Lecture 8:

- Quiz 2
- Derivations (cont)
- Onto Parsing

Announcements:

- HW-1 out (due Mon)

Check In: How can we represent $S \rightarrow \texttt{aa}^*$ using recursion?

$S \rightarrow \texttt{a} \mid \texttt{a}S$

- sometimes denoted as $S \rightarrow \texttt{a}^+$

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

$S \rightarrow \texttt{aa}T\texttt{cc} \mid \texttt{aa}S\texttt{cc}$

$T \rightarrow \texttt{b}T \mid \epsilon$ ... or $T \rightarrow \texttt{b}^*$

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:

$S \rightarrow \texttt{a}\,T\,U\,\texttt{d}$

$T \rightarrow \texttt{b}\,T \mid \epsilon$

$U \rightarrow U\,\texttt{c} \mid \texttt{c}$

$$S \Rightarrow \texttt{a}\,T\,U\,\texttt{d} \Rightarrow \texttt{a}\,\texttt{b}\,T\,U\,\texttt{d} \Rightarrow \texttt{a}\,\texttt{b}\,U\,\texttt{d} \Rightarrow \texttt{a}\,\texttt{b}\,\texttt{c}\,\texttt{d}$$

Parsing: An example grammar

Simple list of assignment statements

```
<stmt_list> ::= <stmt> | <stmt> ';' <stmt_list>

<stmt> ::= <var> '=' <expr>

<var> ::= 'A' | 'B' | 'C'

<expr> ::= <var> | <var> '+' <var> | <var> '-' <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`”

```
<stmt_list> ⇒ <stmt> ; <stmt_list>
            ⇒ <var> = expr ; <stmt_list>
            ⇒ A = expr ; <stmt_list>
            ⇒ A = <var> + <var> ; <stmt>
            ⇒ A = B + <var> ; <stmt_list>
            ⇒ A = B + C ; <stmt_list>
            ⇒ A = B + C ; <stmt>
            ⇒ A = B + C ; <var> = <expr>
            ⇒ A = B + C ; B = <expr>
            ⇒ A = B + C ; B = <var>
            ⇒ A = B + C ; B = C
```

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

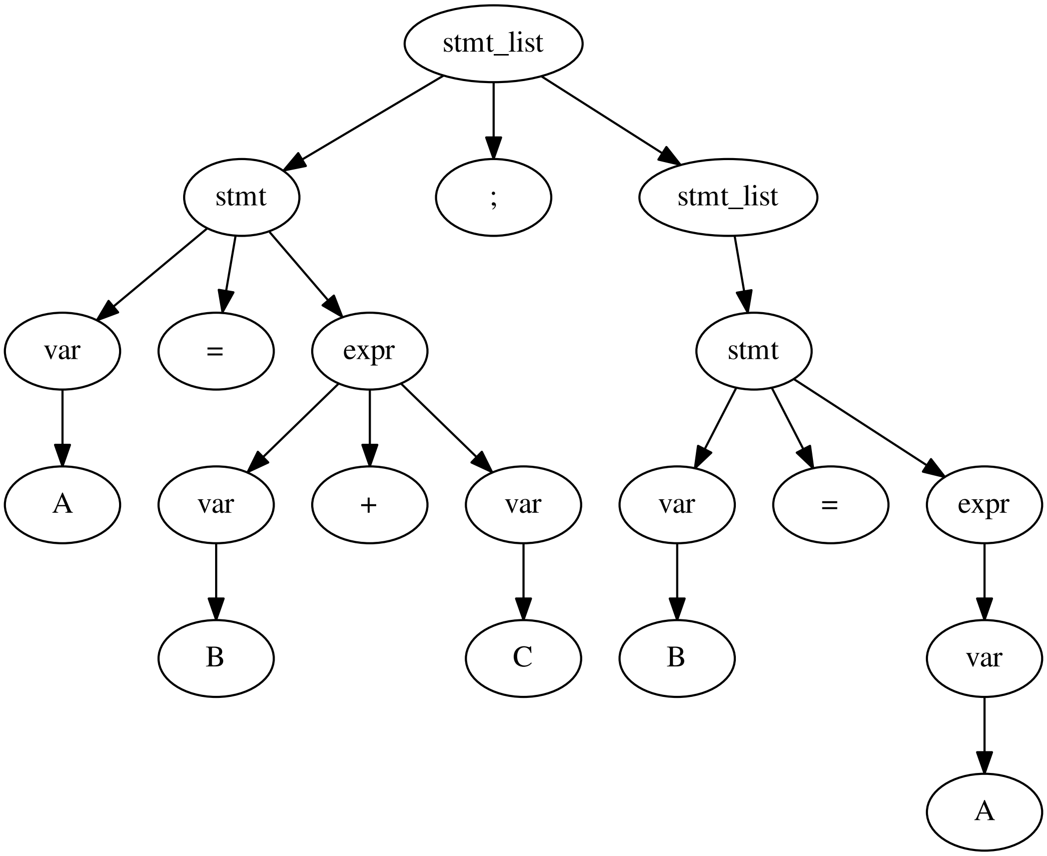
- The opposite is a “right-most” derivation

```
<stmt_list> ⇒ <stmt> ; <stmt_list>
            ⇒ <stmt> ; <stmt>
            ⇒ <stmt> ; <var> = <expr>
            ⇒ <stmt> ; <var> = <var>
            ⇒ <stmt> ; <var> = B
            ⇒ ...
```

- 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.