Lecture 8:

- Quiz 2
- Derivations (cont)
- Onto Parsing

Announcements:

- HW-1 out (due Mon)
Check In: How can we represent $S \to a a^*$ using recursion?

$S \to a \mid aS$

- sometimes denoted as $S \to a^+$

Check In: Define a grammar for strings $a^i b^j c^i$ where $i > 0$, $j \geq 0$, and $i$ is even.

$S \to a a T c c \mid a a S c c$

$T \to b T \mid \epsilon$

... or $T \to b^*$

Types of Derivations:

- **Left-most**: replace left-most non-terminal at each step
- **Right-most**: replace right-most non-terminal at each step
- **Neither left- nor right-most**: doesn’t follow either pattern

*Note*: Can help to rewrite Kleene star and alternation when learning derivations

Check In: Give a left-most derivation of $abcd$ starting from $S$ using grammar:

$S \to a T U d$

$T \to b T \mid \epsilon$

$U \to U c \mid c$

$S \Rightarrow a T U d \Rightarrow a b T U d \Rightarrow a b U d \Rightarrow a b c d$
**Parsing: An example grammar**

Simple list of assignment statements

\[
<\text{stmt}_\text{list}> ::= <\text{stmt}> \mid <\text{stmt}> ';' <\text{stmt}_\text{list}>
\]

\[
<\text{stmt}> ::= <\text{var}> '=' <\text{expr}>
\]

\[
<\text{var}> ::= 'A' \mid 'B' \mid 'C'
\]

\[
<\text{expr}> ::= <\text{var}> \mid <\text{var}> '+' <\text{var}> \mid <\text{var}> '-' <\text{var}>
\]

– Note: many possible grammars for this language!

**Recall: using grammars to generate strings (derivations)**

1. choose a rule (e.g., with start symbol on left-hand side)
2. replace with right-hand side (of rule)
3. pick a non-terminal \( N \) and rule with \( N \) on left side
4. replace \( N \) with rule’s right-hand side
5. repeat from 3 until only terminals remain

Whereas \( \rightarrow \) (or \( ::= \)) denotes a rule, \( \Rightarrow \) denotes a derivation
Example derivation of “$A = B + C; B = A$”

\[
\begin{align*}
<\text{stmt}_\text{list}> & \Rightarrow <\text{stmt}> ; <\text{stmt}_\text{list}> \\
& \Rightarrow <\text{var}> = expr ; <\text{stmt}_\text{list}> \\
& \Rightarrow A = expr ; <\text{stmt}_\text{list}> \\
& \Rightarrow A = <\text{var}> + <\text{var}> ; <\text{stmt}> \\
& \Rightarrow A = B + <\text{var}> ; <\text{stmt}_\text{list}> \\
& \Rightarrow A = B + C ; <\text{stmt}_\text{list}> \\
& \Rightarrow A = B + C ; <\text{stmt}> \\
& \Rightarrow A = B + C ; <\text{stmt}> \\
& \Rightarrow A = B + C ; <\text{var}> = \langle expr \rangle \\
& \Rightarrow A = B + C ; B = \langle expr \rangle \\
& \Rightarrow A = B + C ; B = <\text{var}> \\
& \Rightarrow A = B + C ; B = C \\
\end{align*}
\]

• This is a “left-most” derivation
  – derived the string by replacing left-most non-terminals

• The opposite is a “right-most” derivation

\[
\begin{align*}
<\text{stmt}_\text{list}> & \Rightarrow <\text{stmt}> ; <\text{stmt}_\text{list}> \\
& \Rightarrow <\text{stmt}> ; <\text{stmt}> \\
& \Rightarrow <\text{stmt}> ; <\text{var}> = \langle expr \rangle \\
& \Rightarrow <\text{stmt}> ; <\text{var}> = \langle var \rangle \\
& \Rightarrow <\text{stmt}> ; <\text{var}> = B \\
& \Rightarrow \ldots
\end{align*}
\]

• Can also have derivations that are neither left-most nor right-most
Derivations can also be written as “parse trees”

- Using the previous example derivation of “A = B + C; B = A”
Summary – Things to Know

1. Derivations

2. Types of derivations (left-most, right-most, neither)

3. Be able to give a different type of derivation given a grammar and string to derive.

4. Understand the different notation for grammars (::= and <> ) and the simple language.

5. Parse (syntax) trees and their relationships to derivations.

6. Be able to generate a parse tree from a grammar and string.