LL(1) Parsing

- A Push-Down Automaton (PDA) is a finite state machine augmented with a stack.
  - In addition to changing state on input, you can also look at top of stack and push/pop/chg top element of stack. NPDAs ≠ DPDAs unlike NFA = DFA.
- Can use PDAs to build parsers for CFGs
  - Can use a DPDA to parse LL(1) grammars (deterministic parser to avoid backtracking)
  - The stack contains the unprocessed part of the derivation.
- By analyzing an LL(1) grammar, we can build a Prediction Table:
  - Predict(A, x) = the set of rules we can apply when expanding an A if the next symbol is x.
  - If when building the table, we get > 1 entry in a slot, then the grammar is not LL(1).
  - If the set is $\emptyset$, then we have an error during the parse.
- Table entries get built using the First sets, and if there are $A \rightarrow \varepsilon$ rules, also the Follow sets
  - With rule $A \rightarrow \alpha$ (where $\alpha \neq \varepsilon$), if $x \in \text{First}(\alpha)$, then rule $A \rightarrow \alpha \in \text{Predict}(A, x)$.
  - Put off $A \rightarrow \varepsilon$ handling for a bit.

Example: Grammar for \{ $a^n\ c\ b^n \mid n \geq 0$ \}

- $S' \rightarrow S\ \$\$
- $S \rightarrow a\ S\ b$
- $S \rightarrow c$

[Start 11/30 changes]

- Calculate First(S): [11/30]
  - $a \in \text{First}(S)$ because of $S \rightarrow a\ S\ b$
  - $c \in \text{First}(S)$ because of $S \rightarrow c$
  - So First(S) = \{a, c\}
- Calculate First(S)’:
  - First(S’) $\supseteq$ First(S $\$$) from $S' \rightarrow S\ \$
  - First(S $\$$) = First(S) = \{a, c\}$. We don’t include $\$$ because we don’t have $S \rightarrow^* \varepsilon$.
- We don’t need the follow sets but here they are anyway:
  - Follow(S’) = $\emptyset$
  - $\$ \in \text{Follow}(S)$ from $S' \rightarrow S\ \$
  - $b \in \text{Follow}(S)$ from $S \rightarrow a\ S\ b$
  - So Follow(S) = \{b, \$.\$

[End 11/30 changes]

- Prediction table (number indicates rule to use: Predict(S’, a) = \{ rule 1 \} = \{ $S' \rightarrow S\ \$\}.}
Prediction for Grammars with ε Productions

- Say we have $S \Rightarrow^* w A x \ldots$ and a rule $S \rightarrow \varepsilon$.
  - Should we apply it and get $S \Rightarrow^* w A x \ldots \Rightarrow w x \ldots$?
  - If $w x$ doesn't begin any legal sentence in the language, then no, we shouldn't
  - When expanding $A$, if the next symbol $\in$ Follow($A$), applying $A \rightarrow \varepsilon$ is reasonable.

General rules ($x$ is next symbol)
- If $x \in$ First($\alpha$) (where $\alpha \neq \varepsilon$), then Predict($A$, $x$) includes $A \rightarrow \alpha$.
- If $x \in$ Follow($A$), then Predict($A$, $x$) includes $A \rightarrow \varepsilon$.

Example: Grammar for $\{ a^n b^n \mid n \geq 0 \}$
- $S' \rightarrow S$
- $S \rightarrow a S b$
- $S \rightarrow \varepsilon$ [not $S \rightarrow c$]

Calculate First($S'$) = {a, $}, First($S$) = {a, $\varepsilon$}, Follow($S'$) = $\emptyset$, Follow($S$) = {b, $}.

Expand
- $S'$ using $S' \rightarrow S$ $\rightarrow$ rule if next symbol is a or $ (as in $a^n c b^n$ example)
- $S$ using $S \rightarrow a S b$ rule if next symbol is a (as in $a^n c b^n$ example)
- $S$ using $S \rightarrow \varepsilon$ rule if next symbol is b or $, since Follow($S$) = {b, $}.

Predict table

<table>
<thead>
<tr>
<th>NT</th>
<th>a</th>
<th>b</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S'$</td>
<td>1</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>$S$</td>
<td>2</td>
<td>.</td>
<td>3</td>
</tr>
</tbody>
</table>

Trace parse of input using prediction table
- Stack contains terminals / nonterminals we hope to find
- As we match terminal symbols from the input, we remove them.
- As an example, take the $a^n b^n$ language (has 3 rules: (1) $S' \rightarrow S$, (2) $S \rightarrow a S b$, and (3) $S \rightarrow \varepsilon$).
Trace parse of $a \ a \ b \ b \ \$$

<table>
<thead>
<tr>
<th>Stack (Top to Left)</th>
<th>Rule # or = test</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S'$</td>
<td>1</td>
<td>$a \ a \ b \ b \ $</td>
</tr>
<tr>
<td>$S $</td>
<td>2</td>
<td>$a \ a \ b \ b \ $</td>
</tr>
<tr>
<td>$a \ S \ b $</td>
<td>$a = a$</td>
<td>$a \ a \ b \ b \ $</td>
</tr>
<tr>
<td>$S \ b $</td>
<td>2</td>
<td>$a \ b \ b \ $</td>
</tr>
<tr>
<td>$a \ S \ b \ b $</td>
<td>$a = a$</td>
<td>$a \ b \ b \ $</td>
</tr>
<tr>
<td>$S \ b \ b $</td>
<td>3</td>
<td>$b \ b $</td>
</tr>
<tr>
<td>$b \ b $</td>
<td>$b = b$</td>
<td>$b \ b $</td>
</tr>
<tr>
<td>$b $</td>
<td>$b = b$</td>
<td>$b $</td>
</tr>
<tr>
<td>$$</td>
<td>Success!</td>
<td>$$$</td>
</tr>
</tbody>
</table>

- If the top of the stack is a nonterminal, we want to apply a rule.
- If the top of the stack is a terminal, it must match the first remaining input symbol.

Review for Calculating First and Follow *(added after class)*

**First Sets**
- First(A) = First($\alpha_1$) ∪ First($\alpha_2$) ∪ ... where $A \rightarrow \alpha_1 \mid \alpha_2 \mid ...$
- First($x \ \beta$) = {$x$}
- First($B \ \beta$) = First(B) ∪ (if $B \Rightarrow^* \epsilon$ then First($\beta$) else $\emptyset$)
- First(ε) = {ε}
  - (So $\epsilon \in$ First(A) iff $A \Rightarrow^* \epsilon$)

**Follow Sets**
- For each rule $B \rightarrow ... A \ \alpha$, Follow(A) ⊇ First(α) (less ε),
  and if (α is ε or) $\alpha \Rightarrow^* \epsilon$ then Follow(B) ⊆ Follow(A).
  - (We're looking for contexts like $S \Rightarrow^* ... B \ \beta \rightarrow ... A \ \beta$, so anything that begins a $\beta$ can follow a $B$ and also follow an $A$.)

Typos in notes during lecture: I wrote $\subseteq$ where I needed $\supseteq$. It should've been:
- For $S \rightarrow a \ S \ b$, Follow(S) ⊇ First(b)
- For $S' \rightarrow S \$, Follow(S) ⊇ First($\$)