

## CS 6353 Compiler Construction, Homework #2

1. Construct a type-0/1 grammar for the language  $\delta c \delta$ , where  $\delta$  can be any string of a's and b's. Use some examples to illustrate how your grammar would work.

2. Consider the following grammar. Note that `id`, `+`, `[`, `]`, and `“,”` are terminals.

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow \text{id} \mid \text{id}[] \mid \text{id}[X] \\ X &\rightarrow E, E \mid E \end{aligned}$$

- (a) Eliminate left recursion in the grammar.
- (b) Perform left factoring for the grammar.
- (c) Compute the First set for all symbols in the grammar.
- (d) Compute the Follow set for all non-terminals in the grammar.
- (e) Build an LL(1) parse table for the grammar.
- (f) Parse the string `id + id[id+id, id[]]`. Show the stack, the input, and the action taken.
- (g) Build the parse tree while you are parsing. Show your parse tree.

3. Consider the following grammar.

$$\begin{aligned} S &\rightarrow As \\ A &\rightarrow BCA \\ A &\rightarrow BCa \\ B &\rightarrow b \\ C &\rightarrow c \end{aligned}$$

- (a) Show that the grammar is not LL(1).
- (b) Is the grammar LL(k)? If so, give the k value and show the parsing table.

4. Consider the following grammar. Note that `id`, `(`, and `)` are terminals.

$$\begin{aligned} S &\rightarrow TS \mid \varepsilon \\ T &\rightarrow ( S ) \mid [ F ] \\ F &\rightarrow \text{id} F \mid \text{id} \end{aligned}$$

- (a) Construct the LR(0) Automata for the grammar.
- (b) Construct the SLR(1) parsing table for the grammar.
- (c) Parse the string `([id id id] ()) [id id]`. Show the stack, the input, and the actions taken.
- (d) This grammar is also LL, but not LL(1). Sketch a scheme that can process the grammar using LL. Do not use LL(2). Do not rewrite the grammar. Do it in LL(1) but with some add on rules.
- (e) Discuss the space requirement for SLR(1) and LL(1), including the parsing table size and the maximum parsing stack size for any input string.

5. Consider the following grammar. Note that `SS` and `PP` are each a single non-terminal.

$$\begin{aligned} \text{Prog} &\rightarrow \text{SS} \\ \text{SS} &\rightarrow S \text{SS} \mid \varepsilon \\ S &\rightarrow \{ = \text{id} P \} \\ S &\rightarrow \{ \text{loop id SS} \} \\ P &\rightarrow F \mid \text{id} \mid \text{num} \end{aligned}$$

$$F \rightarrow \{ fn PP \}$$
$$PP \rightarrow P PP \mid P$$

- (a) Show that the grammar is not SLR(1).
- (b) Construct the LR(1) Automata for the grammar.

6. Consider the following grammar.

$$S \rightarrow A-B)$$
$$A \rightarrow B \mid a)$$
$$B \rightarrow aB \mid a$$

- (a) Construct the LR(1) Automata for the grammar.
- (b) Merge the LR(1) states and convert the LR(1) automata to LALR(1).

7. Give example grammars for the following specifications.

- (a) A grammar that is not SLR(5) but is SLR(6).
- (b) A grammar that is LR(1), not LALR(1), and is LALR(2).

8. Assume that  $G$  is a context free grammar and  $G$  is also LL( $n$ ) and LR( $m$ ). Is it possible for  $n < m$ ? If so, show an example grammar that meets the criteria. If not, explain clearly why not. **Note that  $G$  is not LL( $n-1$ ) and is not LR( $m-1$ ).**