The **phfthm** package[1]

Philippe Faist   *philippe.faist@bluewin.ch*

August 15, 2016

phfthm—Goodies for theorems and proofs.

▪ 1   Introduction

The phfthm package provides enhanced theorem and proof environments, based on the amsthm original versions. It allows for hooks to be placed, adds some default goodies and is highly customizable.

There are three generic types of environments provided: theorem environments, proof environments and "thmheading" environments.

1.1   Theorem environments

Theorem environments look like this:

Theorem 1 (Gauss)**.** *For a closed surface S enclosing a volume V , we have*

$$\oint_S \vec{u} \cdot d\vec{S} = \int_V (\vec{\nabla} \cdot \vec{u})\, dV \ . \tag{1}$$

1.2   Proof environments

A proof environment might look like this:

> *Proof of Theorem 1.* The proof of the theorem goes here.
> ...   ∎

The enhanced theorem and proof environments provided by this package allow to pair theorems with proofs, automatically generating references from one to the other (see section 5).

1.3 Theorem-heading environments

Finally, theorem-heading environments are formatted like theorems, but the heading title is set as an argument to the environment. These environments are a nice alternative for definitions, and look like this:

> **Trace Distance.** *The 'trace distance' between ρ and σ is defined as*
> $$\delta(\rho, \sigma) = \frac{1}{2} \left\| \rho - \sigma \right\|_1 , \tag{2}$$
> *where $\left\| \cdot \right\|_1$ is the Schatten-1 norm.*

∎ 2 Quick start and package options

Example: Load the rich theorem set, with separate counters, and with proof-ref mechanism on and always displaying the proof reference in the margin:

```
\usepackage[thmset=rich,sepcounters=true,proofref={always,margin}]{phfthm}
```

By default, some styles are tweaked a bit so that they appear nicely as documented below (for example, by using a filled square instead of a simple square for end-of-proof QED markers). Use the package option `resetstyle` to instruct phfthm not to proceed to these style adjustments; this allows you to enable features individually and selectively:

```
\usepackage[resetstyle,prooftitleitbf=true]{../phfthm}
```

2.1 Predefined theorem environments

You may load predefined theorem sets via the package option `thmset`. Theorem sets group common environments used in mathematical works such as Theorem, Proposition, Definition, etc.

Some package options control the way these environments are defined. If you would like more refined control over the appearance of these environments, or over which environments are defined, you may consider calling \phfLoadThmSet manually or defining individual environments with \phfMakeTheorem.

Possible theorem sets are:

`thmset=`

`thmset=empty`

Do not define any environment at package loading. You may of course invoke \phfLoadThmSet or \phfMakeTheorem manually at any later point.

`thmset=simple`

theorem
proposition
lemma
corollary
definition
Define the environments theorem, proposition, lemma and corollary as theorem-like environments, and definition as a definition-like environment.

`thmset=default`

conjecture
remark
Define the environments theorem, proposition, lemma, corollary, conjecture, remark as theorem-like environments, and definition as a definition-like environment.

`thmset=shortnames`

For if you like typing less: the same environments are defined as the default set, but with shorter names. Define the environments thm, prop, lem, cor, conj, rem as theorem-like environments, and defn as a definition-like environment.

`thmset=rich`

idea
question
problem
Provides the same environments as the default theorem set, as well as in addition: idea, question and problem as theorem-like environments.

You may also load a theorem set at a later point after loading the phfthm package by invoking the \phfLoadThmSet macro, see subsection 3.2.

Further package options modify the style of the theorem-like and definition-like environments defined via the `thmset` package option:

`theoremstyle=⟨`*theorem style name*`⟩`

Use this package option to specify which style to use for theorem-like environments when loading the theorem set specified via the `thmset` package option. The theorem style name should be one of plain, definition, remark, or any other \newtheoremstyle-defined theorem style (see documentation of amsthm).

`definitionstyle=`⟨*theorem style name*⟩

Use this package option to specify which style to use for definition-like environments when loading the theorem set specified via the `thmset` package option. The theorem style name should be one of `plain`, `definition`, `remark`, or any other `\newtheoremstyle`-defined theorem style (see documentation of amsthm).

Further options control various aspects of the environments defined by `thmset`.

`sepcounters=`⟨`true`│`false`⟩

Each theorem environment defined with the `thmset` package option will use a separate counter if this option is set; otherwise (the default), there is a single counter which is shared by all those theorem environments.

The `proofref` package option allows to specify a comma-separated list of attributes to apply to the proof reference ("proof on page XYZ") displayed along with the theorem. The following attributes may be specified:

`proofref={}`

`proofref={default}`

Do not change the default proof reference appearance.

`proofref=false`

Deactivate the proof-ref mechanism.

`proofref={margin,...}`

The proof reference is displayed in the margin, instead of after the theorem.

`proofref={longref,...}`

The proof reference is displayed as a full sentence ("The proof of this ⟨*Theorem Name*⟩ can be found on page ⟨*XYZ*⟩.").

`proofref={off,...}`

Turn off the proof reference mechanism completely for theorems defined with the `thmset` option.

`proofref={always,...}`

Always display the proof reference, even if the proof is on the same page or on a nearby page.

Note: this option has a global effect.

`proofref={onyifveryfar,...}`

> The proof reference is only displayed if the proof is at least two pages back, or four pages ahead.
>
> Note: this option has a global effect.

> **NOTE**
>
> The two package options `proofref={always}` and `proofref={onlyifveryfar}` apply to *all* theorem environments which use the proof-ref mechanism, whether they have already been defined or not (see \phfProofrefPageBackTolerance and \phfProofrefPageAheadTolerance).
> All the other above options apply only to the theorem environments defined via the `thmset` package option.

2.2 The proof environment

proof   By default, the phfthm package overrides the proof environment with a the package's own enhanced version. If you want to preserve the original *AMS* environment, you should use the `proofenv=false` package option.

`proofenv=⟨true | false⟩`

> If set to `true`, then define an enhanced proof environment when loading this package. This will override any previously existing proof environment such as *AMS*'.
>
> If set to `false`, no action is taken at package loading time. You should then directly use the \phfMakeProofEnv macro to define proof environments.

If you want finer control over how the proof environment is defined, or if you want to customize its appearance, you should use the \phfMakeProofEnv macro directly (subsection 4.1).

If you set `proofenv=true`, there are a couple package options which alter the way the proof displays:

`smallproofs=⟨true | false⟩`

> If set to `true`, then proofs display in a smaller font.

`qedsymbolblacksquare=⟨true | false⟩`

> If set to `true`, the QED end-of-proof symbol (usually "□" with amsthm) is replaced by a filled square ("■").

> If set to `true`, then the proof title ("Proof" or "Proof of Theorem 1") is typeset in bold italic font.

2.3 The theorem-heading environment

`thmheading` By default, the `thmheading` environment is provided by the phfthm package:

```
\begin{thmheading}{Trace Distance}
  The 'trace distance' between $\rho$ and $\sigma$ is defined as
  \begin{equation}
    \delta(\rho,\sigma)=\frac12\,\left\Vert\rho-\sigma\right\Vert_1\ ,
  \end{equation}
  where $\lVert\cdot\rVert_1$ is the Schatten-1 norm.
\end{thmheading}
```

> **Trace Distance.** *The 'trace distance' between ρ and σ is defined as*
>
> $$\delta(\rho,\sigma) = \frac{1}{2}\,\left\Vert \rho - \sigma \right\Vert_1 \, , \tag{3}$$
>
> *where $\|\cdot\|_1$ is the Schatten-1 norm.*

You may also use `\label` and `\ref` as usual (`\ref` simply displays the given title).

Some package options control the way this environment is defined.

```
thmheading=⟨true|false⟩
```

> Define the environment `\begin{thmheading}...\end{thmheading}` when loading the phfthm package, with reasonable default settings.

```
thmheadingstyle=⟨theorem style⟩
```

> If `thmheading=true` was specified, you may use this option to specify the theorem style to use for the `thmheading` environment. Possible values are *AMS* theorem style names (e.g. the base styles `plain`, `definition` or `remark`), or any other style defined with `\newtheoremstyle`.

If you want to define theorem-heading environments manually, see .

■ 3   Theorem environments

A theorem environment is based on the environment furnished by amsthm's `\newtheorem` command, but with added goodies.

3.1 Define theorem environments manually

\phfMakeTheorem   If you don't want to load a full theorem set (subsection 2.1), you can define theorem environments individually with `\phfMakeTheorem`:

\phfMakeTheorem[⟨*key-value options*⟩]{⟨*theorem environment name*⟩} {⟨*theorem name*⟩}

This command defines a new environment (given as the first mandatory argument) which behaves as a theorem and is displayed as given by the second mandatory argument. For example, we might call `\phfMakeTheorem{theorem}{Theorem}` to define the environment `\begin{theorem}...\end{theorem}` which displays "*Theorem N. . . .*"

The possible key-value options for the optional argument are:

counter=⟨*LATEX counter* | *(empty)*⟩

> The name of the LATEX counter to use for the theorem environment. If this is empty, then a new counter will be created which is specific to this theorem environment (the default). If not empty, then the theorem environment uses the given counter (or an alias thereof, see aliascounter).
>
> If a counter is specified, the counter should already be defined with LATEX's `\newcounter`.

aliascounter=⟨`true` | `false`⟩

> In some cases (e.g. if you're using hyperref's `\autoref`), it is important to have counters specific to each theorem environment (so you get "Theorem 5" or "Proposition 5" right). However, you may want different theorem environments to share a same logical counter (Say "Definition 1", "Definition 2", "Theorem 3", "Proposition 4"). In this case, you should specify aliascounter=true.
>
> When this option is on, then first we define an alias counter of the one given to the counter option, and then use the alias for the theorem environment. The alias is declared using the aliascnt package. The alias counter is automatically set up correctly for using `\autoref`.
>
> Note that the aliascounter option only has an effect if the counter option is set to some non-empty value. If counter is set to a non-empty value, then aliascounter defaults to true.

thmstyle=⟨*theorem style* | *(empty)*⟩

> The theorem style to use to define this theorem environment. The value of this option should be a valid argument to *AMS*'s `\theoremstyle`. If you leave this empty (the default), then the theorem style is not set explicitly and whatever default style is used.

`defnostar=⟨true|false⟩`

> Set this to `true` if you want the corresponding non-starred theorem environment to be defined, e.g. `\begin{theorem}...\end{theorem}`.
>
> Normal (non-starred) versions of the environments have an associated theorem number, as you expect by default.

`defstar=⟨true|false⟩`

> Set this to `true` if you want the corresponding starred theorem environment to be defined, e.g. `\begin{theorem*}...\end{theorem*}`.
>
> Starred versions of the environments do not have an associated theorem number.

`proofref=⟨true|false⟩`

> Enable or disable the proof-ref mechanism for this theorem environment (enabled by default).

`proofrefstyle=⟨proof-ref style⟩`

> The style to use for the proof references. Here you may specify how the proof ref appears, for example (in the margin, long sentence, ...). Possible styles are `proofrefstyle=default` (the default), `proofrefstyle=margin` (display the proof ref in the margin of the page) and `proofrefstyle=longref` (as by default but with a full sentence). See [subsection 5.3](#) for how to further customize the appearance of the proof reference.

For example, you may use the following command invocation to define a theorem environment named "Remark" implemented as `\begin{remark}...\end{remark}`, also with a starred verison `\begin{remark*}...\end{remark*}`, using the `plain` *AMS* theorem style, and without the proof-ref mechanism:

```
\phfMakeTheorem[defstar=true,defnostar=true,thmstyle=plain,counter=,%
    proofref=false]{remark}{Remark}
```

3.2 Loading theorem sets manually

`\phfLoadThmSet` You may load theorem sets at any time via the macro `\phfLoadThmSet`. This may be useful, for example, to load theorem sets only after you have defined a custom theorem style. The syntax of `\phfLoadThmSet` is:

`\phfLoadThmSet{⟨options to \phfLoadThmSet for theorem-like environments⟩}{⟨options to \phfLoadThmSet for definition-like environments⟩}{⟨name of theorem set to load⟩}`

The first and second argument to this macro are tokens to expand in front of \phfMakeTheorem for theorem-like or definition-like environments. For example:

```
\newcounter{mythmcounter}
\newtheoremstyle{mythmstyle}{...}
\newtheoremstyle{mydefnstyle}{...}
\phfLoadThmSet{[thmstyle=mythmstyle,counter=mythmcounter]}
    {[thmstyle=mydefnstyle,counter=mythmcounter]}{rich}
```

> **WARNING**
>
> The first and second arguments to \phfLoadThmSet must either be empty, or be enclosed in square braces.

\theoremname
\propositionname
\lemmaname
\corollaryname
\conjecturename
\remarkname
\definitionname
\ideaname
\questionname
\problemname

The title of the theorem environments defined in theorem sets use the same scheme as figures, tables, etc. with regard to translations and babel: they use \theoremname, \propositionname, etc.

This package is language agnostic (with titles defined by default in English), and does not provide the titles for other languages. In order to support language switching with babel and \selectlanguage, you should add the relevant names to the corresponding \captions⟨*language name*⟩ macro, for example:

```
\usepackage[francais,...]{babel}
...
\addto\captionsfrancais{%
  \def\theoremname{Th\'eor\`eme}%
  \def\propositionname{Proposition}%
  \def\lemmaname{Lemme}%
  \def\corollaryname{Corollaire}%
  \def\conjecturename{Conjecture}%
  \def\remarkname{Remarque}%
  \def\definitionname{D\'efinition}%
  \def\ideaname{Id\'ee}%
  \def\questionname{Question}%
  \def\problemname{Probl\`eme}%
}
... \selectlanguage{francais} ...
```

3.3 Theorem hooks

Any theorem environment automatically calls some hooks. There are hooks available per theorem environment as well as generic for all theorem environ-

ments.

\phfthm@hook@start@thmname The hook \phfthm@hook@start@⟨*theorem environment name*⟩{⟨*theorem title*⟩} is called at the start of the environment. More precisely, it is called inside the original amsthm base environment; that is, after the heading was generated. It takes one mandatory argument, the optional title provided to the theorem environment which may be empty. By default, the hook defers to the global hook \phfthm@hook@startcommonnostar.

\phfthm@hook@start@thmname* The hook \phfthm@hook@start@⟨*starred theorem environment name*⟩ is completely analogous, and is called for the starred environment. The only difference is that by default, it defers its call to \phfthm@hook@startcommonstar.

\phfthm@hook@startcommonnostar The hook \phfthm@hook@startcommonnostar{⟨*theorem environment name*⟩}{⟨*theorem optional given title*⟩} collects the default definitions for non-starred environments (none by default) and continues **\phfthm@hook@startcommon** to defer to \phfthm@hook@startcommon{⟨*theorem environment* **\phfthm@hook@startcommonstar** *name*⟩}{⟨*theorem optional given title*⟩}. Analogously, the macro \phfthm@hook@startcommonstar{⟨*theorem environment name*⟩}{⟨*theorem optional given title*⟩} groups commands for starred environments (typically doesn't take care of \label stuff) and also defers to \phfthm@hook@startcommon.

\phfthm@hook@end@thmname The end hooks work very much analogously. \phfthm@hook@end@⟨*theorem* **\phfthm@hook@end@thmname*** *environment name*⟩ and \phfthm@hook@end@⟨*starred theorem environment name*⟩ are called respectively for the non-starred and starred version of that theorem environment, and by default they defer to the common **\phfthm@hook@endcommonnostar** \phfthm@hook@endcommonnostar{⟨*theorem environment name*⟩} or **\phfthm@hook@endcommonstar** \phfthm@hook@endcommonstar{⟨*theorem environment name*⟩}. Both these hooks defer their calls to \phfthm@hook@endcommon{⟨*theorem environment name*⟩}.

For theorems using the proof-reference mechanism, i.e. for which proofref=true was specified to \phfMakeTheorem and which uses **\phfthm@hook@afterlabel@thmname** the \label hack (section 5), there is an additional hook. The hook \phfthm@hook@afterlabel@⟨*theorem environment name*⟩ is called just after the \label command corresponding to the theorem is encountered (this should always be at the *beginning* of the theorem, see section 5). Depending on the proof-ref style, this hook may be used to generate the proof reference text (for example, with the margin proof-ref style). The hook is called after the theorem label is set. The label itself can be recovered from the value of the macro \phfthm@val@thmlabel. By default, that hook calls the common hook **\phfthm@hook@afterlabelcommon** \phfthm@hook@afterlabelcommon{⟨*theorem environment name*⟩}. (After the first occurrence of the command \label, the latter's definition is restored.)

■ 4   Proof environments

Proof environments typeset mathematical proofs. The proof environment(s) provided by phfthm give some added functionality with respect to the *AMS*-default proof environment, such as supporting the proof-reference mechanism described in .

A proof environment might look like the following:

Proof of Theorem 5.   Let $\mathcal{T}_{X \to X'}$ be any trace nonincreasing completely positive map such that $\mathcal{T}_{X \to X'}(\Gamma_X)$ lies within the support of $\Gamma_{X'}$. Define the normalized state $\gamma_X = \Gamma_X / \operatorname{tr} \Gamma_X$.

Now consider this and that . . .   ■

The proof environments defined by this package wrap a given proof display environment (such as *AMS*' (amsthm's) or IEEEtran's original proof environment) by adding functionality in the form of hooks. In the following, we refer to the "underlying proof display environment" as the original environment which is wrapped. It may be any LaTeX environment whose task is to format the proof nicely.

4.1   Manually define a proof environment

\phfMakeProofEnv   You may use the macro \phfMakeProofEnv to declare a new proof environment. The syntax is:

\phfMakeProofEnv[⟨*key-value options*⟩]{⟨*proof environment name*⟩}

This defines a new environment with the given name, which may be used to display proofs to theorems. The options may be:

displayenv=⟨*name of LaTeX environment*⟩

> Set a LaTeX environment to use to actually format and display the proof. (The \phfMakeProofEnv command itself doesn't care about how the proof is displayed or formatted; rather it adds a goodies infrastructure in which stuff can be plugged in and provides options for such goodies.)

> You may specify here the name of a LaTeX environment, or give the special value displayenv=* to indicate the default appearance provided by phfthm, or leave the value empty displayenv= to signify that no underlying display environment should be invoked. (The latter may be useful if you are plugging a \phfMakeProofEnv-generated environment into a larger environment which already takes care of the display.)

defaultproofname=⟨*default proof title*⟩

> Specify here the title to use (e.g. "Proof") if no argument was given to the proof environment. If you do not specify any defaultproofname, or pass an empty value, then the value of \proofname is used.

`parselabel=⟨true | false⟩`

> Specify whether the environment should parse its argument for some special information. If set to `true`, then the proof argument is passed on to a command (specified by the `parselabelcmd` option).

`parselabelcmd=⟨LATEX macro⟩`

> If `parselabel` is set to `true`, then specify here a LATEX command which parses whatever it wants from the proof environment's argument. The macro should set the `\phfthm@val@displayargs` macro to tokens which will be expanded just after the invocation of the proof environment's display environment (`displayenv`). It should also set `\phfthm@val@proofoflabel` (if appropriate) to the label corresponding to the theorem for which this is the proof of.

> By default, the command `\phfthm@proof@parselabel` is used, which parses the proof environment's argument for a reference to a theorem in the context of a proof-ref mechanism (see [section 5](#)). The label is parsed to see if it is of the form `[*thm:reference]`, where `thm:reference` is the label pinned to a theorem.

`override=⟨true | false⟩`

> Whether to override any existing environment with the same name as the new proof environment. If `true` is specified here, then `\renewenvironment` is used to define the proof environment, otherwise a simple `\newenvironment` is used.

`internalcounter=⟨name of LATEX counter⟩`

> The name of the internal counter the proof environment should use. The count number is not displayed (by default at least), but it is only used to pin down anchors for PDF hyperlinks.

> The counter should already be defined with `\newcounter`.

`proofofname=⟨LATEX macro⟩`

> Specify here a macro which will be called with a single argument. The macro produces the text to display when the proof environment is parsed as the proof of a specific theorem or proposition (or other theorem environment). The argument which will be given to it is the title of what the proof is of (e.g. "Theorem 3"). Typically, the macro should produce something like "Proof of Theorem XYZ."

> By default, the global macro `\proofofname` is used.

`\proofname`   Text to use to display "Proof." This should be already defined by the LATEX system, and babel should already provide translations in different languages.

`\proofofname`   The globally defined macro `\proofofname` specifies the default way of displaying "Proof of Theorem 5." It is originally defined as something like

```
\newcommand\proofofname[1]{\proofname{} of #1}
```

You may override this to obtain something fancier, of you wish to display the document in a different language:

```
\def\proofofnamefrancais#1{\proofname{} (#1)}
\addto\captionsfrancais{\let\proofofname\proofofnamefrancais}
...
\selectlanguage{francais} ...
```

4.2 Proof hooks

The proof hooks are relatively straightforward. All hooks presented here take no argument.

Information about the argument of the proof, both the raw argument and the possibly parsed reference, are available as macros to some of the hooks (but don't change these values unless you know what you're doing). The macro `\phfthm@val@proofarg` contains the raw argument to the proof environment, and is available to all hooks. If you use the default proof environment argument parsing (which you must have enabled when calling `\phfMakeProofEnv`), then additionally the macros `\phfthm@val@prooftitle` and `\phfthm@val@proofofname` are available containing, respectively, the label of the theorem which is referenced, and the displayable reference to it (e.g. "Theorem 5"). The last two macros are available to all hooks except the first one (`..@start`).

The hooks named `..@start...` are called within the call to `\begin{<proof environment>}`.

`\phfthm@hookproof@...@start`   The hook named `\phfthm@hookproof@`⟨*environment name*⟩`@start` is called at the very beginning of the proof environment.

`\phfthm@hookproof@...@startafterdisplay`

The hook named `\phfthm@hookproof@`⟨*environment name*⟩`@startafterdisplay` is invoked immediately after the beginning of the underlying "display" environment (the environment used to display the proof contents).

`\phfthm@hookproof@...@startlast`

The hook named `\phfthm@hookproof@`⟨*environment name*⟩`@startlast` is called after we are sure that an anchor has been pinned down for the proof. This hook is called last within the commands in `\begin{<proof environment>}`.

`\phfPinProofAnchor`   By the way, the macro `\phfPinProofAnchor` may be used within the hooks

to pin down an anchor for referring to the proof (especially via the proof-ref mechanism). Just call it anywhere appropriate (a good idea is calling it after leaving v-mode before displaying the title, in order to avoid placing it just before a page break). If you do not call this macro, it is automatically called for you just before the `...@startlast` hook.

The two following hooks are called within the call to `\end{<proof environment>}`.

`\phfthm@hookproof@...@end` The hook `\phfthm@hookproof@`⟨*environment name*⟩`@end` is called before the proof display environment is closed.

`\phfthm@hookproof@...@final` The hook `\phfthm@hookproof@`⟨*environment name*⟩`@final` is called after the proof environment display is finished, as the very last.

All proof hooks call are defined by default to defer their call to a common hook. The common hooks each take one argument (the proof environment name). They are named `\phfthm@hookproof@startcommon{`⟨*environment name*⟩`}`, `\phfthm@hookproof@startafterdisplaycommon{`⟨*environment name*⟩`}`, `\phfthm@hookproof@startlastcommon{`⟨*environment name*⟩`}`, `\phfthm@hookproof@endcommon{`⟨*environment name*⟩`}`, and `\phfthm@hookproof@finalcommon{`⟨*environment name*⟩`}`. They are all defined to be empty by default.

■ 5 Pairing theorems to proofs and proof-reference mechanism

One of the goodies provided by the phfthm package is the proof-ref mechanism, where in a theorem environment, the text "see proof on page …" is displayed to direct the reader to the location of the corresponding proof. The mechanism is deactivated by default, but can be enabled with a simple package option.

This only works if the proof is given the label of the corresponding theorem or proposition. For example:

```
\begin{theorem}[Gauss]
  \label{thm:Gauss}
  For a closed surface $S$ enclosing a volume $V$, we have
  \begin{equation}
    \oint_S\vec u\cdot d\vec S = \int_V(\vec\nabla\cdot\vec u)\,dV\ .
  \end{equation}
\end{theorem}

...

\begin{proof}[*thm:Gauss]
  ...
```

```
\end{proof}
```

The above example might produce the following output:

Theorem 17 (Gauss). For a closed surface S enclosing a volume V, we have

$$\oint_S \vec{u} \cdot d\vec{S} = \int_V (\vec{\nabla} \cdot \vec{u}) \, dV \, . \tag{42}$$

(Proof on page XXX.)

...

Proof of Theorem 17. ...

5.1 On the theorem side

On the theorem side, the proof-ref mechanism works by hacking into the definition of LaTeX's `\label`. The `\label` command should be placed first within the theorem (see example above). It is important, in theorems which use the proof-ref mechanism (on by default), to always have a corresponding label: Indeed, you may experience weird results if you don't have a theorem label, but then have labels for other objects in the theorem such as equations or itemize items.

Once the corresponding proof is detected (a proof environment with an optional argument of the form `[*thm:the-label]` for the same label `thm:the-label` as specified to the theorem, see subsection 5.2), then a text is generated (by default "Proof on page ...") and placed after the theorem. The appearance of this text is customizable (subsection 5.3).

More precisely, the hack with the `\label` command works as follows: At the beginning of the theorem, the `\label` command is redefined so that at its fist occurrence, it stores its argument as the theorem's label to use for the proof reference, it then pins down a LaTeX label as the original `\label` command would do, and finally it calls the `...@afterlabel` theorem hook (see 3.3). After the first occurrence of `\label`, the `\label` command is restored to its original LaTeX meaning in case there are other objects within the theorem which are to be referred to.

> **TIP**
>
> The `\label` hack is only active within theorem environments where the proof-ref mechanism has been enabled. Outside these environments, the `\label` macro retains its original LaTeX definition.

`\noproofref`   If, for any reason, you do not want to make sure you don't have any text "Proof on page ..." appearing (for example there is no corresponding proof because

the theorem is obvious), then you should call \noproofref immediately inside
the theorem:

```
\begin{theorem}
  \noproofref
  Theorem text ...
\end{theorem}
```

The command \noproofref temporarily disables the proof-ref mechanism
(and restores \label to LaTeX's original meaning) for the current theorem.

5.2   On the proof side

On the proof side, you just need to specify for which theorem this is the proof of.
For that (unless you override the defaults and plug in your own magic parsing;
see subsection 4.1), you should specify an optional argument to the proof which
is of the following form: \begin{proof}[*⟨label⟩], where ⟨label⟩ is the label
name you have associated with the theorem in question (see example above).

This has two effects: it sets the proof to display "*Proof of ...*," and also does some
background dark magic to display, at the location of the corresponding theorem,
some text like "*Proof on page ...*," where the page number corresponds to the
page on which this proof is located.

5.3   Customizing appearance of the proof reference text

Here we explain the workings of the \phfthm@proofrefstyle@... macros and
how they are called. It allows you to define new proof-ref styles, for example.

When the option proofref=true is given to \phfMakeTheorem
to define an environment (say mytheoremenv), then the hook
phfthm@hook@start@mytheoremenv will automatically include the following
calls:

- The macro \phfthm@proofrefstyle@⟨*proof-ref style*⟩@setup is called
  (for the proof-ref style given via the proofenvstyle=⟨*style name*⟩ key-
  value option to \phfMakeTheorem);

- The macro \phfthm@def@label@thmlabel is invoked, implementing
  the hack on the \label macro;

- The macro \phfthm@proofref@impl@start is called. This macro is ex-
  pected to be defined after calling \phfthm@proofrefstyle@⟨*proof-ref
  style*⟩@setup.

Furthermore, the \phfthm@hook@afterlabel@mytheoremenv hook will include a call to \phfthm@proofref@impl@afterlabel{⟨*label of the theorem*⟩}. Again, the latter macro is expected to be defined after calling \phfthm@proofrefstyle@⟨*proof-ref style*⟩@setup.

Finally, the hook \phfthm@hook@end@mytheoremenv includes a call to \phfthm@proofref@impl@end{⟨*label of the theorem*⟩}. Again, the latter macro is expected to be defined after calling \phfthm@proofrefstyle@⟨*proof-ref style*⟩@setup.

Hence, to define a new proof-ref style, you simply need to define a macro called \phfthm@proofrefstyle@<PROOF-REF-STYLE-NAME>@setup. This macro should include commands to locally define the macros \phfthm@proofref@impl@start, \phfthm@proofref@impl@afterlabel, and \phfthm@proofref@impl@end.

Different proof-ref styles may work similarly and want to share most of the code. A good idea is to build up on the default proof-ref style, which is highly modular and can be instantiated in different flavors. For an example, check the margin proof-ref style which does precisely that. For more documentation, check out the implementation of the default proof-ref style in subsubsection 7.4.3.

■ 6   Theorem-heading definition-like environments

A theorem-heading environment is an environment which displays in the same way as a theorem environment, but where the title may be any text (say, "Trace Distance" instead of, e.g., "Theorem 5").

By default, the phfthm package provides the thmheading environment (see subsection 2.3).

6.1   Define theorem-heading environments manually

\phfMakeThmheadingEnvironment A new theorem-heading environment can be defined by calling \phfMakeThmheadingEnvironment. The syntax is:

\phfMakeThmheadingEnvironment[⟨*key-value options*⟩]{⟨*environment name*⟩}

The key-value options may be any combination of the following:

thmstyle=⟨*theorem style name*⟩

> The theorem style to use to display the environment. You may specify here any default *AMS* style (plain, remark or definition), or any other \newtheoremstyle-defined style.

<u>internalcounter=⟨*name of counter*⟩</u>

> The name of a counter which will internally track environment instances. By default, a common internal counter is used for all theorem-heading environments (named `phfthmheadingcounter`). The counter must be already defined (see LaTeX's `\newcounter`).

You can also use `\label` and `\ref` (the latter simply displays the given title).

6.2 Available hooks for theorem-heading environments

The hook `\phfthm@hook@thmheading@...@start{⟨*title*⟩}` is invoked at start, within an internal environment created with `\newtheorem`. This hook accepts one argument, the title of the theorem-heading.

The hook `\phfthm@hook@thmheading@...@end` is called at the end, but still within the internal theorem environment.

Replace the dots with the name of the theorem-heading environment (such as `thmheading`).

By default, these hooks simply call the common hooks `\phfthm@hook@thmheading@start` and `\phfthm@hook@thmheading@end`. These common hooks are empty by default.

■ 7 Implementation

First, load some packages. General toolboxes:

```
1 \RequirePackage{xkeyval}
2 \RequirePackage{etoolbox}
```

To define alias counters for theorems, load aliascnt:

```
3 \RequirePackage{aliascnt}
```

And finally, load the *AMS* math and theorem (amsmath, amsthm) packages:

```
4 \RequirePackage{amsmath}
5 \RequirePackage{amsthm}
```

7.1 Generic Internal Stuff

`\phfthm@internal@execattribs` Internal command: execute all definitions given in list of attributes. This was copy-pasted from a similar definition in the phfnote package.

#1 = prefix to look for attributes

19

#2 = name of what #1 represents, to use in message in case attribute is not found

#3 = list of attributes

```
6 \def\phfthm@internal@execattribs#1#2#3{%
7   \@for\next:=#3\do{%
8     \ifcsname #1\next\endcsname%
9       \csname #1\next\endcsname%
10    \else%
11      \PackageWarning{phfthm}{Unknown #2: '\next'. Ignoring.}
12    \fi
13  }
14 }
```

7.2 Definitions for theorem environments

7.2.1 \phfMakeTheorem: definition of a new theorem environment

First, define some key-value syntax accepted by the \phfMakeTheorem command.

```
15 \define@cmdkey{phfmkthm}{counter}{}
16 \define@boolkey{phfmkthm}{aliascounter}[true]{}
17 \define@cmdkey{phfmkthm}{thmstyle}{}
18 \define@boolkey{phfmkthm}{defnostar}[true]{}
19 \define@boolkey{phfmkthm}{defstar}[true]{}
20 \define@boolkey{phfmkthm}{proofref}[true]{}
21 \define@cmdkey{phfmkthm}{proofrefstyle}{}
```

\phfMakeTheorem Define a new theorem environment. The syntax is \phfMakeTheorem [⟨*options*⟩]{⟨*theorem environment name*⟩}{⟨*Theorem Display Name*⟩}. For example: \phfMakeTheorem[counter=thmcounter]{prop}{Proposition}

```
22 \newcommand\phfMakeTheorem[3][]{% }
```

Handle the [options]. First, ensure that the defaults are set, and then, parse the input.

```
23   \KV@phfmkthm@aliascountertrue%
24   \def\cmdKV@phfmkthm@counter{}%
25   \def\cmdKV@phfmkthm@thmstyle{}%
26   \KV@phfmkthm@defnostartrue%
27   \KV@phfmkthm@defstartrue%
28   \KV@phfmkthm@proofreftrue%
29   \def\cmdKV@phfmkthm@proofrefstyle{default}%
30   \setkeys{phfmkthm}{#1}%
```

Now, react to whatever was given in the options.

Set the theorem style, if requested.[1]

```
31  \if\relax\detokenize\expandafter{\cmdKV@phfmkthm@thmstyle}\relax%
32  \else%
33    \theoremstyle{\cmdKV@phfmkthm@thmstyle}%
34  \fi%
```

If requested, define the default, unstarred version of the theorem. Use `\newtheorem` for that, which we make sure to call appropriately depending on whether a separate counter is requested or not. Make sure also to define `\...autorefname` for `\autoref`. If an alias counter is requested, create it and pass that one to `\newtheorem`.

At this point, we create a theorem named `phfthm@...` using `\newtheorem` (because we still want to add calls to hooks).

```
35  \ifKV@phfmkthm@defnostar%
36    \if\relax\detokenize\expandafter{\cmdKV@phfmkthm@counter}\relax%
```

—in case we use a separate counter (if <u>counter=</u>):

```
37      \newtheorem{phfthm@#2}{#3}%
38      \csdef{phfthm@#2autorefname}{#3}%
39    \else%
40      \ifKV@phfmkthm@aliascounter%
```

—in case we make a distinct alias counter, eg. for use with `\autoref`:

```
41        \newaliascnt{#2}{\cmdKV@phfmkthm@counter}%
42        \newtheorem{phfthm@#2}[#2]{#3}%
43        \aliascntresetthe{#2}%
44        \csdef{#2autorefname}{#3}%
45      \else%
```

—in case we directly instruct `\newtheorem` to use the other counter (does not work with `\autoref`):

```
46        \newtheorem{phfthm@#2}[\cmdKV@phfmkthm@counter]{#3}%
47      \fi%
48    \fi%
```

And also define the actual theorem environment, adding calls to hooks.

```
49      \newenvironment{#2}[1][]{%
50        \begin{phfthm@#2}[##1]%
51          \begingroup%
52            \csname phfthm@hook@start@#2\endcsname{##1}%
53        }{%
54            \csname phfthm@hook@end@#2\endcsname%
```

---

[1]The construct `\if\relax\detokenize{...}\relax` tests whether ... is empty: see *http://tex.stackexchange.com/a/53091/32188*

```
55        \endgroup%
56        \end{phfthm@#2}%
57      }%
```

Define hooks specific to this theorem with sensible defaults. If proof-ref is on, call the appropriate callbacks. Then, call the common hooks (see \phfthm@hook@startcommonnostar, \phfthm@hook@afterlabelcommon and \phfthm@hook@endcommonnostar, detailed in subsection 3.3).

```
58      \csedef{phfthm@hook@start@#2}##1{%
59        \ifKV@phfmkthm@proofref%
60          \expandafter\noexpand%
61            \csname phfthm@proofrefstyle@\cmdKV@phfmkthm@proofrefstyle @setup\endcsname%
62          \noexpand\phfthm@def@label@thmlabel{#2}%
63          \noexpand\phfthm@proofref@impl@start%
64        \fi%
65        \noexpand\phfthm@hook@startcommonnostar{#2}{##1}%
66      }%
67      \csedef{phfthm@hook@afterlabel@#2}{%
68        \ifKV@phfmkthm@proofref%
69          \noexpand\phfthm@proofref@expandthmlabeltoarg%
70            \noexpand\phfthm@proofref@impl@afterlabel%
71        \fi%
72        \noexpand\phfthm@hook@afterlabelcommon{#2}%
73      }%
74      \csedef{phfthm@hook@end@#2}{%
75        \ifKV@phfmkthm@proofref%
76          \noexpand\phfthm@proofref@expandthmlabeltoarg%
77            \noexpand\phfthm@proofref@impl@end%
78        \fi%
79        \noexpand\phfthm@hook@endcommonnostar{#2}%
80      }%
81    \fi%
```

If requested, define the starred version of the theorem. We call \newtheorem* to define the base theorem environment (which we call phfthm@...), after which as above we define the actual environment which also calls the relevant hooks.

```
82    \ifKV@phfmkthm@defstar%
83      \newtheorem*{phfthm@#2*}{#3}%
84      \newenvironment{#2*}[1][]{%
85        \begin{phfthm@#2*}[##1]%
86          \begingroup%
87            \csname phfthm@hook@start@#2*\endcsname{##1}%
88      }{%
89            \csname phfthm@hook@end@#2*\endcsname%
90          \endgroup%
91        \end{phfthm@#2*}%
92      }%
93    \fi%
```

Finally, define the default hooks specific to the starred version of the theorem (see subsection 3.3).

```
94    \csdef{phfthm@hook@start@#2*}##1{\phfthm@hook@startcommonstar{#2}{##1}}%
95    \csdef{phfthm@hook@end@#2*}{\phfthm@hook@endcommonstar{#2}}%
96 }
```

7.2.2 Default hooks for theorems

`\phfthm@hook@startcommonnostar`
`\phfthm@hook@startcommonstar`
`\phfthm@hook@startcommon`
Common default hooks definitions for start of the theorems.

For all three of these hooks, we have #1 = theorem name, e.g. `proposition` and #2 = full (optional) title of proposition, if given, or empty.

Make sure to invoke the `\label` re-definition hack only for non-starred theorems/propositions; indeed, if no theorem label is set we don't want to interfere with labels set to inner equations, itemizes etc. Hence, call `\phfhtm@def@label@thmlabel` only in the "nostar" hook.

```
97 \def\phfthm@hook@startcommonnostar#1#2{%
98    \phfthm@hook@startcommon{#1}{#2}%
99 }
100 \def\phfthm@hook@startcommonstar#1#2{%
101    \phfthm@hook@startcommon{#1}{#2}%
102 }
103 \def\phfthm@hook@startcommon#1#2{%
```

Furthermore, in any case, set the `\postdisplaypenalty` to avoid an orphan line on a new page after display equation.

```
104    \postdisplaypenalty=10000\relax%
105 }
```

`\phfthm@hook@afterlabelcommon`
`\phfthm@hook@endcommonnostar`
`\phfthm@hook@endcommonstar`
`\phfthm@hook@endcommon`
Further hooks, for after the theorem main `\label` command (`\phfthm@hook@afterlabelcommon`) and for the end of the theorem.

```
106 \def\phfthm@hook@afterlabelcommon#1{}
107 \def\phfthm@hook@endcommonnostar#1{\phfthm@hook@endcommon{#1}}
108 \def\phfthm@hook@endcommonstar#1{\phfthm@hook@endcommon{#1}}
109 \def\phfthm@hook@endcommon#1{}
```

7.2.3 Proof-ref mechanism (on the theorem side)

These macros enable the proof-ref mechanism (section 5). The theorem's label is stored upon calling `\label`, because we (locally) hack into the definition of `\label`. (After the first usage of `\label` its meaning is restored.)

23

<dl>
<dt>\phfthm@def@label@thmlabel</dt>
<dd>

Main macro to invoke at the beginning of the theorem environment, so that the theorem label is stored in a local macro once \label is invoked. This hacks the \label macro locally. Here, #1 = the theorem environment name, e.g. proposition.

</dd>
</dl>

```
110 \def\phfthm@def@label@thmlabel#1{%
111   \ifdefined\phfthm@old@label
112     \PackageWarning{phfthm}{Internal inconsistency: \string\phfthm@def@label@thmlabel
113       called twice for the same theorem environment!}
114   \else
115     \let\phfthm@old@label\label%
116     \edef\label{\noexpand\phfthm@thmlabel{#1}}%
117   \fi
118 }
```

<dl>
<dt>\phfthm@thmlabel</dt>
<dd>

The first call to \label within the theorem redirects to the macro \phfthm@thmlabel. (Applies to theorem environments for which \phfthm@def@label@thmlabel was called, which is the default).

Here #1 = theorem environment name, e.g. proposition; and #2 = the label value (argument to the \label macro).

</dd>
</dl>

```
119 \def\phfthm@thmlabel#1#2{%
```

First, store the label value into a macro called \phfthm@val@thmlabel.

```
120   \def\phfthm@val@thmlabel{#2}%
```

Then, call the original \label macro to do what LaTeX would normally do for a \label{…} call.

```
121   \phfthm@old@label{#2}%
```

Restore the old \label definition, in case there are other items in the theorem environment such as equations, itemizes etc. which may themselves have \label's.

```
122   \let\label\phfthm@old@label%
```

Invoke the \phfthm@hook@afterlabel@thmname hook for this theorem environment.

```
123   \csname phfthm@hook@afterlabel@#1\endcsname%
```

Finally, ignore any spaces following the \label command. (Maybe we should have done something with \@bsphack and \@esphack but oh well…

```
124   \ignorespaces%
125 }
```

24

7.3   Definitions for proof environments

Improved, smarter `proof` environments.

`\phfthm@old@proof`
`\endphfthm@old@proof`   Save old `proof` environment provided by amsthm.

```
126 \let\phfthm@old@proof\proof
127 \let\endphfthm@old@proof\endproof
```

`\proofname`   And provide a default name for proofs (this should normally already be provided by amsthm).

```
128 \providecommand\proofname{Proof}
```

`\proofofname`   Default text to display when we want to say e.g. "Proof of Theorem 3."

```
129 \def\proofofname#1{\proofname\space of #1}
```

The default counter for proofs. The value of this counter is typically not displayed, we just use it to pin down anchors for labels for cross references.

```
130 \newcounter{phfthmproofcnt}
```

Utility: to see if an argument was specified (possibly empty) to the proof environment.

```
131 \def\phfthm@NOPROOFARG{}
132 \def\phfthm@test@NOPROOFARG{\phfthm@NOPROOFARG}
```

7.3.1   Define a proof environment: `\phfMakeProofEnv`

Declare some key-value options accepted by `\phfMakeProofEnv`. See subsection 4.1 for the documentation of these options.

```
133 \define@cmdkey{phfmkprf}{displayenv}{}
134 \define@cmdkey{phfmkprf}{defaultproofname}{}
135 \define@boolkey{phfmkprf}{override}[true]{}
136 \define@cmdkey{phfmkprf}{internalcounter}{}
137 \define@cmdkey{phfmkprf}{proofofname}{}
138 \define@boolkey{phfmkprf}{parselabel}[true]{}
139 \define@cmdkey{phfmkprf}{parselabelcmd}{}
```

`\phfMakeProofEnv`   Make a proof environment. Syntax: `\phfMakeProofEnv[`⟨*options*⟩`]{`⟨*proof environment name*⟩`}`.

```
140 \newcommand\phfMakeProofEnv[2][]{%
```

25

Parse the key-value options. First, make sure that all the defaults are set, then parse the options.

```
141   \KV@phfmkprf@overridefalse%
142   \def\cmdKV@phfmkprf@displayenv{*}%
143   \def\cmdKV@phfmkprf@defaultproofname{\proofname}%
144   \def\cmdKV@phfmkprf@internalcounter{phfthmproofcnt}%
145   \def\cmdKV@phfmkprf@proofofname{\proofofname}%
146   \KV@phfmkprf@parselabeltrue
147   \def\cmdKV@phfmkprf@parselabelcmd{\phfthm@proof@parselabel}%
148   \setkeys{phfmkprf}{#1}%
```

The meaning of the options are detailed in subsection 4.1.

The general idea here is first to pre-process all the options, and save all the useful information in macros named \phfthm@prfenv@<proof-environment-name>@val@<something>.
Then, we can define the begin/end environment macros which will recall the saved information.

Take care of the display environment to use. Recall that if displayenv=*, we use our own default; if displayenv= (empty), there is no display environment. Here, we set \phfthm@prfenv@<proof-environment-name>@val@displayenv to the name of the environment to use (possibly empty), for later reference.

```
149   \def\phfmkprf@tmp@star{*}%
150   \ifx\cmdKV@phfmkprf@displayenv\phfmkprf@tmp@star\relax%
151     \def\cmdKV@phfmkprf@displayenv{phfthm@proof@defaultdisplayenv}%
152   \fi
153   \cslet{phfthm@prfenv@#2@val@displayenv}\cmdKV@phfmkprf@displayenv%
```

Process the default proof name. If none is given, use \proofname and pass no option to the underlying display environment whenever the proof environment is called with no option. Here, we set \phfthm@prfenv@<proof-environment-name>@val@defaultproofnameargs and \phfthm@prfenv@<proof-environment-name>@val@setdefaultprooftitle; the former is the tokens to put in front of the proof environment invocation in case no explicit proof title is given to the proof environment while the latter contains the command to set \phfthm@val@prooftitle to the default proof name.

```
154   \if\relax\detokenize\expandafter{\cmdKV@phfmkprf@defaultproofname}\relax%
155     \csdef{phfthm@prfenv@#2@val@defaultproofnameargs}{}%
156     \csdef{phfthm@prfenv@#2@val@setdefaultprooftitle}{%
157       \def\phfthm@val@prooftitle{\proofname}}%
158   \else
159     \csedef{phfthm@prfenv@#2@val@defaultproofnameargs}{%
160       [\expandonce{\cmdKV@phfmkprf@defaultproofname}]}%
161     \csedef{phfthm@prfenv@#2@val@setdefaultprooftitle}{%
162       \noexpand\def\noexpand\phfthm@val@prooftitle{%
```

```
163            \expandonce{\cmdKV@phfmkprf@defaultproofname}}}%
164    \fi
165    \csedef{phfthm@prfenv@#2@val@parselabelandmkdisplayargs}##1{%
166      \ifKV@phfmkprf@parselabel
167        \expandonce\cmdKV@phfmkprf@parselabelcmd{##1}%
168      \else
169        \noexpand\phfthm@proof@noparselabel{##1}%
170      \fi
171      \noexpand\def\noexpand\phfthm@val@displayargs{[{%
172          \expandafter\noexpand\csname phfthm@prfenv@#2@val@proofofname\endcsname
173          {\noexpand\phfthm@val@prooftitle}%
174        }]}%
175    }
```

Store the macro which creates the "Proof of..." text (`proofofname` option).

```
176    \cslet{phfthm@prfenv@#2@val@proofofname}\cmdKV@phfmkprf@proofofname%
```

Create the macro which will take care of pinning down the label for the proof-ref (see section 5). This macro first ref-steps the internal counter and then pins down a label, if appropriate.

```
177    \csdef{phfthm@prfenv@#2@val@pinproofanchor}{%
178      \csname phfthm@prfenv@#2@val@refstepinternalcounter\endcsname%
179      \if\relax\detokenize\expandafter{\phfthm@val@proofoflabel}\relax\else%
180        \edef\phfthm@tmp@larg{{proof:\phfthm@val@proofoflabel}}%
181        \expandafter\label\phfthm@tmp@larg%
182      \fi
183    }%
184    %
```

The command to ref-step the internal proof counter. Use the value of the `internalcounter` command option.

```
185    \csedef{phfthm@prfenv@#2@val@refstepinternalcounter}{%
186      \noexpand\refstepcounter{\cmdKV@phfmkprf@internalcounter}}%
```

Make macros `\phfthm@prfenv@<proof-environment-name>@val@displayenvbegincmd` and `\phfthm@prfenv@<proof-environment-name>@val@displayenvendcmd`, which essentially expand to `\begin{<the-display-env>}` and `\end{<the-display-env>}` for the display environment given in the option `displayenv`.

```
187    \if\relax\detokenize\expandafter{\cmdKV@phfmkprf@displayenv}\relax%
188      \csdef{phfthm@prfenv@#2@val@displayenvbegincmd}##1{}%
189      \csdef{phfthm@prfenv@#2@val@displayenvendcmd}##1{}%
190    \else
191      \csedef{phfthm@prfenv@#2@val@displayenvbegincmd}##1{%
192        \noexpand\begin{\csname phfthm@prfenv@#2@val@displayenv\endcsname}##1}%
193      \csedef{phfthm@prfenv@#2@val@displayenvendcmd}##1{%
194        \noexpand\end{\csname phfthm@prfenv@#2@val@displayenv\endcsname}##1}%
```

```
195    \fi
```

See if we need to call `\newenvironment` or `\renewenvironment`, depending on the value of the <u>override</u> option.

```
196    \def\phfthm@tmp@defcmd{\newenvironment}%
197    \ifKV@phfmkprf@override\def\phfthm@tmp@defcmd{\renewenvironment}\fi%
```

Finally, (re-)define the environment. The default value of the optional argument is the token `\phfthm@NOPROOFARG`, which indicates that no argument was provided.

Start by storing the value of the argument into a macro, and then call the "start" hook (see proof hooks in).

```
198    \phfthm@tmp@defcmd{#2}[1][\phfthm@NOPROOFARG]{%
199      \def\phfthm@val@proofarg{##1}%
200      \csname phfthm@hookproof@#2@start\endcsname%
```

First, parse the optional argument into proof label (maybe) and proof title. If no optional argument was given, don't give any argument to the underlying display environment. If an empty argument was given, set some defaults; otherwise, use the necessary command to potentially parse the label and create the proper arguments for the underlying display environment.

```
201      \ifx\phfthm@val@proofarg\phfthm@test@NOPROOFARG\relax%
202        \def\phfthm@val@proofoflabel{}%
203        \csname phfthm@prfenv@#2@val@setdefaultprooftitle\endcsname%
204        \letcs\phfthm@val@displayargs{phfthm@prfenv@#2@val@defaultproofnameargs}%
205      \else%
206        \if\relax\detokenize{##1}\relax%
207          \def\phfthm@val@proofoflabel{}%
208          \csname phfthm@prfenv@#2@val@setdefaultprooftitle\endcsname%
209          \def\phfthm@val@displayargs{[{%
210              \csname phfthm@prfenv@#2@val@proofofname\endcsname
211              {\phfthm@val@prooftitle}%
212            }]}%
213        \else
214          \csname phfthm@prfenv@#2@val@parselabelandmkdisplayargs\endcsname{##1}%
215        \fi
216      \fi%
```

Define the `\phfPinProofAnchor` command (locally) in case the display formatting environment takes care of where to place the anchor already.

```
217      \def\phfPinProofAnchor{%
218        \csname phfthm@prfenv@#2@val@pinproofanchor\endcsname%
219        \global\let\phfPinProofAnchor\relax}%
```

[Also provide the obsolete `\phfthmPinProofAnchor` which I previously had in older versions of this package:]

```
220        \def\phfthmPinProofAnchor{\phfPinProofAnchor}%
```

Start the proof's display environment.   Don't be fooled here by
the curly braces after \x, it only protects the argument to the
\phfthm@prfenv@#2@val@displayenvbegincmd command itself: the
\phfthm@val@displayargs are still just tokens which will be expanded in
front of the \begin{<proof-display-env>} command.

```
221        \def\x{\csname phfthm@prfenv@#2@val@displayenvbegincmd\endcsname}%
222        \expandafter\x\expandafter{\phfthm@val@displayargs}%
```

And call the corresponding hook:

```
223        \csname phfthm@hookproof@#2@startafterdisplay\endcsname%
```

If   required,   pin   anchor   after   the   proof-display-environment.
(\phfPinProofAnchor auto-destructs after first use, so it's safe to
potentially call it a second time here). Then, call the corresponding hook.

```
224        \phfPinProofAnchor%
225        \expandafter\noexpand\csname phfthm@hookproof@#2@startlast\endcsname%
226    }%
```

Now, the definitions for the "end" part of the environment. Just call the relevant
hooks and close the display environment.

```
227    {%
228      \expandafter\noexpand\csname phfthm@hookproof@#2@end\endcsname%
229      \csname phfthm@prfenv@#2@val@displayenvendcmd\endcsname
230      \expandafter\noexpand\csname phfthm@hookproof@#2@final\endcsname%
231    }%
```

Finally, define the default values of the proof-environment-specific hooks. These
just call the corresponding global hooks (see subsection 4.2).

```
232  \csdef{phfthm@hookproof@#2@start}{\phfthm@hookproof@startcommon{#2}}%
233  \csdef{phfthm@hookproof@#2@startafterdisplay}{%
234    \phfthm@hookproof@startafterdisplaycommon{#2}}%
235  \csdef{phfthm@hookproof@#2@startlast}{\phfthm@hookproof@startlastcommon{#2}}%
236  \csdef{phfthm@hookproof@#2@end}{\phfthm@hookproof@endcommon{#2}}%
237  \csdef{phfthm@hookproof@#2@final}{\phfthm@hookproof@finalcommon{#2}}%
238 }
```

7.3.2   Common hooks for proofs

The hooks are documented in subsection 4.2.

```
239 \def\phfthm@hookproof@startcommon#1{}
240 \def\phfthm@hookproof@startafterdisplaycommon#1{}
```

```
241 \def\phfthm@hookproof@startlastcommon#1{}
242 \def\phfthm@hookproof@endcommon#1{}
243 \def\phfthm@hookproof@finalcommon#1{}
```

7.3.3 Default display environment for proofs

phfthm@proof@defaultdisplayenv Provide an environment which displays a proof in a similar fashion as *AMS*', but with some small additional features.

```
244 \newenvironment{phfthm@proof@defaultdisplayenv}[1][\proofname]{%
245   \par
246   \pushQED{\qed}%
247   \normalfont \topsep6\p@\@plus6\p@\relax
248   \trivlist\item\relax
249   \phfPinProofAnchor
250   \phfthm@ProofTitleFmt{#1}%
251   \phfthm@ProofTitleHspace
252   \ignorespaces
253 }{%
254   \popQED\endtrivlist\@endpefalse
255 }
```

\phfthm@ProofTitleFmt These macros may be overridden to change the proof title appearance.
\phfthm@ProofTitleHspace

```
256 \def\phfthm@ProofTitleFmt#1{%
257   {\itshape #1.}%
258 }
259 \def\phfthm@ProofTitleHspace{%
260   \hspace{1.5ex plus 0.5ex minus 0.2ex}%
261 }
```

7.3.4 Parsing the proof argument

These macros parse the argument of the proof environment to see if it is of the form *<some-label> (see section 5).

\phfthm@proof@parselabel Call \phfthm@proof@parselabel{⟨*proof environment argument*⟩} to parse the argument string. This macro will set \phfthm@val@proofoflabel and \phfthm@val@prooftitle to appropriate values (respectively, the label name of the corresponding theorem and a representative title such as "Theorem 6").

```
262 \def\phfthm@proof@parselabel#1{%
263   \phfthm@proof@parselabel@maybelabel#1\phfthm@proof@parselabel@END%
264 }
265 \def\phfthm@proof@parselabel@maybelabel{%
266   \@ifnextchar*\phfthm@proof@parselabel@label\phfthm@proof@parselabel@title%
267 }
268 \def\phfthm@proof@parselabel@label*#1\phfthm@proof@parselabel@END{%
```

The use of \detokenize here is a trick to make sure that all chars in the label text have a non-active category (e.g. we would have problems, e.g., if in the label "thm:gauss" the ":" is an active char—such as in French):

```
269    \edef\phfthm@val@proofoflabel{\detokenize{#1}}%
270    \def\phfthm@val@prooftitle{\phfthm@autoref{#1}}%
271 }
272 \def\phfthm@proof@parselabel@title#1\phfthm@proof@parselabel@END{%
273    \def\phfthm@val@proofoflabel{}%
274    \def\phfthm@val@prooftitle{#1}%
275 }
```

\phfthm@proof@noparselabel Enjoys the same syntax as \phfthm@proof@parselabel, i.e., it is a drop-in replacement for the latter, except that it invariably sets \phfhtm@val@proofoflabel to an empty value and \phfthm@val@prooftitle to the argument itself. You could use this as a parselabelcmd macro if you didn't want to parse the label.

```
276 \def\phfthm@proof@noparselabel#1{%
277    \def\phfthm@val@proofoflabel{}%
278    \def\phfthm@val@prooftitle{#1}%
279 }
```

In order to look up what we are a proof of, we use \autoref provided by the hyperref package. If it is not available, fall back to the regular \ref command.

```
280 \def\phfthm@autoref{\ref}
281 \AtBeginDocument{%
282    \@ifpackageloaded{hyperref}{\def\phfthm@autoref{\autoref}}{}
283 }
```

7.4 Implementation of the proof-ref machinery

7.4.1 Small general stuff

\proofonname The macro \proofonname displays "Proof on ...." Here, #2 is the full page reference and #1 is the label name of the referenced theorem.

```
284 \providecommand\proofonname[2]{Proof on #2.}
```

\proofrefsize Format the proof reference "Proof on page ...". This macro is meant to set the font size (or other font properties), but it may also be defined to take one argument, the proof reference text.

```
285 \def\proofrefsize{\footnotesize}
```

\noproofref Use \noproofref inside a theorem to signify that no proof reference should be attempted.

The implementation just defines \phfthm@val@noproofref. If this macro is defined, then no proof ref should be generated for the current thmlabel. Also, restore \label to its original definition in case it was overridden.

```
286 \def\noproofref{%
287   \def\phfthm@val@noproofref{1}%
288   \ifdefined\phfthm@old@label \let\label\phfthm@old@label \fi%
289 }
```

\phfthm@proofref@warnnolabel Produce a warning that no label was provided in order to infer the proof reference.

```
290 \def\phfthm@proofref@warnnolabel{%
291   \PackageWarning{phfthm}{No label provided for proof reference!}%
292 }
```

\phfthm@proofref@expandthmlabeltoarg Utility to expand the value of \phfthm@val@thmlabel as an argument to a callback command. #1 = the macro to relay the call to.

```
293 \def\phfthm@proofref@expandthmlabeltoarg#1{%
```

First, check if the proof-ref mechanism was explicitly temporarily disabled, and do nothing if that is the case.

```
294   \ifdefined\phfthm@val@noproofref\relax%
295   \else%
```

Then make sure \phfthm@val@thmlabel is defined (maybe empty), and then either call the callback macro #1 with the value of \phfthm@val@thmlabel as argument, or generate a warning if that value is empty.

```
296     \providecommand\phfthm@val@thmlabel{}%
297     \edef\phfthm@tmpa{{\phfthm@val@thmlabel}}%
298     \expandafter\notblank\phfthm@tmpa{%
299       \expandafter#1\phfthm@tmpa%
300     }{%
301       \phfthm@proofref@warnnolabel% no label provided
302     }%
303   \fi%
304 }
```

7.4.2 Utilities for interacting with \autoref labels

In this context, we also need some generic utilities for interacting with \autoref labels.

`\phfthm@autorefnameof` The macro `\phfthm@autorefnameof` extracts the name of the counter which generated this reference (e.g. "section" or "theorem").

```
305 \def\phfthm@autorefnameof#1{%
```

Extract the counter part of the reference `section.NN`, which is 4th element in the `\r@label` macro. (Code extracted from `hyperref.sty`.)

```
306   \expandafter\ifx\csname r@#1\endcsname\relax%
307     \textbf{??}%
308   \else%
309     \expandafter\expandafter\expandafter\phfthm@HyPsd@autorefname%
310         \csname r@#1\endcsname{}{}{}{}\@nil%
311   \fi%
312 }
313 \def\phfthm@HyPsd@autorefname#1#2#3#4#5\@nil{%
314   \ifx\\#4\\%
315   \else%
316     \phfthm@HyPsd@@autorefname#4.\@nil%
317   \fi%
318 }
319 \def\phfthm@HyPsd@@autorefname#1.#2\@nil{%
320   \ltx@IfUndefined{#1autorefname}{%
321     \ltx@IfUndefined{#1name}{%
322     }{%
323       \csname#1name\endcsname%
324     }%
325   }{%
326     \csname#1autorefname\endcsname%
327   }%
328 }
```

`\phfthm@min@pageref` A minimal pageref macro, which just extracts the page number on which the given label is located.

The dark magic going on here is beyond me. The code was copied from `hyperref.sty`, in "`\def\HyPsd@@@pageref...`" and seems to work.

```
329 \def\phfthm@min@pageref#1{%
330   \ifcsname r@#1\endcsname%
331     \expandafter\expandafter\expandafter\expandafter
332     \expandafter\expandafter\expandafter\@car
333     \expandafter\expandafter\expandafter\@gobble
334     \csname r@#1\endcsname{}\@nil
335   \else%
336     0%
337   \fi%
338 }
```

7.4.3 Default proof-ref style, with basic machinery

Now we define the relevant callbacks for the default style. See documentation in subsection 5.3. Recall a proof-ref style just needs to define \phfthm@proofrefstyle@<stylename>@setup, which in turn should just define the callbacks \phfthm@proofref@impl@start, \phfthm@proofref@impl@afterlabel and \phfthm@proofref@impl@end. For our default style, these callbacks further call other callbacks of the form \phfthm@proofref@impl@..., such that these definitions can be re-used to create new styles. The main proof-ref generation routine is \phfthm@proofrefstyle@default@main, which can be used for either the ...@afterlabel or the ...@end callback.

`\phfthm@proofrefstyle@default@fmt` Format and display the proof reference. #1 = the theorem's label (e.g. prop:1); #2 = the full reference (e.g. "page XYZ").

This macro is the default value of the callback \phfthm@proofref@impl@fmt, which is called by the default style itself.

Use correct spacing for right-aligning the reference.[2] If there is room on the current line, just right-align the proof-ref text; if not, add it on a separate line. [We can achieve this with the sequences \hfil\null\hfil: if there is space, it all fits on the same line, if not, the line breaks at the \null point.]

```
339 \def\phfthm@proofrefstyle@default@fmt#1#2{%
340   {\parfillskip=0pt\relax%
341     \hfil\null\hfil\null\hfil%
342     \hbox{\proofrefsize{(\proofonname{#1}{#2})}}}\par}%
343 }
```

`@proofrefstyle@default@fmtfarback`
`roofrefstyle@default@fmtfarahead`
`@proofrefstyle@default@fmtcloseby`
These macros are the default values of the callbacks \phfthm@proofref@impl@fmtfarback, \phfthm@proofref@impl@fmtfarahead, and \phfthm@proofref@impl@fmtcloseby, which are called by the default style itself. These callbacks define how to format and (possibly not) display the proof reference depending on whether the proof is "far behind" (several pages back), "far ahead" (several pages ahead) or "close by" (neither far back nor far ahead), as defined by \phfProofrefPageBackTolerance and \phfProofrefPageAheadTolerance.

```
344 \def\phfthm@proofrefstyle@default@fmtfarback#1#2{%
345   \phfthm@proofref@impl@fmt{#1}{#2}}
346 \def\phfthm@proofrefstyle@default@fmtfarahead#1#2{%
347   \phfthm@proofref@impl@fmt{#1}{#2}}
348 \def\phfthm@proofrefstyle@default@fmtcloseby#1#2{}
```

---

[2]Thanks *http://tex.stackexchange.com/a/43239/32188*!

\phfProofrefPageBackTolerance	The macros \phfProofrefPageBackTolerance and
\phfProofrefPageAheadTolerance	\phfProofrefPageAheadTolerance define how many pages back

or ahead the proof should be in order to consider it "far back" or "far ahead."

Either value may be set to -1 to force the proof to be considered "far back" or "far ahead."

```
349 \newcommand\phfProofrefPageBackTolerance{1}
350 \newcommand\phfProofrefPageAheadTolerance{1}
```

Define the internal counter which allows to check on which page we are at the place of the proof reference. This is used by \phfthm@proofrefstyle@default@main.

```
351 \newcounter{phfthmInternalProofrefCounter}
```

phfthm@proofrefstyle@default@main The main proof-ref generation routine. The argument #1 is the current label of the theorem; the referenced label is proof:#1.

```
352 \def\phfthm@proofrefstyle@default@main#1{%
```

Check to see if the proof is far away ahead or back (as defined by the tolerance macros above). Depending on each case, call the corresponding callbacks.[3]

```
353    \refstepcounter{phfthmInternalProofrefCounter}%
354    \label{internalproofref\thephfthmInternalProofrefCounter}%
355    \edef\phfthm@proofref@tmp@proofpage{\phfthm@min@pageref{proof:#1}}%
356    \edef\phfthm@proofref@tmp@thispage{%
357      \phfthm@min@pageref{internalproofref\thephfthmInternalProofrefCounter}}%
358    \edef\phfthm@proofref@tmp@pagediff{%
359      \the\numexpr\phfthm@proofref@tmp@proofpage-\phfthm@proofref@tmp@thispage\relax}%
```

If the proof is "far back," call the corresponding callback.

```
360    \ifnum\numexpr\phfthm@proofref@tmp@pagediff\relax%
361        <\numexpr-\phfProofrefPageBackTolerance\relax%
362      \phfthm@proofref@impl@fmtfarback{#1}{\autopageref{proof:#1}}%
363    \else%
```

If the proof is "far ahead," call the corresponding callback.

```
364      \ifnum\numexpr\phfthm@proofref@tmp@pagediff\relax%
365          >\numexpr\phfProofrefPageAheadTolerance\relax%
366        \phfthm@proofref@impl@fmtfarahead{#1}{\autopageref{proof:#1}}%
```

[3]See *http://tex.stackexchange.com/a/2526* to test whether ref is on same page. Note that was problematic, probably due to hyperref. I needed to use my own \phfthm@min@pageref without any hyper linking mechanism in place.

Otherwise, it is close by.

```
367     \else%
368       \phfthm@proofref@impl@fmtcloseby{#1}{\autopageref{proof:#1}}%
369     \fi%
370   \fi%
371 %%   [\number\numexpr\phfthm@proofref@tmp@proofpage\relax{} vs % DEBUG
372 %%   \number\numexpr\phfthm@proofref@tmp@thispage\relax or % DEBUG
373 %%   \number\numexpr1+\phfthm@proofref@tmp@thispage\relax] % DEBUG
374 }
```

phfthm@proofrefstyle@default@setup The main set-up macro for the `default` proof-ref style. It sets all the call-backs to the default ones.

```
375 \def\phfthm@proofrefstyle@default@setup{%
376   \let\phfthm@proofref@impl@start\relax
377   \let\phfthm@proofref@impl@afterlabel\@gobble
378   \let\phfthm@proofref@impl@end\phfthm@proofrefstyle@default@main
379   \let\phfthm@proofref@impl@fmtfarback\phfthm@proofrefstyle@default@fmtfarback
380   \let\phfthm@proofref@impl@fmtfarahead\phfthm@proofrefstyle@default@fmtfarahead
381   \let\phfthm@proofref@impl@fmtcloseby\phfthm@proofrefstyle@default@fmtcloseby
382   \let\phfthm@proofref@impl@fmt\phfthm@proofrefstyle@default@fmt
383 }
```

7.4.4 Other proof-ref styles: only `margin` **for now**

These styles simply use the same mechanism as the default style, but plug in different sub-callbacks.

phfthm@proofrefstyle@margin@setup Set-up macro for the "`margin`" proof-ref style (displays the proof reference in the margin of the page).

```
384 \def\phfthm@proofrefstyle@margin@setup{%
385   \phfthm@proofrefstyle@default@setup
```

The proof reference should be displayed directly at the top, not at the end of the theorem, so plug in \phfthm@proofref@default@main onto ...@afterlabel and not onto ...@end. Don't forget that these macros accept one argument, the theorem label.

```
386   \let\phfthm@proofref@impl@afterlabel\phfthm@proofrefstyle@default@main
387   \let\phfthm@proofref@impl@end\@gobble
```

Define the formatting callback to put the note in the margin of the page using a \marginpar. We need \leavevmode to make sure it's aligned properly vertically with the paragraph.[4]

---

[4]See *http://tex.stackexchange.com/a/16161/32188*

```
388  \def\phfthm@proofref@impl@fmt##1##2{%
389    \leavevmode\marginpar{\proofrefsize{\proofonname{##1}{##2}}}}%
390  }%
391 }
```

7.5 Thmheading definition-like environments

7.5.1 Manually define a thmheading environment

Define the key-value options accepted by \phfMakeThmheadingEnvironment.

```
392 \define@cmdkey{phfthmmkthmheading}{thmstyle}{}
393 \define@cmdkey{phfthmmkthmheading}{internalcounter}{}
```

```
394 \newcounter{phfthmheadingcounter}%
```

\phfMakeThmheadingEnvironment Creates a new environment \begin{thmheading}{Title}...\end{thmheading} for customizing the heading on-the-fly (see documentation in section 6). Useful for an alternative formatting of definitions. The syntax is:

\phfMakeThmheadingEnvironment[⟨*key-value options*⟩]{⟨*environment name*⟩}

You can also use \label and \ref (the latter simply displays the given title).

```
395 \newcommand\phfMakeThmheadingEnvironment[2][]{% }
```

Parse the options. First set defaults, and then parse the input string.

```
396    \def\cmdKV@phfthmmkthmheading@thmstyle{plain}%
397    \def\cmdKV@phfthmmkthmheading@internalcounter{phfthmheadingcounter}%
398    \setkeys{phfthmmkthmheading}{#1}%
```

And now, produce the relevant definitions:

```
399    \csdef{phfthm@thmheading@#2@val@title}{$\langle$No Title Given$\rangle$}%
400    \theoremstyle{\cmdKV@phfthmmkthmheading@thmstyle}%
```

We use \newtheorem* to create an unnumbered theorem. The fixed title is just a single token, the macro which will be set to the relevant title at the last moment.

```
401    \newtheorem*{phfthm@internal@thmheading@#2}{%
402      \csname phfthm@thmheading@#2@val@title\endcsname}%
```

Define the actual environment.

```
403    \newenvironment{#2}[1]{%}
404      \csdef{phfthm@thmheading@#2@val@title}{##1}%
405      \letcs\thephfthmheadingcounter{phfthm@thmheading@#2@val@title}%
```

Relay call to the internal *AMS*-defined "theorem:"

```
406        \csname phfthm@internal@thmheading@#2\endcsname%
```

Pin down an anchor. The use of \hspace*{0pt} is explained at *http://tex. stackexchange.com/a/88493/32188* (see especially the first comment).

```
407        \hspace*{0pt}\refstepcounter{\cmdKV@phfthmmkthmheading@internalcounter}%
408        \csname phfthm@hook@thmheading@#2@start\endcsname{##1}%
```

Also, let's add some flexibility in the hspace:

```
409        \hskip 0em plus 0.5em minus 0em%
410        \ignorespaces%
411    }%
```

Now, the END part of the environment: just call the callback and close the internal AMS-defined theorem.

```
412    {%
413      \csname phfthm@hook@thmheading@#2@end\endcsname%
414      \csname endphfthm@internal@thmheading@#2\endcsname%
415    }%
```

Also define the relevant callbacks, which just relay their calls to the default callbacks.

```
416    \csdef{phfthm@hook@thmheading@#2@start}##1{%
417      \phfthm@hook@thmheading@start{##1}}%
418    \csdef{phfthm@hook@thmheading@#2@end}{\phfthm@hook@thmheading@end}%
419 }
```

Provide as well the obsolete command \phfthmMakeThmheadingEnvironment which was provided in earlier versions of this package:

```
420 \def\phfthmMakeThmheadingEnvironment{\phfMakeThmheadingEnvironment}
```

\phfthm@hook@thmheading@start Global callbacks which are called for all thmheading-type environments de-
  \phfthm@hook@thmheading@end fined with \phfMakeThmheadingEnvironment (unless their hooks have been changed in order for them not to call these global hooks).

```
421 \def\phfthm@hook@thmheading@start#1{}
422 \def\phfthm@hook@thmheading@end{}
```

7.6   Theorem sets

Here, we define the theorem sets proposed by the package for quick loading.

We first define the names. These are defined in any case regardless of whether we are loading a theorem set or of which theorem set we are loading.

```
423 \def\theoremname{Theorem}
424 \def\propositionname{Proposition}
425 \def\lemmaname{Lemma}
426 \def\corollaryname{Corollary}
427 \def\conjecturename{Conjecture}
428 \def\remarkname{Remark}
429 \def\definitionname{Definition}
430 \def\ideaname{Idea}
431 \def\questionname{Question}
432 \def\problemname{Problem}
```

As we define the theorem sets, remember the names in a comma-separated list which we can display in help text. The \phfthm@def@thmset replaces the \def command and expects the definitions to follow immediately.

```
433 \def\phfthm@def@thmset@optlist{}
434 \def\phfthm@def@thmset#1{%
435   \appto\phfthm@def@thmset@optlist{#1,}\csdef{phfthm@thmset@#1}}
```

\phfthm@def@thmset@mktheorem
\phfthm@def@thmset@mkdefn

In definitions of theorem sets, use these macros to define a new theorem-like environment (theorem, proposition, corollary, etc.) or definition-like environment (definition, remark). The macros \phfthm@val@mkthmoptarg@theorem and \phfthm@val@mkthmoptarg@defn are defined by \phfLoadThmSet.

```
436 \def\phfthm@def@thmset@mktheorem{%
437   \expandafter\phfMakeTheorem\phfthm@val@mkthmoptarg@theorem}
438 \def\phfthm@def@thmset@mkdefn{%
439   \expandafter\phfMakeTheorem\phfthm@val@mkthmoptarg@defn}
```

The default set (empty name, or name "empty") provides no theorem. (The first line uses \def directly so that we don't include an empty item in the list of available choices.)

```
440 \def\phfthm@thmset@{}
441 \phfthm@def@thmset{empty}{}
```

Theorem set simple:

```
442 \phfthm@def@thmset{simple}{
443   \phfthm@def@thmset@mktheorem{theorem}{\theoremname}
444   \phfthm@def@thmset@mktheorem{proposition}{\propositionname}
445   \phfthm@def@thmset@mktheorem{lemma}{\lemmaname}
446   \phfthm@def@thmset@mktheorem{corollary}{\corollaryname}
447   \phfthm@def@thmset@mkdefn{definition}{\definitionname}
448 }
```

39

Theorem set `default`:

```
449 \phfthm@def@thmset{default}{
450   \phfthm@def@thmset@mktheorem{theorem}{\theoremname}
451   \phfthm@def@thmset@mktheorem{proposition}{\propositionname}
452   \phfthm@def@thmset@mktheorem{lemma}{\lemmaname}
453   \phfthm@def@thmset@mktheorem{corollary}{\corollaryname}
454   \phfthm@def@thmset@mktheorem{conjecture}{\conjecturename}
455   \phfthm@def@thmset@mktheorem{remark}{\remarkname}
456   \phfthm@def@thmset@mkdefn{definition}{\definitionname}
457 }
```

Theorem set `shortnames`:

```
458 \phfthm@def@thmset{shortnames}{
459   \phfthm@def@thmset@mktheorem{thm}{\theoremname}
460   \phfthm@def@thmset@mktheorem{prop}{\propositionname}
461   \phfthm@def@thmset@mktheorem{lem}{\lemmaname}
462   \phfthm@def@thmset@mktheorem{cor}{\corollaryname}
463   \phfthm@def@thmset@mktheorem{conj}{\conjecturename}
464   \phfthm@def@thmset@mktheorem{rem}{\remarkname}
465   \phfthm@def@thmset@mkdefn{defn}{\definitionname}
466 }
```

Theorem set `rich`. Add definitions to the `default` set:

```
467 \phfthm@def@thmset{rich}{
468   \phfthm@thmset@default
469   \phfthm@def@thmset@mktheorem{idea}{\ideaname}
470   \phfthm@def@thmset@mktheorem{question}{\questionname}
471   \phfthm@def@thmset@mktheorem{problem}{\problemname}
472 }
```

`\phfLoadThmSet` The macro `\phfLoadThmSet` loads a theorem set. See documentation at sub-section 3.2.

#1 = options to `\phfMakeTheorem` for theorem-like environments

#2 = options to `\phfMakeTheorem` for definition-like environments

#3 = name of the theorem set to load

```
473 \newcommand\phfLoadThmSet[3]{%
474   \ifcsname phfthm@thmset@#3\endcsname%
475     \edef\phfthm@val@mkthmoptarg@theorem{#1}%
476     \edef\phfthm@val@mkthmoptarg@defn{#2}%
477     \csname phfthm@thmset@#3\endcsname%
478   \else%
479     \PackageWarning{phfthm}{Unknown theorem set: '#3'!}%
480   \fi%
481 }
```

For compatibility with my earlier versions of phfthm, also provide the obsolete \phfthmLoadThmSet:

```
482 \def\phfthmLoadThmSet{\phfLoadThmSet}
```

7.7   Package option handling

The machinery is in place, now define and parse the package options.

7.7.1   Declaring the package options

The package options all use the keyval parsing mechanism using the xkeyval package.

Recall when using \define@XXXkey that the optional argument after the second mandatory argument is the value which is assumed if the key is given with no explicit value; it is not the initial default value.

The resetstyle *package option*

An option to reset all options so that the package provides only stand-alone definitions and is not invasive (see section 2).

This option does not expect any argument (i.e., you should specify \usepackage[resetstyle,⟨*other options*⟩]{phfthm}, and not \usepackage[resetstyle=true,⟨*other options*⟩]{phfthm}).

```
483 \define@key{phfthmpkg}{resetstyle}[]{%
484   \KV@phfthmpkg@smallproofsfalse%
485   \KV@phfthmpkg@qedsymbolblacksquarefalse%
486   \KV@phfthmpkg@prooftitleitbffalse%
487   \KV@phfthmpkg@sepcountersfalse%
488   \KV@phfthmpkg@proofreffalse%
489   \if\relax\detokenize{#1}\relax\else%
490     \PackageError{phfthm}{'resetstyle' does not take any argument.}{You
491       specified the 'resetstyle' argument and provided a value to it
492       ('resetstyle=...'). However the 'resetstyle' option does not accept
493       any value argument.}
494   \fi%
495 }
```

Options for loading theorem sets

Define the various package options for the loading of predefined theorem sets (subsection 2.1).

The `sepcounters` option, off by default.

```
496 \define@boolkey{phfthmpkg}{sepcounters}[true]{}
497 \KV@phfthmpkg@sepcountersfalse
```

The `proofref` option. The proof-ref is off initially by default.

```
498 \newif\ifKV@phfthmpkg@proofref
499 \KV@phfthmpkg@proofreffalse
500 \def\cmdKV@phfthmpkg@proofref@style{}
```

Actually define the option itself. Here we do some customized parsing of the value of the `proofref=...` option, to treat the cases `proofref=` (empty argument) and `proofref=false` separately.

```
501 \define@key{phfthmpkg}{proofref}[]{%
502   \ifblank{#1}{%
```

If a blank argument provided, set some sensible defaults with proofref on:

```
503     \KV@phfthmpkg@proofreftrue%
504     \def\cmdKV@phfthmpkg@proofref@style{default}%
505   }{%
```

Otherwise, check to see if the value is `false`, in which case deactivate the proof-ref mechanism, or else, activate it and set the given style value as documented in subsection 2.1.

```
506     \ifstrequal{#1}{false}{%
507       \KV@phfthmpkg@proofreffalse%
508     }{%
509       \KV@phfthmpkg@proofreftrue%
510       \def\cmdKV@phfthmpkg@proofref@style{#1}%
511     }%
512   }%
513 }
```

The `thmset` option. We subtly construct the command `\define@choicekey{phfthmpkg}{thmset}[\val]{\phfthm@def@thmset@optlist}`, but with the last macro (option list) expanded.

```
514 \def\@tmpa{\define@choicekey{phfthmpkg}{thmset}[\val]}
515 \edef\@tmpb{{\phfthm@def@thmset@optlist}}
516 \expandafter\@tmpa\@tmpb{%
517   \xdef\cmdKV@phfthmpkg@thmset{\val}%
518 }
```

By default we should load the `default` set.

```
519 \def\cmdKV@phfthmpkg@thmset{default}
```

The options `theoremstyle` and `definitionstyle` set which theorem style to use for theorems and definitions, when loading the given thmset.

```
520 \define@cmdkey{phfthmpkg}{theoremstyle}{}
521 \def\cmdKV@phfthmpkg@theoremstyle{plain}
522 \define@cmdkey{phfthmpkg}{definitionstyle}{}
523 \def\cmdKV@phfthmpkg@definitionstyle{definition}
```

Proof environment options

Define the package options `proofenv`, `smallproofs`, `qedsymbolblacksquare`, and `prooftitleitbf` (subsection 2.2).

```
524 \define@boolkey{phfthmpkg}{proofenv}[true]{}
525 \define@boolkey{phfthmpkg}{smallproofs}[true]{}
526 \define@boolkey{phfthmpkg}{qedsymbolblacksquare}[true]{}
527 \define@boolkey{phfthmpkg}{prooftitleitbf}[true]{}
```

Set the initial default values for these options.

```
528 \KV@phfthmpkg@smallproofstrue
529 \KV@phfthmpkg@qedsymbolblacksquaretrue
530 \KV@phfthmpkg@proofenvtrue
531 \KV@phfthmpkg@prooftitleitbffalse
```

Options for a theorem-like heading environment

Define the `thmheading` and `thmheadingstyle` package options, documented in section 6.

```
532 \define@boolkey{phfthmpkg}{thmheading}[true]{}
533 \define@cmdkey{phfthmpkg}{thmheadingstyle}{}
```

The `thmheading` environment is provided by default; it's a stand-alone definition anyway. The style defaults to the `plain` style.

```
534 \KV@phfthmpkg@thmheadingtrue
535 \def\cmdKV@phfthmpkg@thmheadingstyle{plain}
```

7.7.2 Parsing the package options

The usual stuff (xkeyval-flavored).

```
536 \DeclareOptionX*{%
537   \PackageWarning{phfthm}{Invalid option: '\CurrentOption'}%
538 }
539 \ProcessOptionsX<phfthmpkg>
```

7.7.3 Execute package options-controlled actions

Loading a theorem set

First, we need to take into account the options which alter the way the theorem sets will be loaded (separate counters, proof-ref, etc.).

Take care of the proof-ref stuff. First, define the possible styles (note that these are not the same as the values to the `proofrefstyle` argument to the `\phfMakeTheorem` command).

```
540 \def\phfthm@val@mkthmopt@proofrefstyle{}
541 \ifKV@phfthmpkg@proofref
542   \def\phfthm@proofref@style@default{}
543   \def\phfthm@proofref@style@{}
```

Note that `proofref=always` and `proofref=onlyifveryfar` have a global effect, because they set `\phfProofrefPageBackTolerance` and `\phfProofrefPageAheadTolerance` (see documentation in subsection 2.1).

```
544   \def\phfthm@proofref@style@always{
545     \def\phfProofrefPageBackTolerance{-1}
546     \def\phfProofrefPageAheadTolerance{-1}
547   }
548   \def\phfthm@proofref@style@onlyifveryfar{
549     \def\phfProofrefPageBackTolerance{2}
550     \def\phfProofrefPageAheadTolerance{4}
551   }
552   \def\phfthm@proofref@style@margin{
553     \def\phfthm@val@mkthmopt@proofrefstyle{proofrefstyle=margin}
554   }
555   \def\phfthm@proofref@style@longref{
```

For `longref`: by setting `\proofonname` globally, this option can be combined with other styles. But then we also change the default style formatting to avoid ugly line breaks.

```
556     \def\proofonname##1##2{The proof of this \phfthm@autorefnameof{##1} can
557       be found on ##2.}
558     \def\phfthm@proofrefstyle@default@fmt##1##2{%
559       \par{\raggedleft\proofrefsize{(\proofonname{##1}{##2})}\par}%
560     }
561   }
562   \def\phfthm@proofref@style@off{
563     \def\phfthm@val@mkthmopt@proofrefstyle{proofref=false}
564   }
```

Now execute the given styles. Construct the command `\phfthm@internal@execattribs{phfthm@proofref@style@}`

`{ProofRef Style}{\cmdKV@phfthmpkg@proofref@style}`, but with the last macro expanded.

```
565   \def\x{%
566     \phfthm@internal@execattribs{phfthm@proofref@style@}{ProofRef Style}}
567   \expandafter\x\expandafter{\cmdKV@phfthmpkg@proofref@style}
568 \fi
```

Take care of counters. In any case, define a common counter, in case we use a common counter for all theorem types. (The counter is defined in any case, to avoid breaking other code which might use it if suddenly the user decides to use `sepcounters=true` for their document.)

```
569 \newcounter{phfthmcounter}
570 \setcounter{phfthmcounter}{0}
```

Prepare an argument to `\phfMakeTheorem` according to the `sepcounters` option.

```
571 \ifKV@phfthmpkg@sepcounters
572   \def\phfthm@val@mkthmopt@counteropts{}
573 \else
574   \def\phfthm@val@mkthmopt@counteropts{counter=phfthmcounter}
575 \fi
```

Finally, load the theorem set defined by the options. The first argument regroups the options for theorem environments (Theorem, Proposition, Corollary, …); the second argument regroups the options for definition environments (Definition); the third argument is the theorem set name itself.

```
576 \phfLoadThmSet%
577 {[\phfthm@val@mkthmopt@counteropts,\phfthm@val@mkthmopt@proofrefstyle,
578   thmstyle=\cmdKV@phfthmpkg@theoremstyle]}%
579 {[\phfthm@val@mkthmopt@counteropts,proofref=false,
580   thmstyle=\cmdKV@phfthmpkg@definitionstyle]}%
581 {\cmdKV@phfthmpkg@thmset}
```

(Note the absence of the proof-ref for definitions.)

Define the proof environment

If requested, define the `proof` environment (subsection 2.2). First, make sure we take into account the options `smallproofs` , `qedsymbolblacksquare` and `prooftitleitbf` .

```
582 \def\phfthm@pkgopterr@require@proofenv#1{%
583   \ifKV@phfthmpkg@proofenv\else%
584     \PackageError{phfthm}{Option '#1' depends on 'proofenv=true'}%
585   \fi
```

```
586 }
587 \ifKV@phfthmpkg@smallproofs
588   \phfthm@pkgopterr@require@proofenv{smallproofs}
589   \apptocmd\phfthm@hookproof@startcommon{%
590     \def\baselinestretch{1.2}\footnotesize}{}{%
591     Failed to change command \string\phfthm@hook@start@proof}
592 \fi
593 \ifKV@phfthmpkg@qedsymbolblacksquare
594   \phfthm@pkgopterr@require@proofenv{qedsymbolblacksquare}
595   \RequirePackage{amssymb}
596   \providecommand\filledsquare{\ensuremath{\blacksquare}}
597   \renewcommand\qedsymbol{\text{\tiny\ensuremath{\filledsquare}}}
598 \fi
599 \ifKV@phfthmpkg@prooftitleitbf
600   \phfthm@pkgopterr@require@proofenv{prooftitleitbf}
601   \def\phfthm@ProofTitleFmt#1{{\itshape\bfseries#1.}}
602 \fi
```

Go ahead and define the proof environment. Because we have already loaded amsthm, we need to override the existing proof environment.

```
603 \ifKV@phfthmpkg@proofenv
604   \phfMakeProofEnv[override=true]{proof}
605 \fi
```

Define the theorem-heading environment

Define the thmheading environment, if requested.

```
606 \ifKV@phfthmpkg@thmheading
607   \phfMakeThmheadingEnvironment%
608       [thmstyle=\cmdKV@phfthmpkg@thmheadingstyle]{thmheading}
609 \fi
```

Change History

v1.0

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.

48

49

50

51