Miscellaneous Formatting (`format.dtx`)

Charles Duan

May 3, 2023

This section describes various formatting utility macros that are used when defining citation formats.

1 Ordinal Numbers

In several situations it is useful to convert numbers to text. Several macros do this, and can add ordinal number suffixes as well.

Produces a textual representation of an ordinal number:

1	First
10	Tenth
45	Forty-Fifth
123	123d
10000	Ten Thousandth

The argument must be the number–in numerical form, not a count register. There is no error checking, so make sure that the input is really a pure number.

```
\def\hi@numtotxt#1{%
    \ifnum#1<100
        \hi@numtotxt@xx#1\@stop
    \else
        \hi@numtotxt@big#1\@stop
    \fi
    \hi@numtoord{#1}%
}
\def\hi@numtotxt@numeric#1{%
    \hi@numtotxt@big#1\@stop
    \hi@numtoord{#1}%
}
\def\hi@numtoord#1{\hi@numtoord@#1\@stop\@empty}
\def\hi@numtoord@#1#2\@stop{%
    \ifstrempty{#2}{%
        \ifcase#1 th\or st\or d\or d\else th\fi
    }{%
        \@ifonechar{#2}{\ifnum#1=\@ne th\expandafter\hi@eatto@empty\fi}{}%
        \hi@numtoord@#2\@stop
    }%
}
% Formats a number between zero and 99.
\def\hi@numtotxt@xx#1#2\@stop{%
    \ifstrempty{#2}{%
        \ifcase#1 Zero\or Fir\or Secon\or Thir\or Four\or Fif\or Six\or Seven\or
            Eigh\or Nin\fi
    }{%
        \ifnum #1=\@ne
            \ifnum #2=9 Nineteen\else
                \ifcase #2 Ten\or Eleven\or Twelf%
                \else \hi@numtotxt@xx#2\@stop teen\fi
            \fi
        \else
            \hi@numtotxt@c#1#2\@stop
        \fi
```

```
    }%
}
% Formats a tens digit between 20 and 90.
\def\hi@numtotxt@c#1#2\@stop{%
    \hi@numtotxt@tens#1{}%
    \ifnum#2=\z@ ie\else y-\hi@numtotxt@xx#2\@stop\fi
}
% Tens digit. The second argument specifies the trailing letter (the "y" that is
% common to all tens digit values is not there) if the value is not 1(0).
\def\hi@numtotxt@tens#1#2{%
    \ifcase #1 \or Ten\or Twent\or Thirt\or Fort\or Fift\or Sixt\or Sevent\or
        Eight\or Ninet\fi
    \ifnum#1=\@ne\else#2\fi
}
% Single units.
\def\hi@numtotxt@ones#1{%
    \ifcase#1 \or One\or Two\or Three\or Four\or Five\or Six\or Seven\or
        Eight\or Nine\fi
}
%
% Formats a number over 100. The rule: if the number is a single digit and then
% zeroes, then write the number as text. Otherwise write it numerically.
%
% This macro uses two tricks: first, it compares the number (without the first
% digit) to zero. Then, it uses the \@expand macro in an innovative way.
% \@expand will expand its second argument once for every token in the third
% argument. So we set the third argument to the list of zeroes, and define a
% ``chain'' of macros reflecting powers of ten. The length of the zeroes list
% will determine how far to expand, and then the remaining macro will tell us
% how to format the text.
\def\hi@numtotxt@big#1#2\@stop{%
    \ifnum0#2=\z@
        \@expand{\hi@numtotxt@bigtonum#1{#2}}\hi@numtotxt@One{#2}%
    \else\hi@numtotxt@comma{#1#2}\fi
}
\def\hi@numtotxt@bigtonum#1#2#3{%
    \ifx#3\relax
        \hi@numtotxt@comma{#1#2}%
    \else
        \@ifundefined{\expandafter\@gobble\string#3@prn}{%
            \hi@numtotxt@ones#1 \expandafter\hi@numtotxt@gobble\string#3\@stop
        }{%
            \@nameuse{\expandafter\@gobble\string#3@prn}{#1}%
        }%
    \fi
}
%
% This defines \hi@numtotxt@gobble, which gobbles the \hi@numtotxt@ that
% precedes the power-of-ten name in the macro.
\edef\act{%
    \noexpand\def
    \noexpand\hi@numtotxt@gobble\string\hi@numtotxt@##1\noexpand\@stop{##1}%
}\act
\def\hi@numtotxt@One{\hi@numtotxt@Ten}
\def\hi@numtotxt@Ten{\hi@numtotxt@Hundred}
\def\hi@numtotxt@Hundred{\hi@numtotxt@Thousand}
\def\hi@numtotxt@Thousand{\hi@numtotxt@tthousand}
\def\hi@numtotxt@tthousand@prn#1{\hi@numtotxt@tens#1{y} Thousand}
\@expand{\def\hi@numtotxt@tthousand}{\csname hi@numtotxt@Hundred
    Thousand\endcsname}{i}
\@namedef{hi@numtotxt@Hundred Thousand}{\hi@numtotxt@Million}
\def\hi@numtotxt@Million{\relax}
%
% Adds commas to a number, in groups of three. Uses a similar trick with
% \@expand for token-counting, and a three-chain circular link of macros, which
% dictate where to place the first comma.
\def\hi@numtotxt@comma#1{%
    \ifnum#1<\@M #1%
    \else\@expand{\hi@numtotxt@comma@{#1}}\hi@numtotxt@commaiii{#1}\fi
}
\def\hi@numtotxt@comma@#1#2{%
    \@nameuse{\expandafter\@gobble\string#2@do}#1\@stop
}
\def\hi@numtotxt@commai{\hi@numtotxt@commaii}
\def\hi@numtotxt@commaii{\hi@numtotxt@commaiii}
\def\hi@numtotxt@commaiii{\hi@numtotxt@commai}
\def\hi@numtotxt@commai@do#1#2\@stop{#1,\hi@numtotxt@commaiii@do#2\@stop}
\def\hi@numtotxt@commaii@do#1#2#3\@stop{#1#2,\hi@numtotxt@commaiii@do#3\@stop}
\def\hi@numtotxt@commaiii@do#1#2#3#4\@stop{%
    #1#2#3\ifstrempty{#4}{}{,\hi@numtotxt@commaiii@do#4\@stop}%
}
```

2 Trailing Dot Management

When the text of a formatted citation item ends with a dot but the citation string is followed by a period, the extraneous dot needs to be removed. To do so, this package provides an infrastructure for tracking whether a citation item ends with a dot. Reference definer macros must use this infrastructure to indicate when the last item they emit to the document is a dot.

```
% Conditional for whether or not the last thing printed was a dot. This is used
% to eliminate double-dotting after abbreviations and such.
\newif\if@hi@dot
\def\hi@dot@after@group{%
    \if@hi@dot\aftergroup\@hi@dottrue\else\aftergroup\@hi@dotfalse\fi
}
%
% Macro to set the spacefactor for dots to 1000.
\def\@no@dot@space{\sfcode`\.\@m\sfcode`\:\@m\sfcode`\?\@m\sfcode`\!\@m}
%
% Restores spacefactor after a dot.
\def\@re@dot@space{\spacefactor\sfcode`\.\relax}
%
% Eats up the next character if it's a dot. Useful if the last thing you just
% wrote was a dot.
\def\@gobble@dot{\futurelet\@dot@check\@@gobble@dot}
\def\@@gobble@dot{%
    \ifx\@dot@check.%
        \@re@dot@space
        \expandafter\@gobble
    \fi
}
%
% Tests whether [text] ends with a dot.
\make@find@end{.}
\def\if@endswithdot#1#2{\find@end{.}{#1}{\@firstoftwo{#2}}{}}%
%
% Set \@hi@dottrue or \@hi@dotfalse depending on whether the text contains a
% dot. If the text is empty, no change is made (as it is assumed that
% \@hi@dotfalse was correctly set).
\def\hi@dot@set#1{%
    \find@end{.}{#1}{\@hi@dottrue\@gobble}{\ifstrempty{#1}{}{\@hi@dotfalse}}%
}
%
% \hi@dot@set using \@this@page as the argument.
\def\hi@dot@set@page{%
    \expandafter\hi@dot@set\expandafter{\@this@page}%
}
%
% For use inside definition macros: #1 is a macro (e.g. \hi@kv@[param]); tests
% if the content of the macro ends with a dot.
\def\hi@testfordot#1{%
    \expandafter\if@endswithdot\expandafter{#1}{\protect\@hi@dottrue}%
}
```

3 Capitalization

At the beginnings of sentence citations, the first letter needs to be capitalized.[1] The first letter could be part of a signal or text within the citation text itself, and the citation text might contain prefatory font or other macros. Thus, this package provides an infrastructure for citation macros to capitalize the first letter of displayed text when necessary, but not to further capitalize subsequent text.

The font macros in `fonts.dtx` use the capitalization system extensively.

Conditionals for managing properties of the citation:

[1] Capitalization also occurs inside sentence citations between signal classes, as explained in `signals.dtx`.

Whether a letter needs to be capitalized ASAP. This is turned on only in the sentence citation form. It is turned off immediately upon printing a citation party, and the signals use the `\@capnext` macro to capitalize a single letter and then turn it off.

```
\newif\if@hi@cap
\DeclareRobustCommand\hi@nocap{\@hi@capfalse}
```

The `\@capfont` and `\@capnext` macros are the foundation of capitalization management in this package.

```
\def\@capfont#1#2{%
    \@test\if@hi@cap\fi{\hi@nocap#1{\@@capnext#2}}{#1{#2}}%
}
\DeclareRobustCommand\@capnext{\if@hi@cap\hi@nocap\expandafter\@@capnext\fi}
\def\@@capnext#1{%
    \@ifonechar{#1}{%
        % Save the first character in \hi@capnext@char, and then collect the
        % subsequent character in \@let@token. Then run \@@@capnext.
        \def\hi@capnext@char{#1}%
        \futurelet\@let@token\@@@capnext
    }{%
        {\@@capnext #1}%
    }%
}
```

Determines how to capitalize the token stored in `\hi@capnext@char`. If it is a known definition, then use that. Otherwise, process the token based on whether it is a control sequence or a letter (anything else).

```
\def\@@@capnext{%
    \@ifundefined{hi@cncs@\expandafter\string\hi@capnext@char}{%
        \@test \ifcat\relax\expandafter\noexpand\hi@capnext@char \fi
            \hi@capnext@macro
            \hi@capnext@letter
    }{%
        \@nameuse{hi@cncs@\expandafter\string\hi@capnext@char}%
    }%
}
```

Where attempting to capitalize an unknown macro (or two, see `\hi@capnext@protect`). See if `\@let@token` indicates the start of a group. If so, then cause capitalization inside the group. Otherwise, capitalize the token after the macro.

```
\def\hi@capnext@macro{%
    \@test\ifcat\noexpand\@let@token\bgroup\fi{%
        \hi@capnext@macro@group
    }{%
        \hi@capnext@char\@@capnext
    }%
}
\def\hi@capnext@macro@group#1{%
    \hi@capnext@char{\@@capnext #1}%
}
```

Given that `\hi@capnext@char` is a letter, determines whether to capitalize it. If `\@let@token` is a letter and it is uppercase, then do not capitalize it. Otherwise, capitalize it.

```
\def\hi@capnext@letter{%
    \@ifletter{\@let@token}{%
        \hi@capnext@letter@next
    }{%
        \MakeUppercase{\hi@capnext@char}%
    }%
}
\def\hi@capnext@letter@next#1{%
    \@test \ifnum\uccode`#1=`#1\relax\fi {\hi@capnext@char}{%
        \MakeUppercase{\hi@capnext@char}%
    }#1%
}
```

These define `\hi@cncs@`⟨*macro*⟩ macros that convert particular command sequences into capitalized forms. It also defines `\hi@cncs@`⟨*macro*⟩⟨*space*⟩ identically, to capture robust commands (see `\hi@capnext@protect`).

```
\def\hi@capnext@makecs#1#2{%
    \@namedef{hi@cncs@\string #1}{#2}%
    \@namedef{hi@cncs@\string #1 }{#2}%
}
%
% For \protect, gather two tokens and essentially run capitalization on them as
```

```
% an unknown macro.
\hi@capnext@makecs{\protect}{\hi@capnext@protect}
\def\hi@capnext@protect#1{%
    \@ifundefined{hi@cncs@\string#1}{%
        \def\hi@capnext@char{\protect#1}%
        \futurelet\@let@token\hi@capnext@macro
    }{%
        \@nameuse{hi@cncs@\string#1}%
    }%
}
%
% Capitalize \hi@inline@the
%
\hi@capnext@makecs\hi@inline@the{\@hi@captrue\hi@inline@the}
%
% Capitalize LaTeX macros for letters
%
\def\reserved@a#1#2{%
    \ifx#1\@nnil\else
        \hi@capnext@makecs#1{\hi@capnext@letter}%
        \hi@capnext@makecs#2{#2}%
        \expandafter\reserved@a
    \fi
}
\expandafter\reserved@a\@uclclist\@nnil\@nnil
%
% Capitalize \textsection or \textparagraph. If inline, then replaces with a
% textual form (given in #1), pluralizing as appropriate by looking at the next
% character.
%
\def\hi@capnext@sp#1{%
    \@test\if@hi@inline\fi{%
        #1%
        \@test\expandafter\ifx\hi@capnext@char\@let@token\fi{%
            s\@gobble % Gobble the macro otherwise in \@let@token
        }{}%
    }{\hi@capnext@char}% If not inline, then just put the original char back
}
\hi@capnext@makecs\textsection{\hi@capnext@sp{Section}}
\hi@capnext@makecs\textparagraph{\hi@capnext@sp{Paragraph}}
```

4 “The”

Statutes and other reference names may start with the word “The.” When those are used inline as adjectives, the word “The” generally ought to be removed. This is done by replacing the leading definite article with a special macro that displays as text in non-inline citations but may be suppressed in adjective inline citations.

Puts the word “the” in front of an inline citation (e.g., for statutes). The rules are that (1) ”the” is only placed in inline citations (not note, sentence, or clause forms), and not for adjectives. If no “the” is placed, then `\@capnext` is added to capitalize the next word as appropriate.

```
\DeclareRobustCommand\hi@inline@the{%
    \@test\if@hi@inline\fi{%
        \@test\if@hi@adjective\fi{\@capnext}{%
            % Conditional doesn't need to be in |\@test| because |\@capnext| is
            % guaranteed to capitalize a word if run
            \if@hi@emph \emph{\@capnext the }\hi@nocap\else \@capnext the \fi
        }%
    }{\@capnext}%
}
```

Replaces the word “the” at the start of a text with the macro `\hi@inline@the`, which will insert and properly capitalize the word ”the” in different situations as described below. #1 is a macro that expands to the text, #2 is the callback to execute if the text starts with “the” (given one argument of the modified text), and #3 is the callback if not (given no arguments).

```
\def\hi@replacethe#1#2#3{%
    \@expand{%
        \find@try\find@start{%
            {The }{\hi@replacethe@{#2}}%
            {the }{\hi@replacethe@{#2}}%
        }%
    }{#1}i{#3}%
```

```
}
\def\hi@replacethe@#1#2{%
    #1{\hi@inline@the#2}%
}
\make@find@start{The }
\make@find@start{the }
```

"Undoes" `\hi@replacethe` by finding `\hi@inline@the` at the beginning of a text, replacing it with the given argument. #1 is the text to search, #2 the replacement text, #3 the callback if successful (given one argument of the modified text), #4 the callback if not found.

```
\def\hi@replaceinlinethe#1#2#3#4{%
    \find@start@cs{/hi@inline@the}{#1}{%
        \hi@replaceinlinethe@{#2}{#3}%
    }{#4}%
}
\def\hi@replaceinlinethe@#1#2#3{#2{#1#3}}
\make@find@start@cs{/hi@inline@the}{\hi@inline@the}
```

5 Other Macros

A variety of other macros deal with using citation parameters, differentiating between citation contexts (inline, nameless, optional arguments), and spacing.

```
\def\hi@name@comma{\if@hi@name\hi@font@comma\fi}
\DeclareRobustCommand\hi@font@comma{%
    \ifdim\lastkern>\z@\unkern\fi, \@hi@dotfalse
}
\def\hi@param@optspc#1#2{%
    \hi@ifset#1{%
        \hi@ifset#2{%
            #1\space#2
        }{#1}%
    }{\hi@ifset#2{#2}{}}%
}
%
% This macro should wrap the main textual contents of a citation, excluding
% parentheticals. It places the contents inside a group, removes large spaces
% after periods, and ensures that subsequent text is not capitalized.
%
\DeclareRobustCommand*\hi@citeguts[1]{%
    \begingroup
        \@no@dot@space
        #1%
        \hi@dot@after@group
    \endgroup
    \hi@nocap
}
%
% Inserts text into a citation macro that will use \hi@kv@defaultopt when
% appropriate.
%
\def\hi@maybeusedefaultopt{%
    \hi@ifset\hi@kv@defaultopt{%
        \noexpand\hi@maybeusedefaultopt@{\hi@kv@defaultopt}%
    }{}%
}
\def\hi@maybeusedefaultopt@#1{%
    \ifx\@this@opt\relax \def\@this@opt{#1}\fi
}
%
% Tests if the optional argument contains a certain character. (A \make@find@in
% on the character must be given in advance.) #1 is the character to find; #2 is
% the default value for the optional argument when none is given. (#3 and #4 are
% implicitly the callbacks when the character is or is not found.)
%
\DeclareRobustCommand\hi@ifinopt[2]{%
    \@test \ifx\@this@opt\relax\fi {%
        \find@in{#1}{#2}{\expandafter\@firstoftwo\@gobbletwo}{\@secondoftwo}%
    }{%
        \@expand{\find@in{#1}}\@this@opt i
            {\expandafter\@firstoftwo\@gobbletwo}{\@secondoftwo}%
    }%
}
%
%
% Pushes stuff onto the end of a token list.
\def\push@toks#1#2{%
```

```
    \long\edef\push@toks@##1{\noexpand#1{\the#1##1}}\push@toks@{#2}%
}
%
% Eat everything up to \@empty.
\def\hi@eatto@empty#1\@empty{}
%
\def\@secondofthree#1#2#3{#2}%
%
% Use \@this@vol, appending a space following it. If no volume was given, then
% use whatever is in the argument.
\def\hi@usevol#1{\ifx\@this@vol\relax#1\else\@this@vol\space\fi}
%
% Titles for statutes and regulations (for purposes of minimizing repetition).
\def\hi@set@title#1{\def\@this@title{#1}}
\def\hi@if@title#1{%
    \def\@this@title{#1}%
    \ifx\@this@title\@last@title  \expandafter\@firstoftwo
    \else                         \expandafter\@secondoftwo  \fi
}
%
% Within citation macros, these macros facilitate determining whether an inline
% citation is currently being constructed.
\DeclareRobustCommand\hi@inline@only[1]{%
    \if@hi@inline#1\fi
}
\DeclareRobustCommand\hi@inline@never[1]{%
    \if@hi@inline\else #1\fi
}
\DeclareRobustCommand\hi@toa@never[1]{%
    \if@hi@in@toa\else #1\fi
}
\DeclareRobustCommand\hi@name@only[1]{%
    \if@hi@name #1\fi
}
```