

The `proofgraph` package

Pierre Senellart
pierre@senellart.com

2026/07/10 v1.0.1

Abstract

The `proofgraph` package automatically produces a graph of the dependencies between the results (theorems, lemmas, propositions) of a mathematical article. It infers an edge from one result to another whenever the proof of the former refers to the latter through an ordinary cross-reference, so that most dependencies need no manual annotation; commands are provided for those a visible reference does not capture. The graph is written as a `Graphviz .dot` file, which can be rendered externally or, optionally, embedded into the document automatically.

1 Introduction

When writing or reading a mathematical article, it is often useful to visualise how the results depend on one another: which lemma is used in the proof of which theorem, and so on. `proofgraph` builds such a dependency graph automatically. Each theorem-like environment becomes a node; each cross-reference (`\ref`, `\cref`...) appearing *inside a proof* becomes an edge from the proved result to the referred-to result. Usually no manual annotation is required, and commands are provided for the dependencies a visible reference does not capture.

The package is best explained by a small example of its use. This manual is itself a `proofgraph` document, so the results stated below are typeset with the package enabled, and the graph shown at the end of this section is the one they produce.

Load `proofgraph` in the preamble, before the `\newtheorem` commands that declare your theorem-like environments (so it sees them as they are created). Here, as in this manual, we pass three options: `autorun` renders the graph with `Graphviz` and embeds it automatically (it needs shell-escape); `cite` captures the `\cites` made inside proofs as dependencies on external work; and `citelabel=tag` labels those citation nodes by their printed tag:

```
\usepackage[autorun,cite,citelabel=tag]{proofgraph}
\newtheorem{theorem}{Theorem}
\newtheorem{lemma}{Lemma}
\newtheorem{corollary}{Corollary}
```

Then state and prove, as usual, a few results that build on one another:

```
\begin{lemma}\label{lem:even}
  The sum of two even integers is even.
\end{lemma}
```

```

\begin{lemma}\label{lem:odd}
  The sum of two odd integers is even.
\end{lemma}
\begin{theorem}\label{thm:parity}
  The sum of two integers of the same parity is even.
\end{theorem}
\begin{proof}
  By Lemmas~\ref{lem:even} and~\ref{lem:odd}; see~\cite{euclid}.
\end{proof}
\begin{corollary}\label{cor:double}
  For every integer  $n$ , the number  $2n$  is even.
\end{corollary}
\begin{proof}
  Apply Theorem~\ref{thm:parity} to  $n$  and  $n$ .
\end{proof}

```

We obtain the following results, exactly as without the package:

Lemma 1. *The sum of two even integers is even.*

Lemma 2. *The sum of two odd integers is even.*

Theorem 1. *The sum of two integers of the same parity is even.*

Proof. By Lemmas 1 and 2; see [1]. □

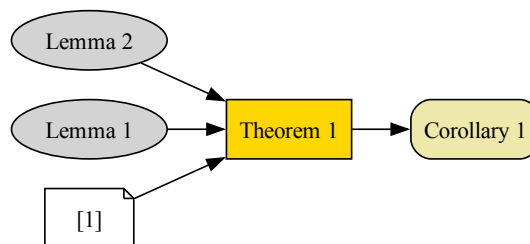
Corollary 1. *For every integer n , the number $2n$ is even.*

Proof. Apply Theorem 1 to n and n . □

In addition, `proofgraph` writes out the dependency graph. With the `autorun` option it is rendered with `Graphviz` and included wherever you call `\proofgraph`:

```
\proofgraph[width=0.7\textwidth]
```

which here produces:



Each cross-reference made inside a proof becomes an arrow *into* the result that proof establishes: the two lemmas point to Theorem 1, which in turn points to Corollary 1. The `\cite` in the proof of Theorem 1 is captured in the same way, thanks to the `cite` option: the external work appears as a distinct node (drawn note-shaped, and labelled here by its printed citation tag, “[1]”, because of `citelabel=tag`) with an arrow into the theorem it helps prove. Moreover, since

this manual loads `hyperref` and `TikZ`, the graph’s nodes are *clickable*: try clicking one to jump to the result it represents (see `\proofgraph` and the `hyperlinks` option).

A handful of commands tune such a graph: `\proofgraphstyle` sets the appearance of each kind of node (the colours here come from it); `\uses` and `\proofgraphedge` record a dependency that no visible `\ref` expresses; `\proofof` attaches a proof to its result; `\proofgraphuntrack` keeps an environment out of the graph; and `\proofgraphignore` and `\proofgraphexclude` drop an unwanted edge or node. The next section puts several of these to work on a real article; Section 3 then documents them, and the package options, in full.

2 A real-world example

The introductory example is deliberately tiny. To show what `proofgraph` produces on a genuine article, the graph in Figure 1 is the one obtained from *Connecting Knowledge Compilation Classes and Width Parameters* [2]. It has 62 numbered results, drawn from five environments (`result`, `theorem`, `proposition`, `corollary` and `lemma`), depends on 19 external works cited inside proofs, and contains 106 edges in all.

The graph was produced from the paper essentially as published. Beyond loading the package, the only additions were the following.

- The `cite` option was switched on, and citation nodes labelled by their printed tag, by loading the package as `\usepackage[cite=true,citelabel=tag]{proofgraph}`. The `\cites` made inside proofs then appear as the note-shaped nodes (“[17]”, “Theorem 4.10, [46]”...) along the left of the graph; a work cited for two different results, or for two of its own theorems, gives one node per `\cite` note, so the 19 works appear as 32 nodes.
- One environment was kept out of the graph with a single `\proofgraphuntrack{observation}` (a lone preliminary observation that nothing else uses).
- The five result kinds were styled, with decreasing prominence, by five `\proofgraphstyle` declarations:

```
\proofgraphstyle{result}{shape=box,style="filled,bold",%
    fillcolor=orange,peripheries=2}
\proofgraphstyle{theorem}{shape=box,style=filled,%
    fillcolor=gold,penwidth=2}
\proofgraphstyle{corollary}{shape=box,style="rounded,filled",%
    fillcolor=palegoldenrod}
\proofgraphstyle{proposition}{shape=box,style="rounded,filled",%
    fillcolor=lightblue}
\proofgraphstyle{lemma}{shape=ellipse,style=filled,%
    fillcolor=lightgray}
```

- Eleven `\proofgraphedge` commands, recording 27 edges in total, were added for the dependencies that no `\ref` inside a `proof` expresses: eight “obvious” corollaries stated without a `proof` environment

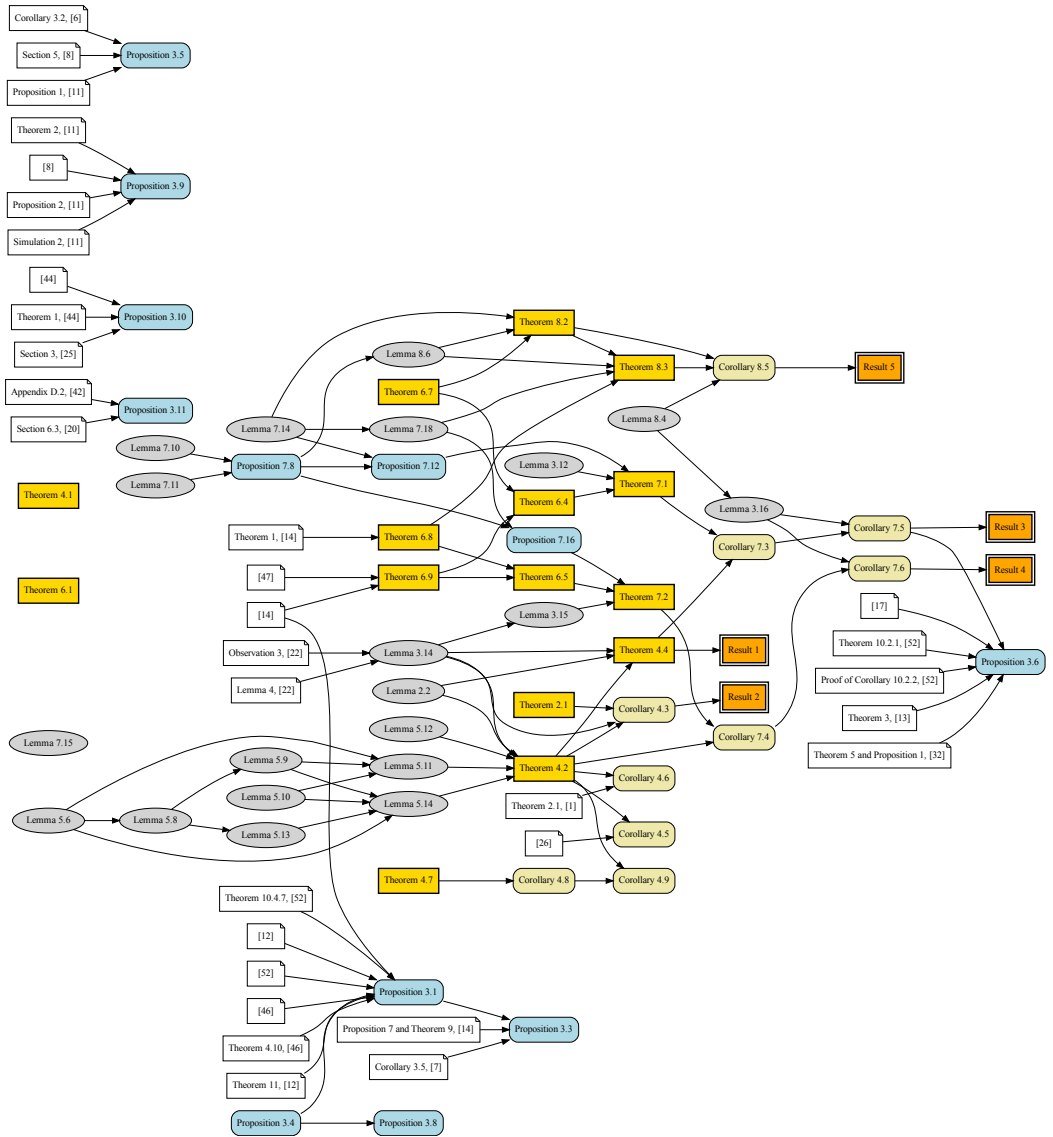


Figure 1: The graph proofgraph produces for [2]. The colours and shapes make the structure legible, with the orange double-bordered introduction results on the right, fed by the gold theorems, the blue and pale-gold propositions and corollaries, and the grey lemmas.

(`\proofgraphedge{cor:upper_bound}{thm:upper_bound,...}`), and three theorems whose proof is developed across a whole section rather than inside a single proof.

Nothing else was annotated: every other edge, and the numbering of all 62 nodes, was inferred automatically from the ordinary cross-references the proofs already contained. In particular, no `\uses`, `\proofof`, `\proofgraphignore` or `\proofgraphexclude` was needed. The five `result` environments that restate the headline results in the introduction (`\begin{result}[(Corollary~\ref{cor:obdd_omega})]...`) are linked to the body results they preview through the `\ref` in their optional title, captured automatically (see the note on reference-carrying titles in Section 3). Two compilation runs (plus `BIBTEX`, needed once for the `tag` labels) suffice; the `.dot` file is then rendered with `Graphviz`, `dot -Tpdf main.dot -o main.pdf`, as with any other document.

3 Usage

3.1 Default behaviour

Load `proofgraph` in the preamble. It only needs to come *after* whatever provides your theorem environments, that is, after `\usepackage{amsthm}` if you load it yourself (some classes, such as `lipics` or `acmart`, provide theorems for you, in which case there is nothing to load first), and *before* the `\newtheorem` commands that declare your theorem-like environments, so that `proofgraph` sees them as they are created:

```
\usepackage{amsthm}      % or nothing, if your class provides theorems
\usepackage{proofgraph}
\newtheorem{theorem}{Theorem}
\newtheorem{lemma}{Lemma}
```

Then write your document as usual. After two compilation runs, a file `\jobname.dot` is produced. Render it with `Graphviz`:

```
dot -Tpdf myfile.dot -o mygraph.pdf
```

Every *numbered* result of a theorem-like environment you declare becomes a node, and every cross-reference (`\ref`, `\cref...`) inside a proof becomes an edge into the result that proof establishes, with no annotation. Nodes are labelled by the result’s formatted name and number (“Theorem 1”), not the environment name. Unnumbered (starred) results are skipped, as they are usually expository or repeated statements; `\proofgraphtrack` draws a chosen one anyway. A cross-reference in the *optional title* of a result is captured the same way: `\begin{theorem}[(generalises~\ref{thm:x})]` in a result labelled $\langle A \rangle$ adds an edge from $\langle A \rangle$ to `thm:x`, handy for restatements (e.g., an introduction that previews results with `\begin{result}[(Corollary~\ref{cor:y})]`).

3.2 Recording dependencies by hand

`\uses{\label}` Inside a proof, records a dependency on the result labelled $\langle label \rangle$ even when no visible reference is made. A comma-separated list is accepted, as in `\uses{lem:a,lem:b}`.

`\proofgraphedge{⟨A⟩}{⟨labels⟩}` Adds edges recording that result $\langle A \rangle$ depends on the result(s) $\langle labels \rangle$ (a comma-separated list), like a `\uses` but stated outside any proof. Handy for an “obvious” corollary that relies on earlier results without a `proof` environment, as in `\proofgraphedge{cor:x}{thm:y,lem:z}`. It may appear anywhere; all labels are resolved when the graph is written.

`\proofof{⟨label⟩}` By default a proof is attached to the most recently started result. A detached proof whose optional title names its result is attached automatically: `\begin{proof}` with `[of Theorem~\ref{thm:x}]` attaches the proof to `thm:x` (for the `amsthm` proof environment, for `proof` defined as a theorem in the Springer classes, and for deferred proofs as with `apxproof`). When none of this applies, `\proofof` inside the proof pins it to the result labelled $\langle label \rangle$.

3.3 Choosing what appears in the graph

Result environments are detected automatically; these commands adjust which results, and which edges, are kept.

`\proofgraphtrack{⟨names⟩}` Forces the listed environment types to be tracked, overriding the default exclusions, which are `definition`, `example`, `remark` and `note`. Use it in the preamble.

`\proofgraphuntrack{⟨names⟩}` Excludes further environment types, as in `\proofgraphuntrack{observation}`. Use it in the preamble.

`\proofgraphignore{⟨from⟩}{⟨to⟩}` Suppresses the edge from $\langle from \rangle$ to $\langle to \rangle$ (e.g., an unwanted forward citation).

`\proofgraphexclude{⟨label⟩}` Removes the result labelled $\langle label \rangle$, together with all its incident edges, from the graph.

3.4 Styling the graph

The attributes are passed verbatim to Graphviz; see its attribute reference, <https://graphviz.org/doc/info/attrs.html>, for the node attributes available (and <https://graphviz.org/doc/info/shapes.html> for the shapes).

`\proofgraphstyle{⟨env⟩}{⟨attrs⟩}` Sets Graphviz node attributes for all results of the environment $\langle env \rangle$, e.g., `shape=ellipse` or `style=filled,fillcolor=gold`.

`\proofgraphstylecite{⟨attrs⟩}` Sets the Graphviz attributes of the external citation nodes added by the `cite` option (default `shape=note`), as in `\proofgraphstylecite{shape=box,style=dashed,color=gray}`.

3.5 Embedding the rendered graph

`\proofgraph[⟨options⟩]` Includes the rendered graph image at this point, passing the optional $\langle options \rangle$ to the underlying `\includegraphics` (e.g., `\proofgraph[width=\linewidth]`). This is the only feature that needs

`graphicx`, which the package therefore does not load on its own: a document that uses `\proofgraph` must load `graphicx` itself (loading `tikz` for the `hyperlinks` overlay also suffices). With the `autorun` option, `Graphviz` is run at the end of every compilation (shell-escape required) to produce that image, so `\proofgraph` embeds the graph from the previous run and is up to date after two runs. Without `autorun`, no `Graphviz` is run: `\proofgraph` embeds a pre-existing `\jobname-proofgraph.<format>` file if you have rendered one under that name, and otherwise warns. When `hyperref` and `TikZ` are both loaded (with `hyperlinks` on and the `-Tplain` file from `autorun` present), every node becomes an internal hyperlink to the result it represents: `Graphviz` writes the node positions to that `-Tplain` file and `\proofgraph` overlays a transparent `\hyperref` area on each node (needed because the link annotations `Graphviz` puts in the `.pdf` are discarded by `\includegraphics`).

3.6 Options

- `selfloops` Either `remove` (default), dropping edges from a result to itself, or `keep`, retaining them.
- `autorun` Boolean (default false). Runs `Graphviz` automatically at the end of the run; requires compilation with shell-escape enabled.
- `engine` `Graphviz` layout engine (default `dot`); any `Graphviz` engine works, such as `neato`, `fdp`, `sfdp`, `circo` or `twopi` (see <https://graphviz.org/docs/layouts/>).
- `format` Output format for `autorun` and `\proofgraph` (default `pdf`); any `Graphviz` output format works, such as `png`, `svg` or `ps` (see <https://graphviz.org/docs/outputs/>).
- `hyperlinks` Boolean (default true). Makes the nodes of an embedded graph clickable when `hyperref` and `TikZ` are available (see `\proofgraph`); set it false to embed the graph as a plain image.
- `direction` Either `usedby` (default), orienting an edge from the result that is used to the result that uses it (so an arrow `a->b` reads “a is used by b”), or `uses`, the reverse.
- `rankdir` `Graphviz` graph direction (default LR).
- `cite` Boolean (default false). Also captures `\cite` inside proofs as dependencies on external work, drawn as distinct nodes (Section 2). A `\cite` note is kept: `\cite[Thm~3]{key}` labels the node “Thm 3, ...”. In the two-optional-argument form of `natbib` and `biblatex`, `\cite[see][Thm~3]{key}`, the post-note likewise labels the node and the pre-note is ignored. When `\citet` and `\citep` are defined (`natbib`, `biblatex`), they are captured in the same way, including their starred variants.
- `citelabel` Either `key` (default), labelling citation nodes by their `BIBTEX` key, or `tag`, using the printed citation tag instead (“[1]”, “[Knu74]”, “[Abiteboul et al. 1995]”...), recovered from the `.aux` (so it needs two runs). It supports the standard, `hyperref`-wrapped and `natbib` (numeric and author–year) bibliographies; if no clean tag can be recovered it keeps the key.

`file` Base name of the output file (default `\jobname`).

3.7 Use with `apxproof`

`apxproof` moves proofs to the appendix and typesets them there, so the references a proof makes are only executed at that later point. To attribute them to the right result, `proofgraph` relies on the hook `\apxproofhook`, which `apxproof` fires inside each deferred proof. This hook is available in `apxproof v1.4.1` and later; load the two packages in either order and no further setup is needed.

With an *earlier* `apxproof` the document still compiles cleanly and every result still becomes a node, but *no proof dependencies are captured for the proofs deferred to the appendix*, because `proofgraph` cannot tell which appendix proof belongs to which result (dependencies from any proofs left in the main text are still captured as usual). In that case, add the deferred ones manually with `\uses` (and `\proofof` to pin a proof to its result), or upgrade `apxproof`.

4 Limitations

- Environments declared through `\newtheorem` (or `\spnewtheorem`) *after* the package is loaded are tracked automatically. Environments predefined by the document class *before* that (as in `lipics`, `acmart` or the Springer classes) are also detected, provided their name is one of the common result names `proofgraph` probes at `\begin{document}` (`theorem`, `lemma`, `proposition`, `corollary`...); an environment with an unusual name predefined before the package is loaded must be added with `\proofgraphtrack`.
- Results must be labelled for references to them to be resolved.
- Two compilation runs are required, as for any cross-reference.

References

- [1] Euclid. *Elements*, Book IX. c. 300 BCE.
- [2] A. Amarilli, F. Capelli, M. Monet, and P. Senellart. Connecting knowledge compilation classes and width parameters. *Theory of Computing Systems*, **64**(5):861–914, 2020. doi:10.1007/s00224-019-09930-2.

5 Implementation

```
1 (*package)
2 \NeedsTeXFormat{LaTeX2e}[2005/12/01]
3 \ProvidesPackage{proofgraph}
4 [2026/07/10 v1.0.1 Dependency graph of results]
```

5.1 Dependencies

`amsthm` (loaded for its heading hooks) defines the `proof` environment with `\newenvironment`, which aborts with “Command `\proof` already defined” under a class that already provides one (e.g. the Springer `svjour3` `\proof`). When we are about to load `amsthm` ourselves (it is not already loaded), we save the class’s `proof/\endproof` and free them so `amsthm` loads cleanly, then restore the class’s versions afterwards, so its proof styling is kept (and `proofgraph` still hooks that `proof` environment). If `amsthm` is already loaded, `\RequirePackage` is a no-op and `\proof` is left untouched.

```
5 \ifpackageloaded{amsthm}{-}{%
6   \ifundefined{proof}{-}{%
7     \let\pgph@class@proof\proof \let\pgph@class@endproof\endproof}%
8   \let\proof\relax \let\endproof\relax}
9 \RequirePackage{amsthm}
10 \ifundefined{pgph@class@proof}{-}{%
11   \let\proof\pgph@class@proof \let\endproof\pgph@class@endproof}
12 \RequirePackage{etoolbox}
13 \RequirePackage{xstring}
14 \RequirePackage{kvoptions}
```

`graphicx` is needed only to embed a rendered graph with `\proofgraph`; the core `.dot`-writing machinery does not use it. Rather than burden every document with it, we leave it unloaded and check for `\includegraphics` at the point of use (see `\proofgraph`). When the `hyperlinks` overlay is active `tikz` is loaded, which pulls `graphicx` in anyway.

`\pgph@trimsp` An expandable space-trimmer (labels inside a `\cite/\cref/\uses` list arrive with the space after each comma), built on the `expl3` primitive so it is robust to leading spaces and braces.

```
15 \ExplSyntaxOn
16 \cs_new:Npn \pgph@trimsp #1 { \tl_trim_spaces:n {#1} }
17 \ExplSyntaxOff
```

5.2 Options

```
18 \SetupKeyvalOptions{family=pgph,prefix=pgph@}
19 \DeclareStringOption[remove]{selfloops}
20 \DeclareBoolOption{autorun}
21 \DeclareStringOption[dot]{engine}
22 \DeclareStringOption[pdf]{format}
23 \DeclareStringOption[LR]{rankdir}
24 \DeclareStringOption[usedby]{direction}
25 \DeclareBoolOption{cite}
26 \DeclareStringOption[key]{citelabel}
27 \DeclareBoolOption[true]{hyperlinks}
28 \DeclareStringOption{file}
```

```

29 \ProcessKeyvalOptions*
30 \ifx\pgph@file\@empty\def\pgph@file{\jobname}\fi

```

5.3 Literal braces

We need catcode-12 braces to write the Graphviz digraph { } block.

```

31 \begingroup
32 \catcode'\[=1 \catcode'\]=2
33 \catcode'\{=12 \catcode'\}=12
34 \gdef\pgph@ob[{}]%
35 \gdef\pgph@cb[{}]%
36 \endgroup

```

5.4 Data structures

A global counter gives each result instance a unique numeric id. Nodes and edges are accumulated in list macros and emitted at end of document, so that editing commands (ignore, exclude, self-loops) can be applied to the complete graph regardless of where they appear.

```

37 \newcount\pgph@idcnt
38 \newcount\pgph@citeslotcnt
39 \def\pgph@nodes{}
40 \def\pgph@edges{}
41 \def\pgph@ignores{}
42 \def\pgph@excludes{}
43 \let\pgph@curresult\@empty
44 \let\pgph@curproof\@empty

```

`\pgph@auxmap` The label-to-id map. `\label` inside a result records, both in memory and in the `.aux` file, that the user label maps to the current node id, so references resolve independently of hyperref.

```

45 \newcommand\pgph@auxmap[2]{\global\@namedef{pgph@map@#1}{#2}%
46 \ifcsname pgph@rmap@#2\endcsname\else
47 \global\@namedef{pgph@rmap@#2}{#1}\fi}
48 \newcommand\pgph@setmap[2]{%
49 \edef\pgph@tmpid{#2}%
50 \global\expandafter\edef\csname pgph@map@#1\endcsname{\pgph@tmpid}%
51 \ifcsname pgph@rmap@\pgph@tmpid\endcsname\else
52 \global\expandafter\def\csname pgph@rmap@\pgph@tmpid\endcsname{#1}%
53 \fi
54 \protected@write\@auxout{}{\string\pgph@auxmap{#1}{\pgph@tmpid}}%
55 }

```

5.5 Registering result environments

`\pgph@register` A result environment is tracked by installing a begin-hook that records a node. In every declaration command (`\newtheorem`, `\spnewtheorem`) the environment *name* is the first mandatory argument, so registration does not need to parse the remaining (varied) arguments. Each name met is queued as a *candidate* (deduplicated); the begin-hooks themselves are installed only at `\begin{document}`, once the inclusion and exclusion lists are settled, so the customisation commands below may appear anywhere in the preamble.

```

56 \let\pgph@candidates\@empty
57 \newcommand\pgph@register[1]{%
58   \edef\pgph@thisname{#1}%
59   \IfBeginWith\pgph@thisname{axp@}{-}{-}{%
60     \ifcsname pgph@cand@\pgph@thisname\endcsname\else
61       \global\@namedef{pgph@cand@\pgph@thisname}{-}%
62       \ifx\pgph@candidates\@empty
63         \global\let\pgph@candidates\pgph@thisname
64       \else
65         \xdef\pgph@candidates{\pgph@candidates,\pgph@thisname}%
66       \fi
67     \fi
68   }%
69 }

```

We wrap `\newtheorem` (and, for Springer classes, `\spnewtheorem`) to register each newly declared environment, replaying the original command so its own argument parsing is untouched. The starred (unnumbered) forms are *not* auto-registered: a results graph wants numbered statements, and these forms are commonly used for unnumbered repeats of a result (e.g., the appendix copies produced by `apxproof` and `myproofs`), which would otherwise appear as spurious unnumbered duplicate nodes. Use `\proofgraphtrack` to track an unnumbered environment deliberately.

```

70 \let\pgph@orig@newtheorem\newtheorem
71 \renewcommand\newtheorem{\@ifstar{\pgph@nt@star}{\pgph@nt@plain}}
72 \newcommand\pgph@nt@plain[1]{%
73   \pgph@register{#1}\pgph@orig@newtheorem{#1}}
74 \newcommand\pgph@nt@star[1]{\pgph@orig@newtheorem*{#1}}
75 \ifdefined\spnewtheorem
76   \let\pgph@orig@spnewtheorem\spnewtheorem
77   \renewcommand\spnewtheorem{%
78     \@ifstar{\pgph@spnt@star}{\pgph@spnt@plain}}
79   \newcommand\pgph@spnt@plain[1]{%
80     \pgph@register{#1}\pgph@orig@spnewtheorem{#1}}
81   \newcommand\pgph@spnt@star[1]{\pgph@orig@spnewtheorem*{#1}}
82 \fi

```

`\proofgraphtrack`

`\proofgraphuntrack`

`\proofgraphtrack`

`\proofgraphuntrack`

Result environments are detected automatically, but the analysis can be tuned. `\proofgraphtrack{⟨names⟩}` forces the listed environments to be tracked, overriding any exclusion; `\proofgraphuntrack{⟨names⟩}` excludes the listed environment *types*. Expository environments are excluded by default: `definition`, `example`, `remark`, `note`. Both commands belong in the preamble. `\pgph@defaultenvs` is the list of class-predefined names probed at `\begin{document}`; `\pgph@defaultexclude` seeds the exclusion set.

```

83 \newcommand\pgph@defaultenvs{theorem,lemma,proposition,corollary,%
84   definition,conjecture,claim,fact,remark,example,observation,%
85   notation,problem,question,property,assumption,hypothesis,principle}
86 \newcommand\pgph@defaultexclude{definition,example,remark,note}
87 \newcommand\proofgraphtrack[1]{%
88   \@for\pgph@one:=#1\do{%
89     \csundef{pgph@xtype@\pgph@one}%
90     \global\@namedef{pgph@force@\pgph@one}{-}%

```

```

91   \expandafter\pgph@register\expandafter{\pgph@one}%
92   }%
93 }
94 \newcommand\proofgraphuntrack[1]{%
95   \@for\pgph@one:=#1\do{%
96     \csundef{pgph@force@\pgph@one}%
97     \global\@namedef{pgph@xtype@\pgph@one}{}%
98   }%
99 }
100 \expandafter\proofgraphuntrack\expandafter{\pgph@defaultexclude}

```

Environments predefined by the document class (acmart, lipics, Springer classes) are declared before the package is loaded, so the wrappers above never see them. At `\begin{document}` we add every common result name that exists to the candidate list, then install a begin-hook for each candidate that is forced or not excluded.

```

101 \newcommand\pgph@dohook[1]{%
102   \ifcsname pgph@hooked@#1\endcsname\else
103     \global\@namedef{pgph@hooked@#1}{}%
104     \AtBeginEnvironment{#1}{\pgph@beginresult{#1}}%
105   \fi
106 }
107 \newcommand\pgph@maybehook[1]{%
108   \ifcsname pgph@force@#1\endcsname
109     \pgph@dohook{#1}%
110   \else
111     \ifcsname pgph@xtype@#1\endcsname\else\pgph@dohook{#1}\fi
112   \fi
113 }
114 \newcommand\pgph@resolverregistration[1]{%
115   \@for\pgph@one:=#1\do{%
116     \ifcsname\pgph@one\endcsname%
117       \expandafter\pgph@register\expandafter{\pgph@one}\fi}%
118   \ifx\pgph@candidates\@empty\else
119     \@for\pgph@one:=\pgph@candidates\do{%
120       \expandafter\pgph@maybehook\expandafter{\pgph@one}}%
121   \fi
122 }
123 \AtBeginDocument{%
124   \expandafter\pgph@resolverregistration\expandafter{\pgph@defaultenvs}}

```

`\pgph@beginresult` Start a result: allocate an id and record the node. The node is keyed and styled by `\pgph@hooknames` the environment name (third argument, also a fallback label), but displayed by the `\pgph@recordresult` result's *formatted* name (“Theorem”, “Lemma”), captured from the heading. We get that name by locally wrapping the heading macros, whose first two arguments are the formatted name and the number. Which macro carries the *optional note* (where a restatement’s cross-reference lives) depends on the setup: `amsthm`, which `proofgraph` always loads, routes every heading through `\@begintheorem`, defined as `#1#2[#3]` with the note as the bracketed `#3` (and leaves `\@opargbegintheorem` `\relax`); the L^AT_EX kernel instead uses a two-argument `\@begintheorem` with a separate three-argument `\@opargbegintheorem` for the note; the Springer classes use `\@spbtheorem/\@spopargbegintheorem`. We detect the `amsthm` form (`\@opargbegintheorem` is `\relax`) and read the bracketed note there, otherwise

we hook `\@opargbegintheorem`; either way the note goes to `\pgph@scanrefs`. These redefinitions are confined to the current environment group, so they revert at `\end`. We also locally redefine `\label` to populate the map.

Nodes are accumulated in *reverse* order of appearance (`\pgph@prependnode`). With `rankdir=LR`, `Graphviz` stacks disconnected components vertically in the order they are read, from the bottom up; emitting the results last-to-first therefore lays them out top-to-bottom in document order (first result on top), while connected results stay grouped. This is a layout tendency, not a guarantee, but needs no `pack` trickery.

```

125 \newcommand\pgph@prependnode[1]{%
126   \xdef\pgph@nodes{\unexpanded{#1}\unexpanded\expandafter{\pgph@nodes}}%
127 }
128 \newcommand\pgph@beginresult[1]{%
129   \global\advance\pgph@idcnt\one
130   \xdef\pgph@curresult{\the\pgph@idcnt}%
131   \protected@edef\pgph@tmp{%
132     \noexpand\pgph@node{\pgph@curresult}{#1}{#1}}%
133   \expandafter\pgph@prependnode\expandafter{\pgph@tmp}%
134   \pgph@hooknames
135   \ifx\label\pgph@maplabel\else\let\pgph@savedlabel\label\fi
136   \let\label\pgph@maplabel
137 }
138 \newcommand\pgph@hooknames{%
139   \ifdefined\@begintheorem
140     \let\pgph@orig@bt\@begintheorem
141     \ifx\@opargbegintheorem\relax
142       \def\@begintheorem##1##2[##3]{%
143         \pgph@recordresult{##1}{##2}\pgph@scanrefs{##3}%
144         \pgph@orig@bt{##1}{##2}[##3]}%
145     \else
146       \def\@begintheorem##1##2{%
147         \pgph@recordresult{##1}{##2}\pgph@orig@bt{##1}{##2}}%
148     \fi
149   \fi
150   \ifdefined\@opargbegintheorem\ifx\@opargbegintheorem\relax\else
151     \let\pgph@orig@obt\@opargbegintheorem
152     \def\@opargbegintheorem##1##2##3{%
153       \pgph@recordresult{##1}{##2}\pgph@scanrefs{##3}%
154       \pgph@orig@obt{##1}{##2}{##3}}%
155   \fi\fi
156   \ifdefined\@spbegintheorem
157     \let\pgph@orig@spbt\@spbegintheorem
158     \def\@spbegintheorem##1##2##3##4{%
159       \pgph@recordresult{##1}{##2}\pgph@orig@spbt{##1}{##2}{##3}{##4}}%
160   \fi
161   \ifdefined\@spopargbegintheorem
162     \let\pgph@orig@spobt\@spopargbegintheorem
163     \def\@spopargbegintheorem##1##2##3##4##5{%
164       \pgph@recordresult{##1}{##2}\pgph@scanrefs{##3}%
165       \pgph@orig@spobt{##1}{##2}{##3}{##4}{##5}}%
166   \fi
167 }

```

`\pgph@scanrefs` A result whose optional title carries a cross-reference, e.g., a `result` whose title is `[(Corollary~\ref{cor:x})]`, is understood to restate or build on that result: we record an edge from the current result to each label referenced in the title. We capture these by typesetting the title once into a discarded box with the reference-capturing commands installed and `\pgph@curproof` pointing at the current result; `\pgph@curproof` is saved and restored, so this affects nothing outside the title.

```

168 \newcommand\pgph@scanrefs[1]{%
169   \ifx\pgph@curresult\@empty\else
170     \begingroup
171       \global\let\pgph@savedit\pgph@curproof\pgph@curproof
172       \global\let\pgph@curproof\pgph@curresult
173       \pgph@install{ref}\pgph@install{cref}\pgph@install{Cref}%
174       \pgph@install{autoref}\pgph@install{eqref}\pgph@install{vref}%
175       \setbox\z@\hbox{#1}%
176       \global\let\pgph@curproof\pgph@savedit\pgph@curproof
177     \endgroup
178   \fi
179 }
```

The formatted name is the first argument of every heading macro (`\@begintheorem/\@opargbegintheorem` for `amsthm/newtheorem`, `\@spbegintheorem/\@spopargbegintheorem` for Springer's `\spnewtheorem`); the second argument is the number. We use that second argument only to tell *whether* the result is numbered (it is empty for the starred, unnumbered forms): an unnumbered result records no number, so it is dropped at emission unless tracked explicitly. The number *value*, when present, is read from `\@currentlabel` rather than from that argument, which some classes pollute (e.g., `lipics` injects `\advance\par@deathcycles\@one`). `\@currentlabel`, set by the result's `\refstepcounter` just before the heading, is the clean number and needs no `\label`. Values are reduced to plain strings now, with fragile wrappers neutralized (notably `apxproof`'s forward-link). The first heading wins (guarded on the name), ignoring nested headings. The emptiness of the number argument is tested without expanding it, so a polluted (but non-empty) number does not trip the test.

```

180 \newcommand\pgph@recordresult[2]{%
181   \ifcsname pgph@name@\pgph@curresult\endcsname\else
182     \begingroup
183       \let\protect\@empty
184       \def\axp@forward@link##1##2{##2}%
185       \edef\pgph@tmpname{#1}%
186       \global\expandafter\let\csname pgph@name@\pgph@curresult\endcsname
187         \pgph@tmpname
188       \global\@namedef{pgph@seen@\pgph@curresult}{}%
189       \def\pgph@tmpn{#2}%
190       \ifx\pgph@tmpn\@empty\else
191         \edef\pgph@tmpnum{\@currentlabel}%
192         \global\expandafter\let\csname pgph@num@\pgph@curresult%
193           \endcsname\pgph@tmpnum
194       \fi
195     \endgroup
196   \fi
197 }
```

`\pgph@maplabel` records the label-to-node mapping. As a fallback for classes whose headings pass through none of the hooked heading macros (so no heading was `seen`), it also captures the number from `\@currentlabel`. We do *not* do this once a heading was seen: there, an absent number means the result is genuinely unnumbered (a starred form), and `\@currentlabel` could be a stale value from an earlier result. The value is expanded to a plain string *now*, inside a group where fragile wrappers are neutralized, so that nothing dangerous survives to the emission pass: in particular, under `apxproof` forward-linking, `\@currentlabel` holds `\axp@forward@link{(target)}{<number>}`, whose hyperlink machinery must never be expanded in a non-typesetting context (it would run away). Reducing it to its number argument keeps it harmless.

```

\pgph@maplabel
198 \newcommand\pgph@maplabel[1]{%
199   \pgph@setmap{#1}{\pgph@curresult}%
200   \ifcsname pgph@num@\pgph@curresult\endcsname\else
201     \ifcsname pgph@seen@\pgph@curresult\endcsname\else
202       \begingroup
203         \let\protect\@empty
204         \def\axp@forward@link##1##2{##2}%
205         \edef\pgph@tmpnum{\@currentlabel}%
206         \global\expandafter\let\csname pgph@num@\pgph@curresult%
207           \endcsname\pgph@tmpnum
208       \endgroup
209     \fi
210   \fi
211   \pgph@savlabel{#1}%
212 }

```

5.6 Hooking proofs

Inside a proof we redefine the reference commands to capture edges from the proved result.

```

213 \newcommand\pgph@adddedge[2]{%
214   \protected@edef\pgph@tmpedge{\noexpand\pgph@edge{#1}{#2}}%
215   \expandafter\g@addto@macro\expandafter\pgph@edges%
216   \expandafter{\pgph@tmpedge}%
217 }
218 \newcommand\pgph@recordref[1]{%
219   \ifx\pgph@curproof\@empty\else
220     \@for\pgph@one:=#1\do{%
221       \edef\pgph@k{\expandafter\pgph@trimsp\expandafter{\pgph@one}}%
222       \pgph@adddedge{\pgph@curproof}{\pgph@k}}%
223   \fi
224 }

```

`\pgph@beginproof` Attach the proof to the current result and install the capturing versions of the reference commands (only those that exist).

The capturing commands are `\protected` so that, when a reference appears in a moving argument inside a proof (e.g., a `\caption`), they do not expand into the `.aux/.lof` and leak internal tokens.

```

225 \protected\def\pgph@cap@ref#1{\pgph@recordref{#1}\pgph@sav@ref{#1}}

```

```

226 \protected\def\pgph@cap@cref#1{\pgph@recordref{#1}\pgph@saved@cref{#1}}
227 \protected\def\pgph@cap@Cref#1{\pgph@recordref{#1}\pgph@saved@Cref{#1}}
228 \protected\def\pgph@cap@autoref#1{%
229   \pgph@recordref{#1}\pgph@saved@autoref{#1}}
230 \protected\def\pgph@cap@eqref#1{%
231   \pgph@recordref{#1}\pgph@saved@eqref{#1}}
232 \protected\def\pgph@cap@vref#1{\pgph@recordref{#1}\pgph@saved@vref{#1}}
233 \newcommand\pgph@install[1]{%
234   \ifcsname #1\endcsname
235     \expandafter\ifx\csname #1%
236       \expandafter\endcsname\csname pgph@cap@#1\endcsname
237     \else
238       \expandafter\let\csname pgph@saved@#1%
239         \expandafter\endcsname\csname #1\endcsname
240       \expandafter\let\csname #1%
241         \expandafter\endcsname\csname pgph@cap@#1\endcsname
242     \fi
243 \fi
244 }
245 \newcommand\pgph@beginproof{%
246   \let\pgph@curproof\pgph@curresult
247   \pgph@install{ref}\pgph@install{cref}\pgph@install{Cref}%
248   \pgph@install{autoref}\pgph@install{eqref}\pgph@install{vref}%
249   \ifpgph@cite\pgph@installcite\fi
250   \pgph@hookproofhead
251 }

```

When the proof environment is itself a theorem-like environment (as with Springer's `\spnewtheorem*{proof}`), its optional title is set by `\@opargbegintheorem/\@spopargbegintheorem`. We wrap these for the duration of the proof so that a title such as `[of Theorem~\ref{thm:x}]` sets `\proofof{thm:x}`, attributing the proof to its result.

```

252 \newcommand\pgph@hookproofhead{%
253   \ifdefined\@opargbegintheorem
254     \let\pgph@orig@obtp\@opargbegintheorem
255     \def\@opargbegintheorem##1##2##3{%
256       \pgph@scanproofarg{##3}\pgph@orig@obtp{##1}{##2}{##3}}%
257   \fi
258   \ifdefined\@spopargbegintheorem
259     \let\pgph@orig@spobtp\@spopargbegintheorem
260     \def\@spopargbegintheorem##1##2##3##4##5{%
261       \pgph@scanproofarg{##3}%
262       \pgph@orig@spobtp{##1}{##2}{##3}{##4}{##5}}%
263   \fi
264   \ifdefined\@Opargbegintheorem
265     \let\pgph@orig@Obtp\@Opargbegintheorem
266     \def\@Opargbegintheorem##1##2##3##4{%
267       \pgph@scanproofarg{##2}\pgph@orig@Obtp{##1}{##2}{##3}{##4}}%
268   \fi
269 }

```

`\pgph@scanproofarg` A proof whose optional title names its result (a detached proof) has its result given by that title, e.g., `thm:x` for an optional argument `[of Theorem~\ref{thm:x}]`, rather than by the most recently opened result. We capture this by wrapping the

proof environment so that, when an optional title is given, the cross-reference it contains is turned into a `\proofof` (the title is typeset once into a discarded box with `\ref` and friends redefined). Without an optional title the original environment is used unchanged.

```

270 \newcommand\pgph@scanproofarg[1]{%
271   \begingroup
272     \let\protect\@empty
273     \def\ref##1{\proofof{##1}}\def\cref##1{\proofof{##1}}%
274     \def\Cref##1{\proofof{##1}}\def\autoref##1{\proofof{##1}}%
275     \setbox\z@\hbox{##1}%
276   \endgroup
277 }

```

To capture the optional title of a detached `amsthm` proof (an optional [of Theorem~\ref{thm:x}]) we redefine the outer `\proof` to read the optional argument and delegate to the original inner macro (saved as `\pgph@proofinner`), so no recursion occurs. We must do this *only* for a genuine optional-argument environment: a package that captures the proof body verbatim (`apxproof`, `myproofs`) redefines `proof` with no optional argument, and wrapping it would break that capture. We recognise the optional-argument form by the `\@protected@ttestopt` in its definition. When `proof` is instead a theorem-like environment (`\spnewtheorem*{proof}`) its title is already handled by `\pgph@hookproofhead`, so here we just install the begin-hook.

```

278 \newcommand\pgph@wrapproof{%
279   \ifdefined\proof
280     \edef\pgph@proofmeaning{\meaning\proof}%
281     \edef\pgph@ttestoptstr{\detokenize{\@protected@ttestopt}}%
282     \IfSubStr\pgph@proofmeaning\pgph@ttestoptstr
283       {\pgph@redefproof}%
284     {\AtBeginEnvironment{proof}{\pgph@beginproof}}%
285   \fi
286 }
287 \newcommand\pgph@redefproof{%
288   \expandafter\let\expandafter\pgph@proofinner%
289   \csname\string\proof\endcsname
290   \def\proof{\pgph@beginproof
291     \@ifnextchar[{\pgph@proofopt}{\pgph@proofinner[\proofname]}}%
292   \def\pgph@proofopt[##1]{\pgph@scanproofarg{##1}\pgph@proofinner[##1]}%
293 }

```

Packages such as `apxproof` capture the body of a proof verbatim (to defer it to the appendix) and replay it there. Patching the `proof` environment then corrupts that capture, and the references are only executed at replay time anyway. So we wrap `proof` only when no such package is active; otherwise the deferred-proof capture (below) takes over. This happens at `\begin{document}`, once we know what is loaded.

```

294 \newif\ifpgph@apxproof
295 \AtBeginDocument{%
296   \@ifpackageloaded{apxproof}%
297     {\pgph@apxprooftrue\pgph@apxinit}%
298     {\pgph@wrapproof}%
299 }

```

`\pgph@apxinit` Under `apxproof`, deferred proofs are typeset in the appendix, where their references finally execute. `apxproof` fires `\apxproofhook` inside each such proof, passing the `axp@r<n>` label of the proved theorem, which (because `apxproof` also attaches that label to the theorem in the main text) is already in our map. We resolve it to the source node and install the reference-capturing commands for the duration of the proof.

```

300 \newcommand\pgph@apxinit{%
301   \def\apxproofhook##1{\pgph@apxproofcapture{##1}}
302 \newcommand\pgph@apxproofcapture[1]{%
303   \edef\pgph@curproof{\pgph@resolve{#1}}%
304   \ifx\pgph@curproof\@empty\else
305     \pgph@install{ref}\pgph@install{cref}\pgph@install{Cref}%
306     \pgph@install{autoref}\pgph@install{eqref}\pgph@install{vref}%
307     \ifpgph@cite\pgph@installcite\fi
308   \fi
309 }

```

`\pgph@hookcite` Optional citation capture, applied to `\cite` and, when they are defined (`natbib`, `biblatex`), to `\citet` and `\citep` as well, all sharing one capture chain parameterized by the command name: external references become `cite:-`prefixed target labels, drawn later as distinct nodes. The node identity is the pair (key, note): a `\cite[<note>]{<key>}` inside a proof is distinguished from a `\cite` of the same key with a *different* note, so that citing two different results of the same work (say two theorems of one book) yields two nodes rather than one mislabelled with whichever note was seen first. Each pair is assigned a numeric *slot* (`\pgph@citeslot`); the edge target is `cite:<slot>`, and the slot remembers the key (for the tag and the bibliography hyperlink) and the note (for the label). The note is reduced to a plain string at capture time (fragile wrappers neutralized, `~` normalised to a space), which also serves as the signature so `[Thm 1]` and `[Thm~1]` coincide. We peek for the optional argument with `\@ifnextchar` rather than declaring one, so that a note-less `\cite{<key>}` is passed on *as written*: re-emitting it as `\cite[]{<key>}` would make the typeset citation grow a spurious empty note (“[1,]”). A second optional argument is peeked for in the same way, to support the two-argument form `\cite[<pre-note>][<note>]{<key>}` of `natbib` and `biblatex`; there the *second* argument is the note recorded for the node (matching the one-argument semantics, where the lone argument is the post-note), while the pre-note (“see”, “e.g.”) qualifies the act of citing rather than identifying the cited result, and is only replayed to the saved `\cite`. Without this peek, the mandatory-argument scan would grab the bare token `[` as the key and typeset the rest of the citation as text, giving confusing errors (“Missing \$ inserted”) deep in the proof body. A star (`\citet*`, `\citep*`: full author list) is likewise peeked for and replayed; it affects rendering only, not the recorded (key, note) pair. The replay head (saved command plus optional star) is frozen in `\pgph@replay` by the star-peek before the arguments are scanned. As in `\pgph@install`, installation is skipped (per command) when the command is already the capture macro; otherwise a proof nested inside another proof would save the capture as the saved command, and the first citation in the inner proof would recurse forever.

```

310 \newcommand\pgph@gxdefcs[2]{%
311   \global\expandafter\edef\csname#1\endcsname{#2}}
312 \newcommand\pgph@cap@cite{\pgph@cap@citegen{cite}}
313 \newcommand\pgph@cap@citet{\pgph@cap@citegen{citet}}

```

```

314 \newcommand\pgph@cap@citep{\pgph@cap@citegen{citep}}
315 \newcommand\pgph@cap@citegen[1]{%
316   \def\pgph@citecmd{#1}%
317   \@ifstar{\pgph@cap@cite@x*}{\pgph@cap@cite@x{}}
318 \def\pgph@cap@cite@x#1{%
319   \edef\pgph@replay{%
320     \expandafter\noexpand\csname pgph@saved@\pgph@citecmd\endcsname#1}%
321   \@ifnextchar[\pgph@cap@cite@opt\pgph@cap@cite@noopt}
322 \def\pgph@cap@cite@opt[#1]{%
323   \@ifnextchar[{\pgph@cap@cite@optii{#1}}{\pgph@cap@cite@opti{#1}}
324 \def\pgph@cap@cite@opti#1#2{%
325   \pgph@cap@citerec{#1}{#2}\pgph@replay[#1]{#2}}
326 \def\pgph@cap@cite@optii#1[#2]#3{%
327   \pgph@cap@citerec{#2}{#3}\pgph@replay[#1][#2]{#3}}
328 \def\pgph@cap@cite@noopt#1{\pgph@cap@citerec{#1}\pgph@replay{#1}}
329 \newcommand\pgph@cap@citerec[2]{%
330   \ifx\pgph@curproof\empty\else
331     \@for\pgph@one:=#2\do{%
332       \edef\pgph@k{\expandafter\pgph@trimsp\expandafter{\pgph@one}}%
333       \pgph@citeslot{\pgph@k}{#1}%
334       \pgph@adddedge{\pgph@curproof}{cite:\pgph@curslot}}%
335   \fi
336 }
337 \newcommand\pgph@citeslot[2]{%
338   \begingroup
339     \pgph@citesanitize
340     \xdef\pgph@gnotestr{#2}%
341   \endgroup
342   \edef\pgph@sig{#1@\detokenize\expandafter{\pgph@gnotestr}}%
343   \ifcsname pgph@cslot@\pgph@sig\endcsname
344     \edef\pgph@curslot{\csname pgph@cslot@\pgph@sig\endcsname}%
345   \else
346     \global\advance\pgph@citeslotcnt\@ne
347     \edef\pgph@curslot{\the\pgph@citeslotcnt}%
348     \pgph@gxdefcs{pgph@cslot@\pgph@sig}{\pgph@curslot}%
349     \pgph@gxdefcs{pgph@cskey@\pgph@curslot}{#1}%
350     \ifx\pgph@gnotestr\empty\else
351       \pgph@gxdefcs{pgph@citenote@\pgph@curslot}{\pgph@gnotestr}%
352     \fi
353   \fi
354 }
355 \newcommand\pgph@installcite{%
356   \pgph@installciteone{cite}%
357   \pgph@installciteone{citet}%
358   \pgph@installciteone{citep}%
359 }
360 \newcommand\pgph@installciteone[1]{%
361   \ifcsname #1\endcsname
362     \expandafter\ifx\csname #1\expandafter\endcsname
363       \csname pgph@cap@#1\endcsname
364     \else
365       \expandafter\let\csname pgph@saved@#1\expandafter\endcsname
366         \csname #1\endcsname
367       \expandafter\let\csname #1\expandafter\endcsname

```

```

368             \csname pgph@cap@#1\endcsname
369     \fi
370 \fi
371 }

```

5.7 User commands

```

372 \newcommand\uses[1]{\pgph@recordref{#1}}
373 \newcommand\proofof[1]{%
374   \ifcsname pgph@map@#1\endcsname
375     \xdef\pgph@curproof{\csname pgph@map@#1\endcsname}%
376   \fi
377 }
378 \newcommand\proofgraphedge[2]{%
379   \g@addto@macro\pgph@edges{\pgph@labeledge{#1}{#2}}%
380 }
381 \newcommand\pgph@labeledge[2]{%
382   \edef\pgph@ef{\pgph@resolve{#1}}%
383   \ifx\pgph@ef\@empty\else
384     \@for\pgph@one:=#2\do{%
385       \edef\pgph@k{\expandafter\pgph@trimsp\expandafter{\pgph@one}}%
386       \pgph@plainedge{\pgph@ef}{\pgph@k}}%
387   \fi
388 }
389 \newcommand\proofgraphignore[2]{%
390   \g@addto@macro\pgph@ignores{\pgph@ignorerule{#1}{#2}}%
391 }
392 \newcommand\proofgraphexclude[1]{%
393   \g@addto@macro\pgph@excludes{\pgph@excluderule{#1}}%
394 }
395 \newcommand\proofgraphstyle[2]{\global\@namedef{pgph@style@#1}{#2}}

```

External citation nodes (option cite) share one style, held in `\pgph@citestyle` and overridable with `\proofgraphstylecite`; it defaults to `shape=note`.

```

396 \newcommand\pgph@citestyle{shape=note}
397 \newcommand\proofgraphstylecite[1]{\renewcommand\pgph@citestyle{#1}}

```

5.8 Resolution helpers

These run at end of document, after the `.aux` has populated the map.

```

398 \newcommand\pgph@resolve[1]{%
399   \ifcsname pgph@map@#1\endcsname\csname pgph@map@#1\endcsname\fi
400 }
401 \newcommand\pgph@ignorerule[2]{%
402   \edef\pgph@f{\pgph@resolve{#1}}\edef\pgph@t{\pgph@resolve{#2}}%
403   \ifx\pgph@f\@empty\else\ifx\pgph@t\@empty\else
404     \@namedef{pgph@ig@\pgph@f @\pgph@t}{}%
405   \fi\fi
406 }
407 \newcommand\pgph@excluderule[1]{%
408   \edef\pgph@x{\pgph@resolve{#1}}%
409   \ifx\pgph@x\@empty\else\@namedef{pgph@ex@\pgph@x}{}\fi
410 }

```

5.9 Emission

`\pgph@dotesc` Escape the two characters that are special in a Graphviz double-quoted string, so a theorem name, number or citation note containing them does not corrupt the `.dot`: a backslash is doubled and a double quote is backslash-escaped (in that order). `\pgph@bschar` and `\pgph@dqchar` hold the catcode-12 backslash and double quote to search for; `xstring`'s expansion mode is saved and restored around the two substitutions.

```

411 \begingroup
412 \catcode'\|=0 \catcode\'"=12 \catcode'\|=12
413 \gdef\pgph@bschar{\}%
414 \gdef\pgph@dqchar{"}%
415 \endgroup
416 \newcommand\pgph@dotesc[1]{%
417 \saveexpandmode\expandarg
418 \StrSubstitute{#1}{\pgph@bschar}{\pgph@bschar\pgph@bschar}[#1]%
419 \StrSubstitute{#1}{\pgph@dqchar}{\pgph@bschar\pgph@dqchar}[#1]%
420 \restoreexpandmode}

```

`\pgph@node` Emit one node line, honouring exclusions and per-environment styling. External (`cite:`) targets are emitted separately as a pass over edges. An *unnumbered* result (no number captured) is omitted: a results graph is about numbered statements, and unnumbered theorem-like environments are typically expository or repeated copies (e.g., a class-predefined `\spnewtheorem*{claim}`). `\proofgraphtrack{<env>}` overrides this, so a deliberately unnumbered environment is still drawn.

```

421 \newcommand\pgph@node[3]{%
422 \ifcsname pgph@ex@#1\endcsname\else
423 \edef\pgph@nm{\ifcsname pgph@num@#1\endcsname%
424 \csname pgph@num@#1\endcsname\fi}%
425 \ifx\pgph@nm\@empty
426 \ifcsname pgph@force@#2\endcsname
427 \pgph@emitnode{#1}{#2}{#3}%
428 \else
429 \@namedef{pgph@ex@#1}{}%
430 \fi
431 \else
432 \pgph@emitnode{#1}{#2}{#3}%
433 \fi
434 \fi
435 }
436 \newcommand\pgph@emitnode[3]{%
437 \edef\pgph@s{\ifcsname pgph@style@#2\endcsname
438 , \csname pgph@style@#2\endcsname\fi}%
439 \edef\pgph@dn{\ifcsname pgph@name@#1\endcsname
440 \csname pgph@name@#1\endcsname\else#3\fi}%
441 \edef\pgph@lbl{\pgph@dn\space\pgph@nm}%
442 \pgph@dotesc\pgph@lbl
443 \immediate\write\pgph@out{%
444 \space\space"n#1" [label="\pgph@lbl"\pgph@s];}%
445 \ifcsname pgph@rmap@#1\endcsname
446 \immediate\write\pgph@mapout{%
447 \string\pgph@nodemap{n#1}{\csname pgph@rmap@#1\endcsname}}%

```

```
448 \fi
449 }
```

`\pgph@edge` Emit one edge, resolving the target label to an id and applying the self-loop, ignore and exclude filters. Targets beginning with `cite:` denote external nodes.

```
450 \newcommand\pgph@edge[2]{%
451 \IfBeginWith{#2}{cite:}%
452 {\pgph@citeedge{#1}{#2}}%
453 {\pgph@plainedge{#1}{#2}}%
454 }
455 \newcommand\pgph@plainedge[2]{%
456 \edef\pgph@t{\pgph@resolve{#2}}%
457 \ifx\pgph@t\@empty\else
458 \ifcsname pgph@ex@#1\endcsname\else
459 \ifcsname pgph@ex@\pgph@t\endcsname\else
460 \ifcsname pgph@ig@#1\pgph@t\endcsname\else
461 \ifboolexpr{ test {\ifnumequal{#1}{\pgph@t}}
462 and test {\ifdefstring{\pgph@selfloops}{remove}} }%
463 {\pgph@emitedge{#1}{\pgph@t}}%
464 \fi\fi\fi
465 \fi
466 }
```

`\pgph@emitedge` The relation recorded is always “the result whose proof we are in (the first argument) uses the target (the second)”. The `direction` option chooses the arrow orientation: `direction=usedby` (default) draws *target to result*, so an arrow `a→b` reads “a is used by b” (i.e. b relies on a); `direction=uses` draws *result to target* (“a uses b”). The filters above work on the recorded relation, so they are unaffected by the chosen orientation. `\pgph@writeedge` deduplicates.

```
467 \newcommand\pgph@emitedge[2]{%
468 \ifdefstring{\pgph@direction}{uses}%
469 {\pgph@writeedge{#1}{#2}}%
470 {\pgph@writeedge{#2}{#1}}%
471 }
472 \newcommand\pgph@writeedge[2]{%
473 \ifcsname pgph@drawn@#1@#2\endcsname\else
474 \@namedef{pgph@drawn@#1@#2}{}%
475 \immediate\write\pgph@out{\space\space"#1" -> "#2";}%
476 \fi
477 }
```

`\pgph@citeedge` A `cite:⟨slot⟩ target:` recover the slot’s `BIBTEX` key, declare the external node once per slot, then draw the edge. The node *label* is built by `\pgph@setcitelabel`: it is the key by default, or, with `citelabel=tag`, the printed citation tag (the “[42]” / “[Knu74]” text) recovered from the `\b@⟨key⟩` macro that `\bibcite` writes to the `.aux` (so two runs are needed, and it falls back to the key when no tag is known, e.g., on the first run or with citation packages that do not populate `\b@⟨key⟩`). The slot’s `\cite` note, when present, is prepended as “⟨note⟩, ...”. The node-to-label map records the *key*, so all slots of one work hyperlink to the same bibliography entry.

```
478 \newcommand\pgph@citeedge[2]{%
479 \ifcsname pgph@ex@#1\endcsname\else
480 \StrBehind{#2}{cite:}[\pgph@slot]%
```

```

481 \edef\pgph@key{\csname pgph@cskey@\pgph@slot\endcsname}%
482 \ifcsname pgph@extnode@\pgph@slot\endcsname\else
483 \namedef{pgph@extnode@\pgph@slot}{}%
484 \expandafter\pgph@setcitelabel\expandafter{\pgph@slot}%
485 \pgph@dotes\pgph@clabel
486 \immediate\write\pgph@out{%
487 \space\space" c@\pgph@slot"
488 [label="\pgph@clabel",\pgph@citestyle];}%
489 \immediate\write\pgph@mapout{%
490 \string\pgphnodemapcite{c@\pgph@slot}{\pgph@key}}%
491 \fi
492 \pgph@emittedge{n#1}{c@\pgph@slot}%
493 \fi
494 }
495 \edef\pgph@obrace{\expandafter\@gobble\string\{ }
496 \newif\ifpgph@natnum
497 \newcommand\pgph@nil{ }

```

\pgph@citesanitize neutralizes the wrappers and spacing cruft that citation labels carry, so that an \edef of \b@{key} yields plain text: hyperref's \hyper@@link (keep its printed last argument) and the boxing/penalty/\ignorespaces noise that natbib bakes into author lists.

```

498 \newcommand\pgph@citesanitize{%
499 \let\protect\@empty
500 \def\hyper@@link[##1]##2##3##4{##4}%
501 \let\unskip\@empty \let\ignorespaces\@empty \let\unhbox\@empty
502 \let\penalty\@gobble \let\@M\@empty \let\voidb@x\@empty
503 \let\nobreakspace\space \let~\space
504 \def\hbox##1{##1}\def\mbox##1{##1}%
505 }

```

A natbib \b@{key} is {\num}{year}{short}{long}. In numbers mode the tag is the number; in author–year mode it is the short author list and the year.

```

506 \def\pgph@natpick#1#2#3#4\pgph@nil{%
507 \ifpgph@natnum\def\pgph@x{#1}\else\def\pgph@x{#3 #2}\fi}
508 \newcommand\pgph@natdo{\expandafter\pgph@natpick\pgph@x}{\pgph@nil}
509 \newcommand\pgph@setcitelabel[1]{%
510 \edef\pgph@ckey{\csname pgph@cskey@#1\endcsname}%
511 \edef\pgph@cbase{\pgph@ckey}%
512 \ifdefstring{\pgph@citelabel}{tag}{%
513 \ifcsname b@\pgph@ckey\endcsname
514 \global\let\pgph@cbaseg\relax
515 \begingroup
516 \pgph@citesanitize
517 \edef\pgph@x{\csname b@\pgph@ckey\endcsname}%
518 \edef\pgph@dx{\detokenize\expandafter{\pgph@x}}%
519 \IfBeginWith\pgph@dx\pgph@obrace{%
520 \pgph@natnumfalse
521 \@ifundefined{ifNAT@numbers}{}{%
522 \expandafter\ifx\csname ifNAT@numbers\endcsname\iftrue
523 \pgph@natnumtrue\fi}%
524 \pgph@natdo
525 \edef\pgph@dx{\detokenize\expandafter{\pgph@x}}%
526 }{ }%
527 \ifx\pgph@x\@empty\else

```

```

528     \IfSubStr\pgph@dx\@backslashchar{}-{}%
529     \global\let\pgph@cbaseg\pgph@x}%
530     \fi
531   \endgroup
532   \ifx\pgph@cbaseg\relax\else\edef\pgph@cbase{[\pgph@cbaseg]}\fi
533   \fi
534 }{}%
535 \edef\pgph@clabel{%
536   \ifcsname pgph@citenote@#1\endcsname
537     \csname pgph@citenote@#1\endcsname, \fi
538   \pgph@cbase}%
539 }

```

`\pgph@emit` Open the file, resolve editing rules, write nodes then edges, close, and optionally run Graphviz.

```

540 \newwrite\pgph@out
541 \newwrite\pgph@mapout
542 \newcommand\pgph@emit{%
543   \immediate\openout\pgph@out=\pgph@file.dot\relax
544   \immediate\openout\pgph@mapout=\pgph@file-proofgraph.map\relax
545   \immediate\write\pgph@out{digraph proofgraph \pgph@ob}%
546   \immediate\write\pgph@out{\space\space rankdir="\pgph@rankdir";}%
547   \begingroup
548     \pgph@ignores
549     \pgph@excludes
550     \pgph@nodes
551     \pgph@edges
552   \endgroup
553   \immediate\write\pgph@out{\pgph@cb}%
554   \immediate\closeout\pgph@out
555   \immediate\closeout\pgph@mapout
556   \ifpgph@autorun
557     \immediate\write18{\pgph@engine\space -T\pgph@format\space
558       \pgph@file.dot -o \pgph@file-proofgraph.\pgph@format}%
559     \immediate\write18{\pgph@engine\space -Tplain\space
560       \pgph@file.dot -o \pgph@file-proofgraph.plain}%
561   \fi
562 }

```

The emission is registered at `\begin{document}` rather than at load time so that it is added to the end-document hook *after* packages loaded later (notably `apxproof`, whose appendix, with the deferred proofs and their references, is built at end of document). This way the graph is written only once that material has been typeset, and the emission does not disturb the verbatim replay of deferred proofs.

```
563 \AtBeginDocument{\AtEndDocument{\pgph@emit}}
```

If TikZ is available, load its `calc` library now (at `\begin{document}`, where category codes are normal) so that the hyperlink overlay in `\proofgraph` can use it without reading a file mid-document.

```
564 \AtBeginDocument{\ifpackageloaded{tikz}{\usetikzlibrary{calc}}{}}
```

`\pgphnodemap` One entry of the node-to-target map written by `\pgph@emit`: it records, for a `\pgphnodemapcite` Graphviz node identifier, what the node should link to. `\pgphnodemap` records the user *label* of the result a node represents; `\pgphnodemapcite` records the

BIB_TE_X *key* of an external citation node, so that (when the cited work is in the same document) the node can link to its bibliography entry, `hyperref's cite.<key>` destination. The map is read back when the graph is embedded, to turn each node into a hyperlink.

```
565 \newcommand\pgphnodemap[2]{\global\@namedef{pgph@lbl@#1}{#2}}
566 \newcommand\pgphnodemapcite[2]{\global\@namedef{pgph@cbl@#1}{#2}}
```

`\proofgraph` Include the rendered graph (from a previous run) in `autorun` mode. When `hyperref` and `TikZ` are both available (and `hyperlinks` is not switched off), each node is made into an internal hyperlink to what it represents: a result node links to the result (its `\label`), and an external citation node (from the `cite` option) links to the cited work's bibliography entry, `hyperref's cite.<key>` destination, when that work is cited in the same document. `Graphviz` produces a `.pdf` whose link annotations would be lost by `\includegraphics` (they sit inside a form `XOBJECT`), so instead we overlay transparent link areas at the node positions, which we read from the `-Tplain` output. Without those packages we fall back to a plain inclusion.

```
567 \newif\ifpgph@canlink
568 \newif\ifpgph@link
569 \newsavebox\pgph@imgbox
570 \newsavebox\pgph@natbox
571 \newread\pgph@plainin
572 \def\pgph@stop{}
573 \newcommand\proofgraph[1][ ]{%
574   \@ifundefined{includegraphics}%
575     {\PackageError{proofgraph}{\string\proofgraph\space requires the
576       graphicx package}{Add \string\usepackage{graphicx} to your preamble
577       to embed the rendered graph.}}%
578     {\IfFileExists{\pgph@file-proofgraph.\pgph@format}%
579       {\pgph@includegraph{#1}}%
580       {\PackageWarning{proofgraph}{Rendered graph
581         \pgph@file-proofgraph.\pgph@format\space not found; compile with
582         shell-escape and the autorun option, then re-run}}}%
583 }
584 \newcommand\pgph@includegraph[1]{%
585   \pgph@canlinkfalse
586   \ifpgph@hyperlinks
587     \@ifpackageloaded{hyperref}%
588       {\@ifpackageloaded{tikz}%
589         {\IfFileExists{\pgph@file-proofgraph.plain}%
590           {\pgph@canlinktrue}{}}}{}}}%
591   \fi
592   \ifpgph@canlink
593     \pgph@graphlinked{#1}%
594   \else
595     \includegraphics[#1]{\pgph@file-proofgraph.\pgph@format}%
596   \fi
597 }
```

The linked version. We read the node-to-target map under safe category codes (a label may contain `_` or, under `babel`, an active `:`), measure the rendered image, place it in a `TikZ` node, then read the `-Tplain` file and drop one transparent link box on each mapped node. `Graphviz` draws the graph inset by a margin, so the image is larger than the layout bounding box the `-Tplain` coordinates live

in; we recover that margin by also measuring the image at its *natural* size and comparing it with the `-Tplain` graph size (the `graph` line gives it in inches, hence the factor 72). Node positions are then expressed as fractions of the natural image, so they line up with the drawn nodes whatever size the image is finally scaled to.

```

598 \newcommand\pgph@graphlinked[1]{%
599   \begingroup
600     \catcode'\:=12 \catcode'\_:=12 \catcode'\;:=12 \catcode'\!=12
601     \catcode'\?=12 \catcode'\&:=12 \catcode'\^:=12 \catcode'\~=12
602     \InputIfFileExists{\pgph@file-proofgraph.map}{}{}%
603   \endgroup
604   \sbox\pgph@imgbox{%
605     \includegraphics[#1]{\pgph@file-proofgraph.\pgph@format}}%
606   \sbox\pgph@natbox{%
607     \includegraphics{\pgph@file-proofgraph.\pgph@format}}%
608   \edef\pgph@imgw{\the\wd\pgph@imgbox}%
609   \edef\pgph@imgh{\the\dimexpr\ht\pgph@imgbox+\dp\pgph@imgbox\relax}%
610   \edef\pgph@natw{\the\wd\pgph@natbox}%
611   \edef\pgph@nath{\the\dimexpr\ht\pgph@natbox+\dp\pgph@natbox\relax}%
612   \begin{tikzpicture}
613     \node[inner sep=\z@,outer sep=\z@] (pgph@P){\usebox\pgph@imgbox};
614     \openin\pgph@plainin=\pgph@file-proofgraph.plain\relax
615     \pgph@loopplain
616     \closein\pgph@plainin
617   \end{tikzpicture}%
618 }
619 \newcommand\pgph@loopplain{%
620   \unless\ifeof\pgph@plainin
621     \read\pgph@plainin to \pgph@line
622     \pgph@procline
623     \expandafter\pgph@loopplain
624   \fi
625 }
626 \newcommand\pgph@procline{%
627   \IfBeginWith{\pgph@line}{graph }%
628     {\expandafter\pgph@dograph\pgph@line\pgph@stop}%
629     {\IfBeginWith{\pgph@line}{node }%
630       {\expandafter\pgph@donode\pgph@line\pgph@stop}{}}%
631 }
632 \def\pgph@dograph graph #1 #2 #3 #4\pgph@stop{%
633   \def\pgph@gw{#2}\def\pgph@gh{#3}}
634 \def\pgph@donode node #1 #2 #3 #4 #5 #6\pgph@stop{%
635   \StrDel{#1}{"}[\pgph@nid]%
636   \pgph@linkfalse
637   \ifcsname pgph@lbl@\pgph@nid\endcsname
638     \edef\pgph@tgt{\csname pgph@lbl@\pgph@nid\endcsname}%
639     \edef\pgph@thislink{\noexpand\hyperref[\pgph@tgt]}%
640     \pgph@linktrue
641   \else
642     \ifcsname pgph@cbl@\pgph@nid\endcsname
643       \edef\pgph@tgt{\csname pgph@cbl@\pgph@nid\endcsname}%
644       \edef\pgph@thislink{\noexpand\hyperlink{cite.\pgph@tgt}}%
645       \pgph@linktrue
646   \fi
647   \fi

```

```

648 \ifpgph@link
649 % Per-side Graphviz margin: half the natural-minus-layout size.
650 \pgfmathsetmacro\pgph@mx{(\pgph@natw-\pgph@gw*72)/2}%
651 \pgfmathsetmacro\pgph@my{(\pgph@nath-\pgph@gh*72)/2}%
652 % Node centre as a fraction of the natural image.
653 \pgfmathsetmacro\pgph@fx{(\pgph@mx+#2*72)/\pgph@natw}%
654 \pgfmathsetmacro\pgph@fy{(\pgph@my+#3*72)/\pgph@nath}%
655 \pgfmathsetlengthmacro\pgph@bw{#4*72/\pgph@natw*\pgph@imgw*0.92}%
656 \pgfmathsetlengthmacro\pgph@bh{#5*72/\pgph@nath*\pgph@imgw*0.9}%
657 \node[anchor=center]
658   at ($($\pgph@P.south west)! \pgph@fx!(\pgph@P.south east)$)%
659   !\pgph@fy!($\pgph@P.north west)! \pgph@fx!(\pgph@P.north east)$)$)
660   {\pgph@thislink{\phantom{\rule{\pgph@bw}{\pgph@bh}}}};%
661 \fi
662 }
663 </package>

```

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	
\!	600
\"	412
\&	601
\:	600
\;	600
\?	601
\@M	502
\@Opargbegintheorem	264, 265, 266
\@auxout	54
\@backslashchar	528
\@begintheorem	139, 140, 142, 146
\@currentlabel	191, 205
\@ifpackageloaded	5, 296, 564, 587, 588
\@ifundefined	6, 10, 521, 574
\@namedef	45, 47, 61, 90, 97, 103, 188, 395, 404, 409, 429, 474, 483, 565, 566
\@opargbegintheorem	141, 150, 151, 152, 253, 254, 255
\@protected@testopt	281
\@spbegintheorem	156, 157, 158
\@spopargbegintheorem	161, 162, 163, 258, 259, 260
\[32
\]	412
\{	33, 495
\}	33, 413
\]	32
\^	601
_	600
\	412
\~	601
A	
\apxproofhook	301
\AtBeginDocument	123, 295, 563, 564
\AtBeginEnvironment	104, 284
\AtEndDocument	563
\autoref	274
\axp@forward@link	184, 204
B	
\begin	612
C	
\catcode	32, 33, 412, 600, 601
\closein	616
\Cref	274
\cref	273
\cs	16
\csundef	89, 96
D	
\DeclareBoolOption	20, 25, 27
\DeclareStringOption	19, 21, 22, 23, 24, 26, 28
\detokenize	281, 342, 518, 525
\dimexpr	609, 611
\dp	609, 611
E	
\end	617
\endproof	7, 8, 11
\expandarg	417
\ExplSyntaxOff	17
\ExplSyntaxOn	15
H	
\hbox	175, 275, 504
\ht	609, 611
\hyper@@link	500
\hyperlink	644
\hyperref	639
I	
\IfBeginWith	59, 451, 519, 627, 629
\ifboolexpr	461
\ifdefstring	462, 468, 512
\ifeof	620
\IfFileExists	578, 589
\ifnumequal	461
\ifpgph@apxproof	294
\ifpgph@autorun	556
\ifpgph@canlink	567, 592
\ifpgph@cite	249, 307
\ifpgph@hyperlinks	586
\ifpgph@link	568, 648
\ifpgph@natnum	496, 507
\IfSubStr	282, 528
\iftrue	522
\ignorespaces	501
\includegraphics	595, 605, 607
\InputIfFileExists	602
J	
\jobname	30

	L		
\label	135, 136	\pgph@cb	35, 553
	M	\pgph@cbase	511, 532, 538
\mbox	504	\pgph@cbaseg	514, 529, 532
\meaning	280	\pgph@citecmd	316, 320
	N	\pgph@citeedge	452, 478
\newcount	37, 38	\pgph@citelabel	512
\newif	294, 496, 567, 568	\pgph@citesanitize	339, 498, 516
\newread	571	\pgph@citeslot	333, 337
\newsavebox	569, 570	\pgph@citeslotcnt	38, 346, 347
\newtheorem	70, 71	\pgph@citestyle	396, 397, 488
\nobreakspace	503	\pgph@ckey	510, 511, 513, 517
\node	613, 657	\pgph@clabel	485, 488, 535
\noexpand	132, 214, 320, 639, 644	\pgph@class@endproof	7, 11
	O	\pgph@class@proof	7, 11
\openin	614	\pgph@curproof	44, 171, 172, 176, 219, 222, 246, 303, 304, 330, 334, 375
	P	\pgph@curresult	43, 130, 132, 169, 172, 181, 186, 188, 192, 199, 200, 201, 206, 246
\PackageError	575	\pgph@curslot	334, 344, 347, 348, 349, 351
\PackageWarning	580	\pgph@defaultenvs	83, 124
\penalty	502	\pgph@defaulttexclude	86, 100
\pgfmathsetlengthmacro	655, 656	\pgph@direction	468
\pgfmathsetmacro	650, 651, 653, 654	\pgph@dn	439, 441
\pgph@addege	213, 222, 334	\pgph@dograph	628, 632
\pgph@apxinit	297, 300	\pgph@dohook	101, 109, 111
\pgph@apxproofcapture	300	\pgph@donode	630, 634
\pgph@apxprooftrue	297	\pgph@dotesc	411, 442, 485
\pgph@auxmap	45	\pgph@dqchar	419
\pgph@beginproof	225, 284, 290	\pgph@dx	518, 519, 525, 528
\pgph@beginresult	104, 125	\pgph@edge	214, 450
\pgph@bh	656, 660	\pgph@edges	40, 215, 379, 551
\pgph@bschar	418, 419	\pgph@ef	382, 383, 386
\pgph@bw	655, 660	\pgph@emit	540
\pgph@candidates	56, 62, 63, 65, 118, 119	\pgph@emitedge	463, 467, 492
\pgph@canlinkfalse	585	\pgph@emitnode	427, 432, 436
\pgph@canlinktrue	590	\pgph@engine	557, 559
\pgph@cap@autoref	228	\pgph@excluderule	393, 407
\pgph@cap@cite	312	\pgph@excludes	42, 393, 549
\pgph@cap@cite@noopt	321, 328	\pgph@f	402, 403, 404
\pgph@cap@cite@opt	321, 322	\pgph@file	30, 543, 544, 558, 560, 578, 581, 589, 595, 602, 605, 607, 614
\pgph@cap@cite@opti	323, 324	\pgph@format	557, 558, 578, 581, 595, 605, 607
\pgph@cap@cite@optii	323, 326	\pgph@fx	653, 658, 659
\pgph@cap@cite@x	317, 318	\pgph@fy	654, 659
\pgph@cap@citegen	312, 313, 314, 315	\pgph@gh	633, 651
\pgph@cap@citep	314	\pgph@gnotestr	340, 342, 350, 351
\pgph@cap@citerec	325, 327, 328, 329	\pgph@graphlinked	593, 598
\pgph@cap@citet	313	\pgph@gw	633, 650
\pgph@cap@Cref	227	\pgph@gxdefcs	310, 348, 349, 351
\pgph@cap@cref	226	\pgph@hookcite	310
\pgph@cap@eqref	230	\pgph@hooknames	125
\pgph@cap@ref	225		
\pgph@cap@vref	232		

<code>\pgph@hookproofhead</code>	250, 252	<code>\pgph@out</code>	443, 475, 486, 540, 543, 545, 546, 553, 554
<code>\pgph@idcnt</code>	37, 129, 130	<code>\pgph@plainedge</code>	386, 453, 455
<code>\pgph@ignorerule</code>	390, 401	<code>\pgph@plainin</code>	571, 614, 616, 620, 621
<code>\pgph@ignores</code>	41, 390, 548	<code>\pgph@prependnode</code>	125, 133
<code>\pgph@imgbox</code>	569, 604, 608, 609, 613	<code>\pgph@procline</code>	622, 626
<code>\pgph@imggh</code>	609, 656	<code>\pgph@proofinner</code>	288, 291, 292
<code>\pgph@imgw</code>	608, 655	<code>\pgph@proofmeaning</code>	280, 282
<code>\pgph@includegraph</code>	579, 584	<code>\pgph@proofopt</code>	291, 292
<code>\pgph@install</code>	173, 174, 233, 247, 248, 305, 306	<code>\pgph@rankdir</code>	546
<code>\pgph@installcite</code>	249, 307, 355	<code>\pgph@recordref</code>	218, 225, 226, 227, 229, 231, 232, 372
<code>\pgph@installciteone</code>	356, 357, 358, 360	<code>\pgph@recordresult</code>	125
<code>\pgph@k</code>	221, 222, 332, 333, 385, 386	<code>\pgph@redefproof</code>	283, 287
<code>\pgph@key</code>	481, 490	<code>\pgph@register</code>	56, 73, 80, 91, 117
<code>\pgph@labeledge</code>	379, 381	<code>\pgph@replay</code>	319, 325, 327, 328
<code>\pgph@lbl</code>	441, 442, 444	<code>\pgph@resolve</code>	303, 382, 398, 402, 408, 456
<code>\pgph@line</code>	621, 627, 628, 629, 630	<code>\pgph@resolveregistration</code>	114, 124
<code>\pgph@linkfalse</code>	636	<code>\pgph@s</code>	437, 444
<code>\pgph@linktrue</code>	640, 645	<code>\pgph@saved@autoref</code>	229
<code>\pgph@loopplain</code>	615, 619, 623	<code>\pgph@saved@Cref</code>	227
<code>\pgph@maplabel</code>	135, 136, 198	<code>\pgph@saved@cref</code>	226
<code>\pgph@mapout</code>	446, 489, 541, 544, 555	<code>\pgph@saved@eqref</code>	231
<code>\pgph@maybehook</code>	107, 120	<code>\pgph@saved@ref</code>	225
<code>\pgph@mx</code>	650, 653	<code>\pgph@saved@vref</code>	232
<code>\pgph@my</code>	651, 654	<code>\pgph@saved@curproof</code>	171, 176
<code>\pgph@natbox</code>	570, 606, 610, 611	<code>\pgph@saved@label</code>	135, 211
<code>\pgph@natdo</code>	508, 524	<code>\pgph@scanproofarg</code>	256, 261, 267, 270
<code>\pgph@nath</code>	611, 651, 654, 656	<code>\pgph@scanrefs</code>	143, 153, 164, 168
<code>\pgph@natnumfalse</code>	520	<code>\pgph@selfloops</code>	462
<code>\pgph@natnumtrue</code>	523	<code>\pgph@setcitelabel</code>	478
<code>\pgph@natpick</code>	506, 508	<code>\pgph@setmap</code>	48, 199
<code>\pgph@natw</code>	610, 650, 653, 655	<code>\pgph@sig</code>	342, 343, 344, 348
<code>\pgph@nid</code>	635, 637, 638, 642, 643	<code>\pgph@slot</code>	480, 481, 482, 483, 484, 487, 490, 492
<code>\pgph@nil</code>	497, 506, 508	<code>\pgph@spnt@plain</code>	78, 79
<code>\pgph@nm</code>	423, 425, 441	<code>\pgph@spnt@star</code>	78, 81
<code>\pgph@node</code>	132, 421	<code>\pgph@stop</code>	572, 628, 630, 632, 634
<code>\pgph@nodes</code>	39, 126, 550	<code>\pgph@t</code>	402, 403, 404, 456, 457, 459, 460, 461, 463
<code>\pgph@nt@plain</code>	71, 72	<code>\pgph@testoptstr</code>	281, 282
<code>\pgph@nt@star</code>	71, 74	<code>\pgph@tgt</code>	638, 639, 643, 644
<code>\pgph@ob</code>	34, 545	<code>\pgph@thislink</code>	639, 644, 660
<code>\pgph@obrace</code>	495, 519	<code>\pgph@thisname</code>	58, 59, 60, 61, 63, 65
<code>\pgph@one</code>	88, 89, 90, 91, 95, 96, 97, 115, 116, 117, 119, 120, 220, 221, 331, 332, 384, 385	<code>\pgph@tmp</code>	131, 133
<code>\pgph@orig@bt</code>	140, 144, 147	<code>\pgph@tmpedge</code>	214, 216
<code>\pgph@orig@newtheorem</code>	70, 73, 74	<code>\pgph@tmpfn</code>	189, 190
<code>\pgph@orig@obt</code>	151, 154	<code>\pgph@tmpid</code>	49, 50, 51, 52, 54
<code>\pgph@orig@Obtp</code>	265, 267	<code>\pgph@tmpname</code>	185, 187
<code>\pgph@orig@obtp</code>	254, 256	<code>\pgph@tmpnum</code>	191, 193, 205, 207
<code>\pgph@orig@spbt</code>	157, 159	<code>\pgph@trimsp</code>	15, 221, 332, 385
<code>\pgph@orig@spnewtheorem</code>	76, 80, 81	<code>\pgph@wrapproof</code>	270, 298
<code>\pgph@orig@spobt</code>	162, 165	<code>\pgph@writeedge</code>	467
<code>\pgph@orig@spobtp</code>	259, 262		

<code>\pgph@x</code>	408, 409, 507, 508, 517, 518, 525, 527, 529	<code>\sbox</code>	604, 606
<code>\pgphnodemap</code>	447, 565	<code>\setbox</code>	175, 275
<code>\pgphnodemapcite</code>	490, 565	<code>\SetupKeyvalOptions</code>	18
<code>\phantom</code>	660	<code>\spnewtheorem</code>	75, 76, 77
<code>\ProcessKeyvalOptions</code>	29	<code>\StrBehind</code>	480
<code>\proof</code>	7, 8, 11, 279, 280, 289, 290	<code>\StrDel</code>	635
<code>\proofgraph</code>	6, 567	<code>\StrSubstitute</code>	418, 419
<code>\proofgraphedge</code>	6, 378		
<code>\proofgraphexclde</code>	6, 392	T	
<code>\proofgraphignore</code>	6, 389	<code>\tl</code>	16
<code>\proofgraphstyle</code>	6, 395		
<code>\proofgraphstylecite</code>	6, 397	U	
<code>\proofgraphtrack</code>	6, 11, 83	<code>\unexpanded</code>	126
<code>\proofgraphuntrack</code>	6, 11, 83	<code>\unhbox</code>	501
<code>\proofname</code>	291	<code>\unless</code>	620
<code>\proofof</code>	6, 273, 274, 373	<code>\unskip</code>	501
<code>\protect</code>	183, 203, 272, 499	<code>\usebox</code>	613
<code>\protected</code>	225, 226, 227, 228, 230, 232	<code>\usepackage</code>	576
<code>\protected@edef</code>	131, 214	<code>\uses</code>	5, 372
<code>\protected@write</code>	54	<code>\usetikzlibrary</code>	564
R		V	
<code>\read</code>	621	<code>\voidb@x</code>	502
<code>\ref</code>	273		
<code>\restoreexpandmode</code>	420	W	
<code>\rule</code>	660	<code>\wd</code>	608, 610
S		Z	
<code>\saveexpandmode</code>	417	<code>\z@</code>	175, 275, 613

Change History

v0.1.0		within another proof: the capture no longer reinstalls over itself	1
General: Initial version	1		
v1.0.0			
General: First released version ...	1		
v1.0.1		Support, with the <code>cite</code> option, the two-optional-argument citation form <code>[pre][post]{key}</code> of <code>natbib</code> and <code>biblatex</code> ; it previously broke compilation with confusing errors inside proofs ...	1
General: Capture <code>citet</code> and <code>citep</code> (including starred variants) like <code>cite</code> when they are defined ...	1		
Fix an infinite loop when, with the <code>cite</code> option, a citation occurred in a proof nested			