LL Parsing
CS 440: Programming Languages and Translators, Spring 2019
Lecture 12, Mon 3/4

Notation
• \( a, b, \ldots \in T \) (a and b stand for members of T; a, b, … are members of T) Was \( \Sigma \) for regular expressions.
• \( x, y, \ldots \in T^* \)
• \( A, B, \ldots \in V \)
• \( W, X, \ldots \in T \cup V \)
• \( \alpha, \beta, \ldots \in (T \cup V)^* \)  Sentential form

LL(k) Parser
Basic idea of LL(k) parsing
• Top-down parser with \( \leq k \) characters lookahead (\( k = 1 \) easiest case)
• Top-down: Begin with start symbol, repeatedly replace leftmost nonterminal with a rhs of one of its rules
• Represents a leftmost derivation of a terminal string

Lookahead
• Given nonterminal \( A \) to replace with rules \( A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \)
• How do we know which rhs \( \alpha \) to use? Are there other rules to use?
• Get to look at next \( k \) characters in LL(k) parsing

Rule prediction using First sets
• Look at LL(1) parsing: Contains basic ideas needed for larger \( k \).
• Define First(\( \beta \)) \( \subseteq T \) using \( \text{First}(\beta) = \{ a \in T \mid \beta \rightarrow^* a x, \text{some } x \in T^* \} \).
  (Put off for a bit how to calculate First sets.)
• To decide which \( \alpha \) to use in \( A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \)
  • Use rule \( A \rightarrow \alpha_i \) where lookahead symbol \( b \in \text{First}(\alpha_i) \)
  • Note requirement that the First sets of \( \alpha_1, \alpha_2, \text{etc.} \) must be pairwise disjoint.
• Once you know First sets, it’s easy to define a parsing table\(^1\) where
  • Table(\( A, a \)) includes rule \( A \rightarrow \alpha \) where \( a \in \text{First}(\alpha) \)
  • The input has a parsing error if no such rule exists.
  • Note if a table entry contains > 1 rule, the parse is not deterministic and the grammar is not LL.

General LL(1) parser
• Modify grammar: Add new start symbol \( S' \) and rule \( S' \rightarrow S \) where \( S \) is the nominal start symbol.

\(^1\) Parsing table is a.k.a. Prediction table.
Symbol $ used to signal end-of-input.

Basic algorithm:
- Concatenate $ to end of input string
- Push $ onto empty stack of terminals / nonterminals [$ on top, $ on bottom]
- while top of stack ≠ $ {
  - Let a be next character of remaining input
  - if $, error: input ended early
  - Pop top of stack
  - if it's a terminal b,
    - If $, remove a from input and continue loop
  - else (it's a nonterminal A)
    - Check Table(A, a) for entry A → a or Error (i.e., empty)
    - On Error, complain and go to recovery routine
    - Otherwise push a onto stack
  }
  // (Loop has ended, top of stack was $)
  - if next input character = $
    - We parsed successfully
  - else Error: we have leftover input

Calculating First Sets
- We want to calculate First(α) for each rule A → α so that when expanding nonterminal A during a parse, we can figure out which A → ⋯ rule to use. (Rule A → α goes into table entry Table(A, b) where b ∈ First(A).)
- Turns into more general problem of First(β) for various β.
- Notation: First(A → α) is First(α) [we're just mentioning which of various possible uses of α we're interested in].
- Note: First(α) can have > 1 element
  - Consider A → B where B → a | b. First(B → a) = {a} and First(B → b) = {b}, but First(A → B) = {a, b}
- Worse, there's First(A → ε). ("ε-rule")
  - We'll put this off for a bit.
- Version 1: Definition of First(α), assuming no rules of form A′ → ε:
  - First(α) = {a ∈ T | α →* a β, some β ∈ (T | V)*}
  - Base case: First(A → a β) = {a}
  - Recursive case: First(A → B β) ⊇ First(B → β) for all rules B → β (rules with B on lhs).
- We have a couple of worries if ε-rules (A → ε) are allowed.
- The easier one to deal with is First(A β) where A → ε.
  - In this case, define First(A β) ⊇ First(A) − {ε} ∪ First(β)
- The trickier one is First(A) where A → ε
  - In this case, we need to know what kinds of symbols can follow any use of A.
  - E.g., say we have A → ε, and B → αA a β.
• When parsing, if the stack is $A \gamma$ (A at top), and the next input symbol is $a$, then applying the rule $A \rightarrow \varepsilon$ is reasonable. (Presumably, we're in the middle of following the $B \rightarrow \alpha A a \beta$ rule.)

• **Version 2** : Definition of $\text{First}(\alpha)$ where grammar can include $\varepsilon$-rules (rules of the form $A' \rightarrow \varepsilon$).

  • **Note**: The definition of $\text{First}$ will be mutually recursive with the definition of $\text{Follow}$.

  • **Define** $\text{First}(\alpha) = \{ \alpha \rightarrow^* a \beta, \text{some } \beta \in (T \cup V)^* \} \cup \{ \varepsilon \text{ if } \alpha \rightarrow^* \varepsilon \text{ is possible} \}$

  • **Base case**: $\text{First}(A \rightarrow a \beta) = \{ a \}$. (This is the same as in version 1.)

  • **Recursive case 1**: $\text{First}(A \rightarrow B \delta) \supseteq \text{First}(B \rightarrow \beta) - \{ \varepsilon \}$ for all rules $B \rightarrow \beta$ (with $B$ on lhs). (This is like version 1 except for dropping $\varepsilon$ from $\text{First}(B \rightarrow \beta)$.)

  • **Recursive case 2**: For $A \rightarrow B \delta$ where $A \rightarrow B \delta$ is a rule and $B \rightarrow^* \varepsilon$ can occur, $\text{First}(A \rightarrow B \delta)$ also includes $\text{Follow}(B)$. This case is new for version 2: If $A \rightarrow B \delta \rightarrow^* \varepsilon \delta = \delta$, then if we want to apply the rule $A \rightarrow B \delta$, if the next character is a first character for $\delta$, we can continue (and eventually turn $B \delta$ into $\delta$).

  • **Define** $\text{Follow}(A)$ to be the set of terminal symbols that can directly follow $A$ in a derivation (not necessarily a leftmost derivation).

    • $\text{Follow}(A) = \{ a \in T | S' \rightarrow^* \alpha A \beta \rightarrow^* \alpha A a \beta' \text{ for some } \alpha, \beta, \beta' \in (T \cup V)^* \}$

    • **Base case**: For each rule $B \rightarrow \alpha A a \beta$ (some $B$ etc.) we get $a \in \text{Follow}(A)$.

    • **Recursive case 1**: For each rule $B \rightarrow \alpha A \beta$, we get $\text{First} (\beta) - \{ \varepsilon \} \subseteq \text{Follow} (A)$ [fixed 3/5]

    • **Recursive case 2**: For the same $B \rightarrow \alpha A \beta$, if $\beta \rightarrow^* \varepsilon$ is possible (i.e., if $\varepsilon \in \text{First}(\beta)$), then $\text{Follow}(B) \subseteq \text{Follow}(A)$ [fixed 3/5]

**Parsing table when $\varepsilon$-rules exist**

• There are now two kinds of entries in the parsing table

  • $\text{Table}(A, a)$ includes rule $A \rightarrow \alpha$ when $a \in \text{First}(\alpha)$ and $\alpha$ is not $\varepsilon$.

  • $\text{Table}(A, a)$ includes rule $A \rightarrow \varepsilon$ when $a \in \text{Follow}(A)$.

• Again, if any table entry contains $>1$ rule, the grammar is not $LL(1)$.

• And if a table entry is empty, then that nonterminal / terminal character causes a parse error

**Sample Calculation (from the textbook)**

**Grammar**

• $S \rightarrow a A B b$

• $A \rightarrow c | \varepsilon$

• $B \rightarrow d | \varepsilon$

**First sets**

• First $S \supseteq$ First $a A B b = \{ a \}$

• First $A \supseteq$ First $c \cup$ First $\varepsilon = \{ c, \varepsilon \}$

• First $B \supseteq$ First$(d) \cup$ First$(\varepsilon) = \{ d, \varepsilon \}$
• These are the only calculations to do, so the $\supseteq$ above can be turned into equality.

**Follow sets**

• Follow($B$): From $S \rightarrow a\ A\ B\ b$, the $B\ b$ part tells us
  • Follow($B$) $\supseteq \{b\}$

• Follow($A$): $S \rightarrow a\ A\ B\ b$, the $A\ B$ tells us
  • Follow($A$) $\supseteq$ First($B\ b$) $\cup$ First($b$) = $\{d,\ e\} \cup \{b\} = \{b,\ d\}$

• These are the only calculations to do, so the $\supseteq$ above can be turned into equality

**Table elements**

nonterminal $S$, terminal $a : S \rightarrow a\ A\ B\ b$ (terminal $b,\ c,\ d$ are errors)
nonterminal $A$, terminal $c : A \rightarrow c$
nonterminal $A$, $b$ or $d : A \rightarrow e$
  
So $A$ with $a$ is an error
nonterminal $B$, terminal $d : B \rightarrow d$
nonterminal $B$, terminal $b : B \rightarrow e$
  
$B$ with $a,\ c$: error

**Parsing table for grammar above**

<table>
<thead>
<tr>
<th>NT</th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>$S \rightarrow a\ A\ B\ b$</td>
<td>$error$</td>
<td>$error$</td>
<td>$error$</td>
</tr>
<tr>
<td>$A$</td>
<td>$error$</td>
<td>$A \rightarrow e$</td>
<td>$A \rightarrow c$</td>
<td>$A \rightarrow e$</td>
</tr>
<tr>
<td>$B$</td>
<td>$error$</td>
<td>$B \rightarrow e$</td>
<td>$error$</td>
<td>$B \rightarrow d$</td>
</tr>
</tbody>
</table>