CS 164 Programming Language and Compilers **Fall 2015**

Section 4: LL(1) tables, LR/Bottom-Up Parsing

GSI: Melanie Cebula, Ben Mehne, Matt Miller

- **Announcement**
  - Midterm Mon (10/05)
  - WA3 due next Wed (9/30) in class
  - PA3 due Tues (10/06) at 11:59pm
  - PA4/WA4 out Wed (9/30)
- **Outline**
  1. LL(1) tables
  2. LR parsing

---

1 LL(1) Parsing Tables

Algorithm 1 Construct a parsing table T for CFG G

```
1: procedure ConstructTable(G, T)
2:     for production A → α in G do
3:         for terminal b ∈ FIRST(α) do
4:             T[A, b] = α
5:         end for
6:         if α → *ϵ for each b ∈ FOLLOW(A): T[A, b] = α
7:     end for
8: end procedure
```

Example 1. Consider the following grammar: $E \rightarrow Num \mid E/E \mid E + E \mid -E$
Here is the grammar rewritten to be LL(1):

$$E \rightarrow TE'$$
$$E' \rightarrow +E \mid \epsilon$$
$$T \rightarrow ST'$$
$$T' \rightarrow /T \mid \epsilon$$
$$S \rightarrow -S \mid Num$$

1. Compute the FIRST and FOLLOW sets for the re-written LL(1) grammar. You may need following definitions:
   - **First(A)**: the set of all terminals that could occur first in an expansion of the terminal or nonterminal A (include ϵ if A can expand to ϵ)
   - **Follow(A)**: the set of all terminals that could follow an occurrence of the terminal or nonterminal A in a (partial) derivation. Make sure to include $ if EOF can follow the terminal.

   Answer:

2. Draw the LL(1) parsing table for the grammar using the provided algorithm.
   Answer:

2 LR Parsing

Idea : LR parsing keeps reduces a string to the start symbol by inverting productions.

1. Consider the following unambiguous grammar that enforces left associativity and correct precedence:

$$E' \to E$$
$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to NUM$$

2. Explain why the gammar is not LL(1).
   Answer:

3. Write down the LR(1) item sets for the above grammar.
   (Recall that an LR(1) item is of the form $X : b \bullet c, a$, which says that we are trying to find X by production bc such that X is followed by nonterminal a and that we have accumulated b on the parsing stack)
   Answer:

4. Draw the parsing DFA. Each state in the DFA consists of an item set, "shift" transitions from one state to another, and "reduce" rules from that state using production rules (i.e. Reduce using production rule 2 on $ or +).

   Production numbers for the DFA:

   1. $E \to E + T$
   2. $E \to T$
   3. $T \to T * F$
   4. $T \to F$
   5. $F \to NUM$

   Answer:

5. Use the DFA to conclude, i.e. prove, that the grammar in is indeed unambiguous. How can you find ambiguity in the DFA?
   Answer: