**CS 164 Programming Languages and Compilers** **Spring 2019**

# Discussion 2/18: LL Parsing and Earley's Algorithm

Instructor: Paul N. Hilfinger GSIs: Nikhil Athreya, Vivant Sakore

1. **LL Parsing Ambiguities.** An LL(k) grammar is a CFG used by a parser that scans input left-to-right ("L"), leftmost derivation ("L"), and uses k tokens of lookahead to predict the correct production. We've previously seen that a grammar is ambiguous if it has a parse tree that is not unique. A more formal definition of LL conflicts uses FIRST and FOLLOW sets.

   - **FIRST(A)**: the set of all terminals that could occur first in an expansion of the terminal or nonterminal A (include $\epsilon$ if A can expand to $\epsilon$)
   - **FOLLOW(A)**: the set of all terminals that could follow an occurrence of the terminal or nonterminal A in a (partial) derivation

   There are two main types of LL(1) conflicts:

   - **FIRST/FIRST**: The FIRST sets of two different productions for same non-terminal intersect.
   - **FIRST/FOLLOW**: The FIRST set of a grammar rule contains an epsilon and the intersection with its FOLLOW set is not empty.

   Is the following grammar LL(1)? Justify your answer using FIRST and FOLLOW sets.

   - $S \to Xd \qquad X \to C \mid Ba \qquad C \to \epsilon \qquad B \to d$

2. **Resolving Conflicts.**

   Consider the following grammar for numerical expressions with division, addition, and unary minus:

   - $E \to Num \mid E/E \mid E + E \mid -E$

   (a) Rewrite the grammar so that it is LL(1), so that '/' has higher precedence than '+', and so that '-' has highest precedence. '+' and '/' should be parsed in a right-associative way.

   (b) Compute the FIRST and FOLLOW sets for your re-written LL(1) grammar.

(c) Draw the LL(1) parsing table for the grammar. You may need the following rules:

- For each production $X \to A_1 ... A_n$:
  - For each $1 \leq i \leq n$, and for each $b$ in **First**$(A_i)$: Set $T[X, b] = X \to A_1 ... A_n$. Stop when $\epsilon$ is not in **First**$(A_i)$.
  - If $A_1 ... A_n \to^* \epsilon$, then for each $b$ in **Follow**$(X)$: Set $T[X, b] = \epsilon$.

3. **Earley's Algorithm.**

- Consider the following grammar:

$$P \to E \dashv$$
$$E \to E + E$$
$$E \to E * E$$
$$E \to ID$$

Use Earley's algorithm to parse `ID + ID * ID` by filling out the chart below (you may not need all of the rows).

| | `ID` | | `+` | | `ID` | | `*` | | `ID` | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| a | | | | | | |
| b | | | | | | |
| c | | | | | | |
| d | | | | | | |
| e | | | | | | |
| f | | | | | | |
| g | | | | | | |
| h | | | | | | |
| i | | | | | | |
| j | | | | | | |
| k | | | | | | |
| l | | | | | | |
| m | | | | | | |

- Draw the accepting parse tree(s) and identify whether there are ambiguities in the grammar.