

repththeorem*

Jesse Straat

2025-09-03

Abstract

When writing a large manuscript, it is sometimes beneficial to repeat a theorem (or lemma or ...) at an earlier or later point for didactical purposes. However, `thmtools`'s built-in `restateable` only allows replicating theorems *after* they have been stated, and only in the same document. `repththeorem` solves the issue by making use of the `.aux` file, and also introduces its own file extension, `.thm`, to replicate theorems in other files.

Contents

1 Repeating theorems	1
2 Replicating theorems between files	3
2.1 Replicating theorems to subfiles	3
3 Source code	3

1 Repeating theorems

Let's say we define a theorem as follows:

```
\begin{theorem}[Yoneda Lemma]
  For  $(F\colon \mathcal{C}^{\mathrm{op}}\to \mathbf{Set})$  a functor,
   $([\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](YA, F) \cong F(A))\%$ 
  for all objects  $A$  in  $\mathcal{C}$ .
\end{theorem}
```

Its output is of course

Theorem 1 (Yoneda Lemma). *For $F\colon \mathcal{C}^{\mathrm{op}} \rightarrow \mathbf{Set}$ a functor, $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](YA, F) \cong F(A)$ for all objects A in \mathcal{C} .*

Now let's say we want to replicate the theorem within the same document. `makethm` (*env.*) That is what the new environment `makethm` is used for.

```
\begin{makethm}{theorem}{thm:Yoneda}[Yoneda Lemma]
  For  $(F\colon \mathcal{C}^{\mathrm{op}}\to \mathbf{Set})$  a functor,
```

*Version v1.4.1, last revised 2025-09-03.

```

\([\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](YA, F) \cong F(A)\)
for all objects  $A$  in  $\mathcal{C}$ .
\end{makethm}

```

Its output is the same (in fact, we’ve secretly used `makethm` in the previous example), but the important difference is that we have saved the theorem for later use.

The `makethm` environment takes two mandatory arguments and one optional one. The first mandatory argument is the type of theorem environment as defined in `amsthm`, like `theorem`, `lemma`, `definition`, etc. The second is the theorem’s label. The label is mandatory since, to replicate the theorem, we need to have a “name” attached to it. `makethm` automatically attaches a `\label`, as well, so `\ref{thm:Yoneda}` becomes `1`. The optional argument is passed right to the optional argument of the theorem environment, giving the theorem a name.

Now let’s say we want to replicate the theorem later or earlier in the text. This may be done if, for example, the theorem is proven at a later point, or we want to “tease” the reader with a powerful theorem that will be proven later in the chapter. To do this, we use the `\repthm` command: `\repthm{thm:Yoneda}`. This outputs the theorem again.

Theorem 1 (Yoneda Lemma). *For $F: \mathcal{C}^{\mathrm{op}} \rightarrow \mathbf{Set}$ a functor, $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](YA, F) \cong F(A)$ for all objects A in \mathcal{C} .*

The label of this theorem is a `\ref`, and automatically links to the original theorem statement.

If the original theorem statement exists in a different file, or has not been created yet, we can add a placeholder alt text to the `\repthm` as an optional argument, which only displays if the theorem is undefined. For example, `\repthm{thm:foo}[bar]` returns

Theorem ??. *bar*

If we do the same without providing an alt text, we get

Theorem ??.

together with a warning: “Package `repthm`: Theorem `thm:foo` not defined; rebuild your project. If the issue persists, create the theorem using `\begin{makethm}` or consider adding alt text to `\repthm` using the optional parameter.”

Since we’re using the `.aux` file, it is possible to replicate a theorem before it is stated. For example,

```

\repthm{thm:later}
\begin{makethm}{theorem}{thm:later}
  Alligator!
\end{makethm}

```

returns

Theorem 2. *Alligator!*

Theorem 2. *Alligator!*

Note that it is necessary to run a `.tex` file twice to replicate theorems ahead of time, similarly to how one has to run a file twice to make sure the references are correct.

`\repthm*` It is also possible to use a starred version, `\repthm*`. It then automatically adds a star to the end of the theorem environment. For example, `theorem` becomes `theorem*`.

2 Replicating theorems between files

Let's say we have the following files for our project:

```
foo.tex
bar.tex
```

Let's say that we have defined a theorem `thm:baz` in `bar.tex`, and we want to replicate it in `foo.tex`. To achieve this, we first use the `\theoremfile` command in the preamble of `bar.tex`. This compiles all theorems defined in `bar.tex` and outputs them into a file `bar.thm`. To then import these into `foo.tex`, we use `\loadtheorems` `\loadtheorems{bar.thm}` in the preamble, which loads all theorems saved in `bar.thm`. One can then use `\repthm` as usual.

Since the `.aux` file is loaded at `\begin{document}`, putting `\loadtheorems` in the preamble of a file will guarantee that the loaded theorem file will be overwritten by the theorems in the `.aux` file, i.e., theorems defined in the same document. In our example, if we also defined a `thm:baz` in `foo.tex`, loading `bar.thm` into `foo.tex` will not overwrite the local `thm:baz`.

2.1 Replicating theorems to subfiles

Replicating theorems to different files is particularly useful when working in big documents with multiple subfiles. For example, let's say we have the files

```
main.tex
foo.tex
bar.tex
```

Here, `main.tex` is generated by including `foo.tex` and `bar.tex` as chapters, creating a single large document. It is now possible to replicate theorems within the subfiles by running `\theoremfile` in `main.tex`, and then using `\loadtheorems{main.thm}` in `foo.tex` and `bar.tex`. This will allow us to use all theorems in the final `main.tex` in each of the subfiles.

3 Source code

```
1 (*package)
2 \ProvidesPackage{repththeorem}[2025-09-03 v1.4.1 Repththeorem package]
\theoremfile Using \theoremfile will output all saved theorems into an output file. By default,
if your LATEX file is foo.tex, the output file is foo.thm.
3 \def\repththeorem@theoremfile{\relax}
```

```

4 \NewDocumentCommand{\theoremfile}{ 0{\jobname.thm} }{
5 % 0: the path of the file to which we should save theorems
6 %
7 \def\repththeorem@theoremfile{#1}
8 \newwrite\thmlist
9 \immediate\openout\thmlist=#1
10 }

```

`\loadtheorems` If you have exported saved theorems to a file, you can load them into another file using the macro `\loadtheorems`.

```

11 \NewDocumentCommand{\loadtheorems}{ m }{
12 \IfFileExists{#1}{
13 \makeatletter
14 \input{#1}
15 \makeatother
16 }{
17 \PackageWarning{repththeorem}{%
18 File #1 not found. I will not import any theorems.%
19 }
20 }
21 }

```

The `\makeatletter` is included here to assure that any macros that are expanded into macros that contain an `@` are interpreted correctly.

`repthm@firstoffive` This returns the first of five arguments. It is used to get the theorem number of a label.

```

22 \NewExpandableDocumentCommand{\repthm@firstoffive}{ m m m m m }{%
23 #1%
24 }

```

`repthm@ifrefexists` This command checks whether a label was defined in the auxiliary file. Using just `\ifcsname r@foo\endcsname` is not sufficient, since `\ref{foo}` defines `\r@foo` using a placeholder variable if the reference does not exist.

```

25 \NewExpandableDocumentCommand{\repthm@ifrefexists}{ m +m +m }{%
26 % m: the label
27 % m: code to run if true
28 % m: code to run if false
29 \ifcsname r@#1\endcsname
30 % reference exists but might be dummy
31 \expandafter\ifx\csname r@#1\endcsname\relax
32 % reference is a dummy
33 #3%
34 \else
35 % reference exists and is not a dummy
36 #2%
37 \fi
38 \else
39 % reference doesn't exist
40 #3%
41 \fi
42 }

```

`makethm (env.)` On to defining the actual theorems to be saved.

```

43 \NewDocumentEnvironment{makethm}{ m m o +b }
44 % m: the type of theorem environment
45 % m: the name of the theorem
46 % o: optional parameter for environment
47 % b: the content of the theorem
48 %
49 {%
50 \IfValueTF{#3}{% Check if theorem has optional arguments
51 \begin{#1}[#3]\label{#2}
52 }{
53 \begin{#1}\label{#2}
54 }
55 % \begin{theorem}
56 #4
57 \providecommand{\label}[1]{
58 \expandafter\gdef\csname thmtype@#2\endcsname{#1}%
59 \expandafter\long\expandafter\gdef\csname thm@#2\endcsname{#4}%
60 \IfValueT{#3}{% Only save theorem name if it exists
61 \expandafter\gdef\csname thmdesc@#2\endcsname{#3}%
62 }
63 % Saving parameters to aux file
64 \expandafter\long\expandafter\gdef\csname thmoutput@#2\endcsname{%
65 \string\expandafter\string\gdef\noexpand%
66 \csname thmtype@#2\string\endcsname{#1}%
67 ^^J%
68 \string\expandafter\string\long\string\expandafter%
69 \string\gdef\noexpand\csname thm@#2\string\endcsname{#4}%
70 \IfValueT{#3}{%
71 ^^J%
72 \string\expandafter\string\gdef\noexpand%
73 \csname thmdesc@#2\string\endcsname{#3}%
74 }%
75 ^^J%
76 \string\expandafter\string\gdef\noexpand%
77 \csname thmlabel@#2\string\endcsname{%
78 \repthm@ifrefexists{#2}{%
79 \expandafter\repthm@firstoffive\expanded{\csname r@#2\endcsname}%
80 }{}}%
81 }%
82 }
83 \write@auxout{\csname thmoutput@#2\endcsname}
84 \if\repthm@theorem@theoremfile\relax
85 % No file has been set
86 \else
87 % We have a theorem file
88 % Saving parameters to theorem file
89 \write@thmlist{\csname thmoutput@#2\endcsname}
90 \fi
91 \end{#1}
92 }{}

```

`\repthm` To repeat a theorem, use the `\repthm` command.

If the theorem type shares its counter with another theorem type, e.g., `lemma` having the same counter as `theorem`, make sure you have `thmtools` imported. Its

\@counteralias macro is essential for the counters to work.

```
93 \newcounter{old@counter}
94 \NewDocumentCommand{\repthm}{ s m +o }{
95 % s: optional star to add to theorem environment
96 % m: the name of the theorem
97 % o: alt text
98 \begingroup
99 % Check if thmtype is given
100 \ifcsname thmtype@#2\endcsname%
101 \expandafter\let\expandafter\@@thmtype\csname thmtype@#2\endcsname%
102 \else%
103 \def\@@thmtype{theorem}%
104 \PackageWarning{repthm}{%
105 Theorem '#2' has unknown theorem type. Assuming it is of
106 type 'theorem'.%
107 }
108 \fi%
109 \edef\@@thmcounter{\@@thmtype}
110 \IfBooleanT{#1}{\edef\@@thmtype{\@@thmtype*}}
111 %
112 % Save theorem counter so we don't increase it
113 \ifcsname c@\@@thmcounter\endcsname
114 \else
115 \PackageWarning{repthm}{%
116 Counter '\@@thmcounter' not defined; if theorem
117 '\@@thmcounter' shares its counter with another
118 theorem, make sure thmtools is imported.%
119 }
120 \fi
121 \setcounter{old@counter}{\value{\@@thmcounter}}
122 \setcounter{\@@thmcounter}{-900}
123 %
124 % Set label number
125 \repthm@ifrefexists{#2}{%
126 % Reference exists: set number as reference
127 \expandafter\def\csname the\@@thmtype\endcsname{\ref{#2}}
128 }{%
129 % Force label number as saved
130 \ifcsname thmlabel@#2\endcsname
131 \expandafter\def\csname the\@@thmtype\endcsname{%
132 \csname thmlabel@#2\endcsname%
133 }
134 \else
135 % No label number saved: revert to ??
136 \expandafter\def\csname the\@@thmtype\endcsname{\ref{#2}}
137 \fi
138 }
139 %
140 \let\@@theoremnotdefined\relax
141 %
142 \ifcsname thm@#2\endcsname% Check if theorem is even defined
143 % Theorem is defined
144 \expandafter\let\expandafter\@@thm\csname thm@#2\endcsname
145 % Output theorem
```

```

146 \ifcstype thmdesc@#2\endcstype % Check if theorem has name
147 \begin{\@thmtype}[\cstype thmdesc@#2\endcstype]
148 \@@thm
149 \end{\@thmtype}
150 \else % No optionals
151 \begin{\@thmtype}
152 \@@thm
153 \end{\@thmtype}
154 \fi
155 \else
156 % Theorem undefined
157 \IfValueTF{#3}{
158 \begin{\@thmtype}
159 #3
160 \end{\@thmtype}
161 }{% No theorem or alt text provided: throw warning
162 \begin{\@thmtype}
163 \end{\@thmtype}
164 \PackageWarning{repthm}{%
165 Theorem '#2' not defined; rebuild your project.
166 If the issue persists, create the theorem using
167 \begin{makethm} or consider adding alt text to \repthm
168 using the optional parameter.%
169 }
170 }
171 \fi
172 \setcounter{\@thmcounter}{\value{oldcounter}}
173 % Reset theorem counter back to original
174 \endgroup
175 }

176 </package>

```

Change History

v1.0		\repthm: Fixed bug where
	General: First public release	theorems got a name even if
		undefined.
v1.1		5
	makethm: Now saves theorem	v1.3
	environment type, breaking	\repthm: Added hyperref named
	backwards compatibility.	destination compatibility by
		setting counter to very low
	\repthm: Now saves theorem	value.
	environment type, breaking	5
	backwards compatibility.	Changed thetheorem to cstype
		to fix compatibility with
v1.2		theorem types not called
	makethm: Environment end moved	“theorem”.
	to fix vertical spacing.	5
	Renamed theorem output	v1.4
	variable to be unique for each	\loadtheorems: Now makes @
	theorem.	catcode 11 to fix
	Theorem name is only saved if it	incompatibility.
	exists.	4
		makethm: Added theorem label to

aux file.	4	firstoffive command	4
<code>\repthm</code> : Added warnings for unknown counter and unknown theorem type.	5	<code>repthm@ifrefexists</code> : Added ifrefexists command	4
If reference doesn't exist, saved label is now used instead of ??. Added star option.	5	<code>makethm</code> : Replaced theorem label saving macro.	4
v1.4.1		<code>\repthm</code> : Added fallback for when no label is saved.	5
<code>repthm@firstoffive</code> : Added			

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	E	P
<code>\@@theoremnotdefined</code>	environments: makethm	<code>\PackageWarning</code>
140	1, <u>43</u>	17, 104, 115, 164
<code>\@thm</code>	L	R
144, 148, 152	<code>\loadtheorems</code>	<code>\reptheorem@theoremfile</code>
<code>\@thmcounter</code>	3, <u>11</u>	3, 7, 84
. 109, 113, 116, 117, 121, 122, 172	M	<code>\repthm</code>
<code>\@thmtype</code> 101, 103, 109, 110, 127, 131, 136, 147, 149, 151, 153, 158, 160, 162, 163	makethm (env.)	<code>\repthm*</code>
	1, <u>43</u>	2
<code>\@auxout</code>	N	<code>\repthm@firstoffive</code>
83	<code>\newwrite</code>	22, 22, 79
<code>\@thmlist</code>	8	<code>\repthm@ifrefexists</code>
8, 9, 89	O	25, 25, 78, 125
	<code>\openout</code>	T
	9	<code>\theoremfile</code>
		3, 3