8. Tracing Sparser's Operations

Sparser has two kinds of traces. Run-time traces that reflect certain parsing events at the moment they occur, and traces that can be run after the parser has finished in order to examine the chart or its edges.

Run-time traces are controlled by the values of globally bound symbols that act as flags. If the value of the flag is non-nil the trace is on. If the value is `nil` the trace is off. All trace output goes to the `*standard-output*` stream, which is typically the lisp listener window.

`*display-word-stream*` `symbol (defparameter)`

Controls the display of the words in the text. When this symbol is non-nil, the parser will print the words in the text as it scans them. This trace is convenient for watching the progress of the parser through the text or simply appreciating that something is happening. The default value is `T`.

`*trace-completion*` `symbol (defparameter)`

Prints a message at the "completion" hook (see §7.1). Every time the hook is called a string will be printed announcing the category of the edge just completed and whether it has an associated action. The default value is `nil`.

The traces for examining Sparser's analysis are functions that you can call. The presentations will include the numerical ids of the edges and positions so that you can access these objects easily and, for example, inspect them using your Lisp's inspection routines.

The traces below are typically used after the parser has finished its analysis; they could also be used during the analysis at a breakpoint in the user's code. They print out the data in the chart in various ways.

`display-chart-terminals ()` `function`

Prints to `*standard-output*` a sequence of the terminals of the chart (the words of the text) interleaved with the numbers of the positions between them. The output is printed as single line with is no provision for wrapping except as provided by the stream.

`display-chart-edges ()` `function`

Prints to `*standard-output*` a description of all of the edges that start at the successive positions of the chart. Each edge is on a new line. Notes if there no edges at a given position.

`display-chart-brackets` `function`

A variation on Display-chart-terminals that shows the segment boundaries of the positions that have them. The boundaries appear as square brackets ("[ ]") in front or behind the position numbers as appropriate.

`tts ()` `function`

Prints to `*standard-output*` the sequence of topmost edges, "treetops", in the chart. Each such edge is printed on a new line as "e" followed by the edge's number in the edge resource, and is accompanied by the words it dominates and the numbers of the positions that they start and stop at. If there is no edge over a word the word is printed by itself.

`the-edges ()` `function`

Prints to `*standard-output*` all the edges that have been completed in the course of the analysis, starting with the earliest.

`edge (integer)` `function`

Returns the edge object that has the argument number as its index in the edge resource array. This routine is useful for closing examining an edge that has been displayed by one of the earlier routines. If the number does not correspond to an edge, an "array index out of bounds" error occurs.

`position (integer)` `function`

Returns the position object that has the argument number as its index in the chart resource array. This routine is useful for closing examining a position that has been displayed by one of the earlier routines. If the number does not correspond to a position an "array index out of bounds" error occurs.

`chart-position (integer)` `function`

Returns the position object that has the argument number as its token index in the chart.