

Thmtools Users' Guide

Dr. Ulrich M. Schwarz – ulmi@absatzen.de*
Yukai Chou – muzimuzhi@gmail.com

2020/07/16 v69

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

1 Thmtools for the impatient	2	3.4 Restatable – hints and caveats	15
1.1 Elementary definitions	2	A Thmtools for the morbidly curious	16
1.2 Frilly references	3	A.1 Core functionality	16
1.3 Styling theorems	4	A.1.1 The main package	16
1.3.1 Declaring new theoremstyles	5	A.1.2 Adding hooks to the relevant commands	17
1.4 Repeating theorems	6	A.1.3 The key-value interfaces	20
1.5 Lists of theorems	6	A.1.4 Lists of theorems	28
1.6 Extended arguments to theorem environments	8	A.1.5 Re-using environments	31
2 Thmtools for the extravagant	9	A.1.6 Restrictions	32
2.1 Understanding thmtools' extension mechanism	9	A.1.7 Fixing <code>autoref</code> and friends	36
2.2 Case in point: the <code>shaded</code> key	9	A.2 Glue code for different backends	38
2.3 Case in point: the <code>thmbox</code> key	11	A.2.1 <code>amsthm</code>	38
2.4 Case in point: the <code>mdframed</code> key	11	A.2.2 <code>beamer</code>	40
2.5 How thmtools finds your extensions	11	A.2.3 <code>ntheorem</code>	41
3 Thmtools for the completionist	13	A.3 Generic tools	43
3.1 Known keys to <code>\declaretheoremstyle</code>	13	A.3.1 A generalized argument parser	43
3.2 Known keys to <code>\declaretheorem</code>	14	A.3.2 Different counters sharing the same register	44
3.3 Known keys to in-document theorems	15	A.3.3 Tracking occurrences: none, one or many	45

*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 \LaTeX 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="Übung"]{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}
```

```
\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}
```

```
\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}
```

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

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

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

Couple 1. *Marc & Anne*

Singleton. *Me.*

Couple 2. *Buck & Britta*

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{
  % \nameref,%\nameref
  % n.b. usually \nameref is autoloading by hyperref
  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. *This is a theorem.*

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

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 value is `\color{white}` [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.

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,
  postheadsapce=1em,
  qed=\qedsymbol
]{mystyle}
\declaretheorem[style=mystyle]{styledtheorem}

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

Styledtheorem 1 (Euclid). For every prime p ... \square

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.

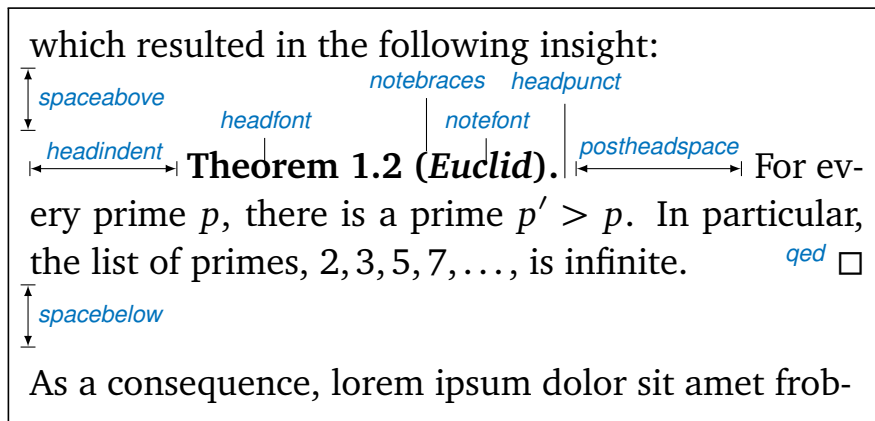


Figure 1.1: Settable parameters of a theorem style.

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, 45, 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`:

`\listoftheorems`

List of Theorems

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	4
1	BoxII	4
1	Boxtheorem L (Euclid) . . .	5
1	Boxtheorem M (Euclid) . .	5
1	Boxtheorem S (Euclid) . . .	5
1	Styledtheorem (Euclid) . .	5
1	Theorem (Euclid)	6
1	Theorem (Euclid)	6
1.1.1	TheoremS (Euclid)	6
3	Theorem (Keyed theorem)	8
3	Theorem (continuing from p. 8)	8
4	Lemma (Zorn)	32
5	Lemma	32
4	Lemma (Zorn)	32

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

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

List of Theorems

1	Theorem (Euclid)	2
	Euclid's Prime Theorem . .	3
1	Theorem	4
1	Theorem (Euclid)	6
1	Theorem (Euclid)	6
3	Theorem (Keyed theorem)	8
3	Theorem (continuing from p. 8)	8

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)	6
1	Theorem (Euclid)	6
3	Theorem (Keyed theorem)	8
3	Theorem (continuing from p. 8)	8

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.8). *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\shaderightshift{#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#3{%
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     \let\oldendproof=\endproof
43     \let\oldexample=\example
44     \let\oldendexample=\endexample
45     \RequirePackage[nothm]{thmbox}
46     \let\proof=\oldproof
47     \let\endproof=\oldendproof
48     \let\example=\oldexample
49     \let\endexample=\oldendexample
50     \def\thmt@theoremdefiner{\newboxtheorem[#1]}%
51   }{}%
52 }%
```

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.

```
53 \define@key{thmdef}{mdframed}[]{}{%
54   \thmt@trytwice{}{%
55     \RequirePackage{mdframed}%
56     \RequirePackage{thm-patch}%
57     \addtotheorempreheadhook[\thmt@envname]{%
58       \begin{mdframed}[#1]}%
59     \addtotheorempostfoothook[\thmt@envname]{\end{mdframed}}%
60   }%
61 }
```

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 will not work 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 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

```
62 \DeclareOption{debug}{%
63   \def\thmt@debug{\typeout}%
64 }
65 % common abbreviations and marker macros.
66 \let\@xa\expandafter
67 \let\@nx\noexpand
68 \def\thmt@debug{\@gobble}
69 \def\thmt@quark{\thmt@quark}
70 \newtoks\thmt@toks
71
72 \@for\thmt@opt:=lowercase,uppercase,anycase\do{%
73   \@xa\DeclareOption\@xa{\thmt@opt}{%
74     \@xa\PassOptionsToPackage\@xa{\CurrentOption}{thm-kv}%
75   }%
76 }
77
78 \ProcessOptions\relax
79
80 % a scratch counter, mostly for fake hyperlinks
81 \newcounter{thmt@dummyctr}%
82 \def\theHthmt@dummyctr{dummy.\arabic{thmt@dummyctr}}%
83 \def\thethmt@dummyctr{}%
84
85
86 \RequirePackage{thm-patch, thm-kv,
87   thm-autoref, thm-listof,
88   thm-restate}
89
90 % Glue code for the big players.
91 \@ifpackageloaded{amsthm}{%
92   \RequirePackage{thm-amsthm}
93 }{%
94   \AtBeginDocument{%
95     \@ifpackageloaded{amsthm}{%
96       \PackageWarningNoLine{thmtools}{%
97         amsthm loaded after thmtools
98       }{}%
99     }{}%
100 }
101 \@ifpackageloaded{ntheorem}{%
102   \RequirePackage{thm-ntheorem}
103 }{%
104   \AtBeginDocument{%
105     \@ifpackageloaded{ntheorem}{%
106       \PackageWarningNoLine{thmtools}{%
107         ntheorem loaded after thmtools
```

```

108     }{}%
109   }{}%
110 }
111 \@ifclassloaded{beamer}{%
112   \RequirePackage{thm-beamer}
113 }{}
114 \@ifclassloaded{llncs}{%
115   \RequirePackage{thm-llncs}
116 }{}

```

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 foothooks 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.

```

117 \RequirePackage{parseargs}
118
119 \newif\ifthmt@isstarred
120 \newif\ifthmt@hassibling
121 \newif\ifthmt@hasparent
122
123 \def\thmt@parsetheoremargs#1{%
124   \parse{%
125     {\parseOpt[]{\def\thmt@optarg{##1}}}{%
126       \let\thmt@shortoptarg\@empty
127       \let\thmt@optarg\@empty}}}%
128   {%
129     \def\thmt@local@preheadhook{}%
130     \def\thmt@local@postheadhook{}%
131     \def\thmt@local@prefoothook{}%
132     \def\thmt@local@postfoothook{}%
133     \thmt@local@preheadhook
134     \csname thmt@#1@preheadhook\endcsname
135     \thmt@generic@preheadhook
136     % change following to \@xa-orgy at some point?
137     % forex, might have keyvals involving commands.
138     %\protected@edef\tmp@args{%
139     % \ifx\@empty\thmt@optarg\else [{\thmt@optarg}]\fi
140     %}%
141     \ifx\@empty\thmt@optarg
142       \def\tmp@args{}%
143     \else
144       \@xa\def\@xa\tmp@args\@xa{\@xa[\@xa{\thmt@optarg}]}%
145     \fi
146     \csname thmt@original@#1\@xa\endcsname\tmp@args

```

```

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

```

```

208 \thmt@tmp
209 }%
210 }
211 \newcommand\thmt@providetheoremhooks[1]{%
212 \@namedef{thmt@#1@preheadhook}{}%
213 \@namedef{thmt@#1@postheadhook}{}%
214 \@namedef{thmt@#1@prefoothook}{}%
215 \@namedef{thmt@#1@postfoothook}{}%
216 \def\thmt@local@preheadhook{}%
217 \def\thmt@local@postheadhook{}%
218 \def\thmt@local@prefoothook{}%
219 \def\thmt@local@postfoothook{}%
220 }
221 \newcommand\thmt@addtheoremhook[1]{%
222 % this adds two command calls to the newly-defined theorem.
223 \@xa\let\csname thmt@original@#1\@xa\endcsname
224 \csname#1\endcsname
225 \@xa\renewcommand\csname #1\endcsname{%
226 \thmt@parsetheoremargs{#1}%
227 }%
228 \@xa\let\csname thmt@original@end#1\@xa\endcsname\csname end#1\endcsname
229 \@xa\def\csname end#1\endcsname{%
230 % these need to be in opposite order of headhooks.
231 \csname thmt@generic@prefoothook\endcsname
232 \csname thmt@#1@prefoothook\endcsname
233 \csname thmt@local@prefoothook\endcsname
234 \csname thmt@original@end#1\endcsname
235 \csname thmt@generic@postfoothook\endcsname
236 \csname thmt@#1@postfoothook\endcsname
237 \csname thmt@local@postfoothook\endcsname
238 }%
239 }
240 \newcommand\thmt@generic@preheadhook{\refstepcounter{thmt@dummysctr}}
241 \newcommand\thmt@generic@postheadhook{}
242 \newcommand\thmt@generic@prefoothook{}
243 \newcommand\thmt@generic@postfoothook{}
244
245 \def\thmt@local@preheadhook{}
246 \def\thmt@local@postheadhook{}
247 \def\thmt@local@prefoothook{}
248 \def\thmt@local@postfoothook{}
249
250
251 \providecommand\g@prependto@macro[2]{%
252 \begingroup
253 \toks@{\@xa{\@xa{#1}{#2}}}%
254 \def\tmp@a##1##2{##2##1}%
255 \@xa\@xa\@xa\gdef\@xa\@xa\@xa#1\@xa\@xa\@xa{\@xa\tmp@a\the\toks@}%
256 \endgroup
257 }
258
259 \newcommand\addtotheoremheadhook[1][generic]{%
260 \expandafter\g@addto@macro\csname thmt@#1@preheadhook\endcsname%
261 }
262 \newcommand\addtotheoremheadhook[1][generic]{%
263 \expandafter\g@addto@macro\csname thmt@#1@postheadhook\endcsname%
264 }
265
266 \newcommand\addtotheoremheadhook[1][generic]{%
267 \expandafter\g@prependto@macro\csname thmt@#1@prefoothook\endcsname%
268 }

```

```

269 \newcommand\addtotheorempostfoothook[1][generic]{%
270   \expandafter\g@prependto@macro\csname thmt@#1@postfoothook\endcsname%
271 }
272

```

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.

```

273 \ifx\proof\endproof\else% yup, that's a quaint way of doing it :)
274 % FIXME: this assumes proof has the syntax of theorems, which
275 % usually happens to be true (optarg overrides "Proof" string).
276 % FIXME: refactor into thmt@addtheoremhook, but we really don't want to
277 % call the generic-hook...
278 \let\thmt@original@proof=\proof
279 \renewcommand\proof{%
280   \thmt@parseproofargs%
281 }%
282 \def\thmt@parseproofargs{%
283   \parse{%
284     {\parseOpt[]{\def\thmt@optarg{##1}}{\let\thmt@optarg@empty}}%
285     {%
286       \thmt@proof@preheadhook
287       %\thmt@generic@preheadhook
288       \protected@edef\tmp@args{%
289         \ifx\@empty\thmt@optarg\else [\thmt@optarg]\fi
290       }%
291       \csname thmt@original@proof\@xa\endcsname\tmp@args
292       \thmt@proof@postheadhook
293       %\thmt@generic@postheadhook
294       \let\@parsecmd\@empty
295     }%
296   }%
297 }%
298
299 \let\thmt@original@endproof=\endproof
300 \def\endproof{%
301   % these need to be in opposite order of headhooks.
302   %\csname thmtgeneric@prefoothook\endcsname
303   \thmt@proof@prefoothook
304   \thmt@original@endproof
305   %\csname thmt@generic@postfoothook\endcsname
306   \thmt@proof@postfoothook
307 }%
308 \@namedef{thmt@proof@preheadhook}{}%
309 \@namedef{thmt@proof@postheadhook}{}%
310 \@namedef{thmt@proof@prefoothook}{}%
311 \@namedef{thmt@proof@postfoothook}{}%
312 \fi

```

A.1.3 The key-value interfaces

```

313
314 \let\@xa\expandafter
315 \let\@nx\noexpand
316
317 \DeclareOption{lowercase}{%
318   \PackageInfo{thm-kv}{Theorem names will be lowercased}%
319   \global\let\thmt@modifycase\MakeLowercase}
320
321 \DeclareOption{uppercase}{%
322   \PackageInfo{thm-kv}{Theorem names will be uppercased}%
323   \global\let\thmt@modifycase\MakeUppercase}

```

```

324
325 \DeclareOption{anycase}{%
326   \PackageInfo{thm-kv}{Theorem names will be unchanged}%
327   \global\let\thmt@modifycase\@empty}
328
329 \ExecuteOptions{uppercase}
330 \ProcessOptions\relax
331
332 \RequirePackage{keyval,kvsetkeys,thm-patch}
333
334 \long\def\thmt@kv@processor@default#1#2#3{%
335   \def\kvsu@fam{#1}% new
336   \@onelevel@sanitize\kvsu@fam% new
337   \def\kvsu@key{#2}% new
338   \@onelevel@sanitize\kvsu@key% new
339   \unless\ifcsname KV@#1@\kvsu@key\endcsname
340     \unless\ifcsname KVS@#1@handler\endcsname
341       \kv@error@unknownkey{#1}{\kvsu@key}%
342     \else
343       \csname KVS@#1@handler\endcsname{#2}{#3}%
344       % still using #2 #3 here is intentional: handler might
345       % be used for strange stuff like implementing key names
346       % that contain strange characters or other strange things.
347       \relax
348     \fi
349   \else
350     \ifx\kv@value\relax
351       \unless\ifcsname KV@#1@\kvsu@key @default\endcsname
352         \kv@error@novalue{#1}{\kvsu@key}%
353       \else
354         \csname KV@#1@\kvsu@key @default\endcsname
355         \relax
356       \fi
357     \else
358       \csname KV@#1@\kvsu@key\endcsname{#3}%
359     \fi
360   \fi
361 }
362
363 \@ifpackagelater{kvsetkeys}{2012/04/23}{%
364   \PackageInfo{thm-kv}{kvsetkeys patch (v1.16 or later)}%
365   \long\def\tmp@KVS@PD#1#2#3{%
366     \def\kv@fam{#1}%
367     \unless\ifcsname KV@#1@#2\endcsname
368       \unless\ifcsname KVS@#1@handler\endcsname
369         \kv@error@unknownkey{#1}{#2}%
370       \else
371         \kv@handled@true
372         \csname KVS@#1@handler\endcsname{#2}{#3}\relax
373         \ifkv@handled@ \else
374           \kv@error@unknownkey{#1}{#2}%
375         \fi
376       \fi
377     \else
378       \ifx\kv@value\relax
379         \unless\ifcsname KV@#1@#2@default\endcsname
380           \kv@error@novalue{#1}{#2}%
381         \else
382           \csname KV@#1@#2@default\endcsname \relax
383         \fi
384       \else

```

```

385     \csname KV@#1@#2\endcsname {#3}%
386   \fi
387 \fi
388 }%
389 \ifx\tmp@KVS@PD\KVS@ProcessorDefault
390   \let\KVS@ProcessorDefault\thmt@kv@processor@default
391   \def\kv@processor@default#1#2{%
392     \begingroup
393       \csname @safe@activetrue\endcsname
394       \@xa\let\csname ifinicsname\@xa\endcsname\csname iftrue\endcsname
395       \edef\KVS@temp{\endgroup
396 % 2019/12/22 removed dependency on etexcmds package
397         \noexpand\KVS@ProcessorDefault{#1}{\unexpanded{#2}}}%
398     }%
399     \KVS@temp
400   }%
401 \else
402   \PackageError{thm-kv}{kvsetkeys patch failed}{Try kvsetkeys v1.16 or earlier}
403 \fi
404 }{\@ifpackagelater{kvsetkeys}{2011/04/06}{%
405   % Patch has disappeared somewhere... thanksalot.
406   \PackageInfo{thm-kv}{kvsetkeys patch (v1.13 or later)}
407   \long\def\tmp@KVS@PD#1#2#3{% no non-etex-support here...
408     \unless\ifcsname KV@#1@#2\endcsname
409     \unless\ifcsname KVS@#1@handler\endcsname
410       \kv@error@unknownkey{#1}{#2}%
411     \else
412       \csname KVS@#1@handler\endcsname{#2}{#3}%
413       \relax
414     \fi
415   \else
416     \ifx\kv@value\relax
417       \unless\ifcsname KV@#1@#2@default\endcsname
418         \kv@error@novalue{#1}{#2}%
419       \else
420         \csname KV@#1@#2@default\endcsname
421         \relax
422       \fi
423     \else
424       \csname KV@#1@#2\endcsname{#3}%
425     \fi
426   \fi
427 }%
428 \ifx\tmp@KVS@PD\KVS@ProcessorDefault
429   \let\KVS@ProcessorDefault\thmt@kv@processor@default
430   \def\kv@processor@default#1#2{%
431     \begingroup
432       \csname @safe@activetrue\endcsname
433       \let\ifinicsname\iftrue
434       \edef\KVS@temp{\endgroup
435         \noexpand\KVS@ProcessorDefault{#1}{\unexpanded{#2}}}%
436     }%
437     \KVS@temp
438   }
439 \else
440   \PackageError{thm-kv}{kvsetkeys patch failed, try kvsetkeys v1.13 or earlier}
441 \fi
442 }{%
443   \RequirePackage{etex}
444   \PackageInfo{thm-kv}{kvsetkeys patch applied (pre-1.13)}%
445   \let\kv@processor@default\thmt@kv@processor@default

```



```

446 }}
447
448 % useful key handler defaults.
449 \newcommand\thmt@mkignoringkeyhandler[1]{%
450   \kv@set@family@handler{#1}{%
451     \thmt@debug{Key '##1' with value '##2' ignored by #1.}%
452   }%
453 }
454 \newcommand\thmt@mkextendingkeyhandler[3]{%
455   % #1: family
456   % #2: prefix for file
457   % #3: key hint for error
458   \kv@set@family@handler{#1}{%
459     \thmt@selfextendingkeyhandler{#1}{#2}{#3}%
460     {##1}{##2}%
461   }%
462 }
463
464 \newcommand\thmt@selfextendingkeyhandler[5]{%
465   % #1: family
466   % #2: prefix for file
467   % #3: key hint for error
468   % #4: actual key
469   % #5: actual value
470   \IfFileExists{#2-#4.sty}{%
471     \PackageInfo{thmtools}%
472     {Automatically pulling in '#2-#4'}%
473     \RequirePackage{#2-#4}%
474     \ifcsname KV@#1@#4\endcsname
475     \csname KV@#1@#4\endcsname{#5}%
476   \else
477     \PackageError{thmtools}%
478     {#3 '#4' not known}
479     {I don't know what that key does.\MessageBreak
480     I've even loaded the file '#2-#4.sty', but that didn't help.
481     }%
482   \fi
483 }{%
484   \PackageError{thmtools}%
485   {#3 '#4' not known}
486   {I don't know what that key does by myself,\MessageBreak
487   and no file '#2-#4.sty' to tell me seems to exist.
488   }%
489 }%
490 }
491
492
493 \newif\if@thmt@firstkeyset
494
495 % many keys are evaluated twice, because we don't know
496 % if they make sense before or after, or both.
497 \def\thmt@trytwice{%
498   \if@thmt@firstkeyset
499     \@xa\@firstoftwo
500   \else
501     \@xa\@secondoftwo
502   \fi
503 }
504
505 \@for\tmp@keyname:=parent,numberwithin,within\do{%
506 \define@key{thmdef}{\tmp@keyname}{\thmt@trytwice{\thmt@setparent{#1}}{}}}%

```

```

507 }
508
509 \@for\tmp@keyname:=sibling,numberlike,sharenumber\do{%
510 \define@key{thmdef}{\tmp@keyname}{\thmt@trytwice{\thmt@setsibling{#1}}{}}%
511 }
512
513 \@for\tmp@keyname:=title,name,heading\do{%
514 \define@key{thmdef}{\tmp@keyname}{\thmt@trytwice{\thmt@setthmname{#1}}{}}%
515 }
516
517 \@for\tmp@keyname:=unnumbered,starred\do{%
518 \define@key{thmdef}{\tmp@keyname}[]{\thmt@trytwice{\thmt@isnumberedfalse}{}}%
519 }
520
521 \def\thmt@YES{yes}
522 \def\thmt@NO{no}
523 \def\thmt@UNIQUE{unless unique}
524 \define@key{thmdef}{numbered}[\thmt@YES]{
525   \def\thmt@tmp{#1}%
526   \thmt@trytwice{%
527     \ifx\thmt@tmp\thmt@YES
528       \thmt@isnumberedtrue
529     \else\ifx\thmt@tmp\thmt@NO
530       \thmt@isnumberedfalse
531     \else\ifx\thmt@tmp\thmt@UNIQUE
532       \RequirePackage[unq]{unique}
533       \ifuniqu{\thmt@envname}{%
534         \thmt@isnumberedfalse
535       }{%
536         \thmt@isnumberedtrue
537       }%
538     \else
539       \PackageError{thmtools}{Unknown value ‘#1’ to key numbered}{}%
540     \fi\fi\fi
541   }{% trytwice: after definition
542     \ifx\thmt@tmp\thmt@UNIQUE
543       \addtotheorempreheadhook[\thmt@envname]{\setuniqmark{\thmt@envname}}%
544       \addtotheorempreheadhook[\thmt@envname]{\def\thmt@dummysctrautorefname{\thmt@thmname\
545     \fi
546   }%
547 }
548
549
550 \define@key{thmdef}{preheadhook}{\thmt@trytwice{}{\addtotheorempreheadhook[\thmt@envname]{}}%
551 \define@key{thmdef}{postheadhook}{\thmt@trytwice{}{\addtotheorempostheadhook[\thmt@envname]{}}%
552 \define@key{thmdef}{prefoothook}{\thmt@trytwice{}{\addtotheoremprefoothook[\thmt@envname]{}}%
553 \define@key{thmdef}{postfoothook}{\thmt@trytwice{}{\addtotheorempostfoothook[\thmt@envname]{}}%
554
555 \define@key{thmdef}{style}{\thmt@trytwice{\thmt@setstyle{#1}}{}}
556
557 % ugly hack: style needs to be evaluated first so its keys
558 % are not overridden by explicit other settings
559 \define@key{thmdef0}{style}{%
560   \ifcsname thmt@style #1@defaultkeys\endcsname
561     \thmt@toks{\kvsetkeys{thmdef}}%
562     \@xa\@xa\@xa\the\@xa\@xa\thmt@toks\@xa\@xa\@xa{%
563       \csname thmt@style #1@defaultkeys\endcsname}%
564   \fi
565 }
566 \thmt@mkignoringkeyhandler{thmdef0}
567

```

```

568 % fallback definition.
569 % actually, only the kernel does not provide \theoremstyle.
570 % is this one worth having glue code for the theorem package?
571 \def\thmt@setstyle#1{%
572   \PackageWarning{thm-kv}{%
573     Your backend doesn't have a '\string\theoremstyle' command.
574   }%
575 }
576
577 \ifcsname theoremstyle\endcsname
578   \let\thmt@originalthmstyle\theoremstyle
579   \def\thmt@outerstyle{plain}
580   \renewcommand\theoremstyle[1]{%
581     \def\thmt@outerstyle{#1}%
582     \thmt@originalthmstyle{#1}%
583   }
584   \def\thmt@setstyle#1{%
585     \thmt@originalthmstyle{#1}%
586   }
587   \g@addto@macro\thmt@newtheorem@postdefinition{%
588     \thmt@originalthmstyle{\thmt@outerstyle}%
589   }
590 \fi
591
592 \newif\ifthmt@isnumbered
593 \newcommand\thmt@setparent[1]{%
594   \def\thmt@parent{#1}%
595 }
596 \newcommand\thmt@setsibling{%
597   \def\thmt@sibling
598 }
599 \newcommand\thmt@setthmname{%
600   \def\thmt@thmname
601 }
602
603 \thmt@mkextendingkeyhandler{thmdef}{thmdef}{\string\declaretheorem\space key}
604
605 \let\thmt@newtheorem\newtheorem
606
607 \newcommand\declaretheorem[2][]{%
608   % why was that here?
609   %\let\thmt@theoremdefiner\thmt@original@newtheorem
610   \def\thmt@envname{#2}%
611   \thmt@setthmname{\thmt@modifycase #2}%
612   \thmt@setparent{}%
613   \thmt@setsibling{}%
614   \thmt@isnumberedtrue%
615   \@thmt@firstkeysettrue%
616   \kvsetkeys{thmdef0}{#1}%
617   \kvsetkeys{thmdef}{#1}%
618   \protected@edef\thmt@tmp{%
619     \@nx\thmt@newtheorem
620     \ifthmt@isnumbered\else *\fi
621     {#2}%
622     \ifx\thmt@sibling\@empty\else [\thmt@sibling]\fi
623     {\thmt@thmname}%
624     \ifx\thmt@parent\@empty\else [\thmt@parent]\fi
625     \relax% added so we can delimited-read everything later
626     % (recall newtheorem is patched)
627   }%\show\thmt@tmp
628   \thmt@tmp

```

```

629 % uniquely ugly kludge: some keys make only sense
630 % afterwards.
631 % and it gets kludgier: again, the default-inherited
632 % keys need to have a go at it.
633 \@thmt@firstkeysetfalse%
634 \kvsetkeys{thmdef0}{#1}%
635 \kvsetkeys{thmdef}{#1}%
636 }
637 \@onlypreamble\declaretheorem
638
639 \providecommand\thmt@quark{\thmt@quark}
640
641 % in-document keyval, i.e. \begin{theorem}[key=val,key=val]
642
643 \thmt@mkextendingkeyhandler{thmuse}{thmuse}{\thmt@envname\space optarg key}
644
645 \addtotheorempreheadhook{%
646   \ifx\thmt@optarg\@empty\else
647     \@xa\thmt@garbleoptarg\@xa{\thmt@optarg}\fi
648 }%
649
650 \newif\ifthmt@thmuse@iskv
651
652 \providecommand\thmt@garbleoptarg[1]{%
653   \thmt@thmuse@iskvfalse
654   \def\thmt@newoptarg{\@gobble}%
655   \def\thmt@newoptargextra{}%
656   \let\thmt@shortoptarg\@empty
657   \def\thmt@warn@unusedkeys{}%
658   \@for\thmt@fam:=\thmt@thmuse@families\do{%
659     \kvsetkeys{\thmt@fam}{#1}%
660   }%
661   \ifthmt@thmuse@iskv
662     \protected@edef\thmt@optarg{%
663       \@xa\thmt@newoptarg
664       \thmt@newoptargextra\@empty
665     }%
666     \ifx\thmt@shortoptarg\@empty
667       \protected@edef\thmt@shortoptarg{\thmt@newoptarg\@empty}%
668     \fi
669     \thmt@warn@unusedkeys
670   \else
671     \def\thmt@optarg{#1}%
672     \def\thmt@shortoptarg{#1}%
673   \fi
674 }
675 \def\thmt@splitopt#1=#2\thmt@quark{%
676   \def\thmt@tmpkey{#1}%
677   \ifx\thmt@tmpkey\@empty
678     \def\thmt@tmpkey{\thmt@quark}%
679   \fi
680   \@onelevel@sanitize\thmt@tmpkey
681 }
682
683 \def\thmt@thmuse@families{thm@track@keys}
684
685 \kv@set@family@handler{thm@track@keys}{%
686   \@onelevel@sanitize\kv@key
687   \@namedef{thmt@unusedkey@\kv@key}{%
688     \PackageWarning{thmtools}{Unused key ‘#1’}%
689   }%

```

```

690 \@xa\g@addto@macro\@xa\thmt@warn@unusedkeys\@xa{%
691 \csname thmt@unusedkey@\kv@key\endcsname
692 }
693 }
694
695 % key, code.
696 \def\thmt@define@thmuse@key#1#2{%
697 \g@addto@macro\thmt@thmuse@families{,#1}%
698 \define@key{#1}{#1}{\thmt@thmuse@iskvtrue
699 \@namedef{thmt@unusedkey@#1}{}}%
700 #2}%
701 \thmt@mkignoringkeyhandler{#1}%
702 }
703
704 \thmt@define@thmuse@key{label}{%
705 \addtotheoremposttheadhook[local]{\label{#1}}%
706 }
707 \thmt@define@thmuse@key{name}{%
708 \thmt@setnewoptarg #1\@iden%
709 }
710 \newcommand\thmt@setnewoptarg[1][]{%
711 \def\thmt@shortoptarg{#1}\thmt@setnewlongoptarg
712 }
713 \def\thmt@setnewlongoptarg #1\@iden{%
714 \def\thmt@newoptarg{#1\@iden}}
715
716 \providecommand\thmt@suspendcounter[2]{%
717 \@xa\protected@edef\csname the#1\endcsname{#2}%
718 \@xa\let\csname c@#1\endcsname\c@thmt@dummysctr
719 }
720
721 \providecommand\thmcontinues[1]{%
722 \ifcsname hyperref\endcsname
723 \hyperref[#1]{continuing}
724 \else
725 continuing
726 \fi
727 from p.\,\pageref{#1}%
728 }
729
730 \thmt@define@thmuse@key{continues}{%
731 \thmt@suspendcounter{\thmt@envname}{\thmt@trivialref{#1}{??}}%
732 \g@addto@macro\thmt@newoptarg{{, }}%
733 \thmcontinues{#1}%
734 \@iden}%
735 }
736
737

```

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.

```

738 \def\thmt@declaretheoremstyle@setup{}
739 \def\thmt@declaretheoremstyle#1{%
740 \PackageWarning{thmtools}{Your backend doesn't allow styling theorems}{}
741 }
742 \newcommand\declaretheoremstyle[2][]{%
743 \def\thmt@style{#2}%
744 \@xa\def\csname thmt@style \thmt@style @defaultkeys\endcsname{%
745 \thmt@declaretheoremstyle@setup
746 \kvsetkeys{thmstyle}{#1}%
747 \thmt@declaretheoremstyle{#2}%

```

```

748 }
749 \@onlypreamble\declaretheoremstyle
750
751 \kv@set@family@handler{thmstyle}{%
752   \@onelevel@sanitize\kv@value
753   \@onelevel@sanitize\kv@key
754   \PackageInfo{thmtools}{%
755     Key ‘\kv@key’ (with value ‘\kv@value’)\MessageBreak
756     is not a known style key.\MessageBreak
757     Will pass this to every \string\declaretheorem\MessageBreak
758     that uses ‘style=\thmt@style’%
759   }%
760   \ifx\kv@value\relax% no value given, don’t pass on {}!
761   \@xa\g@addto@macro\csname thmt@style \thmt@style @defaultkeys\endcsname{%
762     #1,%
763   }%
764   \else
765   \@xa\g@addto@macro\csname thmt@style \thmt@style @defaultkeys\endcsname{%
766     #1={#2},%
767   }%
768   \fi
769 }

```

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.

```

770 \let\@xa=\expandafter
771 \let\@nx=\noexpand
772 \RequirePackage{thm-patch,keyval,kvsetkeys}
773
774 \def\thmtlo@oldchapter{0}%

```

```

775 \newcommand\thmtlo@chaptervspacehack{}
776 \ifcsname c@chapter\endcsname
777   \ifx\c@chapter\relax\else
778     \def\thmtlo@chaptervspacehack{%
779       \ifnum \value{chapter}=\thmtlo@oldchapter\relax\else
780         % new chapter, add vspace to loe.
781         \addtocontents{loe}{\protect\addvspace{10\p}}}%
782       \xdef\thmtlo@oldchapter{\arabic{chapter}}}%
783     \fi
784   }%
785 \fi
786 \fi
787
788
789 \providecommand\listtheoremname{List of Theorems}
790 \newcommand\listoftheorems[1][]{%
791   %% much hacking here to pick up the definition from the class
792   %% without oodles of conditionals.
793   \bgroup
794   \setlisttheoremstyle{#1}%
795   \let\listfigurename\listtheoremname
796   \def\contentsline##1{%
797     \csname thmt@contentsline@##1\endcsname{##1}%
798   }%
799   \@for\thmt@envname:=\thmt@allenvs\do{%
800     \xa\protected@edef\csname l@thmt@envname\endcsname{% CHECK: why p@edef?
801       \nx\@dottedtocline{1}{1.5em}{\nx\thmt@listnumwidth}%
802     }%
803   }%
804   \let\thref@starttoc\@starttoc
805   \def\@starttoc##1{\thref@starttoc{loe}}%
806   % new hack: to allow multiple calls, we defer the opening of the
807   % loe file to AtEndDocument time. This is before the aux file is
808   % read back again, that is early enough.
809   % TODO: is it? crosscheck include/includeonly!
810   \@fileswfalse
811   \AtEndDocument{%
812     \if@filesw
813       \ifundefined{tf@loe}{%
814         \expandafter\newwrite\csname tf@loe\endcsname
815         \immediate\openout \csname tf@loe\endcsname \jobname.loe\relax
816       }{}%
817     \fi
818   }%
819   \expandafter
820   \listoffigures
821   \egroup
822 }
823
824 \newcommand\setlisttheoremstyle[1]{%
825   \kvsetkeys{thmt-listof}{#1}%
826 }
827 \define@key{thmt-listof}{numwidth}{\def\thmt@listnumwidth{#1}}
828 \define@key{thmt-listof}{ignore}{\thmt@allenvs}{\ignoretheorems{#1}}
829 \define@key{thmt-listof}{onlynamed}{\thmt@allenvs}{\onlynamedtheorems{#1}}
830 \define@key{thmt-listof}{show}{\thmt@allenvs}{\showtheorems{#1}}
831 \define@key{thmt-listof}{ignoreall}[true]{\ignoretheorems{\thmt@allenvs}}
832 \define@key{thmt-listof}{showall}[true]{\showtheorems{\thmt@allenvs}}
833 % FMi 2019-09-31 allow local title
834 \define@key{thmt-listof}{title}{\def\listtheoremname{#1}}
835 % -- FMi

```



```

836
837 \providecommand\thmt@listnumwidth{2.3em}
838
839 \providecommand\thmt@formatoptarg[1]{ (#1)}
840
841 \newcommand\thmt@mklistcmd{%
842   \@xa\protected@edef\csname ll@\thmt@envname\endcsname{% CHECK: why p@edef?
843     \@nx\@dottedtocline{1}{1.5em}{\@nx\thmt@listnumwidth}%
844   }%
845   \ifthmt@isstarred
846     \@xa\def\csname ll@\thmt@envname\endcsname{%
847       \protect\numberline{\protect\let\protect\autodot\protect\@empty}%
848       \thmt@thmname
849       \ifx\@empty\thmt@shortoptarg\else\protect\thmt@formatoptarg{\thmt@shortoptarg}\fi
850     }%
851   \else
852     \@xa\def\csname ll@\thmt@envname\endcsname{%
853       \protect\numberline{\csname the\thmt@envname\endcsname}%
854       \thmt@thmname
855       \ifx\@empty\thmt@shortoptarg\else\protect\thmt@formatoptarg{\thmt@shortoptarg}\fi
856     }%
857   \fi
858   \@xa\gdef\csname thmt@contentsline@\thmt@envname\endcsname{%
859     \thmt@contentslineShow% default:show
860   }%
861 }
862 \def\thmt@allenvs{\@gobble}
863 \newcommand\thmt@recordenvname{%
864   \edef\thmt@allenvs{\thmt@allenvs,\thmt@envname}%
865 }
866 \g@addto@macro\thmt@newtheorem@predefinition{%
867   \thmt@mklistcmd
868   \thmt@recordenvname
869 }
870
871 \addtotheorempostheadhook{%
872   \thmtlo@chaptervspacehack
873   \addcontentsline{loe}{\thmt@envname}{%
874     \csname ll@\thmt@envname\endcsname
875   }%
876 }
877
878 \newcommand\showtheorems[1]{%
879   \@for\thmt@thm:=#1\do{%
880     \typeout{showing \thmt@thm}%
881     \@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
882       =\thmt@contentslineShow
883   }%
884 }
885
886 \newcommand\ignoretheorems[1]{%
887   \@for\thmt@thm:=#1\do{%
888     \@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
889       =\thmt@contentslineIgnore
890   }%
891 }
892 \newcommand\onlynamedtheorems[1]{%
893   \@for\thmt@thm:=#1\do{%
894     \global\@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
895       =\thmt@contentslineIfNamed
896   }%

```

```

897 }
898
899 \AtBeginDocument{%
900 \ifpackageloaded{hyperref}{%
901   \let\thmt@hygobble\@gobble
902 }{%
903   \let\thmt@hygobble\@empty
904 }
905 \let\thmt@contentsline\contentsline
906 }
907
908 \def\thmt@contentslineIgnore#1#2#3{%
909   \thmt@hygobble
910 }
911 \def\thmt@contentslineShow{%
912   \thmt@contentsline
913 }
914
915 \def\thmt@contentslineIfNamed#1#2#3{%
916   \thmt@ifhasoptname #2\thmtformatoptarg\@nil{%
917     \thmt@contentslineShow{#1}{#2}{#3}%
918   }{%
919     \thmt@contentslineIgnore{#1}{#2}{#3}%
920     %\thmt@contentsline{#1}{#2}{#3}%
921   }
922 }
923
924 \def\thmt@ifhasoptname #1\thmtformatoptarg#2\@nil{%
925   \ifx\@nil#2\@nil
926     \@xa\@secondoftwo
927   \else
928     \@xa\@firstoftwo
929   \fi
930 }

```

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  $XS$  is upper-bounded,
     $XS$  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 \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 \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.

```
931 \RequirePackage{thmtools}
932 \let\@xa\expandafter
933 \let\@nx\noexpand
934 \ifundefined{c@thmt@dummyctr}{%
935   \newcounter{thmt@dummyctr}%
936 }{}
937 \gdef\theHthmt@dummyctr{dummy.\arabic{thmt@dummyctr}}%
938 \gdef\thethmt@dummyctr{}%
939 \long\def\thmt@collect@body#1#2\end#3{%
940   \@xa\thmt@toks\@xa{\the\thmt@toks #2}%
941   \def\thmttmpa{#3}%\def\thmttmpb{restatable}%
942   \ifx\thmttmpa\@currenvir\thmttmpb
943     \@xa\@firstoftwo% this is the end of the environment.
944   \else
945     \@xa\@secondoftwo% go on collecting
946     \fi{% this is the end, my friend, drop the \end.
947     % and call #1 with the collected body.
948     \@xa#1\@xa{\the\thmt@toks}%
949   }{% go on collecting
950     \@xa\thmt@toks\@xa{\the\thmt@toks\end{#3}}%
951     \thmt@collect@body{#1}%
952   }%
953 }
```

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

```
954 \def\thmt@trivialref#1#2{%
955   \ifcsname r@#1\endcsname
956     \@xa\@xa\@xa\thmt@trivi@lrf\csname r@#1\endcsname\relax\@nil
957   \else #2\fi
```

```

958 }
959 \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 \thctr must not be called from within that group, since it needs the original \@arabic et al.

```

960 \def\thmt@innercounters{%
961   equation}
962 \def\thmt@counterformatters{%
963   @alph,@Alph,@arabic,@roman,@Roman,@fnsymbol}
964
965 \@for\thmt@displ:=\thmt@counterformatters\do{%
966   \@xa\let\csname thmt@\thmt@displ\@xa\endcsname\csname \thmt@displ\endcsname
967 }%
968 \def\thmt@sanitizethe#1{%
969   \@for\thmt@displ:=\thmt@counterformatters\do{%
970     \@xa\protected@edef\csname\thmt@displ\endcsname##1{%
971       \@nx\ifx\@xa\@nx\csname c@#1\endcsname ##1%
972       \@xa\protect\csname \thmt@displ\endcsname{##1}%
973       \@nx\else
974       \@nx\csname thmt@\thmt@displ\endcsname{##1}%
975       \@nx\fi
976     }%
977   }%
978   \expandafter\protected@edef\csname the#1\endcsname{\csname the#1\endcsname}%
979   \ifcsname theH#1\endcsname
980     \expandafter\protected@edef\csname theH#1\endcsname{\csname theH#1\endcsname}%
981   \fi
982 }
983
984 \def\thmt@rst@storecounters#1{%
985   \bgroup
986     % ugly hack: save chapter,...subsection numbers
987     % for equation numbers.
988     %\refstepcounter{thmt@dummyctr}% why is this here?
989     %% temporarily disabled, broke autorefname.
990     \def\@currentlabel{}%
991     \@for\thmt@ctr:=\thmt@innercounters\do{%
992       \thmt@sanitizethe{\thmt@ctr}%
993       \protected@edef\@currentlabel{%
994         \@currentlabel
995         \protect\def\@xa\protect\csname the\thmt@ctr\endcsname{%
996           \csname the\thmt@ctr\endcsname}%
997         \ifcsname theH\thmt@ctr\endcsname
998           \protect\def\@xa\protect\csname theH\thmt@ctr\endcsname{%
999             (restate \protect\theHthmt@dummyctr)\csname theH\thmt@ctr\endcsname}%
1000         \fi
1001         \protect\setcounter{\thmt@ctr}{\number\csname c@\thmt@ctr\endcsname}%
1002       }%
1003     }%
1004     \label{thmt@@#1@data}%
1005   \egroup
1006 }%

```

Now, the main business.

```

1007 \newif\ifthmt@thisistheone
1008 \newenvironment{thmt@restatable}[3][]{%
1009   \thmt@toks{}}% will hold body
1010 %
1011 \stepcounter{thmt@dummyctr}% used for data storage label.
1012 %
1013 \long\def\thmrst@store##1{%
1014   \@xa\gdef\csname #3\endcsname{%
1015     \ifstar{%
1016       \thmt@thisistheonefalse\csname thmt@stored@#3\endcsname
1017     }{%
1018       \thmt@thisistheonetrue\csname thmt@stored@#3\endcsname
1019     }%
1020   }%
1021   \@xa\long\@xa\gdef\csname thmt@stored@#3\@xa\endcsname\@xa{%
1022     \begingroup
1023     \ifthmt@thisistheone
1024       % these are the valid numbers, store them for the other
1025       % occasions.
1026       \thmt@rst@storecounters{#3}%
1027     \else
1028       % this one should use other numbers...
1029       % first, fake the theorem number.
1030       \@xa\protected@edef\csname the#2\endcsname{%
1031         \thmt@trivialref{thmt@@#3}{??}}%
1032       % if the number wasn't there, have a "re-run to get labels right"
1033       % warning.
1034       \ifcsname r@thmt@@#3\endcsname\else
1035         \G@refundefinedtrue
1036       \fi
1037       % prevent stepcountering the theorem number,
1038       % but still, have some number for hyperref, just in case.
1039       \@xa\let\csname c@#2\endcsname=c@thmt@dummyctr
1040       \@xa\let\csname theH#2\endcsname=theHthmt@dummyctr
1041       % disable labeling.
1042       \let\label=\thmt@gobble@label
1043       \let\ltx@label=\@gobble% amsmath needs this
1044       % We shall need to restore the counters at the end
1045       % of the environment, so we get
1046       % (4.2) [(3.1 from restate)] (4.3)
1047       \def\thmt@restorecounters{%
1048         \@for\thmt@ctr:=\thmt@innercounters\do{%
1049           \protected@edef\thmt@restorecounters{%
1050             \thmt@restorecounters
1051             \protect\setcounter{\thmt@ctr}{\arabic{\thmt@ctr}}}%
1052         }%
1053       }%
1054       % pull the new semi-static definition of \theequation et al.
1055       % from the aux file.
1056       \thmt@trivialref{thmt@@#3@data}{}%
1057     \fi
1058     % call the proper begin-env code, possibly with optional argument
1059     % (omit if stored via key-val)
1060     \ifthmt@restatethis
1061       \thmt@restatethisfalse
1062     \else
1063       \csname #2\@xa\endcsname\ifx\@nx#1\@nx\else[#1]\fi
1064     \fi
1065     \ifthmt@thisistheone
1066       % store a label so we can pick up the number later.
1067       \label{thmt@@#3}%

```

```

1068 \fi
1069 % this will be the collected body.
1070 ##1%
1071 \csname end#2\endcsname
1072 % if we faked the counter values, restore originals now.
1073 \ifthmt@thisistheone\else\thmt@restorecounters\fi
1074 \endgroup
1075 }% thmt@stored@#3
1076 % in either case, now call the just-created macro,
1077 \csname #3\@xa\endcsname\ifthmt@thisistheone\else*\fi
1078 % and artificially close the current environment.
1079 \@xa\end\@xa{\@currenvir}
1080 }% thm@rst@store
1081 \thmt@collect@body\thm@rst@store
1082 }{%
1083 %% now empty, just used as a marker.
1084 }
1085
1086 \let\thmt@gobble@label\@gobble
1087 % cleveref extends syntax of \label to \label[...]{...}
1088 \AtBeginDocument{
1089 \ifpackageloaded{cleveref}{
1090 \renewcommand*\thmt@gobble@label[2][]{}
1091 }{}
1092 }
1093
1094 \newenvironment{restatable}{%
1095 \thmt@thisistheonetrue\thmt@restatable
1096 }{%
1097 \endthmt@restatable
1098 }
1099 \newenvironment{restatable*}{%
1100 \thmt@thisistheonefalse\thmt@restatable
1101 }{%
1102 \endthmt@restatable
1103 }
1104
1105 %%% support for keyval-style: restate=foobar
1106 \protected@edef\thmt@thmuse@families{%
1107 \thmt@thmuse@families%
1108 ,restate phase 1%
1109 ,restate phase 2%
1110 }
1111 \newcommand\thmt@splitrestateargs[1][]{%
1112 \g@addto@macro\thmt@storedoptargs{,#1}%
1113 \def\tmp@a##1\@{\def\thmt@storename{##1}}%
1114 \tmp@a
1115 }
1116
1117 \newif\ifthmt@restatethis
1118 \define@key{restate phase 1}{restate}{%
1119 \thmt@thmuse@iskvtrue
1120 \def\thmt@storedoptargs{}% discard the first time around
1121 \thmt@splitrestateargs #1\@
1122 \def\thmt@storedoptargs{}% discard the first time around
1123 %\def\thmt@storename{#1}%
1124 \thmt@debug{we will restate as '\thmt@storename' with more args
1125 '\thmt@storedoptargs'}%
1126 \@namedef{thmt@unusedkey@restate}{}%
1127 % spurious "unused key" fixes itself once we are after tracknames...
1128 \thmt@restatethistrue

```

```

1129 \protected@edef\tmp@a{%
1130   \@nx\thmt@thisistheonetrue
1131   \@nx\def\@nx\@currentvir{\thmt@envname}%
1132   \@nx\@xa\@nx\thmt@restatable\@nx\@xa[\@nx\thmt@storedoptargs]%
1133   {\thmt@envname}{\thmt@storename}%
1134 }%
1135 \@xa\g@addto@macro\@xa\thmt@local@postheadhook\@xa{%
1136   \tmp@a
1137 }%
1138 }
1139 \thmt@mkignoringkeyhandler{restate phase 1}
1140
1141 \define@key{restate phase 2}{restate}{%
1142   % do not store restate as a key for repetition:
1143   % infinite loop.
1144   % instead, retain the added keyvals
1145   % overwriting thmt@storename should be safe here, it's been
1146   % xdefd into the postheadhook
1147   \thmt@splitrestateargs #1\@
1148 }
1149 \kv@set@family@handler{restate phase 2}{%
1150   \ifthmt@restatethis
1151     \@xa\@xa\@xa\g@addto@macro\@xa\@xa\@xa\thmt@storedoptargs\@xa\@xa\@xa{\@xa\@xa\@xa,%
1152       \@xa\kv@key\@xa=\kv@value}%
1153   \fi
1154 }
1155

```

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.

```

1156
1157 \RequirePackage{thm-patch, aliasctr, parseargs, keyval}
1158
1159 \let\@xa=\expandafter
1160 \let\@nx=\noexpand
1161
1162 \newcommand\thmt@autorefsetup{%
1163   \@xa\def\csname\thmt@envname autorefname\@xa\endcsname\@xa{\thmt@thmname}%
1164   \ifthmt@hassibling
1165     \@counteralias{\thmt@envname}{\thmt@sibling}%
1166     \@xa\def\@xa\thmt@autoreffix\@xa{%
1167       \@xa\let\csname the\thmt@envname\@xa\endcsname
1168       \csname the\thmt@sibling\endcsname
1169       \def\thmt@autoreffix{}}%
1170   }%
1171   \protected@edef\thmt@sibling{\thmt@envname}%
1172   \fi
1173 }
1174 \g@addto@macro\thmt@newtheorem@predefinition{\thmt@autorefsetup}%
1175 \g@addto@macro\thmt@newtheorem@postdefinition{\csname thmt@autoreffix\endcsname}%
1176
1177 \def\thmt@refnamewithcomma #1#2#3,#4,#5\@nil{%
1178   \@xa\def\csname\thmt@envname #1utorefname\endcsname{#3}%
1179   \ifcsname #2refname\endcsname
1180     \csname #2refname\@xa\endcsname\@xa{\thmt@envname}{#3}{#4}%
1181   \fi
1182 }

```



```

1183 \define@key{thmdef}{refname}{\thmt@trytwice}{%
1184   \thmt@refnamewithcomma{a}{c}#1,\textbf{?? (pl. #1)},\@nil
1185 }}
1186 \define@key{thmdef}{Refname}{\thmt@trytwice}{%
1187   \thmt@refnamewithcomma{A}{C}#1,\textbf{?? (pl. #1)},\@nil
1188 }}
1189
1190
1191 \ifcsname Autoref\endcsname\else
1192 \let\thmt@HyRef@testreftype\HyRef@testreftype
1193 \def\HyRef@Testreftype#1.#2\{%
1194   \ltx@ifundefined{#1Autorefname}{%
1195     \thmt@HyRef@testreftype#1.#2\%
1196   }{%
1197     \edef\HyRef@currentHtag{%
1198       \expandafter\noexpand\csname#1Autorefname\endcsname
1199       \noexpand~%
1200     }%
1201   }%
1202 }
1203
1204
1205 \let\thmt@HyPsd@@autorefname\HyPsd@@autorefname
1206 \def\HyPsd@@Autorefname#1.#2\@nil{%
1207   \tracingall
1208   \ltx@ifundefined{#1Autorefname}{%
1209     \thmt@HyPsd@@autorefname#1.#2\@nil
1210   }{%
1211     \csname#1Autorefname\endcsname\space
1212   }%
1213 }%
1214 \def\Autoref{%
1215   \parse{%
1216     {\parseFlag*{\def\thmt@autorefstarg{*}}{\let\thmt@autorefstarg\@empty}}%
1217     {\parseMand{%
1218       \bgroup
1219       \let\HyRef@testreftype\HyRef@Testreftype
1220       \let\HyPsd@@autorefname\HyPsd@@Autorefname
1221       \@xa\autoref\thmt@autorefstarg{##1}%
1222       \egroup
1223       \let\@parsecmd\@empty
1224     }}%
1225   }%
1226 }
1227 \fi % ifcsname Autoref
1228
1229 % not entirely appropriate here, but close enough:
1230 \AtBeginDocument{%
1231   \@ifpackageloaded{nameref}{%
1232     \addtotheoremposttheadhook{%
1233       \expandafter\NR@getttitle\expandafter{\thmt@shortoptarg}%
1234     }}{}
1235 }
1236
1237 \AtBeginDocument{%
1238   \@ifpackageloaded{cleveref}{%
1239     \@ifpackagelater{cleveref}{2010/04/30}{%
1240       % OK, new enough
1241     }{%
1242       \PackageWarningNoLine{thmtools}{%
1243         Your version of cleveref is too old!\MessageBreak

```

```

1244         Update to version 0.16.1 or later%
1245     }
1246 }
1247 }{}
1248 }

```

A.2 Glue code for different backends

A.2.1 amsthm

```

1249 \providecommand\thmt@space{ }
1250
1251 \define@key{thmstyle}{spaceabove}{%
1252   \def\thmt@style@spaceabove{#1}%
1253 }
1254 \define@key{thmstyle}{spacebelow}{%
1255   \def\thmt@style@spacebelow{#1}%
1256 }
1257 \define@key{thmstyle}{headfont}{%
1258   \def\thmt@style@headfont{#1}%
1259 }
1260 \define@key{thmstyle}{bodyfont}{%
1261   \def\thmt@style@bodyfont{#1}%
1262 }
1263 \define@key{thmstyle}{notefont}{%
1264   \def\thmt@style@notefont{#1}%
1265 }
1266 \define@key{thmstyle}{headpunct}{%
1267   \def\thmt@style@headpunct{#1}%
1268 }
1269 \define@key{thmstyle}{notebraces}{%
1270   \def\thmt@style@notebraces{\thmt@embrace#1}%
1271 }
1272 \define@key{thmstyle}{break}[]{%
1273   \def\thmt@style@postheadspace{\newline}%
1274 }
1275 \define@key{thmstyle}{postheadspace}{%
1276   \def\thmt@style@postheadspace{#1}%
1277 }
1278 \define@key{thmstyle}{headindent}{%
1279   \def\thmt@style@headindent{#1}%
1280 }
1281
1282 \newtoks\thmt@style@headstyle
1283 \define@key{thmstyle}{headformat}[]{%
1284   \thmt@setheadstyle{#1}%
1285 }
1286 \define@key{thmstyle}{headstyle}[]{%
1287   \thmt@setheadstyle{#1}%
1288 }
1289 \def\thmt@setheadstyle#1{%
1290   \thmt@style@headstyle{%
1291     \def\NAME{\the\thm@headfont ##1}%
1292     \def\NUMBER{\bgroup\@upn{##2}\egroup}%
1293     \def\NOTE{\if=##3=\else\bgroup\thmt@space\the\thm@notefont(##3)\egroup\fi}%
1294   }%
1295   \def\thmt@tmp{#1}%
1296   \@onelevel@sanitize\thmt@tmp
1297   %\tracingall
1298   \ifcsname thmt@headstyle@\thmt@tmp\endcsname

```



```

1360 \@xa\def\csname th@#1\@xa\endcsname\@xa{\the\thmt@toks}%
1361 % \@xa\def\csname th@#1\@xa\@xa\@xa\@xa\@xa\@xa\@xa\endcsname
1362 % \@xa\@xa\@xa\@xa\@xa\@xa\@xa\@xa{%
1363 % \@xa\@xa\@xa\@xa\@xa\@xa\@xa\thm@notefont
1364 % \@xa\@xa\@xa\@xa\@xa\@xa\@xa\@xa{%
1365 % \@xa\@xa\@xa\thmt@style@notefont
1366 % \@xa\@xa\@xa\thmt@style@notebraces
1367 % \@xa\@xa\@xa\csname th@#1\endcsname
1368 % }
1369 }
1370
1371 \define@key{thmdef}{qed}[\qedsymbol]{%
1372 \thmt@trytwice{}{%
1373 \addtotheoremposttheadhook[\thmt@envname]{%
1374 \protected@edef\qedsymbol{#1}%
1375 \pushQED{\qed}%
1376 }%
1377 \addtotheoremprefoothook[\thmt@envname]{%
1378 \protected@edef\qedsymbol{#1}%
1379 \popQED
1380 }%
1381 }%
1382 }
1383
1384 \def\thmt@amsthmlistbreakhack{%
1385 \leavevmode
1386 \vspace{-\baselineskip}%
1387 \par
1388 \everypar{\setbox\z@\lastbox\everypar{}}}%
1389 }
1390
1391 \define@key{thmuse}{listhack}[\relax]{%
1392 \addtotheoremposttheadhook[local]{%
1393 \thmt@amsthmlistbreakhack
1394 }%
1395 }
1396

```

A.2.2 beamer

```

1397 \newif\ifthmt@hasoverlay
1398 \def\thmt@parsetheoremargs#1{%
1399 \parse{%
1400 {\parseOpt<>{\thmt@hasoverlaytrue\def\thmt@overlay{##1}}{}}%
1401 {\parseOpt[]{\def\thmt@optarg{##1}}{}}%
1402 \let\thmt@shortoptarg\@empty
1403 \let\thmt@optarg\@empty}}%
1404 {\ifthmt@hasoverlay\expandafter\@gobble\else\expandafter\@firstofone\fi
1405 {\parseOpt<>{\thmt@hasoverlaytrue\def\thmt@overlay{##1}}{}}}%
1406 }%
1407 {%
1408 \def\thmt@local@pretheadhook{}%
1409 \def\thmt@local@posttheadhook{}%
1410 \def\thmt@local@prefoothook{}%
1411 \def\thmt@local@postfoothook{}%
1412 \thmt@local@pretheadhook
1413 \csname thmt@#1@pretheadhook\endcsname
1414 \thmt@generic@pretheadhook
1415 \protected@edef\tmp@args{%
1416 \ifthmt@hasoverlay <\thmt@overlay>\fi
1417 \ifx\@empty\thmt@optarg\else [{\thmt@optarg}]\fi

```

```

1418 }%
1419 \csname thmt@original@#1\@xa\endcsname\tmp@args
1420 \thmt@local@posttheadhook
1421 \csname thmt@#1@posttheadhook\endcsname
1422 \thmt@generic@posttheadhook
1423 \let\@parsecmd\@empty
1424 }%
1425 }
1426 }%

```

A.2.3 ntheorem

```

1427
1428 \providecommand\thmt@space{ }
1429
1430 % actually, ntheorem's so-called style is nothing like a style at all...
1431 \def\thmt@declaretheoremstyle@setup{}
1432 \def\thmt@declaretheoremstyle#1{%
1433   \ifcsname th@#1\endcsname\else
1434     \@xa\let\csname th@#1\endcsname\th@plain
1435   \fi
1436 }
1437
1438 \def\thmt@notsupported#1#2{%
1439   \PackageWarning{thmtools}{Key '#2' not supported by #1}{}%
1440 }
1441
1442 \define@key{thmstyle}{spaceabove}{%
1443   \setlength\theorempreskipamount{#1}%
1444 }
1445 \define@key{thmstyle}{spacebelow}{%
1446   \setlength\theorempostskipamount{#1}%
1447 }
1448 \define@key{thmstyle}{headfont}{%
1449   \theoremheaderfont{#1}%
1450 }
1451 \define@key{thmstyle}{bodyfont}{%
1452   \theorembodyfont{#1}%
1453 }
1454 % not supported in ntheorem.
1455 \define@key{thmstyle}{notefont}{%
1456   \thmt@notsupported{ntheorem}{notefont}%
1457 }
1458 \define@key{thmstyle}{headpunct}{%
1459   \theoremseparator{#1}%
1460 }
1461 % not supported in ntheorem.
1462 \define@key{thmstyle}{notebraces}{%
1463   \thmt@notsupported{ntheorem}{notebraces}%
1464 }
1465 \define@key{thmstyle}{break}{%
1466   \theoremstyle{break}%
1467 }
1468 % not supported in ntheorem...
1469 \define@key{thmstyle}{postheadspace}{%
1470   %\def\thmt@style@postheadspace{#1}%
1471   \@xa\g@addto@macro\csname thmt@style\thmt@style @defaultkeys\endcsname{%
1472     posttheadhook={\hspace{-\labelsep}\hspace*{#1}},%
1473   }%
1474 }
1475

```

```

1476 % not supported in ntheorem
1477 \define@key{thmstyle}{headindent}{%
1478   \thmt@notsupported{ntheorem}{headindent}%
1479 }
1480 % sorry, only style, not def with ntheorem.
1481 \define@key{thmstyle}{qed}[\qedsymbol]{%
1482   \@ifpackagewith{ntheorem}{thmmarks}{%
1483     \theoremsymbol{#1}%
1484   }{%
1485     \thmt@notsupported
1486       {ntheorem without thmmarks option}%
1487       {headindent}%
1488   }%
1489 }
1490
1491 \let\@upn=\textup
1492 \define@key{thmstyle}{headformat}[]{%
1493   \def\thmt@tmp{#1}%
1494   \@onelevel@sanitize\thmt@tmp
1495   %\tracingall
1496   \ifcsname thmt@headstyle@\thmt@tmp\endcsname
1497     \newtheoremstyle{\thmt@style}{%
1498       \item[\hskip\labelsep\theorem@headerfont%
1499         \def\NAME{\theorem@headerfont #####1}%
1500         \def\NUMBER{\bgroup\@upn{#####2}\egroup}%
1501         \def\NOTE{}}%
1502       \csname thmt@headstyle@#1\endcsname
1503       \theorem@separator
1504     ]
1505   }{%
1506     \item[\hskip\labelsep\theorem@headerfont%
1507       \def\NAME{\theorem@headerfont #####1}%
1508       \def\NUMBER{\bgroup\@upn{#####2}\egroup}%
1509       \def\NOTE{\if=#####3=\else\bgroup\thmt@space(#####3)\egroup\fi}%
1510       \csname thmt@headstyle@#1\endcsname
1511       \theorem@separator
1512     ]
1513   }
1514 \else
1515   \newtheoremstyle{\thmt@style}{%
1516     \item[\hskip\labelsep\theorem@headerfont%
1517       \def\NAME{\the\thm@headfont #####1}%
1518       \def\NUMBER{\bgroup\@upn{#####2}\egroup}%
1519       \def\NOTE{}}%
1520     #1%
1521     \theorem@separator
1522   ]
1523 }{%
1524   \item[\hskip\labelsep\theorem@headerfont%
1525     \def\NAME{\the\thm@headfont #####1}%
1526     \def\NUMBER{\bgroup\@upn{#####2}\egroup}%
1527     \def\NOTE{\if=#####3=\else\bgroup\thmt@space(#####3)\egroup\fi}%
1528     #1%
1529     \theorem@separator
1530   ]
1531 }
1532 \fi
1533 }
1534
1535 \def\thmt@headstyle@margin{%
1536   \makebox[0pt][r]{\NUMBER\ } \NAME\NOTE

```

```

1537 }
1538 \def\thmt@headstyle@swapnumber{%
1539   \NUMBER\ \NAME\NOTE
1540 }
1541
1542
1543

```

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.

```

1544
1545 \newtoks\@parsespec
1546 \def\parse@endquark{\parse@endquark}
1547 \newcommand\parse[1]{%
1548   \@parsespec{#1\parse@endquark}\@parse}
1549
1550 \newcommand\@parse{%
1551   \edef\p@tmp{\the\@parsespec}%
1552   \ifx\p@tmp\parse@endquark
1553     \expandafter\@gobble
1554   \else
1555     \typeout{parsespec remaining: \the\@parsespec}%
1556     \expandafter\@firstofone
1557   \fi{%
1558     \@parsepop
1559   }%
1560 }
1561 \def\@parsepop{%
1562   \expandafter\p@rsepop\the\@parsespec\@nil
1563   \@parsecmd
1564 }
1565 \def\p@rsepop#1#2\@nil{%
1566   #1%
1567   \@parsespec{#2}%
1568 }
1569
1570 \newcommand\parseOpt[4]{%
1571   %\parseOpt{openchar}{closechar}{yes}{no}
1572   \typeout{attempting #1#2...}%
1573   \def\@parsecmd{%
1574     \@ifnextchar#1{\@@reallyparse}{#4\@parse}%
1575   }%
1576   \def\@@reallyparse#1##1#2{%
1577     #3\@parse
1578   }%
1579 }
1580
1581 \newcommand\parseMand[1]{%
1582   %\parseMand{code}
1583   \def\@parsecmd##1{#1\@parse}%
1584 }
1585

```

```

1586 \newcommand\parseFlag[3]{%
1587   \%parseFlag{flagchar}{yes}{no}
1588   \def\@parsecmd{%
1589     \@ifnextchar#1{#2\expandafter\@parse\@gobble}{#3\@parse}%
1590   }%
1591 }

```

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`.

```

1592 \def\aliasctr@follow#1#2\@nil#3{%
1593   \ifx#1\@elt
1594     \noexpand #3%
1595   \else
1596     \expandafter\aliasctr@follow#1\@elt\@nil{#1}%
1597   \fi
1598 }
1599 \newcommand\aliasctr@follow[1]{%
1600   \expandafter\aliasctr@follow

```

Don't be confused: the third parameter is ignored here, we always have recursion here since the *token* `\cl@#1` is (hopefully) not `\@elt`.

```

1601   \csname cl@#1\endcsname\@elt\@nil{\csname cl@#1\endcsname}%
1602 }
1603 \renewcommand*\@addtoreset[2]{\bgroup
1604   \edef\aliasctr@@truelist{\aliasctr@follow{#2}}%
1605   \let\@elt\relax
1606   \expandafter\@cons\aliasctr@@truelist{{#1}}%
1607 \egroup}

```

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

```

1608 % FMI 2019-07-31 \@removefromreset is in the kernel these days
1609 \ifundefined{\removefromreset}{\RequirePackage{remreset}}{}
1610 \renewcommand*\@removefromreset[2]{\bgroup
1611   \edef\aliasctr@@truelist{\aliasctr@follow{#2}}%
1612   \expandafter\let\csname c@#1\endcsname\@removefromreset
1613   \def\@elt##1{%
1614     \expandafter\ifx\csname c@##1\endcsname\@removefromreset
1615     \else
1616       \noexpand\@elt{##1}%
1617     \fi}%
1618   \expandafter\xdef\aliasctr@@truelist{%
1619     \aliasctr@@truelist}
1620 \egroup}

```

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

```

1621 \newcommand\@counteralias[2]{%
1622   \def\@gletover##1##2{%
1623     \expandafter\global
1624     \expandafter\let\csname ##1\endcsname
1625     \csname ##2\endcsname
1626   }%
1627   \ifundefined{c@#2}{\@nocounterr{#2}}{%
1628     \@ifdefinable{c@#1}{%

```


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.

```
1629 \@@gletover{c@#1}{c@#2}%  
1630 \@@gletover{the#1}{the#2}%
```

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

```
1631 \@@gletover{theH#1}{theH#2}%  
1632 \@@gletover{p@#1}{p@#2}%  
1633 \expandafter\global  
1634 \expandafter\def\csname cl@#1\expandafter\endcsname  
1635 \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.

```
1636 %\@addtoreset{#1}{@ckpt}%  
1637 }%  
1638 }%  
1639 }}
```

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. `\ifuniqmark` 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.

```
1640  
1641 \DeclareOption{unq}{%  
1642 \newwrite\uniq@channel  
1643 \InputIfFileExists{\jobname.unq}{\relax}{\relax}%  
1644 \immediate\openout\uniq@channel=\jobname.unq  
1645 \AtEndDocument{%  
1646 \immediate\closeout\uniq@channel%  
1647 }  
1648 }  
1649 \DeclareOption{aux}{%  
1650 \let\uniq@channel\@auxout  
1651 }  
1652
```

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`.

```
1653 \newcommand\setuniqmark[1]{%  
1654 \expandafter\ifx\csname uniq@now@#1\endcsname\relax  
1655 \global\@namedef{uniq@now@#1}{\uniq@ONE}%  
1656 \else
```

```

1657 \expandafter\ifx\csname uniq@now@#1\endcsname\uniq@MANY\else
1658 \immediate\write\uniq@channel{%
1659   \string\uniq@setmany{#1}%
1660 }%
1661 \ifuniq{#1}{%
1662   \uniq@warnnotunique{#1}%
1663 }{}%
1664 \fi
1665 \global\@namedef{uniq@now@#1}{\uniq@MANY}%
1666 \fi
1667 }

```

Companion to \setuniqmark: if the uniqmark given in the first argument was called more than once, execute the second argument, otherwise execute the first 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?

```

1668 \newcommand\ifuniq[1]{%
1669   \expandafter\ifx\csname uniq@last@#1\endcsname\uniq@MANY
1670   \expandafter \@secondoftwo
1671   \else
1672   \expandafter\@firstoftwo
1673   \fi
1674 }

```

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

```

1675 \def\uniq@ONE{\uniq@ONE}
1676 \def\uniq@MANY{\uniq@MANY}

```

Flag: suggest a rerun?

```

1677 \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.

```

1678 \def\uniq@setmany#1{%
1679   \global\@namedef{uniq@last@#1}{\uniq@MANY}%
1680   \AtEndDocument{%
1681     \uniq@warnifunique{#1}%
1682   }%
1683 }

```

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.

```

1684 \def\uniq@warnifunique#1{%
1685   \expandafter\ifx\csname uniq@now@#1\endcsname\uniq@MANY\else
1686   \PackageWarningNoLine{uniq}{%
1687     ‘#1’ is unique now.\MessageBreak
1688     Rerun LaTeX to pick up the change%
1689   }%
1690   \@uniq@reruntrue
1691   \fi
1692 }

```

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.

```

1693 \def\uniq@warnnotunique#1{%
1694   \PackageWarningNoLine{uniq}{%
1695     ‘#1’ is not unique anymore.\MessageBreak
1696     Rerun LaTeX to pick up the change%
1697   }%

```

```

1698 \@uniq@reruntrue
1699 }

```

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.

```

1700 \def\uniq@maybesuggestrerun{%
1701   \if@uniq@rerun
1702     \PackageWarningNoLine{uniq}{%
1703       Uniquenesses have changed. \MessageBreak
1704       Rerun LaTeX to pick up the change%
1705     }%
1706   \fi
1707 }

```

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

```

1708 \AtEndDocument{%
1709   \immediate\write\@auxout{\string\uniq@maybesuggestrerun}%
1710 }
1711 \ExecuteOptions{aux}
1712 \ProcessOptions\relax

```