

Thmtools Users' Guide

2008–2014 Dr. Ulrich M. Schwarz – ulmi@absatzen.de*

2020– Yukai Chou

2022/06/01 v0.73

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

Abstract

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

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

Contents

1 Thmtools for the impatient	2	3.5 Restatable – hints and caveats	16
1.1 Elementary definitions	2	A Thmtools for the morbidly curious	17
1.2 Frilly references	4	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	7	A.1.4 Lists of theorems	31
1.6 Extended arguments to theorem environments	9	A.1.5 Re-using environments	34
2 Thmtools for the extravagant	10	A.1.6 Restrictions	35
2.1 Understanding thmtools' extension mechanism	10	A.1.7 Fixing <code>autoref</code> and friends . .	39
2.2 Case in point: the <code>shaded</code> key	10	A.2 Glue code for different backends	41
2.3 Case in point: the <code>thmbox</code> key	12	A.2.1 <code>amsthm</code>	41
2.4 Case in point: the <code>mdframed</code> key	12	A.2.2 <code>beamer</code>	43
2.5 How thmtools finds your extensions . . .	12	A.2.3 <code>ntheorem</code>	44
3 Thmtools for the completionist	13	A.3 Generic tools	46
3.1 Known keys to <code>\declaretheoremstyle</code>	13	A.3.1 A generalized argument parser .	46
3.2 Known keys to <code>\declaretheorem</code> . .	14	A.3.2 Different counters sharing the same register	47
3.3 Known keys to in-document theorems .	15	A.3.3 Tracking occurrences: none, one or many	48
3.4 Known keys to <code>\listoftheorems</code> . .	15		

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

1 Thmtools for the impatient

How to use this document

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

% Preamble code looks like this.

```
\usepackage{amsthm}
\usepackage{thmtools}
\declaretheorem{theorem}
```

% Document code looks like this.

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

The result looks like this:

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

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

is infinite.

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

1.1 Elementary definitions

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

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

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

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

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

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

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

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

Übung 1. *Prove Euclid's Theorem.*

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

Of course, you do not have to follow the abominal 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}
```

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

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

1.2 Frilly references

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

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

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

Theorem 1 (Simon). *One*

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

1.3 Styling theorems

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

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

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

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

Remark 1. This is a remark.

Thmtools also supports the shadethm and thmbox packages:

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

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

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

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

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

For the thmbox package, use the thmbox key:

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

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

Boxtheorem L 1 (Euclid)

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

Boxtheorem M 1 (Euclid)

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

Boxtheorem S 1 (Euclid)

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

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

1.3.1 Declaring new theoremstyles

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

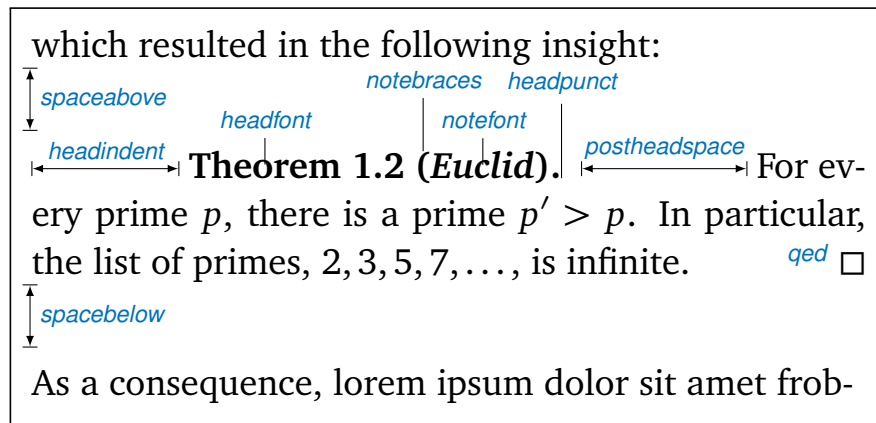


Figure 1.1: Settable parameters of a theorem style.

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

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

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

1.4 Repeating theorems

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

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

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

and to the right, I just use

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

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

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

is infinite.

⋮

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

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

is infinite.

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

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

```
\euclidii*
```

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

1.5 Lists of theorems

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

`\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	5
1	BoxII	5
1	Boxtheorem L (Euclid) . . .	5
1	Boxtheorem M (Euclid) . .	5
1	Boxtheorem S (Euclid) . . .	5
1	Styledtheorem (Euclid) . .	6
1	Theorem (Euclid)	7
1	Theorem (Euclid)	7
1.1.1	TheoremS (Euclid)	7
3	Theorem (Keyed theorem)	9
3	Theorem (continuing from p. 9)	9
4	Lemma (Zorn)	35
5	Lemma	35
4	Lemma (Zorn)	35

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

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

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

List of Theorems

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

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

1.6 Extended arguments to theorem environments

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

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

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

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

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

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

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

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

2 Thmtools for the extravagant

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

2.1 Understanding thmtools’ extension mechanism

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

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

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

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

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

2.2 Case in point: the shaded key

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

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

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

The docs for shadethm say:

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

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

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

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

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

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

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

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

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

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

```
30 \def\thmt@drop@relax#1\relax{}
31 \def\thmt@definecolor#1#2{%
32   \thmt@def@color{#1}#2\thmt@drop@relax
33   {gray}{0.5}%
34   \thmt@colorlet{#1}{#2}%
35   \relax
36 }
37 \def\thmt@def@color#1#2#3{%
38   \definecolor{#1}}
```

2.3 Case in point: the **thmbox** key

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

```
39 \define@key{thmdef}{thmbox}[L]{%
40   \thmt@trytwice{%
41     \let\oldproof=\proof
42     \let\oldendproof=\endproof
43     \let\oldexample=\example
44     \let\oldendexample=\endexample
45     \RequirePackage[nothm]{thmbox}
46     \let\proof=\oldproof
47     \let\endproof=\oldendproof
48     \let\example=\oldexample
49     \let\endexample=\oldendexample
50     \def\thmt@theoremdefiner{\newboxtheorem[#1]}%
51   }{}%
52 }
```

2.4 Case in point: the **mdframed** key

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

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

2.5 How thmtools finds your extensions

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

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

For example, there is no provision in thmtools itself that make the `shaded` and `thmbox` keys described above special: in fact, if you want to use a different package to create frames, you just put a different `thmdef-shaded.sty` into a preferred texmf tree. Of course, if your new package doesn't offer the old keys, your old documents might break!

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

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

3 Thmtools for the completionist

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

3.1 Known keys to `\declaretheoremstyle`

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This list is likely to grow

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

3.2 Known keys to `\declaretheorem`

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

numberwithin (Same as `parent`.)

within (Same as `parent`.)

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

numberlike (Same as `sibling`.)

sharenumber (Same as `sibling`.)

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

name (Same as `title`.)

heading (Same as `title`.)

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

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

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

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

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

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

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

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

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

textwidth The linewidth within the theorem.

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

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

rulewidth The width of the box surrounding the theorem.

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

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

3.3 Known keys to in-document theorems

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

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

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

3.4 Known keys to `\listoftheorems`

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

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

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

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

showall The opposite effect of `ignoreall`.

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

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

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

List of Theorems

4	Lemma (Zorn)	35
4	Lemma (Zorn)	35

List of Theorems

Lemma 4 (Zorn)	35
Lemma 4 (Zorn)	35

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

3.5 Restatable – hints and caveats

TBD.

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

A Thmtools for the morbidly curious

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

A.1 Core functionality

A.1.1 The main package

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

```

107     }{}%
108   }{}%
109 }
110 \@ifclassloaded{beamer}{%
111   \RequirePackage{thm-beamer}
112 }{}
113 \@ifclassloaded{llnccs}{%
114   \RequirePackage{thm-llnccs}
115 }{}

```

A.1.2 Adding hooks to the relevant commands

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

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

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

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

```

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

```

```

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

```

```

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

```

```

268 \newcommand\addtotheoremproofhook[1][generic]{%
269   \expandafter\g@prependto@macro\csname thmt@#1@proofhook\endcsname%
270 }
271

```

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.

```

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

```

A.1.3 The key-value interfaces

```

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

```

```

323
324 \DeclareOption{anycase}{%
325   \PackageInfo{thm-kv}{Theorem names will be unchanged}%
326   \global\let\thmt@modifycase\@empty}
327
328 \ExecuteOptions{uppercase}
329 \ProcessOptions\relax
330
331 \RequirePackage{keyval,kvsetkeys,thm-patch}
332
333 \long\def\thmt@kv@processor@default#1#2#3{%
334   \def\kvsu@fam{#1}% new
335   \@onelevel@sanitize\kvsu@fam% new
336   \def\kvsu@key{#2}% new
337   \@onelevel@sanitize\kvsu@key% new
338   \unless\ifcsname KV@#1@\kvsu@key\endcsname
339     \unless\ifcsname KVS@#1@handler\endcsname
340       \kv@error@unknownkey{#1}{\kvsu@key}%
341     \else
342       \csname KVS@#1@handler\endcsname{#2}{#3}%
343       % still using #2 #3 here is intentional: handler might
344       % be used for strange stuff like implementing key names
345       % that contain strange characters or other strange things.
346       \relax
347     \fi
348   \else
349     \ifx\kv@value\relax
350       \unless\ifcsname KV@#1@\kvsu@key @default\endcsname
351         \kv@error@novalue{#1}{\kvsu@key}%
352       \else
353         \csname KV@#1@\kvsu@key @default\endcsname
354         \relax
355       \fi
356     \else
357       \csname KV@#1@\kvsu@key\endcsname{#3}%
358     \fi
359   \fi
360 }
361
362 \@ifpackagelater{kvsetkeys}{2012/04/23}{%
363   \PackageInfo{thm-kv}{kvsetkeys patch (v1.16 or later)}%
364   \long\def\tmp@KVS@PD#1#2#3{%
365     \def\kv@fam{#1}%
366     \unless\ifcsname KV@#1@#2\endcsname
367       \unless\ifcsname KVS@#1@handler\endcsname
368         \kv@error@unknownkey{#1}{#2}%
369       \else
370         \kv@handled@true
371         \csname KVS@#1@handler\endcsname{#2}{#3}\relax
372         \ifkv@handled@ \else
373           \kv@error@unknownkey{#1}{#2}%
374         \fi
375       \fi
376     \else
377       \ifx\kv@value\relax
378         \unless\ifcsname KV@#1@#2@default\endcsname
379           \kv@error@novalue{#1}{#2}%
380         \else
381           \csname KV@#1@#2@default\endcsname \relax
382         \fi
383       \else

```

```

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

```

```

445 }}
446
447 % useful key handler defaults.
448 \newcommand\thmt@mkignoringkeyhandler[1]{%
449   \kv@set@family@handler{#1}{%
450     \thmt@debug{Key '##1' with value '##2' ignored by #1.}%
451   }%
452 }
453 \newcommand\thmt@mkextendingkeyhandler[3]{%
454 % #1: family
455 % #2: prefix for file
456 % #3: key hint for error
457 \kv@set@family@handler{#1}{%
458   \thmt@selfextendingkeyhandler{#1}{#2}{#3}%
459   {##1}{##2}%
460 }%
461 }
462
463 \newcommand\thmt@selfextendingkeyhandler[5]{%
464 % #1: family
465 % #2: prefix for file
466 % #3: key hint for error
467 % #4: actual key
468 % #5: actual value
469 \IfFileExists{#2-#4.sty}{%
470   \PackageInfo{thmtools}%
471   {Automatically pulling in '#2-#4'}%
472   \RequirePackage{#2-#4}%
473   \ifcsname KV@#1@#4\endcsname
474   \csname KV@#1@#4\endcsname{#5}%
475   \else
476     \PackageError{thmtools}%
477     {#3 '#4' not known}
478     {I don't know what that key does.\MessageBreak
479      I've even loaded the file '#2-#4.sty', but that didn't help.
480     }%
481   \fi
482 }{%
483   \PackageError{thmtools}%
484   {#3 '#4' not known}
485   {I don't know what that key does by myself,\MessageBreak
486    and no file '#2-#4.sty' to tell me seems to exist.
487   }%
488 }%
489 }
490
491
492 \newif\if@thmt@firstkeyset
493
494 % many keys are evaluated twice, because we don't know
495 % if they make sense before or after, or both.
496 \def\thmt@trytwice{%
497   \if@thmt@firstkeyset
498     \@xa\@firstoftwo
499   \else
500     \@xa\@secondoftwo
501   \fi
502 }
503
504 \@for\tmp@keyname:=parent,numberwithin,within\do{%
505   \define@key{thmdef}{\tmp@keyname}{%

```



```

506 \thmt@trytwice{%
507 \thmt@setparent{#1}
508 \thmt@setsibling{}}%
509 }{}%
510 }%
511 }
512 \newcommand\thmt@setparent{%
513 \def\thmt@parent
514 }
515
516 \@for\tmp@keyname:=sibling,numberlike,sharenumber\do{%
517 \define@key{thmdef}{\tmp@keyname}{%
518 \thmt@trytwice{%
519 \thmt@setsibling{#1}%
520 \thmt@setparent{}}%
521 }{}%
522 }%
523 }
524 \newcommand\thmt@setsibling{%
525 \def\thmt@sibling
526 }
527
528 \@for\tmp@keyname:=title,name,heading\do{%
529 \define@key{thmdef}{\tmp@keyname}{\thmt@trytwice{\thmt@setthmname{#1}}{}}%
530 }
531 \newcommand\thmt@setthmname{%
532 \def\thmt@thmname
533 }
534
535 \@for\tmp@keyname:=unnumbered,starred\do{%
536 \define@key{thmdef}{\tmp@keyname}[]{\thmt@trytwice{\thmt@isnumberedfalse}{}}%
537 }
538
539 \def\thmt@YES{yes}
540 \def\thmt@NO{no}
541 \def\thmt@UNIQUE{unless unique}
542 \newif\ifthmt@isnumbered
543 \newif\ifthmt@isunlesunique
544
545 \define@key{thmdef}{numbered}[yes]{
546 \def\thmt@tmp{#1}%
547 \thmt@trytwice{%
548 \ifx\thmt@tmp\thmt@YES
549 \thmt@isnumberedtrue
550 \else\ifx\thmt@tmp\thmt@NO
551 \thmt@isnumberedfalse
552 \else\ifx\thmt@tmp\thmt@UNIQUE
553 \RequirePackage[unq]{unique}
554 \thmt@isunlesuniquetrue
555 \else
556 \PackageError{thmtools}{Unknown value ‘#1’ to key numbered}{}%
557 \fi\fi\fi
558 }{}% trytwice: after definition
559 \ifx\thmt@tmp\thmt@UNIQUE
560 \ifx\thmt@parent\@empty
561 \addtotheorempreheadhook[\thmt@envname]{\setuniqmark{\thmt@envname}}%
562 \else
563 \protected@edef\thmt@tmp{%
564 % expand \thmt@envname and \thmt@parent
565 \@nx\addtotheorempreheadhook[\thmt@envname @unique]{\@nx\setuniqmark{\thmt@envname}}
566 \@nx\addtotheorempreheadhook[\thmt@envname @numbered]{\@nx\setuniqmark{\thmt@envname}}

```

```

567 \@nx\addtotheorempreheadhook[\thmt@envname @unique]{\def\@nx\thmt@dummyctratore
568 \@nx\addtotheorempreheadhook[\thmt@envname @numbered]{\def\@nx\thmt@dummyctratore
569 }%
570 \thmt@tmp
571 \fi
572 % \addtotheorempreheadhook[\thmt@envname]{\def\thmt@dummyctratorefname{\thmt@thmname
573 \fi
574 }%
575 }
576
577
578 \define@key{thmdef}{preheadhook}{%
579 \thmt@trytwice{}{\addtotheorempreheadhook[\thmt@envname]{#1}}
580 \define@key{thmdef}{postheadhook}{%
581 \thmt@trytwice{}{\addtotheorempostheadhook[\thmt@envname]{#1}}
582 \define@key{thmdef}{prefoothook}{%
583 \thmt@trytwice{}{\addtotheoremprefoothook[\thmt@envname]{#1}}
584 \define@key{thmdef}{postfoothook}{%
585 \thmt@trytwice{}{\addtotheorempostfoothook[\thmt@envname]{#1}}
586
587 \define@key{thmdef}{style}{\thmt@trytwice{\thmt@setstyle{#1}}{}}
588
589 % ugly hack: style needs to be evaluated first so its keys
590 % are not overridden by explicit other settings
591 \define@key{thmdef0}{style}{%
592 \ifcsname thmt@style #1@defaultkeys\endcsname
593 \thmt@toks{\kvsetkeys{thmdef}}%
594 \@xa\@xa\@xa\the\@xa\@xa\@xa\thmt@toks\@xa\@xa\@xa{%
595 \csname thmt@style #1@defaultkeys\endcsname}%
596 \fi
597 }
598 \thmt@mkignoringkeyhandler{thmdef0}
599
600 % fallback definition.
601 % actually, only the kernel does not provide \theoremstyle.
602 % is this one worth having glue code for the theorem package?
603 \def\thmt@setstyle#1{%
604 \PackageWarning{thm-kv}{%
605 Your backend doesn't have a '\string\theoremstyle' command.
606 }%
607 }
608
609 \ifcsname theoremstyle\endcsname
610 \let\thmt@originalthmstyle\theoremstyle
611 \def\thmt@outerstyle{plain}
612 \renewcommand\theoremstyle[1]{%
613 \def\thmt@outerstyle{#1}%
614 \thmt@originalthmstyle{#1}%
615 }
616 \def\thmt@setstyle#1{%
617 \thmt@originalthmstyle{#1}%
618 }
619 \g@addto@macro\thmt@newtheorem@postdefinition{%
620 \thmt@originalthmstyle{\thmt@outerstyle}%
621 }
622 \fi
623
624
625 \thmt@mkextendingkeyhandler{thmdef}{thmdef}{\string\declaretheorem\space key}
626
627 \let\thmt@newtheorem\newtheorem

```

```

628
629 % \declaretheorem[option list 1]{thmname list}[option list 1]
630 % #1 = option list 1
631 % #2 = thmname list
632 \newcommand\declaretheorem[2][]{%
633 % TODO: use \NewDocumentCommand from xparse?
634 % xparse will be part of latex2e format from latex2e 2020 Oct.
635 \@ifnextchar[%
636   {\declaretheorem@i{#1}{#2}}
637   {\declaretheorem@i{#1}{#2}[]}]%
638 }
639 \@onlypreamble\declaretheorem
640
641 % #1 = option list 1
642 % #2 = thmname list
643 % #3 = option list 2
644 \def\declaretheorem@i#1#2[#3]{%
645   \@for\thmt@tmp:=#2\do{%
646     % strip spaces, \KV@@sp@def is defined in keyval.sty
647     \@xa\KV@@sp@def\@xa\thmt@tmp\@xa{\thmt@tmp}%
648     \@xa\declaretheorem@ii\@xa{\thmt@tmp}{#1,#3}%
649   }%
650 }
651
652 % #1 = single thmname (#1 and #2 are exchanged)
653 % #2 = option list
654 \def\declaretheorem@ii#1#2{%
655   % why was that here?
656   %\let\thmt@theoremdefiner\thmt@original@newtheorem
657   % init options
658   \thmt@setparent{}%
659   \thmt@setsibling{}%
660   \thmt@isnumberedtrue
661   \thmt@isunlessuniquefalse
662   \def\thmt@envname{#1}%
663   \thmt@setthmname{\thmt@modifycase #1}%
664   % use true code in \thmt@trytwice{<true>}{<false>}
665   \@thmt@firstkeysettrue
666   % parse options
667   \kvsetkeys{thmdef0}{#2}% parse option "style" first
668   \kvsetkeys{thmdef}{#2}%
669   % call patched \newtheorem
670   \ifthmt@isunlessunique
671     \ifx\thmt@parent\@empty
672       % define normal "unless unique" thm env
673       \ifuniq{#1}{\thmt@isnumberedfalse}{\thmt@isnumberedtrue}%
674       \declaretheorem@iii{#1}%
675     \else
676       % define special "unless unique" thm env,
677       % when "numbered=unless unique" and "numberwithin=<counter>" are both used
678       \declaretheorem@iv{#1}%
679       \thmt@isnumberedtrue
680       \declaretheorem@iii{#1@numbered}%
681       \thmt@isnumberedfalse
682       \declaretheorem@iii{#1@unique}%
683     \fi
684   \else
685     % define normal thm env
686     \declaretheorem@iii{#1}%
687   \fi
688   % use false code in \thmt@trytwice{<true>}{<false>}

```

```

689 \def\thmt@envname{#1}%
690 \@thmt@firstkeysetfalse
691 % uniquely ugly kludge: some keys make only sense afterwards.
692 % and it gets kludgier: again, the default-inherited
693 % keys need to have a go at it.
694 \kvsetkeys{thmdef0}{#2}%
695 \kvsetkeys{thmdef}{#2}%
696 }
697
698 % define normal thm env, call \thmt@newtheorem
699 \def\declaretheorem@iii#1{%
700   \protected@edef\thmt@tmp{%
701     \nx\thmt@newtheorem
702     \ifthmt@isnumbered
703       {#1}%
704       \ifx\thmt@sibling\@empty\else [\thmt@sibling]\fi
705       {\thmt@thmname}%
706       \ifx\thmt@parent\@empty\else [\thmt@parent]\fi
707     \else
708       *{#1}{\thmt@thmname}%
709     \fi
710     \relax% added so we can delimited-read everything later
711   }%
712   \thmt@debug{Define theorem ‘#1’ by ^^J\meaning\thmt@tmp}%
713   \thmt@tmp
714 }
715
716 % define special thm env
717 \def\declaretheorem@iv#1{%
718   \protected@edef\thmt@tmp{%
719     % expand \thmt@envname and \thmt@parent
720     \nx\newenvironment{#1}{%
721       \nx\ifuniq{\thmt@envname.\nx\@nameuse{the\thmt@parent}}{%
722         \def\nx\thmt@rawenvname{#1@unique}%
723       }{%
724         \def\nx\thmt@rawenvname{#1@numbered}%
725       }%
726       \begin{\nx\thmt@rawenvname}%
727     }{%
728       \end{\nx\thmt@rawenvname}%
729     }%
730   }%
731   \thmt@debug{Define special theorem ‘#1’ by ^^J\meaning\thmt@tmp}%
732   \thmt@tmp
733 }
734
735 \providecommand\thmt@quark{\thmt@quark}
736
737 % in-document keyval, i.e. \begin{theorem}[key=val,key=val]
738
739 \thmt@mkextendingkeyhandler{thmuse}{thmuse}{\thmt@envname\space optarg key}
740
741 \addtotheoremreheadhook{%
742   \ifx\thmt@optarg\@empty\else
743     \@xa\thmt@garbleoptarg\@xa{\thmt@optarg}\fi
744 }%
745
746 \newif\ifthmt@thmuse@iskv
747
748 \providecommand\thmt@garbleoptarg[1]{%
749   \thmt@thmuse@iskvfalse

```

```

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

```

```

811 \def\thmt@newoptarg{#1\@iden}}
812
813 \providecommand\thmt@suspendcounter[2]{%
814   \@xa\protected@edef\csname the#1\endcsname{#2}%
815   \@xa\let\csname c@#1\endcsname\c@thmt@dummyctr
816 }
817
818 \providecommand\thmcontinues[1]{%
819   \ifcsname hyperref\endcsname
820     \hyperref[#1]{continuing}
821   \else
822     continuing
823   \fi
824   from p.\,\pageref{#1}%
825 }
826
827 \thmt@define@thmuse@key{continues}{%
828   \thmt@suspendcounter{\thmt@envname}{\thmt@trivialref{#1}{??}}%
829   \g@addto@macro\thmt@newoptarg{{, }%
830     \thmcontinues{#1}%
831     \@iden}%
832 }
833
834

```

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.

```

835 \def\thmt@declaretheoremstyle@setup{}
836 \def\thmt@declaretheoremstyle#1{%
837   \PackageWarning{thmtools}{Your backend doesn't allow styling theorems}{}
838 }
839 \newcommand\declaretheoremstyle[2][[]]{%
840   \def\thmt@style{#2}%
841   \@xa\def\csname thmt@style \thmt@style @defaultkeys\endcsname{}%
842   \thmt@declaretheoremstyle@setup
843   \kvsetkeys{thmstyle}{#1}%
844   \thmt@declaretheoremstyle{#2}%
845 }
846 \@onlypreamble\declaretheoremstyle
847
848 \kv@set@family@handler{thmstyle}{%
849   \@onelevel@sanitize\kv@value
850   \@onelevel@sanitize\kv@key
851   \PackageInfo{thmtools}{%
852     Key '\kv@key' (with value '\kv@value')\MessageBreak
853     is not a known style key.\MessageBreak
854     Will pass this to every \string\declaretheorem\MessageBreak
855     that uses 'style=\thmt@style'%
856   }%
857   \ifx\kv@value\relax% no value given, don't pass on {}!
858     \@xa\g@addto@macro\csname thmt@style \thmt@style @defaultkeys\endcsname{%
859       #1,%
860     }%
861   \else
862     \@xa\g@addto@macro\csname thmt@style \thmt@style @defaultkeys\endcsname{%
863       #1={#2},%
864     }%
865   \fi
866 }

```

A.1.4 Lists of theorems

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

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

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

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

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

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

ignoreall

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

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

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

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

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

```
867 \let\@xa=\expandafter
868 \let\@nx=\noexpand
869 \RequirePackage{thm-patch,keyval,kvsetkeys}
870
871 \def\thmtlo@oldchapter{0}%
872 \newcommand\thmtlo@chaptervspacehack{}
873 \ifcsname c@chapter\endcsname
874   \ifx\c@chapter\relax\else
875     \def\thmtlo@chaptervspacehack{%
876       \ifnum \value{chapter}=\thmtlo@oldchapter\relax\else
877         % new chapter, add vspace to loe.
878         \addtocontents{loe}{\protect\addvspace{10\p@}}%
879         \xdef\thmtlo@oldchapter{\arabic{chapter}}%
880       \fi
881     }%
882   \fi
883 \fi
884
885
886 \providecommand\listtheoremname{List of Theorems}
887 \newcommand\listoftheorems[1][1]{%
888   %% much hacking here to pick up the definition from the class
889   %% without oodles of conditionals.
890   \begingroup
891     \setlisttheoremstyle{#1}%
892     \let\listfigurename\listtheoremname
893     \def\contentsline##1{%
894       \csname thmt@contentsline@##1\endcsname{##1}%
895     }
```

```

895 }%
896 \@for\thmt@envname:=\thmt@allenvs\do{%
897   % CHECK: is \cs{l@\thmt@envname} repeatedly defined?
898   \thmtlo@newentry
899 }%
900 \let\thref@starttoc\@starttoc
901 \def\@starttoc##1{\thref@starttoc{loe}}%
902 % new hack: to allow multiple calls, we defer the opening of the
903 % loe file to AtEndDocument time. This is before the aux file is
904 % read back again, that is early enough.
905 % TODO: is it? crosscheck include/includeonly!
906 \@fileswfalse
907 \AtEndDocument{%
908   \if@filesw
909     \ifundefined{tf@loe}{%
910       \expandafter\newwrite\csname tf@loe\endcsname
911       \immediate\openout \csname tf@loe\endcsname \jobname.loe\relax
912     }{}%
913   \fi
914 }%
915 %\expandafter
916 \listoffigures
917 \endgroup
918 }
919
920 \newcommand\setlisttheoremstyle[1]{%
921   \kvsetkeys{thmt-listof}{#1}%
922 }
923 \define@key{thmt-listof}{numwidth}{\def\thmt@listnumwidth{#1}}
924 \define@key{thmt-listof}{ignore}{\thmt@allenvs}{\ignoretheorems{#1}}
925 \define@key{thmt-listof}{onlynamed}{\thmt@allenvs}{\onlynamedtheorems{#1}}
926 \define@key{thmt-listof}{show}{\thmt@allenvs}{\showtheorems{#1}}
927 \define@key{thmt-listof}{ignoreall}[true]{\ignoretheorems{\thmt@allenvs}}
928 \define@key{thmt-listof}{showall}[true]{\showtheorems{\thmt@allenvs}}
929 % FMI 2019-09-31 allow local title
930 \define@key{thmt-listof}{title}{\def\listtheoremname{#1}}
931 % -- FMI
932 \newif\ifthmt@listswap
933 \def\thmt@TRUE{true}
934 \def\thmt@FALSE{false}
935 \define@key{thmt-listof}{swapnumber}[true]{%
936   \def\thmt@tmp{#1}%
937   \ifx\thmt@tmp\thmt@TRUE
938     \thmt@listswaptrue
939   \else\ifx\thmt@tmp\thmt@FALSE
940     \thmt@listswapfalse
941   \else
942     \PackageError{thmttools}{Unknown value '#1' to key swapnumber}{}%
943   \fi\fi
944 }
945
946 \ifdefined\@tocline
947   % for ams classes (amsart.cls, amsproc.cls, amsbook.cls) which
948   % don't use \@dottedtocline and don't provide \@dotsep
949   \def\thmtlo@newentry{%
950     \@xa\def\csname l@\thmt@envname\endcsname{% CHECK: why p@edef?
951       % similar to \l@figure defined in ams classes
952       \@tocline{0}{3pt plus2pt}{0pt}{\thmt@listnumwidth}}{}%
953   }%
954 }
955 \providecommand*\thmt@listnumwidth{1.5pc}

```



```

956 \else
957   \def\thmtlo@newentry{%
958     \@xa\def\csname ll@\thmt@envname\endcsname{% CHECK: why p@edef?
959       \@dottedtocline{1}{1.5em}{\thmt@listnumwidth}%
960     }%
961   }
962   \providecommand*\thmt@listnumwidth{2.3em}
963 \fi
964
965 \providecommand\thmtformatoptarg[1]{ (#1)}
966
967 \newcommand\thmt@mklistcmd{%
968   \thmtlo@newentry
969   \ifthmt@isstarred
970     \@xa\def\csname ll@\thmt@envname\endcsname{%
971       \protect\ifthmt@listswap
972       \protect\else
973         \protect\numberline{\protect\let\protect\autodot\protect\@empty}%
974       \protect\fi
975       \thmt@thmname
976       \ifx\@empty\thmt@shortoptarg\else\protect\thmtformatoptarg{\thmt@shortoptarg}\fi
977     }%
978   \else
979     \@xa\def\csname ll@\thmt@envname\endcsname{%
980       \protect\ifthmt@listswap
981         \thmt@thmname~\csname the\thmt@envname\endcsname
982       \protect\else
983         \protect\numberline{\csname the\thmt@envname\endcsname}%
984         \thmt@thmname
985       \protect\fi
986       \ifx\@empty\thmt@shortoptarg\else\protect\thmtformatoptarg{\thmt@shortoptarg}\fi
987     }%
988   \fi
989   \@xa\gdef\csname thmt@contentsline@\thmt@envname\endcsname{%
990     \thmt@contentslineShow% default:show
991   }%
992 }
993 \def\thmt@allenvs{\@gobble}
994 \newcommand\thmt@recordenvname{%
995   \edef\thmt@allenvs{\thmt@allenvs,\thmt@envname}%
996 }
997 \g@addto@macro\thmt@newtheorem@predefinition{%
998   \thmt@mklistcmd
999   \thmt@recordenvname
1000 }
1001
1002 \addtotheoremheadhook{%
1003   \thmtlo@chaptervspacehack
1004   \addcontentsline{loe}{\thmt@envname}{%
1005     \csname ll@\thmt@envname\endcsname
1006   }%
1007 }
1008
1009 \newcommand\showtheorems[1]{%
1010   \@for\thmt@thm:=#1\do{%
1011     \typeout{showing \thmt@thm}%
1012     \@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
1013       =\thmt@contentslineShow
1014   }%
1015 }
1016

```

```

1017 \newcommand\ignoretheorems[1]{%
1018   \@for\thmt@thm:=#1\do{%
1019     \@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
1020     =\thmt@contentslineIgnore
1021   }%
1022 }
1023 \newcommand\onlynamedtheorems[1]{%
1024   \@for\thmt@thm:=#1\do{%
1025     \global\@xa\let\csname thmt@contentsline@\thmt@thm\endcsname
1026     =\thmt@contentslineIfNamed
1027   }%
1028 }
1029
1030 \AtBeginDocument{%
1031 \ifpackageloaded{hyperref}{%
1032   \let\thmt@hygobble\@gobble
1033 }{%
1034   \let\thmt@hygobble\@empty
1035 }
1036 \let\thmt@contentsline\contentsline
1037 }
1038
1039 \def\thmt@contentslineIgnore#1#2#3{%
1040   \thmt@hygobble
1041 }
1042 \def\thmt@contentslineShow{%
1043   \thmt@contentsline
1044 }
1045
1046 \def\thmt@contentslineIfNamed#1#2#3{%
1047   \thmt@ifhasoptname #2\thmtformatoptarg\@nil{%
1048     \thmt@contentslineShow{#1}{#2}{#3}%
1049   }{%
1050     \thmt@contentslineIgnore{#1}{#2}{#3}%
1051     %\thmt@contentsline{#1}{#2}{#3}%
1052   }
1053 }
1054
1055 \def\thmt@ifhasoptname #1\thmtformatoptarg#2\@nil{%
1056   \ifx\@nil#2\@nil
1057     \@xa\@secondoftwo
1058   \else
1059     \@xa\@firstoftwo
1060   \fi
1061 }

```

A.1.5 Re-using environments

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

```

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

```

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

A.1.6 Restrictions

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

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

```
1062 \RequirePackage{thmtools}
1063 \let\@xa\expandafter
1064 \let\@nx\noexpand
1065 \@ifundefined{c@thmt@dummyctr}{%
1066   \newcounter{thmt@dummyctr}%
1067 }{}
1068 \gdef\theHthmt@dummyctr{dummy.\arabic{thmt@dummyctr}}%
1069 \gdef\thethmt@dummyctr{}%
1070 \long\def\thmt@collect@body#1#2\end#3{%
1071   \@xa\thmt@toks\@xa{\the\thmt@toks #2}%
1072   \def\thmttmpa{#3}%\def\thmttmpb{restatable}%
1073   \ifx\thmttmpa\@currenvir\thmttmpb
1074     \@xa\@firstoftwo% this is the end of the environment.
1075   \else
1076     \@xa\@secondoftwo% go on collecting
1077     \fi{% this is the end, my friend, drop the \end.
1078     % and call #1 with the collected body.
1079     \@xa#1\@xa{\the\thmt@toks}%

```

```

1080 }{% go on collecting
1081   \@xa\thmt@toks\@xa{\the\thmt@toks\end{#3}}}%
1082   \thmt@collect@body{#1}%
1083 }%
1084 }

```

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

```

1085 \def\thmt@trivialref#1#2{%
1086   \ifcsname r@#1\endcsname
1087     \@xa\@xa\@xa\thmt@trivi@lr@f\csname r@#1\endcsname\relax\@nil
1088   \else #2\fi
1089 }
1090 \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.

```

1091 \def\thmt@innercounters{%
1092   equation}
1093 \def\thmt@counterformatters{%
1094   @alph,@Alph,@arabic,@roman,@Roman,@fnsymbol}
1095
1096 \@for\thmt@displ:=\thmt@counterformatters\do{%
1097   \@xa\let\csname thmt@\thmt@displ\@xa\endcsname\csname \thmt@displ\endcsname
1098 }%
1099 \def\thmt@sanitizethe#1{%
1100   \@for\thmt@displ:=\thmt@counterformatters\do{%
1101     \@xa\protected@edef\csname\thmt@displ\endcsname##1{%
1102       \@nx\ifx\@xa\@nx\csname c@#1\endcsname ##1%
1103       \@xa\protect\csname \thmt@displ\endcsname{##1}%
1104       \@nx\else
1105         \@nx\csname thmt@\thmt@displ\endcsname{##1}%
1106       \@nx\fi
1107     }%
1108   }%
1109   \expandafter\protected@edef\csname the#1\endcsname{\csname the#1\endcsname}%
1110   \ifcsname theH#1\endcsname
1111     \expandafter\protected@edef\csname theH#1\endcsname{\csname theH#1\endcsname}%
1112   \fi
1113 }
1114
1115 \def\thmt@rst@storecounters#1{%
1116   \bgroup
1117     % ugly hack: save chapter,...subsection numbers
1118     % for equation numbers.
1119   %\refstepcounter{thmt@dummyctr}% why is this here?
1120   %% temporarily disabled, broke autorefname.
1121   \def\@currentlabel{}%
1122   \@for\thmt@ctr:=\thmt@innercounters\do{%
1123     \thmt@sanitizethe{\thmt@ctr}%
1124     \protected@edef\@currentlabel{%
1125       \@currentlabel
1126       \protect\def\@xa\protect\csname the\thmt@ctr\endcsname{%
1127         \csname the\thmt@ctr\endcsname}%

```

```

1128 \ifcsname theH\thmt@ctr\endcsname
1129 \protect\def\@xa\protect\csname theH\thmt@ctr\endcsname{%
1130 (restate \protect\theHthmt@dummyctr)\csname theH\thmt@ctr\endcsname}%
1131 \fi
1132 \protect\setcounter{\thmt@ctr}{\number\csname c@\thmt@ctr\endcsname}%
1133 }%
1134 }%
1135 \label{thmt@@#1@data}%
1136 \egroup
1137 }%

```

Now, the main business.

```

1138 \newif\ifthmt@thisistheone
1139 \newenvironment{thmt@restatable}[3][]{%
1140 \thmt@toks{}}% will hold body
1141 %
1142 \stepcounter{thmt@dummyctr}% used for data storage label.
1143 %
1144 \long\def\thmrst@store##1{%
1145 \@xa\gdef\csname #3\endcsname{%
1146 \@ifstar{%
1147 \thmt@thisistheonefalse\csname thmt@stored@#3\endcsname
1148 }{%
1149 \thmt@thisistheonetrue\csname thmt@stored@#3\endcsname
1150 }%
1151 }%
1152 \@xa\long\@xa\gdef\csname thmt@stored@#3\@xa\endcsname\@xa{%
1153 \begingroup
1154 \ifthmt@thisistheone
1155 % these are the valid numbers, store them for the other
1156 % occasions.
1157 \thmt@rst@storecounters{#3}%
1158 \else
1159 % this one should use other numbers...
1160 % first, fake the theorem number.
1161 \@xa\protected@edef\csname the#2\endcsname{%
1162 \thmt@trivialref{thmt@@#3}{??}}%
1163 % if the number wasn't there, have a "re-run to get labels right"
1164 % warning.
1165 \ifcsname r@thmt@@#3\endcsname\else
1166 \G@refundefinedtrue
1167 \fi
1168 % prevent stepcountering the theorem number,
1169 % but still, have some number for hyperref, just in case.
1170 \@xa\let\csname c@#2\endcsname=c@thmt@dummyctr
1171 \@xa\let\csname theH#2\endcsname=\theHthmt@dummyctr
1172 % disable labeling.
1173 \let\label=\thmt@gobble@label
1174 \let\ltx@label=\@gobble% amsmath needs this
1175 % We shall need to restore the counters at the end
1176 % of the environment, so we get
1177 % (4.2) [(3.1 from restate)] (4.3)
1178 \def\thmt@restorecounters{%
1179 \@for\thmt@ctr:=\thmt@innercounters\do{%
1180 \protected@edef\thmt@restorecounters{%
1181 \thmt@restorecounters
1182 \protect\setcounter{\thmt@ctr}{\arabic{\thmt@ctr}}}%
1183 }%
1184 }%
1185 % pull the new semi-static definition of \theequation et al.
1186 % from the aux file.

```

```

1187 \thmt@trivialref{thmt@@#3@data}{}%
1188 \fi
1189 % call the proper begin-env code, possibly with optional argument
1190 % (omit if stored via key-val)
1191 \ifthmt@restatethis
1192 \thmt@restatethisfalse
1193 \else
1194 \csname #2\@xa\endcsname\ifx\@nx#1\@nx\else[{\#1}]\fi
1195 \fi
1196 \ifthmt@thisistheone
1197 % store a label so we can pick up the number later.
1198 \label{thmt@@#3}%
1199 \fi
1200 % this will be the collected body.
1201 ##1%
1202 \csname end#2\endcsname
1203 % if we faked the counter values, restore originals now.
1204 \ifthmt@thisistheone\else\thmt@restorecounters\fi
1205 \endgroup
1206 }% thmt@stored@#3
1207 % in either case, now call the just-created macro,
1208 \csname #3\@xa\endcsname\ifthmt@thisistheone\else*\fi
1209 % and artificially close the current environment.
1210 \@xa\end\@xa{\@currenvir}
1211 }% thm@rst@store
1212 \thmt@collect@body\thm@rst@store
1213 }{%
1214 %% now empty, just used as a marker.
1215 }
1216
1217 \let\thmt@gobble@label\@gobble
1218 % cleveref extends syntax of \label to \label[...]{...}
1219 \AtBeginDocument{
1220 \ifpackageloaded{cleveref}{
1221 \renewcommand*\thmt@gobble@label[2][{}]{
1222 }{}
1223 }
1224
1225 \newenvironment{restatable}{}%
1226 \thmt@thisistheonetrue\thmt@restatable
1227 }{%
1228 \endthmt@restatable
1229 }
1230 \newenvironment{restatable*}{}%
1231 \thmt@thisistheonefalse\thmt@restatable
1232 }{%
1233 \endthmt@restatable
1234 }
1235
1236 %%% support for keyval-style: restate=foobar
1237 \protected@edef\thmt@thmuse@families{%
1238 \thmt@thmuse@families%
1239 ,restate phase 1%
1240 ,restate phase 2%
1241 }
1242 \newcommand\thmt@splitrestateargs[1][{}]{%
1243 \g@addto@macro\thmt@storedoptargs{,\#1}%
1244 \def\tmp@a##1\@{\def\thmt@storename{##1}}%
1245 \tmp@a
1246 }
1247

```

```

1248 \newif\ifthmt@restatethis
1249 \define@key{restate phase 1}{restate}{%
1250   \thmt@thmuse@iskvtrue
1251   \def\thmt@storedoptargs{}% discard the first time around
1252   \thmt@splitrestateargs #1\@
1253   \def\thmt@storedoptargs{}% discard the first time around
1254   %\def\thmt@storename{#1}%
1255   \thmt@debug{we will restate as '\thmt@storename' with more args
1256   '\thmt@storedoptargs'}%
1257   \@namedef{thmt@unusedkey@restate}{}%
1258   % spurious "unused key" fixes itself once we are after tracknames...
1259   \thmt@restatethistrue
1260   \protected@edef\tmp@a{%
1261     \nx\thmt@thisistheonettrue
1262     \nx\def\nx\@currentenv{\thmt@envname}%
1263     \nx\@xa\nx\thmt@restatable\nx\@xa[\nx\thmt@storedoptargs]%
1264     {\thmt@envname}{\thmt@storename}%
1265   }%
1266   \@xa\g@addto@macro\@xa\thmt@local@postheadhook\@xa{%
1267     \tmp@a
1268   }%
1269 }
1270 \thmt@mkignoringkeyhandler{restate phase 1}
1271
1272 \define@key{restate phase 2}{restate}{%
1273   % do not store restate as a key for repetition:
1274   % infinite loop.
1275   % instead, retain the added keyvals
1276   % overwriting thmt@storename should be safe here, it's been
1277   % xdefd into the postheadhook
1278   \thmt@splitrestateargs #1\@
1279 }
1280 \kv@set@family@handler{restate phase 2}{%
1281   \ifthmt@restatethis
1282     \@xa\@xa\@xa\g@addto@macro\@xa\@xa\@xa\thmt@storedoptargs\@xa\@xa\@xa{\@xa\@xa\@xa,%
1283       \@xa\kv@key\@xa=\kv@value}%
1284   \fi
1285 }
1286

```

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.

```

1287
1288 \RequirePackage{thm-patch, aliasctr, parseargs, keyval}
1289
1290 \let\@xa=\expandafter
1291 \let\@nx=\noexpand
1292
1293 \newcommand\thmt@autorefsetup{%
1294   \@xa\def\csname\thmt@envname autorefname\@xa\endcsname\@xa{\thmt@thmname}%
1295   \ifthmt@hassibling
1296     \@counteralias{\thmt@envname}{\thmt@sibling}%
1297     \@xa\def\@xa\thmt@autoreffix\@xa{%
1298       \@xa\global\@xa\let\csname the\thmt@envname\@xa\endcsname
1299       \csname the\thmt@sibling\endcsname
1300       \def\thmt@autoreffix{}%
1301     }%

```

```

1302 \protected@edef\thmt@sibling{\thmt@envname}%
1303 \fi
1304 }
1305 \g@addto@macro\thmt@newtheorem@predefinition{\thmt@autorefsetup}%
1306 \g@addto@macro\thmt@newtheorem@postdefinition{\csname thmt@autoreffix\endcsname}%
1307
1308 \def\thmt@refnamewithcomma #1#2#3,#4,#5\@nil{%
1309 \@xa\def\csname\thmt@envname #1utorefname\endcsname{#3}%
1310 \ifcsname #2refname\endcsname
1311 \csname #2refname\@xa\endcsname\@xa{\thmt@envname}{#3}{#4}%
1312 \fi
1313 }
1314 \define@key{thmdef}{refname}{\thmt@trytwice}{%
1315 \thmt@refnamewithcomma{a}{c}#1,\textbf{?? (pl. #1)},\@nil
1316 }}
1317 \define@key{thmdef}{Refname}{\thmt@trytwice}{%
1318 \thmt@refnamewithcomma{A}{C}#1,\textbf{?? (pl. #1)},\@nil
1319 }}
1320
1321
1322 \ifcsname Autoref\endcsname\else
1323 \let\thmt@HyRef@testreftype\HyRef@testreftype
1324 \def\HyRef@Testreftype#1.#2\{%
1325 \ltx@ifundefined{#1Autorefname}{%
1326 \thmt@HyRef@testreftype#1.#2\%
1327 }{%
1328 \edef\HyRef@currentHtag{%
1329 \expandafter\noexpand\csname#1Autorefname\endcsname
1330 \noexpand~%
1331 }%
1332 }%
1333 }
1334
1335
1336 \let\thmt@HyPsd@@autorefname\HyPsd@@autorefname
1337 \def\HyPsd@@Autorefname#1.#2\@nil{%
1338 \tracingall
1339 \ltx@ifundefined{#1Autorefname}{%
1340 \thmt@HyPsd@@autorefname#1.#2\@nil
1341 }{%
1342 \csname#1Autorefname\endcsname\space
1343 }%
1344 }%
1345 \def\Autoref{%
1346 \parse{%
1347 {\parseFlag*\def\thmt@autorefstar{*}}{\let\thmt@autorefstar\@empty}}%
1348 {\parseMand{%
1349 \bgroup
1350 \let\HyRef@testreftype\HyRef@Testreftype
1351 \let\HyPsd@@autorefname\HyPsd@@Autorefname
1352 \@xa\autoref\thmt@autorefstar{##1}%
1353 \egroup
1354 \let\@parsecmd\@empty
1355 }}%
1356 }%
1357 }
1358 \fi % ifcsname Autoref
1359
1360 % not entirely appropriate here, but close enough:
1361 \AtBeginDocument{%
1362 \@ifpackageloaded{nameref}{%

```



```

1363 \addtotheorempostheadhook{%
1364 \expandafter\NR@getttitle\expandafter{\thmt@shortoptarg}%
1365 }{}
1366 }
1367
1368 \AtBeginDocument{%
1369 \ifpackageloaded{cleveref}{%
1370 \ifpackagelater{cleveref}{2010/04/30}{%
1371 % OK, new enough
1372 }{%
1373 \PackageWarningNoLine{thmtools}{%
1374 Your version of cleveref is too old!\MessageBreak
1375 Update to version 0.16.1 or later%
1376 }
1377 }
1378 }{}
1379 }

```

A.2 Glue code for different backends

A.2.1 amsthm

```

1380 \providecommand\thmt@space{ }
1381
1382 \define@key{thmstyle}{spaceabove}{%
1383 \def\thmt@style@spaceabove{#1}%
1384 }
1385 \define@key{thmstyle}{spacebelow}{%
1386 \def\thmt@style@spacebelow{#1}%
1387 }
1388 \define@key{thmstyle}{headfont}{%
1389 \def\thmt@style@headfont{#1}%
1390 }
1391 \define@key{thmstyle}{bodyfont}{%
1392 \def\thmt@style@bodyfont{#1}%
1393 }
1394 \define@key{thmstyle}{notefont}{%
1395 \def\thmt@style@notefont{#1}%
1396 }
1397 \define@key{thmstyle}{headpunct}{%
1398 \def\thmt@style@headpunct{#1}%
1399 }
1400 \define@key{thmstyle}{notebraces}{%
1401 \def\thmt@style@notebraces{\thmt@embrace#1}%
1402 }
1403 \define@key{thmstyle}{break}[]{}
1404 \def\thmt@style@postheadspace{\newline}%
1405 }
1406 \define@key{thmstyle}{postheadspace}{%
1407 \def\thmt@style@postheadspace{#1}%
1408 }
1409 \define@key{thmstyle}{headindent}{%
1410 \def\thmt@style@headindent{#1}%
1411 }
1412
1413 \newtoks\thmt@style@headstyle
1414 \define@key{thmstyle}{headformat}[]{}
1415 \thmt@setheadstyle{#1}%
1416 }
1417 \define@key{thmstyle}{headstyle}[]{}

```

```

1418 \thmt@setheadstyle{#1}%
1419 }
1420 \def\thmt@setheadstyle#1{%
1421 \thmt@style@headstyle{%
1422 \def\NAME{\the\thm@headfont ##1}%
1423 \def\NUMBER{\bgroup\@upn{##2}\egroup}%
1424 \def\NOTE{\if=##3=\else\bgroup\thmt@space\the\thm@notefont(##3)\egroup\fi}%
1425 }%
1426 \def\thmt@tmp{#1}%
1427 \@onelevel@sanitize\thmt@tmp
1428 %\tracingall
1429 \ifcsname thmt@headstyle@\thmt@tmp\endcsname
1430 \thmt@style@headstyle\@xa{%
1431 \the\thmt@style@headstyle
1432 \csname thmt@headstyle@#1\endcsname
1433 }%
1434 \else
1435 \thmt@style@headstyle\@xa{%
1436 \the\thmt@style@headstyle
1437 #1%
1438 }%
1439 \fi
1440 %\showthe\thmt@style@headstyle
1441 }
1442 % examples:
1443 \def\thmt@headstyle@margin{%
1444 \makebox[Opt][r]{\NUMBER\ }\NAME\NOTE
1445 }
1446 \def\thmt@headstyle@swapnumber{%
1447 \NUMBER\ \NAME\NOTE
1448 }
1449
1450
1451
1452 \def\thmt@embrace#1#2(#3){#1#3#2}
1453
1454 \def\thmt@declaretheoremstyle@setup{%
1455 \let\thmt@style@notebraces\@empty%
1456 \thmt@style@headstyle{}}%
1457 \kvsetkeys{thmstyle}{%
1458 spaceabove=3pt,
1459 spacebelow=3pt,
1460 headfont=\bfseries,
1461 bodyfont=\normalfont,
1462 headpunct={.},
1463 postheadspace={ },
1464 headindent={},
1465 notefont={\fontseries\mddefault\upshape}
1466 }%
1467 }
1468 \def\thmt@declaretheoremstyle#1{%
1469 %\show\thmt@style@spaceabove
1470 \thmt@toks{\newtheoremstyle{#1}}%
1471 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@spaceabove}}%
1472 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@spacebelow}}%
1473 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@bodyfont}}%
1474 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@headindent}}% indent1 FIXM
1475 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@headfont}}%
1476 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@headpunct}}%
1477 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\thmt@style@postheadspace}}%
1478 \thmt@toks\@xa\@xa\@xa{\@xa\the\@xa\thmt@toks\@xa{\the\thmt@style@headstyle}}% headspec

```



```

1537 }%
1538 {%
1539 \def\thmt@local@preheadhook{%
1540 \def\thmt@local@postheadhook{%
1541 \def\thmt@local@prefoothook{%
1542 \def\thmt@local@postfoothook{%
1543 \thmt@local@preheadhook
1544 \csname thmt@#1@preheadhook\endcsname
1545 \thmt@generic@preheadhook
1546 \protected@edef\tmp@args{%
1547 \ifthmt@hasoverlay <\thmt@overlay>\fi
1548 \ifx\@empty\thmt@optarg\else [{\thmt@optarg}]\fi
1549 }%
1550 \csname thmt@original@#1\@xa\endcsname\tmp@args
1551 \thmt@local@postheadhook
1552 \csname thmt@#1@postheadhook\endcsname
1553 \thmt@generic@postheadhook
1554 \let\@parsecmd\@empty
1555 }%
1556 }
1557 }%

```

A.2.3 ntheorem

```

1558
1559 \providecommand\thmt@space{ }
1560
1561 % actually, ntheorem's so-called style is nothing like a style at all...
1562 \def\thmt@declaretheoremstyle@setup{
1563 \def\thmt@declaretheoremstyle#1{%
1564 \ifcsname th@#1\endcsname\else
1565 \@xa\let\csname th@#1\endcsname\th@plain
1566 \fi
1567 }
1568
1569 \def\thmt@notsupported#1#2{%
1570 \PackageWarning{thmttools}{Key '#2' not supported by #1}{}%
1571 }
1572
1573 \define@key{thmstyle}{spaceabove}{%
1574 \setlength\theorempreskipamount{#1}%
1575 }
1576 \define@key{thmstyle}{spacebelow}{%
1577 \setlength\theorempostskipamount{#1}%
1578 }
1579 \define@key{thmstyle}{headfont}{%
1580 \theoremheaderfont{#1}%
1581 }
1582 \define@key{thmstyle}{bodyfont}{%
1583 \theorembodyfont{#1}%
1584 }
1585 % not supported in ntheorem.
1586 \define@key{thmstyle}{notefont}{%
1587 \thmt@notsupported{ntheorem}{notefont}%
1588 }
1589 \define@key{thmstyle}{headpunct}{%
1590 \theoremseparator{#1}%
1591 }
1592 % not supported in ntheorem.
1593 \define@key{thmstyle}{notebraces}{%
1594 \thmt@notsupported{ntheorem}{notebraces}%

```

```

1595 }
1596 \define@key{thmstyle}{break}{%
1597   \theoremstyle{break}%
1598 }
1599 % not supported in ntheorem...
1600 \define@key{thmstyle}{postheadspace}{%
1601   %\def\thmt@style@postheadspace{#1}%
1602   \@xa\g@addto@macro\csname thmt@style \thmt@style @defaultkeys\endcsname{%
1603     postheadhook={\hspace{-\labelsep}\hspace*{#1}},%
1604   }%
1605 }
1606
1607 % not supported in ntheorem
1608 \define@key{thmstyle}{headindent}{%
1609   \thmt@notsupported{ntheorem}{headindent}%
1610 }
1611 % sorry, only style, not def with ntheorem.
1612 \define@key{thmstyle}{qed}[\qedsymbol]{%
1613   \@ifpackagewith{ntheorem}{thmmarks}{%
1614     \theoremsymbol{#1}%
1615   }{%
1616     \thmt@notsupported
1617       {ntheorem without thmmarks option}%
1618       {headindent}%
1619   }%
1620 }
1621
1622 \let\@upn=\textup
1623 \define@key{thmstyle}{headformat}[]{%
1624   \def\thmt@tmp{#1}%
1625   \@onelevel@sanitize\thmt@tmp
1626   %\tracingall
1627   \ifcsname thmt@headstyle@\thmt@tmp\endcsname
1628     \newtheoremstyle{\thmt@style}{%
1629       \item[\hskip\labelsep\theorem@headerfont%
1630         \def\NAME{\theorem@headerfont #####1}%
1631         \def\NUMBER{\bgroup\@upn{#####2}\egroup}%
1632         \def\NOTE{}}%
1633       \csname thmt@headstyle@#1\endcsname
1634       \theorem@separator
1635     ]
1636   }{%
1637     \item[\hskip\labelsep\theorem@headerfont%
1638       \def\NAME{\theorem@headerfont #####1}%
1639       \def\NUMBER{\bgroup\@upn{#####2}\egroup}%
1640       \def\NOTE{\if=#####3=\else\bgroup\thmt@space(#####3)\egroup\fi}%
1641       \csname thmt@headstyle@#1\endcsname
1642       \theorem@separator
1643     ]
1644   }
1645 \else
1646   \newtheoremstyle{\thmt@style}{%
1647     \item[\hskip\labelsep\theorem@headerfont%
1648       \def\NAME{\the\thm@headfont #####1}%
1649       \def\NUMBER{\bgroup\@upn{#####2}\egroup}%
1650       \def\NOTE{}}%
1651     #1%
1652     \theorem@separator
1653   ]
1654 }{%
1655   \item[\hskip\labelsep\theorem@headerfont%

```

```

1656 \def\NAME{\the\thm@headfont ####1}%
1657 \def\NUMBER{\bgroup\@upn{####2}\egroup}%
1658 \def\NOTE{\if=####3=\else\bgroup\thmt@space(####3)\egroup\fi}%
1659 #1%
1660 \theorem@separator
1661 ]
1662 }
1663 \fi
1664 }
1665
1666 \def\thmt@headstyle@margin{%
1667 \makebox[0pt][r]{\NUMBER\ }\NAME\NOTE
1668 }
1669 \def\thmt@headstyle@swapnumber{%
1670 \NUMBER\ \NAME\NOTE
1671 }
1672
1673
1674

```

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.

```

1675
1676 \newtoks\@parsespec
1677 \def\parse@endquark{\parse@endquark}
1678 \newcommand\parse[1]{%
1679 \@parsespec{#1\parse@endquark}\@parse}
1680
1681 \newcommand\@parse{%
1682 \edef\p@tmp{\the\@parsespec}%
1683 \ifx\p@tmp\parse@endquark
1684 \expandafter\@gobble
1685 \else
1686 % \typeout{parsespec remaining: \the\@parsespec}%
1687 \expandafter\@firstofone
1688 \fi{%
1689 \@parsepop
1690 }%
1691 }
1692 \def\@parsepop{%
1693 \expandafter\p@rsepop\the\@parsespec\@nil
1694 \@parsecmd
1695 }
1696 \def\p@rsepop#1#2\@nil{%
1697 #1%
1698 \@parsespec{#2}%
1699 }
1700
1701 \newcommand\parseOpt[4]{%
1702 %\parseOpt{openchar}{closechar}{yes}{no}
1703 % \typeout{attempting #1#2...}%
1704 \def\@parsecmd{%

```

```

1705 \@@ifnextchar#1{\@@reallyparse}{#4\@parse}%
1706 }%
1707 \def\@@reallyparse#1##1#2{%
1708   #3\@parse
1709 }%
1710 }
1711
1712 \newcommand\parseMand[1]{%
1713   %\parseMand{code}
1714   \def\@parsecmd##1{#1\@parse}%
1715 }
1716
1717 \newcommand\parseFlag[3]{%
1718   %\parseFlag{flagchar}{yes}{no}
1719   \def\@parsecmd{%
1720     \@@ifnextchar#1{#2\expandafter\@parse\@gobble}{#3\@parse}%
1721   }%
1722 }

```

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`.

```

1723 \def\aliasctr@follow#1#2\@nil#3{%
1724   \ifx#1\@elt
1725     \noexpand #3%
1726   \else
1727     \expandafter\aliasctr@follow#1\@elt\@nil{#1}%
1728   \fi
1729 }
1730 \newcommand\aliasctr@follow[1]{%
1731   \expandafter\aliasctr@follow

```

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

```

1732   \csname cl@#1\endcsname\@elt\@nil{\csname cl@#1\endcsname}%
1733 }
1734 \renewcommand*\@addtoreset[2]{\bgroup
1735   \edef\aliasctr@@truelist{\aliasctr@follow{#2}}%
1736   \let\@elt\relax
1737   \expandafter\@cons\aliasctr@@truelist{#{1}}%
1738 \egroup}

```

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

```

1739 % FMi 2019-07-31 \@removereset is in the kernel these days
1740 \@ifundefined{\removefromreset}{\RequirePackage{remreset}}{}
1741 \renewcommand*\@removefromreset[2]{\bgroup
1742   \edef\aliasctr@@truelist{\aliasctr@follow{#2}}%
1743   \expandafter\let\csname c@#1\endcsname\@removefromreset
1744   \def\@elt##1{%
1745     \expandafter\ifx\csname c@##1\endcsname\@removefromreset
1746     \else
1747       \noexpand\@elt{##1}%
1748     \fi}%

```

```

1749 \expandafter\xdef\aliasctr@@truelist{%
1750     \aliasctr@@truelist}
1751 \egroup}

make #1 a counter that uses counter #2's count register.
1752 \newcommand\@counteralias[2]{\%
1753     \def\@@gletover##1##2{%
1754         \expandafter\global
1755         \expandafter\let\csname ##1\expandafter\endcsname
1756         \csname ##2\endcsname
1757     }%
1758     \@ifundefined{c@#2}{\@nocounterr{#2}}{%
1759         \expandafter\ifdefinable\csname c@#1\endcsname{%

```

Four values make a counter foo:

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

hyperref adds \theHfoo in particular.

```

1760     \@@gletover{c@#1}{c@#2}%
1761     \@@gletover{the#1}{the#2}%

```

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

```

1762     \@@gletover{theH#1}{theH#2}%

```

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

```

1763     \@ifpackageloaded{cleveref}{%
1764         \edef\aliasctr@temp{%
1765             \noexpand\cref@resetby{#2}{\noexpand\cref@result}}%
1766         \aliasctr@temp
1767         \ifx\cref@result\relax\else%
1768             \cref@addtoreset{#1}{\cref@result}%
1769         \fi
1770     }{}%
1771     \@@gletover{p@#1}{p@#2}%
1772     \expandafter\global
1773     \expandafter\def\csname cl@#1\expandafter\endcsname
1774     \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.

```

1775     %\@addtoreset{#1}{@ckpt}%
1776     }%
1777     }%
1778 }%

```

A.3.3 Tracking occurrences: none, one or many

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

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

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

```

1779
1780 \DeclareOption{uniq}{%
1781   \newwrite\uniq@channel
1782   \InputIfFileExists{\jobname.unq}{\relax}{\relax}%
1783   \immediate\openout\uniq@channel=\jobname.unq
1784   \AtEndDocument{%
1785     \immediate\closeout\uniq@channel%
1786   }
1787 }
1788 \DeclareOption{aux}{%
1789   \let\uniq@channel\@auxout
1790 }
1791

```

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

```

1792 \newcommand\setuniqmark[1]{%
1793   \expandafter\ifx\csname uniq@now@#1\endcsname\relax
1794     \global\@namedef{uniq@now@#1}{\uniq@ONE}%
1795   \else
1796     \expandafter\ifx\csname uniq@now@#1\endcsname\uniq@MANY
1797     \else
1798       \immediate\write\uniq@channel{%
1799         \string\uniq@setmany{#1}%
1800       }%
1801       \ifuniq{#1}{%
1802         \uniq@warnnotunique{#1}%
1803       }{}%
1804     \fi
1805     \global\@namedef{uniq@now@#1}{\uniq@MANY}%
1806   \fi
1807 }

```

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

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

```

1808 \newcommand\ifuniq[1]{%
1809   \expandafter\ifx\csname uniq@last@#1\endcsname\uniq@MANY
1810     \expandafter\@secondoftwo
1811   \else
1812     \expandafter\@firstoftwo
1813   \fi
1814 }

```

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

```

1815 \def\uniq@ONE{\uniq@ONE}
1816 \def\uniq@MANY{\uniq@MANY}

```

Flag: suggest a rerun?

```

1817 \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.

```

1818 \def\uniq@setmany#1{%
1819   \global\@namedef{uniq@last@#1}{\uniq@MANY}%
1820   \AtEndDocument{%
1821     \uniq@warnifunique{#1}%
1822   }%
1823 }

```

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.

```

1824 \def\uniq@warnifunique#1{%
1825   \expandafter\ifx\csname uniq@now@#1\endcsname\uniq@MANY\else
1826     \PackageWarningNoLine{uniq}{%
1827       ‘#1’ is unique now.\MessageBreak
1828       Rerun LaTeX to pick up the change%
1829     }%
1830     \@uniq@reruntrue
1831   \fi
1832 }

```

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.

```

1833 \def\uniq@warnnotunique#1{%
1834   \PackageWarningNoLine{uniq}{%
1835     ‘#1’ is not unique anymore.\MessageBreak
1836     Rerun LaTeX to pick up the change%
1837   }%
1838   \@uniq@reruntrue
1839 }

```

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.

```

1840 \def\uniq@maybesuggestrerun{%
1841   \if@uniq@rerun
1842     \PackageWarningNoLine{uniq}{%
1843       Uniquenesses have changed. \MessageBreak
1844       Rerun LaTeX to pick up the change%
1845     }%
1846   \fi
1847 }

```

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

```

1848 \AtEndDocument{%
1849   \immediate\write\@auxout{\string\uniq@maybesuggestrerun}%
1850 }
1851 \ExecuteOptions{aux}
1852 \ProcessOptions\relax

```