ECE 468/573 — Midterm 1

October 1, 2014

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____

Note: ECE 468 students *do not* have to complete Part 6.

Part	Points	Score
1	10	
2	18	
3	14	
4	24	
5	34	
6	15	
Total	115	

Part 1: Short answers (10 points)

1) In recent years, many companies have switched from GCC-based compiler toolchains to LLVM-based compiler toolchains even though LLVM re-used GCC's front ends. Given that, what reason might a company have had for switching to LLVM? (5 points)

LLVM may have a better optimizing back end — even if the same front-end is used, LLVM can do a better job of producing optimized code

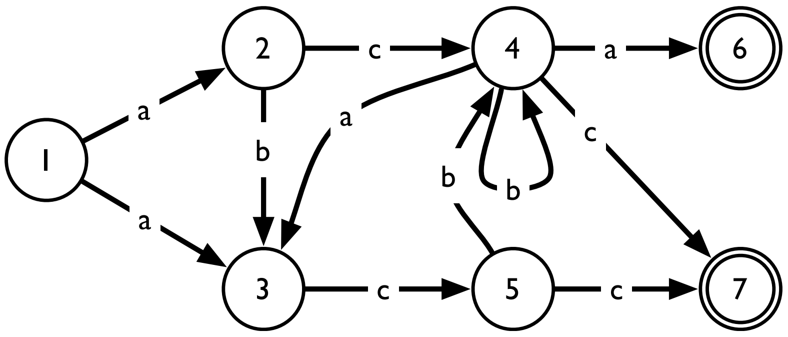
Some people thought that LLVM was a true virtual machine with a JIT compiler; I gave credit for reasonable answers along those lines.

2) One nice thing about maintaining a common ISA is that when a new processor is released, you don't have to recompile any code, or change the compiler. Nevertheless, when Intel releases new x86 processors, companies update their compilers anyway. Why might they want to do that? (5 points)

Even if the ISA is the same, the implementation of that ISA might be different—some instructions may be more efficient—so it can be important to change the backend to account for these differences. Also, newer x86 processors may support instruction set extensions (SIMD, x86-64) and compiler backends need to be changed to generate code targeting those extensions.

Part 2: Regular expressions, finite automata and scanners (18 points)

1) **Consider the following NFA. Fill in the transition table below with its corresponding DFA using the subset construction. You may not use all of the rows (16 points):**



State	Final?	a	b	c
1	No	2, 3	X	X
2, 3	No	X	3	4, 5
3	No	X	X	5
4, 5	No	3, 6	4	7
5	No	X	4	7
3, 6	Yes	X	X	5
4	No	3, 6	4	7
7	Yes	X	X	X
2 pts/row				

2) List which states (if any) should be merged when you reduce the DFA you just derived (2 points):

{4, 5} and 4
Note that 3 and {3, 6} can't be merged because one is final and the other isn't.

Part 3: Grammars (14 points)

Let G be the grammar:

$$S \rightarrow A\$$$
$$A \rightarrow (AB)$$
$$A \rightarrow \lambda$$
$$B \rightarrow (A)$$
$$B \rightarrow x$$

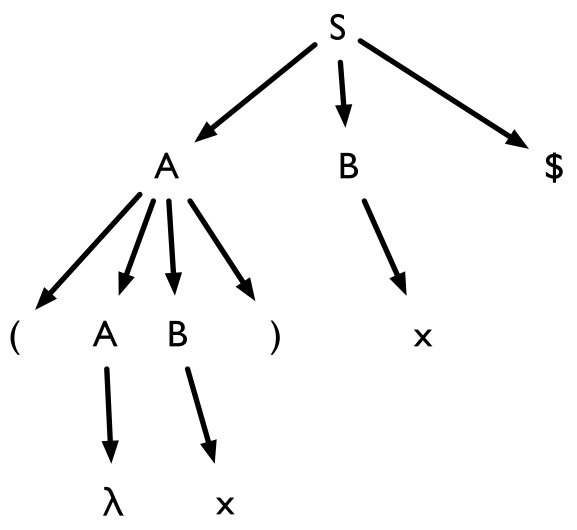
Using this grammar, answer the following questions.

1) What are the terminals and non-terminals of this grammar? (6 points)

Terminals: (,), x, $
Non-terminals: S, A, B

1 point per mistake (note that λ is not a terminal!)

2) Draw the parse tree for the string "((x)x)$" (8 points)



Part 4: LL parsers (24 points)

Answer the questions in this part using the same grammar from Part 3.

1) Give the First sets for each non-terminal in the grammar (6 points)

First(S) : {(, $} [3 points]
First(A) : {(, λ} [2 points]
First(B) : {(, x} [1 point]

2) Give the Follow sets for each non-terminal in the grammar (6 points)

Follow(S) : { } [1 point]
Follow(A) : {x, (, $,)} [4 points]
Follow(B) : {) } [1 point

3) Give the Predict sets for each *production* in the grammar (5 points)

Predict(1) : {(, $}
Predict(2) : { (}
Predict(3) : { x, (, $,) }
Predict(4) : { (}
Predict(5) : { x }

4) Fill in the following parse table (6 points)

	(	)	x	$
S	1			1
A	2, 3	3	3	3
B	4		5	

5) Is the grammar LL(1)? Why or why not? (1 points)

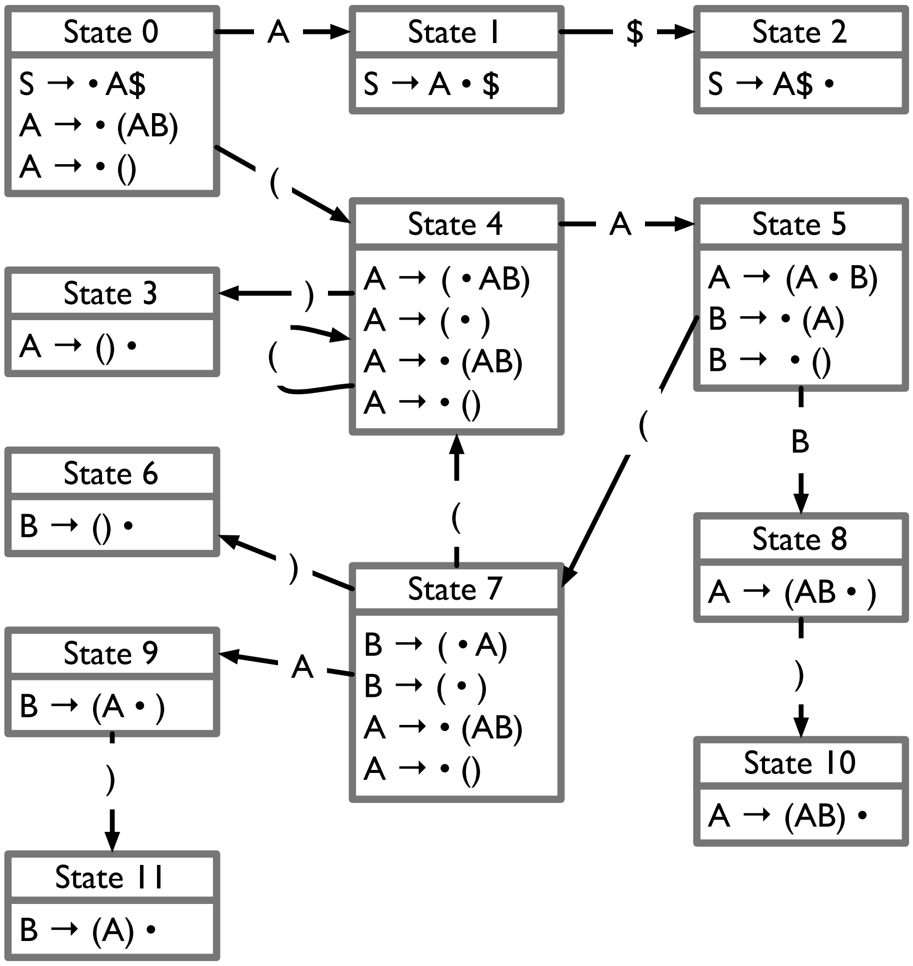
No, there is a predict conflict when predicting an A with a lookahead of '('

Part 5: LR(0) Parsers (34 points)

Use the following grammar for the next questions:

$$S \rightarrow A\$$$
$$A \rightarrow (AB)$$
$$A \rightarrow ()$$
$$B \rightarrow (A)$$
$$B \rightarrow ()$$

1) Fill in the missing information for the for the following CFSM (15 points)



2) List the shift and reduce states in the above CFSM (6 points)

Shift: 0, 1, 4, 5, 7, 8, 9
Reduce: 2 (accept), 3, 6, 10, 11

3) Is this an LR(0) grammar? Why or why not? (1 point)
Yes. No S/R or R/R conflicts

4) For the following sub-questions, use the CFSM you built in the previous question. Each question will provide the state of the parser in mid-parse, giving the state stack (most recent state on the right) and the next token. For each question, give the action the parser will take next, using the format “Shift X” for shift actions (where X is the state being shifted to) and “Reduce R, goto X” for reduce actions (where R is the rule being reduced, and X is the state the parser winds up in after finishing the reduction). Also provide the new state stack. (3 points each)

a) State stack: 0 4 5 7. Next token: (

Shift 4
0 4 5 7 4

b) State stack: 0 4 4 5 8 10. Next token: (

Reduce 2 goto 5
0 4 5

c) State stack: 0 4 4 5. Next token:)

Parse error

d) State stack: 0 4 5 7 4. Next token:)

Shift 3
0 4 5 7 4 3

Part 6 (573 only): LR(1) Parsers (15 points):

1) For the following grammar, show what the *first* state of the LR(1) machine would be (10 points):

$$S \rightarrow AB\$$$

$$A \rightarrow xy$$

$$A \rightarrow Bz$$

$$B \rightarrow Cz$$

$$B \rightarrow \lambda$$

$$C \rightarrow w$$

S → • AB$, { }
A → • xy, {w, $}
A → • Bz, {w, $}
B → • Cz, {z}
B → λ •, {z}
C → • w, {z}

2) Give the action table entries for the first state (i.e., for each token that could be coming up, say whether the parser would shift or reduce). For reduce actions, say what rule would be reduced. (5 points)

Lookahead	Action
x	Shift
y	Error
z	Reduce 5
w	Shift
$	Error