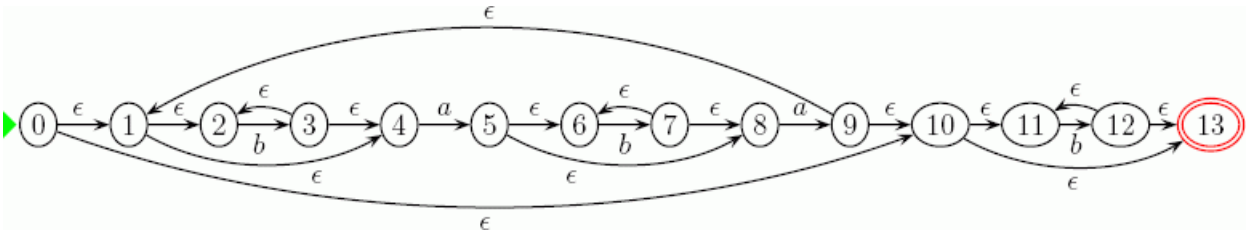
Computer Science Comprehensive Exam—Fall 2012
Compiler Construction

Instructions: Do *not* put your name on the exam, please answer all the questions directly on the exam itself. You may write on the back of the pages. You may need scratch paper to work out the answers before writing them on the exam. Answer **all** the questions. You have 90 minutes. Explain answers as fully as possible, give examples or define terms, if appropriate.

1. An important part of a compiler is type-checking. What is type-checking? How does it relate to the other parts of a compiler? How is type-checking done?

2. A compiler's scanner or lexer is very similar to a table-driven finite state machine. But there are differences. What are the differences between a finite state machine and a scanner? What are the differences in output? What are the diffences in implementation?

3. Convert the following NFA over the Σ = {*a*, *b*} to a DFA using the subset construction. The start state of the NFA, marked by a triangle, is 0; the only final state, marked by double lines, is 13. Your result should have five states. Label your states with capital letters *A*, *B*, *C*, *D*, and *E* and fill in the table below so that the correspondence is clear between the states of your DFA and *sets* of the NFA's state labels. Fill in the table with the transition on each state of your DFA. Do not simplify.



DFA	*NFA*	*a*	*b*
A			
B			
C			
D			
E			

4. Consider the following grammar:

$$
\begin{aligned}
S &\rightarrow u\,B\,D\,z \\
B &\rightarrow B\,v \\
B &\rightarrow w \\
D &\rightarrow E\,F \\
E &\rightarrow y \\
E &\rightarrow \varepsilon \\
F &\rightarrow x \\
F &\rightarrow \varepsilon
\end{aligned}
$$

(a) Compute nullable, FIRST and FOLLOW for all nonterminals.

	nullable	FIRST	FOLLOW
S			
B			
D			
E			
F			

(b) Compute the FIRST of the right-hand side of all productions.

	α	FIRST(α)
1	$S \rightarrow u\,B\,D\,z$	
2	$B \rightarrow B\,v$	
3	$B \rightarrow w$	
4	$D \rightarrow E\,F$	
5	$E \rightarrow y$	
6	$E \rightarrow \varepsilon$	
7	$F \rightarrow x$	
8	$F \rightarrow \varepsilon$	

(c) Fill in the 11(1) parse table for the grammar. Explain clearly why the grammar is *not* 11(1).

	w	*u*	*v*	*x*	*y*	*z*
S						
B						
D						
E						
F						

5. Consider the algorithm to compute CLOSE[I] for the set I of LR(1) items for some grammar. Suppose the grammar contains the production $X \rightarrow \gamma$ where X is some non-terminal and γ is some string of terminals and non-terminals. Answer the following questions assuming A is some non-terminal, α and β are strings of terminals and non-terminals, and y and z are terminal symbols.

   (a) If $A \rightarrow \alpha \bullet X,\ z$ is in I, which item or items (if any) would be added to CLOSE[I]?

   (b) If $A \rightarrow \alpha \bullet X\, y,\ z$ is in I, which item or items (if any) would be added to CLOSE[I]?

   (c) If $A \rightarrow \alpha \bullet X\, \beta,\ z$ is in I, which item or items (if any) would be added to CLOSE[I]?

6. For the following augmented grammar:

$$
\begin{array}{lrcl}
0 & S' & \rightarrow & S\,\$ \\
1 & S & \rightarrow & a\,A \\
2 & S & \rightarrow & b\,B \\
3 & A & \rightarrow & C\,a \\
4 & A & \rightarrow & D\,b \\
5 & B & \rightarrow & C\,b \\
6 & B & \rightarrow & D\,a \\
7 & C & \rightarrow & E \\
8 & D & \rightarrow & E \\
9 & E & \rightarrow &
\end{array}
$$

(a) Give a diagram of the LR(1) states and transitions.

(b) Give the LR(1) parsing tables.

(c) Is the grammar LR(1)? Explain.

(d) Is the grammar LALR(1)? Explain.