

CS 352 – Compilers: Principles and Practice – Homework 1
Due: in class Wednesday, October 26, 2011

1. Consider the following regular expression:

$$((x|y|z)^*(a|b)^+)|xyzzy$$

- (a) Give an NFA corresponding to this regular expression.
- (b) Devise (without going through the full subset construction!) a DFA for this regular expression.

2. Consider the following regular expression and the regular language it denotes:

$$a^*bc|ab^*(b|c)$$

- (a) As described in class, construct an NFA that recognizes this regular language.
- (b) Using the subset construction, convert the NFA to a DFA. Optimize the resulting DFA by merging equivalent states (if any).

3. Consider the following simple grammar:

$$\begin{array}{l} S \rightarrow L \quad L \rightarrow Ra \quad R \rightarrow aba \quad Q \rightarrow bbc \\ L \rightarrow Qba \quad R \rightarrow caba \quad Q \rightarrow bc \\ R \rightarrow Rbc \end{array}$$

- (a) This grammar is not suitable for a top-down predictive (i.e., LL) parser. What problems does it have?
- (b) Fix the problems by rewriting the grammar.
- (c) Construct the LL(1) parsing table for your new grammar.
- (d) Consider the following input string: *cababca*. Show the steps of an LL parser as it uses your parsing table to predict the expansion of each non-terminal for this input, showing the input as it is consumed, and the parse stack at each step of the parse.

4. Consider the following simple (augmented) grammar:

$$\begin{array}{l} S \rightarrow L\$ \\ L \rightarrow E;L \\ L \rightarrow E \\ E \rightarrow \text{nil} \end{array}$$

- (a) Show the steps of a bottom-up parser as it parses the input *nil;nil;nil*, showing the input as it is consumed, the parse stack at each step of the parse, and the action applied at each step.
- (b) Construct the LR(0) item sets for this grammar:
- (c) Is this grammar LR(0)? Why or why not?
- (d) Construct the SLR(1) parse table for this grammar.
- (e) Is this grammar SLR(1)? Why or why not?