Recursive Descent Parsing Overview

This note describes the recursive decent parsing idea in an intuitive way to focus solely on the strategy.

This is an approach for an algorithm that determines, given a BNF definition, a non-terminal A and a string w of terminals and non-terminals, whether or not $A \overset{*}{\Rightarrow} w$.

The idea is, for each production for A, say $A \rightarrow BC$:

- determine if for some prefix x of w (i.e., w = xy) so that $B \overset{*}{\Rightarrow} x$, and also
- if $C \overset{*}{\Rightarrow} y$;

each of these sub-problems is of the same kind as the original, so the algorithm is recursive.

Actually “parsing” refers to somewhat more than determining whether or not the string can be derived — if it can be derived, we also intend to construct its derivation tree. Since this algorithm can be visualized as developing the derivation tree beginning at the root and repeatedly proceeding from parent to child nodes, this approach is referred to as ”recursive descent”.

There are several conceptual and technical problems to be resolved. For one thing, if for instance $x = \varepsilon$ (i.e., $B \overset{*}{\Rightarrow} \varepsilon$ — *erasing rules*), then the second sub-problem $C \overset{*}{\Rightarrow} y$ is actually $C \overset{*}{\Rightarrow} w$ which can lead to an infinite loop since it is no simpler problem than the original. Also, if the production is $A \rightarrow AC$ (i.e., B = A — *left recursion*), then the first sub-problem is the same as the original and again an infinite loop will result. We will see later how to resolve these complications.