ECE 573 — Midterm 1

September 29, 2009

Name: _____

Purdue email: _____

Please sign the following:
I affirm that the answers given on this test are mine and mine alone. I did not receive help from any person or material (other than those explicitly allowed).

X _____

Part	Points	Score
1	8	
2	15	
3	10	
4	22	
5	25	
6	20	
Total	100	

Part 1: Short answers (8 points)

1) Place the following parts of a compiler in order and identify which are part of the front-end, and which are part of the back-end. *Parser, Code generator, Scanner, Optimizer (of IR), Semantic routines* (2 points)

2) Name two (of four) different types of compilers, and provide a real-world example for each that you name (2 points)

3) Explain (briefly—you shouldn't need more than one sentence) the difference between the *syntax* of a language and its *semantics* (1 point)

4) Explain the difference between a context-free grammar and a context-sensitive grammar (1 point)

5) Why do semantic routines need to be placed at the end of productions in LR grammars? (2 points)

Part 2: Regular expressions, finite automata and scanners (15 points)

1) Describe, in one sentence, the strings captured by the following regular expression (2 points):

$$(ab)^+c^*$$

2) Give a finite automaton that accepts languages defined by the following regular expression (this should be a non-deterministic FA) (6 points):

$$((xy^*z)|(x^*yz))$$

3) Give the deterministic equivalent of the NFA you created in problem 2 (7 points)

Part 3: Grammars (10 points)

Let G be the grammar:

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow xB \mid \lambda \\ B &\rightarrow yA \mid zB \end{aligned}$$

Using this grammar, answer the following questions.

1) What are the terminals and non-terminals of this grammar? (1 point)

2) Give 4 examples of strings in the language defined by this grammar. (2 points)

3) Draw the parse tree for the following partial derivation (*i.e.*, some of the leaves of your parse tree may be non-terminals) (4 points)

$$S \Rightarrow xyyA$$

4) Did this partial derivation get produced by left-derivation or right-derivation? (1 points)

5) Give an example of a partial derivation produced from S in 2 steps by right derivation (2 points)

Part 4: LL parsers (22 points)

Answer the questions in this part using the following grammar:

$$
\begin{array}{lcl}
S & \rightarrow & Ab\$ \\
A & \rightarrow & (bAb) \\
A & \rightarrow & (Ab) \\
A & \rightarrow & \lambda
\end{array}
$$

1) Define the following sets: (8 points)

First(A b *)*	
First ((b a b) *)*	
First ((A b) *)*	
Follow (A *)*	

2) Give the predict sets for the productions: (8 points)

Predict(1)	
Predict(2)	
Predict(3)	
Predict(4)	

3) Fill in the LL(1) parse table based on your predict sets (4 points)

	(	)	b	$
S				
A				

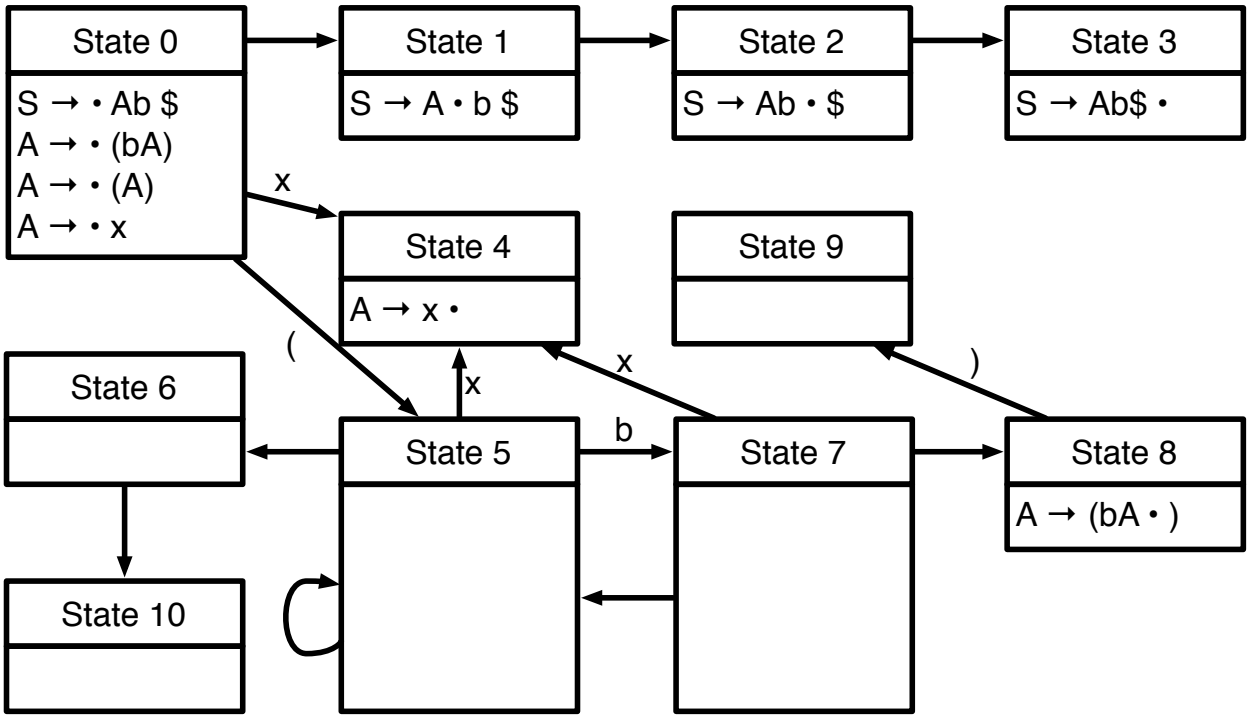
4) Is this an LL(1) grammar? Why or why not? (2 points)

Part 5: LR(0) Parsers (25 points)

Use the following grammar for the next two questions:

$$
\begin{aligned}
S &\rightarrow Ab\$ \\
A &\rightarrow (bA) \\
A &\rightarrow (A) \\
A &\rightarrow x
\end{aligned}
$$

1) Fill in the missing states for the for the following CFSM (12 points) and fill in the missing edge labels (1 point)



2) List the actions the parser will take when parsing the following string. For shift actions, indicate which state the parser will go to; for reduce actions, indicate which rule is being reduced and which state the parser will go to after reducing. (You do not have to show the parse stack or the remaining input—though it may help. Assume the parser accepts when it gets to state 2) (12 points)

(b (x)) b $

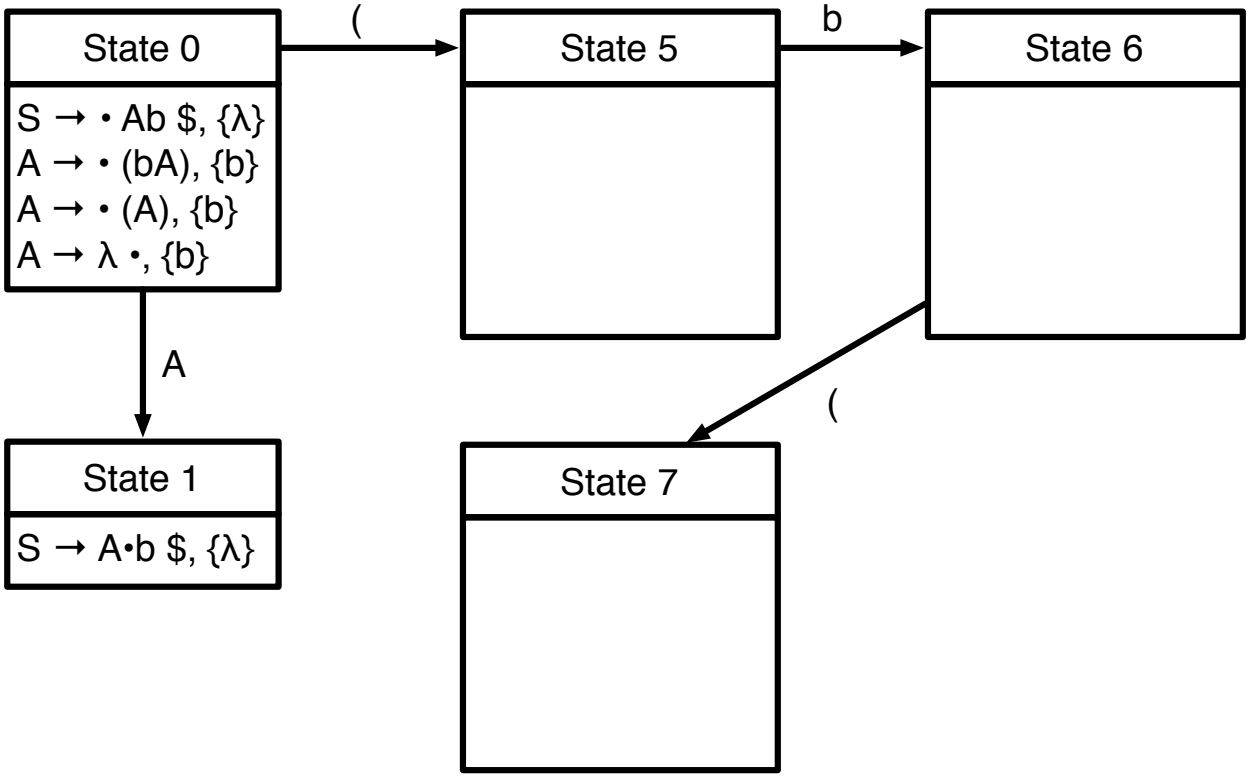
Part 6: LR(1) Parsers (20 points)

Consider the grammar:

$$
\begin{aligned}
S &\rightarrow Ab\$ \\
A &\rightarrow (bA) \\
A &\rightarrow (A) \\
A &\rightarrow \lambda
\end{aligned}
$$

1) Is this grammar LR(0)? Why or why not? (1 points)

2) Fill in the missing states in the partial LR(1) machine given below (13 points)



3) Fill in the action and goto tables for State 0 (4 points)

Action table			
b	$	(	)

Goto table					
b	$	(	)	S	A

4) How would an SLR parser differ from this LR(1) parser? (1 point) An LALR parser? (1 point)