CS 241 — Lecture 14

Bartosz Antczak *Instructor: Kevin Lanctot* *February 16, 2017*

Recall — Ambiguous Grammar

Since grammar can be ambiguous (i.e., "$9 + 3/3 = 4 or 10?$), we can have multiple parse trees for the same expression. The resulting string from a parse tree depends on how we *traverse* it. To make it unambiguous, we need to have a more formal set of production rules:

- $\alpha A \beta$ *directly derives* $\alpha\gamma\beta$ if there is a production rule $A \to \gamma$, where:
  - $A \in N$ (non-terminals), and
  - $\alpha, \beta, \gamma \in (N \cup T)$ (non-terminals, terminals, empty string)

  Informally, "directly derives" means it takes one derivation step or one application of a production rule.

- $\alpha A \beta$ *derives* $\alpha\gamma\beta$ if there is a finite sequence of productions $\alpha A \beta \to \alpha\Theta_1\beta \to \alpha\Theta_2\beta \to \cdots \to \alpha\gamma\beta$, where again:
  - $A \in N$ (non-terminals), and
  - $\alpha, \beta, \gamma \in (N \cup T)$ (non-terminals, terminals, empty string)

  It is written as $\alpha A \beta \implies {}^*\alpha\gamma\beta$

To reduce ambiguity, we will set up some standard for reading strings:

- **Associativity:** how we evaluate symbols (e.g., $6 - 3 + 4$: do we read it as $(6 - 3) + 4$ or $6 - (3 + 4)$?). We will set a standard for *left associativity*
- **Precedence:** grouping non-equivalent terminals. For instance, in arithmetic, multiplication takes precedence over addition.

14.1 Top-Down Parsing

Parsing is the approach to determining if a certain string is valid in a given grammar. In other words, given a grammar G and a word w, *find a derivation for* w.
Our goal in this section is to look at the characters in w and decide which rules derived w from the start symbol.

14.1.1 Approach 1 — Backtracking

We can use a *backtracking algorithm* for parsing in a CFG using a simple algorithm: But this approach is very exhaustive, so let's try another approach.

14.1.2 Approach 2 — Stack-based Parsing

.... Now that we know how this works, one problem still stands: how are we able to correctly predict which rule applies? No rule $\implies$ error.