Class Announcements

- Second homework has been posted
- We will have around 8 homeworks. Homeworks count for 10% of your overall course grade. **The 2 homeworks with the lowest grades will not count.** Homeworks are meant for exam preparation.
- My office hours will be Fridays, 2:20pm to 4:00pm, in CoRE 305.
Syntax Analysis (Scott 2.3 - 2.5 - skip 2.3.4)

Top-Down Parsing - LL(1)

Basic Idea:

- The parse tree is constructed from the root, expanding **non-terminal** nodes on the tree’s frontier following a left-most derivation.

- The input program is read from left to right, and input tokens are read (consumed) as the program is parsed.

- The next **non-terminal** symbol (A in above figure) is replaced by one of its rules. The particular choice **has to be unique**, and uses parts of the input (partially parsed program), for instance the first **token** (here y[1]) of the remaining input.
Top-Down Parsing - LL(1) (cont.)

How can we parse (automatically construct a left-most derivation) an input string, for example a b a b c, using a PDA (push-down automaton) and only the first symbol of the remaining input?

Example:

S ::= a S | b S | c

INPUT: a b a b c eof
Predictive Parsing

Basic idea:

For any two productions \( A ::= \alpha | \beta \) with \( \alpha \in (T \cup N)^* \) and \( \beta \in (T \cup N)^* \), we would like a distinct way of choosing the correct production to expand.

For \( \alpha \in (T \cup N)^* \), define \( \text{FIRST}(\alpha) \) as the set of tokens that appear as the first token in some string derived from \( \alpha \).

That is
\[
a \in \text{FIRST}(\alpha) \iff \alpha \Rightarrow^* a\gamma \text{ for some } \gamma \in (T \cup N)^* \text{ and } a \text{ is a token } (a \in T), \text{ and}
\]
\[
\epsilon \in \text{FIRST}(\alpha) \iff \alpha \Rightarrow^* \epsilon
\]

For a non-terminal \( A \), define \( \text{FOLLOW}(A) \) as the set
\[
a \in \text{FOLLOW}(A) \iff S \Rightarrow^* \alpha A a \gamma \text{ for some } \alpha, \gamma \in (T \cup N)^*, \ a \in T, \text{ and } S \text{ the start symbol.}
\]

Thus, a non-terminal’s FOLLOW set specifies the tokens that can legally appear after it.

FOLLOW sets are not defined for terminal symbols.

\[
\text{FIRST and FOLLOW sets can be constructed automatically}
\]
Key Property:
Whenever two productions $A ::= \alpha$ and $A ::= \beta$ both appear in the grammar, we would like

- $FIRST(\alpha) \cap FIRST(\beta) = \emptyset$, and
- if $\alpha \Rightarrow^* \epsilon$ then $FIRST(\beta) \cap FOLLOW(A) = \emptyset$
- Analogue case for $\beta \Rightarrow^* \epsilon$. Note: due to first condition, at most one of $\alpha$ or $\beta$ can derive $\epsilon$.

This would allow the parser to make a correct choice with a lookahead of only one symbol!
LL(1) Grammar

Define $FIRST^+(\delta)$ for rule $A ::= \delta$

- $FIRST(\delta) - \{\epsilon\} \cup \text{Follow}(A)$, if $\epsilon \in FIRST(\delta)$
- $FIRST(\delta)$ otherwise

A grammar is LL(1) iff

$(A ::= \alpha$ and $A ::= \beta)$ implies

$FIRST^+(\alpha) \cap FIRST^+(\beta) = \emptyset$
Next Lecture

Things to do:
Start programming in C. Check out the web for tutorials.

Next time:

- FIRST and FOLLOW set examples
- More syntax-directed translation examples
- Project 1 overview
- Programming in C, pointers, explicit memory allocation
- Read Scott 5.1 - 5.3 (some background - chapter on CD)