ECE 468/573 — Midterm 1

September 28, 2012

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	10	
2	15	
3	15	
4	25	
5	35	
6	20	
Total	120	

Part 1: Short answers (10 points)

Half credit taken off for not providing justification.

1) You have just come up with a new kind of compiler optimization that can improve the run-time of code by a factor of 10, but is very slow. For code that normally takes 10 seconds to run, the compiler takes 1000 seconds to perform the optimization, but the optimized code runs in 1 second. Would you rather implement this optimization in a standard compiler (that compiles code before it runs) or in a just-in-time compiler? Justify your answer (4 points).

Because this optimization takes a long time, it is hard to justify performing it in a JIT compiler, as the code would have to run for a very long time to make up for the compilation time. Hence, we would prefer to do this in a standard compiler.

2) Suppose this optimization requires seeing the high-level structure of your code (e.g., the loops in the code). Should this optimization take place in the front-end of your compiler? Or the back-end? Justify your answer (3 points).

Because the backend starts with an intermediate representation and produces assembly, it never gets to see the high-level structure of the code (e.g., loops are converted into branches). Hence, the optimization needs to take place in the front-end, where the code structure is still available.

3) You have just been employed by Intel to work on their compiler technology and to give it more powerful optimizations than compilers for ARM. Should you be working on front-end optimizations? Or back-end? Justify your answer (3 points.)

Because the optimizations you are doing are architecture specific, this needs to be done in the back-end, which can be specialized for particular architectures. Note, also, that any optimizations done in the front-end would likely help ARM as well, since they're not architecture-specific: probably not something Intel would want to do!

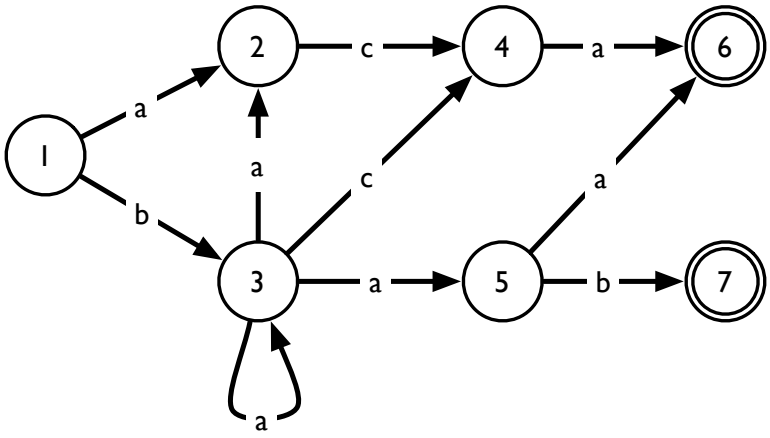
Part 2: Regular expressions, finite automata and scanners (15 points)

1) Give an argument for why we cannot use regular expressions to define programming languages like Micro (hint: think about the way blocks of code can nest inside each other). (6 points):

Languages like Micro have nested blocks that start with BEGIN and end with END (or { and }, etc.) Parsing the language requires matching an equal number of BEGINs with ENDs, which we cannot do with a regular expression.

2) Consider the following NFA. Fill in the transition table below with its corresponding DFA using the subset construction. (8 points):

One point per row



State	Final?	a	b	c
1	no	2	3	err
2	no	err	err	4
3	no	2, 3, 5	err	4
4	no	6	no	no
2, 3, 5	no	2, 3, 5, 6	7	4
6	yes	err	err	err
2, 3, 5, 6	yes	2, 3, 5, 6	7	4
7	yes	err	err	err

3) List which states should be merged when you reduce the above DFA (1 point):

We can combine states 6 and 7: they are both final and they have the same behavior on every input. Note that we cannot combine 2, 3, 5 and 2, 3, 5, 6 because one is final and one is not.

Part 3: Grammars (15 points)

Let G be the grammar:

$$
\begin{aligned}
S &\rightarrow AC\$ \\
A &\rightarrow xAB \\
A &\rightarrow \lambda \\
B &\rightarrow y \\
C &\rightarrow BCz \\
C &\rightarrow \lambda
\end{aligned}
$$

Using this grammar, answer the following questions.

1) Draw the parse tree for the string "xyyz$" (3 points)

Not given here (but everyone got this right)

2) Describe the kinds of strings this grammar can generate. (3 points)

$x^i y^i y^k z^k$ (This question did not get counted)

3) Can the language of this grammar be captured by a regular expression? If so, give the regular expression. If not, give a short argument why not. (4 points)

No, because you need to be able to count the numbers of x's and z's to get the right number of y's, which a regex cannot do. (-2 points for no explanation)

4) Give a grammar that captures the following language, which has some number of a's or b's, followed by the same number of c's or d's (5 points)

$$(a|b)^i(c|d)^i$$

Many possible answers. Here's one:

$$
\begin{aligned}
S &\rightarrow M \\
M &\rightarrow AMC \mid \lambda \\
A &\rightarrow a \mid b \\
C &\rightarrow c \mid d
\end{aligned}
$$

Part 4: LL parsers (25 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 (8 pts)

2 points per set
First(S) = {$, x, y}
First(A) = {x, λ}
First(B) = {y}
First(C) = {y, λ}

2) Give the Follow sets for each non-terminal in the grammar (8 pts)

2 points per set
Follow(S) = {}
Follow(A) = {y, $}
Follow(B) = {y, z, $}
Follow(C) = {z, $}

3) Fill in the following parse table (8 pts)
0.5 points per box

	x	y	z	$
S	1	1		1
A	2	3		
B		4		
C		5	6	6

4) Is this grammar LL(1) or not? Why or why not? (1 pt)

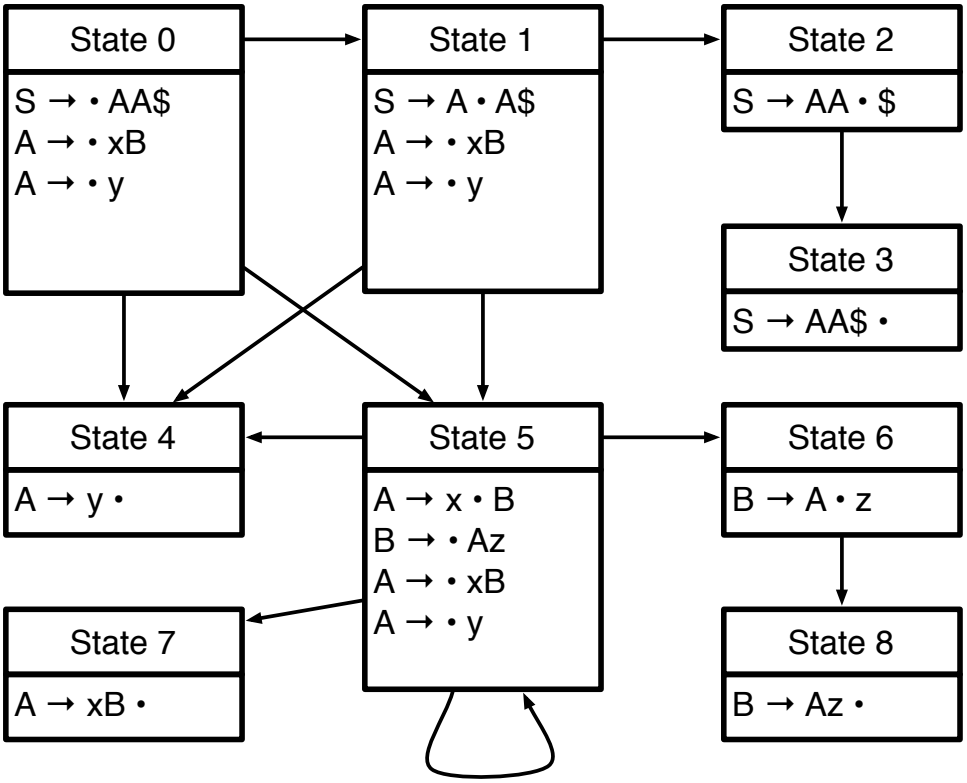
Yes, because there are no conflicts in the parse table.

Part 5: LR(0) Parsers (35 points)

Use the following grammar for the next two questions:

$$
\begin{aligned}
1. S &\rightarrow AA\$ \\
2. A &\rightarrow xB \\
3. A &\rightarrow y \\
4. B &\rightarrow Az
\end{aligned}
$$

1) Fill in the missing information for the for the following CFSM (including edge labels) (14 points) 1 point per configuration, 2 points for edge labels



2) List the reduce states in the above CFSM, and the shift states, assuming state 3 is an accept state. (4 points)

Shift states: 0, 1, 2, 5, 6
Reduce states: 4, 7, 8 (3 ok)

3) Is this an LR(0) grammar? Why or why not? (2 points)

Yes. No S/R conflicts

4) Show the actions taken by the parser as it parses "xyzy$". Actions should either be of the form "Shift X" or "Reduce R (goto X)" If the parser gets to state 3, it accepts. You can use the following table to help show your work (you may not need all of the rows) (15 points)

-3 points per mistake (partial credit given for carrying through errors)

Parse stack	Remaining input	Action
0	x y z y $	Shift 5
0 5	y z y $	Shift 4
0 5 4	z y $	Reduce 3 (goto 6)
0 5 6	z y $	Shift 8
0 5 6 8	y $	Reduce 4 (goto 7)
0 5 7	y $	Reduce 2 (goto 1)
0 1	y $	Shift 4
0 1 4	$	Reduce 3 (goto 2)
0 1 2	$	Shift 3
0 1 2 3		Accept

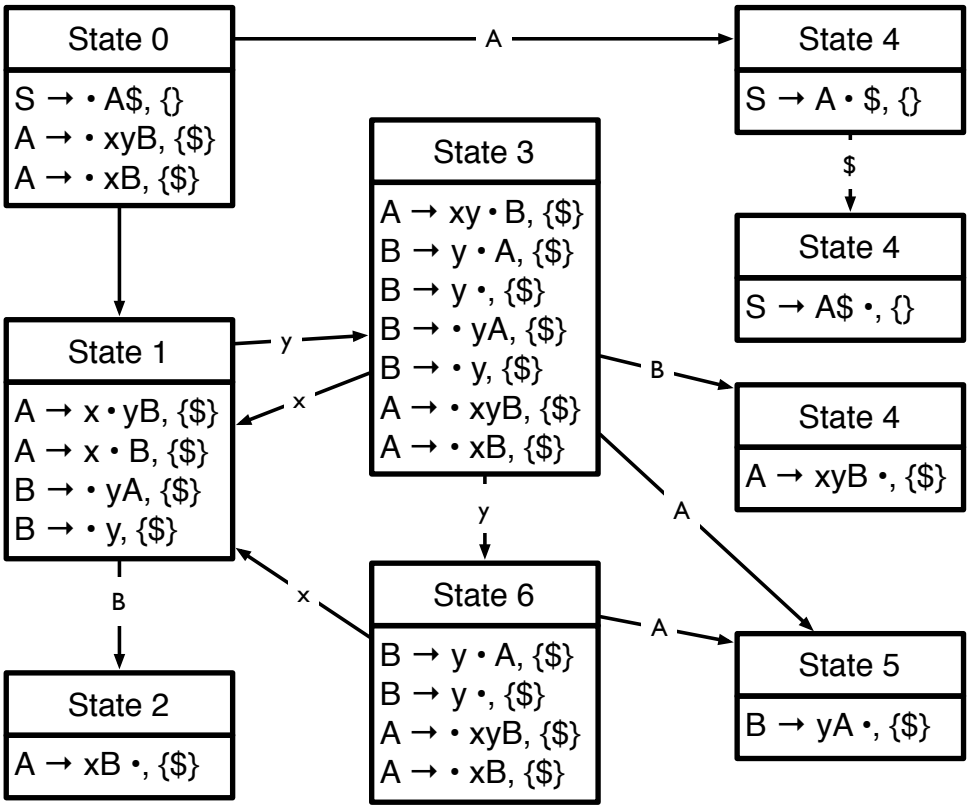
Part 6: LR(1) Parsers (ECE 573 only) (20 points):

Consider the following grammar:

$$
\begin{aligned}
1. S &\rightarrow A\$ \\
2. A &\rightarrow xyB \\
3. A &\rightarrow xB \\
4. B &\rightarrow yA \\
5. B &\rightarrow y
\end{aligned}
$$

1) Fill in the missing information from this partial LR(1) machine (15 points)

1 point per configuration



2) Show what the action table entries for State 3 would be (i.e., give the row for State 3). For reduce actions, give what rule would be reduced (4 points)

x Shift
y Shift
$ Reduce(5)

3) Is this grammarLR(0)? Why or why not? (1 point)

No. Without lookahead, there would be shift/reduce conflicts in states 3 and 6.