More passing
Today:

- HW due
- Next HW, cover flex (on hopper)
- Git for next HW (due Friday)
- I am gone Mon & Tues (you have class)
Last time

- More parsing:
  - removing ambiguity
  - eliminating left recursion
Back to the practical:

- Any CFG can be parsed
  \rightarrow Chomsky Normal Form
  CYK algorithm
  Run time: \( O(n^3) \)

This is too slow!

Most modern parsers look for certain restricted families of CFGs.

Result:
- LL - faster, but weaker
- LR - stronger but "slower"

Have \( O(n) \) time parsing
LL:
- left to right parsing
- left-most derivation

 Anything accepted by this type of parser is called an LL grammar.

Picture:

\[ A \rightarrow \overline{\alpha} \rightarrow \overline{\alpha} \overline{BC} \]
Top down parsing (for LLs)
Called predictive parsing.
Works well on $LL(1)$ grammars.
Table based in practice

Simple Ex: $S \rightarrow cAd$
$A \rightarrow ab/a$

Parse: cad

Rule: String w/ S, apply rules until one matches the next input (backtrack if there is a mistake)
Note: Left recursion is very bad on these!

A → A b

A / b

A / b

A / b

A

A

X never matches an input or hits a conflict.

So never forced to backtrack.
How predictive parsing works:
- the input string w is in an input buffer.
- Construct a predictive parsing table for G.
- if you can match a terminal, do it (move to next character)
- otherwise, look in table for rule to get transition that will eventually match

Hard part:
- build the table!
  (need to decide a transition if at a non-terminal based on the next input(s))

LL(k)
**FIRST & FOLLOW Sets (for LL(1))**:

FIRST (α) is any string of non-terminals

FIRST (α) = set of possible first terminals in any derivation of α by the grammar

So:

1) if x is a terminal,
   
   \[ \text{FIRST}(x) = x \]

2) if \( X \rightarrow \varepsilon \) is a production,
   add \( \varepsilon \) to \( \text{FIRST}(x) \)

3) If \( X \) is a nonterminal:
   if \( X \rightarrow Y_1 Y_2 \ldots Y_k \) is a production:
   add a if a is in \( \text{FIRST}(Y_i) \) and \( \varepsilon \) is in \( \text{FIRST}(Y_1), \ldots, \text{FIRST}(Y_{i-1}) \)
   add \( \varepsilon \) if \( \varepsilon \) is in \( \text{FIRST}(Y_k) \)
\[ \text{Ex: } \begin{align*} & E \rightarrow TE' \\
& E' \rightarrow +TE' | \varepsilon \\
& T \rightarrow FT' \\
& T' \rightarrow FT' | \varepsilon \\
& F \rightarrow (E) | \text{id} \end{align*} \]

\[
\text{FIRST} \ (E) = \{ \exists \ C, \text{id} \} \\
\text{FIRST} \ (E') = \{ \exists +, \varepsilon \} \\
\text{FIRST} \ (T) = \{ \exists C, \text{id} \} \\
\text{FIRST} \ (T') = \{ \exists *, \varepsilon \} \\
\text{FIRST} \ (F) = \{ \exists C, \text{id} \} \]
Follow Sets:
(We'll assume any input ends in \\
$\$, just to have an \end{math} character)

Rules:

1) Put $\$ in Follow(s)
 where $S$ is start symbol.

2) Given a production:
 $A \rightarrow \alpha B$

 everything in \texttt{FIRST}(B) goes
 in \texttt{Follow}(B) (except $ɛ$, if it is there).

3) Given a production:
 $A \rightarrow \alpha B$
 or $A \rightarrow \alpha BB$ with $ɛ \in \texttt{FIRST}(B)$
 then everything in \texttt{Follow}(A)
 also goes in \texttt{Follow}(B)
\[
\begin{align*}
E & \rightarrow TE' \\
E' & \rightarrow +TE' | \epsilon \\
T & \rightarrow FT' \\
T' & \rightarrow *FT' | \epsilon \\
F & \rightarrow (E) \mid \text{id}
\end{align*}
\]

We had:
\[
\begin{align*}
\text{FIRST}(E) &= \text{FIRST}(T) = \text{FIRST}(F) \\
&= \{\#, \text{id}\} \\
\text{FIRST}(E') &= \{\#, \text{id}\} \\
\text{FIRST}(T') &= \{\#, *, \epsilon\}
\end{align*}
\]

So:
\[
\begin{align*}
\text{FOLLOW}(E) &= \{\#, \$\} \\
\text{FOLLOW}(E') &= \{\#, \$\} \\
\text{FOLLOW}(T) &= \{\#, \$, \text{+}, \text{,} \} \\
\text{FOLLOW}(T') &= \{\#, \$, \text{+}, \text{,} \} \\
\text{FOLLOW}(F) &= \{\#, \$, \text{+}, *, \text{,} \}
\end{align*}
\]
Then, the Table: $M$:

For any production $X \rightarrow \alpha$, do:

1) for each terminal $a$ in \textsc{First}$(\alpha)$, add $X \rightarrow \alpha$ to $M[A,a]$.

2) If $\varepsilon$ is in \textsc{First}$(\alpha)$, add $X \rightarrow \alpha$ to $M[A,\varepsilon]$ for each terminal $b$ in \textsc{Follow}$(A)$.

If $\varepsilon$ is in \textsc{First}$(\alpha)$ and $\$ \text{ is in } \textsc{Follow}$(A)$, add $A \rightarrow \alpha$ to $M[A,\$]$. 

Any other entries are errors.

(construct on board)
### End result:

<table>
<thead>
<tr>
<th>Nonterminal</th>
<th>Inputs</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$E \rightarrow TE'$</td>
<td>$+$</td>
<td>$#$</td>
<td>$E \rightarrow TE'$</td>
<td>$E \rightarrow TE'$</td>
</tr>
<tr>
<td>$E'$</td>
<td>$E' \rightarrow +TE'$</td>
<td>$+$</td>
<td>$#$</td>
<td>$E' \rightarrow +TE'$</td>
<td>$E' \rightarrow +TE'$</td>
</tr>
<tr>
<td>$T$</td>
<td>$T \rightarrow FT'$</td>
<td>$+$</td>
<td>$#$</td>
<td>$T \rightarrow FT'$</td>
<td>$T' \rightarrow 3$</td>
</tr>
<tr>
<td>$T'$</td>
<td>$T' \rightarrow 3$</td>
<td>$+$</td>
<td>$#$</td>
<td>$T' \rightarrow 3$</td>
<td>$T' \rightarrow 3$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F \rightarrow id$</td>
<td>$+$</td>
<td>$#$</td>
<td>$F \rightarrow (E)$</td>
<td>$F \rightarrow (E)$</td>
</tr>
</tbody>
</table>
Then: Parsing!

<table>
<thead>
<tr>
<th>Stack</th>
<th>Input</th>
<th>Action</th>
<th>Matched</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E$</td>
<td>$id + d * id$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>