

Thmtools Users' Guide

Dr. Ulrich M. Schwarz – ulmi@absatzen.de*

2014/04/21 v66

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	17
1.2 Frilly references	3	A.1 Core functionality	17
1.3 Styling theorems	4	A.1.1 The main package	17
1.3.1 Declaring new theoremstyles . .	5	A.1.2 Adding hooks to the relevant commands	18
1.4 Repeating theorems	6	A.1.3 The key-value interfaces	21
1.5 Lists of theorems	6	A.1.4 Lists of theorems	29
1.6 Extended arguments to theorem environments	8	A.1.5 Re-using environments	32
2 Thmtools for the extravagant	9	A.1.6 Restrictions	33
2.1 Understanding thmtools' extension mechanism	9	A.1.7 Fixing autoref and friends	37
2.2 Case in point: the shaded key	9	A.2 Glue code for different backends	38
2.3 Case in point: the thmbox key	11	A.2.1 amsthm	38
2.4 Case in point: the mdframed key	11	A.2.2 beamer	41
2.5 How thmtools finds your extensions	11	A.2.3 ntheorem	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	46

*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=\ "Ubung]{exercise}

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

Übung 1. *Prove Euclid's Theorem.*

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

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

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

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

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

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

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

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

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

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

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

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

Couple 1. *Marc & Anne*

Singleton. *Me.*

Couple 2. *Buck & Britta*

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

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

ready 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 `rbg` model to manually specify white. (Shadethm’s default value is some sort of gray.)

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

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

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

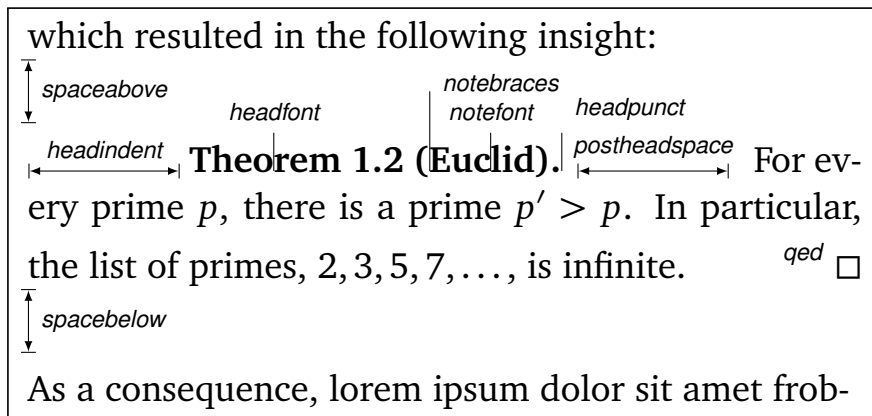


Figure 1.1: Settable parameters of a theorem style.

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`, 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 environment is optional because there is also a set of “generic” hooks added to every theorem that you define.

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

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

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

2.2 Case in point: the shaded key

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

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

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

The docs for shadethm say:

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

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

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

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

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

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

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

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

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

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

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

2.3 Case in point: the thmbox key

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

```
39 \define@key{thmdef}{thmbox}[L]{%
40   \thmt@trytwice{%
41     \let\oldproof=\proof
42     \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 protude 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 protude 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 Value: a counter name. The theorem will be reset whenever that counter is incremented. Usually, this will be a sectioning level, `chapter` or `section`. (Same as `parent`.)

within Value: a counter name. The theorem will be reset whenever that counter is incremented. Usually, this will be a sectioning level, `chapter` or `section`. (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 Value: a counter name. The theorem will use this counter for numbering. Usually, this is the name of another theorem environment. (Same as `sibling`.)

sharenumber Value: a counter name. The theorem will use this counter for numbering. Usually, this is the name of another theorem environment. (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 a accented character, for example.

name 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 a accented character, for example. (Same as `title`.)

heading 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 a accented character, for example. (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 string 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 `.`

Refname Value: one string, or two string 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 above.

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 footooks are executed LIFO (last addition first) and headhooks are executed FIFO (first addition first). There is a special kind called generic that is called for all theorems. This is the default if no kind is given.

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

```

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      \let\@parsecmd\@empty
153  }%
154 }%
155 }%
156
157 \let\thmt@original@newtheorem\newtheorem
158 \let\thmt@theoremdefiner\thmt@original@newtheorem
159
160 \def\newtheorem{%
161   \thmt@isstarredfalse
162   \thmt@hassiblingfalse
163   \thmt@hasparentfalse
164   \parse{%
165     {\parseFlag*{\thmt@isstarredtrue}{}}%
166     {\parseMand{\def\thmt@envname{##1}}}%
167     {\parseOpt[]{\thmt@hassiblingtrue\def\thmt@sibling{##1}}}%
168     {\parseMand{\def\thmt@thmname{##1}}}%
169     {\parseOpt[]{\thmt@hasparenttrue\def\thmt@parent{##1}}}%
170     {\let\@parsecmd\thmt@newtheoremiv}%
171   }%
172 }
173
174 \newcommand\thmt@newtheoremiv{%
175   \thmt@newtheorem@predefinition
176   % whee, now reassemble the whole shebang.
177   \protected@edef\thmt@args{%
178     \@nx\thmt@theoremdefiner%
179     \ifthmt@isstarred *\fi
180     {\thmt@envname}%
181     \ifthmt@hassibling [\thmt@sibling]\fi
182     {\thmt@thmname}%
183     \ifthmt@hasparent [\thmt@parent]\fi
184   }
185   \thmt@args
186   \thmt@newtheorem@postdefinition
187 }
188
189 \newcommand\thmt@newtheorem@predefinition{}
190 \newcommand\thmt@newtheorem@postdefinition{%
191   \let\thmt@theoremdefiner\thmt@original@newtheorem
192 }
193
194 \g@addto@macro\thmt@newtheorem@predefinition{%
195   \@xa\thmt@providetheoremhooks\@xa{\thmt@envname}%
196 }
197 \g@addto@macro\thmt@newtheorem@postdefinition{%
198   \@xa\thmt@addtheoremhook\@xa{\thmt@envname}%
199   \ifthmt@isstarred\@namedef{the\thmt@envname}{}\fi
200   \protected@edef\thmt@tmp{%
201     \def\@nx\thmt@envname{\thmt@envname}%
202     \def\@nx\thmt@thmname{\thmt@thmname}%
203   }%
204   \@xa\addtotheoremreheadhook\@xa[\@xa\thmt@envname\@xa]\@xa{%
205     \thmt@tmp
206   }%
207 }

```

```

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

```

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.

```

270 \ifx\proof\endproof\else% yup, that's a quaint way of doing it :)
271 % FIXME: this assumes proof has the syntax of theorems, which
272 % usually happens to be true (optarg overrides "Proof" string).
273 % FIXME: refactor into thmt@addtheoremhook, but we really don't want to
274 % call the generic-hook...
275 \let\thmt@original@proof=\proof
276 \renewcommand\proof{%
277   \thmt@parseproofargs%
278 }%
279 \def\thmt@parseproofargs{%
280   \parse{%
281     {\parseOpt[]{\def\thmt@optarg{##1}}{\let\thmt@optarg@empty}}%
282     {%
283       \thmt@proof@preheadhook
284       %\thmt@generic@preheadhook
285       \protected@edef\tmp@args{%
286         \ifx\@empty\thmt@optarg\else [\thmt@optarg]\fi
287       }%
288       \csname thmt@original@proof\@xa\endcsname\tmp@args
289       \thmt@proof@postheadhook
290       %\thmt@generic@postheadhook
291       \let\@parsecmd\@empty
292     }%
293   }%
294 }%
295
296 \let\thmt@original@endproof=\endproof
297 \def\endproof{%
298   % these need to be in opposite order of headhooks.
299   %\csname thmtgeneric@prefoothook\endcsname
300   \thmt@proof@prefoothook
301   \thmt@original@endproof
302   %\csname thmt@generic@postfoothook\endcsname
303   \thmt@proof@postfoothook
304 }%
305 \@namedef{thmt@proof@preheadhook}{}%
306 \@namedef{thmt@proof@postheadhook}{}%
307 \@namedef{thmt@proof@prefoothook}{}%
308 \@namedef{thmt@proof@postfoothook}{}%
309 \fi

```

A.1.3 The key-value interfaces

```

310
311 \let\@xa\expandafter
312 \let\@nx\noexpand
313
314 \DeclareOption{lowercase}{%
315   \PackageInfo{thm-kv}{Theorem names will be lowercased}%
316   \global\let\thmt@modifycase\MakeLowercase}
317
318 \DeclareOption{uppercase}{%
319   \PackageInfo{thm-kv}{Theorem names will be uppercased}%
320   \global\let\thmt@modifycase\MakeUppercase}
321
322 \DeclareOption{anycase}{%
323   \PackageInfo{thm-kv}{Theorem names will be unchanged}%

```

```

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

```

```

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

```

```

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

```



```

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

```

```

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

```

```

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

```

```

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

```

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.

```

734 \def\thmt@declaretheoremstyle@setup{}
735 \def\thmt@declaretheoremstyle#1{%
736   \PackageWarning{thmtools}{Your backend doesn't allow styling theorems}{}
737 }
738 \newcommand\declaretheoremstyle[2][]{%
739   \def\thmt@style{#2}%
740   \@xa\def\csname thmt@style \thmt@style @defaultkeys\endcsname{%
741     \thmt@declaretheoremstyle@setup
742     \kvsetkeys{thmstyle}{#1}%
743     \thmt@declaretheoremstyle{#2}%
744 }
745 \@onlypreamble\declaretheoremstyle
746
747 \kv@set@family@handler{thmstyle}{%

```

```

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

```

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.

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.

```

766 \let\@xa=\expandafter
767 \let\@nx=\noexpand
768 \RequirePackage{thm-patch,keyval,kvsetkeys}
769
770 \def\thmtlo@oldchapter{0}%
771 \newcommand\thmtlo@chaptervspacehack{}
772 \ifcsname c@chapter\endcsname
773   \ifx\c@chapter\relax\else
774     \def\thmtlo@chaptervspacehack{%
775       \ifnum \value{chapter}=\thmtlo@oldchapter\relax\else
776         % new chapter, add vspace to loe.

```

```

777     \addtocontents{loe}{\protect\addvspace{10\p@}}%
778     \xdef\thmtlo@oldchapter{\arabic{chapter}}%
779     \fi
780   }%
781   \fi
782 \fi
783
784
785 \providecommand\listtheoremname{List of Theorems}
786 \newcommand\listoftheorems[1][1]{%
787   %% much hacking here to pick up the definition from the class
788   %% without oodles of conditionals.
789   \bgroup
790   \setlisttheoremstyle{#1}%
791   \let\listfigurename\listtheoremname
792   \def\contentsline##1{%
793     \csname thmt@contentsline@##1\endcsname{##1}%
794   }%
795   \@for\thmt@envname:=\thmt@allenvs\do{%
796     \@xa\protected@edef\csname l@\thmt@envname\endcsname{% CHECK: why p@edef?
797       \@nx\@dottedtocline{1}{1.5em}{\@nx\thmt@listnumwidth}%
798     }%
799   }%
800   \let\thref@starttoc\@starttoc
801   \def\@starttoc##1{\thref@starttoc{loe}}%
802   % new hack: to allow multiple calls, we defer the opening of the
803   % loe file to AtEndDocument time. This is before the aux file is
804   % read back again, that is early enough.
805   % TODO: is it? crosscheck include/includeonly!
806   \@fileswfalse
807   \AtEndDocument{%
808     \if@filesw
809       \ifundefined{tf@loe}{%
810         \expandafter\newwrite\csname tf@loe\endcsname
811         \immediate\openout \csname tf@loe\endcsname \jobname.loe\relax
812       }{}%
813     \fi
814   }%
815   %\expandafter
816   \listoffigures
817   \egroup
818 }
819
820 \newcommand\setlisttheoremstyle[1]{%
821   \kvsetkeys{thmt-listof}{#1}%
822 }
823 \define@key{thmt-listof}{numwidth}{\def\thmt@listnumwidth{#1}}
824 \define@key{thmt-listof}{ignore}{\thmt@allenvs}{\ignoretheorems{#1}}
825 \define@key{thmt-listof}{onlynamed}{\thmt@allenvs}{\onlynamedtheorems{#1}}
826 \define@key{thmt-listof}{show}{\thmt@allenvs}{\showtheorems{#1}}
827 \define@key{thmt-listof}{ignoreall}[true]{\ignoretheorems{\thmt@allenvs}}
828 \define@key{thmt-listof}{showall}[true]{\showtheorems{\thmt@allenvs}}
829
830 \providecommand\thmt@listnumwidth{2.3em}
831
832 \providecommand\thmtformatoptarg[1]{ (#1)}
833
834 \newcommand\thmt@mklistcmd{%
835   \@xa\protected@edef\csname l@\thmt@envname\endcsname{% CHECK: why p@edef?
836     \@nx\@dottedtocline{1}{1.5em}{\@nx\thmt@listnumwidth}%
837   }%

```

```

838 \ifthmt@isstarred
839   \@xa\def\csname ll@\thmt@envname\endcsname{%
840     \protect\numberline{\protect\let\protect\autodot\protect\@empty}%
841     \thmt@thmname
842     \ifx\@empty\thmt@shortoptarg\else\protect\thmtformatoptarg{\thmt@shortoptarg}\fi
843   }%
844 \else
845   \@xa\def\csname ll@\thmt@envname\endcsname{%
846     \protect\numberline{\csname the\thmt@envname\endcsname}%
847     \thmt@thmname
848     \ifx\@empty\thmt@shortoptarg\else\protect\thmtformatoptarg{\thmt@shortoptarg}\fi
849   }%
850 \fi
851 \@xa\gdef\csname thmt@contentsline@\thmt@envname\endcsname{%
852   \thmt@contentslineShow% default:show
853 }%
854 }
855 \def\thmt@allenvs{\@gobble}
856 \newcommand\thmt@recordenvname{%
857   \edef\thmt@allenvs{\thmt@allenvs,\thmt@envname}%
858 }
859 \g@addto@macro\thmt@newtheorem@predefinition{%
860   \thmt@mklistcmd
861   \thmt@recordenvname
862 }
863
864 \addtotheorempostheadhook{%
865   \thmtlo@chaptervspacehack
866   \addcontentsline{loe}{\thmt@envname}{%
867     \csname ll@\thmt@envname\endcsname
868   }%
869 }
870
871 \newcommand\showtheorems[1]{%
872   \@for\thmt@thm:=#1\do{%
873     \typeout{showing \thmt@thm}%
874     \@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
875       =\thmt@contentslineShow
876   }%
877 }
878
879 \newcommand\ignoretheorems[1]{%
880   \@for\thmt@thm:=#1\do{%
881     \@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
882       =\thmt@contentslineIgnore
883   }%
884 }
885 \newcommand\onlynamedtheorems[1]{%
886   \@for\thmt@thm:=#1\do{%
887     \global\@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
888       =\thmt@contentslineIfNamed
889   }%
890 }
891
892 \AtBeginDocument{%
893   \@ifpackageloaded{hyperref}{%
894     \let\thmt@hygobble\@gobble
895   }{%
896     \let\thmt@hygobble\@empty
897   }
898   \let\thmt@contentsline\contentsline

```

```

899 }
900
901 \def\thmt@contentslineIgnore#1#2#3{%
902   \thmt@hygobble
903 }
904 \def\thmt@contentslineShow{%
905   \thmt@contentsline
906 }
907
908 \def\thmt@contentslineIfNamed#1#2#3{%
909   \thmt@ifhasoptname #2\thmtformatoptarg\@nil{%
910     \thmt@contentslineShow{#1}{#2}{#3}%
911   }{%
912     \thmt@contentslineIgnore{#1}{#2}{#3}%
913   }%\thmt@contentsline{#1}{#2}{#3}%
914 }
915 }
916
917 \def\thmt@ifhasoptname #1\thmtformatoptarg#2\@nil{%
918   \ifx\@nil#2\@nil
919     \@xa\@secondoftwo
920   \else
921     \@xa\@firstoftwo
922   \fi
923 }

```

A.1.5 Re-using environments

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

```

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

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

```

which yields

Lemma 4 (Zorn). *If every chain in X is upper-bounded, X has a maximal element. It's true, you know!*

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

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

Lemma 4 (Zorn). *If every chain in X is upper-bounded, X has a maximal element. It's true, you know!*

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

As you can see, we use the starred form `\mylemma*`. As in many cases in \TeX , 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 \TeX `\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.

```

924 \RequirePackage{thmtools}
925 \let\@xa\expandafter
926 \let\@nx\noexpand
927 \@ifundefined{c@thmt@dummysctr}{%
928   \newcounter{thmt@dummysctr}%
929   }{}
930 \gdef\theHthmt@dummysctr{dummy.\arabic{thmt@dummysctr}}%
931 \gdef\thethmt@dummysctr{}%
932 \long\def\thmt@collect@body#1#2\end#3{%
933   \@xa\thmt@toks\@xa{\the\thmt@toks #2}%
934   \def\thmttmpa{#3}\def\thmttmpb{restatable}%
935   \ifx\thmttmpa\@currentenv\thmttmpb
936     \@xa\@firstoftwo% this is the end of the environment.
937   \else
938     \@xa\@secondoftwo% go on collecting
939   \fi}% this is the end, my friend, drop the \end.
940 % and call #1 with the collected body.
941   \@xa#1\@xa{\the\thmt@toks}%
942 }{% go on collecting
943   \@xa\thmt@toks\@xa{\the\thmt@toks\end{#3}}%
944   \thmt@collect@body{#1}%
945 }%
946 }

```

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

```

947 \def\thmt@trivialref#1#2{%
948   \ifcsname r@#1\endcsname
949     \@xa\@xa\@xa\thmt@trivi@lr@f\csname r@#1\endcsname\relax\@nil
950   \else #2\fi
951 }
952 \def\thmt@trivi@lr@f#1#2\@nil{#1}

```

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

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

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

```

953 \def\thmt@innercounters{%
954   equation}
955 \def\thmt@counterformatters{%
956   @alph,@Alph,@arabic,@roman,@Roman,@fnsymbol}
957
958 \@for\thmt@displ:=\thmt@counterformatters\do{%
959   \@xa\let\csname thmt@\thmt@displ\@xa\endcsname\csname \thmt@displ\endcsname
960 }%
961 \def\thmt@sanitizethe#1{%
962   \@for\thmt@displ:=\thmt@counterformatters\do{%
963     \@xa\protected@edef\csname\thmt@displ\endcsname##1{%
964       \@nx\ifx\@xa\@nx\csname c@#1\endcsname ##1%
965       \@xa\protect\csname \thmt@displ\endcsname{##1}%
966       \@nx\else
967       \@nx\csname thmt@\thmt@displ\endcsname{##1}%
968       \@nx\fi
969     }%
970   }%
971   \expandafter\protected@edef\csname the#1\endcsname{\csname the#1\endcsname}%
972   \ifcsname theH#1\endcsname
973     \expandafter\protected@edef\csname theH#1\endcsname{\csname theH#1\endcsname}%
974   \fi
975 }
976
977 \def\thmt@rst@storecounters#1{%
978   \bgroup
979     % ugly hack: save chapter,..subsection numbers
980     % for equation numbers.
981     %\refstepcounter{thmt@dummyctr}% why is this here?
982     %% temporarily disabled, broke autorefname.
983     \def\@currentlabel{}%
984     \@for\thmt@ctr:=\thmt@innercounters\do{%
985       \thmt@sanitizethe{\thmt@ctr}%
986       \protected@edef\@currentlabel{%
987         \@currentlabel
988         \protect\def\@xa\protect\csname the\thmt@ctr\endcsname{%
989           \csname the\thmt@ctr\endcsname}%
990         \ifcsname theH\thmt@ctr\endcsname
991           \protect\def\@xa\protect\csname theH\thmt@ctr\endcsname{%
992             (restate \protect\theHthmt@dummyctr)\csname theH\thmt@ctr\endcsname}%
993           \fi
994         \protect\setcounter{\thmt@ctr}{\number\csname c@\thmt@ctr\endcsname}%
995       }%
996     }%
997     \label{thmt@@#1@data}%
998   \egroup
999 }%

```

Now, the main business.

```

1000 \newif\ifthmt@thisistheone
1001 \newenvironment{thmt@restatable}[3][[]]{%
1002   \thmt@toks{}}% will hold body
1003 %
1004 \stepcounter{thmt@dummyctr}% used for data storage label.
1005 %
1006 \long\def\thmrst@store##1{%
1007   \@xa\gdef\csname #3\endcsname{%
1008     \@ifstar{}%

```

```

1009     \thmt@thisistheonefalse\csname thmt@stored@#3\endcsname
1010   }{%
1011     \thmt@thisistheonettrue\csname thmt@stored@#3\endcsname
1012   }%
1013 }%
1014 \@xa\long\@xa\gdef\csname thmt@stored@#3\@xa\endcsname\@xa{%
1015   \begingroup
1016   \ifthmt@thisistheone
1017     % these are the valid numbers, store them for the other
1018     % occasions.
1019     \thmt@rst@storecounters{#3}%
1020   \else
1021     % this one should use other numbers...
1022     % first, fake the theorem number.
1023     \@xa\protected@edef\csname the#2\endcsname{%
1024       \thmt@trivialref{thmt@@#3}{??}}%
1025     % if the number wasn't there, have a "re-run to get labels right"
1026     % warning.
1027     \ifcsname r@thmt@@#3\endcsname\else
1028       \G@refundefinedtrue
1029     \fi
1030     % prevent stepcountering the theorem number,
1031     % but still, have some number for hyperref, just in case.
1032     \@xa\let\csname c@#2\endcsname=\c@thmt@dummyctr
1033     \@xa\let\csname theH#2\endcsname=\theHthmt@dummyctr
1034     % disable labeling.
1035     \let\label=\@gobble
1036     \let\ltx@label=\@gobble% amsmath needs this
1037     % We shall need to restore the counters at the end
1038     % of the environment, so we get
1039     % (4.2) [(3.1 from restate)] (4.3)
1040     \def\thmt@restorecounters{%
1041       \@for\thmt@ctr:=\thmt@innercounters\do{%
1042         \protected@edef\thmt@restorecounters{%
1043           \thmt@restorecounters
1044           \protect\setcounter{\thmt@ctr}{\arabic{\thmt@ctr}}%
1045         }%
1046       }%
1047     % pull the new semi-static definition of \theequation et al.
1048     % from the aux file.
1049     \thmt@trivialref{thmt@@#3@data}{}%
1050   \fi
1051   % call the proper begin-env code, possibly with optional argument
1052   % (omit if stored via key-val)
1053   \ifthmt@restatethis
1054     \thmt@restatethisfalse
1055   \else
1056     \csname #2\@xa\endcsname\ifx\@nx#1\@nx\else[#{#1}]\fi
1057   \fi
1058   \ifthmt@thisistheone
1059     % store a label so we can pick up the number later.
1060     \label{thmt@@#3}%
1061   \fi
1062   % this will be the collected body.
1063   ##1%
1064   \csname end#2\endcsname
1065   % if we faked the counter values, restore originals now.
1066   \ifthmt@thisistheone\else\thmt@restorecounters\fi
1067   \endgroup
1068 }% thmt@stored@#3
1069 % in either case, now call the just-created macro,

```

```

1070 \csname #3\@xa\endcsname\ifthmt@thisistheone\else*\fi
1071 % and artificially close the current environment.
1072 \@xa\end\@xa{\@currenvir}
1073 }% thm@rst@store
1074 \thmt@collect@body\thm@rst@store
1075 }{%
1076 %% now empty, just used as a marker.
1077 }
1078
1079 \newenvironment{restatable}{%
1080 \thmt@thisistheonetrue\thmt@restatable
1081 }{%
1082 \endthmt@restatable
1083 }
1084 \newenvironment{restatable*}{%
1085 \thmt@thisistheonefalse\thmt@restatable
1086 }{%
1087 \endthmt@restatable
1088 }
1089
1090 %% support for keyval-style: restate=foobar
1091 \protected@edef\thmt@thmuse@families{%
1092 \thmt@thmuse@families%
1093 ,restate phase 1%
1094 ,restate phase 2%
1095 }
1096 \newcommand\thmt@splitrestateargs[1][]{%
1097 \g@addto@macro\thmt@storedoptargs{,#1}%
1098 \def\tmp@a##1\@{\def\thmt@storename{##1}}%
1099 \tmp@a
1100 }
1101
1102 \newif\ifthmt@restatethis
1103 \define@key{restate phase 1}{restate}{%
1104 \thmt@thmuse@iskvtrue
1105 \def\thmt@storedoptargs{}% discard the first time around
1106 \thmt@splitrestateargs #1\@
1107 \def\thmt@storedoptargs{}% discard the first time around
1108 %\def\thmt@storename{#1}%
1109 \thmt@debug{we will restate as '\thmt@storename' with more args
1110 '\thmt@storedoptargs'}%
1111 \@namedef{thmt@unusedkey@restate}{}%
1112 % spurious "unused key" fixes itself once we are after tracknames...
1113 \thmt@restatethistrue
1114 \protected@edef\tmp@a{%
1115 \@nx\thmt@thisistheonetrue
1116 \@nx\def\@nx\@currenvir{\thmt@envname}%
1117 \@nx\@xa\@nx\thmt@restatable\@nx\@xa[\@nx\thmt@storedoptargs]%
1118 {\thmt@envname}\thmt@storename}%
1119 }%
1120 \@xa\g@addto@macro\@xa\thmt@local@postheadhook\@xa{%
1121 \tmp@a
1122 }%
1123 }
1124 \thmt@mkignoringkeyhandler{restate phase 1}
1125
1126 \define@key{restate phase 2}{restate}{%
1127 % do not store restate as a key for repetition:
1128 % infinite loop.
1129 % instead, retain the added keyvals
1130 % overwriting thmt@storename should be safe here, it's been

```

```

1131 % xdef into the postheadhook
1132 \thmt@splitrestateargs #1\@
1133 }
1134 \kv@set@family@handler{restate phase 2}{%
1135 \ifthmt@restate this
1136 \@xa\@xa\@xa@g@addto@macro\@xa\@xa\@xa\thmt@storedoptargs\@xa\@xa\@xa{\@xa\@xa\@xa,%
1137 \@xa\kv@key\@xa=\kv@value}%
1138 \fi
1139 }
1140

```

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.

```

1141
1142 \RequirePackage{thm-patch, aliasctr, parseargs, keyval}
1143
1144 \let\@xa=\expandafter
1145 \let\@nx=\noexpand
1146
1147 \newcommand\thmt@autorefsetup{%
1148 \@xa\def\csname\thmt@envname autorefname\@xa\endcsname\@xa{\thmt@thmname}%
1149 \ifthmt@hassibling
1150 \@counteralias{\thmt@envname}{\thmt@sibling}%
1151 \@xa\def\@xa\thmt@autoreffix\@xa{%
1152 \@xa\let\csname the\thmt@envname\@xa\endcsname
1153 \csname the\thmt@sibling\endcsname
1154 \def\thmt@autoreffix{}}%
1155 }%
1156 \protected@edef\thmt@sibling{\thmt@envname}%
1157 \fi
1158 }
1159 \g@addto@macro\thmt@newtheorem@predefinition{\thmt@autorefsetup}%
1160 \g@addto@macro\thmt@newtheorem@postdefinition{\csname thmt@autoreffix\endcsname}%
1161
1162 \def\thmt@refnamewithcomma #1#2#3,#4,#5\@nil{%
1163 \@xa\def\csname\thmt@envname #1\thmt@autorefname\endcsname{#3}%
1164 \ifcsname #2refname\endcsname
1165 \csname #2refname\@xa\endcsname\@xa{\thmt@envname}{#3}{#4}%
1166 \fi
1167 }
1168 \define@key{thmdef}{refname}{\thmt@trytwice}{%
1169 \thmt@refnamewithcomma{a}{c}#1,\textbf{?? (pl. #1)},\@nil
1170 }}
1171 \define@key{thmdef}{Refname}{\thmt@trytwice}{%
1172 \thmt@refnamewithcomma{A}{C}#1,\textbf{?? (pl. #1)},\@nil
1173 }}
1174
1175
1176 \ifcsname Autoref\endcsname\else
1177 \let\thmt@HyRef@testreftype\HyRef@testreftype
1178 \def\HyRef@Testreftype#1.#2\{\%
1179 \ltx@ifundefined{#1Autorefname}{%
1180 \thmt@HyRef@testreftype#1.#2\%
1181 }{%
1182 \edef\HyRef@currentHtag{%
1183 \expandafter\noexpand\csname#1Autorefname\endcsname
1184 \noexpand~%

```

```

1185 }%
1186 }%
1187 }
1188
1189
1190 \let\thmt@HyPsd@@autorefname\HyPsd@@autorefname
1191 \def\HyPsd@@Autorefname#1.#2\@nil{%
1192   \tracingall
1193   \ltx@ifundefined{#1Autorefname}{%
1194     \thmt@HyPsd@@autorefname#1.#2\@nil
1195   }{%
1196     \csname#1Autorefname\endcsname\space
1197   }%
1198 }%
1199 \def\Autoref{%
1200   \parse{%
1201     {\parseFlag*{\def\thmt@autorefstar{*}}{\let\thmt@autorefstar\@empty}}%
1202     {\parseMand{%
1203       \bgroup
1204         \let\HyRef@testreftype\HyRef@Testreftype
1205         \let\HyPsd@@autorefname\HyPsd@@Autorefname
1206         \@xa\autoref\thmt@autorefstar{##1}%
1207       \egroup
1208       \let\@parsecmd\@empty
1209     }}%
1210   }%
1211 }
1212 \fi % ifcsname Autoref
1213
1214 % not entirely appropriate here, but close enough:
1215 \AtBeginDocument{%
1216   \@ifpackageloaded{nameref}{%
1217     \addtotheorempostheadhook{%
1218       \expandafter\NR@getttitle\expandafter{\thmt@shortoptarg}%
1219     }}{}
1220 }
1221
1222 \AtBeginDocument{%
1223   \@ifpackageloaded{cleveref}{%
1224     \@ifpackagelater{cleveref}{2010/04/30}{%
1225       % OK, new enough
1226     }{%
1227       \PackageWarningNoLine{thmtools}{%
1228         Your version of cleveref is too old!\MessageBreak
1229         Update to version 0.16.1 or later%
1230       }
1231     }
1232   }}{}
1233 }

```

A.2 Glue code for different backends

A.2.1 amsthm

```

1234 \providecommand\thmt@space{ }
1235
1236 \define@key{thmstyle}{spaceabove}{%
1237   \def\thmt@style@spaceabove{#1}%
1238 }
1239 \define@key{thmstyle}{spacebelow}{%

```

```

1240 \def\thmt@style@spacebelow{#1}%
1241 }
1242 \define@key{thmstyle}{headfont}{%
1243 \def\thmt@style@headfont{#1}%
1244 }
1245 \define@key{thmstyle}{bodyfont}{%
1246 \def\thmt@style@bodyfont{#1}%
1247 }
1248 \define@key{thmstyle}{notefont}{%
1249 \def\thmt@style@notefont{#1}%
1250 }
1251 \define@key{thmstyle}{headpunct}{%
1252 \def\thmt@style@headpunct{#1}%
1253 }
1254 \define@key{thmstyle}{notebraces}{%
1255 \def\thmt@style@notebraces{\thmt@embrace#1}%
1256 }
1257 \define@key{thmstyle}{break}[]{%
1258 \def\thmt@style@postheadspace{\newline}%
1259 }
1260 \define@key{thmstyle}{postheadspace}{%
1261 \def\thmt@style@postheadspace{#1}%
1262 }
1263 \define@key{thmstyle}{headindent}{%
1264 \def\thmt@style@headindent{#1}%
1265 }
1266
1267 \newtoks\thmt@style@headstyle
1268 \define@key{thmstyle}{headformat}[]{%
1269 \thmt@setheadstyle{#1}%
1270 }
1271 \define@key{thmstyle}{headstyle}[]{%
1272 \thmt@setheadstyle{#1}%
1273 }
1274 \def\thmt@setheadstyle#1{%
1275 \thmt@style@headstyle{%
1276 \def\NAME{\the\thm@headfont ##1}%
1277 \def\NUMBER{\bgroup\@upn{##2}\egroup}%
1278 \def\NOTE{\if=##3=\else\bgroup\thmt@space\the\thm@notefont(##3)\egroup\fi}%
1279 }%
1280 \def\thmt@tmp{#1}%
1281 \@onelevel@sanitize\thmt@tmp
1282 %\tracingall
1283 \ifcsname thmt@headstyle@\thmt@tmp\endcsname
1284 \thmt@style@headstyle\@xa{%
1285 \the\thmt@style@headstyle
1286 \csname thmt@headstyle@#1\endcsname
1287 }%
1288 \else
1289 \thmt@style@headstyle\@xa{%
1290 \the\thmt@style@headstyle
1291 #1%
1292 }%
1293 \fi
1294 %\showthe\thmt@style@headstyle
1295 }
1296 % examples:
1297 \def\thmt@headstyle@margin{%
1298 \makebox[Opt][r]{\NUMBER\ } \NAME\NOTE
1299 }
1300 \def\thmt@headstyle@swapnumber{%

```

```

1301 \NUMBER\ \NAME\NOTE
1302 }
1303
1304
1305
1306 \def\thmt@embrace#1#2(#3){#1#3#2}
1307
1308 \def\thmt@declaretheoremstyle@setup{%
1309 \let\thmt@style@notebraces\@empty%
1310 \thmt@style@headstyle{}}%
1311 \kvsetkeys{thmstyle}{%
1312 spaceabove=3pt,
1313 spacebelow=3pt,
1314 headfont=\bfseries,
1315 bodyfont=\normalfont,
1316 headpunct={.},
1317 postheadspace={ },
1318 headindent={},
1319 notefont={\fontseries\mddefault\upshape}
1320 }%
1321 }
1322 \def\thmt@declaretheoremstyle#1{%
1323 %\show\thmt@style@spaceabove
1324 \thmt@toks{\newtheoremstyle{#1}}%
1325 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@spaceabove}}%
1326 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@spacebelow}}%
1327 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@bodyfont}}%
1328 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@headindent}}% indent1 FIXM
1329 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@headfont}}%
1330 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@headpunct}}%
1331 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@postheadspace}}%
1332 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\the\thmt@style@headstyle}}% headspec
1333 \the\thmt@toks
1334 %1 Indent amount: empty = no indent, \parindent = normal paragraph indent
1335 %2 Space after theorem head: { } = normal interword space; \newline = linebreak
1336 %% BUGFIX: amsthm ignores notefont setting altogether:
1337 \thmt@toks\@xa\@xa\@xa{\csname th@#1\endcsname}%
1338 \thmt@toks
1339 \@xa\@xa\@xa\@xa\@xa\@xa\@xa{\%
1340 \@xa\@xa\@xa\@xa\@xa\@xa\@xa\thm@notefont
1341 \@xa\@xa\@xa\@xa\@xa\@xa\@xa{\%
1342 \@xa\@xa\@xa\thmt@style@notefont
1343 \@xa\thmt@style@notebraces
1344 \@xa}\the\thmt@toks}%
1345 \@xa\def\csname th@#1\@xa\endcsname\@xa{\the\thmt@toks}%
1346 % \@xa\def\csname th@#1\@xa\@xa\@xa\@xa\@xa\@xa\endcsname
1347 % \@xa\@xa\@xa\@xa\@xa\@xa\@xa{\%
1348 % \@xa\@xa\@xa\@xa\@xa\@xa\@xa\thm@notefont
1349 % \@xa\@xa\@xa\@xa\@xa\@xa\@xa{\%
1350 % \@xa\@xa\@xa\thmt@style@notefont
1351 % \@xa\@xa\@xa\thmt@style@notebraces
1352 % \@xa\@xa\@xa}\csname th@#1\endcsname
1353 % }
1354 }
1355
1356 \define@key{thmdef}{qed}{[\qedsymbol]}{%
1357 \thmt@trytwice{}}%
1358 \addtotheorempostheadhook[\thmt@envname]{%
1359 \protected@edef\qedsymbol{#1}%
1360 \pushQED{\qed}}%
1361 }%

```



```

1362 \addtotheoremprefhook[\thmt@envname]{%
1363 \protected@edef\qedsymbol{#1}%
1364 \popQED
1365 }%
1366 }%
1367 }
1368
1369 \def\thmt@amsthmlistbreakhack{%
1370 \leavevmode
1371 \vspace{-\baselineskip}%
1372 \par
1373 \everypar{\setbox\z@\lastbox\everypar{}}}%
1374 }
1375
1376 \define@key{thmuse}{listhack}[\relax]{%
1377 \addtotheorempostheadhook[local]{%
1378 \thmt@amsthmlistbreakhack
1379 }%
1380 }
1381

```

A.2.2 beamer

```

1382 \newif\ifthmt@hasoverlay
1383 \def\thmt@parsetheoremargs#1{%
1384 \parse{%
1385 {\parseOpt<>\thmt@hasoverlaytrue\def\thmt@overlay{##1}}}%
1386 {\parseOpt[]\def\thmt@optarg{##1}}}%
1387 \let\thmt@shortoptarg\@empty
1388 \let\thmt@optarg\@empty}}%
1389 {\ifthmt@hasoverlay\expandafter\@gobble\else\expandafter\@firstofone\fi
1390 {\parseOpt<>\thmt@hasoverlaytrue\def\thmt@overlay{##1}}}%
1391 }%
1392 {%
1393 \def\thmt@local@preheadhook{}%
1394 \def\thmt@local@postheadhook{}%
1395 \def\thmt@local@prefoothook{}%
1396 \def\thmt@local@postfoothook{}%
1397 \thmt@local@preheadhook
1398 \csname thmt@#1@preheadhook\endcsname
1399 \thmt@generic@preheadhook
1400 \protected@edef\tmp@args{%
1401 \ifthmt@hasoverlay <\thmt@overlay>\fi
1402 \ifx\@empty\thmt@optarg\else [{\thmt@optarg}]\fi
1403 }%
1404 \csname thmt@original@#1\@xa\endcsname\tmp@args
1405 \thmt@local@postheadhook
1406 \csname thmt@#1@postheadhook\endcsname
1407 \thmt@generic@postheadhook
1408 \let\@parsecmd\@empty
1409 }%
1410 }
1411 }%

```

A.2.3 ntheorem

```

1412
1413 \providecommand\thmt@space{ }
1414
1415 % actually, ntheorem's so-called style is nothing like a style at all...
1416 \def\thmt@declaretheoremstyle@setup{}
1417 \def\thmt@declaretheoremstyle#1{%

```

```

1418 \ifcsname th@#1\endcsname\else
1419 \@xa\let\csname th@#1\endcsname\th@plain
1420 \fi
1421 }
1422
1423 \def\thmt@notsupported#1#2{%
1424 \PackageWarning{thmtools}{Key ‘#2’ not supported by #1}{}%
1425 }
1426
1427 \define@key{thmstyle}{spaceabove}{%
1428 \setlength\theorempreskipamount{#1}%
1429 }
1430 \define@key{thmstyle}{spacebelow}{%
1431 \setlength\theorempostskipamount{#1}%
1432 }
1433 \define@key{thmstyle}{headfont}{%
1434 \theoremheaderfont{#1}%
1435 }
1436 \define@key{thmstyle}{bodyfont}{%
1437 \theorembodyfont{#1}%
1438 }
1439 % not supported in ntheorem.
1440 \define@key{thmstyle}{notefont}{%
1441 \thmt@notsupported{ntheorem}{notefont}%
1442 }
1443 \define@key{thmstyle}{headpunct}{%
1444 \theoremseparator{#1}%
1445 }
1446 % not supported in ntheorem.
1447 \define@key{thmstyle}{notebraces}{%
1448 \thmt@notsupported{ntheorem}{notebraces}%
1449 }
1450 \define@key{thmstyle}{break}{%
1451 \theoremstyle{break}%
1452 }
1453 % not supported in ntheorem...
1454 \define@key{thmstyle}{postheadspace}{%
1455 %\def\thmt@style@postheadspace{#1}%
1456 \@xa@g@addto@macro\csname thmt@style \thmt@style @defaultkeys\endcsname{%
1457 postheadhook={\hspace{-\labelsep}\hspace*{#1}},%
1458 }%
1459 }
1460
1461 % not supported in ntheorem
1462 \define@key{thmstyle}{headindent}{%
1463 \thmt@notsupported{ntheorem}{headindent}%
1464 }
1465 % sorry, only style, not def with ntheorem.
1466 \define@key{thmstyle}{qed}[\qedsymbol]{%
1467 \@ifpackagewith{ntheorem}{thmmarks}{%
1468 \theoremsymbol{#1}%
1469 }{%
1470 \thmt@notsupported
1471 {ntheorem without thmmarks option}%
1472 {headindent}%
1473 }%
1474 }
1475
1476 \let\@upn=\textup
1477 \define@key{thmstyle}{headformat}[]{}%
1478 \def\thmt@tmp{#1}%

```

```

1479 \@onelevel@sanitize\thmt@tmp
1480 %\tracingall
1481 \ifcsname thmt@headstyle@\thmt@tmp\endcsname
1482   \newtheoremstyle{\thmt@style}{%
1483     \item[\hskip\labelsep\theorem@headerfont%
1484       \def\NAME{\theorem@headerfont #####1}%
1485       \def\NUMBER{\bgroup\@upn{####2}\egroup}%
1486       \def\NOTE{}}%
1487     \csname thmt@headstyle@#1\endcsname
1488     \theorem@separator
1489   ]
1490   }{%
1491     \item[\hskip\labelsep\theorem@headerfont%
1492       \def\NAME{\theorem@headerfont #####1}%
1493       \def\NUMBER{\bgroup\@upn{####2}\egroup}%
1494       \def\NOTE{\if=####3=\else\bgroup\thmt@space(####3)\egroup\fi}%
1495       \csname thmt@headstyle@#1\endcsname
1496       \theorem@separator
1497     ]
1498   }
1499 \else
1500   \newtheoremstyle{\thmt@style}{%
1501     \item[\hskip\labelsep\theorem@headerfont%
1502       \def\NAME{\the\thm@headfont #####1}%
1503       \def\NUMBER{\bgroup\@upn{####2}\egroup}%
1504       \def\NOTE{}}%
1505     #1%
1506     \theorem@separator
1507   ]
1508   }{%
1509     \item[\hskip\labelsep\theorem@headerfont%
1510       \def\NAME{\the\thm@headfont #####1}%
1511       \def\NUMBER{\bgroup\@upn{####2}\egroup}%
1512       \def\NOTE{\if=####3=\else\bgroup\thmt@space(####3)\egroup\fi}%
1513       #1%
1514       \theorem@separator
1515     ]
1516   }
1517 \fi
1518 }
1519
1520 \def\thmt@headstyle@margin{%
1521   \makebox[Opt][r]{\NUMBER\ }\NAME\NOTE
1522 }
1523 \def\thmt@headstyle@swapnumber{%
1524   \NUMBER\ \NAME\NOTE
1525 }
1526
1527
1528

```

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.

```
1529
1530 \newtoks\@parsespec
1531 \def\parse@endquark{\parse@endquark}
1532 \newcommand\parse[1]{%
1533   \@parsespec{#1\parse@endquark}\@parse}
1534
1535 \newcommand\@parse{%
1536   \edef\p@tmp{\the\@parsespec}%
1537   \ifx\p@tmp\parse@endquark
1538     \expandafter\@gobble
1539   \else
1540     \typeout{parsespec remaining: \the\@parsespec}%
1541     \expandafter\@firstofone
1542   \fi{%
1543     \@parsepop
1544   }%
1545 }
1546 \def\@parsepop{%
1547   \expandafter\p@rsepop\the\@parsespec\@nil
1548   \@parsecmd
1549 }
1550 \def\p@rsepop#1#2\@nil{%
1551   #1%
1552   \@parsespec{#2}%
1553 }
1554
1555 \newcommand\parseOpt[4]{%
1556   %\parseOpt{openchar}{closechar}{yes}{no}
1557   \typeout{attempting #1#2...}%
1558   \def\@parsecmd{%
1559     \@ifnextchar#1{\@@reallyparse}{#4\@parse}%
1560   }%
1561   \def\@@reallyparse#1##1#2{%
1562     #3\@parse
1563   }%
1564 }
1565
1566 \newcommand\parseMand[1]{%
1567   %\parseMand{code}
1568   \def\@parsecmd##1{#1\@parse}%
1569 }
1570
1571 \newcommand\parseFlag[3]{%
1572   %\parseFlag{flagchar}{yes}{no}
1573   \def\@parsecmd{%
1574     \@ifnextchar#1{#2\expandafter\@parse\@gobble}{#3\@parse}%
1575   }%
1576 }
```

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`.

```
1577 \def\aliasctr@f@llow#1#2\@nil#3{%
1578   \ifx#1\@elt
```

```

1579 \noexpand #3%
1580 \else
1581 \expandafter\aliasctr@f@llow#1\@elt\@nil{#1}%
1582 \fi
1583 }
1584 \newcommand\aliasctr@follow[1]{%
1585 \expandafter\aliasctr@f@llow

```

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

```

1586 \csname cl@#1\endcsname\@elt\@nil{\csname cl@#1\endcsname}%
1587 }
1588 \renewcommand*\@addtoreset[2]{\bgroup
1589 \edef\aliasctr@@truelist{\aliasctr@follow{#2}}%
1590 \let\@elt\relax
1591 \expandafter\@cons\aliasctr@@truelist{{#1}}%
1592 \egroup}

```

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

```

1593 \RequirePackage{remreset}
1594 \renewcommand*\@removefromreset[2]{\bgroup
1595 \edef\aliasctr@@truelist{\aliasctr@follow{#2}}%
1596 \expandafter\let\csname c@#1\endcsname\@removefromreset
1597 \def\@elt##1{%
1598 \expandafter\ifx\csname c@##1\endcsname\@removefromreset
1599 \else
1600 \noexpand\@elt{##1}%
1601 \fi}%
1602 \expandafter\xdef\aliasctr@@truelist{%
1603 \aliasctr@@truelist}
1604 \egroup}

```

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

```

1605 \newcommand\@counteralias[2]{%
1606 \def\@@gletover##1##2{%
1607 \expandafter\global
1608 \expandafter\let\csname ##1\expandafter\endcsname
1609 \csname ##2\endcsname
1610 }%
1611 \@ifundefined{c@#2}{\@nocounterr{#2}}{%
1612 \@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.

```

1613 \@@gletover{c@#1}{c@#2}%
1614 \@@gletover{the#1}{the#2}%

```

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

```

1615 \@@gletover{theH#1}{theH#2}%
1616 \@@gletover{p@#1}{p@#2}%

```

```

1617     \expandafter\global
1618     \expandafter\def\csname cl@#1\expandafter\endcsname
1619     \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.

```

1620     %\@addtoreset{#1}{@ckpt}%
1621     }%
1622     }%
1623 }}

```

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.

```

1624
1625 \DeclareOption{unq}{%
1626   \newwrite\uniq@channel
1627   \InputIfFileExists{\jobname.unq}{\}{}%
1628   \immediate\openout\uniq@channel=\jobname.unq
1629   \AtEndDocument{%
1630     \immediate\closeout\uniq@channel%
1631   }
1632 }
1633 \DeclareOption{aux}{%
1634   \let\uniq@channel\@auxout
1635 }
1636

```

Call this with a name to set the corresponding uniqmark. The name must be suitable for `\csname`-constructs, i.e. fully expansible 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`.

```

1637 \newcommand\setuniqmark[1]{%
1638   \expandafter\ifx\csname uniq@now@#1\endcsname\relax
1639   \global\@namedef{uniq@now@#1}{\uniq@ONE}%
1640   \else
1641   \expandafter\ifx\csname uniq@now@#1\endcsname\uniq@MANY\else
1642   \immediate\write\uniq@channel{%
1643     \string\uniq@setmany{#1}%
1644   }%
1645   \ifuniq{#1}{%
1646     \uniq@warnnotunique{#1}%
1647   }{}%
1648   \fi
1649   \global\@namedef{uniq@now@#1}{\uniq@MANY}%
1650   \fi
1651 }

```

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?

```

1652 \newcommand\ifuniq[1]{%
1653   \expandafter\ifx\csname uniq@last@#1\endcsname\uniq@MANY
1654   \expandafter \@secondoftwo
1655   \else
1656   \expandafter\@firstoftwo
1657   \fi
1658 }

```

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

```

1659 \def\uniq@ONE{\uniq@ONE}
1660 \def\uniq@MANY{\uniq@MANY}

```

Flag: suggest a rerun?

```

1661 \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.

```

1662 \def\uniq@setmany#1{%
1663   \global\@namedef{uniq@last@#1}{\uniq@MANY}%
1664   \AtEndDocument{%
1665     \uniq@warnifunique{#1}%
1666   }%
1667 }

```

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.

```

1668 \def\uniq@warnifunique#1{%
1669   \expandafter\ifx\csname uniq@now@#1\endcsname\uniq@MANY\else
1670   \PackageWarningNoLine{uniq}{%
1671     '#1' is unique now.\MessageBreak
1672     Rerun LaTeX to pick up the change%
1673   }%
1674   \@uniq@reruntrue
1675   \fi
1676 }

```

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.

```

1677 \def\uniq@warnnotunique#1{%
1678   \PackageWarningNoLine{uniq}{%
1679     '#1' is not unique anymore.\MessageBreak
1680     Rerun LaTeX to pick up the change%
1681   }%
1682   \@uniq@reruntrue
1683 }

```

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.

```

1684 \def\uniq@maybesuggestrerun{%
1685   \if@uniq@rerun
1686   \PackageWarningNoLine{uniq}{%
1687     Uniquenesses have changed. \MessageBreak
1688     Rerun LaTeX to pick up the change%
1689   }%
1690   \fi
1691 }

```

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

```

1692 \AtEndDocument{%
1693   \immediate\write\@auxout{\string\uniq@maybesuggestrerun}%

```

```
1694 }  
1695 \ExecuteOptions{aux}  
1696 \ProcessOptions\relax
```