratefirst{%
6430 \forest@dbep@mp@iterateone\forest@input@prefix@one\c@pgf@counta\forest@output@prefix@one
6431 }
6432 \def\forest@dbep@mp@iteratesecond{%
6433 \forest@dbep@mp@iterateone\forest@input@prefix@two\c@pgf@countb\forest@output@prefix@two
6434 }
6435 \def\forest@dbep@mp@iterateone#1#2#3{%
6436 \forest@loop
6437 \ifnum#2<\csname#1n\endcsname\relax
6438 \forest@dbep@mp@@store@p#1#2%
6439 \forest@dbep@mp@@store@o#1#2#3%
6440 \advance\c@pgf@countc 1
6441 \advance#21
6442 \forest@repeat
6443 }

```

## 8.6 Utilities

Equality test: points are considered equal if they differ less than `\pgfintersectiontolerance` in each coordinate.

```

6444 \newif\ifforest@equaltotolerance
6445 \def\forest@equaltotolerance#1#2{%
6446 \pgfpointdiff{#1}{#2}%
6447 \ifdim\pgf@x<0pt \multiply\pgf@x -1 \fi
6448 \ifdim\pgf@y<0pt \multiply\pgf@y -1 \fi
6449 \global\forest@equaltotolerancefalse
6450 \ifdim\pgf@x<\pgfintersectiontolerance\relax

```

```

6451 \ifdim\pgf@y<\pgfintersectiontolerance\relax
6452 \global\forest@equaltotolerancetrue
6453 \fi
6454 \fi
6455 }}

Save/restore pgfs \pgfsyssoftpath@...token definitions.
6456 \def\forest@save@pgfsyssoftpath@tokendefs{%
6457 \let\forest@origmovetotoken\pgfsyssoftpath@movetotoken
6458 \let\forest@origlinetotoken\pgfsyssoftpath@linetotoken
6459 \let\forest@origcurvetosupportatoken\pgfsyssoftpath@curvetosupportatoken
6460 \let\forest@origcurvetosupportbtoken\pgfsyssoftpath@curvetosupportbtoken
6461 \let\forest@origcurvetotoken\pgfsyssoftpath@curvetotoken
6462 \let\forest@origrectcornertoken\pgfsyssoftpath@rectcornertoken
6463 \let\forest@origrectsizenetoken\pgfsyssoftpath@rectsizenetoken
6464 \let\forest@origclosepathtoken\pgfsyssoftpath@closepathtoken
6465 \let\pgfsyssoftpath@movetotoken\forest@badtoken
6466 \let\pgfsyssoftpath@linetotoken\forest@badtoken
6467 \let\pgfsyssoftpath@curvetosupportatoken\forest@badtoken
6468 \let\pgfsyssoftpath@curvetosupportbtoken\forest@badtoken
6469 \let\pgfsyssoftpath@curvetotoken\forest@badtoken
6470 \let\pgfsyssoftpath@rectcornertoken\forest@badtoken
6471 \let\pgfsyssoftpath@rectsizenetoken\forest@badtoken
6472 \let\pgfsyssoftpath@closepathtoken\forest@badtoken
6473 }
6474 \def\forest@badtoken{%
6475 \PackageError{forest}{This token should not be in this path}{}%
6476 }
6477 \def\forest@restore@pgfsyssoftpath@tokendefs{%
6478 \let\pgfsyssoftpath@movetotoken\forest@origmovetotoken
6479 \let\pgfsyssoftpath@linetotoken\forest@origlinetotoken
6480 \let\pgfsyssoftpath@curvetosupportatoken\forest@origcurvetosupportatoken
6481 \let\pgfsyssoftpath@curvetosupportbtoken\forest@origcurvetosupportbtoken
6482 \let\pgfsyssoftpath@curvetotoken\forest@origcurvetotoken
6483 \let\pgfsyssoftpath@rectcornertoken\forest@origrectcornertoken
6484 \let\pgfsyssoftpath@rectsizenetoken\forest@origrectsizenetoken
6485 \let\pgfsyssoftpath@closepathtoken\forest@origclosepathtoken
6486 }

Extend path #1 with path #2 translated by point #3.
6487 \def\forest@extendpath#1#2#3{%
6488 \pgf@process{#3}%
6489 \pgfsyssoftpath@setcurrentpath#1%
6490 \forest@save@pgfsyssoftpath@tokendefs
6491 \let\pgfsyssoftpath@movetotoken\forest@extendpath@moveto
6492 \let\pgfsyssoftpath@linetotoken\forest@extendpath@lineto
6493 #2%
6494 \forest@restore@pgfsyssoftpath@tokendefs
6495 \pgfsyssoftpath@getcurrentpath#1%
6496 }
6497 \def\forest@extendpath@moveto#1#2{%
6498 \forest@extendpath@do{#1}{#2}\pgfsyssoftpath@moveto
6499 }
6500 \def\forest@extendpath@lineto#1#2{%
6501 \forest@extendpath@do{#1}{#2}\pgfsyssoftpath@lineto
6502 }
6503 \def\forest@extendpath@do#1#2#3{%
6504 {%
6505 \advance\pgf@x #1
6506 \advance\pgf@y #2
6507 #3{\the\pgf@x}{\the\pgf@y}%
6508 }%

```

```

6509 }
    Get bounding rectangle of the path. #1 = the path, #2 = grow. Returns (\pgf@xa=min x/l,
    \pgf@ya=max y/s, \pgf@xb=min x/l, \pgf@yb=max y/s). (If path #1 is empty, the result is undefined.)
6510 \def\forest@path@getboundingrectangle@ls#1#2{%
6511   {%
6512     \pgftransformreset
6513     \pgftransformrotate{-(#2)}%
6514     \forest@pgfpathtransformed#1%
6515   }%
6516   \pgfsyssoftpath@getcurrentpath\forest@gbr@rotatedpath
6517   \forest@path@getboundingrectangle@xy\forest@gbr@rotatedpath
6518 }
6519 \def\forest@path@getboundingrectangle@xy#1{%
6520   \forest@save@pgfsyssoftpath@tokendefs
6521   \let\pgfsyssoftpath@movetotoken\forest@gbr@firstpoint
6522   \let\pgfsyssoftpath@linetotoken\forest@gbr@firstpoint
6523   #1%
6524   \forest@restore@pgfsyssoftpath@tokendefs
6525 }
6526 \def\forest@gbr@firstpoint#1#2{%
6527   \pgf@xa=#1 \pgf@xb=#1 \pgf@ya=#2 \pgf@yb=#2
6528   \let\pgfsyssoftpath@movetotoken\forest@gbr@point
6529   \let\pgfsyssoftpath@linetotoken\forest@gbr@point
6530 }
6531 \def\forest@gbr@point#1#2{%
6532   \ifdim#1<\pgf@xa\relax\pgf@xa=#1 \fi
6533   \ifdim#1>\pgf@xb\relax\pgf@xb=#1 \fi
6534   \ifdim#2<\pgf@ya\relax\pgf@ya=#2 \fi
6535   \ifdim#2>\pgf@yb\relax\pgf@yb=#2 \fi
6536 }

```

## 9 The outer UI

### 9.1 Externalization

```

6537 \pgfkeys{/forest/external/.cd,
6538   copy command/.initial={cp "\source" "\target"},
6539   optimize/.is if=forest@external@optimize@,
6540   context/.initial={%
6541     \forestOve{\csname forest@id@of@standard node\endcsname}{environment@formula}},
6542   depends on macro/.style={context/.append/.expanded={%
6543     \expandafter\detokenize\expandafter{#1}}},
6544 }
6545 \def\forest@external@copy#1#2{%
6546   \pgfkeysgetvalue{/forest/external/copy command}\forest@copy@command
6547   \ifx\forest@copy@command\pgfkeysnovalue\else
6548     \IfFileExists{#1}{%
6549       {%
6550         \def\source{#1}%
6551         \def\target{#2}%
6552         \immediate\write18{\forest@copy@command}%
6553       }%
6554     }-{}%
6555   \fi
6556 }
6557 \newif\ifforest@external@optimize@
6558 \forest@external@optimize@true
6559 \ifforest@install@keys@to@tikz@path@
6560 \tikzset{

```



```

6561 fit to/.style={
6562   /forest/for nodewalk=%
6563   {TeX={\def\forest@fitto{}},#1}%
6564   {TeX={\eappto\forest@fitto{(\forestove{name})}}},
6565   fit/.expanded={\forest@fitto}
6566 },
6567 }
6568 \fi
6569 \ifforest@external@
6570 \ifdefined\tikzexternal@tikz@replacement\else
6571   \usetikzlibrary{external}%
6572 \fi
6573 \pgfkeys{%
6574   /tikz/external/failed ref warnings for={},
6575   /pgf/images/aux in dpth=false,
6576 }%
6577 \tikzifexternalizing{}{%
6578   \forest@external@copy{\jobname.aux}{\jobname.aux.copy}%
6579 }%
6580 \AtBeginDocument{%
6581   \tikzifexternalizing{%
6582     \IfFileExists{\tikzexternalrealjob.aux.copy}{%
6583       \makeatletter
6584       \input \tikzexternalrealjob.aux.copy
6585       \makeatother
6586     }{}%
6587   }{%
6588     \newwrite\forest@auxout
6589     \immediate\openout\forest@auxout=\tikzexternalrealjob.for.tmp
6590   }%
6591   \IfFileExists{\tikzexternalrealjob.for}{%
6592     {%
6593       \makehashother\makeatletter
6594       \input \tikzexternalrealjob.for
6595     }%
6596   }{}%
6597 }%
6598 \AtEndDocument{%
6599   \tikzifexternalizing{}{%
6600     \immediate\closeout\forest@auxout
6601     \forest@external@copy{\jobname.for.tmp}{\jobname.for}%
6602   }%
6603 }%
6604 \fi

```

## 9.2 The forest environment

There are three ways to invoke FOREST: the environment and the starless and the starred version of the macro. The latter creates no group.

Most of the code in this section deals with externalization.

```

6605 \NewDocumentEnvironment{forest}{D()}{}{%
6606   \forest@config{#1}%
6607   \Collect@Body
6608   \forest@env
6609 }{}
6610 \NewDocumentCommand{\Forest}{s D()}{m}{%
6611   \forest@config{#2}%
6612   \IfBooleanTF{#1}{\let\forest@next\forest@env}{\let\forest@next\forest@group@env}%
6613   \forest@next{#3}%
6614 }
6615 \def\forest@config#1{%

```

```

6616 \def\forest@stages{stages}%
6617 \forestset{@config/.cd,#1}%
6618 }
6619 \forestset{@config/.cd,
6620   stages/.store in=\forest@stages,
6621   .unknown/.code={\PackageError{forest}{Unknown config option for forest environment/command.}{In Forest v2.0}}
6622 }
6623 \def\forest@group@env#1{{\forest@env{#1}}}
6624 \newif\ifforest@externalize@tree@
6625 \newif\ifforest@was@tikzexternalwasenable
6626 \newcommand\forest@env[1]{%
6627   \let\forest@external@next\forest@begin
6628   \forest@was@tikzexternalwasenablefalse
6629   \ifdefined\tikzexternal@tikz@replacement
6630     \ifx\tikz\tikzexternal@tikz@replacement
6631       \forest@was@tikzexternalwasenabletrue
6632       \tikzexternaldisable
6633     \fi
6634   \fi
6635   \forest@externalize@tree@false
6636   \ifforest@external@
6637     \ifforest@was@tikzexternalwasenable
6638     \forest@env@
6639   \fi
6640   \fi
6641   \forest@standardnode@calibrate
6642   \forest@external@next{#1}%
6643 }
6644 \def\forest@env@{%
6645   \iftikzexternalexportnext
6646     \tikzifexternalizing{%
6647       \let\forest@external@next\forest@begin@externalizing
6648     }{%
6649       \let\forest@external@next\forest@begin@externalize
6650     }%
6651   \else
6652     \tikzexternalexportnexttrue
6653   \fi
6654 }

```

We're externalizing, i.e. this code gets executed in the embedded call.

```

6655 \long\def\forest@begin@externalizing#1{%
6656   \forest@external@setup{#1}%
6657   \let\forest@external@next\forest@begin
6658   \forest@externalize@inner@n=-1
6659   \ifforest@external@optimize@\forest@externalizing@maybeoptimize\fi
6660   \forest@external@next{#1}%
6661   \tikzexternalenable
6662 }
6663 \def\forest@externalizing@maybeoptimize{%
6664   \edef\forest@temp{\tikzexternalrealjob-forest-\forest@externalize@outer@n}%
6665   \edef\forest@marshal{%
6666     \noexpand\pgfutil@in@
6667     {\expandafter\detokenize\expandafter{\forest@temp}.}
6668     {\expandafter\detokenize\expandafter{\pgfactualjobname}.}%
6669   }\forest@marshal
6670   \ifpgfutil@in@
6671   \else
6672     \let\forest@external@next@gobble
6673   \fi
6674 }

```

Externalization is enabled, we're in the outer process, deciding if the picture is up-to-date.

```

6675 \long\def\forest@begin@externalize#1{%
6676   \forest@external@setup{#1}%
6677   \iftikzexternal@file@isuptodate
6678     \setbox0=\hbox{%
6679       \csname forest@externalcheck@\forest@externalize@outer@n\endcsname
6680     }%
6681   \fi
6682   \iftikzexternal@file@isuptodate
6683     \csname forest@externalload@\forest@externalize@outer@n\endcsname
6684   \else
6685     \forest@externalize@tree@true
6686     \forest@externalize@inner@n=-1
6687     \forest@begin{#1}%
6688     \ifcsdef{forest@externalize@@\forest@externalize@id}{-}{%
6689       \immediate\write\forest@auxout{%
6690         \noexpand\forest@external
6691         {\forest@externalize@outer@n}%
6692         {\expandafter\detokenize\expandafter{\forest@externalize@id}}%
6693         {\expandonce\forest@externalize@checkimages}%
6694         {\expandonce\forest@externalize@loadimages}%
6695       }%
6696     }%
6697   \fi
6698   \tikzexternalenable
6699 }
6700 \def\forest@includeexternal@check#1{%
6701   \tikzsetnextfilename{#1}%
6702   \IfFileExists{\tikzexternal@filenameprefix/#1}{\tikzexternal@file@isuptodatetrue}{\tikzexternal@file@isuptodatefalse}
6703 }
6704 \def\makehashother{\catcode'\# =12}%
6705 \long\def\forest@external@setup#1{%
6706   % set up \forest@externalize@id and \forest@externalize@outer@n
6707   % we need to deal with #s correctly (\write doubles them)
6708   \setbox0=\hbox{\makehashother\makeatletter
6709     \scantokens{\forest@temp@toks{#1}}\expandafter
6710   }%
6711   \expandafter\forest@temp@toks\expandafter{\the\forest@temp@toks}%
6712   \edef\forest@temp{\pgfkeysvalueof{/forest/external/context}}%
6713   \edef\forest@externalize@id{%
6714     \expandafter\detokenize\expandafter{\forest@temp}%
6715     @@%
6716     \expandafter\detokenize\expandafter{\the\forest@temp@toks}%
6717   }%
6718   \letcs\forest@externalize@outer@n{forest@externalize@@\forest@externalize@id}%
6719   \ifdefined\forest@externalize@outer@n
6720     \global\tikzexternal@file@isuptodatetrue
6721   \else
6722     \global\advance\forest@externalize@max@outer@n 1
6723     \edef\forest@externalize@outer@n{\the\forest@externalize@max@outer@n}%
6724     \global\tikzexternal@file@isuptodatefalse
6725   \fi
6726   \def\forest@externalize@loadimages{}%
6727   \def\forest@externalize@checkimages{}%
6728 }
6729 \newcount\forest@externalize@max@outer@n
6730 \global\forest@externalize@max@outer@n=0
6731 \newcount\forest@externalize@inner@n

```

The .for file is a string of calls of this macro.

```

6732 \long\def\forest@external#1#2#3#4{% #1=n,#2=context+source code,#3=update check code, #4=load code

```



```

6785 \forest@fornode{\the\forest@node@maxid}{%
6786 \forest@node@init
6787 \forest@node@setname@silent{standard node}%
6788 }%
6789 }
6790 \def\forest@standardnode@calibrate{%
6791 \forest@fornode{\forest@node@Nametoid{standard node}}{%
6792 \edef\forest@environment{\forestove{environment@formula}}%
6793 \forestoget{previous@environment}\forest@previous@environment
6794 \ifx\forest@environment\forest@previous@environment\else
6795 \forestolet{previous@environment}\forest@environment
6796 \forest@node@typeset
6797 \forestoget{calibration@procedure}\forest@temp
6798 \expandafter\forestset\expandafter{\forest@temp}%
6799 \fi
6800 }%
6801 }

```

Usage: `\forestStandardNode[#1]{#2}{#3}{#4}`. #1 = standard node specification — specify it as any other node content (but without children, of course). #2 = the environment fingerprint: list the values of parameters that influence the standard node’s height and depth; the standard will be adjusted whenever any of these parameters changes. #3 = the calibration procedure: a list of usual forest options which should calculating the values of exported options. #4 = a comma-separated list of exported options: every newly created node receives the initial values of exported options from the standard node. (The standard node definition is local to the  $\TeX$  group.)

```

6802 \def\forestStandardNode[#1]#2#3#4{%
6803 \let\forest@standardnode@restoretikzexternal\relax
6804 \ifdefined\tikzexternaldisable
6805 \ifx\tikz\tikzexternal@tikz@replacement
6806 \tikzexternaldisable
6807 \let\forest@standardnode@restoretikzexternal\tikzexternalenable
6808 \fi
6809 \fi
6810 \forest@standardnode@new
6811 \forest@fornode{\forest@node@Nametoid{standard node}}{%
6812 \forestset{content=#1}%
6813 \forestoset{environment@formula}{#2}%
6814 \edef\forest@temp{\unexpanded{#3}}%
6815 \forestolet{calibration@procedure}\forest@temp
6816 \def\forest@calibration@initializing@code{%
6817 \pgfkeys{/forest/initializing@code}{#4}%
6818 \forestolet{initializing@code}\forest@calibration@initializing@code
6819 \forest@standardnode@restoretikzexternal
6820 }
6821 }
6822 \forestset{initializing@code/.unknown/.code={%
6823 \eappto\forest@calibration@initializing@code{%
6824 \noexpand\forestOget{\forest@node@Nametoid{standard node}}{\pgfkeyscurrentname}\noexpand\forest@temp
6825 \noexpand\forestolet{\pgfkeyscurrentname}\noexpand\forest@temp
6826 }%
6827 }
6828 }

```

This macro is called from a new (non-standard) node’s init.

```

6829 \def\forest@initializefromstandardnode{%
6830 \forestOve{\forest@node@Nametoid{standard node}}{initializing@code}%
6831 }

```

Define the default standard node. Standard content: `dj` — in Computer Modern font, `d` is the highest and `j` the deepest letter (not character!). Environment fingerprint: the height of the strut and the values of inner and outer seps. Calibration procedure: (i) `1 sep` equals the height of the strut plus the value of

inner ysep, implementing both font-size and inner sep dependency; (ii) The effect of l on the standard node should be the same as the effect of l sep, thus, we derive l from l sep by adding to the latter the total height of the standard node (plus the double outer sep, one for the parent and one for the child). (iii) s sep is straightforward: a double inner xsep. Exported options: options, calculated in the calibration. (Tricks: to change the default anchor, set it in #1 and export it; to set a non-forest node option (such as draw or blue) as default, set it in #1 and export the (internal) option node options.)

```

6832 \forestStandardNode[dj]
6833   {%
6834     \forestOve{\forest@node@Nametoid{standard node}}{content},%
6835     \the\ht\strutbox,\the\pgflinewidth,%
6836     \pgfkeysvalueof{/pgf/inner ysep},\pgfkeysvalueof{/pgf/outer ysep},%
6837     \pgfkeysvalueof{/pgf/inner xsep},\pgfkeysvalueof{/pgf/outer xsep}%
6838   }
6839   {
6840     l sep={\the\ht\strutbox+\pgfkeysvalueof{/pgf/inner ysep}},
6841     l={l_sep()+abs(max_y()-min_y()+2*\pgfkeysvalueof{/pgf/outer ysep}},
6842     s sep={2*\pgfkeysvalueof{/pgf/inner xsep}}
6843   }
6844   {l sep,l,s sep}

```

## 9.4 ls coordinate system

```

6845 \pgfqkeys{/forest/@cs}{%
6846   name/.code={%
6847     \edef\forest@cn{\forest@node@Nametoid{#1}}%
6848     \forest@forestcs@resetxy},
6849   id/.code={%
6850     \edef\forest@cn{#1}%
6851     \forest@forestcs@resetxy},
6852   go/.code={%
6853     \forest@go{#1}%
6854     \forest@forestcs@resetxy},
6855   anchor/.code={\forest@forestcs@anchor{#1}},
6856   l/.code={%
6857     \pgfmathsetlengthmacro\forest@forestcs@l{#1}%
6858     \forest@forestcs@ls
6859   },
6860   s/.code={%
6861     \pgfmathsetlengthmacro\forest@forestcs@s{#1}%
6862     \forest@forestcs@ls
6863   },
6864   .unknown/.code={%
6865     \expandafter\pgfutil@in@\expandafter.\expandafter{\pgfkeyscurrentname}%
6866     \ifpgfutil@in@
6867       \expandafter\forest@forestcs@namegoanchor\pgfkeyscurrentname\forest@end
6868     \else
6869       \expandafter\forest@nameandgo\expandafter{\pgfkeyscurrentname}%
6870       \forest@forestcs@resetxy
6871     \fi
6872   }
6873 }
6874 \def\forest@forestcs@resetxy{%
6875   \ifnum\forest@cn=0 \forest@cs@invalidnodeerror\fi
6876   \global\pgf@x\forestove{x}%
6877   \global\pgf@y\forestove{y}%
6878 }
6879 \def\forest@forestcs@ls{%
6880   \ifdefined\forest@forestcs@l
6881     \ifdefined\forest@forestcs@s
6882     {%

```

```

6883     \pgftransformreset
6884     \pgftransformrotate{\forestove{grow}}%
6885     \pgfpointransformed{\pgfpoint{\forest@forestcs@1}{\forest@forestcs@s}}%
6886     }%
6887     \global\advance\pgf@x\forestove{x}%
6888     \global\advance\pgf@y\forestove{y}%
6889     \fi
6890   \fi
6891 }
6892 \def\forest@forestcs@anchor#1{%
6893   \edef\forest@marshal{%
6894     \noexpand\forest@original@tikz@parse@node\relax
6895     (\forestove{name}\ifx\relax#1\relax\else.\fi#1)%
6896   }\forest@marshal
6897 }
6898 \def\forest@forestcs@namegoanchor#1.#2\forest@end{%
6899   \forest@nameandgo{#1}%
6900   \ifnum\forest@cn=0 \forest@cs@invalidnodeerror\fi
6901   \forest@forestcs@anchor{#2}%
6902 }
6903 \def\forest@cs@invalidnodeerror{%
6904   \PackageError{forest}{Attempt to refer to the invalid node by "forest cs"}{}%
6905 }
6906 \tikzdeclarecoordinatesystem{forest}{%
6907   \forest@forthis{%
6908     \forest@forestcs@resetxy
6909     \ifdefined\forest@forestcs@1\undef\forest@forestcs@1\fi
6910     \ifdefined\forest@forestcs@s\undef\forest@forestcs@s\fi
6911     \pgfqkeys{/forest/@cs}{#1}%
6912   }%
6913 }

```

## 9.5 Relative node names in TikZ

A hack into TikZ's coordinate parser: implements relative node names!

```

6914 \def\forest@tikz@parse@node#1(#2){%
6915   \pgfutil@in@.#2}%
6916   \ifpgfutil@in@
6917     \expandafter\forest@tikz@parse@node@checkiftikzname@withdot
6918   \else%
6919     \expandafter\forest@tikz@parse@node@checkiftikzname@withoutdot
6920   \fi%
6921   #1(#2)\forest@end
6922 }
6923 \def\forest@tikz@parse@node@checkiftikzname@withdot#1(#2.#3)\forest@end{%
6924   \forest@tikz@parse@node@checkiftikzname#1{#2}{.#3}}
6925 \def\forest@tikz@parse@node@checkiftikzname@withoutdot#1(#2)\forest@end{%
6926   \forest@tikz@parse@node@checkiftikzname#1{#2}{}}
6927 \def\forest@tikz@parse@node@checkiftikzname#1#2#3{%
6928   \expandafter\ifx\csname pgf@sh@ns@#2\endcsname\relax
6929     \forest@forthis{%
6930       \forest@nameandgo{#2}%
6931       \ifnum\forest@cn=0 \forest@cs@invalidnodeerror\fi
6932       \edef\forest@temp@relativenodename{\forestove{name}}%
6933     }%
6934   \else
6935     \def\forest@temp@relativenodename{#2}%
6936   \fi
6937   \expandafter\forest@original@tikz@parse@node\expandafter#1\expandafter(\forest@temp@relativenodename#3)%
6938 }
6939 \def\forest@nameandgo#1{%

```

```

6940 \pgfutil@in@!{#1}%
6941 \ifpgfutil@in@
6942 \forest@nameandgo@(#1)%
6943 \else
6944 \ifstrempy{#1}{-}{\edef\forest@cn{\forest@node@Nametoid{#1}}}%
6945 \fi
6946 }
6947 \def\forest@nameandgo@(#1!#2){%
6948 \ifstrempy{#1}{-}{\edef\forest@cn{\forest@node@Nametoid{#1}}}%
6949 \forest@go{#2}%
6950 }

```

## 9.6 Anchors

FOREST anchors are (child/parent)\_anchor and growth anchors parent/children\_first/last. The following code resolves them into TikZ anchors, based on the value of option (child/parent)\_anchor and values of grow and reversed.

We need to access rotate for the anchors below to work in general.

```

6951 \forestset{
6952 declare count={rotate}{0},
6953 autoforward'={rotate}{node options},
6954 }

```

Variants of parent/children\_first/last without ' snap border anchors to the closest compass direction.

```
6955 \newif\ifforest@anchor@snabbordertocompass
```

The code is used both in generic anchors (then, the result should be forwarded to TikZ for evaluation into coordinates) and in the UI macro \forestanchortotikzanchor.

```
6956 \newif\ifforest@anchor@forwardtotikz
```

Growth-based anchors set this to true to signal that the result is a border anchor.

```
6957 \newif\ifforest@anchor@isborder
```

The UI macro.

```

6958 \def\forestanchortotikzanchor#1#2{% #1 = forest anchor, #2 = macro to receive the tikz anchor
6959 \forest@anchor@forwardtotikzfalse
6960 \forest@anchor@do{#1}{\forest@cn}%
6961 \let#2\forest@temp@anchor
6962 }

```

Generic anchors.

```

6963 \pgfdeclaregenericanchor{child anchor}{%
6964 \forest@anchor@forwardtotikztrue
6965 \forest@anchor@do{#1}{child anchor}{\forest@referencednodeid}%
6966 }
6967 \pgfdeclaregenericanchor{parent anchor}{%
6968 \forest@anchor@forwardtotikztrue
6969 \forest@anchor@do{#1}{parent anchor}{\forest@referencednodeid}%
6970 }
6971 \pgfdeclaregenericanchor{anchor}{%
6972 \forest@anchor@forwardtotikztrue
6973 \forest@anchor@do{#1}{anchor}{\forest@referencednodeid}%
6974 }
6975 \pgfdeclaregenericanchor{children}{%
6976 \forest@anchor@forwardtotikztrue
6977 \forest@anchor@do{#1}{children}{\forest@referencednodeid}%
6978 }
6979 \pgfdeclaregenericanchor{children first}{%
6980 \forest@anchor@forwardtotikztrue
6981 \forest@anchor@do{#1}{children first}{\forest@referencednodeid}%
6982 }

```



```

6983 \pgfdeclaregenericanchor{first}{%
6984   \forest@anchor@forwardtotikztrue
6985   \forest@anchor@do{#1}{first}{\forest@referencednodeid}%
6986 }
6987 \pgfdeclaregenericanchor{parent first}{%
6988   \forest@anchor@forwardtotikztrue
6989   \forest@anchor@do{#1}{parent first}{\forest@referencednodeid}%
6990 }
6991 \pgfdeclaregenericanchor{parent}{%
6992   \forest@anchor@forwardtotikztrue
6993   \forest@anchor@do{#1}{parent}{\forest@referencednodeid}%
6994 }
6995 \pgfdeclaregenericanchor{parent last}{%
6996   \forest@anchor@forwardtotikztrue
6997   \forest@anchor@do{#1}{parent last}{\forest@referencednodeid}%
6998 }
6999 \pgfdeclaregenericanchor{last}{%
7000   \forest@anchor@forwardtotikztrue
7001   \forest@anchor@do{#1}{last}{\forest@referencednodeid}%
7002 }
7003 \pgfdeclaregenericanchor{children last}{%
7004   \forest@anchor@forwardtotikztrue
7005   \forest@anchor@do{#1}{children last}{\forest@referencednodeid}%
7006 }
7007 \pgfdeclaregenericanchor{children'}{%
7008   \forest@anchor@forwardtotikztrue
7009   \forest@anchor@do{#1}{children'}{\forest@referencednodeid}%
7010 }
7011 \pgfdeclaregenericanchor{children first'}{%
7012   \forest@anchor@forwardtotikztrue
7013   \forest@anchor@do{#1}{children first'}{\forest@referencednodeid}%
7014 }
7015 \pgfdeclaregenericanchor{first'}{%
7016   \forest@anchor@forwardtotikztrue
7017   \forest@anchor@do{#1}{first'}{\forest@referencednodeid}%
7018 }
7019 \pgfdeclaregenericanchor{parent first'}{%
7020   \forest@anchor@forwardtotikztrue
7021   \forest@anchor@do{#1}{parent first'}{\forest@referencednodeid}%
7022 }
7023 \pgfdeclaregenericanchor{parent'}{%
7024   \forest@anchor@forwardtotikztrue
7025   \forest@anchor@do{#1}{parent'}{\forest@referencednodeid}%
7026 }
7027 \pgfdeclaregenericanchor{parent last'}{%
7028   \forest@anchor@forwardtotikztrue
7029   \forest@anchor@do{#1}{parent last'}{\forest@referencednodeid}%
7030 }
7031 \pgfdeclaregenericanchor{last'}{%
7032   \forest@anchor@forwardtotikztrue
7033   \forest@anchor@do{#1}{last'}{\forest@referencednodeid}%
7034 }
7035 \pgfdeclaregenericanchor{children last'}{%
7036   \forest@anchor@forwardtotikztrue
7037   \forest@anchor@do{#1}{children last'}{\forest@referencednodeid}%
7038 }

```

The driver. The result is being passed around in \forest@temp@anchor.

```

7039 \def\forest@anchor@do#1#2#3{% #1 = shape name, #2 = (potentially) forest anchor, #3 = node id
7040   \forest@fornode{#3}{%
7041     \def\forest@temp@anchor{#2}%

```

```

7042 \forest@anchor@snapbordertocompassfalse
7043 \forest@anchor@isborderfalse
7044 \forest@anchor@to@tikz@anchor
7045 \forest@anchor@border@to@compass
7046 \ifforest@anchor@forwardtotikz
7047 \forest@anchor@forward{#1}%
7048 \else
7049 \fi
7050 }%
7051 }

```

This macro will loop (resolving the anchor) until the result is not a FOREST macro.

```

7052 \def\forest@anchor@to@tikz@anchor{%
7053 \ifcsdef{forest@anchor@@\forest@temp@anchor}{%
7054 \csuse{forest@anchor@@\forest@temp@anchor}%
7055 \forest@anchor@to@tikz@anchor
7056 }{}%
7057 }

```

Actual computation.

```

7058 \csdef{forest@anchor@@parent anchor}{%
7059 \foresttoget{parent anchor}\forest@temp@anchor}
7060 \csdef{forest@anchor@@child anchor}{%
7061 \foresttoget{child anchor}\forest@temp@anchor}
7062 \csdef{forest@anchor@@anchor}{%
7063 \foresttoget{anchor}\forest@temp@anchor}
7064 \csdef{forest@anchor@@children'}{%
7065 \forest@anchor@isbordertrue
7066 \edef\forest@temp@anchor{\number\numexpr\forestove{grow}-\forestove{rotate}}%
7067 }
7068 \csdef{forest@anchor@@parent'}{%
7069 \forest@anchor@isbordertrue
7070 \edef\forest@temp@anchor{\number\numexpr\forestove{grow}-\forestove{rotate}+180}%
7071 }
7072 \csdef{forest@anchor@@first'}{%
7073 \forest@anchor@isbordertrue
7074 \edef\forest@temp@anchor{\number\numexpr\forestove{grow}-\forestove{rotate}\ifnum\forestove{reversed}=0 -\e
7075 }
7076 \csdef{forest@anchor@@last'}{%
7077 \forest@anchor@isbordertrue
7078 \edef\forest@temp@anchor{\number\numexpr\forestove{grow}-\forestove{rotate}\ifnum\forestove{reversed}=0 +\e
7079 }
7080 \csdef{forest@anchor@@parent first'}{%
7081 \forest@anchor@isbordertrue
7082 \edef\forest@temp@anchor@parent{\number\numexpr\forestove{grow}-\forestove{rotate}+180}%
7083 \edef\forest@temp@anchor@first{\number\numexpr\forestove{grow}-\forestove{rotate}\ifnum\forestove{reversed}
7084 \forest@getaverageangle{\forest@temp@anchor@parent}{\forest@temp@anchor@first}\forest@temp@anchor
7085 }
7086 \csdef{forest@anchor@@parent last'}{%
7087 \forest@anchor@isbordertrue
7088 \edef\forest@temp@anchor@parent{\number\numexpr\forestove{grow}-\forestove{rotate}+180}%
7089 \edef\forest@temp@anchor@last{\number\numexpr\forestove{grow}-\forestove{rotate}\ifnum\forestove{reversed}=
7090 \forest@getaverageangle{\forest@temp@anchor@parent}{\forest@temp@anchor@last}\forest@temp@anchor
7091 }
7092 \csdef{forest@anchor@@children first'}{%
7093 \forest@anchor@isbordertrue
7094 \edef\forest@temp@anchor@first{\number\numexpr\forestove{grow}-\forestove{rotate}\ifnum\forestove{reversed}
7095 \forest@getaverageangle{\forestove{grow}-\forestove{rotate}}{\forest@temp@anchor@first}\forest@temp@anchor
7096 }
7097 \csdef{forest@anchor@@children last'}{%
7098 \forest@anchor@isbordertrue
7099 \edef\forest@temp@anchor@last{\number\numexpr\forestove{grow}-\forestove{rotate}\ifnum\forestove{reversed}=

```

```

7100 \forest@getaverageangle{\forestove{grow}-\forestove{rotate}}{\forest@temp@anchor@last}\forest@temp@anchor
7101 }
7102 \csdef{forest@anchor@@children}{%
7103 \forest@anchor@snapbordertocompasstrue
7104 \csuse{forest@anchor@@children'}%
7105 }
7106 \csdef{forest@anchor@@parent}{%
7107 \forest@anchor@snapbordertocompasstrue
7108 \csuse{forest@anchor@@parent'}%
7109 }
7110 \csdef{forest@anchor@@first}{%
7111 \forest@anchor@snapbordertocompasstrue
7112 \csuse{forest@anchor@@first'}%
7113 }
7114 \csdef{forest@anchor@@last}{%
7115 \forest@anchor@snapbordertocompasstrue
7116 \csuse{forest@anchor@@last'}%
7117 }
7118 \csdef{forest@anchor@@parent first}{%
7119 \forest@anchor@snapbordertocompasstrue
7120 \csuse{forest@anchor@@parent first'}%
7121 }
7122 \csdef{forest@anchor@@parent last}{%
7123 \forest@anchor@snapbordertocompasstrue
7124 \csuse{forest@anchor@@parent last'}%
7125 }
7126 \csdef{forest@anchor@@children first}{%
7127 \forest@anchor@snapbordertocompasstrue
7128 \csuse{forest@anchor@@children first'}%
7129 }
7130 \csdef{forest@anchor@@children last}{%
7131 \forest@anchor@snapbordertocompasstrue
7132 \csuse{forest@anchor@@children last'}%
7133 }

```

This macro computes the "average" angle of #1 and #2 and stores in into #3. The angle computed is the geometrically "closer" one. The formula is adapted from <http://stackoverflow.com/a/1159336/624872>.

```

7134 \def\forest@getaverageangle#1#2#3{%
7135 \edef\forest@temp{\number\numexpr #1-#2+540}%
7136 \pgfmathMod{\forest@temp}{360}\pgfmathtruncatemacro\pgfmathresult{\pgfmathresult}
7137 \edef\forest@temp{360+#2+(\pgfmathresult-180)/2}%
7138 \pgfmathMod{\forest@temp}{360}\pgfmathtruncatemacro#3{\pgfmathresult}%
7139 }

```

The first macro changes border anchor to compass anchor. The second one does this only if the node shape allows it.

```

7140 \def\forest@anchor@border@to@compass{%
7141 \ifforest@anchor@isborder
7142 \ifforest@anchor@snapbordertocompass
7143 \forest@anchor@snap@border@to@compass
7144 \else
7145 \pgfmathMod{\forest@temp@anchor}{360}%
7146 \pgfmathtruncatemacro\forest@temp@anchor{\pgfmathresult}%
7147 \fi
7148 \ifforest@anchor@forwardtotikz
7149 \ifcsname pgf@anchor%
7150 @\csname pgf@sh@ns@\pgfreferencednodename\endcsname
7151 @\csname forest@compass@\forest@temp@anchor\endcsname
7152 \endcsname
7153 \letcs\forest@temp@anchor{forest@compass@\forest@temp@anchor}%

```

```

7154     \fi
7155     \else
7156         \ifforest@anchor@snapbordertocompass
7157             \letcs\forest@temp@anchor{forest@compass@\forest@temp@anchor}%
7158         \fi
7159     \fi
7160 \fi
7161 }
7162 \csdef{forest@compass@0}{east}
7163 \csdef{forest@compass@45}{north east}
7164 \csdef{forest@compass@90}{north}
7165 \csdef{forest@compass@135}{north west}
7166 \csdef{forest@compass@180}{west}
7167 \csdef{forest@compass@225}{south west}
7168 \csdef{forest@compass@270}{south}
7169 \csdef{forest@compass@315}{south east}
7170 \csdef{forest@compass@360}{east}

```

This macro approximates an angle (stored in `\forest@temp@anchor`) with a compass direction (stores it in the same macro).

```

7171 \def\forest@anchor@snap@border@to@compass{%
7172     \pgfmathMod{\forest@temp@anchor}{360}%
7173     \pgfmathdivide{\pgfmathresult}{45}%
7174     \pgfmathround{\pgfmathresult}%
7175     \pgfmathmultiply{\pgfmathresult}{45}%
7176     \pgfmathtruncatemacro\forest@temp@anchor{\pgfmathresult}%
7177 }

```

This macro forwards the resulting anchor to *TikZ*.

```

7178 \def\forest@anchor@forward#1{% #1 = shape name
7179     \ifdefempty\forest@temp@anchor{%
7180         \pgf@sh@reanchor{#1}{center}%
7181         \xdef\forest@hack@tikzshapeborder{%
7182             \noexpand\tikz@shapebordertrue
7183             \def\noexpand\tikz@shapeborder@name{\pgfreferencednodename}%
7184             }\aftergroup\forest@hack@tikzshapeborder
7185     }{%
7186         \pgf@sh@reanchor{#1}{\forest@temp@anchor}%
7187     }%
7188 }

```

Expandably strip "not yet positionedPGFINTERNAL" from `\pgfreferencednodename` if it is there.

```

7189 \def\forest@referencednodeid{\forest@node@Nametoid{\forest@referencednodename}}%
7190 \def\forest@referencednodename{%
7191     \expandafter\expandafter\expandafter\forest@referencednodename@\expandafter\pgfreferencednodename\forest@pgf@notyetpositioned
7192 }
7193 \expandafter\def\expandafter\forest@referencednodename@\expandafter#\expandafter1\forest@pgf@notyetpositioned
7194     \if\relax#1\relax\forest@referencednodename@stripafter#2\relax\fi
7195     \if\relax#2\relax#1\fi
7196 }
7197 \expandafter\def\expandafter\forest@referencednodename@stripafter\expandafter#\expandafter1\forest@pgf@notyetpositioned

```

This macro sets up `\pgf@x` and `\pgf@y` to the given anchor's coordinates, within the node's coordinate system. It works even before the node was positioned. If the anchor is empty, i.e. if is the implicit border anchor, we return the coordinates for the center.

```

7198 \def\forest@Pointanchor#1#2{% #1 = node id, #2 = anchor
7199     {%
7200         \def\forest@pa@temp@name{name}%
7201         \forestOifdefined{#1}{@box}{%
7202             \forestOget{#1}{@box}\forest@temp
7203             \ifdefempty\forest@temp{}{%
7204                 \def\forest@pa@temp@name{later@name}%

```

```

7205     }%
7206   }{}%
7207   \setbox0\hbox{%
7208     \begin{pgfpicture}%
7209       \if\relax\forestOve{#1}{#2}\relax
7210         \pgfpointanchor{\forestOve{#1}{\forest@pa@temp@name}}{center}%
7211       \else
7212         \pgfpointanchor{\forestOve{#1}{\forest@pa@temp@name}}{\forestOve{#1}{#2}}%
7213       \fi
7214       \xdef\forest@global@marshal{%
7215         \noexpand\global\noexpand\pgf@x=\the\pgf@x\relax
7216         \noexpand\global\noexpand\pgf@y=\the\pgf@y
7217       }%
7218     \end{pgfpicture}%
7219   }%
7220 }%
7221 \forest@global@marshal
7222 }
7223 \def\forest@pointanchor#1{% #1 = anchor
7224   \forest@Pointanchor{\forest@cn}{#1}%
7225 }

```

## 10 Compatibility with previous versions

```

7226 \ifdefempty{\forest@compat}{}{%
7227   \RequirePackage{forest-compat}
7228 }

```