An Overview of the Earley algorithm

The general idea

Earley presents a parsing algorithm for CFG that is more efficient than other available parsing algorithms, because it avoids repeating computation by storing intermediate results in a table. A partial solution that was previously computed can be looked up very quickly in the table, instead of computing the same solution anew each time it is needed.

This general method—storing intermediate results in a table—is known among computer scientists as "dynamic programming".

Earley's algorithm is a top-down, breadth-first algorithm, but the same general idea can be applied to construct an efficient bottom-up parsing algorithm. Parsers that use the strategy of storing intermediate results are known among computational linguists as "active chart parsers" or just "chart parsers" (the table of intermediate results is sometimes referred to as a "chart".

The working of the algorithm:

In Earley's description, a partial parse is referred to as a "state". The list of all such states the table is called a "state list". Each state constitutes a hypothesis about a particular span (i.e. substring) of the input string. The hypothesis takes the form of a rule of the grammar, annotated to indicate how much of the hypothesis has been confirmed.[1]

For example, the state below represents the hypothesis that there is an **S** consisting of a **NP** followed by a **VP**, starting at point 0 (the beginning) of the input. The 0 represents the starting point, the 4 represents how much of the string has been processed so far, and the position of the "dot" in the right-hand-side of the rule indicates that so far (from 0 to 4) an **NP** has been found, predicting that a **VP** will be found beginning at position 4.

S ® .NP VP 0, 4

The algorithm works from left-to-right through the input by processing every state in the list of states.

When the algorithm begins, there is only one state in the state list (the "seed state"). But as processing proceeds, new states are added to the end of the state list.

The algorithm works by applying to each state one of three operators: the predictor, the completer, or the scanner. The conditions for choosing an operator are mutually exclusive, so that only one operation can apply to a given state. The function of each operator is to draw top-down consequences from the state to which the operator is

applied. The result of applying a given operation *may* result in adding one or more new states to the state list.

Processing continues to the end of the list of states, at which the algorithm terminates, and the state list is inspected for states that represent successful complete parses of the input string.

Looking for completed parses:

A state in the state list represents a complete parse of the input string if the state meets the following conditions:

a. The left-hand side symbol of the state is the initial symbol of the grammar.
b. The dot is at the end of the right-hand side of the state.
c. The two integers of the state are 0 and n, where n is the length of the input.

[1] Earley's statement of his algorithm also includes a lookahead mechanism. The benefits of this mechanism are minimal, and it is generally not used in practice. At any rate, omitting it here makes it much easier to describe and understand the working of the algorithm.