Lecture 9:

- Intro to Parsing

Announcements:

- HW-1 due
- HW-2 out

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!

Check In: Create a parse tree for the string (program): `"A = B"`

Parsing

- A context free grammar (derivation) is a "generator"
- Whereas a parser is a "recognizer"
  - given a token stream
  - determine if the stream is a derivation of the grammar
- A parser also (typically) builds an Abstract Syntax Tree (AST)

We'll look at $LL(k)$ parsers

- read from left-to-right, performing a left-most derivation
- parses top down (parse tree from the root down)
- at most k look ahead symbols (more later)

Consider these (modified) rules:

```
<stmt> ::= 'A' '=' <expr>
<stmt> ::= 'B' '=' <expr>
<stmt> ::= 'C' '=' <expr>
```

Assuming the parser knows <`stmt`> is to be applied ...

1. calls lexer's `nextToken`
2. checks if it is a literal `"A"`, `"B"`, or `"C"`, picking the corresponding rule
3. calls lexer's `next_token`
4. checks that it is an `ASSIGN` token
5. and so on until it finishes the <`stmt`> rule

- parser produces an error if it finds a token it isn't expecting

Tips for $LL(k)$

Watch out for left recursion!

R1: $e \rightarrow n$

R2: $e \rightarrow e + n$

Q: how far do we need to look ahead for "`5 + 4 + 3`"?

- we have to go to the end of the expression ...
- even though we're doing a left-most derivation!

1. Looking at `5` (1 lookahead), we don't know whether to apply R1 or R2
2. To decide R2, need to know if the string ends in "+ n"
3. This means we have to read the entire string to know which rule to apply
4. If the string is longer than our fixed size k, then we are stuck!
5. This means this grammar is not $LL(k)$ since has no fixed size k

One solution

$e \rightarrow n + e \mid n$

Q: How many look aheads needed? ... 2 (see "left factoring")

Can rewrite left recursion to be in $LL(k)$ ...

$e \rightarrow n\ e'$

$e' \rightarrow + n\ e' \mid \epsilon$

Q: now how far do we need to look ahead for "`5 + 4 + 3`"?

The above example involved immediate (direct) left recursion

A grammar can also have indirect left recursion

$s \rightarrow t$ a | a

$t \rightarrow s$ b | b

- allows derivations: $s \Rightarrow t$ a $\Rightarrow s$ b a
- having strings of the form: a, ba, aba, baba, ababa, ...

Example rewriting for this grammar

- By replacing RHS of t in s, we get:

  $s \rightarrow s$ b a | b a | a

Now we can rewrite the above

$s \rightarrow$ a s' | ba s'

$s' \rightarrow$ ba s' | ϵ