The **nccthm** package[*]

Alexander I. Rozhenko
rozhenko@oapmg.sscc.ru

2004/12/16

Contents

---

[*] This file has version number v1.1, last revised 2004/12/16.

1 Overview

This is yet another extension of the LaTeX theorem making environment. Further, we'll call the *theorem* any mathematical statement consisting of one or more paragraphs, and starting with a header containing an optional number. A theorem is set off from the main text with an extra space. Its title and body are often emphasized with a font change.

In comparison with the `theorem` package by Frank Mittelbach and the `amsthm` package by American Mathematical Society, the `nccthm` package is distinct in the following:

1 All new theorem environments generated with the `nccthm` package use the *dynamic counters* (see `dcounter` package from this bungle for more details). As a result, this allows a package writer to generate predefined theorems and do not think on how they will be numbered, because the decision is later made by a user. For example, if you want all theorems to have enumeration subordinated to sections, you need to write down the command

```
\countstyle{section}
```

in the preamble. In this case, counters of all new theorems will be set to zero at the beginning of every section and their enumeration will be composed from the section number and the theorem counter number itself. You can also change a enumeration style for concrete counters (not only for dynamic counters) with the `\countstyle` command specifying the list of counters in its optional parameter. For example, if you prepare a book and want to subordinate the equation and table counters to sections (they are subordinated to chapters by default), write down in the preamble the following

```
\countstyle[equation,table]{section}
```

The last example: to make a plain enumeration of sections and reject their dependance on any other counter, write down in the preamble `\countstyle[section]{}`.

2 We have divided all theorem modification features into three orthogonal groups.

A *type* defines font shapes used in the theorem. Two types are predefined: *theorem* and *remark*. You can define more types or redefine existing types.

A *mode* defines the precedence of number in the header. In standard mode, a number goes after a theorem title. The reverse mode (number starts header), named here the *APAR* mode, is special. All theorems using this mode are counted with the same `apar` counter. To generate theorems in the standard mode, the `\newtheorem` command is used. The `\newtheorem*` command is used for generation *apar* theorems.

A *style* specifies the appearance of theorems. It consists of three modifiers: *breaking*, *indentation*, and *marginal numbering*. The breaking modifier determines the separation of theorem header from theorem body (run-in header or line break after header). The indentation modifier has two values: *noindent* and *parindent*. It defines an indentation style of a theorem header. The marginal numbering modifier affects on apar theorems: in *margin* style, the number of an apar theorem is typed on margins; in *nomargin* style, the number starts a paragraph.

The breaking modifier is controlled with special commands but other modifiers are specified with package options.

3 We do not base the implementation of theorems on the `trivlist` environment. This allows us to use two methods of theorem markup: environment-like markup (`\begin{theorem}` … `\end{theorem}`) or command-like markup (`\theorem` …`\qef`). The `\qef` command is used as the end of any theorem. It resets the font to the normal font and produces a vertical skip. The command-like markup is useful in apar theorems.

4 Service commands are introduced. Using package options you can select a type of Q.E.D. symbol: black or white. You can use both of them. The `\proof` command starts a proof of statement. We have no `proof` environment, because the proof is prepared in the normal font. To complete a proof, use `\qed` or `\qef` (the first one additionally prints the right adjusted Q.E.D. symbol). You can easy customize delimiters of theorem and proof headers.

2 Quod Erat Demonstrandum

The Q.E.D. symbol is usually used at the end of proof of a math statement. Sometimes, another symbol finishes a math statement itself. The package allows using two types of Q.E.D. symbols: black (■) and white (□).

`\qedsymbol` The default Q.E.D. symbol is white. It is printed with the `\qedsymbol` command.

`\qed` `\qef` The `\qed` command is used to finish a proof. It prints right-adjusted Q.E.D. symbol and applies the `\qef` command (it finishes this paragraph). The `\qef` command finishes a paragraph, changes the current font to the normal font, and skips a `\paragraph`-like vertical space. □

`\qed*` The star-form of `\qed` command prints the Q.E.D. symbol but not applies the `\qef` command.

The usage of the above mentioned commands at the end of proof is the following: if the proof end with an ordinary paragraph, use the `\qed` command at its end; if the proof ends with a list, use the `\qed*` command at the end of the last list item and the `\qef` command after the end of list; and if the proof ends with a display formula, use the `\qedsymbol` command as its tag (`\tag*{\qedsymbol}` when use the `amsmath` package).

`\blackqed` `\blackqedsymbol` If you load the package with the '`blackqed`' option, two additional commands are generated, `\blackqedsymbol` and `\blackqed`. The default `\qedsymbol` and `\qed` commands are let to be equal to their black versions.

`\whiteqed` `\whiteqedsymbol` Analogously, the '`whiteqed`' option generates the `\whiteqedsymbol` and

`\whiteqed` commands and lets the default `\qedsymbol` and `\qed` commands to be equal to their white versions.

If both these options are used, the default `\qedsymbol` and `\qed` commands are let to be equal the version loaded last. For example, the following command loads both versions of Q.E.D. and lets the white version to be the default version:

```
\usepackage[blackqed,whiteqed]{nccthm}
```

3 Other Package Options

The package loads options in the order they are written in the options list. Along with the '`blackqed`' and '`whiteqed`' options described above, the following options are available:

`noindent`	theorems are typed without indentation;
`indent`	theorems are typed with paragraph indentation;
`nomargin`	numbers for theorems in apar mode are typed normally;
`margin`	numbers for theorems in apar mode are typed on margin.

The default options are `noindent` and `nomargin`. The following examples show how the combinations of `(no)indent` and `(no)margin` options interact.

Theorem 1 *The standard theorem in the `noindent + nomargin` style.*

3.1 *Remark* The apar theorem in the `noindent + nomargin` style.

Theorem 2 *The standard theorem in the `indent + nomargin` style.*

3.2 *Remark* The apar remark in the `indent + nomargin` style.

3.3 Theorem *The apar theorem in the `noindent + margin` style.*

3.4 *Remark* The apar remark in the `indent + margin` style.

As you can see, mixing of all these styles in the same document leads to bad results. This is the reason why these styles are implemented using options.

As you can also see from the last example, the `indent` style is ignored for apar theorems typed in the `margin` style.

4 Structure and Types of Theorems

The structure of a theorem is the following:

⟨*theorem*⟩	⟨*header*⟩ ⟨*space-or-break*⟩ ⟨*body*⟩
⟨*header*⟩	⟨*normal-header*⟩ `|` ⟨*apar-header*⟩
⟨*normal-header*⟩	⟨*title*⟩ `[`⟨*number*⟩`]` `[`⟨*comment*⟩`]` ⟨*after-char*⟩
⟨*apar-header*⟩	⟨*apar-tag*⟩ ⟨*title*⟩ `[`⟨*comment*⟩`]` ⟨*after-char*⟩

A theorem type controls the appearance of the following elements of theorems: ⟨*title*⟩, ⟨*comment*⟩, and ⟨*body*⟩. The ⟨*number*⟩ element inherits the style from ⟨*title*⟩. The ⟨*after-char*⟩ element inherits either the style of ⟨*comment*⟩ if it presents or the style of ⟨*title*⟩ otherwise. The style of ⟨*apar-tag*⟩ is controlled with the special way and will be described later.

`\newtheoremtype`
`\renewtheoremtype`

The package provides the following theorem type generation commands:

```
\newtheoremtype{⟨type⟩}{⟨title-style⟩}{⟨comment-style⟩}{⟨body-style⟩}
\renewtheoremtype{⟨type⟩}{⟨title-style⟩}{⟨comment-style⟩}{⟨body-style⟩}
```

The ⟨*type*⟩ parameter is a type name. Other parameters specify fonts to be used in the corresponding parts of a theorem. Font style parameters are considered to be used after the `\normalfont` command.

`\like...`

When a new theorem type is generated, the `\like`⟨*type*⟩ command is created for it. It has two forms: normal and starred. The normal version prints a theorem in the standard mode and the starred version prints an apar theorem. The syntax:

```
\like⟨type⟩{⟨title⟩}{⟨number⟩}[⟨comment⟩]
\like⟨type⟩*{⟨title⟩}[⟨comment⟩]
```

The starred version of this command has no ⟨*number*⟩ argument, because it is numbered using the `apar` counter. If the ⟨*number*⟩ argument in the non-starred version of this command is empty, the number will be omitted in the theorem header. If the ⟨*comment*⟩ argument presents, the comment is typed enclosed in round brackets. This behaviour can be changed with customization commands.

Two theorem types, '`theorem`' and '`remark`', are predefined as follows:

```
\newtheoremtype{theorem}{\bfseries}{}{\itshape}
\newtheoremtype{remark}{\itshape}{}{}
```

`\liketheorem`
`\likeremark`

Using the `\liketheorem` and `\likeremark` commands, you can produce a theorem of the given type with arbitrary title without generation a special environment for it. It is very useful if a theorem with the given title appears in a document only once.

Note: Type generation commands are available in the preamble only.

5 Generate New Theorems

`\newtheorem`

A standard theorem environment is generated with the `\newtheorem` command:

`\newtheorem{⟨env-name⟩}[⟨counter⟩]{⟨title⟩}[⟨type⟩]`

In comparison with the standard LaTeX version of this command, the last optional parameter has another meaning: it specifies a theorem type. This is because its standard meaning (the base counter) useless here. If the ⟨*type*⟩ parameter is omitted, the '`theorem`' type is used. The optional argument ⟨*counter*⟩ is a counter this environment will be counted with. If it is omitted, the counter name equal to the ⟨*env-name*⟩ is used. We do not test the ⟨*counter*⟩ on existence when a new theorem environment is generated. The theorem counter is declared to be the dynamic counter. It is defined at the first use and inherits the style declared by the latest use of the command

`\countstyle{⟨base-counter⟩}`

Its argument contains a name of base-counter for all dynamically created counters. Dynamically created counter is set to zero when the base counter is stepped. Its `\the` command is the following:

`\the⟨base-counter⟩.\arabic{⟨dynamic-counter⟩}`

If the ⟨*base-counter*⟩ is empty, a dynamic counter will be numbered in the plain style.

Note: In contrast with the standard definition, the described `\newtheorem` command may be used with all four parameters.

`\newtheorem*`

To generate a new apar theorem environment, the starred version of the `\newtheorem` command is applied:

`\newtheorem*{⟨env-name⟩}{⟨title⟩}[⟨type⟩]`

All apar theorems are counted with the '`apar`' counter.

`\renewtheorem`
`\renewtheorem*`

You can also redefine already defined theorem environments using the commands

`\renewtheorem{⟨env-name⟩}[⟨counter⟩]{⟨title⟩}[⟨type⟩]`
`\renewtheorem*{⟨env-name⟩}{⟨title⟩}[⟨type⟩]`

While redefinition a theorem environment, you can change values of all other parameters after ⟨*env-name*⟩.

`\TheoremBreakStyle`
`\TheoremNoBreakStyle`

When a theorem environment is defined or redefined, a decision what must be inserted after the theorem header (space or break) is made on the base of current break style. The `\TheoremBreakStyle` and `\TheoremNoBreakStyle` commands change this style to the 'break' and 'no-break' respectively. The default style is 'no-break'.

Note: Theorem generation commands are available in the preamble only.

6 Using Theorems

The syntax of using theorem environments is the following:

```
\begin{⟨env-name⟩}[⟨comment⟩] ⟨body⟩ \end{⟨env-name⟩}
```

You can also use the command-like syntax:

```
\⟨env-name⟩[⟨comment⟩] ⟨body⟩ \qef
```

which is more likely for apar theorems.

`\breakafterheader`
`\nobreakafterheader`

You can change a break style for a theorem applying the `\breakafterheader` and `\nobreakafterheader` commands just before using the theorem.

Let us do the following in the preamble:

```
\countstyle[apar]{section}
\newtheorem{theorem}{Theorem}
\newtheorem*{atheorem}{Theorem}
\TheoremBreakStyle
\newtheorem{definition}{Definition}[remark]
\TheoremNoBreakStyle
\newtheorem{lemma}[theorem]{Lemma}
```

This code generates 4 theorem environments: the '`theorem`' provides a standard Theorem statement; the '`atheorem`' provides a Theorem statement in the apar mode with per-section numbering; the '`definition`' provides a standard Definition statement prepared as a remark; and the '`lemma`' provides a standard Lemma statement counted with the `theorem` counter. Definitions are printed in the break style.

```
\begin{theorem} A theorem. \end{theorem}
\begin{lemma} A lemma. \end{lemma}
\breakafterheader
\begin{theorem}[A comment] A theorem with break.\end{theorem}
\atheorem A theorem in apar mode. \qef
\begin{definition} A definition. \end{definition}
```

This code produces the following:

Theorem 1 *A theorem.*

Lemma 2 *A lemma.*

Theorem 3 (A comment)
A theorem with break.

6.1 Theorem *A theorem in apar mode.*

Definition 1
A definition.

To prepare a theorem without number or having a special number, use the `\like`⟨*type*⟩ command. Examples:

```
\liketheorem{Theorem}{A} Special theorem.\qef
\liketheorem{Proposition}{}[Comment] It has no number.\qef
\breakafterheader
\likeremark{Example}{2.3.5} An example.\qef
\likeremark*{Remark} An apar remark. \qef
```

This code produces the following:

Theorem A *Special theorem.*

Proposition (Comment) *It has no number.*

Example 2.3.5
An example.

6.2 *Remark* An apar remark.

`\proof` The `\proof` command prints the proof of a math statement. Syntax:

`\proof[`⟨*of-theorem*⟩`]` ⟨*body*⟩ `\qed`

The optional parameter ⟨*of-theorem*⟩ contains a text appended to the title of proof. The break-style change commands can be applied to this command. Examples:

```
\proof An ordinary proof.\qed
\proof[of Theorem A] A special proof.\qed
\breakafterheader
\proof[of the Pythagor Theorem] A proof.\qed
```

This code produces the following:

Proof An ordinary proof. □

Proof of Theorem A A special proof. □

Proof of the Pythagor Theorem
A proof. □

---

7 Apar Sections

Header of an apar theorem is similar to the header printed by the `\paragraph` or `\subparagraph` command (except paragraph numbering that is usually omitted). Moreover, from the logical point of view, the apar theorems are specially designed enumerated paragraphs. Therefore, it is a good idea to use apar markup as some kind of special sectioning.

`\apar` The following command supports sectioning in the apar mode:

```
\apar[⟨title⟩]
```

It produces a new paragraph starting with the ⟨*apar-tag*⟩ element and having the optional title. The indentation style and marginal style of apar section is the same as for apar theorems. The vertical skip before the apar section is just the same as before theorems. If the `nccsect` package is loaded, the apar skip is equal to the skip produced with the `\paragraph` and `\subparagraph` commands.

These properties of apar sections are useful in design of articles having short sections. For example, if an article consists of many short sections prepared with the `\subsection` command, it looks very loose, because subsections are produced in the display style. It will be better to allow subsections run-in paragraph. Using the `\apar` command, you can do this very easy: add the following command to the preamble

```
\countstyle[apar]{section}
```

and prepare subsections with the `\apar` command. Example:

```
\apar[Subsection title] Subsection body ...
\apar[Next title:] The body ...
\atheorem In fact, this is a special apar section ...
\apar Subsection without title ...
\breakafterheader
\apar[One more title] Break before its body
```

This code produces the following:

7.1 Subsection title Subsection body ...

7.2 Next title: The body ...

7.3 Theorem *In fact, this is a special apar section ...*

7.4 Subsection without title ...

7.5 One more title
Break before its body

---

8 Customization Commands

`\NCC@runskip`

The vertical skips before and after theorems are identic. They are produced with the `\qef` command. The length of the skip is coded in the inner command `\NCC@runskip`. This command is also used in the `nccsect` package as a skip inserted before the `\paragraph` and `\subparagraph` commands. Its default value

```
2.75ex plus 1ex minus 0.2ex
```

`\TheoremCommentDelimiters`

The `\TheoremCommentDelimiters{`⟨*left*⟩`}{`⟨*right*⟩`}` command specifies delimiters inserted before and after a theorem comment. The default setting is:

```
\TheoremCommentDelimiters{(}{)}
```

`\AfterTheoremHeaderChar`

The `\AfterTheoremHeaderChar{`⟨*after-char*⟩`}` command specifies an ⟨*after-char*⟩ element that ends header of theorem and header of proof. The default setting is an empty element.

`\AfterTheoremHeaderSkip`

The `\AfterTheoremHeaderSkip{`⟨*skip-command*⟩`}` command specifies a command inserted between theorem header and body. In break style, this command is ignored. The default setting is:

```
\AfterTheoremHeaderSkip{\hskip 1em plus 0.2em minus 0.2em}
```

`\ProofStyleParameters`

The `\ProofStyleParameters{`⟨*style*⟩`}{`⟨*title*⟩`}` command specifies style parameters used in the `\proof` command: the first one is a font style and the last one is a proof title. The default setting is:

```
\ProofStyleParameters{\bfseries}{Proof}
```

`\AparStyleParameters`

The `\AparStyleParameters{`⟨*style*⟩`}{`⟨*prefix*⟩`}{`⟨*suffix*⟩`}` command specifies the style of apar sections: the ⟨*style*⟩ is a style of apar section title; the tag of apar theorems and sections is prepared using ⟨*prefix*⟩ and ⟨*suffix*⟩ specified with this command as follows ⟨*prefix*⟩`\theapar`⟨*suffix*⟩. The default setting is:

```
\AparStyleParameters{\bfseries}{\bfseries}{\enskip}
```

Note: All customization commands except the `\NCC@runskip` are allowed in the preamble only.

9 The Implementation

`\NCC@secskip`
`\NCC@runskip`

The package shares the following commands with the `nccsect` package:

`\NCC@secskip{`⟨*skip*⟩`}` adds the ⟨*skip*⟩ before a section,
`\NCC@runskip` is a skip inserted before run-in sections.

We protect the definitions of these commands with testing the `nccsect` package to be already loaded.

```
⟨*package⟩
```

```
\@ifpackageloaded{nccsect}{}{%
  \def\NCC@secskip#1{%
    \if@noskipsec \leavevmode \fi \par
    \if@nobreak \everypar{}\else
      \addpenalty\@secpenalty
      \addvspace{#1}%
    \fi
  }
  \def\NCC@runskip{2.75ex \@plus 1ex \@minus .2ex}
}
```

9.1 Q.E.D. Support

`\NCC@qedsymbol`
`\qedsymbol`

The `\NCC@qedsymbol` command is the base implementation of white Q.E.D. symbol. If no Q.E.D. selection options used, the `\qedsymbol` is equal to the base version.

```
\DeclareRobustCommand{\NCC@qedsymbol}{%
  \mbox{\normalsize\normalfont\thinlines \@tempdima 1.5ex
    \advance\@tempdima-2\@wholewidth
    \edef\@tempa{\the\@tempdima}%
    \kern \@wholewidth
    \raisebox\@wholewidth[1.5ex]{%
      \frame{\rule\z@\@tempa\rule\@tempa\z@}}%
    \kern \@wholewidth
  }%
}
\let\qedsymbol\NCC@qedsymbol
```

`\qed`
`\qed*`

The `\qed` command produces the flush-right Q.E.D. symbol and applies the `\qef` command in the non-starred case.

```
\newcommand{\qed}{\NCC@qed{\qedsymbol}}
\def\NCC@qed#1{\unskip\allowbreak%
  \hspace*{1em plus 1fill minus .2em}#1\@ifstar{}{\qef}%
}
```

`\qef`

The `\qef` finishes a paragraph, adds the `\@secpenalty`, and produces the vertical skip of `\NCC@runskip` size. If also sets the normal font and removes the no-break condition suppressing indentation in the next paragraph.

```
\newcommand{\qef}{\NCC@secskip{\NCC@runskip}\@nobreakfalse\normalfont}
```

9.2 Package Options

`\whiteqedsymbol`
`\whiteqed`

The ‘`whiteqed`’ option:

```
\DeclareOption{whiteqed}{%
  \let\whiteqedsymbol\NCC@qedsymbol
  \newcommand{\whiteqed}{\NCC@qed{\whiteqedsymbol}}
  \let\qedsymbol\whiteqedsymbol
}
```

`\blackqedsymbol`
`\blackqed`

The ‘`blackqed`’ option:

```
\DeclareOption{blackqed}{%
  \DeclareRobustCommand{\blackqedsymbol}{%
    \begingroup\normalsize\normalfont\thinlines
      \rule{1.5ex}{1.5ex}\endgroup
  }
  \newcommand{\blackqed}{\NCC@qed{\blackqedsymbol}}
  \let\qedsymbol\blackqedsymbol
}
```

`\ifNCC@thmindent`
`\ifNCC@thmmargin`

Theorem indentation and marginal numbering options are based on two if-macros: the `NCC@thmindent` controls the indentation of headers and the `NCC@thmmargin` controls the marginal numbering in the apar mode.

```
\newif\ifNCC@thmindent
\newif\ifNCC@thmmargin
\DeclareOption{noindent}{\NCC@thmindentfalse}
\DeclareOption{indent}{\NCC@thmindenttrue}
\DeclareOption{nomargin}{\NCC@thmmarginfalse}
\DeclareOption{margin}{\NCC@thmmargintrue}
```

Set defaults and process all options in the order they appear in the options list.

```
\ExecuteOptions{noindent,nomargin}
\ProcessOptions*
```

9.3 The Kernel

We use the `\@ifempty` command from the `amsgen` package and dynamic counters from the `dcounter` package.

```
\RequirePackage{amsgen}
\RequirePackage{dcounter}[1998/12/19]
```

`\NCC@thmbrmode`
`\ifNCC@thmbr`

A theorem break mode is controlled with the `\NCC@thmbrmode` macro and the `\ifNCC@thmbr`. The `\NCC@thmbrmode` can have three possible values:

`\relax`	undefined mode;
`\z@`	break mode;
`\@ne`	no-break mode.

If the break mode is undefined, the decision is made on the analyzes of the `NCC@thmbr` value: true value means break, false value means no-break.

```
\let\NCC@thmbrmode\relax
\newif\ifNCC@thmbr
```

`\breakafterheader`
`\nobreakafterheader`

The `\breakafterheader` and `\nobreakafterheader` commands test the current break mode and set an appropriate mode if it is undefined yet. As a result, usage of these commands before a theorem leads to overriding the default break mode specified in theorem's macro.

```
\newcommand\breakafterheader{%
  \ifx\NCC@thmbrmode\relax \global\chardef\NCC@thmbrmode\z@\fi
}
\newcommand\nobreakafterheader{%
  \ifx\NCC@thmbrmode\relax \global\chardef\NCC@thmbrmode\@ne\fi
}
```

`\TheoremBreakStyle`
`\TheoremNoBreakStyle`
`\TheoremCommentDelimiters`
`\AfterTheoremHeaderChar`
`\AfterTheoremHeaderSkip`

Theorem customization commands:

```
\newcommand{\TheoremBreakStyle}{\NCC@thmbrtrue}
\newcommand{\TheoremNoBreakStyle}{\NCC@thmbrfalse}
\newcommand*{\TheoremCommentDelimiters}[2]{%
  \def\NCC@thmcmt@##1{#1\ignorespaces##1\unskip#2}%
}
\newcommand*{\AfterTheoremHeaderChar}[1]{\def\NCC@thmchar{#1}}
\newcommand*{\AfterTheoremHeaderSkip}[1]{\def\NCC@thmskip{#1}}
\@onlypreamble\TheoremBreakStyle
\@onlypreamble\TheoremNoBreakStyle
\@onlypreamble\TheoremCommentDelimiters
\@onlypreamble\AfterTheoremHeaderChar
\@onlypreamble\AfterTheoremHeaderSkip
```

`\NCC@thmhdr`

The `\NCC@thmhdr{`⟨*style*⟩`}{`⟨*header*⟩`}` prints a theorem header in the required ⟨*style*⟩ and implements the current break mode. At the end of macro, the break mode is reset to `\relax`. The header is prepared in a group.

```
\def\NCC@thmhdr#1#2{%
  \qef
```

Insert a negative low penalty to increase a chance of page break before the beginning of theorem.

```
  \addpenalty{-\@lowpenalty}%
  \begingroup
    #1%
```

Test the break mode. If it is undefined, we set it basing on the value of `\ifNCC@thmbr` switch.

```
    \ifx\NCC@thmbrmode\relax
      \ifNCC@thmbr \chardef\NCC@thmbrmode\z@
      \else \chardef\NCC@thmbrmode\@ne
      \fi
    \fi
```

The break case: to implement the break in the vertical mode, it is enough to put the header in hbox. To allow multi-line header, we prepare it in inner vbox.

```
    \ifnum\NCC@thmbrmode=\z@
      \@tempdima\parindent
      \hbox{\vbox{\hsize\linewidth\@parboxrestore
        \ifNCC@thmindent\parindent\@tempdima\fi
        \leavevmode\strut#2\strut
      }}\nobreak\noindent
```

The no-break case: The `\ncc@thmskip` command

```
    \else
      \ifNCC@thmindent\else\noindent\fi
      \leavevmode{#2\normalfont\NCC@thmskip}\nobreak
    \fi
  \endgroup
```

Reset the break mode to the initial value:

```
  \global\let\NCC@thmbrmode\relax
}
```

`\NCC@thmcmt` The `\NCC@thmcmt{`⟨*comment-style*⟩`}{`⟨*comment*⟩`}` tests a comment to be nonempty and produces it in corresponding style.

```
\def\NCC@thmcmt#1#2{%
  \@ifempty{#2}{}{\/\space\normalfont#1\NCC@thmcmt@{#2}}%
}
```

`\NCC@lthm` Standard mode basic command:

> `\NCC@lthm{`⟨*header-style*⟩`}{`⟨*comment-style*⟩`}{`⟨*body-style*⟩`}{`⟨*title*⟩`}`
> `{`⟨*number*⟩`}[`⟨*comment*⟩`]`

```
\def\NCC@lthm#1#2#3#4#5{%
  \@ifnextchar[{\NCC@lthm@{#1}{#2}{#3}{#4}{#5}}%
               {\NCC@lthm@{#1}{#2}{#3}{#4}{#5}[]}%
}
\def\NCC@lthm@#1#2#3#4#5[#6]{%
```

Prepare theorem header.

```
  \NCC@thmhdr{}{%
    #1#4\@ifempty{#5}{}{\space#5}%
    \NCC@thmcmt{#2}{#6}\NCC@thmchar
  }%
```

Set the body style and do a small skip to avoid extra space after the `\label` command.

```
  #3\hskip 0.001\p@ \ignorespaces
}
```

`\NCC@thxhdr` The `\NCC@thxhdr{`⟨*style*⟩`}{`⟨*header*⟩`}` produces an apar theorem header.

```
\def\NCC@thxhdr#1#2{%
  \refstepcounter{apar}%
  \NCC@thmhdr{%
```

In margin style, the indent style is turned off for apar theorems:

```
    \ifNCC@thmmargin \NCC@thmindentfalse \fi
    #1%
  }{%
```

Put the ⟨*apar-tag*⟩ (prepared in the `\NCC@thmapar` command) before the header using `\llap` or `\hbox`. Then put the header.

```
    \ifNCC@thmmargin \llap{\NCC@thmapar}\else \hbox{\NCC@thmapar}\fi
    #2%
  }%
}
```

`\NCC@lthx` Apar mode basic command:

\NCC@lthx{⟨*header-style*⟩}{⟨*comment-style*⟩}{⟨*body-style*⟩}{⟨*title*⟩}
[⟨*comment*⟩]

```
\def\NCC@lthx#1#2#3#4{%
  \@ifnextchar[{\NCC@lthx@{#1}{#2}{#3}{#4}}%
               {\NCC@lthx@{#1}{#2}{#3}{#4}[]}%
}
\def\NCC@lthx@#1#2#3#4[#5]{%
  \NCC@thxhdr{}{%
    \normalfont#1#4\NCC@thmcmt{#2}{#5}\NCC@thmchar
  }%
  #3\hskip 0.001\p@ \ignorespaces
}
```

`\NCC@likethm` The base for `\like`⟨*type*⟩ commands. It passes the control to `\NCC@lthm` or `\NCC@lthx` depending on the star appearing after the third parameter:

\NCC@likethm{⟨*header-style*⟩}{⟨*comment-style*⟩}{⟨*body-style*⟩}{⟨*title*⟩}
{⟨*number*⟩}[⟨*comment*⟩]
\NCC@likethm{⟨*header-style*⟩}{⟨*comment-style*⟩}{⟨*body-style*⟩}*{⟨*title*⟩}
[⟨*comment*⟩]

```
\def\NCC@likethm#1#2#3{%
  \@ifstar{\NCC@lthx{#1}{#2}{#3}}{\NCC@lthm{#1}{#2}{#3}}%
}
```

9.4 Apar Sections

The counter used in the apar mode is dynamic:

```
\DeclareDynamicCounter{apar}
```

`\AparStyleParameters` Apar style parameters provider:

```
\newcommand*{\AparStyleParameters}[3]{%
  \def\NCC@thmaparstyle{#1}%
  \def\NCC@thmapar{#2\theapar#3}%
}
\@onlypreamble\AparStyleParameters
```

`\apar` The `\apar[`⟨*title*⟩`]` command starts a new apar-numbered paragraph. If the ⟨*title*⟩ is omitted or empty, we must ignore the `\NCC@thmskip`.

```
\newcommand*{\apar}[1][]{%
  \NCC@thxhdr{\NCC@thmaparstyle}
    {\@ifempty{#1}{\let\NCC@thmskip\@empty}{\ignorespaces#1\unskip}}%
  \hskip 0.001\p@ \ignorespaces
}
```

9.5 Proof of Theorem

`\ProofStyleParameters` Proof style parameters provider:

```
\newcommand*{\ProofStyleParameters}[2]{%
  \def\NCC@thmproofstyle{#1}%
  \def\NCC@thmproof{#2}%
}
\@onlypreamble\ProofStyleParameters
```

`\proof` The `\proof[`⟨*of-theorem*⟩`]` command:

```
\newcommand*{\proof}[1][]{%
  \NCC@thmhdr{\NCC@thmproofstyle}{%
    \NCC@thmproof
    \@ifempty{#1}{}{\space\ignorespaces#1\unskip}%
    \NCC@thmchar
  }%
  \hskip 0.001\p@ \ignorespaces
}
```

9.6 Generate New Theorem Types

`\like...` New theorem type generation means definition a `\like`⟨*type*⟩ command preparing theorems of corresponding type. The syntax of a `\like`⟨*type*⟩ command is the following:

> `\like`⟨*type*⟩`{`⟨*title*⟩`}{`⟨*number*⟩`}[`⟨*comment*⟩`]`
> `\like`⟨*type*⟩`*{`⟨*title*⟩`}[`⟨*comment*⟩`]`

The first one produces a standard theorem and the last one produces an apar theorem.

`\newtheoremtype`
`\renewtheoremtype` Theorem type generation commands:

> `\newtheoremtype{`⟨*type*⟩`}{`⟨*title-style*⟩`}{`⟨*comment-style*⟩`}{`⟨*body-style*⟩`}`
> `\renewtheoremtype{`⟨*type*⟩`}{`⟨*title-style*⟩`}{`⟨*comment-style*⟩`}{`⟨*body-style*⟩`}`

```
\newcommand*{\newtheoremtype}[1]{%
  \edef\@tempa{\noexpand\newcommand*{\expandafter\noexpand
    \csname like#1\endcsname}}\NCC@nthmtype
}
\newcommand*{\renewtheoremtype}[1]{%
  \edef\@tempa{\noexpand\renewcommand*{\expandafter\noexpand
    \csname like#1\endcsname}}\NCC@nthmtype
}
```

```
\def\NCC@nthmtype#1#2#3{\@tempa{\NCC@likethm{#1}{#2}{#3}}}
\@onlypreamble\newtheoremtype
\@onlypreamble\renewtheoremtype
\@onlypreamble\NCC@nthmtype
```

9.7 Generate New Theorems

`\NCC@thmdef` Basic theorem generation command:

`\NCC@thmdef{`⟨*env-name*⟩`}{`⟨*action*⟩`}{`⟨*parameters*⟩`}[`⟨*type*⟩`]`

The `\@tempa` command must contain either `\noexpand\newenvironment` or `\noexpand\renewenvironment` before the call. The ⟨*action*⟩ is an action applied at the beginning of theorem. The ⟨*parameters*⟩ contains parameters passed to the `\like`⟨*type*⟩ command.

```
\def\NCC@thmdef#1#2#3{%
  \@ifnextchar[{\NCC@thmdef@{#1}{#2}{#3}}%
               {\NCC@thmdef@{#1}{#2}{#3}[theorem]}%
}
\def\NCC@thmdef@#1#2#3[#4]{%
```

Generate an error if the given type is unknown.

```
  \@ifundefined{like#4}{%
    \PackageError{nccthm}{Unknown theorem type `#4'}{}%
  }%
```

`\@tempa := \[re]newenvironment{`⟨*env-name*⟩`}{#1\like`⟨*type*⟩`#2}`

```
  \edef\@tempa##1##2{%
    \@tempa{#1}{##1\expandafter\noexpand\csname like#4\endcsname##2}%
  }%
```

Generate a theorem envirinment:

```
  \ifNCC@thmbr
    \@tempa{#2\breakafterheader}{#3}{\qef\ignorespacesafterend}%
  \else
    \@tempa{#2\nobreakafterheader}{#3}{\qef\ignorespacesafterend}%
  \fi
}
\@onlypreamble\NCC@thmdef
\@onlypreamble\NCC@thmdef@
```

`\newtheorem`
`\renewtheorem`
`\newtheorem*`
`\renewtheorem*`

Theorem generation commands:

`\newtheorem{`⟨*env-name*⟩`}[`⟨*counter*⟩`]{`⟨*title*⟩`}[`⟨*type*⟩`]`
`\renewtheorem{`⟨*env-name*⟩`}[`⟨*counter*⟩`]{`⟨*title*⟩`}[`⟨*type*⟩`]`
`\newtheorem*{`⟨*env-name*⟩`}{`⟨*title*⟩`}[`⟨*type*⟩`]`
`\renewtheorem*{`⟨*env-name*⟩`}{`⟨*title*⟩`}[`⟨*type*⟩`]`

```
\renewcommand*{\newtheorem}{\def\@tempa{\noexpand\newenvironment}%
  \@ifstar{\NCC@nthx}{\NCC@nthm}}
\newcommand*{\renewtheorem}{\def\@tempa{\noexpand\renewenvironment}%
```

```
  \@ifstar{\NCC@nthx}{\NCC@nthm}}
\def\NCC@nthx#1#2{\NCC@thmdef{#1}{}{*{#2}}}
\def\NCC@nthm#1{\@ifnextchar[{\NCC@nthm@{#1}}{\NCC@nthm@{#1}[#1]}}
\def\NCC@nthm@#1[#2]#3{%
  \DeclareDynamicCounter{#2}%
  \NCC@thmdef{#1}{\refstepcounter{#2}}{{#3}{\csname the#2\endcsname}}%
}
\@onlypreamble\newtheorem
\@onlypreamble\renewtheorem
\@onlypreamble\NCC@nthx
\@onlypreamble\NCC@nthm
\@onlypreamble\NCC@nthm@
```

9.8 Defaults

```
\newtheoremtype{theorem}{\bfseries}{}{\itshape}
\newtheoremtype{remark}{\itshape}{}{}
\TheoremNoBreakStyle
\TheoremCommentDelimiters{(}{)}
\AfterTheoremHeaderChar{}
\AfterTheoremHeaderSkip{\hskip 1em \@plus .2em \@minus .2em}
\AparStyleParameters{\bfseries}{\bfseries}{\enskip}
\ProofStyleParameters{\bfseries}{Proof}
⟨/package⟩
```