

# Thmtools Users' Guide

2008–2014 Dr. Ulrich M. Schwarz – ulmi@absatzen.de\*  
2020– Yukai Chou

2023/02/17 v0.75

<https://github.com/muzimuzhi/thmtools>

## Abstract

The thmtools bundle is a collection of packages that is designed to provide an easier interface to theorems, and to facilitate some more advanced tasks.

If you are a first-time user and you don't think your requirements are out of the ordinary, browse the examples in [chapter 1](#). If you're here because the other packages you've tried so far just can't do what you want, take inspiration from [chapter 2](#). If you're a repeat customer, you're most likely to be interested in the reference section in [chapter 3](#).

## Contents

<b>1 Thmtools for the impatient</b>	<b>2</b>	3.5 Restatable – hints and caveats . . . . .	17
1.1 Elementary definitions . . . . .	2	<b>A Thmtools for the morbidly curious</b>	<b>18</b>
1.2 Frilly references . . . . .	4	A.1 Core functionality . . . . .	18
1.3 Styling theorems . . . . .	4	A.1.1 The main package . . . . .	18
1.3.1 Declaring new theoremstyles . .	6	A.1.2 Adding hooks to the relevant commands . . . . .	19
1.4 Repeating theorems . . . . .	7	A.1.3 The key-value interfaces . . . . .	22
1.5 Lists of theorems . . . . .	8	A.1.4 Lists of theorems . . . . .	32
1.6 Extended arguments to theorem environments . . . . .	9	A.1.5 Re-using environments . . . . .	35
<b>2 Thmtools for the extravagant</b>	<b>10</b>	A.1.6 Restrictions . . . . .	36
2.1 Understanding thmtools' extension mechanism . . . . .	10	A.1.7 Fixing <code>autoref</code> and friends . .	40
2.2 Case in point: the <code>shaded</code> key . . . . .	10	A.2 Glue code for different backends . . . . .	42
2.3 Case in point: the <code>thmbox</code> key . . . . .	12	A.2.1 <code>amsthm</code> . . . . .	42
2.4 Case in point: the <code>mdframed</code> key . . . . .	12	A.2.2 <code>beamer</code> . . . . .	44
2.5 How thmtools finds your extensions . . .	12	A.2.3 <code>ntheorem</code> . . . . .	45
<b>3 Thmtools for the completionist</b>	<b>14</b>	A.3 Generic tools . . . . .	47
3.1 Known keys to <code>\declaretheoremstyle</code>	14	A.3.1 A generalized argument parser .	47
3.2 Known keys to <code>\declaretheorem</code> . .	15	A.3.2 Different counters sharing the same register . . . . .	48
3.3 Known keys to in-document theorems .	16	A.3.3 Tracking occurrences: none, one or many . . . . .	49
3.4 Known keys to <code>\listoftheorems</code> . .	16		

---

\*who would like to thank the users for testing, encouragement, feature requests, and bug reports. In particular, Denis Bitouzé prompted further improvement when thmtools got stuck in a “good enough for me” slump.

# 1 Thmtools for the impatient

## How to use this document

This guide consists mostly of examples and their output, sometimes with a few additional remarks. Since theorems are defined in the preamble and used in the document, the snippets are two-fold:

```
% Preamble code looks like this.
\usepackage{amsthm}
\usepackage{thmtools}
\declaretheorem{theorem}

% Document code looks like this.
\begin{theorem}[Euclid]
  \label{thm:euclid}%
  For every prime  $p$ , there is a prime  $p' > p$ .
  In particular, the list of primes,
  \begin{equation}\label{eq:1}
    2, 3, 5, 7, \dots
  \end{equation}
  is infinite.
\end{theorem}
```

The result looks like this:

**Theorem 1** (Euclid). *For every prime  $p$ , there is a prime  $p' > p$ . In particular, the list of primes,*

$$2, 3, 5, 7, \dots \quad (1.1)$$

*is infinite.*

Note that in all cases, you will need a *backend* to provide the command `\newtheorem` with the usual behaviour. The  $\text{\TeX}$  kernel has a built-in backend which cannot do very much; the most common backends these days are the `amsthm` and `ntheorem` packages. Throughout this document, we'll use `amsthm`, and some of the features won't work with `ntheorem`.

## 1.1 Elementary definitions

As you have seen above, the new command to define theorems is `\declaretheorem`, which in its most basic form just takes the name of the environment. All other options can be set through a key-val interface:

```
\usepackage{amsthm}
\usepackage{thmtools}
\declaretheorem[numberwithin=section]{theoremS}

\begin{theoremS}[Euclid]
  For every prime  $p$ , there is a prime  $p' > p$ .
  In particular, there are infinitely many primes.
\end{theoremS}
```

**TheoremS 1.1.1** (Euclid). *For every prime  $p$ , there is a prime  $p' > p$ . In particular, there are infinitely many primes.*

Instead of `numberwithin=`, you can also use `parent=` and `within=`. They're all the same, use the one you find easiest to remember.

Note the example above looks somewhat bad: sometimes, the name of the environment, with the first letter uppercased, is not a good choice for the theorem's title.

```
\usepackage{amsthm}
\usepackage{thmtools}
\declaretheorem[name=\\"Ubung]{exercise}

\begin{exercise}
  Prove Euclid's Theorem.
\end{exercise}
```

**Übung 1.** *Prove Euclid's Theorem.*

To save you from having to look up the name of the key every time, you can also use `title=` and `heading=` instead of `name=`; they do exactly the same and hopefully one of these will be easy to remember for you.

Of course, you do not have to follow the abominable practice of numbering theorems, lemmas, etc., separately:

```
\usepackage{amsthm}
\usepackage{thmtools}
\declaretheorem[sibling=theorem]{lemma}
```

**Lemma 2.** *For every prime  $p$ , there is a prime  $p' > p$ . In particular, there are infinitely many primes.*

```
\begin{lemma}
  For every prime  $p$ , there is a prime  $p' > p$ .
  In particular, there are infinitely many primes.
\end{lemma}
```

Again, instead of `sibling=`, you can also use `numberlike=` and `sharecounter=`.

Some theorems have a fixed name and are not supposed to get a number. To this end, `amsthm` provides `\newtheorem*`, which is accessible through `thmtools`:

```
\usepackage{amsthm}
\usepackage{thmtools}
\declaretheorem[numbered=no,
  name=Euclid's Prime Theorem]{euclid}
```

**Euclid's Prime Theorem.** *For every prime  $p$ , there is a prime  $p' > p$ . In particular, there are infinitely many primes.*

```
\begin{euclid}
  For every prime  $p$ , there is a prime  $p' > p$ .
  In particular, there are infinitely many primes.
\end{euclid}
```

As a somewhat odd frill, you can turn off the number if there's only one instance of the kind in the document. This might happen when you split and join your papers into short conference versions and longer journal papers and tech reports. Note that this doesn't combine well with the `sibling` key: how do you count like somebody who suddenly doesn't count anymore? Also, it takes an extra  $\TeX$  run to settle.

```
\usepackage{amsthm}
\usepackage{thmtools}
\usepackage[unq]{unique}
\declaretheorem[numbered=unless unique]{singleton}
\declaretheorem[numbered=unless unique]{couple}
```

**Couple 1.** *Marc & Anne*

**Singleton.** *Me.*

**Couple 2.** *Buck & Britta*

```
\begin{couple}
  Marc \& Anne
\end{couple}
\begin{singleton}
  Me.
\end{singleton}
\begin{couple}
  Buck \& Britta
\end{couple}
```

(New: 2020/08/01) Actually, the mandatory argument of `\declaretheorem` accepts a list of environment names, so you can define similar theorems at once. Moreover, similar to `\setmainfont` from `fontspec` package, the key-value interface can be used both before and after the mandatory argument.

```
\declaretheorem[numberwithin=section]
  {theorem, definition}
\declaretheorem{lemma, proposition, corollary}[
  style=plain,
  numberwithin=theorem
]
```

## 1.2 Frilly references

In case you didn't know, you should: `hyperref`, `nameref` and `cleveref` offer ways of “automagically” knowing that `\label{foo}` was inside a theorem, so that a reference adds the string “Theorem”. This is all done for you, but there's one catch: you have to tell `thmtools` what the name to add is. By default, it will use the title of the theorem, in particular, it will be uppercased. (This happens to match the guidelines of all publishers I have encountered.) But there is an alternate spelling available, denoted by a capital letter, and in any case, if you use `cleveref`, you should give two values separated by a comma, because it will generate plural forms if you reference many theorems in one `\cite`.

```
\usepackage{amsthm, thmtools}
\usepackage{
  hyperref,%\autoref
  % n.b. \Autoref is defined by thmtools
  cleveref,% \cref
  % n.b. cleveref after! hyperref
}
\declaretheorem[name=Theorem,
  refname={theorem,theorems},
  Refname={Theorem,Theorems}]{callmeal}
```

```
\begin{callmeal}[Simon]\label{simon}
  One
\end{callmeal}
\begin{callmeal}\label{garfunkel}
  and another, and together,
  \autoref{simon}, “\nameref{simon}”,
  and \cref{garfunkel} are referred
  to as \cref{simon,garfunkel}.
  \Cref{simon,garfunkel}, if you are at
  the beginning of a sentence.
\end{callmeal}
```

**Theorem 1** (Simon). *One*

**Theorem 2.** *and another, and together, theorem 1, “Simon”, and theorem 2 are referred to as theorems 1 and 2. Theorems 1 and 2, if you are at the beginning of a sentence.*

## 1.3 Styling theorems

The major backends provide a command `\theoremstyle` to switch between looks of theorems. This is handled as follows:

```
\usepackage{amsthm}
\usepackage{thmtools}
\declaretheorem[style=remark]{remark}
\declaretheorem{Theorem}
```

```
\begin{Theorem}
  Note how it still retains the default style,
  ‘plain’.
\end{Theorem}
\begin{remark}
  This is a remark.
\end{remark}
```

**Theorem 1.** *Note how it still retains the default style, ‘plain’.*

*Remark 1.* This is a remark.

Thmtools also supports the shadethm and thmbox packages:

```
\usepackage{amsthm}
\usepackage{thmtools}
\usepackage[dvipsnames]{xcolor}
\declaretheorem[shaded={bgcolor=Lavender,
textwidth=12em}]{BoxI}
\declaretheorem[shaded={rulecolor=Lavender,
rulewidth=2pt, bgcolor={rgb}{1,1,1}}]{BoxII}

\begin{BoxI}[Euclid]
  For every prime  $p$ , there is a prime  $p' > p$ .
  In particular, there are infinitely many primes.
\end{BoxI}
\begin{BoxII}[Euclid]
  For every prime  $p$ , there is a prime  $p' > p$ .
  In particular, there are infinitely many primes.
\end{BoxII}
```

**BoxI 1.** *For every prime  $p$ , there is a prime  $p' > p$ . In particular, there are infinitely many primes.*

**BoxII 1.** *For every prime  $p$ , there is a prime  $p' > p$ . In particular, there are infinitely many primes.*

As you can see, the color parameters can take two forms: it's either the name of a color that is already defined, without curly braces, or it can start with a curly brace, in which case it is assumed that `\definecolor{colorname}{what you said}` will be valid  $\TeX$  code. In our case, we use the `rgb` model to manually specify white. (shadethm's default background color is `\color{gray}{0.92}`)

For the thmbox package, use the thmbox key:

```
\usepackage{amsthm}
\usepackage{thmtools}
\declaretheorem[thmbox=L]{boxtheorem L}
\declaretheorem[thmbox=M]{boxtheorem M}
\declaretheorem[thmbox=S]{boxtheorem S}

\begin{boxtheorem L}[Euclid]
  For every prime  $p$ , there is a prime  $p' > p$ .
  In particular, there are infinitely many primes.
\end{boxtheorem L}
\begin{boxtheorem M}[Euclid]
  For every prime  $p$ , there is a prime  $p' > p$ .
  In particular, there are infinitely many primes.
\end{boxtheorem M}
\begin{boxtheorem S}[Euclid]
  For every prime  $p$ , there is a prime  $p' > p$ .
  In particular, there are infinitely many primes.
\end{boxtheorem S}
```

**Boxtheorem L 1 (Euclid)**

*For every prime  $p$ , there is a prime  $p' > p$ . In particular, there are infinitely many primes.*

**Boxtheorem M 1 (Euclid)**

*For every prime  $p$ , there is a prime  $p' > p$ . In particular, there are infinitely many primes.*

**Boxtheorem S 1 (Euclid)**

*For every prime  $p$ , there is a prime  $p' > p$ . In particular, there are infinitely many primes.*

Note that for both thmbox and shaded keys, it's quite possible they will not cooperate with a style key you give at the same time.

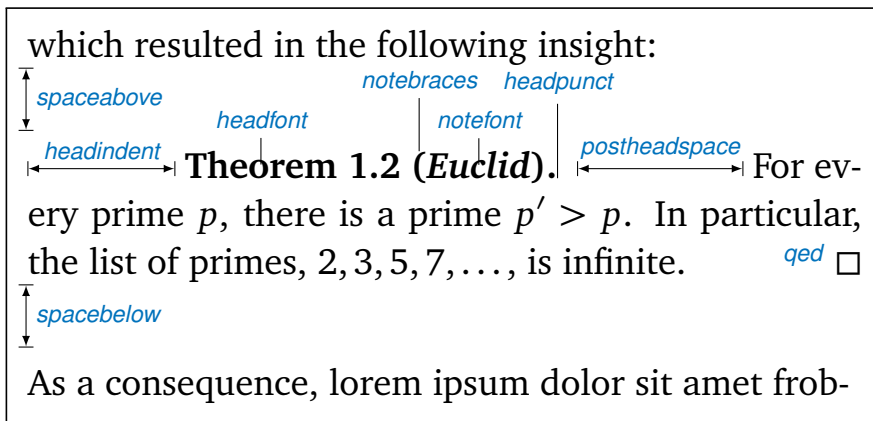


Figure 1.1: Settable parameters of a theorem style.

### 1.3.1 Declaring new theoremstyles

Thmtools also offers a new command to define new theoremstyles. It is partly a frontend to the `\newtheoremstyle` command of `amsthm` or `ntheorem`, but it offers (more or less successfully) the settings of both to either. So we are talking about the same things, consider the sketch in [Figure 1.1](#). To get a result like that, you would use something like

```
\declaretheoremstyle[
  spaceabove=6pt, spacebelow=6pt,
  headfont=\normalfont\bfseries,
  notefont=\mdseries, notebraces={({})},
  bodyfont=\normalfont,
  postheadspace=1em,
  qed=\qedsymbol
]{mystyle}
\declaretheorem[style=mystyle]{styledtheorem}

\begin{styledtheorem}[Euclid]
  For every prime  $p$ \dots
\end{styledtheorem}
```

**Styledtheorem 1 (Euclid).** For every prime  $p$ ... □

Again, the defaults are reasonable and you don't have to give values for everything.

There is one important thing you cannot see in this example: there are more keys you can pass to `\declaretheoremstyle`: if `thmtools` cannot figure out at all what to do with it, it will pass it on to the `\declaretheorem` commands that use that style. For example, you may use the `boxed` and `shaded` keys here.

To change the order in which title, number and note appear, there is a key `headformat`. Currently, the values "margin" and "swapnumber" are supported. The daring may also try to give a macro here that uses the commands `\NUMBER`, `\NAME` and `\NOTE`. You cannot circumvent the fact that `headpunct` comes at the end, though, nor the fonts and braces you select with the other keys.

## 1.4 Repeating theorems

Sometimes, you want to repeat a theorem you have given in full earlier, for example you either want to state your strong result in the introduction and then again in the full text, or you want to re-state a lemma in the appendix where you prove it. For example, I lied about [Theorem 1](#) on p. 2: the true code used was

```
\usepackage{thmtools, thm-restate}
\declaretheorem{theorem}
```

```
\begin{restatable}[Euclid]{theorem}{firsteuclid}
  \label{thm:euclid}%
  For every prime  $p$ , there is a prime  $p' > p$ .
  In particular, the list of primes,
  \begin{equation}\label{eq:1}
    2, 3, 5, 7, \dots
  \end{equation}
  is infinite.
\end{restatable}
```

and to the right, I just use

```
\firsteuclid*
\vdots
\firsteuclid*
```

**Theorem 1** (Euclid). *For every prime  $p$ , there is a prime  $p' > p$ . In particular, the list of primes,*

$$2, 3, 5, 7, \dots \quad (1.1)$$

*is infinite.*

⋮

**Theorem 1** (Euclid). *For every prime  $p$ , there is a prime  $p' > p$ . In particular, the list of primes,*

$$2, 3, 5, 7, \dots \quad (1.1)$$

*is infinite.*

Note that in spite of being a theorem-environment, it gets number one all over again. Also, we get equation number (1.1) again. The star in `\firsteuclid*` tells thmtools that it should redirect the label mechanism, so that this reference: [Theorem 1](#) points to p. 2, where the unstarred environment is used. (You can also use a starred environment and an unstarred command, in which case the behaviour is reversed.) Also, if you use hyperref (like you see in this manual), the links will lead you to the unstarred occurrence.

Just to demonstrate that we also handle more involved cases, I repeat another theorem here, but this one was numbered within its section: note we retain the section number which does not fit the current section:

```
\euclidii*
```

**TheoremS 1.1.1** (Euclid). *For every prime  $p$ , there is a prime  $p' > p$ . In particular, there are infinitely many primes.*

## 1.5 Lists of theorems

To get a list of theorems with default formatting, just use `\listoftheorems`:

<code>\listoftheorems</code>	<b>List of Theorems</b>
	1 Theorem (Euclid) . . . . . 2
	1.1.1 TheoremS (Euclid) . . . . . 2
	1 Übung . . . . . 2
	2 Lemma . . . . . 3
	Euclid's Prime Theorem . . . 3
	1 Couple . . . . . 3
	Singleton . . . . . 3
	2 Couple . . . . . 3
	1 Theorem (Simon) . . . . . 4
	2 Theorem . . . . . 4
	1 Theorem . . . . . 4
	1 Remark . . . . . 4
	1 BoxI . . . . . 5
	1 BoxII . . . . . 5
	1 Boxtheorem L (Euclid) . . . 5
	1 Boxtheorem M (Euclid) . . . 5
	1 Boxtheorem S (Euclid) . . . 5
	1 Styledtheorem (Euclid) . . . 6
	1 Theorem (Euclid) . . . . . 7
	1 Theorem (Euclid) . . . . . 7
	1.1.1 TheoremS (Euclid) . . . . . 7
	3 Theorem (Keyed theorem) . . . 9
	3 Theorem (continuing from p. 9) . . . . . 9
	4 Lemma (Zorn) . . . . . 36
	5 Lemma . . . . . 36
	4 Lemma (Zorn) . . . . . 36

Not everything might be of the same importance, so you can filter out things by environment name:

<code>\listoftheorems[ignoreall, show={theorem,Theorem,euclid}]</code>	<b>List of Theorems</b>
	1 Theorem (Euclid) . . . . . 2
	Euclid's Prime Theorem . . . 3
	1 Theorem . . . . . 4
	1 Theorem (Euclid) . . . . . 7
	1 Theorem (Euclid) . . . . . 7
	3 Theorem (Keyed theorem) . . . 9
	3 Theorem (continuing from p. 9) . . . . . 9

And you can also restrict to those environments that have an optional argument given. Note that two theorems disappear compared to the previous example. You could also say just `onlynamed`, in which case it will apply to *all* theorem environments you have defined.



```
\listoftheorems[ignoreall,
  onlynamed={theorem,Theorem,euclid}]
```

## List of Theorems

1	Theorem (Euclid) . . . . .	2
1	Theorem (Euclid) . . . . .	7
1	Theorem (Euclid) . . . . .	7
3	Theorem (Keyed theorem)	9
3	Theorem (continuing from p. 9) . . . . .	9

As might be expected, the heading given is defined in `\listtheoremname`.

### 1.6 Extended arguments to theorem environments

Usually, the optional argument of a theorem serves just to give a note that is shown in the theorem's head. Thmtools allows you to have a key-value list here as well. The following keys are known right now:

**name** This is what used to be the old argument. It usually holds the name of the theorem, or a source. This key also accepts an *optional* argument, which will go into the list of theorems. Be aware that since we already are within an optional argument, you have to use an extra level of curly braces: `\begin{theorem}[name={ [Short name]A long name, ...}]`

**label** This will issue a `\label` command after the head. Not very useful, more of a demo.

**continues** Saying `continues=foo` will cause the number that is given to be changed to `\ref{foo}`, and a text is added to the note. (The exact text is given by the macro `\thmcontinues`, which takes the label as its argument.)

**restate** Saying `restate=foo` will hopefully work like wrapping this theorem in a restatable environment. (It probably still fails in cases that I didn't think of.) This key also accepts an optional argument: when restating, the `restate` key is replaced by this argument, for example, `restate=[name=Boring rehash]foo` will result in a different name. (Be aware that it is possible to give the same key several times, but I don't promise the results. In case of the name key, the names happen to override one another.)

```
\begin{theorem}[name=Keyed theorem,
  label=thm:key]
  This is a
  key-val theorem.
\end{theorem}
\begin{theorem}[continues=thm:key]
  And it's spread out.
\end{theorem}
```

**Theorem 3** (Keyed theorem). *This is a key-val theorem.*

**Theorem 3** ([continuing](#) from p. 9). *And it's spread out.*

## 2 Thmtools for the extravagant

This chapter will go into detail on the slightly more technical offerings of this bundle. In particular, it will demonstrate how to use the general hooks provided to extend theorems in the way you want them to behave. Again, this is done mostly by some examples.

### 2.1 Understanding thmtools' extension mechanism

Thmtools draws most of its power really only from one feature: the `\newtheorem` of the backend will, for example, create a theorem environment, i.e. the commands `\theorem` and `\endtheorem`. To add functionality, four places immediately suggest themselves: “immediately before” and “immediately after” those two.

There are two equivalent ways of adding code there: one is to call `\addtotheoremheadhook` and its brothers and sisters `...postheadhook`, `...prefoothook` and `...postfoothook`. All of these take an *optional* argument, the name of the environment, and the new code as a mandatory argument. The name of environment is optional because there is also a set of “generic” hooks added to every theorem that you define.

The other way is to use the keys `preheadhook` et al. in your `\declaretheorem`. (There is no way of accessing the generic hook in this way.)

The hooks are arranged in the following way: first the specific prehead, then the generic one. Then, the original `\theorem` (or whatever) will be called. Afterwards, first the specific posthead again, then the generic one. (This means that you cannot wrap the head alone in an environment this way.) At the end of the theorem, it is the other way around: first the generic, then the specific, both before and after that `\endtheorem`. This means you can wrap the entire theorem easily by adding to the prehead and the postfoot hooks. Note that thmtools does not look inside `\theorem`, so you cannot get inside the head formatting, spacing, punctuation in this way.

In many situations, adding static code will not be enough. Your code can look at `\thmt@envname`, `\thmt@thmname` and `\thmt@optarg`, which will contain the name of the environment, its title, and, if present, the optional argument (otherwise, it is `\@empty`). *However*, you should not make assumptions about the optional argument in the preheadhook: it might still be key-value, or it might already be what will be placed as a note. (This is because the key-val handling itself is added as part of the headkeys.)

### 2.2 Case in point: the shaded key

Let us look at a reasonably simple example: the `shaded` key, which we've already seen in the first section. You'll observe that we run into a problem similar to the four-hook mess: your code may either want to modify parameters that need to be set beforehand, or it wants to modify the environment after it has been created. To hide this from the user, the code you define for the key is actually executed twice, and `\thmt@trytwice{A}{B}` will execute A on the first pass, and B on the second. Here, we want to add to the hooks, and the hooks are only there in the second pass.

Mostly, this key wraps the theorem in a `shadebox` environment. The parameters are set by treating the value we are given as a new key-val list, see below.

```
1 \define@key{thmdef}{shaded}[]{}{%
2 \thmt@trytwice{}{%
3   \RequirePackage{shadethm}%
4   \RequirePackage{thm-patch}%
5   \addtotheoremheadhook[\thmt@envname]{%
6     \setlength\shadedtextwidth{\linewidth}%
7     \kvsetkeys{thmt@shade}{#1}\begin{shadebox}}%
8   \addtotheorempostfoothook[\thmt@envname]{\end{shadebox}}%
9   }%
10 }
```

The docs for shadethm say:

There are some parameters you could set the default for (try them as is, first).

- `shadethmcolor` The shading color of the background. See the documentation for the color package, but with a ‘gray’ model, I find .97 looks good out of my printer, while a darker shade like .92 is needed to make it copy well. (Black is 0, white is 1.)
- `shaderulecolor` The shading color of the border of the shaded box. See (i). If `shadeboxrule` is set to 0pt then this won’t print anyway.
- `shadeboxrule` The width of the border around the shading. Set it to 0pt (not just 0) to make it disappear.
- `shadeboxsep` The length by which the shade box surrounds the text.

So, let’s just define keys for all of these.

```
11 \define@key{thmt@shade}{textwidth} {\setlength\shadedtextwidth{#1}}
12 \define@key{thmt@shade}{bgcolor} {\thmt@definecolor{shadethmcolor}{#1}}
13 \define@key{thmt@shade}{rulecolor} {\thmt@definecolor{shaderulecolor}{#1}}
14 \define@key{thmt@shade}{rulewidth} {\setlength\shadeboxrule{#1}}
15 \define@key{thmt@shade}{margin} {\setlength\shadeboxsep{#1}}
16 \define@key{thmt@shade}{padding} {\setlength\shadeboxsep{#1}}
17 \define@key{thmt@shade}{leftmargin} {\setlength\shadeleftshift{#1}}
18 \define@key{thmt@shade}{rightmargin} {\setlength\shaderrightshift{#1}}
```

What follows is wizardry you don’t have to understand. In essence, we want to support two notions of color: one is “everything that goes after `\definecolor{shadethmcolor}`”, such as `{rgb}{0.8,0.85,1}`. On the other hand, we’d also like to recognize an already defined color name such as `blue`.

To handle the latter case, we need to copy the definition of one color into another. The `xcolor` package offers `\colorlet` for that, for the color package, we just cross our fingers.

```
19 \def\thmt@colorlet#1#2{%
20   %\typeout{don't know how to let color '#1' be like color '#2'!}%
21   \@xa\let\csname\string\color@#1\@xa\endcsname
22   \csname\string\color@#2\endcsname
23   % this is dubious at best, we don't know what a backend does.
24 }
25 \AtBeginDocument{%
26   \ifcsname colorlet\endcsname
27     \let\thmt@colorlet\colorlet
28   \fi
29 }
```

Now comes the interesting part: we assume that a simple color name must not be in braces, and a color definition starts with an opening curly brace. (So, if `\definecolor` ever gets an optional arg, we are in a world of pain.)

If the second argument to `\thmt@definecolor` (the key) starts with a brace, then `\thmt@def@color` will have an empty second argument, delimited by the brace of the key. Hopefully, the key will have exactly enough arguments to satisfy `\definecolor`. Then, `\thmt@drop@relax` will be executed and gobble the fallback values and the `\thmt@colorlet`.

If the key does not contain an opening brace, `\thmt@def@color` will drop everything up to `{gray}{0.5}`. So, first the color gets defined to a medium gray, but then, it immediately gets overwritten with the definition corresponding to the color name.

```
30 \def\thmt@drop@relax#1\relax{}
31 \def\thmt@definecolor#1#2{%
32   \thmt@def@color{#1}#2\thmt@drop@relax
33   {gray}{0.5}%
34   \thmt@colorlet{#1}{#2}%
35   \relax
36 }
37 \def\thmt@def@color#1#2#{%
38   \definecolor{#1}}
```

## 2.3 Case in point: the `thmbox` key

The `thmbox` package does something else: instead of having a separate environment, we have to use a command different from `\newtheorem` to get the boxed style. Fortunately, `thmtools` stores the command as `\thmt@theoremdefiner`, so we can modify it. (One of the perks if extension writer and framework writer are the same person.) So, in contrast to the previous example, this time we need to do something before the actual `\newtheorem` is called.

```
39 \define@key{thmdef}{thmbox}[L]{%
40   \thmt@trytwice{%
41     \let\oldproof=\proof
42     % backup \proof, gh32
43     \expandafter\let\csname old\@backslashchar proof\expandafter\endcsname
44       \csname \@backslashchar proof\endcsname
45     \let\oldendproof=\endproof
46     \let\oldexample=\example
47     \let\oldendexample=\endexample
48     \RequirePackage[nothm]{thmbox}
49     \let\proof=\oldproof
50     % restore thmbox's change to \proof, gh32
51     \expandafter\let\csname \@backslashchar proof\expandafter\endcsname
52       \csname old\@backslashchar proof\endcsname
53     \let\endproof=\oldendproof
54     \let\example=\oldexample
55     \let\endexample=\oldendexample
56     \def\thmt@theoremdefiner{\newboxtheorem[#1]}%
57   }{}%
58 }
```

## 2.4 Case in point: the `mdframed` key

Mostly, this key wraps the theorem in a `mdframed` environment. The parameters are set by treating the value we are given as a new key-val list, see below.

```
59 \define@key{thmdef}{mdframed}[{}]{%
60   \thmt@trytwice{}{}%
61   \RequirePackage{mdframed}%
62   \RequirePackage{thm-patch}%
63   \addtotheorempreheadhook[\thmt@envname]{\begin{mdframed}[#1]}%
64   \addtotheorempostfoothook[\thmt@envname]{\end{mdframed}}%
65 }{}%
66 }
```

## 2.5 How `thmtools` finds your extensions

Up to now, we have discussed how to write the code that adds functionality to your theorems, but you don't know how to activate it yet. Of course, you can put it in your preamble, likely embraced by `\makeatletter` and `\makeatother`, because you are using internal macros with `@` in their name (viz., `\thmt@envname` and friends). You can also put them into a package (then, without the `\makeat...`), which is simply a file ending in `.sty` put somewhere that  $\TeX$  can find it, which can then be loaded with `\usepackage`. To find out where exactly that is, and if you'd need to update administrative helper files such as a filename database FNDB, please consult the documentation of your  $\TeX$  distribution.

Since you most likely want to add keys as well, there is a shortcut that `thmtools` offers you: whenever you use a key `key` in a `\declaretheorem` command, and `thmtools` doesn't already know what to do with it, it will try to `\usepackage{thmdef-key}` and evaluate the key again. (If that doesn't work, `thmtools` will cry bitterly.)

For example, there is no provision in `thmtools` itself that make the `shaded` and `thmbox` keys described above special: in fact, if you want to use a different package to create frames, you just put a different

`thmdef-shaded.sty` into a preferred `texmf` tree. Of course, if your new package doesn't offer the old keys, your old documents might break!

The behaviour for the keys in the style definition is slightly different: if a key is not known there, it will be used as a "default key" to every theorem that is defined using this style. For example, you can give the `shaded` key in a style definition.

Lastly, the `key-val` arguments to the theorem environments themselves need to be loaded manually, not least because inside the document it's too late to call `\usepackage`.

## 3 Thmtools for the completionist

This will eventually contain a reference to all known keys, commands, etc.

### 3.1 Known keys to `\declaretheoremstyle`

N.b. implementation for `amsthm` and `ntheorem` is separate for these, so if it doesn't work for `ntheorem`, try if it works with `amsthm`, which in general supports more things.

Also, all keys listed as known to `\declaretheorem` are valid.

**spaceabove** Value: a length. Vertical space above the theorem, possibly discarded if the theorem is at the top of the page.

**spacebelow** Value: a length. Vertical space after the theorem, possibly discarded if the theorem is at the top of the page.

**headfont** Value:  $\TeX$  code. Executed just before the head of the theorem is typeset, inside a group. Intended use it to put font switches here.

**notefont** Value:  $\TeX$  code. Executed just before the note in the head is typeset, inside a group. Intended use it to put font switches here. Formatting also applies to the braces around the note. Not supported by `ntheorem`.

**bodyfont** Value:  $\TeX$  code. Executed before the begin part of the theorem ends, but before all afterhead-hooks. Intended use it to put font switches here.

**headpunct** Value:  $\TeX$  code, usually a single character. Put at the end of the theorem's head, prior to linebreaks or indents.

**notebraces** Value: Two characters, the opening and closing symbol to use around a theorem's note. (Not supported by `ntheorem`.)

**postheadspace** Value: a length. Horizontal space inserted after the entire head of the theorem, before the body. Does probably not apply (or make sense) for styles that have a linebreak after the head.

**headformat** Value:  $\LaTeX$  code using the special placeholders `\NUMBER`, `\NAME` and `\NOTE`, which correspond to the (formatted, including the braces for `\NOTE` etc.) three parts of a theorem's head. This can be used to override the usual style "1.1 Theorem (Foo)", for example to let the numbers protrude in the margin or put them after the name.

Additionally, a number of keywords are allowed here instead of  $\LaTeX$  code:

**margin** Lets the number protrude in the (left) margin.

**swapnumber** Puts the number before the name. Currently not working so well for unnumbered theorems.

*This list is likely to grow*

**headindent** Value: a length. Horizontal space inserted before the head. Some publishers like `\parindent` here for remarks, for example.

### 3.2 Known keys to `\declaretheorem`

**parent** Value: a counter name. The theorem will be reset whenever that counter is incremented. Usually, this will be a sectioning level, chapter or section.

**numberwithin** (Same as parent.)

**within** (Same as parent.)

**sibling** Value: a counter name. The theorem will use this counter for numbering. Usually, this is the name of another theorem environment.

**numberlike** (Same as sibling.)

**sharenumber** (Same as sibling.)

**title** Value:  $\TeX$  code. The title of the theorem. Default is the name of the environment, with `\MakeUppercase` prepended. You'll have to give this if your title starts with an accented character, for example.

**name** (Same as title.)

**heading** (Same as title.)

**numbered** Value: one of the keywords `yes`, `no` or `unless unique`. The theorem will be numbered, not numbered, or only numbered if it occurs more than once in the document. (The latter requires another  $\TeX$  run and works well combined with `sibling`.)

**style** Value: the name of a style defined with `\declaretheoremstyle` or `\newtheoremstyle`. The theorem will use the settings of this style.

**preheadhook** Value:  $\TeX$  code. This code will be executed at the beginning of the environment, even before vertical spacing is added and the head is typeset. However, it is already within the group defined by the environment.

**postheadhook** Value:  $\TeX$  code. This code will be executed after the call to the original `begin-theorem` code. Note that all backends seem to delay typesetting the actual head, so code here should probably enter horizontal mode to be sure it is after the head, but this will change the spacing/wrapping behaviour if your body starts with another list.

**prefoothook** Value:  $\TeX$  code. This code will be executed at the end of the body of the environment.

**postfoothook** Value:  $\TeX$  code. This code will be executed at the end of the environment, even after eventual vertical spacing, but still within the group defined by the environment.

**refname** Value: one string, or two strings separated by a comma (no spaces). This is the name of the theorem as used by `\autoref`, `\cref` and friends. If it is two strings, the second is the plural form used by `\cref`. Default value is the value of `name`, i.e. usually the environment name, with `\MakeUppercase` prepended.

**Refname** Value: one string, or two strings separated by a comma (no spaces). This is the name of the theorem as used by `\Autoref`, `\Cref` and friends. If it is two strings, the second is the plural form used by `\Cref`. This can be used for alternate spellings, for example if your style requests no abbreviations at the beginning of a sentence. No default.

**shaded** Value: a key-value list, where the following keys are possible:

**textwidth** The linewidth within the theorem.

**bgcolor** The color of the background of the theorem. Either a color name or a color spec as accepted by `\definecolor`, such as `{gray}{0.5}`.

**rulecolor** The color of the box surrounding the theorem. Either a color name or a color spec.

**rulewidth** The width of the box surrounding the theorem.

**margin** The length by which the shade box surrounds the text.

**thmbox** Value: one of the characters L, M and S; see examples in [section 1.3](#).

### 3.3 Known keys to in-document theorems

**label** Value: a legal `\label` name. Issues a `\label` command after the theorem's head.

**name** Value:  $\TeX$  code that will be typeset. What you would have put in the optional argument in the non-keyval style, i.e. the note to the head. This is *not* the same as the `name` key to `\declaretheorem`, you cannot override that from within the document.

**listhack** Value: doesn't matter. (But put something to trigger key-val behaviour, maybe `listhack=true`.) Linebreak styles in `amsthm` don't linebreak if they start with another list, like an `enumerate` environment. Giving the `listhack` key fixes that. *Don't* give this key for non-break styles, you'll get too little vertical space! (Just use `\leavevmode` manually there.) An all-around `listhack` that handles both situations might come in a cleaner rewrite of the style system.

### 3.4 Known keys to `\listoftheorems`

**title** Value: title of `\listoftheorems`. Initially `List of Theorems`.

**ignore** Value: list of theorem environment names. Filter out things by environment names. Default value is list of all defined theorem environments.

**ignoreall** Ignore every theorem environment. This key is usually followed by keys `show` and `onlynamed`.

**show** Value: list of theorem environments. Leave theorems that belong to specified list and filter out others. Default value is list of all defined theorem environments.

**showall** The opposite effect of `ignoreall`.

**onlynamed** Value: list of theorem environments. Leave things that are given an optional argument and belong to specified list, and filter out others. Default value is list of all defined theorem environments.



**swapnumber** Value: true or false. Initially false and default value is true. No default.

```
\listoftheorems[ignoreall, onlynamed={lemma}]  
\listoftheorems[ignoreall, onlynamed={lemma},  
  swapnumber  
]
```

### List of Theorems

4	Lemma (Zorn) . . . . .	36
4	Lemma (Zorn) . . . . .	36

### List of Theorems

	Lemma 4 (Zorn) . . . . .	36
	Lemma 4 (Zorn) . . . . .	36

**numwidth** Value: a length. If `swapnumber=false`, the theorem number is typeset in a box of width `numwidth`. Initially 1.5pc for AMS classes and 2.3em for others.

## 3.5 Restatable – hints and caveats

TBD.

- Some counters are saved so that the same values appear when you re-use them. The list of these counters is stored in the macro `\thmt@innercounters` as a comma-separated list without spaces; default: `equation`.
- To preserve the influence of other counters (think: equation numbered per section and recall the theorem in another section), we need to know all macros that are used to turn a counter into printed output. Again, comma-separated list without spaces, without leading backslash, stored as `\thmt@counterformatters`. Default: `@alph,@Alph,@arabic,@roman,@Roman,@fnsymbol`. All these only take the  $\TeX$  counter `\c@foo` as arguments. If you bypass this and use `\romannumeral`, your numbers go wrong and you get what you deserve. Important if you have very strange numbering, maybe using greek letters or `somesuch`.
- I think you cannot have one stored counter within another one's typeset representation. I don't think that ever occurs in reasonable circumstances, either. Only one I could think of: multiple subequation blocks that partially overlap the theorem. Dude, that doesn't even nest. You get what you deserve.
- `\label` and `amsmath's \ltx@label` are disabled inside the starred execution. Possibly, `\phantomsection` should be disabled as well?

## A Thmtools for the morbidly curious

This chapter consists of the implementation of thmtools, in case you wonder how this or that feature was implemented. Read on if you want a look under the bonnet, but you enter at your own risk, and bring an oily rag with you.

### A.1 Core functionality

#### A.1.1 The main package

```
67 \DeclareOption{debug}{%
68   \def\thmt@debug{\typeout}%
69 }
70 % common abbreviations and marker macros.
71 \let\@xa\expandafter
72 \let\@nx\noexpand
73 \def\thmt@debug{\@gobble}
74 \def\thmt@quark{\thmt@quark}
75 \newtoks\thmt@toks
76
77 \@for\thmt@opt:=lowercase,uppercase,anycase\do{%
78   \@xa\DeclareOption\@xa{\thmt@opt}{%
79     \@xa\PassOptionsToPackage\@xa{\CurrentOption}{thm-kv}%
80   }%
81 }
82
83 \ProcessOptions\relax
84
85 % a scratch counter, mostly for fake hyperlinks
86 \newcounter{thmt@dummyctr}%
87 \def\theHthmt@dummyctr{dummy.\arabic{thmt@dummyctr}}%
88 \def\thethmt@dummyctr{}%
89
90
91 \RequirePackage{thm-patch, thm-kv,
92   thm-autoref, thm-listof,
93   thm-restate}
94
95 % Glue code for the big players.
96 \@ifpackageloaded{amsthm}{%
97   \RequirePackage{thm-amsthm}
98 }{%
99   \AtBeginDocument{%
100     \@ifpackageloaded{amsthm}{%
101       \PackageWarningNoLine{thmtools}{%
102         amsthm loaded after thmtools
103       }{}%
104     }{}%
105   }
106 \@ifpackageloaded{ntheorem}{%
107   \RequirePackage{thm-ntheorem}
108 }{%
109   \AtBeginDocument{%
110     \@ifpackageloaded{ntheorem}{%
111       \PackageWarningNoLine{thmtools}{%
112         ntheorem loaded after thmtools
```

```

113   }{}%
114  }{}%
115 }
116 \@ifclassloaded{beamer}{%
117   \RequirePackage{thm-beamer}
118 }{}
119 \@ifclassloaded{llncs}{%
120   \RequirePackage{thm-llncs}
121 }{}

```

### A.1.2 Adding hooks to the relevant commands

This package is maybe not very suitable for the end user. It redefines `\newtheorem` in a way that lets other packages (or the user) add code to the newly-defined theorems, in a reasonably cross-compatible (with the kernel, theorem and amsthm) way.

**Warning:** the new `\newtheorem` is a superset of the allowed syntax. For example, you can give a star and both optional arguments, even though you cannot have an unnumbered theorem that shares a counter and yet has a different reset-regimen. At some point, your command is re-assembled and passed on to the original `\newtheorem`. This might complain, or give you the usual “Missing `\begin{document}`” that marks too many arguments in the preamble.

A call to `\addtotheoremheadhook[kind]{code}` will insert the code to be executed whenever a *kind* theorem is opened, before the actual call takes place. (I.e., before the header “Kind 1.3 (Foo)” is typeset.) There are also posthooks that are executed after this header, and the same for the end of the environment, even though nothing interesting ever happens there. These are useful to put `\begin{shaded}... \end{shaded}` around your theorems. Note that footooks are executed LIFO (last addition first) and headhooks are executed FIFO (first addition first). There is a special kind called generic that is called for all theorems. This is the default if no kind is given.

The added code may examine `\thmt@thmname` to get the title, `\thmt@envname` to get the environment’s name, and `\thmt@optarg` to get the extra optional title, if any.

```

122 \RequirePackage{parseargs}
123
124 \newif\ifthmt@isstarred
125 \newif\ifthmt@hassibling
126 \newif\ifthmt@hasparent
127
128 \def\thmt@parsetheoremargs#1{%
129   \parse{%
130     {\parseOpt[]{\def\thmt@optarg{##1}}}{%
131       \let\thmt@shortoptarg\@empty
132       \let\thmt@optarg\@empty}}%
133   {%
134     \def\thmt@local@preheadhook{}%
135     \def\thmt@local@postheadhook{}%
136     \def\thmt@local@prefoothook{}%
137     \def\thmt@local@postfoothook{}%
138     \thmt@local@preheadhook
139     \csname thmt@#1@preheadhook\endcsname
140     \thmt@generic@preheadhook
141     % change following to \@xa-orgy at some point?
142     % forex, might have keyvals involving commands.
143     %\protected@edef\tmp@args{%
144     % \ifx\@empty\thmt@optarg else [{\thmt@optarg}]\fi
145     %}%
146     \ifx\@empty\thmt@optarg
147       \def\tmp@args{}%
148     \else
149       \@xa\def\@xa\tmp@args\@xa{\@xa[\@xa{\thmt@optarg}]}%
150     \fi
151     \csname thmt@original@#1\@xa\endcsname\tmp@args

```

```

152      %%moved down: \thmt@local@posttheadhook
153      %% (give posttheadhooks a chance to re-set nameref data)
154      \csname thmt@#1@posttheadhook\endcsname
155      \thmt@generic@posttheadhook
156      \thmt@local@posttheadhook
157%FMi 2019-07-31
158 %      \let\@parsecmd\@empty
159 %      \let\@parsecmd\ignorespaces
160%FMi ---
161 %    }%
162 %  }%
163 }%
164
165 \let\thmt@original@newtheorem\newtheorem
166 \let\thmt@theoremdefiner\thmt@original@newtheorem
167
168 \def\newtheorem{%
169   \thmt@isstarredfalse
170   \thmt@hassiblingfalse
171   \thmt@hasparentfalse
172   \parse{%
173     {\parseFlag*{\thmt@isstarredtrue}{}}%
174     {\parseMand{\def\thmt@envname{##1}}}%
175     {\parseOpt[]{\thmt@hassiblingtrue\def\thmt@sibling{##1}}}%
176     {\parseMand{\def\thmt@thmname{##1}}}%
177     {\parseOpt[]{\thmt@hasparenttrue\def\thmt@parent{##1}}}%
178     {\let\@parsecmd\thmt@newtheoremiv}%
179   }%
180 }
181
182 \newcommand\thmt@newtheoremiv{%
183   \thmt@newtheorem@predefinition
184   % whee, now reassemble the whole shebang.
185   \protected@edef\thmt@args{%
186     \@nx\thmt@theoremdefiner%
187     \ifthmt@isstarred *\fi
188     {\thmt@envname}%
189     \ifthmt@hassibling [\thmt@sibling]\fi
190     {\thmt@thmname}%
191     \ifthmt@hasparent [\thmt@parent]\fi
192   }
193   \thmt@args
194   \thmt@newtheorem@postdefinition
195 }
196
197 \newcommand\thmt@newtheorem@predefinition{}
198 \newcommand\thmt@newtheorem@postdefinition{%
199   \let\thmt@theoremdefiner\thmt@original@newtheorem
200 }
201
202 \g@addto@macro\thmt@newtheorem@predefinition{%
203   \@xa\thmt@providetheoremhooks\@xa{\thmt@envname}%
204 }
205 \g@addto@macro\thmt@newtheorem@postdefinition{%
206   \@xa\thmt@addtheoremhook\@xa{\thmt@envname}%
207   \ifthmt@isstarred\@namedef{the\thmt@envname}}\fi
208   \protected@edef\thmt@tmp{%
209     \def\@nx\thmt@envname{\thmt@envname}%
210     \def\@nx\thmt@thmname{\thmt@thmname}%
211   }%
212   \@xa\addtotheoremheadhook\@xa[\@xa\thmt@envname\@xa]\@xa{%

```

```

213 \thmt@tmp
214 }%
215 }
216 \newcommand\thmt@providetheoremhooks[1]{%
217 \@namedef{thmt@#1@preheadhook}{}%
218 \@namedef{thmt@#1@postheadhook}{}%
219 \@namedef{thmt@#1@prefoothook}{}%
220 \@namedef{thmt@#1@postfoothook}{}%
221 \def\thmt@local@preheadhook{}%
222 \def\thmt@local@postheadhook{}%
223 \def\thmt@local@prefoothook{}%
224 \def\thmt@local@postfoothook{}%
225 }
226 \newcommand\thmt@addtheoremhook[1]{%
227 % this adds two command calls to the newly-defined theorem.
228 \@xa\let\csname thmt@original@#1\@xa\endcsname
229 \csname#1\endcsname
230 \@xa\renewcommand\csname #1\endcsname{%
231 \thmt@parsetheoremargs{#1}%
232 }%
233 \@xa\let\csname thmt@original@end#1\@xa\endcsname\csname end#1\endcsname
234 \@xa\def\csname end#1\endcsname{%
235 % these need to be in opposite order of headhooks.
236 \csname thmt@generic@prefoothook\endcsname
237 \csname thmt@#1@prefoothook\endcsname
238 \csname thmt@local@prefoothook\endcsname
239 \csname thmt@original@end#1\endcsname
240 \csname thmt@generic@postfoothook\endcsname
241 \csname thmt@#1@postfoothook\endcsname
242 \csname thmt@local@postfoothook\endcsname
243 }%
244 }
245 \newcommand\thmt@generic@preheadhook{\refstepcounter{thmt@dummysctr}}
246 \newcommand\thmt@generic@postheadhook{}
247 \newcommand\thmt@generic@prefoothook{}
248 \newcommand\thmt@generic@postfoothook{}
249
250 \def\thmt@local@preheadhook{}
251 \def\thmt@local@postheadhook{}
252 \def\thmt@local@prefoothook{}
253 \def\thmt@local@postfoothook{}
254
255
256 \providecommand\g@prependto@macro[2]{%
257 \begingroup
258 \toks@\@xa{\@xa{#1}{#2}}%
259 \def\tmp@a##1##2{##2##1}%
260 \@xa\@xa\@xa\gdef\@xa\@xa\@xa#1\@xa\@xa\@xa{\@xa\tmp@a\the\toks@}%
261 \endgroup
262 }
263
264 \newcommand\addtotheoremheadhook[1][generic]{%
265 \expandafter\g@addto@macro\csname thmt@#1@preheadhook\endcsname%
266 }
267 \newcommand\addtotheoremheadhook[1][generic]{%
268 \expandafter\g@addto@macro\csname thmt@#1@postheadhook\endcsname%
269 }
270
271 \newcommand\addtotheoremheadhook[1][generic]{%
272 \expandafter\g@prependto@macro\csname thmt@#1@prefoothook\endcsname%
273 }

```

```

274 \newcommand\addtotheoremfoothook[1][generic]{%
275   \expandafter\g@prependto@macro\csname thmt@#1@postfoothook\endcsname%
276 }
277

```

Since rev1.16, we add hooks to the proof environment as well, if it exists. If it doesn't exist at this point, we're probably using ntheorem as backend, where it goes through the regular theorem mechanism anyway.

```

278 \ifx\proof\endproof\else% yup, that's a quaint way of doing it :)
279 % FIXME: this assumes proof has the syntax of theorems, which
280 % usually happens to be true (optarg overrides "Proof" string).
281 % FIXME: refactor into thmt@addtheoremhook, but we really don't want to
282 % call the generic-hook...
283 \let\thmt@original@proof=\proof
284 \renewcommand\proof{%
285   \thmt@parseproofargs%
286 }%
287 \def\thmt@parseproofargs{%
288   \parse{%
289     {\parseOpt[]{\def\thmt@optarg{##1}}{\let\thmt@optarg@empty}}%
290     {%
291       \thmt@proof@preheadhook
292       %\thmt@generic@preheadhook
293       \protected@edef\tmp@args{%
294         \ifx\@empty\thmt@optarg\else [\thmt@optarg]\fi
295       }%
296       \csname thmt@original@proof\@xa\endcsname\tmp@args
297       \thmt@proof@postheadhook
298       %\thmt@generic@postheadhook
299       \let\@parsecmd\@empty
300     }%
301   }%
302 }%
303
304 \let\thmt@original@endproof=\endproof
305 \def\endproof{%
306   % these need to be in opposite order of headhooks.
307   %\csname thmtgeneric@prefoothook\endcsname
308   \thmt@proof@prefoothook
309   \thmt@original@endproof
310   %\csname thmt@generic@postfoothook\endcsname
311   \thmt@proof@postfoothook
312 }%
313 \@namedef{thmt@proof@preheadhook}{}%
314 \@namedef{thmt@proof@postheadhook}{}%
315 \@namedef{thmt@proof@prefoothook}{}%
316 \@namedef{thmt@proof@postfoothook}{}%
317 \fi

```

### A.1.3 The key-value interfaces

```

318
319 \let\@xa\expandafter
320 \let\@nx\noexpand
321
322 \DeclareOption{lowercase}{%
323   \PackageInfo{thm-kv}{Theorem names will be lowercased}%
324   \global\let\thmt@modifycase\MakeLowercase}
325
326 \DeclareOption{uppercase}{%
327   \PackageInfo{thm-kv}{Theorem names will be uppercased}%
328   \global\let\thmt@modifycase\MakeUppercase}

```

```

329
330 \DeclareOption{anycase}{%
331   \PackageInfo{thm-kv}{Theorem names will be unchanged}%
332   \global\let\thmt@modifycase\@empty}
333
334 \ExecuteOptions{uppercase}
335 \ProcessOptions\relax
336
337 \RequirePackage{keyval,kvsetkeys,thm-patch}
338
339 \long\def\thmt@kv@processor@default#1#2#3{%
340   \def\kvsu@fam{#1}% new
341   \@onelevel@sanitize\kvsu@fam% new
342   \def\kvsu@key{#2}% new
343   \@onelevel@sanitize\kvsu@key% new
344   \unless\ifcsname KV@#1@\kvsu@key\endcsname
345     \unless\ifcsname KVS@#1@handler\endcsname
346       \kv@error@unknownkey{#1}{\kvsu@key}%
347     \else
348       \csname KVS@#1@handler\endcsname{#2}{#3}%
349     % still using #2 #3 here is intentional: handler might
350     % be used for strange stuff like implementing key names
351     % that contain strange characters or other strange things.
352     \relax
353   \fi
354 \else
355   \ifx\kv@value\relax
356     \unless\ifcsname KV@#1@\kvsu@key @default\endcsname
357       \kv@error@novalue{#1}{\kvsu@key}%
358     \else
359       \csname KV@#1@\kvsu@key @default\endcsname
360       \relax
361     \fi
362   \else
363     \csname KV@#1@\kvsu@key\endcsname{#3}%
364   \fi
365 \fi
366 }
367
368 \@ifpackagelater{kvsetkeys}{2012/04/23}{%
369   \PackageInfo{thm-kv}{kvsetkeys patch (v1.16 or later)}%
370   \long\def\tmp@KVS@PD#1#2#3{%
371     \def\kv@fam {#1}%
372     \unless\ifcsname KV@#1@#2\endcsname
373       \unless\ifcsname KVS@#1@handler\endcsname
374         \kv@error@unknownkey {#1}{#2}%
375       \else
376         \kv@handled@true
377         \csname KVS@#1@handler\endcsname {#2}{#3}\relax
378         \ifkv@handled@ \else
379           \kv@error@unknownkey {#1}{#2}%
380         \fi
381       \fi
382     \else
383       \ifx\kv@value\relax
384         \unless\ifcsname KV@#1@#2@default\endcsname
385           \kv@error@novalue {#1}{#2}%
386         \else
387           \csname KV@#1@#2@default\endcsname \relax
388         \fi
389       \else

```

```

390     \csname KV@#1@#2\endcsname {#3}%
391     \fi
392 \fi
393 }%
394 \ifx\tmp@KVS@PD\KVS@ProcessorDefault
395     \let\KVS@ProcessorDefault\thmt@kv@processor@default
396     \def\kv@processor@default#1#2{%
397         \begingroup
398             \csname @safe@activestruel\endcsname
399             \@xa\let\csname ifin\csname\@xa\endcsname\csname iftrue\endcsname
400             \edef\KVS@temp{\endgroup
401 % 2019/12/22 removed dependency on etexcmds package
402                 \noexpand\KVS@ProcessorDefault{#1}{\unexpanded{#2}}}%
403             }%
404             \KVS@temp
405         }%
406     \else
407         \PackageError{thm-kv}{kvsetkeys patch failed}{Try kvsetkeys v1.16 or earlier}
408     \fi
409 }{\@ifpackagelater{kvsetkeys}{2011/04/06}{%
410 % Patch has disappeared somewhere... thanksalot.
411 \PackageInfo{thm-kv}{kvsetkeys patch (v1.13 or later)}
412 \long\def\tmp@KVS@PD#1#2#3{% no non-etex-support here...
413     \unless\ifcsname KV@#1@#2\endcsname
414     \unless\ifcsname KVS@#1@handler\endcsname
415         \kv@error@unknownkey{#1}{#2}%
416     \else
417         \csname KVS@#1@handler\endcsname{#2}{#3}%
418         \relax
419     \fi
420 \else
421     \ifx\kv@value\relax
422     \unless\ifcsname KV@#1@#2@default\endcsname
423         \kv@error@novalue{#1}{#2}%
424     \else
425         \csname KV@#1@#2@default\endcsname
426         \relax
427     \fi
428 \else
429     \csname KV@#1@#2\endcsname{#3}%
430 \fi
431 \fi
432 }%
433 \ifx\tmp@KVS@PD\KVS@ProcessorDefault
434     \let\KVS@ProcessorDefault\thmt@kv@processor@default
435     \def\kv@processor@default#1#2{%
436         \begingroup
437             \csname @safe@activestruel\endcsname
438             \let\ifin\csname\iftrue
439             \edef\KVS@temp{\endgroup
440                 \noexpand\KVS@ProcessorDefault{#1}{\unexpanded{#2}}}%
441             }%
442             \KVS@temp
443         }
444     \else
445         \PackageError{thm-kv}{kvsetkeys patch failed, try kvsetkeys v1.13 or earlier}
446     \fi
447 }{\%
448 \RequirePackage{etex}
449 \PackageInfo{thm-kv}{kvsetkeys patch applied (pre-1.13)}%
450 \let\kv@processor@default\thmt@kv@processor@default

```



```

451 }}
452
453 % useful key handler defaults.
454 \newcommand\thmt@mkignoringkeyhandler[1]{%
455   \kv@set@family@handler{#1}{%
456     \thmt@debug{Key '##1' with value '##2' ignored by #1.}%
457   }%
458 }
459 \newcommand\thmt@mkextendingkeyhandler[3]{%
460 % #1: family
461 % #2: prefix for file
462 % #3: key hint for error
463   \kv@set@family@handler{#1}{%
464     \thmt@selfextendingkeyhandler{#1}{#2}{#3}%
465     {##1}{##2}%
466   }%
467 }
468
469 \newcommand\thmt@selfextendingkeyhandler[5]{%
470 % #1: family
471 % #2: prefix for file
472 % #3: key hint for error
473 % #4: actual key
474 % #5: actual value
475   \IfFileExists{#2-#4.sty}{%
476     \PackageInfo{thmtools}%
477     {Automatically pulling in '#2-#4'}%
478     \RequirePackage{#2-#4}%
479     \ifcsname KV@#1@#4\endcsname
480     \csname KV@#1@#4\endcsname{#5}%
481   \else
482     \PackageError{thmtools}%
483     {#3 '#4' not known}
484     {I don't know what that key does.\MessageBreak
485     I've even loaded the file '#2-#4.sty', but that didn't help.
486     }%
487   \fi
488 }{%
489   \PackageError{thmtools}%
490   {#3 '#4' not known}
491   {I don't know what that key does by myself,\MessageBreak
492   and no file '#2-#4.sty' to tell me seems to exist.
493   }%
494 }%
495 }
496
497
498 \newif\if@thmt@firstkeyset
499
500 % many keys are evaluated twice, because we don't know
501 % if they make sense before or after, or both.
502 \def\thmt@trytwice{%
503   \if@thmt@firstkeyset
504     \@xa\@firstoftwo
505   \else
506     \@xa\@secondoftwo
507   \fi
508 }
509
510 \@for\tmp@keyname:=parent,numberwithin,within\do{%
511   \define@key{thmdef}{\tmp@keyname}{%

```

```

512 \thmt@trytwice{%
513 \thmt@setparent{#1}
514 \thmt@setsibling{}}%
515 }{}%
516 }%
517 }
518 \newcommand\thmt@setparent{%
519 \def\thmt@parent
520 }
521
522 \@for\tmp@keyname:=sibling,numberlike,sharenumber\do{%
523 \define@key{thmdef}{\tmp@keyname}{%
524 \thmt@trytwice{%
525 \thmt@setsibling{#1}}%
526 \thmt@setparent{}}%
527 }{}%
528 }%
529 }
530 \newcommand\thmt@setsibling{%
531 \def\thmt@sibling
532 }
533
534 \@for\tmp@keyname:=title,name,heading\do{%
535 \define@key{thmdef}{\tmp@keyname}{\thmt@trytwice{\thmt@setthmname{#1}}{}}%
536 }
537 \newcommand\thmt@setthmname{%
538 \def\thmt@thmname
539 }
540
541 \@for\tmp@keyname:=unnumbered,starred\do{%
542 \define@key{thmdef}{\tmp@keyname}[]{\thmt@trytwice{\thmt@isnumberedfalse}{}}%
543 }
544
545 \def\thmt@YES{yes}
546 \def\thmt@NO{no}
547 \def\thmt@UNIQUE{unless unique}
548 \newif\ifthmt@isnumbered
549 \newif\ifthmt@isunlesunique
550
551 \define@key{thmdef}{numbered}[yes]{
552 \def\thmt@tmp{#1}%
553 \thmt@trytwice{%
554 \ifx\thmt@tmp\thmt@YES
555 \thmt@isnumberedtrue
556 \else\ifx\thmt@tmp\thmt@NO
557 \thmt@isnumberedfalse
558 \else\ifx\thmt@tmp\thmt@UNIQUE
559 \RequirePackage[unq]{unique}
560 \thmt@isunlesuniquetrue
561 \else
562 \PackageError{thmtools}{Unknown value '#1' to key numbered}{}}%
563 \fi\fi\fi
564 }{} trytwice: after definition
565 \ifx\thmt@tmp\thmt@UNIQUE
566 \ifx\thmt@parent\@empty
567 \addtotheorempreheadhook[\thmt@envname]{\setuniqmark{\thmt@envname}}%
568 \else
569 \protected@edef\thmt@tmp{%
570 % expand \thmt@envname and \thmt@parent
571 \@nx\addtotheorempreheadhook[\thmt@envname @unique]{%
572 \@nx\setuniqmark{\thmt@envname.\@nx\@nameuse{the\thmt@parent}}}%

```

```

573     \@nx\addtotheoremheadhook[\thmt@envname @numbered]{%
574         \@nx\setuniqmark{\thmt@envname.\@nx\@nameuse{the\thmt@parent}}}%
575     \@nx\addtotheoremheadhook[\thmt@envname @unique]{%
576         \def\@nx\thmt@dummysctratorefname{\thmt@thmname\@nx\@gobble}}%
577     \@nx\addtotheoremheadhook[\thmt@envname @numbered]{%
578         \def\@nx\thmt@dummysctratorefname{\thmt@thmname\@nx\@gobble}}%
579     }%
580     \thmt@tmp
581     \fi
582     % \addtotheoremheadhook[\thmt@envname]{%
583     %     \def\thmt@dummysctratorefname{\thmt@thmname\@gobble}}%
584     \fi
585 }%
586 }
587
588
589 \define@key{thmdef}{preheadhook}{%
590     \thmt@trytwice{}{\addtotheoremheadhook[\thmt@envname]{#1}}
591 \define@key{thmdef}{postheadhook}{%
592     \thmt@trytwice{}{\addtotheoremheadhook[\thmt@envname]{#1}}
593 \define@key{thmdef}{prefoothook}{%
594     \thmt@trytwice{}{\addtotheoremheadhook[\thmt@envname]{#1}}
595 \define@key{thmdef}{postfoothook}{%
596     \thmt@trytwice{}{\addtotheoremheadhook[\thmt@envname]{#1}}
597
598 \define@key{thmdef}{style}{\thmt@trytwice{\thmt@setstyle{#1}}{}}
599
600 % ugly hack: style needs to be evaluated first so its keys
601 % are not overridden by explicit other settings
602 \define@key{thmdef0}{style}{%
603     \ifcsname thmt@style #1@defaultkeys\endcsname
604         \thmt@toks{\kvsetkeys{thmdef}}%
605         \@xa\@xa\@xa\the\@xa\@xa\@xa\thmt@toks\@xa\@xa\@xa{%
606             \csname thmt@style #1@defaultkeys\endcsname}%
607     \fi
608 }
609 \thmt@mkignoringkeyhandler{thmdef0}
610
611 % fallback definition.
612 % actually, only the kernel does not provide \theoremstyle.
613 % is this one worth having glue code for the theorem package?
614 \def\thmt@setstyle#1{%
615     \PackageWarning{thm-kv}{%
616         Your backend doesn't have a '\string\theoremstyle' command.
617     }%
618 }
619
620 \ifcsname theoremstyle\endcsname
621     \let\thmt@originalthmstyle\theoremstyle
622     \def\thmt@outerstyle{plain}
623     \renewcommand\theoremstyle[1]{%
624         \def\thmt@outerstyle{#1}%
625         \thmt@originalthmstyle{#1}%
626     }
627     \def\thmt@setstyle#1{%
628         \thmt@originalthmstyle{#1}%
629     }
630     \g@addto@macro\thmt@newtheorem@postdefinition{%
631         \thmt@originalthmstyle{\thmt@outerstyle}%
632     }
633 \fi

```

```

634
635
636 \thmt@mkextendingkeyhandler{thmdef}{thmdef}{\string\declaretheorem\space key}
637
638 \let\thmt@newtheorem\newtheorem
639
640 % \declaretheorem[option list 1]{thmname list}[option list 1]
641 % #1 = option list 1
642 % #2 = thmname list
643 \newcommand\declaretheorem[2][]{%
644 % TODO: use \NewDocumentCommand from xparse?
645 % xparse will be part of latex2e format from latex2e 2020 Oct.
646 \@ifnextchar[%
647   {\declaretheorem@i{#1}{#2}}
648   {\declaretheorem@i{#1}{#2}[ ]}]%
649 }
650 \@onlypreamble\declaretheorem
651
652 % #1 = option list 1
653 % #2 = thmname list
654 % #3 = option list 2
655 \def\declaretheorem@i#1#2[#3]{%
656   \@for\thmt@tmp:=#2\do{%
657     % strip spaces, \KV@@sp@def is defined in keyval.sty
658     \@xa\KV@@sp@def\@xa\thmt@tmp\@xa{\thmt@tmp}%
659     \@xa\declaretheorem@ii\@xa{\thmt@tmp}{#1,#3}%
660   }%
661 }
662
663 % #1 = single thmname (#1 and #2 are exchanged)
664 % #2 = option list
665 \def\declaretheorem@ii#1#2{%
666 % why was that here?
667 %\let\thmt@theoremdefiner\thmt@original@newtheorem
668 % init options
669 \thmt@setparent{}%
670 \thmt@setsibling{}%
671 \thmt@isnumberedtrue
672 \thmt@isunlessequivalfalse
673 \def\thmt@envname{#1}%
674 \thmt@setthmname{\thmt@modifycase #1}%
675 % use true code in \thmt@trytwice{<true>}{<false>}
676 \@thmt@firstkeysettrue
677 % parse options
678 \kvsetkeys{thmdef0}{#2}% parse option "style" first
679 \kvsetkeys{thmdef}{#2}%
680 % call patched \newtheorem
681 \ifthmt@isunlessequival
682   \ifx\thmt@parent\@empty
683     % define normal "unless unique" thm env
684     \ifuniq{#1}{\thmt@isnumberedfalse}{\thmt@isnumberedtrue}%
685     \declaretheorem@iii{#1}%
686   \else
687     % define special "unless unique" thm env,
688     % when "numbered=unless unique" and "numberwithin=<counter>" are both used
689     \declaretheorem@iv{#1}%
690     \thmt@isnumberedtrue
691     \declaretheorem@iii{#1@numbered}%
692     \thmt@isnumberedfalse
693     \declaretheorem@iii{#1@unique}%
694 \fi

```

```

695 \else
696   % define normal thm env
697   \declaretheorem@iii{#1}%
698 \fi
699 % use false code in \thmt@trytwice{<true>}{<false>}
700 \def\thmt@envname{#1}%
701 \@thmt@firstkeysetfalse
702 % uniquely ugly kludge: some keys make only sense afterwards.
703 % and it gets kludgier: again, the default-inherited
704 % keys need to have a go at it.
705 \kvsetkeys{thmdef0}{#2}%
706 \kvsetkeys{thmdef}{#2}%
707 }
708
709 % define normal thm env, call \thmt@newtheorem
710 \def\declaretheorem@iii#1{%
711   \protected@edef\thmt@tmp{%
712     \@nx\thmt@newtheorem
713     \ifthmt@isnumbered
714       {#1}%
715       \ifx\thmt@sibling\@empty\else [\thmt@sibling]\fi
716       {\thmt@thmname}%
717       \ifx\thmt@parent\@empty\else [\thmt@parent]\fi
718     \else
719       *{#1}{\thmt@thmname}%
720     \fi
721     \relax% added so we can delimited-read everything later
722   }%
723   \thmt@debug{Define theorem ‘#1’ by ^^J\meaning\thmt@tmp}%
724   \thmt@tmp
725 }
726
727 % define special thm env
728 \def\declaretheorem@iv#1{%
729   \protected@edef\thmt@tmp{%
730     % expand \thmt@envname and \thmt@parent
731     \@nx\newenvironment{#1}{%
732       \@nx\ifuniq{\thmt@envname.\@nx\@nameuse{the\thmt@parent}}{%
733         \def\@nx\thmt@rawenvname{#1@unique}%
734       }{%
735         \def\@nx\thmt@rawenvname{#1@numbered}%
736       }%
737       \begin{\@nx\thmt@rawenvname}%
738     }{%
739       \end{\@nx\thmt@rawenvname}%
740     }%
741   }%
742   \thmt@debug{Define special theorem ‘#1’ by ^^J\meaning\thmt@tmp}%
743   \thmt@tmp
744 }
745
746 \providecommand\thmt@quark{\thmt@quark}
747
748 % in-document keyval, i.e. \begin{theorem}[key=val,key=val]
749
750 \thmt@mkextendingkeyhandler{thmuse}{thmuse}{\thmt@envname\space optarg key}
751
752 \addtotheoremreheadhook{%
753   \ifx\thmt@optarg\@empty\else
754     \@xa\thmt@garbleoptarg\@xa{\thmt@optarg}\fi
755 }%

```

```

756
757 \newif\ifthmt@thmuse@iskv
758
759 \providecommand\thmt@garbleoptarg[1]{%
760   \thmt@thmuse@iskvfalse
761   \def\thmt@newoptarg{\@gobble}%
762   \def\thmt@newoptargextra{}%
763   \let\thmt@shortoptarg\@empty
764   \def\thmt@warn@unusedkeys{}%
765   \@for\thmt@fam:=\thmt@thmuse@families\do{%
766     \kvsetkeys{\thmt@fam}{#1}%
767   }%
768   \ifthmt@thmuse@iskv
769     \protected@edef\thmt@optarg{%
770       \@xa\thmt@newoptarg
771       \thmt@newoptargextra\@empty
772     }%
773     \ifx\thmt@shortoptarg\@empty
774       \protected@edef\thmt@shortoptarg{\thmt@newoptarg\@empty}%
775     \fi
776     \thmt@warn@unusedkeys
777   \else
778     \def\thmt@optarg{#1}%
779     \def\thmt@shortoptarg{#1}%
780   \fi
781 }
782 % FIXME: not used?
783 \def\thmt@splitopt#1=#2\thmt@quark{%
784   \def\thmt@tmpkey{#1}%
785   \ifx\thmt@tmpkey\@empty
786     \def\thmt@tmpkey{\thmt@quark}%
787   \fi
788   \@onelevel@sanitize\thmt@tmpkey
789 }
790
791 \def\thmt@thmuse@families{thm@track@keys}
792
793 \kv@set@family@handler{thm@track@keys}{%
794   \@onelevel@sanitize\kv@key
795   \@namedef{thmt@unusedkey@\kv@key}{%
796     \PackageWarning{thmtools}{Unused key '#1'}%
797   }%
798   \@xa@g@addto@macro\@xa\thmt@warn@unusedkeys\@xa{%
799     \csname thmt@unusedkey@\kv@key\endcsname
800   }
801 }
802
803 % key, code.
804 \def\thmt@define@thmuse@key#1#2{%
805   \g@addto@macro\thmt@thmuse@families{,#1}%
806   \define@key{#1}{#1}{\thmt@thmuse@iskvtrue
807     \@namedef{thmt@unusedkey@#1}{}%
808     #2}%
809   \thmt@mkignoringkeyhandler{#1}%
810 }
811
812 \thmt@define@thmuse@key{label}{%
813   \addtotheoremshook[local]{\label{#1}}%
814 }
815 \thmt@define@thmuse@key{name}{%
816   \thmt@setnewoptarg #1\@iden%

```

```

817 }
818 \newcommand\thmt@setnewoptarg[1][]{%
819 \def\thmt@shortoptarg{#1}\thmt@setnewlongoptarg
820 }
821 \def\thmt@setnewlongoptarg #1\@iden{%
822 \def\thmt@newoptarg{#1\@iden}}
823
824 \providecommand\thmt@suspendcounter[2]{%
825 \@xa\protected@edef\csname the#1\endcsname{#2}%
826 \@xa\let\csname c@#1\endcsname\c@thmt@dummysctr
827 }
828
829 \providecommand\thmcontinues[1]{%
830 \ifcsname hyperref\endcsname
831 \hyperref[#1]{continuing}
832 \else
833 continuing
834 \fi
835 from p.\,\pageref{#1}%
836 }
837
838 \thmt@define@thmuse@key{continues}{%
839 \thmt@suspendcounter{\thmt@envname}{\thmt@trivialref{#1}{??}}%
840 \g@addto@macro\thmt@newoptarg{{, }%
841 \thmcontinues{#1}%
842 \@iden}%
843 }
844
845

```

Defining new theorem styles; keys are in opt-arg even though not having any doesn't make much sense. It doesn't do anything exciting here, it's up to the glue layer to provide keys.

```

846 \def\thmt@declaretheoremstyle@setup{}
847 \def\thmt@declaretheoremstyle#1{%
848 \PackageWarning{thmtools}{Your backend doesn't allow styling theorems}}
849 }
850 \newcommand\declaretheoremstyle[2][]{%
851 \def\thmt@style{#2}%
852 \@xa\def\csname thmt@style \thmt@style @defaultkeys\endcsname{%
853 \thmt@declaretheoremstyle@setup
854 \kvsetkeys{thmstyle}{#1}%
855 \thmt@declaretheoremstyle{#2}%
856 }
857 \@onlypreamble\declaretheoremstyle
858
859 \kv@set@family@handler{thmstyle}{%
860 \@onelevel@sanitize\kv@value
861 \@onelevel@sanitize\kv@key
862 \PackageInfo{thmtools}{%
863 Key '\kv@key' (with value '\kv@value')\MessageBreak
864 is not a known style key.\MessageBreak
865 Will pass this to every \string\declaretheorem\MessageBreak
866 that uses 'style=\thmt@style'%
867 }%
868 \ifx\kv@value\relax% no value given, don't pass on {}!
869 \@xa\g@addto@macro\csname thmt@style \thmt@style @defaultkeys\endcsname{%
870 #1,%
871 }%
872 \else
873 \@xa\g@addto@macro\csname thmt@style \thmt@style @defaultkeys\endcsname{%
874 #1={#2},%

```

```

875     }%
876   \fi
877 }

```

#### A.1.4 Lists of theorems

This package provides two main commands: `\listoftheorems` will generate, well, a list of all theorems, lemmas, etc. in your document. This list is hyperlinked if you use `hyperref`, and it will list the optional argument to the theorem.

Currently, some options can be given as an optional argument keyval list:

**numwidth** The width allocated for the numbers, default 2.3em. Since you are more likely to have by-section numbering than with figures, this needs to be accessible.

**ignore=foo,bar** A last-second call to `\ignoretheorems`, see below.

**onlynamed=foo,bar** Only list those foo and bar environments that had an optional title. This weeds out unimportant definitions, for example. If no argument is given, this applies to all environments defined by `\newtheorem` and `\declaretheorem`.

**show=foo,bar** Undo a previous `\ignoretheorems` and restore default formatting for these environments. Useful in combination with `ignoreall`.

**ignoreall**

**showall** Like applying ignore or show with a list of all theorems you have defined.

**title** Provide a title for this list overwriting the default in `\listtheoremname`.

The heading name is stored in the macro `\listtheoremname` and is “List of Theorems” by default. All other formatting aspects are taken from `\listoffigures`. (As a matter of fact, `\listoffigures` is called internally.)

`\ignoretheorems{remark,example,...}` can be used to suppress some types of theorem from the LoTh. Be careful not to have spaces in the list, those are currently *not* filtered out.

There’s currently no interface to change the look of the list. If you’re daring, the code for the theorem type “lemma” is in `\l@lemma` and so on.

```

878 \let\@xa=\expandafter
879 \let\@nx=\noexpand
880 \RequirePackage{thm-patch,keyval,kvsetkeys}
881
882 \def\thmtlo@oldchapter{0}%
883 \newcommand\thmtlo@chaptervspacehack{}
884 \ifcsname c@chapter\endcsname
885   \ifx\c@chapter\relax\else
886     \def\thmtlo@chaptervspacehack{%
887       \ifnum \value{chapter}=\thmtlo@oldchapter\relax\else
888         % new chapter, add vspace to loe.
889         \addtocontents{loe}{\protect\addvspace{10\p@}}%
890         \xdef\thmtlo@oldchapter{\arabic{chapter}}%
891       \fi
892     }%
893   \fi
894 \fi
895
896
897 \providecommand\listtheoremname{List of Theorems}
898 \newcommand\listoftheorems[1][]{%
899   %% much hacking here to pick up the definition from the class
900   %% without oodles of conditionals.
901   \begingroup

```



```

902 \setlisttheoremstyle{#1}%
903 \let\listfigurename\listtheoremname
904 \def\contentsline##1{%
905   \csname thmt@contentsline@##1\endcsname{##1}%
906 }%
907 \@for\thmt@envname:=\thmt@allenvs\do{%
908   % CHECK: is \cs{l@\thmt@envname} repeatedly defined?
909   \thmtlo@newentry
910 }%
911 \let\thref@starttoc\@starttoc
912 \def\@starttoc##1{\thref@starttoc{loe}}%
913 % new hack: to allow multiple calls, we defer the opening of the
914 % loe file to AtEndDocument time. This is before the aux file is
915 % read back again, that is early enough.
916 % TODO: is it? crosscheck include/includeonly!
917 \@fileswfalse
918 \AtEndDocument{%
919   \if@filesw
920     \@ifundefined{tf@loe}{%
921       \expandafter\newwrite\csname tf@loe\endcsname
922       \immediate\openout \csname tf@loe\endcsname \jobname.loe\relax
923     }{}%
924   \fi
925 }%
926 %\expandafter
927 \listoffigures
928 \endgroup
929 }
930
931 \newcommand\setlisttheoremstyle[1]{%
932   \kvsetkeys{thmt-listof}{#1}%
933 }
934 \define@key{thmt-listof}{numwidth}{\def\thmt@listnumwidth{#1}}
935 \define@key{thmt-listof}{ignore}[\thmt@allenvs]{\ignoretheorems{#1}}
936 \define@key{thmt-listof}{onlynamed}[\thmt@allenvs]{\onlynamedtheorems{#1}}
937 \define@key{thmt-listof}{show}[\thmt@allenvs]{\showtheorems{#1}}
938 \define@key{thmt-listof}{ignoreall}[true]{\ignoretheorems{\thmt@allenvs}}
939 \define@key{thmt-listof}{showall}[true]{\showtheorems{\thmt@allenvs}}
940 % FMi 2019-09-31 allow local title
941 \define@key{thmt-listof}{title}{\def\listtheoremname{#1}}
942 % -- FMi
943 \newif\ifthmt@listswap
944 \def\thmt@TRUE{true}
945 \def\thmt@FALSE{false}
946 \define@key{thmt-listof}{swapnumber}[true]{%
947   \def\thmt@tmp{#1}%
948   \ifx\thmt@tmp\thmt@TRUE
949     \thmt@listswaptrue
950   \else\ifx\thmt@tmp\thmt@FALSE
951     \thmt@listswapfalse
952   \else
953     \PackageError{thmtools}{Unknown value '#1' to key swapnumber}{}%
954   \fi\fi
955 }
956
957 \ifdefined\@tocline
958 % for ams classes (amsart.cls, amsproc.cls, amsbook.cls) which
959 % don't use \@dottedtocline and don't provide \@dotsep
960 \def\thmtlo@newentry{%
961   \@xa\def\csname l@\thmt@envname\endcsname{% CHECK: why p@edef?
962     % similar to \l@figure defined in ams classes

```

```

963     \@tocline{0}{3pt plus2pt}{0pt}{\thmt@listnumwidth}{}%
964 }%
965 }
966 \providecommand*\thmt@listnumwidth{1.5pc}
967 \else
968 \def\thmtlo@newentry{%
969   \@xa\def\csname ll@\thmt@envname\endcsname{% CHECK: why p@edef?
970     \@dottedtocline{1}{1.5em}{\thmt@listnumwidth}%
971   }%
972 }
973 \providecommand*\thmt@listnumwidth{2.3em}
974 \fi
975
976 \providecommand\thmtformatoptarg[1]{ (#1)}
977
978 \newcommand\thmt@mklistcmd{%
979   \thmtlo@newentry
980   \ifthmt@isstarred
981     \@xa\def\csname ll@\thmt@envname\endcsname{%
982       \protect\ifthmt@listswap
983       \protect\else
984         \protect\numberline{\protect\let\protect\autodot\protect\@empty}%
985       \protect\fi
986       \thmt@thmname
987       \ifx\@empty\thmt@shortoptarg\else\protect\thmtformatoptarg{\thmt@shortoptarg}\fi
988     }%
989   \else
990     \@xa\def\csname ll@\thmt@envname\endcsname{%
991       \protect\ifthmt@listswap
992         \thmt@thmname~\csname the\thmt@envname\endcsname
993       \protect\else
994         \protect\numberline{\csname the\thmt@envname\endcsname}%
995         \thmt@thmname
996       \protect\fi
997       \ifx\@empty\thmt@shortoptarg\else\protect\thmtformatoptarg{\thmt@shortoptarg}\fi
998     }%
999   \fi
1000   \@xa\gdef\csname thmt@contentsline@\thmt@envname\endcsname{%
1001     \thmt@contentslineShow% default:show
1002   }%
1003 }
1004 \def\thmt@allenvs{\@gobble}
1005 \newcommand\thmt@recordenvname{%
1006   \edef\thmt@allenvs{\thmt@allenvs,\thmt@envname}%
1007 }
1008 \g@addto@macro\thmt@newtheorem@predefinition{%
1009   \thmt@mklistcmd
1010   \thmt@recordenvname
1011 }
1012
1013 \addtotheoremsoheadhook{%
1014   \thmtlo@chaptervspacehack
1015   \addcontentsline{loe}{\thmt@envname}{%
1016     \csname ll@\thmt@envname\endcsname
1017   }%
1018 }
1019
1020 \newcommand\showtheorems[1]{%
1021   \@for\thmt@thm:=#1\do{%
1022     \typeout{showing \thmt@thm}%
1023     \@xa\let\csname thmt@contentsline@\thmt@thm\endcsname

```

```

1024     =\thmt@contentslineShow
1025 }%
1026 }
1027
1028 \newcommand\ignoretheorems[1]{%
1029   \@for\thmt@thm:=#1\do{%
1030     \@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
1031     =\thmt@contentslineIgnore
1032   }%
1033 }
1034 \newcommand\onlynamedtheorems[1]{%
1035   \@for\thmt@thm:=#1\do{%
1036     \global\@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
1037     =\thmt@contentslineIfNamed
1038   }%
1039 }
1040
1041 \AtBeginDocument{%
1042   \@ifpackageloaded{hyperref}{%
1043     \let\thmt@hygobble\@gobble
1044   }{%
1045     \let\thmt@hygobble\@empty
1046   }
1047   \let\thmt@contentsline\contentsline
1048 }
1049
1050 \def\thmt@contentslineIgnore#1#2#3{%
1051   \thmt@hygobble
1052 }
1053 \def\thmt@contentslineShow{%
1054   \thmt@contentsline
1055 }
1056
1057 \def\thmt@contentslineIfNamed#1#2#3{%
1058   \thmt@ifhasoptname #2\thmtformatoptarg\@nil{%
1059     \thmt@contentslineShow{#1}{#2}{#3}%
1060   }{%
1061     \thmt@contentslineIgnore{#1}{#2}{#3}%
1062     %\thmt@contentsline{#1}{#2}{#3}%
1063   }
1064 }
1065
1066 \def\thmt@ifhasoptname #1\thmtformatoptarg#2\@nil{%
1067   \ifx\@nil#2\@nil
1068     \@xa\@secondoftwo
1069   \else
1070     \@xa\@firstoftwo
1071   \fi
1072 }

```

### A.1.5 Re-using environments

Only one environment is provided: `restatable`, which takes one optional and two mandatory arguments. The first mandatory argument is the type of the theorem, i.e. if you want `\begin{lemma}` to be called on the inside, give `lemma`. The second argument is the name of the macro that the text should be stored in, for example `mylemma`. Be careful not to specify existing command names! The optional argument will become the optional argument to your theorem command. Consider the following example:

```

\documentclass{article}
\usepackage{amsmath, amsthm, thm-restate}

```

```

\newtheorem{lemma}{Lemma}
\begin{document}
  \begin{restatable}[Zorn]{lemma}{zornlemma}\label{thm:zorn}
    If every chain in  $X$  is upper-bounded,
     $X$  has a maximal element.

    It's true, you know!
  \end{restatable}
  \begin{lemma}
    This is some other lemma of no import.
  \end{lemma}
  And now, here's Mr. Zorn again: \zornlemma*
\end{document}

```

which yields

**Lemma 4** (Zorn). *If every chain in  $X$  is upper-bounded,  $X$  has a maximal element.*

*It's true, you know!*

**Lemma 5.** *This is some other lemma of no import.*

Actually, we have set a label in the environment, so we know that it's Lemma 4 on page 4. And now, here's Mr. Zorn again:

**Lemma 4** (Zorn). *If every chain in  $X$  is upper-bounded,  $X$  has a maximal element.*

*It's true, you know!*

Since we prevent the label from being set again, we find that it's still Lemma 4 on page 4, even though it occurs later also.

As you can see, we use the starred form `\mylemma*`. As in many cases in  $\text{\LaTeX}$ , the star means “don't give a number”, since we want to retain the original number. There is also a starred variant of the `restatable` environment, where the first call doesn't determine the number, but a later call to `\mylemma` without star would. Since the number is carried around using  $\text{\LaTeX}$  `\label` mechanism, you'll need a rerun for things to settle.

### A.1.6 Restrictions

The only counter that is saved is the one for the theorem number. So, putting floats inside a `restatable` is not advised: they will appear in the LoF several times with new numbers. Equations should work, but the code handling them might turn out to be brittle, in particular when you add/remove `hyperref`. In the same vein, numbered equations within the statement appear again and are numbered again, with new numbers. (This is vaguely non-trivial to do correctly if equations are not numbered consecutively, but per-chapter, or there are multiple numbered equations.) Note that you cannot successfully reference the equations since all labels are disabled in the starred appearance. (The reference will point at the unstarred occurrence.)

You cannot nest `restatables` either. You *can* use the `\restatable... \endrestatable` version, but everything up to the next matching `\end{...}` is scooped up. I've also probably missed many border cases.

```

1073 \RequirePackage{thmtools}
1074 \let\@xa\expandafter
1075 \let\@nx\noexpand
1076 \@ifundefined{c@thmt@dummysctr}{%
1077   \newcounter{thmt@dummysctr}%
1078   }{}
1079 \gdef\theHthmt@dummysctr{dummy.\arabic{thmt@dummysctr}}%
1080 \gdef\thethmt@dummysctr{}%
1081 \long\def\thmt@collect@body#1#2\end#3{%
1082   \@xa\thmt@toks\@xa{\the\thmt@toks #2}%
1083   \def\thmttmpa{#3}%\def\thmttmpb{restatable}%
1084   \ifx\thmttmpa@\currentenv\thmttmpb
1085     \@xa@firstoftwo% this is the end of the environment.
1086   \else

```

```

1087 \@xa\@secondoftwo% go on collecting
1088 \fi{% this is the end, my friend, drop the \end.
1089 % and call #1 with the collected body.
1090 \@xa#1\@xa{\the\thmt@toks}%
1091 }{% go on collecting
1092 \@xa\thmt@toks\@xa{\the\thmt@toks\end{#3}}%
1093 \thmt@collect@body{#1}%
1094 }%
1095 }

```

A totally ignorant version of `\ref`, defaulting to #2 if label not known yet. Otherwise, return the formatted number.

```

1096 \def\thmt@trivialref#1#2{%
1097 \ifcsname r@#1\endcsname
1098 \@xa\@xa\@xa\thmt@trivi@lr@f\csname r@#1\endcsname\relax\@nil
1099 \else #2\fi
1100 }
1101 \def\thmt@trivi@lr@f#1#2\@nil{#1}

```

Counter safeties: some counters' values should be stored, such as equation, so we don't get a new number. (We cannot reference it anyway.) We cannot store everything, though, think page counter or section number! There is one problem here: we have to remove all references to other counters from `\theequation`, otherwise your equation could get a number like (3.1) in one place and (4.1) in another section.

The best solution I can come up with is to override the usual macros that counter display goes through, to check if their argument is one that should be fully-expanded away or retained.

The following should only be called from within a group, and the sanitized `\thectr` must not be called from within that group, since it needs the original `\@arabic` et al.

```

1102 \def\thmt@innercounters{%
1103 equation}
1104 \def\thmt@counterformatters{%
1105 @alph,@Alph,@arabic,@roman,@Roman,@fnsymbol}
1106
1107 \@for\thmt@displ:=\thmt@counterformatters\do{%
1108 \@xa\let\csname thmt@\thmt@displ\@xa\endcsname\csname \thmt@displ\endcsname
1109 }%
1110 \def\thmt@sanitizethe#1{%
1111 \@for\thmt@displ:=\thmt@counterformatters\do{%
1112 \@xa\protected@edef\csname\thmt@displ\endcsname##1{%
1113 \@nx\ifx\@xa\@nx\csname c@#1\endcsname ##1%
1114 \@xa\protect\csname \thmt@displ\endcsname{##1}%
1115 \@nx\else
1116 \@nx\csname thmt@\thmt@displ\endcsname{##1}%
1117 \@nx\fi
1118 }%
1119 }%
1120 \expandafter\protected@edef\csname the#1\endcsname{\csname the#1\endcsname}%
1121 \ifcsname theH#1\endcsname
1122 \expandafter\protected@edef\csname theH#1\endcsname{\csname theH#1\endcsname}%
1123 \fi
1124 }
1125
1126 \def\thmt@rst@storecounters#1{%
1127 \bgroup
1128 % ugly hack: save chapter,..subsection numbers
1129 % for equation numbers.
1130 %\refstepcounter{thmt@dummysctr}% why is this here?
1131 %% temporarily disabled, broke autorefname.
1132 \def\@currentlabel{}%
1133 \@for\thmt@ctr:=\thmt@innercounters\do{%
1134 \thmt@sanitizethe{\thmt@ctr}%

```

```

1135 \protected@edef\@currentlabel{%
1136 \@currentlabel
1137 \protect\def\@xa\protect\csname the\thmt@ctr\endcsname{%
1138 \csname the\thmt@ctr\endcsname}%
1139 \ifcsname theH\thmt@ctr\endcsname
1140 \protect\def\@xa\protect\csname theH\thmt@ctr\endcsname{%
1141 (restate \protect\theH\thmt@dummysctr)\csname theH\thmt@ctr\endcsname}%
1142 \fi
1143 \protect\setcounter{\thmt@ctr}{\number\csname c@\thmt@ctr\endcsname}%
1144 }%
1145 }%
1146 \label{thmt@@#1@data}%
1147 \egroup
1148 }%

```

Now, the main business.

```

1149 \newif\ifthmt@thisistheone
1150 \newenvironment{thmt@restatable}[3][[]]{%
1151 \thmt@toks{}}% will hold body
1152 %
1153 \stepcounter{thmt@dummysctr}% used for data storage label.
1154 %
1155 \long\def\thmrst@store##1{%
1156 \@xa\gdef\csname #3\endcsname{%
1157 \@ifstar{%
1158 \thmt@thisistheonefalse\csname thmt@stored@#3\endcsname
1159 }{%
1160 \thmt@thisistheonettrue\csname thmt@stored@#3\endcsname
1161 }%
1162 }%
1163 \@xa\long\@xa\gdef\csname thmt@stored@#3\@xa\endcsname\@xa{%
1164 \begingroup
1165 \ifthmt@thisistheone
1166 % these are the valid numbers, store them for the other
1167 % occasions.
1168 \thmt@rst@storecounters{#3}%
1169 \else
1170 % this one should use other numbers...
1171 % first, fake the theorem number.
1172 \@xa\protected@edef\csname the#2\endcsname{%
1173 \thmt@trivialref{thmt@@#3}{??}}%
1174 % if the number wasn't there, have a "re-run to get labels right"
1175 % warning.
1176 \ifcsname r@thmt@@#3\endcsname\else
1177 \G@refundefinedtrue
1178 \fi
1179 % prevent stepcountering the theorem number,
1180 % but still, have some number for hyperref, just in case.
1181 \@xa\let\csname c#2\endcsname=c@thmt@dummysctr
1182 \@xa\let\csname theH#2\endcsname=\theH\thmt@dummysctr
1183 % disable labeling.
1184 \let\label=\thmt@gobble@label
1185 \let\ltx@label=\@gobble% amsmath needs this
1186 % We shall need to restore the counters at the end
1187 % of the environment, so we get
1188 % (4.2) [(3.1 from restate)] (4.3)
1189 \def\thmt@restorecounters{%
1190 \@for\thmt@ctr:=\thmt@innercounters\do{%
1191 \protected@edef\thmt@restorecounters{%
1192 \thmt@restorecounters
1193 \protect\setcounter{\thmt@ctr}{\arabic{\thmt@ctr}}%

```

```

1194     }%
1195   }%
1196   % pull the new semi-static definition of \theequation et al.
1197   % from the aux file.
1198   \thmt@trivialref{thmt@@#3@data}{}%
1199   \fi
1200   % call the proper begin-env code, possibly with optional argument
1201   % (omit if stored via key-val)
1202   \ifthmt@restatethis
1203     \thmt@restatethisfalse
1204   \else
1205     \csname #2\@xa\endcsname\ifx\@nx#1\@nx\else[#1]\fi
1206   \fi
1207   \ifthmt@thisistheone
1208     % store a label so we can pick up the number later.
1209     \label{thmt@@#3}%
1210   \fi
1211   % this will be the collected body.
1212   ##1%
1213   \csname end#2\endcsname
1214   % if we faked the counter values, restore originals now.
1215   \ifthmt@thisistheone\else\thmt@restorecounters\fi
1216   \endgroup
1217 }% thmt@stored@#3
1218 % in either case, now call the just-created macro,
1219 \csname #3\@xa\endcsname\ifthmt@thisistheone\else*\fi
1220 % and artificially close the current environment.
1221 \@xa\end\@xa{\@currenvir}
1222 }% thm@rst@store
1223 \thmt@collect@body\thm@rst@store
1224 }{%
1225 %% now empty, just used as a marker.
1226 }
1227
1228 \let\thmt@gobble@label\@gobble
1229 % cleveref extends syntax of \label to \label[...]{...}
1230 \AtBeginDocument{
1231   \@ifpackageloaded{cleveref}{
1232     \renewcommand*\thmt@gobble@label[2][{}]{
1233   }{}
1234 }
1235
1236 \newenvironment{restatable}{%
1237   \thmt@thisistheonetrue\thmt@restatable
1238 }{%
1239   \endthmt@restatable
1240 }
1241 \newenvironment{restatable*}{%
1242   \thmt@thisistheonefalse\thmt@restatable
1243 }{%
1244   \endthmt@restatable
1245 }
1246
1247 %%% support for keyval-style: restate=foobar
1248 \protected@edef\thmt@thmuse@families{%
1249   \thmt@thmuse@families%
1250   ,restate phase 1%
1251   ,restate phase 2%
1252 }
1253 \newcommand\thmt@splitrestateargs[1][{}]{%
1254   \g@addto@macro\thmt@storedoptargs{,#1}%

```

```

1255 \def\tmp@a##1\@{\def\thmt@storename{##1}}%
1256 \tmp@a
1257 }
1258
1259 \newif\ifthmt@restatethis
1260 \define@key{restate phase 1}{restate}{%
1261 \thmt@thmuse@iskvtrue
1262 \def\thmt@storedoptargs{}% discard the first time around
1263 \thmt@splitrestateargs #1\@
1264 \def\thmt@storedoptargs{}% discard the first time around
1265 %\def\thmt@storename{##1}%
1266 \thmt@debug{we will restate as '\thmt@storename' with more args
1267 '\thmt@storedoptargs'}%
1268 \@namedef{thmt@unusedkey@restate}{}%
1269 % spurious "unused key" fixes itself once we are after tracknames...
1270 \thmt@restatethistrue
1271 \protected@edef\tmp@a{%
1272 \@nx\thmt@thisistheonetrue
1273 \@nx\def\@nx\@currenvir{\thmt@envname}%
1274 \@nx\@xa\@nx\thmt@restatable\@nx\@xa[\@nx\thmt@storedoptargs]%
1275 {\thmt@envname}{\thmt@storename}%
1276 }%
1277 \@xa\g@addto@macro\@xa\thmt@local@postheadhook\@xa{%
1278 \tmp@a
1279 }%
1280 }
1281 \thmt@mkignoringkeyhandler{restate phase 1}
1282
1283 \define@key{restate phase 2}{restate}{%
1284 % do not store restate as a key for repetition:
1285 % infinite loop.
1286 % instead, retain the added keyvals
1287 % overwriting thmt@storename should be safe here, it's been
1288 % xdefd into the postheadhook
1289 \thmt@splitrestateargs #1\@
1290 }
1291 \kv@set@family@handler{restate phase 2}{%
1292 \ifthmt@restatethis
1293 \@xa\@xa\@xa\g@addto@macro\@xa\@xa\@xa\thmt@storedoptargs\@xa\@xa\@xa{\@xa\@xa\@xa,%
1294 \@xa\kv@key\@xa=\kv@value}%
1295 \fi
1296 }
1297

```

### A.1.7 Fixing `\autoref` and friends

hyperref's `\autoref` command does not work well with theorems that share a counter: it'll always think it's a Lemma even if it's a Remark that shares the Lemma counter. Load this package to fix it. No further intervention needed.

```

1298
1299 \RequirePackage{thm-patch, aliasctr, parseargs, keyval}
1300
1301 \let\@xa=\expandafter
1302 \let\@nx=\noexpand
1303
1304 \newcommand\thmt@autorefsetup{%
1305 \@xa\def\csname\thmt@envname autorefname\@xa\endcsname\@xa{\thmt@thmname}%
1306 \ifthmt@hassibling
1307 \@counteralias{\thmt@envname}{\thmt@sibling}%
1308 \@xa\def\@xa\thmt@autoreffix\@xa{%

```



```

1309     \@xa\global\@xa\let\csname the\thmt@envname\@xa\endcsname
1310     \csname the\thmt@sibling\endcsname
1311     \def\thmt@autoreffix{}%
1312     }%
1313     \protected@edef\thmt@sibling{\thmt@envname}%
1314 \fi
1315 }
1316 \g@addto@macro\thmt@newtheorem@predefinition{\thmt@autoreffix}%
1317 \g@addto@macro\thmt@newtheorem@postdefinition{\csname thmt@autoreffix\endcsname}%
1318
1319 \def\thmt@refnamewithcomma #1#2#3,#4,#5\@nil{%
1320   \@xa\def\csname\thmt@envname #1\utorefname\endcsname{#3}%
1321   \ifcsname #2refname\endcsname
1322     \csname #2refname\@xa\endcsname\@xa{\thmt@envname}{#3}{#4}%
1323   \fi
1324 }
1325 \define@key{thmdef}{refname}{\thmt@trytwice}{%
1326   \thmt@refnamewithcomma{a}{c}#1,\textbf{?? (pl. #1)},\@nil
1327 }}
1328 \define@key{thmdef}{Refname}{\thmt@trytwice}{%
1329   \thmt@refnamewithcomma{A}{C}#1,\textbf{?? (pl. #1)},\@nil
1330 }}
1331
1332
1333 \ifcsname Autoref\endcsname\else
1334 \let\thmt@HyRef@testreftype\HyRef@testreftype
1335 \def\HyRef@Testreftype#1.#2\@nil{%
1336   \ltx@ifundefined{#1Autorefname}{%
1337     \thmt@HyRef@testreftype#1.#2\@nil
1338   }{%
1339     \edef\HyRef@currentHtag{%
1340       \expandafter\noexpand\csname#1Autorefname\endcsname
1341       \noexpand~%
1342     }%
1343   }%
1344 }
1345
1346
1347 \let\thmt@HyPsd@@autorefname\HyPsd@@autorefname
1348 \def\HyPsd@@Autorefname#1.#2\@nil{%
1349   \tracingall
1350   \ltx@ifundefined{#1Autorefname}{%
1351     \thmt@HyPsd@@autorefname#1.#2\@nil
1352   }{%
1353     \csname#1Autorefname\endcsname\space
1354   }%
1355 }%
1356 \def\Autoref{%
1357   \parse{%
1358     {\parseFlag*\def\thmt@autorefstar{*}}{\let\thmt@autorefstar\@empty}}%
1359   {\parseMand{%
1360     \bgroup
1361     \let\HyRef@testreftype\HyRef@Testreftype
1362     \let\HyPsd@@autorefname\HyPsd@@Autorefname
1363     \@xa\autoref\thmt@autorefstar{##1}%
1364     \egroup
1365     \let\@parsecmd\@empty
1366   }}%
1367 }%
1368 }
1369 \fi % ifcsname Autoref

```

```

1370
1371 % not entirely appropriate here, but close enough:
1372 \AtBeginDocument{%
1373   \@ifpackageloaded{nameref}{%
1374     \addtotheorempostheadhook{%
1375       \expandafter\NR@getttitle\expandafter{\thmt@shortoptarg}%
1376     }}{}
1377 }
1378
1379 \AtBeginDocument{%
1380   \@ifpackageloaded{cleveref}{%
1381     \@ifpackagelater{cleveref}{2010/04/30}{%
1382       % OK, new enough
1383     }{%
1384       \PackageWarningNoLine{thmtools}{%
1385         Your version of cleveref is too old!\MessageBreak
1386         Update to version 0.16.1 or later%
1387       }
1388     }
1389   }{}
1390 }

```

## A.2 Glue code for different backends

### A.2.1 amsthm

```

1391 \providecommand\thmt@space{ }
1392
1393 \define@key{thmstyle}{spaceabove}{%
1394   \def\thmt@style@spaceabove{#1}%
1395 }
1396 \define@key{thmstyle}{spacebelow}{%
1397   \def\thmt@style@spacebelow{#1}%
1398 }
1399 \define@key{thmstyle}{headfont}{%
1400   \def\thmt@style@headfont{#1}%
1401 }
1402 \define@key{thmstyle}{bodyfont}{%
1403   \def\thmt@style@bodyfont{#1}%
1404 }
1405 \define@key{thmstyle}{notefont}{%
1406   \def\thmt@style@notefont{#1}%
1407 }
1408 \define@key{thmstyle}{headpunct}{%
1409   \def\thmt@style@headpunct{#1}%
1410 }
1411 \define@key{thmstyle}{notebraces}{%
1412   \def\thmt@style@notebraces{\thmt@embrace#1}%
1413 }
1414 \define@key{thmstyle}{break}[]{}
1415 \def\thmt@style@postheadspace{\newline}%
1416 }
1417 \define@key{thmstyle}{postheadspace}{%
1418   \def\thmt@style@postheadspace{#1}%
1419 }
1420 \define@key{thmstyle}{headindent}{%
1421   \def\thmt@style@headindent{#1}%
1422 }
1423
1424 \newtoks\thmt@style@headstyle

```

```

1425 \define@key{thmstyle}{headformat}[]{%
1426   \thmt@setheadstyle{#1}%
1427 }
1428 \define@key{thmstyle}{headstyle}[]{%
1429   \thmt@setheadstyle{#1}%
1430 }
1431 \def\thmt@setheadstyle#1{%
1432   \thmt@style@headstyle{%
1433     \def\NAME{\the\thm@headfont ##1}%
1434     \def\NUMBER{\bgroup\@upn{##2}\egroup}%
1435     \def\NOTE{\if=##3=\else\bgroup\thmt@space\the\thm@notefont(##3)\egroup\fi}%
1436   }%
1437   \def\thmt@tmp{#1}%
1438   \@onelevel@sanitize\thmt@tmp
1439   %\tracingall
1440   \ifcsname thmt@headstyle@\thmt@tmp\endcsname
1441     \thmt@style@headstyle\@xa{%
1442       \the\thmt@style@headstyle
1443       \csname thmt@headstyle@#1\endcsname
1444     }%
1445   \else
1446     \thmt@style@headstyle\@xa{%
1447       \the\thmt@style@headstyle
1448       #1%
1449     }%
1450   \fi
1451   %\showthe\thmt@style@headstyle
1452 }
1453 % examples:
1454 \def\thmt@headstyle@margin{%
1455   \makebox[Opt][r]{\NUMBER\ }\NAME\NOTE
1456 }
1457 \def\thmt@headstyle@swapnumber{%
1458   \NUMBER\ \NAME\NOTE
1459 }
1460
1461
1462
1463 \def\thmt@embrace#1#2(#3){#1#3#2}
1464
1465 \def\thmt@declaretheoremstyle@setup{%
1466   \let\thmt@style@notebraces\@empty%
1467   \thmt@style@headstyle{ }%
1468   \kvsetkeys{thmstyle}{%
1469     spaceabove=3pt,
1470     spacebelow=3pt,
1471     headfont=\bfseries,
1472     bodyfont=\normalfont,
1473     headpunct={.},
1474     postheadspace={ },
1475     headindent={},
1476     notefont={\fontseries\mdefault\upshape}
1477   }%
1478 }
1479 \def\thmt@declaretheoremstyle#1{%
1480   %\show\thmt@style@spaceabove
1481   \thmt@toks{\newtheoremstyle{#1}}%
1482   \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@spaceabove}}%
1483   \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@spacebelow}}%
1484   \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@bodyfont}}%
1485   \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@headindent}}% indent1 FIXM

```

```

1486 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@headfont}}%
1487 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@headpunct}}%
1488 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@postheadspace}}%
1489 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\the\thmt@style@headstyle}}% headspec
1490 \the\thmt@toks
1491 %1 Indent amount: empty = no indent, \parindent = normal paragraph indent
1492 %2 Space after theorem head: { } = normal interword space; \newline = linebreak
1493 %% BUGFIX: amsthm ignores notefont setting altogether:
1494 \thmt@toks\@xa\@xa\@xa{\csname th@#1\endcsname}%
1495 \thmt@toks
1496 \@xa\@xa\@xa\@xa\@xa\@xa\@xa{\%
1497 \@xa\@xa\@xa\@xa\@xa\@xa\@xa\thm@notefont
1498 \@xa\@xa\@xa\@xa\@xa\@xa\@xa{\%
1499 \@xa\@xa\@xa\thmt@style@notefont
1500 \@xa\thmt@style@notebraces
1501 \@xa}\the\thmt@toks}%
1502 \@xa\def\csname th@#1\@xa\endcsname\@xa{\the\thmt@toks}%
1503 % \@xa\def\csname th@#1\@xa\@xa\@xa\@xa\@xa\@xa\@xa\endcsname
1504 %   \@xa\@xa\@xa\@xa\@xa\@xa\@xa\@xa{\%
1505 %   \@xa\@xa\@xa\@xa\@xa\@xa\@xa\thm@notefont
1506 %   \@xa\@xa\@xa\@xa\@xa\@xa\@xa\@xa{\%
1507 %   \@xa\@xa\@xa\thmt@style@notefont
1508 %   \@xa\@xa\@xa\thmt@style@notebraces
1509 %   \@xa\@xa\@xa}\csname th@#1\endcsname
1510 % }
1511 }
1512
1513 \define@key{thmdef}{qed}[\qedsymbol]{%
1514   \thmt@trytwice}{%
1515     \addtotheorempostheadhook[\thmt@envname]{%
1516       \protected@edef\qedsymbol{#1}%
1517       \pushQED{\qed}%
1518     }%
1519     \addtotheoremreffoothook[\thmt@envname]{%
1520       \protected@edef\qedsymbol{#1}%
1521       \popQED
1522     }%
1523   }%
1524 }
1525
1526 \def\thmt@amsthmlistbreakhack{%
1527   \leavevmode
1528   \vspace{-\baselineskip}%
1529   \par
1530   \everypar{\setbox\z@\lastbox\everypar{}}}%
1531 }
1532
1533 \define@key{thmuse}{listhack}[\relax]{%
1534   \addtotheorempostheadhook[local]{%
1535     \thmt@amsthmlistbreakhack
1536   }%
1537 }
1538

```

## A.2.2 beamer

```

1539 \newif\ifthmt@hasoverlay
1540 \def\thmt@parsetheoremargs#1{%
1541   \parse{%
1542     {\parseOpt<>{\thmt@hasoverlaytrue\def\thmt@overlay{##1}}}{}}%
1543     {\parseOpt[]{\def\thmt@optarg{##1}}}{%

```

```

1544     \let\thmt@shortoptarg\@empty
1545     \let\thmt@optarg\@empty}}%
1546     {\ifthmt@hasoverlay\expandafter\@gobble\else\expandafter\@firstofone\fi
1547       {\parseOpt<>\thmt@hasoverlaytrue\def\thmt@overlay{##1}}}%
1548   }%
1549   {%
1550     \def\thmt@local@preheadhook{}%
1551     \def\thmt@local@postheadhook{}%
1552     \def\thmt@local@prefoothook{}%
1553     \def\thmt@local@postfoothook{}%
1554     \thmt@local@preheadhook
1555     \csname thmt@#1@preheadhook\endcsname
1556     \thmt@generic@preheadhook
1557     \protected@edef\tmp@args{%
1558       \ifthmt@hasoverlay <\thmt@overlay>\fi
1559       \ifx\@empty\thmt@optarg\else [{\thmt@optarg}]\fi
1560     }%
1561     \csname thmt@original@#1\@xa\endcsname\tmp@args
1562     \thmt@local@postheadhook
1563     \csname thmt@#1@postheadhook\endcsname
1564     \thmt@generic@postheadhook
1565     \let\@parsecmd\@empty
1566   }%
1567 }
1568 }%

```

### A.2.3 ntheorem

```

1569
1570 \providecommand\thmt@space{ }
1571
1572 % actually, ntheorem's so-called style is nothing like a style at all...
1573 \def\thmt@declaretheoremstyle@setup{}
1574 \def\thmt@declaretheoremstyle#1{%
1575   \ifcsname th@#1\endcsname\else
1576     \@xa\let\csname th@#1\endcsname\th@plain
1577   \fi
1578 }
1579
1580 \def\thmt@notsupported#1#2{%
1581   \PackageWarning{thmtools}{Key '#2' not supported by #1}}%
1582 }
1583
1584 \define@key{thmstyle}{spaceabove}{%
1585   \setlength\theorempreskipamount{#1}%
1586 }
1587 \define@key{thmstyle}{spacebelow}{%
1588   \setlength\theorempostskipamount{#1}%
1589 }
1590 \define@key{thmstyle}{headfont}{%
1591   \theoremheaderfont{#1}%
1592 }
1593 \define@key{thmstyle}{bodyfont}{%
1594   \theorembodyfont{#1}%
1595 }
1596 % not supported in ntheorem.
1597 \define@key{thmstyle}{notefont}{%
1598   \thmt@notsupported{ntheorem}{notefont}%
1599 }
1600 \define@key{thmstyle}{headpunct}{%
1601   \theoremseparator{#1}%

```

```

1602 }
1603 % not supported in ntheorem.
1604 \define@key{thmstyle}{notebraces}{%
1605   \thmt@notsupported{ntheorem}{notebraces}%
1606 }
1607 \define@key{thmstyle}{break}{%
1608   \theoremstyle{break}%
1609 }
1610 % not supported in ntheorem...
1611 \define@key{thmstyle}{postheadspace}{%
1612   %\def\thmt@style@postheadspace{#1}%
1613   \@xa\g@addto@macro\csname thmt@style \thmt@style @defaultkeys\endcsname{%
1614     postheadhook={\hspace{-\labelsep}\hspace*{#1}},%
1615   }%
1616 }
1617
1618 % not supported in ntheorem
1619 \define@key{thmstyle}{headindent}{%
1620   \thmt@notsupported{ntheorem}{headindent}%
1621 }
1622 % sorry, only style, not def with ntheorem.
1623 \define@key{thmstyle}{qed}[\qedsymbol]{%
1624   \@ifpackagewith{ntheorem}{thmmarks}{%
1625     \theoremsymbol{#1}%
1626   }{%
1627     \thmt@notsupported
1628     {ntheorem without thmmarks option}%
1629     {headindent}%
1630   }%
1631 }
1632
1633 \let\@upn=\textup
1634 \define@key{thmstyle}{headformat}[]{%
1635   \def\thmt@tmp{#1}%
1636   \@onelevel@sanitize\thmt@tmp
1637   %\tracingall
1638   \ifcsname thmt@headstyle@\thmt@tmp\endcsname
1639     \newtheoremstyle{\thmt@style}{%
1640       \item[\hskip\labelsep\theorem@headerfont%
1641         \def\NAME{\theorem@headerfont #####1}%
1642         \def\NUMBER{\bgroup\@upn{####2}\egroup}%
1643         \def\NOTE{}}%
1644       \csname thmt@headstyle@#1\endcsname
1645       \theorem@separator
1646     ]
1647   }{%
1648     \item[\hskip\labelsep\theorem@headerfont%
1649       \def\NAME{\theorem@headerfont #####1}%
1650       \def\NUMBER{\bgroup\@upn{####2}\egroup}%
1651       \def\NOTE{\if=####3=\else\bgroup\thmt@space(####3)\egroup\fi}%
1652       \csname thmt@headstyle@#1\endcsname
1653       \theorem@separator
1654     ]
1655   }
1656 \else
1657   \newtheoremstyle{\thmt@style}{%
1658     \item[\hskip\labelsep\theorem@headerfont%
1659       \def\NAME{\the\thm@headfont #####1}%
1660       \def\NUMBER{\bgroup\@upn{####2}\egroup}%
1661       \def\NOTE{}}%
1662     #1%

```

```

1663     \theoremseparator
1664   ]
1665 }{%
1666   \item[\hskip\labelsep\theoremheaderfont%
1667     \def\NAME{\the\thm@headfont ####1}%
1668     \def\NUMBER{\bgroup\@upn{####2}\egroup}%
1669     \def\NOTE{\if=####3=\else\bgroup\thmt@space(####3)\egroup\fi}%
1670     #1%
1671     \theoremseparator
1672   ]
1673 }
1674 \fi
1675 }
1676
1677 \def\thmt@headstyle@margin{%
1678   \makebox[Opt][r]{\NUMBER\ }\NAME\NOTE
1679 }
1680 \def\thmt@headstyle@swapnumber{%
1681   \NUMBER\ \NAME\NOTE
1682 }
1683
1684
1685

```

## A.3 Generic tools

### A.3.1 A generalized argument parser

The main command provided by the package is `\parse{spec}`. *spec* consists of groups of commands. Each group should set up the command `\@parsecmd` which is then run. The important point is that `\@parsecmd` will pick up its arguments from the running text, not from the rest of *spec*. When it's done storing the arguments, `\@parsecmd` must call `\@parse` to continue with the next element of *spec*. The process terminates when we run out of *spec*.

Helper macros are provided for the three usual argument types: mandatory, optional, and flag.

```

1686
1687 \newtoks\@parsespec
1688 \def\parse@endquark{\parse@endquark}
1689 \newcommand\parse[1]{%
1690   \@parsespec{#1\parse@endquark}\@parse}
1691
1692 \newcommand\@parse{%
1693   \edef\p@tmp{\the\@parsespec}%
1694   \ifx\p@tmp\parse@endquark
1695     \expandafter\@gobble
1696   \else
1697     \typeout{parsespec remaining: \the\@parsespec}%
1698     \expandafter\@firstofone
1699   \fi}%
1700 \@parsepop
1701 }%
1702 }
1703 \def\@parsepop{%
1704   \expandafter\p@rsepop\the\@parsespec\@nil
1705   \@parsecmd
1706 }
1707 \def\p@rsepop#1#2\@nil{%
1708   #1%
1709   \@parsespec{#2}%
1710 }
1711

```

```

1712 \newcommand\parseOpt[4]{%
1713 %\parseOpt{openchar}{closechar}{yes}{no}
1714% \typeout{attempting #1#2...}%
1715 \def\@parsecmd{%
1716 \@ifnextchar#1{\@reallyparse}{#4\@parse}%
1717 }%
1718 \def\@reallyparse#1##1#2{%
1719 #3\@parse
1720 }%
1721 }
1722
1723 \newcommand\parseMand[1]{%
1724 %\parseMand{code}
1725 \def\@parsecmd##1{#1\@parse}%
1726 }
1727
1728 \newcommand\parseFlag[3]{%
1729 %\parseFlag{flagchar}{yes}{no}
1730 \def\@parsecmd{%
1731 \@ifnextchar#1{#2\expandafter\@parse\@gobble}{#3\@parse}%
1732 }%
1733 }

```

### A.3.2 Different counters sharing the same register

`\@counteralias{#1}{#2}` makes #1 a counter that uses #2's count register. This is useful for things like hyperref's `\autoref`, which otherwise can't distinguish theorems and definitions if they share a counter.

For detailed information, see Die TeXnische Komödie 3/2006.

`add \@elt{#1}` to `\cl@#2`. This differs from the kernel implementation insofar as we trail the `cl` lists until we find one that is empty or starts with `\@elt`.

```

1734 \def\aliasctr@f@llow#1#2\@nil#3{%
1735 \ifx#1\@elt
1736 \noexpand #3%
1737 \else
1738 \expandafter\aliasctr@f@llow#1\@elt\@nil{#1}%
1739 \fi
1740 }
1741
1742 \newcommand\aliasctr@follow[1]{%
1743 \expandafter\aliasctr@f@llow
1744 }
1745
1746 \renewcommand*\@addtoreset[2]{\bgroup
1747 \edef\aliasctr@truelist{\aliasctr@follow{#2}}%
1748 \let\@elt\relax
1749 \expandafter\@cons\aliasctr@truelist{#1}}%
1750 }
1751
1752 \renewcommand*\@removefromreset[2]{\bgroup
1753 \edef\aliasctr@truelist{\aliasctr@follow{#2}}%
1754 \expandafter\let\csname c@#1\endcsname\@removefromreset
1755 \def\@elt##1{%

```

This code has been adapted from David Carlisle's `remreset`. We load that here only to prevent it from being loaded again.

```

1750% FMi 2019-07-31 \@removefromreset is in the kernel these days
1751 \@ifundefined{\@removefromreset}{\RequirePackage{remreset}}{}
1752 \renewcommand*\@removefromreset[2]{\bgroup
1753 \edef\aliasctr@truelist{\aliasctr@follow{#2}}%
1754 \expandafter\let\csname c@#1\endcsname\@removefromreset
1755 \def\@elt##1{%

```



```

1756 \expandafter\ifx\csname c@##1\endcsname\@removefromreset
1757 \else
1758 \noexpand\@elt{##1}%
1759 \fi}%
1760 \expandafter\xdef\aliasctr@@truelist{%
1761 \aliasctr@@truelist}
1762 \egroup}

```

make #1 a counter that uses counter #2's count register.

```

1763 \newcommand\@counteralias[2]{%
1764 \def\@gletover##1##2{%
1765 \expandafter\global
1766 \expandafter\let\csname ##1\expandafter\endcsname
1767 \csname ##2\endcsname
1768 }%
1769 \@ifundefined{c@#2}{\@nocounterr{#2}}{%
1770 \expandafter\@ifdefinable\csname c@#1\endcsname{%

```

Four values make a counter foo:

- the count register accessed through `\c@foo`,
- the output macro `\thefoo`,
- the prefix macro `\p@foo`,
- the reset list `\cl@foo`.

hyperref adds `\theHfoo` in particular.

```

1771 \@@gletover{c@#1}{c@#2}%
1772 \@@gletover{the#1}{the#2}%

```

I don't see `\@counteralias` being called hundreds of times, let's just unconditionally create `\theHctr`-macros for hyperref.

```

1773 \@@gletover{theH#1}{theH#2}%

```

YkC: Compatibility with `cleveref`, copied from `cleveref`'s support for `aliascnt`. Here `\cref@resetby` requires its first argument to be the actual counter name, not a macro storing the name. Thanks to Willie Wong.

```

1774 \@ifpackageloaded{cleveref}{%
1775 \edef\aliasctr@temp{%
1776 \noexpand\cref@resetby{#2}{\noexpand\cref@result}}%
1777 \aliasctr@temp
1778 \ifx\cref@result\relax\else%
1779 \cref@addtoreset{#1}{\cref@result}%
1780 \fi
1781 }{}%
1782 \@@gletover{p@#1}{p@#2}%
1783 \expandafter\global
1784 \expandafter\def\csname cl@#1\expandafter\endcsname
1785 \expandafter{\csname cl@#2\endcsname}%

```

It is not necessary to save the value again: since we share a count register, we will pick up the restored value of the original counter.

```

1786 %\@addtoreset{#1}{@ckpt}%
1787 }%
1788 }%
1789 }}

```

### A.3.3 Tracking occurrences: none, one or many

Two macros are provided: `\setuniqmark` takes a single parameter, the name, which should be a string of letters. `\ifuniq` takes three parameters: a name, a true-part and a false-part. The true part is executed if and only if there was exactly one call to `\setuniqmark` with the given name during the previous  $\TeX$  run.

Example application: legal documents are often very strongly numbered. However, if a section has only a single paragraph, this paragraph is not numbered separately, this only occurs from two paragraphs onwards.

It's also possible to not-number the single theorem in your paper, but fall back to numbering when you add another one.

```

1790
1791 \DeclareOption{unq}{%
1792   \newwrite\uniq@channel
1793   \InputIfFileExists{\jobname.unq}{\relax}{\relax}%
1794   \immediate\openout\uniq@channel=\jobname.unq
1795   \AtEndDocument{%
1796     \immediate\closeout\uniq@channel%
1797   }
1798 }
1799 \DeclareOption{aux}{%
1800   \let\uniq@channel\@auxout
1801 }
1802

```

Call this with a name to set the corresponding uniqmark. The name must be suitable for `\csname`-constructs, i.e. fully expandable to a string of characters. If you use some counter values to generate this, it might be a good idea to try and use hyperref's `\theH...` macros, which have similar restrictions. You can check whether a particular `\setuniqmark` was called more than once during *the last run* with `\ifuniq`.

```

1803 \newcommand\setuniqmark[1]{%
1804   \expandafter\ifx\csname uniq@now@#1\endcsname\relax
1805     \global\@namedef{uniq@now@#1}{\uniq@ONE}%
1806   \else
1807     \expandafter\ifx\csname uniq@now@#1\endcsname\uniq@MANY
1808     \else
1809       \immediate\write\uniq@channel{%
1810         \string\uniq@setmany{#1}%
1811       }%
1812       \ifuniq{#1}{%
1813         \uniq@warnnotunique{#1}%
1814       }{}%
1815     \fi
1816     \global\@namedef{uniq@now@#1}{\uniq@MANY}%
1817   \fi
1818 }

```

Companion to `\setuniqmark`: if the uniqmark given in the first argument was called more than once, execute the second argument, otherwise execute the third argument. Note that no call to `\setuniqmark` for a particular uniqmark at all means that this uniqmark is unique.

This is a lazy version: we could always say false if we already had two calls to `\setuniqmark` this run, but we have to rerun for any `\ifuniq` prior to the first `\setuniqmark` anyway, so why bother?

```

1819 \newcommand\ifuniq[1]{%
1820   \expandafter\ifx\csname uniq@last@#1\endcsname\uniq@MANY
1821   \expandafter\@secondoftwo
1822   \else
1823     \expandafter\@firstoftwo
1824   \fi
1825 }

```

Two quarks to signal if we have seen an uniqmark more than once.

```

1826 \def\uniq@ONE{\uniq@ONE}
1827 \def\uniq@MANY{\uniq@MANY}

```

Flag: suggest a rerun?

```

1828 \newif\if@uniq@rerun

```

Helper macro: a call to this is written to the .aux file when we see an uniqmark for the second time. This sets the right information for the next run. It also checks on subsequent runs if the number of uniqmarks drops to less than two, so that we'll need a rerun.

```
1829 \def\uniq@setmany#1{%
1830   \global\@namedef{uniq@last@#1}{\uniq@MANY}%
1831   \AtEndDocument{%
1832     \uniq@warnifunique{#1}%
1833   }%
1834 }
```

Warning if something is unique now. This always warns if the setting for this run is not “many”, because it was generated by a setmany from the last run.

```
1835 \def\uniq@warnifunique#1{%
1836   \expandafter\ifx\csname uniq@now@#1\endcsname\uniq@MANY\else
1837     \PackageWarningNoLine{uniq}{%
1838       ‘#1’ is unique now.\MessageBreak
1839       Rerun LaTeX to pick up the change%
1840     }%
1841     \@uniq@reruntrue
1842   \fi
1843 }
```

Warning if we have a second uniqmark this run around. Since this is checked immediately, we could give the line of the second occurrence, but we do not do so for symmetry.

```
1844 \def\uniq@warnnotunique#1{%
1845   \PackageWarningNoLine{uniq}{%
1846     ‘#1’ is not unique anymore.\MessageBreak
1847     Rerun LaTeX to pick up the change%
1848   }%
1849   \@uniq@reruntrue
1850 }
```

Maybe advise a rerun (duh!). This is executed at the end of the second reading of the aux-file. If you manage to set uniqmarks after that (though I cannot imagine why), you might need reruns without being warned, so don't to that.

```
1851 \def\uniq@maybesuggestrerun{%
1852   \if@uniq@rerun
1853     \PackageWarningNoLine{uniq}{%
1854       Uniquenesses have changed. \MessageBreak
1855       Rerun LaTeX to pick up the change%
1856     }%
1857   \fi
1858 }
```

Make sure the check for rerun is pretty late in processing, so it can catch all of the uniqmarks (hopefully).

```
1859 \AtEndDocument{%
1860   \immediate\write\@auxout{\string\uniq@maybesuggestrerun}%
1861 }
1862 \ExecuteOptions{aux}
1863 \ProcessOptions\relax
```