Signals (`signals.dtx`)

Charles Duan

August 16, 2024

Items in a citation string may be preceded by a signal, indicating how the writer of the citation understands the relationship of a cited reference to the writer's argument. A citation preceded with *see* indicates that the reference supports the writer's proposition; *compare* can identify warring sides of a debate; *cf.* can identify cleverly related connections or coyly disguise contradictory authorities; and *see generally* can point to background reading. Signals introduce a great deal of expressiveness into a system of legal citation, enabling writers to compactly distill a complete literature into a single citation string.

1 Data Model (List of Signals)

List of signals to search for. This is structured like an argument to `\find@try`.

```
\let\hi@pse@findlist\@empty
```

Adds a new signal. New signals are added to the front of the list, so they will be checked first. This means that longer signals (see also) should be added after shorter signals (see). A macro of `\hi@sig@`⟨*signal*⟩ is also defined, with the value of the macro being the signal type.

```
\def\AddSignal#1#2{%
    \prepend@macro\hi@pse@findlist{%
        {#1 }{\hi@pse@sigvolref@withsig{#1}}%
    }%
    \make@find@start{#1 }%
    \@namedef{hi@sig@#1}{#2}%
}
```

The empty signal must be separately defined to be type "s". It is not included with `\AddSignal` because the empty signal should not be looked for (it is selected by default when no other signal matches) and thus not added to the finding list.

```
\def\hi@sig@{s}
```

Commands to define the existing signals follow in the code. However, in the documentation this code is being used to produce the table of signals that follows in the next section. As a result, consult the code itself to see those commands.

Each signal belongs to one of four classes: support, comparison, contrary, and background. Here are all the signals available, along with their classes.

Signal	Class
see	Support
see, e.g.,	Support
see also	Support
see also, e.g.,	Support
e.g.,	Support
accord	Support
accord, e.g.,	Support
cf.	Support
compare	Comparison
and	Comparison
with	Comparison
contra	Contrary
but see	Contrary
but see, e.g.,	Contrary
but cf.	Contrary
see generally	Background

2 Input Syntax

As described in `parse.dtx`, the signal for a citation item is placed at the beginning of the input text for the citation item, in all lowercase text. As a convenience, duplicative sequential signals may be omitted. For example, the following are equivalent:

```
see ref1; ref2; see also ref3; ref4
see ref1; see ref2; see also ref3; see also ref4
```

In both cases, the *see* signal will apply to `ref1` and `ref2`, and *see also* to `ref3` and `ref4`.

This package does not deal with two aspects of citation signals. First, it does not enforce an order in which signals must be used. This provides slightly more flexibility to writers, although it is expected that most writers will follow conventional ordering of signals. Second, comparison signal citations typically include *and* before the penultimate item of each side of the comparison, for example:

Compare A, B, *and* C, *with* D, *and* F.

For implementation reasons,[1] this package does not add the *and* automatically.

[1] As described in the implementation discussion of citation formatting, each citation item is formatted by executing a group of commands for the citation. To figure out whether an *and* is needed in the middle of a comparison citation string, it would be necessary to read ahead to the next citation item in advance. While this could be done, it would be cumbersome to program, and the workaround described here is not enough of an inconvenience to make the effort seem worthwhile.

Instead, *and* is defined as a comparison signal for the writer to insert in the appropriate place in a citation string.

3 Formatting

There are four matters to be handled with respect to formatting of signals.

1. The citation must be formatted with the correct fonts.
2. The punctuation between citation items depends on the surrounding signals. Citation items are generally separated by semicolons, but with two exceptions:
    - Between citation items using comparison-class signals, commas are used.
    - In a sentence-style citation, citation items are separated with a period if the signal class changes. The signal is also capitalized.
3. If the same signal is used two or more times in a row, it is only displayed on the first use.
4. In a series of citation items with contrary-class signals, the word "but" is omitted from all but the first negative signal.

These commands provide for the formatting of signals, and are called during the execution of `\hi@draw@citation` when drawing a citation string.

This is called at the beginning of a citation string, to keep track of the last used signal.

```
\def\hi@signals@init{%
    \let\hi@signals@last\relax
}
```

This macro is called for each citation. It receives the signal text found in the input citation string, and computes the formatted signal and the inter-citation punctuation. #1 is the input signal text, and #2 is a callback that takes the punctuation and formatted signal as two arguments.

```
\def\hi@signals@set#1#2{%
    \def\hi@signals@this{#1}%
    \@test\ifx\hi@signals@last\relax\fi{%
        %
        % If this is the first signal.
        \expandafter\hi@signals@drawfirst\expandafter{\hi@signals@this}%
    }{%
        %
        % If this is not the first signal. If the signal is empty, it defaults
        % to the last signal.
        \ifx\hi@signals@this\@empty
            \let\hi@signals@this\hi@signals@last
        \fi
        \@expand{%
            \@expand{\hi@signals@drawmid}{%
                \csname hi@sig@\hi@signals@this\endcsname
            }{i}%
        }{\csname hi@sig@\hi@signals@last\endcsname}{i}%
    }{%
        %
        % We tack on, before the callback, an update to the signal state that
        % should happen right before we give control to the callback.
        %
        \let\hi@signals@last\hi@signals@this#2%
    }%
}
```

For the first signal, punctuation is empty. #1 is the signal text, and #2 is the callback.

```
\def\hi@signals@drawfirst#1#2{%
    \ifstrempty{#1}{#2{}{}}{#2{}{\hi@fn@sig{#1}\ }}%
}
```

A signal not at the start of the citation string. #1 is this signal's macro (`\hi@sig@`⟨*signal*⟩); #2 is last signal's macro; #3 is the callback.

```
\def\hi@signals@drawmid#1#2#3{%
    \@test\ifx\hi@signals@last\hi@signals@this\fi{%
        % For identical signals, no text given
        \hi@signals@punct{}% See extra args later
    }{%
        \@test\ifx#2\hi@sig@contra\fi{%
            % If the previous citation signal was negative, drop any "but" from
            % this citation signal.
            \@expand{\find@start{but }}{\hi@signals@this}{i}{%
                % One arg comes from \find@start; the rest are below
                \hi@signals@fmt
            }{\expandafter\hi@signals@fmt\expandafter{\hi@signals@this}}%
        }{%
            \expandafter\hi@signals@fmt\expandafter{\hi@signals@this}%
        }%
    }{#1}{#2}{#3}% These args get tacked on no matter what
}
\make@find@start{but }
```

This is called if the signal has changed from the last one. #1 is the text of the signal; the three arguments on the stack that remain are passed directly to `\hi@signals@punct`.

```
\def\hi@signals@fmt#1{\hi@signals@punct{\hi@fn@sig{#1}\ }}
```

Computes the punctuation between signals and runs the callback. #1 is the signal text to use, #2 the current signal's macro, #3 the last signal's macro, and #4 the callback.

```
\def\hi@signals@punct#1#2#3#4{%
    \@test\ifx#2#3\fi{%
        %
        % Same signal class. Use semicolon or comma depending on if it's
        % compare.
        %
        \@test\ifx#2\hi@sig@compare\fi{#4{,}{#1}}{#4{;}{#1}}%
    }{%
        % Different signal class. Depending on whether this is a sentence or
        % clause, choose punctuation.
        \@test\if@hi@sent\fi{%
            #4{\if@hi@dot\@re@dot@space\else.\fi\@hi@captrue}{#1}%
        }{%
            \@test\ifx#2\hi@sig@compare\fi{#4{,}{#1}}{#4{;}{#1}}%
        }%
    }%
}
```