The apxproof package

Pierre Senellart
pierre@senellart.com
http://github.com/PierreSenellart/apxproof

2026/08/25 v1.5.0

Abstract

This package makes it easier to write articles where proofs and other material are deferred to the appendix. The appendix material is written in the LaTeX code along with the main text which it naturally complements, and it is automatically deferred. The package can automatically send proofs to the appendix, can repeat in the appendix the theorem environments stated in the main text, can section the appendix automatically based on the sectioning of the main text, and supports a separate bibliography for the appendix material.

1 Usage

The apxproof package is intended to simplify the writing of articles where some of the content needs to be deferred to an appendix. This is in particular useful for the submission of scientific articles to conferences or journals that limit the number of pages in the main text but allow an extra appendix, where proofs of theorems and other material can be added.

1.1 Basics

To use apxproof, first load it in the header of your document:

```
\usepackage{apxproof}
```

On its own, this does not do anything and should not change the appearance of your document. To add an appendix with some material from your document, use the `toappendix` environment:

`toappendix` (*env.*)

```
\begin{toappendix}
  ...
\end{toappendix}
```

The content will appear at the end of your document, in an automatically generated section that refers to the current section in the main text.

Example 1. *Throughout this documentation, all examples produce content deferred to the appendix, at the very end of this document.*

```
\begin{toappendix}
```

```
This content is in the appendix.
\end{toappendix}
```

When the content to put in appendix is an entire section, make sure that `\section` is the very first command that appears within the `toappendix` environment. It will disable the automatic production of a section heading.

1.2 Repeated Theorems and Proofs

In some scientific papers that include proofs, it is common to defer proofs to the appendix. This can easily be achieved using the `appendixproof` environment:

`appendixproof` (*env.*)

```
\begin{appendixproof}
  ...
\end{appendixproof}
```

This behaves like the `toappendix` environment, except that a proof environment is generated.

Example 2. *We now send a proof to the appendix:*

```
\begin{appendixproof}
This proof is in the appendix.
\end{appendixproof}
```

When deferring proofs to the appendix, an annoying problem is that the statement of the theorem remains in the main text; it is hard to read a proof that is far away from the statement it proves. apxproof offers two ways to address this.

`\newtheoremrep` `\newtheoremrep*` With `\newtheoremrep`, the statement of the theorem is *repeated* in the appendix, immediately above the deferred proof. With the starred variant `\newtheoremrep*`, the theorem is *not* repeated, but the proof is still automatically deferred to the appendix; this is convenient when the proofs are short enough or close enough to the main text that a restatement would be redundant, especially when combined with the `\appendixproofname` hook (see Section 1.5) to title proofs with the type and number of the theorem they refer to (e.g., "Proof of Theorem 1"). In summary, three behaviors are available for a theorem environment:

`\newtheorem` (LaTeX's standard command): the theorem appears in the main text only; the `proof` environment is left unchanged by apxproof.

`\newtheoremrep` : the theorem appears in the main text *and* is restated in the appendix above its proof; the `proof` that follows in the source is automatically deferred to the appendix.

`\newtheoremrep*` : the theorem appears in the main text only; the `proof` that follows is automatically deferred to the appendix, without restating the theorem there.

In all cases, theorems can share counters and a single document can mix the three behaviors freely.

To use the repeated-theorem feature, you can define a new *repeated theorem* environment using the `\newtheoremrep` command:

`\newtheoremrep` `\newtheoremrep{`⟨*name*⟩`}[`⟨*counter*⟩`]{`⟨*title*⟩`}[`⟨*countersec*⟩`]`

or, equivalently, its starred form to obtain a *proof-deferred* (but not repeated) theorem:

`\newtheoremrep*` `\newtheoremrep*{`⟨*name*⟩`}[`⟨*counter*⟩`]{`⟨*title*⟩`}[`⟨*countersec*⟩`]`

Usage of either form is exactly the same as that of AMS LaTeX's `\newtheorem` macro:

- ⟨*name*⟩ (e.g., `theorem`) is the name of an environment that is created for this kind of theorem;
- ⟨*counter*⟩ (e.g., `definition`) is an optional counter describing from which kind of environment the numbering of these environments should be inherited;
- ⟨*title*⟩ (e.g., `Theorem`) is the title that will be used to display this theorem environment;
- ⟨*countersec*⟩ (e.g., `section`) is an optional counter of a sectioning command indicating that counters for this theorem should be prefixed by this counter (and reset at each occurrence of the sectioning command).

⟨*counter*⟩ and ⟨*countersec*⟩ should not be used together. What differs from `\newtheorem` is that, when the following is written:

```
\newtheoremrep{foobar}{Foobar}
```

then *two* environments are defined: the `foobar` environment, which behaves as if `\newtheorem` had been used, and the `foobarrep` environment, which results in the statement of this environment being repeated in the appendix. The starred form `\newtheoremrep*{foobarstar}{Foobarstar}` similarly defines two environments `foobarstar` and `foobarstarrep`, except that `\begin{foobarstarrep}` does not produce any restatement of the theorem in the appendix; only the following `proof` is deferred.

`proof` (*env.*)

One interesting feature of apxproof is that in most situations, there is no need to use the `appendixproof` environment. Indeed, the `proof` environment is redefined by apxproof to automatically put the proof either in the main text (if it follows a regular theorem) or in the appendix (if it follows a repeated theorem).

Example 3. *Assume we have first defined a repeated theorem environment* `foobar` *as above. We can now use this theorem environment, first for a regular theorem in the main text, then for a theorem repeated in the main text and in the appendix:*

```
\begin{foobar}
This foobar is a regular one, in the main text.
\end{foobar}
\begin{proof}
This is the proof of the regular foobar.
\end{proof}
```

We obtain:

Foobar 1. *This foobar is a regular one, in the main text.*

Proof. This is the proof of the regular foobar. □

Now, if we use a repeated theorem:

```
\begin{foobarrep}
This foobar is repeated in the appendix.
\end{foobarrep}
\begin{proof}
This is the proof of the repeated foobar.
\end{proof}
```

We now obtain:

Foobar 2. *This foobar is repeated in the appendix.*

Note that, since `hyperref` *is loaded, there are hyperlinks created between the statements of the theorems in the main text and in the appendix.*

Example 4. *In contrast, the starred form of* `\newtheoremrep` *produces a theorem whose proof is deferred to the appendix* without *restating the theorem there. Assuming the declaration* `\newtheoremrep*{foobarstar}{Foobarstar}` *at the top of the document, writing:*

```
\begin{foobarstarrep}
This foobar will not be repeated in the appendix,
but the proof below will be deferred there.
\end{foobarstarrep}
\begin{proof}
This is the proof of the (non-repeated) foobar.
\end{proof}
```

yields:

Foobarstar 1. *This foobar will not be repeated in the appendix, but the proof below will be deferred there.*

In this mode, the title of the deferred proof in the appendix is controlled by the `\appendixproofname` *hook documented in Section 1.5; by default, this hook is empty, so proofs are titled simply "Proof." as usual. The forward hyperlink from the theorem number in the main text (when* hyperref *is loaded) points to the location of the proof in the appendix.*

When the proof is deferred to the appendix, it is common practice to add a proof sketch in the main text. apxproof defines a simple `proofsketch` environment for this purpose:

`proofsketch` (*env.*)

```
\begin{proofsketch}
  ...
\end{proofsketch}
```

The proof sketch is typeset similarly to a proof, but is always in the main text.

`inlineproof` (*env.*) Similarly, an `inlineproof` environment is provided so as to be able to have both a proof in the appendix (using the regular `proof` environment, or alternatively the `appendixproof` environment) and a different proof in the main text (using the `inlineproof` environment).

Example 5. *Here are simple examples of proof sketches and inline proofs:*

```
\begin{proofsketch}
This is a proof sketch.
\end{proofsketch}
```

Proof sketch. This is a proof sketch. □

```
\begin{inlineproof}
This is an inline proof.
\end{inlineproof}
```

Proof. This is an inline proof. □

1.3 Bibliography

By default, apxproof automatically adds a bibliography in the appendix with only the references cited in the appendix material. This allows for a clean separation of references used solely in the main text, and those used in the appendix.

Example 6. *Assume we have citations both in the main text and in the appendix.*

```
This is a citation in the main text~\cite{lamport86}.
\begin{toappendix}
This is a citation in the appendix~\cite{proofsAreHard}.
\end{toappendix}
```

This is a citation in the main text [1].

The bibliography in the appendix can use a different style and heading than the bibliography in the main text (and, by default, it does). See Section 1.5 for how to configure the appearance of that bibliography.

option `bibliography`

In order to use a single appendix for the main text and the bibliography, one can specify the value `common` to the `bibliography` option when loading the package.

option `bibengine`

By default, apxproof relies on the bibunits package to typeset a separate bibliography for the appendix material; this is the `bibengine=bibtex` mode (the default), suitable for documents that use bibtex (with or without natbib) for citations and references. Documents using biblatex can instead set `bibengine=biblatex`, in which case apxproof wraps the appendix material in a biblatex `refsection` environment and emits the appendix bibliography with `\printbibliography`. In this mode the user is responsible for loading biblatex and declaring bibliography resources with `\addbibresource` in the preamble; resources are shared between the main document and the appendix `refsection`, but citations and the printed bibliography are scoped independently. The `bibengine` option only has an effect when `bibliography=separate`. (By default this option is set to `separate`.)

1.4 Mode

option `appendix`

An optional ⟨*mode*⟩ can be specified when loading the package:

```
\usepackage[appendix=⟨mode⟩]{apxproof}
```

⟨*mode*⟩ can take one of the following four values:

`append` This is the default. Appendix material gathered by apxproof is appended to the main text.

`inline` In this mode, apxproof simply inlines the content along with the main text.

`strip` This mode functions similarly to `append` except that the appendix is not appended at the end of the document. All appendix material is therefore removed.

`chapterend` In this mode, appendix material for each chapter is gathered separately and emitted at the end of that chapter (i.e., just before the next `\chapter`, `\part`, `\frontmatter`, `\mainmatter`, or `\backmatter` command, or at the end of the document), using the `subappendices` environment from the appendix package. Only `bibliography=common` is supported in this mode. See Section 1.5 for the `\chapterappendixprelim` and `\flushchapterappendix` customization macros.

1.5 Customization

apxproof provides a few macros that can be redefined (using `\renewcommand`) to customize the appearance of the appendix:

`\mainbodyrepeatedtheorem` `\mainbodyrepeatedtheorem` is a macro that is executed at the beginning of the body of every repeated theorem. This can be used to notify the reader that the theorem is repeated in appendix in some way, e.g., with a margin note.

`\appendixsectionformat` `\appendixsectionformat{`⟨***number***⟩`}{`⟨***title***⟩`}` is a macro that indicates how to format the section titles in the Appendix, given the number and title of the section in the main text. By default, they appear as "Proofs for Section ⟨*number*⟩ (⟨*title*⟩)".

`\appendixproofname` `\appendixproofname{`⟨***type***⟩`}{`⟨***ref***⟩`}` is a macro that determines the title of a proof deferred to the appendix when it follows a `\newtheoremrep` or `\newtheoremrep*` environment. By default, this macro is empty, which means proofs are titled "Proof." as usual. When redefined to a non-empty value, the expansion is used as the title of the appendix proof; ⟨*type*⟩ is the title of the theorem environment (e.g., `Theorem`) and ⟨*ref*⟩ is a reference to the theorem statement in the main text. To title all such proofs "Proof of ⟨*type*⟩ ⟨*number*⟩", one would typically write:

```
\renewcommand{\appendixproofname}[2]{Proof of #1~#2}
```

This hook applies whether the theorem is restated in the appendix (`\newtheoremrep`) or not (`\newtheoremrep*`). It has no effect on manually written `appendixproof` environments or on proofs given an explicit optional title.

`\appendixrefname` `\appendixrefname` contains the heading that is displayed before the bibliography. By default, this is "References for the Appendix". (Note that this command is also defined and used by the `memoir` document class.)

`\appendixbibliographystyle` `\appendixbibliographystyle` contains the `.bst` bibliography style that is used in the bibliography in appendix. By default, this is `alpha`.

`\appendixbibliographyprelim` `\appendixbibliographyprelim` contains arbitrary code that is executed just before the production of the bibliography in appendix, which can be used to configure the way it is displayed.

`\appendixprelim` `\appendixprelim` contains arbitrary code that is executed just before the production of the appendix, which can be used to configure the way it is displayed. By default, this command contains `\clearpage\onecolumn` (the appendix is typeset on a new page in single-column mode) but redefining this option allows changing this behavior. This macro has no effect in `chapterend` mode; use `\chapterappendixprelim` instead.

`\chapterappendixprelim` `\chapterappendixprelim` (`chapterend` mode only) contains arbitrary code that is executed just before each chapter's appendix block. By default, this command contains `\clearpage` (each chapter appendix starts on a new page). This macro has no effect in `append` or `strip` mode; use `\appendixprelim` instead.

`\flushchapterappendix` `\flushchapterappendix` (`chapterend` mode only) manually emits the current chapter's accumulated appendix material at the point of invocation, and starts a fresh per-chapter buffer. The package automatically inserts an equivalent flush before each `\chapter`, `\part`, `\frontmatter`, `\mainmatter`, `\backmatter`, and at the end of the document, so this macro is only useful in unusual document structures where an appendix needs to be emitted at a location that is not a standard structural boundary.

option `repeqn` Another customization capability concerns *numbered equations* that are present within repeated theorems. An optional `repeqn` option can be specified when loading the package, which controls whether equation numbers should be as in the main text (by setting this option to `same`, the default) or independently numbered (by setting this option to `independent`). In the latter case, whenever a referenceable counter is set with `\label{`⟨*counter*⟩`}`, `\ref{`⟨*counter*⟩`}` references the counter in the main text, while `\ref{`⟨*counter*⟩`-apx}` references the counter in the appendix (except in `inline` mode, where both have the same effect).

Enhanced cross-referencing packages are supported: the optional argument that cleveref adds to `\label` (as in `\label[equation]{`⟨*counter*⟩`}`) may be used within repeated material, and so may the `\zlabel` command of zref-clever, both giving rise to the same `-apx` variants as `\label` does. Note that cleveref loses that optional argument inside amsmath display environments (`equation`, `align`, etc.), where the label ends up unreferenceable; this happens with or without apxproof, and affects neither plain `\label` nor zref-clever.

option `forwardlinking` Another customization option concerns hyperlinking. Usually, when hyperref is loaded, `foobarrep` environments in the main text have their number link to their repetition in the appendix. To suppress this behavior and have `foobarrep` environments treated as if hyperref were not loaded, one can specify the value `no` to the `forwardlinking` option when loading the package. (By default this option is set to `yes`.)

1.6 SyncTEX Support

option `synctex`

SyncTEX lets editors and PDF viewers jump between a position in the PDF and the source line that produced it. apxproof supports it in the appendix as well as in the main text, and nothing has to be done for that: rather than copying appendix material and repeated theorems to an auxiliary file, which would attribute them to that file, the package typesets them from the source itself.

The `synctex` option says when it does so. With `auto` (the default), exactly when SyncTEX output was requested (i.e., when the engine was run with `-synctex=1` or a similar option, which is what editors and latexmk do); with `yes`, always; with `no`, never, and SyncTEX is then not supported. The typeset output is the same in all cases. Should the re-reading cause trouble in a document, `synctex=no` restores the previous behavior.

Jumping from the PDF to the source (inverse search) works with every engine. Jumping from the source to the PDF (forward search) finds appendix material with LuaTEX; with pdfTEX and XeTEX, it does so only if the `apxproof-synctex` script distributed with the package (it requires Python 3) is run after each compilation:

```
apxproof-synctex ⟨jobname⟩
```

With latexmk, adding

```
$success_cmd = 'apxproof-synctex %R';
```

to a `latexmkrc` file does this automatically.

Section 4 lists what re-reading the source asks of a document, and Section 8.4 explains how it is done.

1.7 Advanced Features

We now describe a few advanced macros and environments, the usage of which is limited to special cases:

`nestedproof` (*env.*) `nestedproof` is an environment that can be used within a `proof` environment deferred in the appendix; this is required because, for technical reasons, no `proof` environment can be nested within a deferred `proof` environment.

`\noproofinappendix` `\noproofinappendix` can be used inside repeated theorems that are not followed by a `proof` or `appendixproof` environment; the point is to ensure that a further `proof` environment cannot be mistakenly understood as a proof of the repeated theorem. It should not be needed in most situations as apxproof tries figuring out when a proof follows a repeated theorem automatically, but may occasionally be needed in complex scenarios.

`\nosectionappendix` `\nosectionappendix` is to be used inside a section that *does* contain appendix material, but for which a section in the appendix should not be created. This should be rarely needed. When this command is present, appendix material is appended to the end of the previously created section.

`\apxproofhook` `\apxproofhook` is a one-argument hook, doing nothing by default, that apxproof runs inside every proof it typesets in the appendix, and inside an inline proof whose title names its result through a cross-reference (`\ref`, `\cref`, `\Cref`, `\autoref`, `\vref` or `\zcref`). Its argument is a label of the result the proof

establishes: the label of the preceding repeated theorem in the first case, the target of the title's cross-reference in the second. Redefining it is how a companion package, such as proofgraph, attributes each proof to the result it proves; documents have no reason to.

1.8 Grouping Appendix Material Under a Single Section

By default, apxproof produces one section in the appendix for each section of the main text that contains appendix material (with the format controlled by `\appendixsectionformat`). If you would rather have a single section in the appendix (e.g., titled "Proofs") with the per-section entries appearing as *subsections* of that single section, two recipes are available.

The first is fully explicit: disable the automatic per-section heading with `\nosectionappendix`, and write the desired structure directly inside `toappendix` environments.

```
\begin{toappendix}
  \section{Proofs}
\end{toappendix}

\section{First}\label{sec:first}
\nosectionappendix
\begin{toappendix}
  \subsection{Proofs of Section~\ref{sec:first}}
\end{toappendix}

% ... theorems and proofs of section ``First'' ...

\section{Second}\label{sec:second}
\nosectionappendix
\begin{toappendix}
  \subsection{Proofs of Section~\ref{sec:second}}
\end{toappendix}

% ... theorems and proofs of section ``Second'' ...
```

The reader of the source can then see exactly what will be produced in the appendix, at the cost of some verbosity.

The second recipe is more concise. The section level used by apxproof for the auto-generated appendix headings is swapped from `\section` to `\subsection` for the duration of an explicit `\begingroup`/`\endgroup` group:

```
\begin{toappendix}
  \section{Proofs}
  \begingroup\makeatletter
    \let\axp@oldsection\subsection
  \makeatother
\end{toappendix}

\section{First}\label{sec:first}
% ... theorems and proofs of section ``First'' ...

\section{Second}\label{sec:second}
```

```
% ... theorems and proofs of section ``Second'' ...

\begin{toappendix}
  \endgroup
\end{toappendix}
```

The wrapping `\section{Proofs}` is written once, and every subsequent auto-generated appendix section heading is demoted to a `\subsection` for the scope of the group. The trade-off is that this recipe relies on an internal macro (`\axp@oldsection`) and on a manually balanced `\begingroup`/`\endgroup` pair across two separate `toappendix` environments.

2 Supported Document Classes

Because apxproof modifies sectioning commands, bibliographies, and proofs, it may not work straight away with arbitrary document classes. It has currently been tested with and is supported for the following document classes:

- LaTeX standard document classes (`article.cls`, `book.cls`)
- KOMA-Script (`scrartcl.cls`, `scrbook.cls`)
- `memoir.cls`
- ACM journals and proceedings (`acmart.cls`)
- IEEE transactions and conferences (`IEEEtran.cls`)
- Springer's Lecture Notes in Computer Science (`llncs.cls`)
- Schloß Dagstuhl's Leibniz International Proceedings in Informatics (`lipics.cls`, `lipics-v2016.cls`, `lipics-v2019.cls`, `lipics-v2021.cls`)

Other classes may work out of the box. Adding support for specific classes is possible and can be requested from the author of this package.

3 HTML Conversion with LaTeXML

LaTeXML (https://math.nist.gov/~BMiller/LaTeXML/) converts LaTeX documents to HTML. It cannot read `apxproof.sty` itself, so the package comes with a binding, `apxproof.sty.ltxml`, which reproduces its behavior in the conversion: proofs deferred to the appendix, repeated theorem statements with their original numbering, automatic appendix sectioning, forward links from theorems to their proofs, and a separate appendix bibliography.

The binding is installed with the package, next to `apxproof.sty`. LaTeXML looks for it there, and in the directory of the document; should it be kept anywhere else, point LaTeXML at that directory:

```
latexmlc --path=⟨dir⟩ --dest=paper.html paper.tex
```

Do *not* pass `--includestyles`: the binding is self-contained, and that option makes LaTeXML process the dependencies of the real `apxproof.sty`, which fails. LaTeXML 0.8.8 or later is required.

4 Known Issues and Limitations

We report here some issues we are currently aware of:

- When using hyperref, the appendix in the bibliography is not hyperlinked. This is to avoid possible issues with multiply defined bibliography entries.
- `appendixproof`, `proof`, `toappendix` environments cannot be nested. This is a limitation of the fancyvrb package that apxproof relies on. Note the existence of the `nestedproof` environment for nested proofs.
- SyncTEX support (Section 1.6) works by re-reading the source when the appendix is typeset, which requires each captured environment to appear literally in a source file: `\begin{foobarrep}`, `\begin{proof}`, `\begin{toappendix}` and their `\end` counterparts cannot come from a macro (nor could they be captured before, except for repeated theorems). Conversely, wrapping a `foobarrep` environment in a user-defined environment is supported only with this option.
- SyncTEX support also requires the source files to be readable, under the same names, when the appendix is typeset, which is the case in any normal compilation. The main file is `\jobname.tex`, or, when TEX was given a path to it, the name recorded in the `.fls` file written by the `-recorder` option (latexmk and most editors pass it). When the main file cannot be located, apxproof says so and copies appendix material to the `.axp` file as with `synctex=no`, which leaves the document complete but without SyncTEX support for that material; when a single piece cannot be re-read, an error message explains why and that piece is missing from the appendix.
- SyncTEX forward search, from the source to the PDF, reaches appendix material with LuaTEX only: with pdfTEX and XeTEX, a line of a proof or of a `toappendix` environment lands on the nearest line of the main text unless the `apxproof-synctex` script is run (Section 1.6), the way SyncTEX identifies files making it impossible to do better without breaking forward search from the main text. SyncTEX support also has a cost: each piece of appendix material is reached by reading its source file from the beginning, which a document with hundreds of proofs in a very long file will notice.
- By default, when `bibliography=separate`, apxproof uses the bibunits package to generate a second bibliography. Packages incompatible with bibunits (notably biblatex) will therefore not work in that default configuration. For biblatex users, set `bibengine=biblatex` (in addition to `bibliography=separate`) to use biblatex's own `refsection` machinery instead. The `bibliography=common` mode remains a third option that works with most bibliography packages.

Issues not listed here should be reported to the author.

5 License

Copyright © 2016–2026 by Pierre Senellart.

This work may be distributed and/or modified under the conditions of the LaTeX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is in http://www.latex-project.org/lppl.txt and version 1.3 or later is part of all distributions of LaTeX version 2005/12/01 or later.

6 Contact

- https://github.com/PierreSenellart/apxproof
- Pierre Senellart <pierre@senellart.com>

Bug reports and feature requests should preferably be submitted through the *Issues* feature of GitHub.

7 Acknowledgments

Thanks to Antoine Amarilli for feedback and proofreading. Thanks to K. D. Bauer for the implementation of the forward-linking mechanism, and for various bugfixes. Thanks to Leonid Kostrykin for an initial implementation of the `forwardlinking` option.

8 Implementation

We now describe the entire code of the package, in a literate programming fashion. Throughout the package, we use the `axp@` prefix to identify local macros and environment names, which are not meant to be used by the final user.

8.1 Dependencies

We first load a few package dependencies:

- environ to easily define the repeated theorem environments.

```
\RequirePackage{environ}
```

- etoolbox to define simple toggles.

```
\RequirePackage{etoolbox}
```

- fancyvrb for the bulk of the work of exporting appendix material in an auxiliary file.

```
\RequirePackage{fancyvrb}
```

- ifthen for easier comparison of character strings.

```
\RequirePackage{ifthen}
```

- kvoptions to manage options passed to the package.

```
\RequirePackage{kvoptions}
```

- catchfile to be able to check the content of files `\input` within appendix content.

```
\RequirePackage{catchfile}
```

- amsthm for its `\newteorem` macro. Some document classes (e.g., lipics) preload amsthm: this is fine, `\RequirePackage{amsthm}` will simply have no effect. On the other hand, some other document classes (e.g., llncs or sig-alternate) define a `proof` environment that conflicts with amsthm, so we have to undefine this environment before loading amsthm. In that case, we reestablish the existing proof environments, in case they had been customized (e.g., sig-alternate)

```
\@ifpackageloaded{amsthm}{
  }{
    \let\apx@oldamsthmproof\proof
    \let\apx@oldamsthmendproof\endproof
    \let\proof\undefined
    \let\endproof\undefined
  }
\RequirePackage{amsthm}
\ifdefined\apx@oldamsthmproof
  \let\proof\apx@oldamsthmproof
  \let\endproof\apx@oldamsthmendproof
\fi
```

8.2 Option Processing

Many names throughout the package use an arobase (`@`) to avoid name conflict with user-defined names. To simplify the compilation of the documentation, we simply make it a regular character in all the rest.

```
\makeatletter
```

We setup the processing of options using keyval facilities.

```
\SetupKeyvalOptions{
  family=axp,
  prefix=axp@
}
```

We declare the following options:

- `appendix`, with a default value of `append` (other possible values: `strip`, `inline`);
- `bibliography`, with a default value of `separate` (other possible value: `common`);
- `repeqn`, with a default value of `same` (other possible value: `independent`).

`\axp@appendix`

```
\DeclareStringOption[append]{appendix}
```

`\axp@bibliography`

```
\DeclareStringOption[separate]{bibliography}
```

\axp@bibengine

```
\DeclareStringOption[bibtex]{bibengine}
```

\axp@repeqn

```
\DeclareStringOption[same]{repeqn}
```

\axp@forwardlinking

```
\DeclareStringOption[yes]{forwardlinking}
```

\axp@synctex

```
\DeclareStringOption[auto]{synctex}

\ProcessLocalKeyvalOptions*
```

We check that the value of the options are valid, and add a message to the compilation log.

```
\ifthenelse{\equal{\axp@appendix}{append}}{
  \message{apxproof: Appendix material appended to the document}
}{\ifthenelse{\equal{\axp@appendix}{strip}}{
  \message{apxproof: Appendix material stripped}
}{\ifthenelse{\equal{\axp@appendix}{inline}}{
  \message{apxproof: Appendix material inlined within the document}
}{\ifthenelse{\equal{\axp@appendix}{chapterend}}{
  \message{apxproof: Appendix material appended at the end of each chapter}
}{
  \errmessage{Error: unsupported option appendix=\axp@appendix\ for
  package apxproof}
}}}}
\ifthenelse{\equal{\axp@bibengine}{bibtex}}{
}{\ifthenelse{\equal{\axp@bibengine}{biblatex}}{
}{
  \errmessage{Error: unsupported option bibengine=\axp@bibengine\ for
  package apxproof}
}}
\ifthenelse{\equal{\axp@bibliography}{separate}}{
```

In `bibengine=bibtex` mode (the default), the external bibunits package is used to add a second bibliography for the appendix material. In `bibengine=biblatex` mode, no extra package is loaded here: biblatex is expected to have been loaded by the user, and its `refsection` environment is used at appendix typeset time to scope citations.

```
  \ifthenelse{\equal{\axp@bibengine}{bibtex}}{
    \RequirePackage{bibunits}
    \message{apxproof: Separate bibliography for appendix material (bibunits)}
  }{
    \message{apxproof: Separate bibliography for appendix material (biblatex)}
  }
}{\ifthenelse{\equal{\axp@bibliography}{common}}{
  \message{apxproof: Common bibliography for appendix and main text}
}{
  \errmessage{Error: unsupported option bibliography=\axp@bibliography\ for
  package apxproof}
}}
\ifthenelse{\equal{\axp@appendix}{chapterend}
```

```
            \AND\equal{\axp@bibliography}{separate}}{
  \PackageError{apxproof}{%
    bibliography=separate is not supported with appendix=chapterend}{%
    Use bibliography=common instead.}
}{}
\ifthenelse{\equal{\axp@repeqn}{same}}{
  \message{apxproof: Repeated equations keep the same numbering}
}{\ifthenelse{\equal{\axp@repeqn}{independent}}{
  \message{apxproof: Repeated equations are independently numbered}
}{
  \errmessage{Error: unsupported option repeqn=\axp@repeqn\ for
  package apxproof}
}}
```

`\axp@forward@suppress`

```
\newbool{axp@forward@suppress}
\ifthenelse{\equal{\axp@forwardlinking}{yes}}{
}{\ifthenelse{\equal{\axp@forwardlinking}{no}}{
  \message{apxproof: Disable forward linking}
  \global\booltrue{axp@forward@suppress}%
}{
  \errmessage{Error: unsupported option forwardlinking=\axp@repeqn\ for
  package apxproof}
}}
```

`axp@synctex` The `axp@synctex` Boolean records whether appendix material is to be re-read from the source files rather than copied to the `.axp` file (see Section 8.4). With `synctex=auto`, this is decided by the `\synctex` primitive of the engine, which is nonzero exactly when SyncTEX output was requested (positive for compressed, negative for uncompressed `.synctex` files); the primitive is only tested if it exists.

```
\newbool{axp@synctex}
\ifthenelse{\equal{\axp@synctex}{no}}{
}{\ifthenelse{\equal{\axp@synctex}{yes}}{
  \booltrue{axp@synctex}
}{\ifthenelse{\equal{\axp@synctex}{auto}}{
  \ifdefined\synctex
    \ifnum\synctex=\z@\else
      \booltrue{axp@synctex}
    \fi
  \fi
}{
  \errmessage{Error: unsupported option synctex=\axp@synctex\ for
  package apxproof}
}}}
```

Whether re-reading is actually possible further depends on the main source file being locatable, which is settled at the end of Section 8.4; the Boolean may still be cleared there.

8.3 Macros Common to All Compilation Modes

`xp@newtheoremrep@definetheorem` Common to all compilation modes, we define `\axp@newtheoremrep@definetheorem`. When called with first argument `foobar`, we first undefine the existing `foobar` environment (and its counter) if it has already been defined (e.g., by the document

class), then invoke \axp@newtheorem for the regular version of the theorem foobar, saving and restoring any existing theorem counter unless the \newtheoremdep redefines the base counter.

```
\def\axp@newtheoremrep@definetheorem#1#2#3#4{%
  \expandafter\let\csname #1\endcsname\undefined
  \expandafter\let\csname the#1\endcsname\undefined
  \ifcsname c@#1\endcsname
    \expandafter\expandafter\expandafter\let\expandafter\expandafter
      \csname c@axp@#1\endcsname\csname c@#1\endcsname
    \expandafter\let\csname c@#1\endcsname\undefined
  \fi
  \axp@newtheorem{#1}{#2}{#3}{#4}%
  \ifcsname c@axp@#1\endcsname
    \ifx\relax#2\relax
      \expandafter\expandafter\expandafter\let\expandafter\expandafter
        \csname c@#1\endcsname\csname c@axp@#1\endcsname
    \else
    \fi
  \fi
}
```

\axp@newtheorem
\@axp@newtheorem
\@@axp@newtheorem

We introduce an intermediate \axp@newtheorem command to define a new theorem, differently depending on whether there is a section counter or not. This will be useful, in particular to allow changing this definition depending on the document class. This command uses two intermediary commands, \@axp@newtheorem and \@@axp@newtheorem, for the non-starred and starred versions.

```
\def\axp@newtheorem{\@ifstar\@@axp@newtheorem\@axp@newtheorem}
\def\@axp@newtheorem#1#2#3#4{%
  \ifx\relax#4\relax
    \newtheorem{#1}[#2]{#3}%
  \else
    \newtheorem{#1}{#3}[#4]%
  \fi
}
\def\@@axp@newtheorem#1#2{%
  \newtheorem*{#1}{#2}%
}
```

\newtheoremrep
\axp@newtheoremreptmp

We define the high-level \newtheoremrep to have the same syntax as amsthm's \newtheorem, plus an optional * variant that defers the proof to the appendix without restating the theorem. For this purpose, we need a little trick to deal with the second and fourth optional arguments, which is what \@oparg and \axp@newtheoremreptmp are used for. \axp@newtheoremrep is defined differently depending on the compilation mode.

```
\newcommand\newtheoremrep{%
  \@ifstar{\axp@newtheoremrep@star}{\axp@newtheoremrep@nostar}%
}
\newcommand\axp@newtheoremrep@star[1]{%
  \global\csdef{axp@proofof@#1}{}%
  \@oparg{\axp@newtheoremreptmp{#1}}[]%
}
\newcommand\axp@newtheoremrep@nostar[1]{%
  \global\csundef{axp@proofof@#1}%
```

```
  \@oparg{\axp@newtheoremreptmp{#1}}[]%
}
\def\axp@newtheoremreptmp#1[#2]#3{%
  \@oparg{\axp@newtheoremrep{#1}[#2]{#3}}[]%
}
```

`proofsketch` (*env.*) Simple `proofsketch` environment.

```
\newenvironment{proofsketch}{\begin{axp@oldproof}[Proof sketch]}{\end{axp@oldproof}}
```

`\mainbodyrepeatedtheorem` `\appendixrefname` `\appendixbibliographystyle` `\appendixbibliographyprelim` `\appendixprelim` `\appendixsectionformat` `\appendixproofname` We provide sensible defaults for these user-customizable macros. Even though they are not all useful in all modes, we define them for all modes so that a `\renewcommand` works in all cases.

```
\newcommand{\mainbodyrepeatedtheorem}{}
\providecommand{\appendixrefname}{References for the Appendix}
\newcommand{\appendixbibliographystyle}{alpha}
\newcommand{\appendixbibliographyprelim}{}
\newcommand{\appendixprelim}{\clearpage\onecolumn}
\newcommand{\appendixsectionformat}[2]{Proofs for Section~#1\ (#2)}
\newcommand{\appendixproofname}[2]{}
```

`axp@oldproof` (*env.*) We save the definition of the existing `proof` environment.

```
\let\axp@oldproof\proof
\let\endaxp@oldproof\endproof
```

We define a utility macro that will be used to properly set the `\label` command (and its amsmath counterpart, `\label@in@display`) for equations within repeated theorems, depending on the compilation mode.

Some packages, cleveref in particular, give `\label` (and `\label@in@display`) an optional argument specifying the type of the counter being labeled. Our replacements must therefore look for such an optional argument and pass it on unchanged, or the opening bracket would be taken as the label name and the rest of the argument typeset as text. We do *not* pass the optional argument on to the extra `-apx` label of the repeated material: that label is set within an unnumbered copy of the theorem, where overriding the counter type would apply to state that cleveref did not set up, and the type of the counter (for equations, the only case where the `-apx` label is really of use) is correctly inferred from the context anyway.

zref-clever, the maintained successor of cleveref, does not touch `\label` at all: it sets its own labels through `\zlabel`, and, with its `labelhook` option (set by default), also through the kernel's `label` hook, so that a plain `\label` sets a zref label as well. The latter needs nothing from us, since our replacement of `\label` calls the original one. A `\zlabel` used directly, however, must be given the same treatment as `\label`; it has no optional argument.

`\axp@label` `\axp@label@in@display`

```
\def\axp@label{\@ifnextchar[\axp@label@opt\axp@label@noopt}%]
\def\axp@label@opt[#1]#2{\axp@label@do{[#1]}{#2}}
\def\axp@label@noopt#1{\axp@label@do{}{#1}}
\def\axp@label@in@display{%
  \@ifnextchar[\axp@label@in@display@opt\axp@label@in@display@noopt}%]
\def\axp@label@in@display@opt[#1]#2{\axp@label@in@display@do{[#1]}{#2}}
\def\axp@label@in@display@noopt#1{\axp@label@in@display@do{}{#1}}
```

\axp@redefinelabels

```
\newcommand{\axp@redefinelabels}{%
  \providecommand\label@in@display{}%
  \let\axp@oldlabel\label
  \let\axp@oldlabel@in@display\label@in@display
  \ifdefined\zlabel\let\axp@oldzlabel\zlabel\fi
  \ifthenelse{\equal{\axp@appendix}{inline}}{%
    \def\axp@label@do##1##2{%
      \axp@oldlabel##1{##2}%
      \axp@oldlabel{##2-apx}%
    }%
    \def\axp@label@in@display@do##1##2{%
      \axp@oldlabel@in@display##1{##2}%
      \axp@oldlabel{##2-apx}%
    }%
    \def\axp@zlabel##1{\axp@oldzlabel{##1}\axp@oldzlabel{##1-apx}}%
  }{%
    \def\axp@label@do##1##2{\axp@oldlabel{##2-apx}}%
    \def\axp@label@in@display@do##1##2{\axp@oldlabel@in@display{##2-apx}}%
    \def\axp@zlabel##1{\axp@oldzlabel{##1-apx}}%
  }%
  \let\label\axp@label
  \let\label@in@display\axp@label@in@display
  \ifdefined\zlabel\let\zlabel\axp@zlabel\fi
}
```

8.3.1 Class-Specific Behavior

Finally, some class-specific behavior common to all compilation modes.

llncs and other Springer document classes

```
  \ifdefined\spnewtheorem
```

\@axp@newtheorem
\@@axp@newtheorem

It is necessary to use \spnewtheorem instead of \newtheorem in Springer document classes to obtain standard formatting.

```
\def\@axp@newtheorem#1#2#3#4{%
  \ifx\relax#4\relax
    \ifx\relax#2\relax
      \spnewtheorem{#1}{#3}{\bfseries}{\itshape}%
    \else
      \spnewtheorem{#1}[#2]{#3}{\bfseries}{\itshape}%
    \fi
  \else
    \spnewtheorem{#1}{#3}[#4]{\bfseries}{\itshape}%
  \fi
}
\def\@@axp@newtheorem#1#2{%
  \spnewtheorem*{#1}{#2}{\upshape\bfseries}{\itshape}%
}
```

proofsketch

We redefine the proofsketch environment, which is used differently in the base class.

```
\renewenvironment{proofsketch}{\begin{axp@oldproof}[sketch]}{\end{axp@oldproof}}
```

We have to redefine the macro `\@thmcountersep` for proper sectioned counters.

```
\def\@thmcountersep{.}
  \fi
```

acmart Some versions of acmart define a `\ACM@origsection` macro and verify that `\section` has the same content. We need to update `\ACM@origsection` accordingly.

```
\ifdefined\ACM@origsection
  \AtBeginDocument{\let\ACM@origsection\section}
\fi
```

8.4 SyncTEX Support

SyncTEX records, for every box in the output, the file and line from which the tokens that produced it were read. Two things defeat it in apxproof: the body of a `foobarrep` environment collected by environ is typeset from a macro, so all of it is attributed to the line of the `\end{foobarrep}`; and appendix material copied verbatim to the `.axp` file is attributed to that file when it is typeset. There is no way to tell SyncTEX that some tokens "really" come from elsewhere; the only tokens it attributes correctly are those read directly from a file. When the `synctex` option is enabled, we therefore do just that: repeated theorems are typeset directly from the source instead of being collected, and instead of copying appendix material to the `.axp` file, we write there the name of the source file, the line where the environment begins, the line where it ends and its name; when the appendix is typeset, the source file is opened again with the primitive `\input`, all lines up to the beginning of the environment are skipped, and the environment is executed anew, with its begin and end code replaced by what the appendix needs. All of this happens with the source file open, so SyncTEX attributes every box to the right file and line.

We store, at the beginning of each captured environment, where it was found. The line comes from `\@currenvline`, which `\begin` sets before reading any optional argument (`\inputlineno` would be off when the argument or the body starts on a later line). The file comes from the kernel's `\CurrentFileUsed` and `\CurrentFilePathUsed`, which are empty for the main file, in which case we use `\axp@region@mainfile`, the name under which the main file can be read (see `\axp@region@findmain` below); older kernels lack these macros, and we then have to assume that every environment is in the main file.

We also need to know how SyncTEX identifies the file, see `\axp@region@input` below: with LuaTEX, we record the tag it gave the file (`\axp@region@tag` is 0 elsewhere).

`\axp@region@file`
`\axp@region@line`
`\axp@region@env`
`\axp@region@tag`
`\axp@region@mainfile`
`\axp@extract@line`
`\axp@region@record`

```
\def\axp@region@file{}
\def\axp@region@line{}
\def\axp@region@env{}
\def\axp@region@tag{0}
\edef\axp@region@mainfile{\jobname.tex}
\def\axp@extract@line#1line #2\@nil#3\@nnil{#2}
\def\axp@region@record{%
  \ifdefined\CurrentFileUsed
```

```
    \ifx\CurrentFileUsed\@empty
      \let\axp@region@file\axp@region@mainfile
    \else
      \ifx\CurrentFilePathUsed\@empty
        \edef\axp@region@file{\CurrentFileUsed}%
      \else
        \edef\axp@region@file{\CurrentFilePathUsed/\CurrentFileUsed}%
      \fi
    \fi
  \else
    \let\axp@region@file\axp@region@mainfile
  \fi
  \edef\axp@region@line{%
    \expandafter\axp@extract@line\@currenvline\@nil line 0\@nil\@nnil}%
  \edef\axp@region@env{\@currenvir}%
  \ifdefined\directlua
    \edef\axp@region@tag{\directlua{tex.sprint(tex.get_synctex_tag())}}%
  \else
    \def\axp@region@tag{0}%
  \fi
}
\ifbool{axp@synctex}{
  \ifdefined\CurrentFileUsed\else
    \PackageWarning{apxproof}{%
      This LaTeX kernel does not provide \string\CurrentFileUsed;
      the synctex option assumes\MessageBreak
      that all appendix material is in \jobname.tex}
  \fi
}{}
```

`\axp@region@write` At the end of the environment, this writes to the `.axp` file the `\axp@replaybody` command replaying it, with the recorded position, the current line (that of the `\end`), the two arguments passed by the caller: the tokens that will replace the begin code of the environment in the appendix, and those that will replace its end code, and the SyncTEX tag. The trailing `\relax` ensures that the end of the line in the `.axp` file, which is only tokenized once the replay is over, does not produce a space, as the copied material never ended with one.

```
\def\axp@region@write#1#2{%
  \immediate\write\axp@proofsfile{%
    \noexpand\axp@replaybody{\axp@region@file}{\axp@region@line}%
      {\the\inputlineno}{\axp@region@env}{#1}{#2}{\axp@region@tag}\relax
  }%
}
```

`\axp@replaybody` `\axp@replaybody{`⟨*file*⟩`}{`⟨*start*⟩`}{`⟨*end*⟩`}{`⟨*env*⟩`}{`⟨*begin*⟩`}{`⟨*end code*⟩`}{`⟨*tag*⟩`}` is executed in the appendix. It first checks, by reading the file with `\readline` (which does not affect SyncTEX), that the file exists and that its ⟨*start*⟩ line does contain `\begin{`⟨*env*⟩`}` and its ⟨*end*⟩ line an `\end{`; if not, an error is raised and nothing is typeset. Otherwise the file is replayed. The state of the replay is kept in `\axp@replay@...` macros, distinct from the `\axp@region@...` ones set by `\axp@region@record`: a `foobarrep` environment nested in a `toappendix` environment runs its main-text begin code when replayed, and must not disturb the replay of the enclosing environment.

```
\newif\ifaxp@region@ok
\newif\ifaxp@region@wrapped
\def\axp@replaybody#1#2#3#4#5#6#7{%
  \def\axp@replay@file{#1}%
  \def\axp@replay@line{#2}%
  \def\axp@replay@endline{#3}%
  \def\axp@replay@env{#4}%
  \def\axp@replay@begin{#5}%
  \def\axp@replay@end{#6}%
  \def\axp@replay@tag{#7}%
  \axp@region@check
  \ifaxp@region@ok
    \axp@region@replay
  \else
    \axp@region@flushsection
  \fi
}
```

`\axp@region@section` `\axp@region@flushsection` For `toappendix` environments, the appendix section heading is written to the `.axp` file as `\axp@region@section`, which stores it; it is typeset by `\axp@region@toappendix` below with the same look-ahead as `\axp@section`, once the content of the environment is available, or by `\axp@region@flushsection` without look-ahead if the environment cannot be replayed. The assignments are global, as the heading is typeset from within the group of the environment.

```
\let\axp@region@pendingsection\relax
\def\axp@region@section#1{\gdef\axp@region@pendingsection{#1}}
\def\axp@region@flushsection{%
  \ifx\axp@region@pendingsection\relax\else
    \expandafter\axp@oldsection\expandafter
      {\axp@region@pendingsection}%
    \global\let\axp@region@pendingsection\relax
  \fi
}
```

`\axp@region@check` `\axp@region@readto` `\axp@region@readone` Lines are read one at a time into `\axp@line`, with `\endlinechar` disabled so that no end-of-line character is appended, until the desired line number is reached; `\axp@region@readcount` counts the lines read so far. We never read from a stream on which `\ifeof` is true, since TEX would then read from the terminal. As the environments of a given file are replayed in the order in which they appear, we keep the stream open from one replay to the next and only reopen the file when we need an earlier line or a different file (`\axp@region@openfile` records the file currently open); each source file is thus read at most a few times overall. This state is global, as `\axp@replaybody` may be called within a group.

```
\newread\axp@readfile
\newcount\axp@region@readcount
\def\axp@region@openfile{}
\def\axp@line{}
\def\axp@region@readto#1{%
  \loop
    \ifnum\axp@region@readcount<#1\relax
      \axp@region@readone{#1}%
  \repeat
}
```

```
\def\axp@region@readone#1{%
  \ifeof\axp@readfile
    \def\axp@line{}%
    \global\axp@region@readcount=#1\relax
  \else
    \global\advance\axp@region@readcount\@ne
    \readline\axp@readfile to\axp@line
  \fi
}
\def\axp@region@check{%
  \axp@region@okfalse
  \edef\axp@region@saveendlinechar{\the\endlinechar}%
  \endlinechar\m@ne
  \ifx\axp@region@openfile\axp@replay@file
    \ifnum\axp@region@readcount<\axp@replay@line\relax
    \else
      \gdef\axp@region@openfile{}%
    \fi
  \else
    \gdef\axp@region@openfile{}%
  \fi
  \ifx\axp@region@openfile\@empty
    \closein\axp@readfile
    \openin\axp@readfile=\axp@replay@file\relax
    \global\axp@region@readcount=\z@
    \global\let\axp@region@openfile\axp@replay@file
  \fi
  \ifeof\axp@readfile
    \ifnum\axp@region@readcount=\z@
      \PackageError{apxproof}{%
        Cannot open `\axp@replay@file' to re-read the\MessageBreak
        \axp@replay@env\space environment starting on line
        \axp@replay@line}{%
        The synctex option of apxproof requires the source files of the
        document\MessageBreak
        to be readable under the same names when the appendix is
        typeset.\MessageBreak
        This appendix material will be missing.}%
      \gdef\axp@region@openfile{}%
    \else
      \axp@region@toofar
    \fi
  \else
    \axp@region@readto{\axp@replay@line}%
    \ifeof\axp@readfile
      \axp@region@toofar
    \else
      \axp@region@analyze
      \ifaxp@region@ok
        \axp@region@readto{\axp@replay@endline}%
        \axp@region@checkend
      \fi
    \fi
  \fi
```

```
  \endlinechar\axp@region@saveendlinechar\relax
}
\def\axp@region@toofar{%
  \axp@region@okfalse
  \PackageError{apxproof}{%
    File `\axp@replay@file' has fewer than \axp@replay@line\space
    lines, where\MessageBreak
    the \axp@replay@env\space environment to re-read was expected}{%
    The source file has probably changed since the main text was
    processed,\MessageBreak
    or the main file is not `\jobname.tex'.
    This appendix material will be missing.}%
}
```

Several macros need `\begin{`, `\end{`, `}` or an end-of-line character as delimiters, with the catcode 12 that these characters have when read from `\readline` or with the catcodes we set to skip lines. We define them with `\lowercase`, mapping placeholder characters to the ones we need, which preserves catcodes.

`\axp@region@scanline` `\axp@region@scan` `\axp@region@getname` `\axp@region@scanendline` `\axp@region@scanend` `\axp@gobbleline` `\axp@gobbletobegin`

- `\axp@region@scanline` splits its argument at the first `\begin{`;
- `\axp@region@getname` splits its argument at the first `}`;
- `\axp@region@scanendline` splits its argument at the first `\end{`;
- `\axp@gobbleline` discards the input up to and including the next end of line;
- `\axp@gobbletobegin` discards the input up to and including the next `\begin{`.

In the first three, a sentinel is appended so that a match is always found; the last two then continue with the macros indicated.

```
\begingroup
  \lccode`\!=`\\ \lccode`\?=`\{ \lccode`\|=`\} \lccode`\"=`\^^M
  \lccode`\+=`\b \lccode`\-=`\e \lccode`\*=`\g \lccode`\/=`\i
  \lccode`\==`\n \lccode`\,=`\d
  \catcode`\!=12 \catcode`\?=12 \catcode`\|=12 \catcode`\"=12
  \catcode`\+=12 \catcode`\-=12 \catcode`\*=12 \catcode`\/=12
  \catcode`\==12 \catcode`\,=12
  \lowercase{\endgroup
    \long\def\axp@region@scanline#1{%
      \axp@region@scan#1!+-*/=?\axp@region@marker\@nil}
    \long\def\axp@region@scan#1!+-*/=?#2\@nil{\axp@region@scan@i{#2}}
    \long\def\axp@region@getname#1{\axp@region@getname@i#1|\@nil}
    \long\def\axp@region@getname@i#1|#2\@nil{\def\axp@region@name{#1}}
    \long\def\axp@region@scanendline#1{%
      \axp@region@scanend#1!-=,?\axp@region@marker\@nil}
    \long\def\axp@region@scanend#1!-=,?#2\@nil{%
      \def\axp@region@rest{#2}}
    \long\def\axp@gobbleline#1"{\axp@skiplines}
    \long\def\axp@gobbletobegin#1!+-*/=?{\axp@region@nextbegin}
  }
```

`\axp@region@analyze` `\axp@region@scan@i` We look on the start line for `\begin{`⟨*env*⟩`}`. As the environment may be wrapped in a user-defined environment (in which case the line contains, say,

\begin{mythm}), we settle for the first \begin{ on the line if the exact one is absent. \axp@region@begins is set to the number of \begin{ to skip on the line before the one to use.

```
\newcount\axp@region@begins
\newcount\axp@skipcount
\newif\ifaxp@region@found
\def\axp@region@marker{\axp@region@marker}
\def\axp@region@analyze{%
  \axp@region@begins=\z@
  \axp@region@foundfalse
  \edef\axp@region@target{%
    \expandafter\detokenize\expandafter{\axp@replay@env}}%
  \expandafter\axp@region@scanline\expandafter{\axp@line}%
  \ifaxp@region@found
    \axp@region@oktrue
  \else
    \ifnum\axp@region@begins>\z@
      \axp@region@begins=\z@
      \axp@region@oktrue
    \else
      \PackageError{apxproof}{%
        Cannot find \string\begin\@charlb\axp@replay@env\@charrb\space
        on line \axp@replay@line\space of\MessageBreak
        `\axp@replay@file' to re-read it}{%
        The synctex option of apxproof requires
        \string\begin\@charlb\axp@replay@env\@charrb\space
        to appear\MessageBreak
        literally in the source file, on the line where the environment
        starts.\MessageBreak
        This appendix material will be missing.}%
    \fi
  \fi
}
```

\axp@region@scan@i receives what follows a \begin{. If this is only the sentinel, there is no further \begin{ on the line. Otherwise we extract the environment name and compare it with the target, moving on to the next \begin{ if it differs.

```
\long\def\axp@region@scan@i#1{%
  \def\axp@region@rest{#1}%
  \ifx\axp@region@rest\axp@region@marker
  \else
    \axp@region@getname{#1}%
    \ifx\axp@region@name\axp@region@target
      \axp@region@foundtrue
    \else
      \advance\axp@region@begins\@ne
      \axp@region@scan#1\@nil
    \fi
  \fi
}
```

\axp@region@checkend Similarly, we check that the end line contains an \end{.

```
\def\axp@region@checkend{%
  \expandafter\axp@region@scanendline\expandafter{\axp@line}%
```

```
  \ifx\axp@region@rest\axp@region@marker
    \axp@region@okfalse
    \PackageError{apxproof}{%
      Cannot find the \string\end\space of the \axp@replay@env\space
      environment\MessageBreak
      starting on line \axp@replay@line\space of `\axp@replay@file'
      (expected on line \axp@replay@endline)}{%
      The synctex option of apxproof requires the \string\end\space
      of the environment\MessageBreak
      to appear literally in the source file. This appendix material
      will be missing.}%
  \fi
}
```

`\axp@region@replay` `\axp@region@cleanup` `\axp@region@catcodes`

Now the actual replay. Within the appendix, `\begin{`⟨*env*⟩`}` must run the begin code we were given and `\end{`⟨*env*⟩`}` the end code; we save the current meaning of the corresponding macros, and of `\end` itself (see below), to restore them afterwards. We also save the catcodes of all characters, as they will be modified at the end of the replay (see below): `\axp@region@catcodes` expands to the assignments restoring them. We deliberately do not wrap the replay in a group, so that whatever the `\end` of the environment does to the paragraph that follows (as `\@doendpe` does) survives it, exactly as when the material is copied to the `.axp` file. The file is then opened by the primitive `\input` in a group where every character is ignored (catcode 9) except the end-of-line character, which is an ordinary one: lines are skipped without TEX building any token, until the start line, which is read with every character an ordinary one (so that none is special, no control sequence is formed and no line can be mistaken for a paragraph break); the group ends when we hand over to the environment.

```
\def\axp@region@catcodes#1{%
  \catcode#1=\the\catcode#1\relax
  \ifnum#1<255
    \expandafter\axp@region@catcodes\expandafter
      {\number\numexpr#1+1\expandafter}%
  \fi}
\def\axp@region@setcatcodes#1{%
  \count@=\z@
  \loop
    \catcode\count@=#1\relax
    \ifnum\count@<255
    \advance\count@\@ne
  \repeat}
\def\axp@region@replay{%
  \edef\axp@region@catcoderestore{\axp@region@catcodes{0}}%
  \expandafter\let\expandafter\axp@region@savedbegin
    \csname\axp@replay@env\endcsname
  \expandafter\let\expandafter\axp@region@savedend
    \csname end\axp@replay@env\endcsname
  \expandafter\let\csname\axp@replay@env\endcsname\axp@replay@begin
  \expandafter\let\csname end\axp@replay@env\endcsname\axp@replay@end
  \let\axp@region@savedendcmd\end
  \axp@region@wrappedfalse
  \begingroup
    \axp@region@setcatcodes{9}%
```

```
    \catcode13=12\relax
    \endlinechar=13\relax
    \axp@skipcount=\axp@replay@line\relax
    \advance\axp@skipcount\m@ne
    \axp@region@input
  \axp@region@cleanup
}
\def\axp@region@cleanup{%
  \axp@region@input@done
  \expandafter\let\csname\axp@replay@env\endcsname\axp@region@savedbegin
  \expandafter\let\csname end\axp@replay@env\endcsname\axp@region@savedend
  \let\end\axp@region@savedendcmd
  \expandafter\let\csname axp@region@isregion@\axp@replay@env\endcsname
    \undefined
  \ifaxp@region@wrapped
    \expandafter\let\csname end\axp@region@name\endcsname
      \axp@region@savedwrappedend
  \fi
  \axp@region@catcoderestore
}
```

`\axp@region@input` `\axp@region@input@done` `\axp@region@altfile`

Opening the file is where we have to care about the way SyncTEX names files. TEX gives every file it opens a new tag, and records the tag and name in the `.synctex` file, so re-reading a file registers it a second time under the same name; and the parser used by viewers and editors, when asked (forward search, from the editor to the PDF) for a source line of some file, resolves the name to a single tag, the last one registered under that name. With the re-read registered as is, forward search from the main text would then land in the appendix, or nowhere. Two remedies:

- With LuaTEX, we can have the nodes created while re-reading carry the file's original tag, with the Lua function `tex.force_synctex_tag` (which requires `tex.set_synctex_mode(2)`; line numbers remain the actual ones), and not register the re-read at all (`tex.set_synctex_no_files`). Both directions then work everywhere. Files `\input` from within the material are handled normally again during their reading, thanks to the kernel's file hooks. All this is done in Lua, see below.
- With other engines, we open the file under a name that the parser does not consider equivalent to the recorded one, but which designates the same file and which editors normalize: ⟨*cwd*⟩`/../`⟨*cwd name*⟩`/`⟨*file*⟩. Forward search then behaves as when the material was copied (main text only), while inverse search from the appendix gives the right file and line. The working directory is read from the `.fls` file written by the `-recorder` option (used by latexmk and most editors); failing that, the file is opened under its plain name.

```
\ifdefined\directlua
  \def\axp@region@input{%
    \ifnum\axp@replay@tag>\z@
      \directlua{apxproof.synctex_force(\axp@replay@tag)}%
      \global\let\axp@region@forcedtag\axp@replay@tag
    \fi
    \expandafter\axp@skiplines\@@input\axp@replay@file\relax}
  \def\axp@region@input@done{%
    \ifnum\axp@replay@tag>\z@
```

```
    \directlua{apxproof.synctex_unforce()}%
    \global\let\axp@region@forcedtag\relax
  \fi}
\let\axp@region@forcedtag\relax
\ifdefined\AddToHook
  \AddToHook{file/before}{%
    \ifx\axp@region@forcedtag\relax\else
      \directlua{apxproof.synctex_unforce()}%
    \fi}
  \AddToHook{file/after}{%
    \ifx\axp@region@forcedtag\relax\else
      \directlua{apxproof.synctex_force(\axp@region@forcedtag)}%
    \fi}
\fi
```

The Lua side. Forcing the tag does not reach nodes created from Lua (such as the glyphs that luaotfload inserts for ligatures), which in this mode get no tag and line at all and would be attributed, when the page is shipped out, to wherever TEX is then. A filter run before line breaking and box packing therefore gives such nodes, while a tag is forced, the tag and the line of the nearest preceding node (or following one, at the start of a list). (Font kerns are a different matter: LuaTEX records them with the position of the enclosing `\input` rather than with the position of the glyphs they sit between, in the main text just as well, and does not use the fields we could set; this is a LuaTEX peculiarity that we leave alone.)

```
\directlua{
  apxproof = apxproof or {}
  apxproof.forced_tag = 0
  local get, set = node.direct.get_synctex_fields,
                   node.direct.set_synctex_fields
  local traverse, todirect = node.direct.traverse, node.direct.todirect
  function apxproof.synctex_force(tag)
    apxproof.saved_tag, apxproof.saved_line =
      tex.get_synctex_tag(), tex.get_synctex_line()
    tex.set_synctex_no_files(1)
    tex.set_synctex_mode(2)
    tex.force_synctex_tag(tag)
    tex.set_synctex_tag(tag)
    tex.set_synctex_line(0)
    apxproof.forced_tag = tag
  end
  function apxproof.synctex_unforce()
    apxproof.forced_tag = 0
    tex.force_synctex_tag(0)
    tex.set_synctex_tag(apxproof.saved_tag or 0)
    tex.set_synctex_line(apxproof.saved_line or 0)
    tex.set_synctex_mode(0)
    tex.set_synctex_no_files(0)
  end
  local getprev, tail = node.direct.getprev, node.direct.tail
  function apxproof.synctex_fix(head)
    local tag = apxproof.forced_tag
    if tag == 0 then return true end
    local last_l = 0
    local pending = false
```

```
      for n in traverse(todirect(head)) do
        local t, l = get(n)
        if t == 0 or (t == tag and l == 0) then
          if last_l > 0 then
            set(n, tag, last_l)
          else
            pending = true
          end
        elseif t == tag then
          last_l = l
        end
      end
      if pending then
        local n = tail(todirect(head))
        last_l = 0
        while n do
          local t, l = get(n)
          if t == 0 or (t == tag and l == 0) then
            if last_l > 0 then set(n, tag, last_l) end
          elseif t == tag then
            last_l = l
          end
          n = getprev(n)
        end
      end
      return true
    end
    luatexbase.add_to_callback("pre_linebreak_filter",
      apxproof.synctex_fix, "apxproof.synctex")
    luatexbase.add_to_callback("hpack_filter",
      apxproof.synctex_fix, "apxproof.synctex")
  }
\else
  \def\axp@region@input{%
    \axp@region@altfile
    \expandafter\axp@skiplines\@@input\axp@replay@altfile\relax}
  \let\axp@region@input@done\relax
\fi
```

`\axp@region@altfile` defines `\axp@replay@altfile` from `\axp@replay@file`. The working directory is looked up once, in the first line of the `.fls` file, which reads `PWD` ⟨*directory*⟩; its last component is extracted by `\axp@region@lastcomponent`. An absolute file name (an unusual case) is prefixed with a second `/` instead, which has the same effect. As the same read stream is used, we forget the file the check pass may have kept open.

```
\let\axp@region@pwd\relax
\def\axp@region@altfile{%
  \ifx\axp@region@pwd\relax
    \axp@region@findpwd
  \fi
  \ifx\axp@region@pwd\@empty
    \let\axp@replay@altfile\axp@replay@file
  \else
    \expandafter\axp@region@altfile@i\axp@replay@file\@nil
```

```
  \fi}
\def\axp@region@altfile@i#1#2\@nil{%
  \if/#1%
    \edef\axp@replay@altfile{/\axp@replay@file}%
  \else
    \edef\axp@replay@altfile{%
      \axp@region@pwd/../\axp@region@pwdbase/\axp@replay@file}%
  \fi}
\def\axp@region@findpwd{%
  \gdef\axp@region@pwd{}%
  \IfFileExists{\jobname.fls}{%
    \begingroup
      \endlinechar\m@ne
      \openin\axp@readfile=\jobname.fls\relax
      \ifeof\axp@readfile\else
        \readline\axp@readfile to\axp@line
        \expandafter\axp@region@findpwd@i\axp@line
          \relax\relax\relax\relax\@nil
      \fi
      \closein\axp@readfile
    \endgroup
    \gdef\axp@region@openfile{}%
  }{}%
}
\def\axp@region@findpwd@i#1#2#3#4\@nil{%
  \def\axp@tmp{#1#2#3}%
  \edef\axp@tmpa{\detokenize{PWD}}%
  \ifx\axp@tmp\axp@tmpa
    \axp@region@findpwd@ii#4\@nil
  \fi}
\def\axp@region@findpwd@ii#1\relax#2\@nil{%
  \axp@region@findpwd@iii#1\@nnil\@nnil}
\def\axp@region@findpwd@iii#1#2\@nnil#3\@nnil{%
  \ifx\@nnil#1\else
    \xdef\axp@region@pwd{#1#2}%
    \xdef\axp@region@pwdbase{\axp@region@lastcomponent#1#2/\@nil}%
    \ifx\axp@region@pwdbase\@empty
      \gdef\axp@region@pwd{}%
    \fi
  \fi}
\def\axp@region@lastcomponent#1/#2\@nil{%
  \ifx\@empty#2\@empty
    #1%
  \else
    \axp@region@lastcomponent#2\@nil
  \fi}
```

`\axp@region@findmain` The kernel leaves `\CurrentFileUsed` empty for the main file and tells us nothing about the name TEX opened it under, so `\jobname.tex` is the obvious guess; it is wrong whenever the document was compiled by giving TEX a path, as in `pdflatex sub/doc.tex`, and the material would then all be missing from the appendix. The `.fls` file records the name that was actually used, in the first `INPUT` line whose last component is `\jobname.tex` (the entries before it are the configuration files and the format). We look it up once, here, rather than at replay

time: the Boolean is consulted while theorems are being defined in the preamble, and the `.fls` is already complete that early, since TEX opens the main file before reading anything of it. The lines are parsed as in `\axp@region@findpwd` above, with `INPUT` in place of `PWD`, and `\axp@region@stopmain` ends the scan on the first match or at the end of the file.

```
\newif\ifaxp@region@stopmain
\def\axp@region@findmain{%
  \IfFileExists{\jobname.fls}{%
    \begingroup
      \endlinechar\m@ne
      \edef\axp@tmp{\jobname.tex}%
      \edef\axp@region@jobnametex{%
        \expandafter\detokenize\expandafter{\axp@tmp}}%
      \openin\axp@readfile=\jobname.fls\relax
      \axp@region@stopmainfalse
      \loop
        \ifaxp@region@stopmain\else
          \ifeof\axp@readfile
            \axp@region@stopmaintrue
          \else
            \readline\axp@readfile to\axp@line
            \expandafter\axp@region@findmain@i\axp@line
              \relax\relax\relax\relax\relax\relax\@nil
          \fi
      \repeat
      \closein\axp@readfile
    \endgroup
    \gdef\axp@region@openfile{}%
  }{}%
}
\def\axp@region@findmain@i#1#2#3#4#5#6\@nil{%
  \def\axp@tmp{#1#2#3#4#5}%
  \edef\axp@tmpa{\detokenize{INPUT}}%
  \ifx\axp@tmp\axp@tmpa
    \axp@region@findmain@ii#6\@nil
  \fi}
\def\axp@region@findmain@ii#1\relax#2\@nil{%
  \axp@region@findmain@iii#1\@nnil\@nnil}
\def\axp@region@findmain@iii#1#2\@nnil#3\@nnil{%
  \ifx\@nnil#1\else
    \edef\axp@tmpa{\axp@region@lastcomponent#1#2/\@nil}%
    \ifx\axp@tmpa\axp@region@jobnametex
      \xdef\axp@region@mainfile{#1#2}%
      \axp@region@stopmaintrue
    \fi
  \fi}
```

`\axp@region@setup` We can now settle whether re-reading is possible at all. If the main file cannot be read under the name we have, nothing of it could be replayed, and every piece of appendix material coming from it would be lost; we therefore clear the Boolean, so that the whole package behaves exactly as with `synctex=no`, and say so. The document is then complete, only without SyncTEX support for appendix material. This has to happen before any `\newtheoremrep`, which consults the Boolean to

decide how to define the `foobarrep` environment.

```
\ifbool{axp@synctex}{
  \axp@region@findmain
  \IfFileExists{\axp@region@mainfile}{%
    \message{apxproof: SyncTeX support enabled, appendix material re-read
      from source files}%
  }{%
    \boolfalse{axp@synctex}%
    \PackageWarning{apxproof}{%
      Cannot read the main source file `\axp@region@mainfile';
      the synctex\MessageBreak
      option cannot re-read appendix material from it. Falling back
      to\MessageBreak
      copying it to the .axp file, as with synctex=no: the document
      is\MessageBreak
      complete, but SyncTeX does not cover appendix material.
      This\MessageBreak
      happens when TeX is given a path to the main file and no .fls
      file\MessageBreak
      is available to resolve it (option -recorder)}%
  }%
}{}
```

`\axp@skiplines` `\axp@region@nextbegin` `\axp@region@afterbegin`

Note that the tokens following `\@@input` in `\axp@region@replay` are only read once the file has been closed, and that when `\axp@skiplines` starts, the first line of the file has been read into TEX's buffer but nothing has been tokenized yet. We gobble whole lines until the start line, which is read with all characters ordinary, then `\begin{` as many times as needed, and close the catcode group: the rest of the line, still in the buffer, will be tokenized with the normal catcodes, but with `@` an ordinary character as in the rest of the document.

```
\def\axp@skiplines{%
  \ifnum\axp@skipcount>\z@
    \advance\axp@skipcount\m@ne
    \expandafter\axp@gobbleline
  \else
    \axp@region@setcatcodes{12}%
    \axp@skipcount=\axp@region@begins
    \expandafter\axp@gobbletobegin
  \fi}
\def\axp@region@nextbegin{%
  \ifnum\axp@skipcount>\z@
    \advance\axp@skipcount\m@ne
    \expandafter\axp@gobbletobegin
  \else
    \expandafter\axp@region@afterbegin
  \fi}
\def\axp@region@afterbegin{%
  \endgroup
  \catcode`\@=12
  \def\axp@region@name{}%
  \axp@region@readname}
```

`\axp@region@readname` `\axp@region@readname@go`

The name of the environment is then read character by character up to the closing

brace, which is consumed by a `\let`. (A macro cannot insert an unbalanced `{` into the input, and a delimited argument cannot be delimited by `}`.)

If the name is the expected one, we do not execute `\begin` at all, as material copied to the `.axp` file was not enclosed in a group, and we want the same here (a group would, e.g., undo what a `\subsection` in a `toappendix` environment does to the paragraph that follows it). We run the begin code directly, and let `\end` be `\axp@region@endcmd`, which runs the end code and `\axp@region@finish` for the expected environment, and delegates every other name to the saved `\end`; nested environments in the material thus behave as usual. The test on the name is done with `\ifcsname` on a flag set for the expected environment, so that the wrapper is expandable, as the kernel's `\end` is, up to the point where the original `\end` takes over: this matters when `\end{tabular}` is met while TEX scans the beginning of an alignment row, where anything unexpandable would open a new row. `\axp@region@finish` makes every character ignored, so that whatever remains of the `\end` line is dropped by TEX's tokenizer, and stops reading the file with `\endinput`. Doing this with catcodes rather than by grabbing the rest of the line is necessary, because pending tokens of the macros being executed would otherwise be grabbed as well.

If the name is different (the environment is wrapped in a user environment), we execute `\begin` with that name; the environment's end code (the user's) is extended with `\aftergroup\axp@region@finish` so that the file is closed right after the group of the environment ends.

```
\def\axp@region@readname{%
  \futurelet\axp@region@next\axp@region@readname@i}
\def\axp@region@readname@i{%
  \ifx\axp@region@next\egroup
    \expandafter\axp@region@readname@done
  \else
    \expandafter\axp@region@readname@ii
  \fi}
\def\axp@region@readname@ii#1{%
  \edef\axp@region@name{\axp@region@name#1}\axp@region@readname}
\def\axp@region@readname@done{%
  \afterassignment\axp@region@readname@go\let\axp@region@next= }
\def\axp@region@readname@go{%
  \ifx\axp@region@name\axp@replay@env
    \expandafter\axp@region@go@direct
  \else
    \expandafter\axp@region@go@wrapped
  \fi}
\def\axp@region@go@direct{%
  \expandafter\let\csname axp@region@isregion@\axp@replay@env\endcsname
    \@empty
  \let\end\axp@region@endcmd
  \axp@replay@begin}
\def\axp@region@go@wrapped{%
  \axp@region@wrappedtrue
  \expandafter\let\expandafter\axp@region@savedwrappedend
    \csname end\axp@region@name\endcsname
  \expandafter\def\csname end\axp@region@name\endcsname{%
    \axp@region@savedwrappedend\aftergroup\axp@region@finish}%
  \expandafter\begin\expandafter{\axp@region@name}}
```

```
\def\axp@region@endcmd#1{%
  \ifcsname axp@region@isregion@#1\endcsname
    \expandafter\axp@region@endregion
  \else
    \expandafter\axp@region@savedendcmd
  \fi
  {#1}}
\def\axp@region@endregion#1{\axp@replay@end\axp@region@finish}
\def\axp@region@finish{\axp@region@setcatcodes{9}\endinput}
```

`\axp@region@proof` `\axp@region@rp` Finally, the begin codes used in the appendix, which read the optional argument of the environment from the source. For proofs, the first argument is the proof environment to use and the second is `1` if the proof is to be titled by `\axp@apxproofstart` (this is decided in the main text, and thus passed along). The optional argument is looked for exactly as in the main text, without skipping spaces. The end of the `\begin{proof}` line was not copied to the `.axp` file, and we `\ignorespaces` it, which also prevents the proof environment from mistaking a `[` at the start of the body for an optional argument.

```
\def\axp@region@proof#1#2{%
  \ltx@ifnextchar@nospace[%
    {\axp@region@proof@opt{#1}}{\axp@region@proof@noopt{#1}{#2}}}
\def\axp@region@proof@opt#1[#2]{\begin{#1}[#2]\ignorespaces}
\def\axp@region@proof@noopt#1#2{%
  \ifnum#2=\@ne
    \expandafter\axp@apxproofstart
  \else
    \expandafter\begin
  \fi
  {#1}\ignorespaces}
```

For `toappendix` environments, we typeset the pending appendix section heading, if any, using `\axp@section` which looks at what follows in the source, and otherwise skip the end of the `\begin{toappendix}` line, which was not copied to the `.axp` file.

```
\def\axp@region@toappendix{%
  \ifx\axp@region@pendingsection\relax
    \expandafter\ignorespaces
  \else
    \expandafter\axp@region@toappendix@section
  \fi}
\def\axp@region@toappendix@section{%
  \edef\axp@tmp{\expandonce\axp@region@pendingsection}%
  \global\let\axp@region@pendingsection\relax
  \expandafter\axp@section\expandafter{\axp@tmp}%
}
```

For repeated theorems, the arguments are the theorem name and the value of `axp@rpcounter`; the heading is built as in `\axp@newtheoremrep` below.

```
\def\axp@region@rp#1#2{%
  \@ifnextchar[{\axp@region@rp@i{#1}{#2}}{\axp@region@rp@i{#1}{#2}[]}}
\def\axp@region@rp@i#1#2[#3]{%
  \begin{axp@#1rp}[%
    \axp@refstar{axp@r#2}%
    \@ifnotempty{#3}{%
```

```
      \ifdefined\theopargself\else
        \ifdefined\@acmplainnotefont
          \@acmplainnotefont
        \else
          \ifdefined\thm@notefont\the\thm@notefont\fi
        \fi
      \fi
      {} ({#3})%
    }%
  ]%
  \axp@forward@target{axp@fw@r#2}{}%
  \axp@redefinelabels
}
```

8.5 Inline Compilation Mode

```
\ifthenelse{\equal{\axp@appendix}{inline}}{
```

`\axp@newtheoremrep` In inline mode, `\axp@newtheoremrep` uses `\axp@newtheoremrep@definetheorem` to define the regular theorem environment and creates a repeated theorem environment that behaves exactly as the regular theorem environment, while calling `\axp@redefinelabels` to make sure that `-axp` variants of equation counters are defined. With the `synctex` option, the environment is defined with `\newenvironment` so that its body is typeset directly from the source file (see Section 8.4).

```
  \def\axp@newtheoremrep#1[#2]#3[#4]{%
    \axp@newtheoremrep@definetheorem{#1}{#2}{#3}{#4}%
    \ifbool{axp@synctex}{%
      \newenvironment{#1rep}[1][]{%
        \ifx\relax##1\relax
          \begin{#1}\axp@redefinelabels
        \else
          \begin{#1}[##1]\axp@redefinelabels
        \fi
      }{\end{#1}}%
    }{%
      \NewEnviron{#1rep}[1][]{%
        \ifx\relax##1\relax
          \begin{#1}\axp@redefinelabels\BODY\end{#1}%
        \else
          \begin{#1}[##1]\axp@redefinelabels\BODY\end{#1}%
        \fi
      }%
    }%
  }
```

`inlineproof` (*env.*) In inline mode, these environments behave like the regular `proof` environment.
`nestedproof` (*env.*)
`appendixproof` (*env.*)

```
  \let\inlineproof\proof
  \let\endinlineproof\endproof
  \let\nestedproof\proof
  \let\endnestedproof\endproof
  \let\appendixproof\proof
  \let\endappendixproof\endproof
```

`toappendix` (*env.*) In inline mode, this environment and these macros are no-ops.
`\noproofinappendix`
`\nosectionappendix`

```
\newenvironment{toappendix}{}{}
\let\noproofinappendix\relax
\let\nosectionappendix\relax
}
```

8.6 Append, Strip, or Chapter End Compilation Modes

```
{
```

We now deal with the case where apxproof really does something useful: either append the appendix material to the document, or strip it entirely.

8.6.1 Auxiliary File for the Appendix

`\axp@proofsfile` We open a new auxiliary file, with extension `.axp`, where the appendix material will be dumped.

```
\AtBeginDocument{
  \newwrite\axp@proofsfile
  \ifthenelse{\equal{\axp@appendix}{chapterend}}{%
    \axp@openchapterfile
  }{%
    \immediate\openout\axp@proofsfile=\jobname.axp
  }%
}
```

`proof` (*env.*) `\section` At the beginning of this file, we make `@` a regular character (since it will be used in several places for internal names) and reestablish the original definition of the `proof` environment and the `\section` macro.

```
\newcommand{\axp@writeproofsfilepreamble}{%
  \immediate\write\axp@proofsfile{%
    \noexpand\makeatletter
    \noexpand\let\noexpand\proof\noexpand\axp@replayproof
    \noexpand\let\noexpand\endproof\noexpand\endaxp@oldproof
    \noexpand\let\noexpand\claimproof\noexpand\axp@oldclaimproof
    \noexpand\let\noexpand\endclaimproof\noexpand\endaxp@oldclaimproof
    \noexpand\let\noexpand\section\noexpand\axp@oldsection
  }%
}
\AtBeginDocument{
  \ifthenelse{\equal{\axp@appendix}{chapterend}}{}{%
    \axp@writeproofsfilepreamble
  }%
}
```

`\axp@unactivateeightbit` We need an auxiliary macro to disable active characters that have the high bit set when writing to the `.axp` file. See https://tex.stackexchange.com/a/145361/166858

```
\def\axp@unactivateeightbit{%
  \count@=128%
  \loop
    \catcode\count@=12%
    \ifnum\count@<255%
    \advance\count@\@ne
  \repeat}
```

`axp@VerbatimOut` (*env.*)
`\FVB@axp@VerbatimOut`
`\FVE@axp@VerbatimOut`

Using the functionalities of the fancyvrb package, we define a custom verbatim environment `axp@VerbatimOut` that writes every line to the `\axp@proofsfile`. We also use the previous macro to disable active characters with the eighth bit set, and we make sure the catcode of `@` is reset for every verbatim environment, in case it is used by the user (e.g., as in the xypic package). Finally, as an additional precaution, we reset `\FV@CatCodesHook` that is for example set by the `commandchars` or `commentchar` option of `\fvset`. With the `synctex` option, the environment is still scanned line by line by fancyvrb (this is how we skip its content in the main text and find its end), but the lines are not written to the `.axp` file, which instead receives an `\axp@replaybody` command (Section 8.4).

```
\DefineVerbatimEnvironment{axp@VerbatimOut}{axp@VerbatimOut}{}
\def\FVB@axp@VerbatimOut{%
  \@bsphack
  \begingroup
    \axp@unactivateeightbit
    \FV@DefineWhiteSpace
    \def\FV@Space{\space}%
    \FV@DefineTabOut
    \ifbool{axp@synctex}{%
      \let\FV@ProcessLine\@gobble
    }{%
      \def\FV@ProcessLine{\immediate\write\axp@proofsfile}%
    }%
    \let\FV@FontScanPrep\relax
    \let\@noligs\relax
    \def\FV@CatCodesHook{}%
    \FV@Scan}
\def\FVE@axp@VerbatimOut{%
    \immediate\write\axp@proofsfile{\noexpand\makeatletter}%
    \endgroup\@esphack}
```

`\axp@markchapterhasappendix`

No-op by default; overridden in `chapterend` mode to set a flag indicating that the current chapter has appendix content.

```
\newcommand{\axp@markchapterhasappendix}{}
```

`toappendix` (*env.*)

The entire content of this environment is put in appendix, starting a new appendix section beforehand if needed. With the `synctex` option, we record where the environment starts and, at its end, write the replay command; in the appendix, the begin code is `\axp@region@toappendix` and there is no end code. As `\axp@section` looks at what follows it to decide whether to typeset a heading, and the content is now only available once the environment is replayed, we have the section command written as `\axp@region@section`, which merely stores the heading for `\axp@region@toappendix` to handle.

```
\newenvironment{toappendix}
  {\axp@markchapterhasappendix
   \ifbool{axp@synctex}{%
     \def\axp@sectioncmd{\axp@region@section}%
     \axp@writesection
     \axp@region@record
   }{%
     \axp@writesection
   }%
```

```
    \axp@VerbatimOut}
  {\endaxp@VerbatimOut
   \ifbool{axp@synctex}{%
     \axp@region@write{\noexpand\axp@region@toappendix}{\relax}%
   }{}}
```

8.6.2 Definition of New Theorems

`axp@seenreptheorem` Used to indicate whether a repeated theorem was just typeset, without its proof.

```
\newtoggle{axp@seenreptheorem}
```

`axp@rpcounter` Sequentially incremented for every repeated theorem, used to create labels.

```
\newcounter{axp@rpcounter}
```

`axp@equation` Used to save the value of the `equation` counter, when `repeqn` is set to `same`.
`axp@equationx`

```
\newcounter{axp@equation}
\newcounter{axp@equationx}
```

`axp@newtheoremrep` With first argument `foobar`, we use `\axp@newtheoremrep@definetheorem` to define the regular version of the theorem `foobar`. We then patch `\begin{foobar}` so as not to expect a proof in the appendix and define an internal theorem `axp@foobarrp` that will be used in the appendix to restate the existing theorem. In the starred mode, the theorem is not restated in the appendix, so we do not define the `axp@foobarrp` environment.

```
\def\axp@newtheoremrep#1[#2]#3[#4]{%
  \axp@newtheoremrep@definetheorem{#1}{#2}{#3}{#4}%
  \expandafter\pretocmd\csname #1\endcsname{\noproofinappendix}{}{}%
  \ifcsdef{axp@proofof@#1}{}{%
    \axp@newtheorem*{axp@#1rp}{#3}%
  }%
  \axp@forward@setup{#1}{#2}{#3}{#4}%
```

We then define a `foobarrep` environment, whose begin code and code writing the appendix material are the two macros described below, `\axp@newtheoremrep@begin` and `\axp@newtheoremrep@end`. Normally, the body of the environment is collected by environ, typeset, and stored in a macro for the appendix. With the `synctex` option, the environment is instead an ordinary one, so that its body is typeset directly from the source file, and the appendix will re-read the body from there (Section 8.4); we first record where the environment starts.

```
  \ifbool{axp@synctex}{%
    \newenvironment{#1rep}[1][]{%
      \axp@region@record
      \axp@newtheoremrep@begin{#1}{##1}%
    }{%
      \end{#1}%
      \axp@newtheoremrep@end{#1}{#3}{}%
    }%
  }{%
    \NewEnviron{#1rep}[1][]{%
      \axp@newtheoremrep@begin{#1}{##1}%
      \BODY\end{#1}%
      \global\expandafter\let\csname rplet\roman{axp@rpcounter}%
                                \endcsname
```

```
      \BODY
      \axp@newtheoremrep@end{#1}{#3}{##1}%
    }%
  }%
}
```

`\axp@newtheoremrep@begin` The begin code of a `foobarrep` environment increments the `axp@rpcounter` and typesets the regular `foobar` theorem with a label derived from the counter, along with a possible custom command to identify repeated theorems. We distinguish the case when the theorem argument has a note and when it does not. We save the equation counter before typesetting the theorem environment, to reset it to the same value in the repeated environment when `repeqn` is set to `same`.

```
\def\axp@newtheoremrep@begin#1#2{%
  \ifthenelse{\equal{\axp@repeqn}{same}}{%
    \setcounter{axp@equation}{\value{equation}}%
  }{}%
  \addtocounter{axp@rpcounter}{1}%
  \ifx\relax#2\relax
    \axp@with@forward{#1}{\begin{#1}}\label{axp@r\roman{axp@rpcounter}}%
  \else
    \axp@with@forward{#1}{\begin{#1}[{#2}]}\label{axp@r\roman{axp@rpcounter}}%
  \fi
  \mainbodyrepeatedtheorem
}
```

`\axp@newtheoremrep@end` Once the theorem has been typeset, with arguments the theorem name, its title and its note (the latter is only needed when the body was collected), we set the `axp@seenreptheorem` toggle to indicate that we are looking for the proof of the theorem.

```
\def\axp@newtheoremrep@end#1#2#3{%
  \global\toggletrue{axp@seenreptheorem}%
```

Possibly after starting a new appendix section if needed, we typeset a repeated version of the theorem using the `axp@foobarrp` environment and a reference to the previously defined label. We use `\axp@redefinelabels` in this environment to avoid multiply defined labels. We have to deal in a careful way with theorem notes: we want to use a theorem note to display the number of the repeated theorem, but theorem notes are usually typeset in a much different way (different font, parentheses) than theorem headings. In the case of the Springer document classes, we use the `\theopargself` macro to disable parentheses. For other document classes, we need to manually patch the `\thmhead` command at the right time. We also specially cover the case of the ACM document class where `\@acmplainnotefont` is used instead of `\thm@notefont`.

```
  \axp@markchapterhasappendix
  \axp@writesection%
  \immediate\write\axp@proofsfile{\noexpand\makeatletter}%
  \ifthenelse{\equal{\axp@repeqn}{same}}{%
    \immediate\write\axp@proofsfile{%
      \noexpand\setcounter{axp@equationx}{\value{equation}}%
      \noexpand\setcounter{equation}{\theaxp@equation}%
    }%
  }{}%
  \ifbool{axp@forward@suppress}{%
```

```
    \global\def\axp@refstar{\ref*}
  }{%
    \global\def\axp@refstar{\ref}
  }
```

We write the auto-title state to the `.axp` file so that the following deferred proof can title itself via `\appendixproofname`. In starred mode, we also emit a forward-link target into the appendix so that hyperlinks from the inline theorem number always land somewhere defined, even if the user does not provide a proof.

```
  \immediate\write\axp@proofsfile{%
    \noexpand\gdef\noexpand\axp@nextproofof@type{#2}%
    \noexpand\gdef\noexpand\axp@nextproofof@label{axp@r\roman{axp@rpcounter}}%
  }%
  \ifcsdef{axp@proofof@#1}{%
    \immediate\write\axp@proofsfile{%
      \noexpand\axp@forward@target{axp@fw@r\roman{axp@rpcounter}}{}%
    }%
  }{%
```

In the default (unstarred) mode, we emit the repeated version of the theorem to the appendix; the forward-link target sits inside the restated theorem. In starred mode, we skip this block entirely. With the `synctex` option, the restated theorem is produced by replaying the source, and the heading is built by `\axp@region@rp`; otherwise we write it out here, followed by the stored body.

```
    \immediate\write\axp@proofsfile{{%
      \ifdefined\theopargself
        \noexpand\theopargself
      \else
        \noexpand\pretocmd{\noexpand\@begintheorem}{%
          \noexpand\patchcmd{\noexpand\thmhead}{\noexpand\@acmplainnotefont}{}{}{}%
          \noexpand\patchcmd{\noexpand\thmhead}{\noexpand\the\noexpand\thm@notefont}{}{}{}
          \noexpand\patchcmd{\noexpand\thmhead}{(}{}{}{}%
          \noexpand\patchcmd{\noexpand\thmhead}{)}{}{}{}%
        }{}{}
      \fi
      \ifbool{axp@synctex}{%
        \noexpand\axp@replaybody{\axp@region@file}{\axp@region@line}%
          {\the\inputlineno}{\axp@region@env}%
          {\noexpand\axp@region@rp{#1}{\roman{axp@rpcounter}}}%
          {\noexpand\end{axp@#1rp}}{\axp@region@tag}%
      }{%
        \noexpand\begin{axp@#1rp}
          [%
            \noexpand\axp@refstar{axp@r\roman{axp@rpcounter}}%
            \@ifnotempty{#3}{%
              \ifdefined\theopargself
              \else
                \ifdefined\@acmplainnotefont
                  \noexpand\@acmplainnotefont
                \else
                  \noexpand\ifdefined\noexpand\thm@notefont
                    \noexpand\the\noexpand\thm@notefont
                  \noexpand\fi
                \fi
```

```
                \fi
                {} (\unexpanded{{#3}})%
              }%
            ]%
            \noexpand\axp@forward@target{axp@fw@r\roman{axp@rpcounter}}{}%
            \noexpand\axp@redefinelabels
            \expandafter\noexpand\csname rplet\roman{axp@rpcounter}%
                                \endcsname
          \noexpand\end{axp@#1rp}
        }%
      }}%
    }%
    \ifthenelse{\equal{\axp@repeqn}{same}}{%
      \immediate\write\axp@proofsfile{%
        \noexpand\setcounter{equation}{\value{axp@equationx}}%
      }%
    }{}%
  }
```

8.6.3 Forward-Linking Mechanism

When hyperref is loaded, `foobarrep` environments in the main text have their number link to their repetition in the appendix.

`\axp@with@forward` In order to make the number of the `foobarrep` theorem a link to its repeated version, we temporarily redefine the `\thefoobar` command, or, if we inherited the counter from a `bazbar` environment, the `\thebazbar` command. This seems to be the only robust way to make the number a `\hyperlink`, without adding extensive dependence on internals of `amsthm`, the builtin `\newtheorem` and possibly document-class specific definitions.

In order to allow users to redefine `\thefoobar` without breaking this feature, we redefine `\thefoobar` only for the duration of the `\begin{foobar}` form, resetting it to the old value as soon as possible.

Redefining `\thefoobar` has the side effect of changing `\newlabel` entries in the `.aux` file, so we need to to be able to disable addition of the hyperlink, which is why we use an intermediate `\axp@forward@link{`⟨*target*⟩`}{`⟨*text*⟩`}` macro, We also redefine `\theHfoobar` which is used by `hyperref` but not defined if `hyperref` was loaded after `\newtheoremrep` was used and `\protect` it to output it verbatim into the `.aux` file.

These hyperlinks are of course disabled in the `strip` compilation mode.

```
\newcommand{\axp@with@forward}[2]{%
  \ifthenelse{\equal{\axp@appendix}{strip}}{#2}{
    \global\booltrue{axp@forward}%
    \ifcsundef{axp@old@the\csname axp@cn@#1\endcsname}{%
      \csletcs{axp@old@the\csname axp@cn@#1\endcsname}{the\csname axp@cn@#1\endcsname}%
      \csletcs{theH\csname axp@cn@#1\endcsname}{the\csname axp@cn@#1\endcsname}%
      \csdef{the\csname axp@cn@#1\endcsname}{%
        \protect\axp@forward@link{axp@fw@r\roman{axp@rpcounter}}%
          {\csname axp@old@the\csname axp@cn@#1\endcsname\endcsname}%
      }%
    }{}%
    #2%
```

```
      \ifcsdef{axp@old@the\csname axp@cn@#1\endcsname}{%
        \csletcs{the\csname axp@cn@#1\endcsname}{axp@old@the\csname axp@cn@#1\endcsname}%
      }{}%
      \global\boolfalse{axp@forward}
    }}%
```

`\axp@forward@link` `axp@forward` Dummy macro, for handling the unwanted change of the `\newlabel` entry in the `.aux` file caused by changing the definition of `\thefoobar`. We also use `\texorpdfstring` to detect whether the reference appears within a PDF bookmark, in which case we skip the test altogether.

```
\newbool{axp@forward}
\newcommand{\axp@forward@link}[2]{%
  \ifdefined\texorpdfstring\else\newcommand\texorpdfstring[2]{#2}\fi
  \texorpdfstring{%
    \ifboolexpr{ bool {axp@forward} and not bool {axp@forward@suppress} }{%
      \ifcsdef{hyperlink}{%
        \hyperlink{#1}{#2}%
      }{%
        #2%
      }%
    }{%
      #2%
    }%
  }{%
    #2%
  }%
}%
```

`\axp@forward@target` Provides the needed `\hypertarget`. Intended to be written to the `.axp` file.

```
\newcommand{\axp@forward@target}[2]{%
  \ifcsname hypertarget\endcsname
    \hypertarget{#1}{#2}%
  \else
    #2%
  \fi
}
```

`\axp@forward@setup` In order to support counter inheritance with the first optional argument of `\newtheoremrep`, we need access to the name of the counter. For compliance with the behavior of `\@axp@newtheorem`, the first optional argument (`#2`) is ignored if the second optional argument (`#4`) is given. We also check whether `aliascnt` was used to define this counter; if so, we use the original name as the counter name to handle the way such aliases work.

```
\newcommand{\axp@forward@setup}[4]{%
  \csedef{axp@cn@#1}{\ifblank{#4}{\ifblank{#2}{#1}{\ifcsdef{AC@cnt@#1}{#1}{#2}}}{#1}}%
}
```

8.6.4 Proof Environments

`\noproofinappendix` Utility macro that toggles `axp@seenreptheorem` to false.

```
\newcommand\noproofinappendix{%
  \global\togglefalse{axp@seenreptheorem}%
}
```

`\axp@apxproofstart` Helper invoked from the `.axp` file at appendix typeset time to open a deferred proof. Reads the `\axp@nextproofof@*` state (set by the preceding `theoremrep`) and consults `\appendixproofname` to decide whether to pass an optional title argument.

```
\gdef\axp@nextproofof@type{}
\gdef\axp@nextproofof@label{}
```

`\apxproofhook` A hook fired, inside the proof environment, each time a deferred proof is typeset in the appendix (and also for an inline proof whose title names its result through a `\ref`, see `\axp@inlineproofhook` below). Its argument is the label of the theorem the proof establishes: the automatic `axp@r`⟨*n*⟩ label of the preceding `theoremrep` for a deferred proof, or the target of the title's `\ref` for a titled inline proof. A companion package (such as proofgraph) can thereby attribute, to the right result, the references the proof makes (which are only executed at typeset time). It is a no-op by default.

```
\providecommand\apxproofhook[1]{}
\def\axp@apxproofstart#1{%
  \protected@xdef\axp@apxproofstart@title{%
    \appendixproofname{\axp@nextproofof@type}%
                      {\noexpand\ref{\axp@nextproofof@label}}%
  }%
  \ifx\axp@apxproofstart@title\@empty
    \begin{#1}%
  \else
    \expandafter\axp@apxproofstart@withtitle
      \expandafter{\axp@apxproofstart@title}{#1}%
  \fi
  \apxproofhook{\axp@nextproofof@label}%
}
\def\axp@apxproofstart@withtitle#1#2{\begin{#2}[#1]}
```

`\axp@inlineproofhook` A proof that is *not* deferred (no preceding `theoremrep`, so it is typeset inline) never goes through `\axp@apxproofstart`, hence never fires `\apxproofhook`. When such a proof carries an explicit title that names its result through a `\ref` (the idiom `\begin{proof}[of Theorem~\ref{thm:x}]`), we fire the hook here, inside the proof, with that target, so a companion package can attribute the proof to it. We only fire when a cross-reference is found in the title (the result is then unambiguous). The title is scanned for the first `\ref` (or `\cref`, `\Cref`, `\autoref`, `\vref`, `\zcref`) by pure token matching, never typesetting it: a title typically contains an active `~` and may carry other fragile material, which an auxiliary typeset pass would perturb or, with `\protect` neutralised, send into an infinite loop. zref-clever's `\zcref` is the only one of these that is not simply ⟨*cs*⟩`{`⟨*label*⟩`}`: it is `\zcref*[`⟨*options*⟩`]{`⟨*labels*⟩`}`, whose star and options are to be skipped and whose mandatory argument is a comma-separated list, of which we keep the first label. Each grabber is handed the whole remainder of the title and swallows it up to `\axp@refstop`, so that nothing of it is left to be typeset.

The matching is `\in@`, hence TEX delimited-argument matching, which only sees tokens at brace level 0. A cross-reference buried in a brace group is therefore not found, and the hook does not fire for such a title. This is the case when the cross-reference takes an optional argument of its own, since LATEX then requires it to be braced within the optional title, as in

`\begin{proof}[Proof of {\zcref[noname]{thm:x}}]`. Nothing else is affected: the title is typeset as given, and `\apxproofhook` is a no-op unless a companion package has defined it.

```
\newif\ifaxp@inlineproofof
\long\def\axp@inlineproofhook#1{%
  \global\axp@inlineproofoffalse
  \axp@tryrefcmd\ref{#1}%
  \ifaxp@inlineproofof\else\axp@tryrefcmd\cref{#1}\fi
  \ifaxp@inlineproofof\else\axp@tryrefcmd\Cref{#1}\fi
  \ifaxp@inlineproofof\else\axp@tryrefcmd\autoref{#1}\fi
  \ifaxp@inlineproofof\else\axp@tryrefcmd\vref{#1}\fi
  \ifaxp@inlineproofof\else
    \axp@tryrefcmd@\zcref\axp@grabzcarg{#1}%
  \fi
  \ifaxp@inlineproofof\apxproofhook{\axp@inlineproofof@label}\fi
}
\let\axp@refmark\relax
\long\def\axp@tryrefcmd#1#2{\axp@tryrefcmd@#1\axp@grabarg{#2}}
\long\def\axp@tryrefcmd@#1#2#3{%
  \in@{#1}{#3}%
  \ifin@
    \long\def\axp@grabref##1#1##2\axp@refstop{#2##2\axp@refstop}%
    \axp@grabref#3\axp@refmark\axp@refstop
  \fi
}
\long\def\axp@grabarg#1#2\axp@refstop{%
  \global\axp@inlineproofoftrue\xdef\axp@inlineproofof@label{#1}}
\def\axp@grabzcarg{\@ifstar\axp@grabzcarg@i\axp@grabzcarg@i}
\def\axp@grabzcarg@i{\@ifnextchar[\axp@grabzcarg@opt\axp@grabzcarg@ii}%]
\def\axp@grabzcarg@opt[#1]{\axp@grabzcarg@ii}
\long\def\axp@grabzcarg@ii#1{%
  \axp@grabzcarg@iii#1,\@nil
  \axp@gobbletorefstop}
\long\def\axp@grabzcarg@iii#1,#2\@nil{%
  \global\axp@inlineproofoftrue\xdef\axp@inlineproofof@label{#1}}
\long\def\axp@gobbletorefstop#1\axp@refstop{}
```

Proofs placed by hand in a `toappendix` environment are written verbatim to the `.axp` file and replayed there with `\proof` let to `\axp@oldproof` (see `\axp@writeproofsfilepreamble`). Such proofs therefore never reach `\axp@@proof`. So that a titled one still fires `\apxproofhook`, we let, at replay time, `\proof` to `\axp@replayproof`, which reproduces the inline behaviour: open the `amsthm` proof, then fire the hook for the title's `\ref`.

```
\def\axp@replayproof{\@ifnextchar[{\axp@replayproof@opt}{\axp@oldproof}}
\def\axp@replayproof@opt[#1]{\axp@oldproof[#1]\axp@inlineproofhook{#1}}
```

`appendixproof` (*env.*) We dump the content of this in appendix, within an original `proof` environment, possibly after creating a new appendix section. We support optional arguments in proofs, but we need to be careful not to gobble any newline character, that would make it impossible to process the first line with fancyvrb. When the proof is auto-routed from a `theoremrep` (the toggle `axp@seenreptheorem` is true at this point and no explicit optional argument was given), we delegate the proof opening to `\axp@apxproofstart` so that `\appendixproofname` is consulted at appendix

typeset time. A user-supplied optional argument or a manual `\appendixproof` call (toggle false) bypasses this and uses the standard "Proof." heading. With the `synctex` option, nothing is written when the proof starts: we record its position and whether it is to be titled by `\axp@apxproofstart`, and the replay command written at the end (by `\axp@appendixproof@write`, with the name of the proof environment to use in the appendix) carries this information to `\axp@region@proof`, which reads the optional argument, if any, from the source.

```
\def\appendixproof{\catcode`\^^M=\active\@ifnextchar[{\catcode`\^^M=5\@@appendixproof[axp@
\def\@appendixproof[#1]%
  {%
    \axp@markchapterhasappendix
    \axp@writesection
    \iftoggle{axp@seenreptheorem}{%
      \ifbool{axp@synctex}{%
        \axp@region@record
        \def\axp@region@flag{1}%
      }{%
        \immediate\write\axp@proofsfile{%
          \noexpand\makeatletter\noexpand\axp@apxproofstart{#1}\noexpand\makeatother%
        }%
      }%
    }{%
      \ifbool{axp@synctex}{%
        \axp@region@record
        \def\axp@region@flag{0}%
      }{%
        \immediate\write\axp@proofsfile{%
          \noexpand\makeatletter\noexpand\begin{#1}\noexpand\makeatother%
        }%
      }%
    }%
    \axp@VerbatimOut
  }
\def\@@appendixproof[#1][#2]%
  {%
    \axp@markchapterhasappendix
    \axp@writesection
    \ifbool{axp@synctex}{%
      \axp@region@record
      \def\axp@region@flag{0}%
    }{%
      \immediate\write\axp@proofsfile{%
        \noexpand\makeatletter\noexpand\begin{#1}[\unexpanded{#2}]\noexpand\makeatother%
      }%
    }%
    \axp@VerbatimOut
  }
\def\axp@appendixproof@write#1{%
  \ifbool{axp@synctex}{%
    \axp@region@write
      {\noexpand\axp@region@proof{#1}{\axp@region@flag}}%
      {\noexpand\end{#1}}%
  }{%
```

```
      \immediate\write\axp@proofsfile{%
        \noexpand\end{#1}%
      }%
    }%
  }
  \def\endappendixproof
    {%
      \endaxp@VerbatimOut
      \axp@appendixproof@write{axp@oldproof}%
      \noproofinappendix
    }
```

`proof` (*env.*) This environment either puts the proof in appendix, if we are after a repeated theorem without its proof, or inlines it otherwise. We support optional arguments in proofs, but we need to be careful not to gobble any newline character, that would make it impossible to process the first line with fancyvrb.

```
  \def\proof{\catcode`\^^M=\active\ltx@ifnextchar@nospace[{\catcode`\^^M=5\axp@@proof}{\catc
  \def\axp@proof
    {%
      \iftoggle{axp@seenreptheorem}{%
        \appendixproof
      }{%
        \axp@oldproof
      }%
    }
  \def\axp@@proof[#1]%
    {%
      \iftoggle{axp@seenreptheorem}{%
        \appendixproof[#1]
      }{%
        \axp@oldproof[#1]%
        \axp@inlineproofhook{#1}%
      }%
    }
  \def\endproof
    {%
      \iftoggle{axp@seenreptheorem}{%
        \endappendixproof
      }{%
        \endaxp@oldproof
      }%
    }
```

`claimproof` (*env.*) If the `claimproof` environment exists (lipics document class or user-defined), we also redefine it, in the same way.

```
  \AtBeginDocument{
    \ifdefined\claimproof
    \let\axp@oldclaimproof\claimproof
    \let\endaxp@oldclaimproof\endclaimproof
    \def\claimproof{\catcode`\^^M=\active\ltx@ifnextchar@nospace[{\catcode`\^^M=5\axp@@claim
    \def\appendixclaimproof{\catcode`\^^M=\active\@ifnextchar[{\catcode`\^^M=5\@@appendixpro
    \def\axp@claimproof
      {%
        \iftoggle{axp@seenreptheorem}{%
```

```
          \appendixclaimproof
        }{%
          \axp@oldclaimproof
        }%
      }
    \def\axp@@claimproof[#1]%
      {%
        \iftoggle{axp@seenreptheorem}{%
          \appendixclaimproof[#1]%
        }{%
          \axp@oldclaimproof[#1]%
        }%
      }
    \def\endclaimproof
      {%
        \iftoggle{axp@seenreptheorem}{%
          \endappendixclaimproof
        }{%
          \endaxp@oldclaimproof
        }%
      }
    \def\endappendixclaimproof
    {%
      \endaxp@VerbatimOut
      \axp@appendixproof@write{axp@oldclaimproof}%
      \noproofinappendix
    }
    \fi
}
```

`inlineproof` (*env.*) `nestedproof` (*env.*) These two environments are synonyms for the original `proof` environment.

```
\let\inlineproof\axp@oldproof
\let\endinlineproof\endaxp@oldproof
\let\nestedproof\axp@oldproof
\let\endnestedproof\endaxp@oldproof
```

8.6.5 Section Management

`axp@seccounter` Sequentially incremented for every section, used to create labels.

```
\newcounter{axp@seccounter}
```

`\axp@sectitle` Saves the title of the last encountered section.

```
\def\axp@sectitle{}
```

`\axp@section` `\axp@@sectionskippar` `\axp@@sectiontestinput` `\axp@@sectiontestinclude` `\axp@@sectionfinishinclude` `\axp@@sectiontestsection` This command `\axp@section` behaves similarly to `\axp@oldsection`, except that it first tests whether a `\section` follows, and if so, does not produce anything. This is useful to avoid producing empty sections in the appendix. Using the catchfile package, we also check whether a `\section` is within an `\input` or `\include` that immediately follows. We also skip any `\par` tokens (produced by blank lines) before performing the check.

```
\def\axp@section#1{%
  \@ifnextchar\par
    {\axp@@sectionskippar{#1}}%
```

```
    {\@ifnextchar\input
      {\axp@@sectiontestinput{#1}}%
      {\@ifnextchar\include
        {\axp@@sectiontestinclude{#1}}%
        {\axp@@sectiontestsection{#1}}}}%
}
\def\axp@@sectionskippar#1\par{\axp@section{#1}}
\def\axp@@sectiontestinput#1\input#2{%
  \CatchFileDef{\axp@tmp}{#2}{}%
  \def\axp@tmptmp{\axp@@sectiontestsection{#1}}%
  \expandafter\axp@tmptmp\axp@tmp%
}
\def\axp@@sectiontestinclude#1\include#2{%
  \set@curr@file{#2}%
  \edef\axp@tmp{\@strip@tex@ext\@curr@file.tex}%
  \expandafter\CatchFileDef\expandafter\axp@tmp\expandafter{\axp@tmp}{}%
  \def\axp@tmptmp{\axp@@sectionfinishinclude{#1}{#2}}%
  \expandafter\axp@tmptmp\axp@tmp\@nil
}
\long\def\axp@@sectionfinishinclude#1#2#3#4\@nil{%
  \ifx#3\section
    \makeatother
  \else
    \axp@oldsection{#1}\makeatother
  \fi
  \include{#2}%
}
\def\axp@@sectiontestsection#1{\@ifnextchar\section{\makeatother}{\axp@oldsection{#1}\make
```

`\axp@oldsection` `\section` `\@section` `\@@section`

We redefine the `\section` command to create a label based on `axp@seccounter` and to store its title in `\axp@sectitle`. In order to support starred and unstarred versions, as well as the optional short-title argument, the intermediate macros `\@section` and `\@@section` are needed.

```
\let\axp@oldsection\section
\def\section{\@ifstar\@section\@@section}
\newcommand{\@section}[2][\relax]{\axp@@@section{*}{#1}{#2}}%
\newcommand{\@@section}[2][\relax]{\axp@@@section{}{#1}{#2}}%
\newcommand{\axp@@@section}[3]{%
  \global\def\axp@sectitle{#3}%
  \ifx\relax#2\relax
    \axp@oldsection#1{#3}%
  \else
    \axp@oldsection#1[{#2}]{#3}%
  \fi
  \addtocounter{axp@seccounter}{1}%
  \label{axp@s\roman{axp@seccounter}}%
}
```

`\nosectionappendix`

We remove the current section title, to indicate no section should be created in the appendix.

```
\newcommand{\nosectionappendix}{
  \global\def\axp@sectitle{}%
}
```

\axp@writesection If \axp@sectitle is not empty, we create a new section in the appendix, referring to the main text section.

Here, we wrap \ref{axp@si} into \axp@protectref@i, in order to protect the label name from wrongly being converted to uppercase, e.g., in fancyhdr with \pagestyle{fancy}.

This macro is defined both in the .aux file (in order to ensure availability when typesetting the \tableofcontents), and immediately before typesetting the appendix section (to ensure availability in the \section command).

The command written is normally \axp@section; the toappendix environment substitutes \axp@region@section for it with the synctex option, so that the heading is decided when the material is replayed (Section 8.4).

```
\def\axp@sectioncmd{\axp@section}
\newcommand\axp@writesection{%
  \ifx\axp@sectitle\@empty
  \else
    \edef\axp@tmp{%
      \noexpand\global\noexpand\def
      \expandonce{\csname axp@protectref@\roman{axp@seccounter}\endcsname}{%
        \noexpand\ref{axp@s\roman{axp@seccounter}}%
      }%
    }%
    \immediate\write\@auxout{\expandonce\axp@tmp}
    \immediate\write\axp@proofsfile{%
      \expandonce\axp@tmp^^J%
      \expandafter\noexpand\axp@sectioncmd{%
        \noexpand\appendixsectionformat{%
          \protect
          \expandonce{\csname axp@protectref@\roman{axp@seccounter}\endcsname}%
        }{\expandonce\axp@sectitle}%
      }%
    }%
    \nosectionappendix
  \fi
}
```

Finally, in a somewhat ad hoc manner, we disable the whole section management for \tableofcontents, which may be typeset using a section heading, but for which automatic section management does not make sense.

\axp@oldtableofcontents
\tableofcontents

```
\let\axp@oldtableofcontents\tableofcontents
\def\tableofcontents{{\let\section\axp@oldsection\axp@oldtableofcontents}}
```

8.6.6 Append Compilation Mode

```
\ifthenelse{\equal{\axp@appendix}{append}}{
```

\axp@oldbibliography
\bibliography In bibengine=bibtex mode, when the bibliography option is set to separate, we need to set the appendix bibliography source to be the same as that of the main text, thanks to bibunits's \defaultbibliography macro. In bibengine=biblatex mode, no such rebinding is needed: biblatex resources declared with \addbibresource are visible to all refsections globally.

```
  \ifthenelse{\equal{\axp@bibliography}{separate}
```

```
                \AND\equal{\axp@bibengine}{bibtex}}{
    \let\axp@oldbibliography\bibliography
    \renewcommand\bibliography[1]{%
      \defaultbibliography{#1}%
      \axp@oldbibliography{#1}%
    }
  }{}
```

After the end of the main text, we add the appendix (after the command `\appendixprelim` is issued) within a `bibunit` environment so as to typeset a separate bibliography for the appendix (unless the `bibliography` option is set to `common`). This is done using `\pretocmd` on `\@enddocumenthook` instead of `\AtEndDocument` because we want the code to be run before any code in the `\@enddocumenthook` that has been set in the document class, as in the amsart document class. There is an extra test to ensure an empty bibliography environment is not produced. The name of the bibliography is changed to `\appendixrefname`; in most document classes, it is called `\refname` but it is occasionally (`scrartcl`, `scrreprt`) called `\bibname`. An ad-hoc test is added to fix a conflict with the natbib package which redefines `bibcite` at the end of the document.

```
  \pretocmd{\@enddocumenthook}{%
    \ifdefined\NAT@testdef
      \renewcommand\bibcite[2]{%
        \global\@namedef{b@#1\@extra@binfo}{#2}%
      }
    \fi
    \appendixprelim
    \appendix
    \ifthenelse{\equal{\axp@bibliography}{separate}}{
      \ifthenelse{\equal{\axp@bibengine}{bibtex}}{
        \begin{bibunit}[\appendixbibliographystyle]
      }{
        \begin{refsection}
      }
    }{}
      \immediate\closeout\axp@proofsfile
      \input{\jobname.axp}
    \ifthenelse{\equal{\axp@bibliography}{separate}}{
      \ifthenelse{\equal{\axp@bibengine}{bibtex}}{
        \ifdefined\refname
          \renewcommand{\refname}{\appendixrefname}
        \else\ifdefined\bibname
          \renewcommand{\bibname}{\appendixrefname}
        \fi\fi
        \let\axp@oldthebibliography\thebibliography
        \let\endaxp@oldthebibliography\endthebibliography
        \renewenvironment{thebibliography}[1]{%
          \def\axp@tmp{#1}%
          \ifx\axp@tmp\empty
            \gdef\axp@noappendixbibliography1\relax
          \else
            \begin{axp@oldthebibliography}{#1}%
          \fi
        }{%
```

```
            \ifdefined\axp@noappendixbibliography\relax\else\end{axp@oldthebibliography}%
          \fi}
          \appendixbibliographyprelim
          \putbib
          \end{bibunit}
        }{
          \appendixbibliographyprelim
          \printbibliography[title=\appendixrefname]
          \end{refsection}
        }
        \ifdefined\NAT@testdef
          \let\bibcite\NAT@testdef
        \fi
      }{}
    }{}{}
}{}
```

8.6.7 Chapter End Compilation Mode

In `chapterend` mode, appendix material for each chapter is gathered in a per-chapter auxiliary file and emitted immediately after that chapter's content (and bibliography). This requires the appendix package for its `subappendices` environment.

```
\ifthenelse{\equal{\axp@appendix}{chapterend}}{
  \RequirePackage{appendix}
```

`\chapterappendixprelim` `\axp@beginchapterappendix` `\axp@endchapterappendix` User-customizable hooks surrounding each chapter's appendix block. By default they open and close a `subappendices` environment.

```
  \newcommand{\chapterappendixprelim}{\clearpage}
  \newcommand{\axp@beginchapterappendix}{%
    \chapterappendixprelim
    \begin{subappendices}%
  }
  \newcommand{\axp@endchapterappendix}{%
    \end{subappendices}%
  }
```

`axp@chaptercounter` `axp@chapterhasappendix` `\axp@currentaxpfile` Counter for unique per-chapter file names, a boolean tracking whether the current chapter has any appendix content, and the current file name.

```
  \newcounter{axp@chaptercounter}
  \newbool{axp@chapterhasappendix}
  \def\axp@currentaxpfile{}
```

`\axp@markchapterhasappendix` Override the no-op to set the boolean when content is written.

```
  \renewcommand{\axp@markchapterhasappendix}{%
    \global\booltrue{axp@chapterhasappendix}%
  }
```

`\axp@openchapterfile` Open a fresh auxiliary file for the current chapter. We write `\begingroup` first so that all the preamble `\let` assignments (and any catcode changes from `\makeatletter`) are scoped to this chapter's appendix and do not leak into the rest of the document when the file is `\input`'d. We also reset the section-title

tracker and the repeated-theorem toggle so that state from the previous chapter cannot bleed into the new file.

```
\newcommand{\axp@openchapterfile}{%
  \edef\axp@currentaxpfile{\jobname-\arabic{axp@chaptercounter}.axp}%
  \immediate\openout\axp@proofsfile=\axp@currentaxpfile
  \immediate\write\axp@proofsfile{\noexpand\begingroup}%
  \axp@writeproofsfilepreamble
  \global\boolfalse{axp@chapterhasappendix}%
  \global\def\axp@sectitle{}%
  \global\togglefalse{axp@seenreptheorem}%
}
```

`\axp@flushchapterappendix` Close the current chapter's auxiliary file (appending `\endgroup` to close the group opened in `\axp@openchapterfile`) and, if it contains any content, emit it inside a `subappendices` block.

```
\newcommand{\axp@flushchapterappendix}{%
  \immediate\write\axp@proofsfile{\noexpand\endgroup}%
  \immediate\closeout\axp@proofsfile
  \ifbool{axp@chapterhasappendix}{%
    \axp@beginchapterappendix
    \input{\axp@currentaxpfile}%
    \axp@endchapterappendix
  }{}%
}
```

`\axp@chapterboundary` `\flushchapterappendix` `\axp@chapterboundary` flushes the previous chapter's auxiliary file, increments the counter, and opens the next one. It is used as a hook for `\chapter`, `\part`, and the `book`-class matter-switching commands `\frontmatter`, `\mainmatter`, `\backmatter`, so that per-chapter appendices are emitted before any of these structural transitions (otherwise the appendix of the last chapter of a part/matter would end up under the next part/matter). `\flushchapterappendix` is a public alias for the same operation, which can be invoked manually wherever an appendix flush boundary is needed (e.g., before a bibliography that does not produce its own chapter heading).

```
\newcommand{\axp@chapterboundary}{%
  \axp@flushchapterappendix
  \stepcounter{axp@chaptercounter}%
  \axp@openchapterfile
}
\let\flushchapterappendix\axp@chapterboundary
```

`\axp@hookboundary` Helper to install `\axp@chapterboundary` as a `\pretocmd` hook on a structural command (`\chapter`, `\part`, `\frontmatter`, etc.), issuing a package warning if the hook fails.

```
\newcommand{\axp@hookboundary}[1]{%
  \pretocmd{#1}{\axp@chapterboundary}{}{%
    \PackageWarning{apxproof}{%
      Could not hook into \string#1;^^J%
      chapter appendices may not be emitted correctly}}%
}
```

At the start of each `\chapter`, `\part`, or matter-switching command, run the boundary hook. At `\end{document}`, flush the last chapter's file.

```
    \AtBeginDocument{%
      \ifdef{\chapter}{%
        \axp@hookboundary{\chapter}%
      }{%
        \PackageWarning{apxproof}{%
          appendix=chapterend used but \string\chapter\space
          is not defined}%
      }%
      \ifdef{\part}{\axp@hookboundary{\part}}{}%
      \ifdef{\frontmatter}{\axp@hookboundary{\frontmatter}}{}%
      \ifdef{\mainmatter}{\axp@hookboundary{\mainmatter}}{}%
      \ifdef{\backmatter}{\axp@hookboundary{\backmatter}}{}%
    }
    \pretocmd{\@enddocumenthook}{%
      \axp@flushchapterappendix
    }{}{}
  }{}
```

8.6.8 Class-Specific Behavior

We conclude with some class-specific behavior.

ACM Document Classes (old versions, till 2017)

```
  \ifdefined\@acmtitlebox
```

We first redefine the `proofsketch` environment, which is used differently in the base class.

```
    \renewenvironment{proofsketch}{\begin{axp@oldproof}[sketch]}{\end{axp@oldproof}}
```

We adjust the styling of theorems for the needs of apxproof.

```
    \newtheoremstyle{mystyle}
      {6pt}
      {6pt}
      {\itshape}
      {10pt}
      {\scshape}
      {.}
      {.5em}
      {}
    \theoremstyle{mystyle}
```

`\thebibliography` `\refname` `\appendixrefname`

The section title of the bibliography is in uppercase in these document classes. In addition, the `\thebibliography` macro hard-codes twice the section title, so we un-hardcode it so that it can be modified in the appendix.

```
    \patchcmd{\thebibliography}{References}{\protect\refname}{}{}
    \patchcmd{\thebibliography}{References}{\protect\refname}{}{}
    \newcommand{\refname}{REFERENCES}
    \renewcommand{\appendixrefname}{REFERENCES FOR THE APPENDIX}
```

```
  \fi
```

lipcs

```
\ifdefined\lipics@opterrshort
```

`\appendixbibliographyprelim` The default bibliography in the lipics document class formatting is not compatible with the alpha bibliography style. We fix this here.

```
  \renewcommand{\appendixbibliographyprelim}{%
    \global\let\@oldbiblabel\@biblabel
    \def\@biblabel{\hspace*{-2em}\small\@oldbiblabel}%
  }

\fi
}
```

Change History

v1.0.0
General: Initial released version . . 1
v1.0.1
General: Prevent empty bibliography environment; fix typos 1
v1.0.2
`\axp@newtheoremrep@end`: Fix display of repeated theorem counter in some document classes 38
Fix missing space between repeated theorem counter and theorem note 38
v1.0.3
General: Note on entire sections in appendix 2
`\appendixbibliographyprelim`: Support for lipics-v2016 53
`proofsketch`: Ignore spaces after beginning of Proof sketch . . . 17
v1.0.4
General: More faithful theorem style for ACM templates 52
More robust coherent styling of proof sketches 52
Re-establish custom proof environments 13
Show options commented on in margin and index 1
`\appendixprelim`: Configurable appendix style 17
`\axp@bibliography`: `bibliography` option 13
v1.0.5
General: Ability to specify a sectioning counter in newtheoremrep 2
Fix compilation of proofsketch environment in inline mode . . . 4
v1.0.6
General: Better support of Springer document classes 18
Deal with document classes where the bibliography is called `\bibname` 49
Support of new ACM document class (`acmart.cls`) 52
`\axp@newtheorem`: Introduce intermediary command for theorem macro 16
`\axp@newtheoremrep@begin`: Better handling of note-free theorems in document classes that treat theorems differently when they have an empty note 38
`\axp@newtheoremrep@end`: Fix incorrect use of `\noexpand` in optional argument of macro environment 38
`\axp@writesection`: Fix extraneous space after section number in appendix titles . . . 48
v1.1.0
General: Added forward-link mechanism (K. D. Bauer) . . . 40
`\appendixsectionformat`: Fix missing space in default `\appendixsectionformat` . . . 17
`\axp@proofsfile`: Initialization deferred to `\AtBeginDocument` for compatibility with `\dump`ed

precompiled preambles (K. D. Bauer) . . . 35
\axp@redefinelabels: Fix \label not being disabled in amsmath environments, where \label@in@display is used instead (K. D. Bauer) . . . 18
\axp@repeqn: repeqn option . . . 14
\axp@unactivateeightbit: Fix compilation of non-ASCII characters with \usepackage[utf8]{inputenc} . . . 35
\axp@writesection: Make \axp@tmp wrapper more robust. Resolves issues from use of section title in fancyhdr, and in \tableofcontents (K. D. Bauer). . . . 48
\FVE@axp@VerbatimOut: Make apxproof compatible with independent use of fancyvrb . . 36
\section: Fix handling of fragile macros within section headings. See #22. . . . 47
Rewrote definition of \section to enable optional argument. See #23. (K. D. Bauer) . . . 47
\tableofcontents: Disable section management for table of contents . . . 48
v1.2.0
General: Do not load bibunits if bibliography is set to common 14
\axp@newtheoremrep: Fix formatting of theorems without notes in some document classes in inline mode . . . 34
\axp@newtheoremrep@definetheorem: Restore predefined theorem counters . . . 15
\axp@newtheoremrep@end: Fix display of theorem notes . . . 38
\axp@redefinelabels: Fix extra spacing erroneously introduced within the \axp@redefinelabels macro . 18
\mainbodyrepeatedtheorem: Configurable repeated theorem command . . . 17
proofsketch: Fix proof sketches in inline compilation mode for Springer document classes . . . 18
v1.2.1
General: Ad hoc fix for natbib package conflict . . . 49
\appendixrefname: Fix compatibility with memoir document class . . . 17
\FVE@axp@VerbatimOut: Fix compatibility with xypic package . . . 36
Fix compatibility with other uses of fancyvrb that set \FV@CatCodesHook . . . 36
v1.2.2
General: Compatibility with AMS document classes: do not use \AtEndDocument . . . 49
\axp@newtheoremrep@begin: Fix handling of optional arguments of repeated theorems containing optional arguments . . . 38
\axp@section: Detect a section within an included file to avoid produced useless sections . . . 46
v1.2.3
General: More robust redefinition of thebibliography environment, for compatibility with tocbibind 49
\axp@forwardlinking: forwardlinking option . . . 14
v1.2.4
appendixproof: Support optional arguments in proofs . . . 43
claimproof: Support for claimproof environment from lipics . . . 45
v1.2.5
claimproof: Support for locally defined claimproof environment . . . 45
axp@forward: Remove forward linking command from PDF bookmarks . . . 41
v1.2.6
General: Fix compilation issues in recent versions of acmart . . . 19
\axp@forward@setup: Support for aliascnt counters . . . 41
\axp@newtheoremrep@definetheorem: Undefine counter printing command to avoid warning . . 15
v1.3.0
General: Added chapterend appendix mode . . . 50

\axp@section: Detect a section within an \include'd file 46
Skip blank lines before detecting a section command 46
v1.4.0
General: Documented workarounds for grouping per-section appendices under a single wrapping section. See #56. . . . 9
Flush chapterend appendices at \part, \frontmatter, \mainmatter, and \backmatter boundaries so the last chapter of a part/matter is not emitted under the next one 50
Public \flushchapterappendix macro for manually emitting the current chapter's accumulated appendix in chapterend mode 50
appendixproof: Route deferred proofs through \axp@apxproofstart to support \appendixproofname 43
\appendixproofname: Added \appendixproofname hook for customizing the title of a proof deferred to the appendix 17
\axp@apxproofstart: Helper that consults \appendixproofname at appendix typeset time to auto-title deferred proofs 42
\axp@bibengine: bibengine option to select the engine used for separate bibliographies: bibunits (default) or biblatex. See #29. 14
\axp@newtheoremrep@end: Write auto-title state for \appendixproofname hook . . . 39
\axp@newtheoremreptmp: Added starred form of \newtheoremrep for proof-deferred theorems that are not restated in the appendix 16
\bibliography: Only rebind for bibunits engine. See #29. . . . 48
v1.4.1
\apxproofhook: Hook fired when a deferred proof is typeset, or for an inline proof whose title names its result through a \ref; used by the proofgraph package 42
v1.5.0
General: synctex option, auto by default 8
Document the LaTeXML binding 10
SyncTEX support: appendix material can be re-read from the source files instead of being copied to the .axp file 19
\axp@inlineproofhook: Recognize \zcref of zref-clever in the title of an inline proof 42
\axp@label@in@display: Support \zlabel of zref-clever 17
Support the optional argument that cleveref adds to \label (R. Rowe) 17
\axp@region@findmain: Locate the main source file through the .fls file, and fall back to copying when it cannot be read 29
\axp@synctex: synctex option, auto by default 14
axp@newtheoremrep: Repeated theorems typeset directly from the source with the synctex option 37

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

\! 350, 353
\" 350, 353
* 351, 354
\+ 351, 354
\, ... 352, 355
\- 351, 354
\/ 351, 354
\= 352, 355
\? 350, 353
\@ 698
\@@appendixproof . . . 1092, 1118, 1180

\@@axp@newtheorem 116, 183
\@@input 478, 560
\@@section 1250
\@acmplainnotefont 769, 770, 950, 968, 969
\@acmtitlebox 1414
\@appendixproof 1092, 1093, 1180
\@auxout 1277
\@axp@newtheorem 116, 183
\@begintheorem 949
\@biblabel 1433, 1434
\@bsphack 845
\@charlb 387, 391
\@charrb 387, 391
\@curr@file 1236
\@currenvir 225
\@currenvline 224
\@enddocumenthook 1301, 1410
\@esphack 862
\@extra@binfo 1304
\@gobble 852
\@ifnextchar 151, 155, 763, 1082, 1090, 1092, 1180, 1220, 1222, 1224, 1249
\@ifpackageloaded 7
\@ifstar 116, 128, 1081, 1251
\@namedef 1304
\@ne 287, 407, 438, 745, 841
\@nil . . 208, 224, 358, 359, 360, 361, 363, 364, 408, 571, 573, 589, 596, 600, 602, 607, 612, 616, 635, 643, 647, 649, 653, 1085, 1087, 1239, 1241
\@nnil 208, 224, 603, 604, 605, 650, 651, 652
\@noligs 857
\@oldbiblabel 1433, 1434
\@oparg 132, 136, 139
\@section 1250
\@strip@tex@ext 1236
\@thmcountersep 198
\\ 350
\{ 350
\} 350
\␣ 40, 46, 59, 73, 82, 96, 147
\^ 350, 1092, 1149, 1179, 1180
\| 350, 353

A

\ACM@origsection 200, 201
\active 1092, 1149, 1179, 1180
\AddToHook 485, 486, 490
\advance 287, 407, 438, 455, 682, 691, 841
\afterassignment 712
\aftergroup 729
\AND 63, 1294
\appendix 1308
appendix (option) *5*
\appendixbibliographyprelim *7*, 142, 1337, 1341, 1432
\appendixbibliographystyle *7*, 142, 1311
\appendixclaimproof . 1180, 1184, 1192
\appendixprelim *7*, 142, 1307
\appendixproof . . 806, 1092, 1153, 1161
appendixproof (env.) *2*, 802, 1092
\appendixproofname . . . *2*, *6*, 142, 1045
\appendixrefname *6*, 142, 1321, 1323, 1342, 1426
\appendixsectionformat . *6*, 142, 1281
\apx@oldamsthmendproof 10, 17
\apx@oldamsthmproof 9, 15, 16
\apxproofhook *8*, 1042, 1068
\arabic 1368
\AtBeginDocument 201, 813, 831, 1175, 1397
\autoref 1063
\axp@@@section 1252, 1253, 1254
\axp@@claimproof 1179, 1189
\axp@@proof 1149, 1158
\axp@@sectionfinishinclude 1219
\axp@@sectionskippar 1219
\axp@@sectiontestinclude 1219
\axp@@sectiontestinput 1219
\axp@@sectiontestsection 1219
\axp@appendix *24*, 31, 33, 35, 37, 40, 62, 163, 781, 815, 832, 994, 1292, 1351
\axp@appendixproof@write 1132, 1146, 1208
\axp@apxproofstart . . . 746, 1040, 1103
\axp@apxproofstart@title 1044, 1048, 1052
\axp@apxproofstart@withtitle 1051, 1056
\axp@beginchapterappendix 1353, 1380
\axp@bibengine 26, 43, 44, 46, 50, 1294, 1310, 1319
\axp@bibliography 25, 49, 56, 59, 63, 1293, 1309, 1318
\axp@chapterboundary 1385, 1392
\axp@chaptercounter 1361
\axp@chapterhasappendix 1361
\axp@claimproof 1179, 1181
\axp@currentaxpfile 1361, 1368, 1369, 1381
\axp@endchapterappendix . 1353, 1382
\axp@equation 880
\axp@equationx 880

\axp@extract@line 203
\axp@flushchapterappendix 1376, 1386, 1411
\axp@forward 1010
\axp@forward@link 1000, 1010
\axp@forward@setup 888, 1034
\axp@forward@suppress 76
\axp@forward@target 778, 942, 979, 1027
\axp@forwardlinking 28, 77, 78
\axp@gobbleline 349, 683
\axp@gobbletobegin 349, 687, 692
\axp@gobbletorefstop 1086, 1089
\axp@grabarg 1071, 1079
\axp@grabref 1075, 1076
\axp@grabzcarg 1066, 1081
\axp@grabzcarg@i 1081, 1082
\axp@grabzcarg@ii . . . 1082, 1083, 1084
\axp@grabzcarg@iii 1085, 1087
\axp@grabzcarg@opt 1082, 1083
\axp@hookboundary 1391, 1399, 1405, 1406, 1407, 1408
\axp@inlineproofhook 1057, 1164
\axp@inlineproofof@label 1068, 1080, 1088
\axp@inlineproofoffalse 1059
\axp@inlineproofoftrue . . 1080, 1088
\axp@label 151, 178
\axp@label@do 152, 153, 164, 174
\axp@label@in@display 151, 179
\axp@label@in@display@do 156, 157, 168, 175
\axp@label@in@display@noopt 155, 157
\axp@label@in@display@opt . . 155, 156
\axp@label@noopt 151, 153
\axp@label@opt 151, 152
\axp@line 275, 284, 288, 378, 413, 587, 588, 633, 634
\axp@markchapterhasappendix 863, 865, 922, 1095, 1120, 1364
\axp@newtheorem 107, 116, 886
\axp@newtheoremrep . 139, 782, 882, 882
\axp@newtheoremrep@begin 892, 899, 908
\axp@newtheoremrep@definetheorem 99, 783, 883
\axp@newtheoremrep@end 895, 904, 920
\axp@newtheoremrep@nostar . . 128, 134
\axp@newtheoremrep@star . . . 128, 130
\axp@newtheoremreptmp 127
\axp@nextproofof@label 938, 1041, 1046, 1054
\axp@nextproofof@type 937, 1040, 1045
\axp@noappendixbibliography 1330, 1335
\axp@oldbibliography 1293
\axp@oldclaimproof 826, 1177, 1186, 1194
\axp@oldlabel . . 160, 165, 166, 170, 174
\axp@oldlabel@in@display 161, 169, 175
\axp@oldproof 149, 1090, 1091, 1155, 1163, 1213, 1215
axp@oldproof (env.) 149
\axp@oldsection 267, 828, 1245, 1249, 1250, 1291
\axp@oldtableofcontents 1290
\axp@oldthebibliography 1325
\axp@oldzlabel 162, 172, 176
\axp@openchapterfile . 816, 1367, 1388
\axp@proof 1149, 1150
\axp@proofsfile 241, 813, 822, 854, 861, 924, 926, 936, 941, 945, 988, 1102, 1111, 1126, 1138, 1278, 1316, 1369, 1370, 1377, 1378
\axp@readfile 272, 283, 288, 304, 305, 309, 326, 585, 586, 587, 591, 626, 630, 633, 638
\axp@redefinelabels 158, 779, 787, 789, 795, 797, 980
\axp@refmark 1070, 1076
\axp@refstar 766, 932, 934, 964
\axp@refstop . . . 1075, 1076, 1079, 1089
\axp@region@afterbegin 680
\axp@region@altfile 472
\axp@region@altfile@i 571, 573
\axp@region@analyze 329, 369
\axp@region@begins 369, 374, 382, 383, 407, 686
\axp@region@catcoderestore . 441, 470
\axp@region@catcodes 427
\axp@region@check 256, 272
\axp@region@checkend 332, 412
\axp@region@cleanup 427
\axp@region@endcmd 722, 731
\axp@region@endregion 733, 738
\axp@region@env 203, 243, 958
\axp@region@file 203, 242, 957
\axp@region@findmain 618, 660
\axp@region@findmain@i 634, 643
\axp@region@findmain@ii . . . 647, 649
\axp@region@findmain@iii . . . 650, 651
\axp@region@findpwd 566, 580
\axp@region@findpwd@i 588, 596
\axp@region@findpwd@ii 600, 602
\axp@region@findpwd@iii . . . 603, 604
\axp@region@finish 729, 738, 739
\axp@region@flag 1100, 1109, 1124, 1135
\axp@region@flushsection . . . 260, 263
\axp@region@forcedtag 476, 482, 484, 487, 491, 492
\axp@region@foundfalse 375
\axp@region@foundtrue 405

\axp@region@getname 349, 403
\axp@region@getname@i 360, 361
\axp@region@go@direct 715, 719
\axp@region@go@wrapped 717, 724
\axp@region@input 456, 472
\axp@region@input@done 460, 472
\axp@region@jobnametex 624, 654
\axp@region@lastcomponent 607, 612, 616, 653
\axp@region@line 203, 242, 957
\axp@region@mainfile 203, 655, 661, 667
\axp@region@marker 358, 363, 372, 401, 414
\axp@region@name 361, 404, 467, 699, 710, 714, 727, 728, 730
\axp@region@next 702, 704, 712
\axp@region@nextbegin 367, 680
\axp@region@okfalse 292, 339, 415
\axp@region@oktrue 380, 384
\axp@region@openfile . . . 274, 295, 298, 301, 303, 307, 320, 593, 640
\axp@region@pendingsection . . 263, 264, 266, 268, 269, 752, 758, 759
\axp@region@proof 740, 1135
\axp@region@proof@noopt . . . 742, 744
\axp@region@proof@opt 742, 743
\axp@region@pwd 563, 565, 568, 578, 581, 606, 609
\axp@region@pwdbase 578, 607, 608
\axp@region@readcount 273, 278, 285, 287, 296, 306, 310
\axp@region@readname 700, 701
\axp@region@readname@done . . 705, 711
\axp@region@readname@go 701
\axp@region@readname@i 702, 703
\axp@region@readname@ii . . . 707, 709
\axp@region@readone 272
\axp@region@readto 272
\axp@region@record 203, 869, 891, 1099, 1108, 1123
\axp@region@replay 258, 427
\axp@region@rest . . 365, 400, 401, 414
\axp@region@rp 740, 959
\axp@region@rp@i 763, 764
\axp@region@savedbegin 442, 461
\axp@region@savedend 444, 462
\axp@region@savedendcmd 448, 463, 735
\axp@region@savedwrappedend 468, 726, 729
\axp@region@saveendlinechar 293, 336
\axp@region@scan 349, 408
\axp@region@scan@i 359, 369
\axp@region@scanend 349
\axp@region@scanendline . . . 349, 413
\axp@region@scanline 349, 378
\axp@region@section 263, 867
\axp@region@setcatcodes 433, 451, 685, 739
\axp@region@setup 659
\axp@region@stopmainfalse 627
\axp@region@stopmaintrue . . . 631, 656
\axp@region@tag 203, 243, 960
\axp@region@target 376, 404
\axp@region@toappendix 751, 876
\axp@region@toappendix@section 755, 757
\axp@region@toofar 322, 327, 338
\axp@region@wrappedfalse 449
\axp@region@wrappedtrue 725
\axp@region@write 240, 876, 1134
\axp@repeqn 27, 68, 70, 73, 82, 909, 925, 987
\axp@replay@altfile 560, 569, 575, 577
\axp@replay@begin 253, 446, 723
\axp@replay@end 254, 447, 738
\axp@replay@endline 251, 331, 420
\axp@replay@env 252, 313, 343, 377, 387, 391, 417, 443, 445, 446, 447, 461, 462, 464, 714, 720
\axp@replay@file 249, 295, 305, 307, 312, 341, 389, 419, 478, 569, 571, 575, 578
\axp@replay@line 250, 296, 314, 325, 341, 388, 419, 454
\axp@replay@tag 255, 474, 475, 476, 480
\axp@replaybody 242, 246, 957
\axp@replayproof 824, 1090
\axp@replayproof@opt 1090, 1091
\axp@rpcounter 879
\axp@seccounter 1217
\axp@section 760, 1219, 1267
\axp@sectioncmd 867, 1267, 1280
\axp@sectitle 1218, 1255, 1265, 1269, 1284, 1373
\axp@seenreptheorem 878
\axp@skipcount 370, 454, 455, 681, 682, 686, 690, 691
\axp@skiplines 366, 478, 560, 680
\axp@synctex 29, 85, 86, 87, 89, 96
\axp@tmp 597, 599, 623, 625, 644, 646, 758, 760, 1230, 1232, 1236, 1237, 1239, 1271, 1277, 1279, 1328, 1329
\axp@tmpa . 598, 599, 645, 646, 653, 654
\axp@tmptmp 1231, 1232, 1238, 1239
\axp@tryrefcmd 1060, 1061, 1062, 1063, 1064, 1071
\axp@tryrefcmd@ 1066, 1071, 1072
\axp@unactivateeightbit . . . 836, 847

\axp@VerbatimOut 873, 1116, 1130
axp@VerbatimOut (env.) 843
\axp@with@forward 914, 916, 993
\axp@writeproofsfilepreamble 821, 833, 1371
\axp@writesection 868, 871, 923, 1096, 1121, 1267
\axp@zlabel 172, 176, 180

B

\b 351
\backmatter 1408
\bfseries 186, 188, 191, 195
\bibcite 1303, 1346
bibengine (option) 5
\bibliography 1293
bibliography (option) 5
\bibname 1322, 1323
\BODY 795, 797, 900, 903
\boolfalse 665, 1008, 1372
\booltrue 80, 88, 92, 995, 1365

C

\CatchFileDef 1230, 1237
\catcode . . . 353, 354, 355, 428, 436, 452, 698, 839, 1092, 1149, 1179, 1180
\chapter 1398, 1399, 1402
\chapterappendixprelim 7, 1353
\claimproof 826, 1176, 1177, 1179
claimproof (env.) 1175
\clearpage 146, 1353
\closein 304, 591, 638
\closeout 1316, 1378
\count@ 434, 436, 437, 438, 837, 839, 840, 841
\Cref 1062
\cref 1061
\csdef 131, 999
\csedef 1035
\csletcs 997, 998, 1006
\csundef 135
\CurrentFilePathUsed 214, 217
\CurrentFileUsed 210, 211, 215, 217, 233, 235

D

\d 352
\DeclareStringOption 24, 25, 26, 27, 28, 29
\defaultbibliography 1297
\DefineVerbatimEnvironment 843
\detokenize 377, 598, 625, 645
\directlua 226, 227, 472, 475, 481, 488, 492, 495

E

\e 351
\egroup 704
\empty 1329
\endappendixclaimproof . . 1200, 1205
\endappendixproof 807, 1143, 1170
\endaxp@oldclaimproof 827, 1178, 1202
\endaxp@oldproof 150, 825, 1172, 1214, 1216
\endaxp@oldthebibliography 1326
\endaxp@VerbatimOut . . 874, 1145, 1207
\endclaimproof 827, 1178, 1197
\endinlineproof 803, 1214
\endinput 739
\endlinechar 293, 294, 336, 453, 584, 622
\endnestedproof 805, 1216
\endproof 10, 12, 17, 150, 803, 805, 807, 825, 1167
\endthebibliography 1326
environments:
appendixproof 2, 802, 1092
axp@oldproof 149
axp@VerbatimOut 843
claimproof 1175
inlineproof 4, 802, 1213
nestedproof 8, 802, 1213
proof 3, 821, 1149
proofsketch 4, 141
toappendix 1, 808, 864
\errmessage 40, 46, 59, 73, 82, 96
\expandonce 758, 1273, 1277, 1279, 1283, 1284

F

\flushchapterappendix 7, 1385
forwardlinking (option) 7
\frontmatter 1406
\futurelet 702
\FV@CatCodesHook 858
\FV@DefineTabOut 850
\FV@DefineWhiteSpace 848
\FV@FontScanPrep 856
\FV@ProcessLine 852, 854
\FV@Scan 859
\FV@Space 849
\FVB@axp@VerbatimOut 843
\FVE@axp@VerbatimOut 843

G

\g 351
\gdef . . 264, 298, 301, 320, 581, 593, 609, 640, 937, 938, 1040, 1041, 1330

H

\hspace 1434

\hyperlink 1016
\hypertarget 1029

I

\i 351
\ifaxp@inlineproofof 1057, 1061, 1062, 1063, 1064, 1065, 1068
\ifaxp@region@found 371, 379
\ifaxp@region@ok 246, 257, 330
\ifaxp@region@stopmain 618, 629
\ifaxp@region@wrapped 247, 466
\ifblank 1035
\ifbool 232, 659, 784, 851, 866, 875, 889, 931, 956, 1098, 1107, 1122, 1133, 1379
\ifboolexpr 1014
\ifcsdef . . . 885, 940, 1005, 1015, 1035
\ifcsname 102, 108, 732, 1028
\ifcsundef 996
\ifdef . . . 1398, 1405, 1406, 1407, 1408
\ifeof 283, 309, 326, 586, 630
\IfFileExists 582, 620, 661
\ifin@ 1074
\ifnum . . 91, 278, 296, 310, 382, 429, 437, 474, 480, 681, 690, 745, 840
\ignorespaces 743, 750, 753
\in@ 1073
\include 1224, 1234, 1247
\inlineproof 802, 1213
inlineproof (env.) *4*, 802, 1213
\input 1222, 1229, 1317, 1381
\inputlineno 243, 958
\itshape 186, 188, 191, 195, 1419

L

\label 160, 178, 914, 916, 1262
\label@in@display 159, 161, 179
\lccode 350, 351, 352
\lipics@opterrshort 1431
\long . . 357, 359, 360, 361, 362, 364, 366, 367, 399, 1058, 1071, 1072, 1075, 1079, 1084, 1087, 1089, 1241
\loop 277, 435, 628, 838
\lowercase 356
\ltx@ifnextchar@nospace 741, 1149, 1179

M

\m@ne 294, 455, 584, 622, 682, 691
\mainbodyrepeatedtheorem . *6*, 142, 918
\mainmatter 1407
\message 32, 34, 36, 38, 52, 54, 57, 69, 71, 79, 662
\MessageBreak 236, 312, 316, 318, 342, 345, 388, 392, 394, 418, 422, 668, 670, 672, 674, 676

N

\n 352
\NAT@testdef 1302, 1345, 1346
\nestedproof 804, 1215
nestedproof (env.) *8*, 802, 1213
\newbool 76, 85, 1010, 1362
\newcount 273, 369, 370
\newif 246, 247, 371, 618, 1057
\newread 272
\newtheorem 119, 121, 125
\newtheoremrep *2*, *3*, 127
\newtheoremrep* *2*, *3*
\newtheoremstyle 1416
\noproofinappendix 8, 808, 884, 1037, 1147, 1209
\nosectionappendix . *8*, 808, 1264, 1287
\number 431
\numexpr 431

O

\onecolumn 146
\openin 305, 585, 626
\openout 818, 1369
options:
appendix *5*
bibengine *5*
bibliography *5*
forwardlinking *7*
repeqn *7*
synctex *8*

P

\PackageError . . 64, 311, 340, 386, 416
\PackageWarning . 234, 666, 1393, 1401
\par 1220, 1228
\part 1405
\pretocmd . . 884, 949, 1301, 1392, 1410
\printbibliography 1342
\ProcessLocalKeyvalOptions 30
\proof 9, 11, 16, 149, 802, 804, 806, 824, 1149
proof (env.) *3*, 821, 1149
\proofsketch 197
proofsketch (env.) *4*, 141
\protected@xdef 1044
\providecommand 143, 159, 1042
\putbib 1338

R

\readline 288, 587, 633
\ref 932, 934, 1046, 1060, 1274
\refname 1320, 1321, 1426
\repeat 280, 439, 637, 842
repeqn (option) *7*

S

\scshape 1421
\section 201, 821, 1242, 1249, 1250, 1291
\set@curr@file 1235
\setcounter 910, 927, 928, 989
\SetupKeyvalOptions 20
\small 1434
\spnewtheorem . 182, 186, 188, 191, 195
\stepcounter 1387
\string 235, 387, 391, 417, 421, 1394, 1402
\synctex 90, 91
synctex (option) *8*

T

\tableofcontents 1290
\texorpdfstring 1012, 1013
\the . . 243, 293, 428, 772, 951, 958, 972
\theaxp@equation 928
\thebibliography 1325, 1426
\theopargself 768, 946, 947, 966
\theoremstyle 1425
\thm@notefont 772, 951, 971, 972
\thmhead 950, 951, 952, 953
toappendix (env.) *1*, 808, 864
\togglefalse 1038, 1374
\toggletrue 921

U

\upshape 195

V

\value 910, 927, 989
\vref 1064

W

\write . 241, 822, 854, 861, 924, 926, 936, 941, 945, 988, 1102, 1111, 1126, 1138, 1277, 1278, 1370, 1377

X

\xdef 606, 607, 655, 1080, 1088

Z

\z@ 91, 306, 310, 374, 382, 383, 434, 474, 480, 681, 690
\zcref 1066
\zlabel 162, 180

References

[1] Leslie Lamport. *LaTeX: A Document Preparation System.* Addison–Wesley Pub. Co., Reading, MA, 1986.

A Proofs for Section 1 (Usage)

Proof. □

Foobar 2. *This foobar is repeated in the appendix.*

Proof. □

Proof. □