Simplified Input (`cparse.dtx`)

Charles Duan

May 3, 2023

While all references can be defined using the parameter-value input described previously, that input is fairly verbose. To simplify the entry process, common reference types accept an alternate form of input, in which the elements of the reference are given in a citation-like string. The package parses that string to extract parameters, thereby defining the reference.

To use this simplified input, ensure that the reference type is one of the types listed in this section, and then enter the simplified input string in lieu of the parameter-value pairs as follows:

```
\defcase{baker}{
    Baker v. Selden, 101 U.S. 99 (1880)
}
```

Please ensure that the simplified input has no equal signs. If there are additional parameters for the reference definition, they may be entered after a semicolon as follows:

```
\defcase{baker}{
    Baker v. Selden, 101 U.S. 99 (1880);
    docket=No. 95
}
```

1 Common Elements

The simplified input formats use several common elements, described below.

1(a) Parentheticals Text at the end of a formatted reference input surrounded by parentheses is detected and interpreted as a parenthetical. In the specifications for reference type parsers below, text matching this form will be identified as content inside parentheses.

Helper for pulling off trailing parenthetical. Takes two callbacks: #2 is run with the contents of the parenthetical (if any), and #3 is run with the remaining content (or the whole content if there was no parenthetical).

```
\def\hi@cp@paren#1#2#3{%
    \find@end{)}{#1}{\hi@cp@paren@{#2}{#3}}{#3{#1}}%
}
\def\hi@cp@paren@#1#2#3{%
    \find@last{ (}{#3}{\hi@cp@paren@@{#1}{#2}}{#2{#3)}}%
}
\def\hi@cp@paren@@#1#2#3#4{#1{#4}#2{#3}}
```

The finders are defined at the end because of syntax highlighting problems.

1(b) Volume/Reporter/Page Many of the reference type parsers will detect this pattern commonly found in reported case citations and consecutively-paginated journal citations:

> ⟨*volume*⟩ ⟨*reporter*⟩ ⟨*page*⟩ (⟨*date*⟩)
>
> *Example:*
> `123 F.3d 456 (2000)`

Text matching this format can be detected and separated into parts, according to the following algorithm:

1. If there is a parenthetical, then remove it and use it as the date.
2. The first word of the input is the volume.
3. If the second-to-last word is a division name (see `pages.dtx`), then the last two words are the page. Otherwise, just the last word is the page.
4. Any words remaining are the reporter/journal name.

Alternatively, the volume, reporter, and page elements can be separated with slashes (i.e., ⟨*volume*⟩/⟨*reporter*⟩/⟨*page*⟩).

In explaining reference inputs below, text matching this form will be identified as ⟨*vol-rep-page*⟩.

When no slashes are present, the following determination is used to split off the first and third parts:

- The first part is the first word (defined as text with no space)
- The last part is the last word, possibly including the second-to-last word if the second-to-last word is a subdivision (Table 17)
- The middle word is everything else

In either form, an optional year in parentheses may be provided.
Takes five arguments. The first is the text, and the remaining four are callbacks for the parts.

```
\def\hi@cp@cite#1#2#3#4#5{%
    \hi@cp@paren{#1}{#5}{\hi@cp@cite@text{#2}{#3}{#4}}%
}
\def\hi@cp@cite@text#1#2#3#4{\hi@cp@cite@text@{#4}{#1}{#2}{#3}}
\def\hi@cp@cite@text@#1{%
    \find@in{/}{#1}{\hi@cp@cite@slash}{%
        \find@in{ }{#1}{\hi@cp@cite@space}{%
            \PackageError\hi@pkgname{%
                Cite parameter `#1' needs two slashes or spaces%
            }{Should be `cite=.../.../...'}%
        }%
    }%
}
\make@find@in{/}
\def\hi@cp@cite@slash#1#2{%
    \find@last{/}{#2}{\hi@cp@cite@set{#1}}{%
        \PackageError\hi@pkgname{%
            Cite parameter `#1' needs two slashes%
        }{Should be `cite=.../.../...'}%
    }%
}%
\def\hi@cp@cite@space#1#2{%
    \find@last{ }{#2}{\hi@cp@cite@space@{#1}}{%
        \PackageError\hi@pkgname{%
```

```
            Cite parameter `#1' needs two spaces%
        }{Should be `cite=.../.../...'}%
    }%
}
\def\hi@cp@cite@space@#1#2#3{%
    \find@last{ }{#2}{%
        \hi@cp@cite@space@@{#1}{#3}%
    }{\hi@cp@cite@set{#1}{#2}{#3}}%
}
\def\hi@cp@cite@space@@#1#2#3#4{%
    \@ifundefined{hi@div@#4}{%
        \hi@cp@cite@set{#1}{#3 #4}{#2}%
    }{%
        \hi@cp@cite@set{#1}{#3}{#4 #2}%
    }%
}
\def\hi@cp@cite@set#1#2#3#4#5#6{%
    \ifstrempty{#1}{}{#4{#1}}%
    \ifstrempty{#2}{}{#5{#2}}%
    \ifstrempty{#3}{}{#6{#3}}%
}
```

1(c) Journal-Like Titles Many reference types can accept parseable input of the form:

> ⟨*journal-text*⟩ := ⟨*authors*⟩ **,** ⟨*title*⟩ **,** ⟨*journal-id*⟩
> ⟨*authors*⟩ := ⟨*author*⟩ [**&** ⟨*author*⟩]
>
> *Example:*
> `John Doe & Jane Q. Public, A Theory of Law, ...`

The author and title parameters are automatically set from this, and ⟨*journal-id*⟩ is passed back to the reference type parser for further analysis. In parsing this text, ⟨*authors*⟩ is taken to be anything before the first comma in the text, and ⟨*journal-id*⟩ is text following the last comma. (Thus, ⟨*journal-text*⟩ must contain at least two commas.)

Text matching this form will be denoted as ⟨*journal-text*⟩ in explaining the parsers below, and those explanations will further describe how ⟨*journal-id*⟩ is used.

```
\def\hi@cparse@jrnart@text#1{%
    \find@in{, }{#1}{\hi@cparse@jrnart@author}{%
        \PackageError\hi@pkgname{%
            No commas found in citation `#1'%
        }{You need to enter a citation with author and citation.}%
    }%
}
\make@find@in{, }
\def\hi@cparse@jrnart@author#1#2{%
    \hi@cparse@jrnart@findauthor{#1}%
    \hi@cparse@jrnart@title{#2}%
}
\make@find@in{ & }
\make@find@in{ and }
\def\hi@cparse@jrnart@findauthor#1{%
    \find@try\find@in{%
        { & }{%
            \@tworun\hi@cparse@jrnart@findauthor\hi@cparse@jrnart@findauthor
        }%
        { and }{%
            \@tworun\hi@cparse@jrnart@findauthor\hi@cparse@jrnart@findauthor
        }%
    }{#1}{\KV@hi@author{#1}}%
}
\def\hi@cparse@jrnart@title#1{%
    \find@last{, }{#1}{\hi@cparse@jrnart@cite}{%
        \PackageError\hi@pkgname{%
            No commas found in citation `#1'%
```

```
        }{You need to enter the citation with author and citation.}%
    }%
}
\def\hi@cparse@jrnart@cite#1#2{%
    \KV@hi@name{#1}\hi@cparse@jrnart@citemacro{#2}%
}
```

1(d) URLs Any parseable reference input can have a URL appended to it, set off with a comma. (The URL must start with `http://` or `https://`.) The URL will be extracted from the input text and automatically set as parameter url. Because every parseable reference type will accept this, it is not shown in the explanations below. The URL must be the last item in the parseable reference input.

```
\def\hi@cparse@url#1#2#3{%
    \KV@hi@url{#1#3}%
    \hi@cparse{#2}%
}
\make@find@in{, http://}
\make@find@in{, https://}
```

Generic method for parsing cites. This is the main entry point for parseable reference input. It relies on `\hi@kv@kind` being set, to choose the macro `\hi@cparse@`⟨*kind*⟩ to run after URL parsing.

```
\def\hi@cparse#1{%
    \find@try\find@in{%
        {, http://}{\hi@cparse@url{http://}}%
        {, https://}{\hi@cparse@url{https://}}%
    }{#1}{%
        \@ifundefined{hi@cparse@\hi@kv@kind}{%
            \PackageError\hi@pkgname{No automatic parser for \hi@kv@kind}{%
                No automatic parser for that citation type has been\MessageBreak
                created. You must enter the parameters manually.
            }%
        }{\@nameuse{hi@cparse@\hi@kv@kind}{#1}}%
    }%
}
```

2 Cases

The case reference type can parse the following input syntax:

⟨*case*⟩ := ⟨*case-name*⟩ `,` ⟨*case-cite*⟩ `(` ⟨*court*⟩ ⟨*date*⟩ `)`
⟨*case-name*⟩ := ⟨*p*⟩ `v.` ⟨*d*⟩ ‖ ⟨*name*⟩
⟨*case-cite*⟩ := ⟨*vol-rep-page*⟩ ‖ `No.` ⟨*docket*⟩

Example:

```
\defcase{wheaton}{Wheaton v. Peters, 33 U.S. 591
(1834)}
```

This sets parameters p and d or name as the case party name(s); vol, rep, and page or docket from the locator information (detected as the last pre-parenthetical text following a comma), and court and date from the parenthetical.

```
\def\hi@cparse@case#1{%
    \hi@cp@paren{#1}{\hi@cparse@case@paren}{\hi@cparse@case@text}%
}
\let\hi@cparse@admincase\hi@cparse@case
\make@find@in{, }
\def\hi@cparse@case@text#1{%
    \find@last{, }{#1}{\@tworun\hi@cparse@case@name\hi@cparse@case@cite}{%
        \PackageError\hi@pkgname{%
            No comma found when parsing case parameters%
```

```
        }{The provided parameters do not look like a case citation for parsing}%
    }%
}
\make@find@start{No}
\def\hi@cparse@case@cite#1{%
    \find@start{No}{#1}{%
        \KV@hi@slipop{}\KV@hi@docket{#1}\@gobble
    }{\KV@hi@cite{#1}}%
}
\make@find@in{ }
\def\hi@cparse@case@paren#1{%
    \find@in{ }{#1}{%
        \hi@cparse@case@paren@
    }{\KV@hi@year{#1}}%
}
\def\hi@cparse@case@paren@#1#2{%
    \@ifundefined{hi@month@#1}{%
        \hi@cparse@case@paren@@{#1}{#2}%
    }{\KV@hi@year{#1 #2}}%
}
\def\hi@cparse@case@paren@@#1#2{%
    \find@in{ }{#2}{%
        \hi@cparse@case@paren@@@{#1}%
    }{\KV@hi@court{#1}\KV@hi@year{#2}}%
}
\def\hi@cparse@case@paren@@@#1#2#3{%
    \@ifundefined{hi@dtmac@#2}{%
        \hi@cparse@case@paren@@{#1 #2}{#3}%
    }{\KV@hi@court{#1}\KV@hi@year{#2 #3}}%
}
\make@find@in{ v. }
\def\hi@cparse@case@name#1{%
    \find@in{ v. }{#1}{\hi@cparse@case@parties}{\KV@hi@name{#1}}%
}
\def\hi@cparse@case@parties#1#2{%
    \KV@hi@p{#1}\KV@hi@d{#2}%
}
```

3 Codified Statutes

The statcode reference type can parse input conforming to ⟨*vol-rep-page*⟩. For example:

```
\defstatcode{patent-eligibility}{35 U.S.C. S 101}
```

```
\def\hi@cparse@statcode#1{\KV@hi@cite{#1}}
\let\hi@cparse@regcode\hi@cparse@statcode
%
⟨/package⟩
⟨*doc⟩

\subsection{Journal Articles}

The \rtype{jrnart} reference type can parse the following input syntax:
\begin{grammar}
\meta{jrnart} := \meta{journal-text} \grammarparen{date} \\
where \meta{journal-id} := \meta{cite} \\
\Example |\defjrnart{doe}{| \\
|    John Doe, Article, 12 Law Review 345 (1950)| \\
|}|
\end{grammar}
In other words, the start of the parseable input is used for parameters
\param{author} and \param{name},
as explained for \meta{journal-text} above. The text after the last comma before
the parenthetical is passed as the \param{cite} parameter, and the parenthetical
is interpreted as \param{year}.

⟨/doc⟩
⟨*package⟩
%
%    \begin{macrocode}
\def\hi@cparse@procart{\hi@cparse@jrnart}

\def\hi@cparse@jrnart#1{%
    \let\hi@cparse@jrnart@citemacro\KV@hi@cite
    \hi@cp@paren{#1}{\KV@hi@year}{\hi@cparse@jrnart@text}%
}
```

4 Websites and Magazines

The website reference type can parse the following input syntax:

⟨*website*⟩ := ⟨*journal-text*⟩ _(⟨*date*⟩)_
where ⟨*journal-id*⟩ := ⟨*rep*⟩

Example:

```
\defmagart{doe}{
    John Doe, Latest Memes, Slate (June 5 2000)
}
```

In other words, a parseable website reference input is just like a journal article except without a volume or page number; anything following the title is passed to the parameter rep to be used as a journal title.

Magazines (magart) are identical except a page number is given after the magazine title:

⟨*journal-id*⟩ := ⟨*rep*⟩ ⟨*page*⟩

Example:

```
\defmagart{doe}{
    John Doe, Politics, Newsweek 16 (June 5 2000)
}
```

As a result, the parseable reference input for magazines does *not* look like conventional magazine citations, which typically set the date with a comma after the magazine title, rather than putting the date in a parenthetical.

```
\def\hi@cparse@website#1{%
    \let\hi@cparse@jrnart@citemacro\KV@hi@rep
    \hi@cp@paren{#1}{\KV@hi@year}{\hi@cparse@jrnart@text}%
}
\def\hi@cparse@magart#1{%
    \let\hi@cparse@jrnart@citemacro\hi@cparse@magart@cite
    \hi@cp@paren{#1}{\KV@hi@year}{\hi@cparse@jrnart@text}%
}
\def\hi@cparse@magart@cite#1{%
    \KV@hi@cite{X #1}%
    \hi@undefine\hi@kv@vol
}
```

5 Books

The book reference type can parse the following input syntax:

⟨*book*⟩ := ⟨*authors*⟩ _, ⟨*name*⟩ _(⟨*date*⟩)_

Example:

```
\defbook{blackstone}{
    William Blackstone, Commentaries on the Laws of
England (1765)
}
```

where ⟨*authors*⟩ is as defined above with respect to ⟨*journal-text*⟩ (i.e., one author or two separated with an ampersand).

```
\def\hi@cparse@book#1{%
    \hi@cp@paren{#1}{\KV@hi@year}{\hi@cparse@book@text}%
}
\def\hi@cparse@book@text#1{%
    \find@in{, }{#1}{\hi@cparse@book@author}{%
        \PackageError\hi@pkgname{%
            No commas found in book citation `#1'%
        }{You need to enter a book citation with author and title.}%
    }%
}
\def\hi@cparse@book@author#1#2{%
    \hi@cparse@jrnart@findauthor{#1}%
    \KV@hi@name{#2}%
}
```

6 Court Documents

The courtdoc reference type can parse the following input syntax:

⟨*courtdoc*⟩ := ⟨*name*⟩ __(⟨*date*⟩)_

Example:

```
\defcourtdoc{motion}{Motion for Leave (June 5 2000)}
```

```
\def\hi@cparse@courtdoc#1{%
    \hi@cp@paren{#1}{\KV@hi@year}{\KV@hi@name}%
}
```

7 Abbreviations

The abbrev reference type can parse the following input syntax:

⟨*abbrev*⟩ := ⟨*name*⟩ __(⟨*inline*⟩)_

Example:

```
\defabbrev{fda}{
    the Food and Drug Administration (the FDA)
}
```

In other words, the full text is before the parentheses and the abbreviation inside them.

```
\def\hi@cparse@abbrev#1{%
    \hi@cp@paren{#1}{\KV@hi@inline}{\KV@hi@name}%
}

\def\hi@cparse@selfcite#1{}%
```

8 The `cite` Parameter

The parameter cite accepts as input a string conforming to ⟨*vol-rep-page*⟩ described above. It sets the parameters vol, rep, page, and possibly year.