CSC173 FLAT
Module Exam

Chris Brown

September 26, 2008

Please write your name on the bluebook. This is a closed-book Exam. There are 75 possible points (one per minute). Best not to spend more minutes on a question than it's worth. Stay cool and please write neatly.

1 FFQ

Notable problems:

1. Not knowing what an ambiguous grammar is. Just because it's not LL(0) (needs some lookahead) doesn't mean it's ambiguous, or even that it's not linear-time. Ambiguous ONLY means "more than one derivation" or "more than one parse tree". In question 3 it's easy to derive b from B in different ways.

2. Not knowing how to recognize CFG or CSG. I thought I said time and again that context free means just one symbol on LHS of production. More than one symbol on the left hand side of a production (e.g. 0A $\rightarrow$ 01) means the LHS has context, so it's not context-free.

3. Buzzwords (sorry, technical terms) are OK but better is demonstrating what they mean: 'left factoring' is what comes to mind this time. If you know this means replace
   A $\rightarrow$ CD
   A $\rightarrow$ CE
   with
   A $\rightarrow$ C Ctail
   Ctail $\rightarrow$ D
   Ctail $\rightarrow$ E
   then I'd expect you to say so.

4. Proof: Your Google boss (Q. 4) isn't just going to take your word for it that "DFA's or Reg. Exprs have no memory", true as it is. A statement isn't a proof, even if true. Easiest argument is that if you give me a DFA of any size N (N states), then for any $m > N$, the input a^m will force it to finish in a state it has already visited (Pigeonhole principle, CSC172 (I would hope)), so it can't tell the diff. between a^m and some shorter $a^k, k < m$.

5. On MY side, it's highly ironic but not surprising that I had a misprint in my "misprint" question. Sigh...

   The misprint I had in mind (his) meant you should change S $\rightarrow \lambda$ to C $\rightarrow \lambda$).

   My misprint gave the grammar the production C $\rightarrow$ C0CAB for the correct C $\rightarrow$ 0CAB. This can generate what we want as many of you discovered if you always vanish the initial C with $C \rightarrow \lambda$. But if you DON'T send that initial C to λ but expand it you generate rubbish like 012012. The guy who discovered this cute little gotcha received extra credit.

6. Michael wanted me to be sure you knew what FIRST and FOLLOW sets are and I certainly didn't since you certainly don't. Q.3.2. was probably the most Fly Fed Q in the exam. Just for example, lots of youse said $FIRST(A) = c$. Look up the definition! I claim $FIRST(A) = c, d, e, \lambda$. The set answers the question: what tokens can you possibly find when looking for A? Also, these sets are sets of tokens: as a parser you're seeing the output of the scanner, which is terminals (tokens), not non-terminals (figments of the grammar).

2 Regular Expression to Minimized DFA (30 min)

Given the regular expression `(ab)* | a*` :

1. Create an equivalent nondeterministic finite automaton by our "mini-machine" method (Thompson's method).

2. Use the subset construction to convert the NDFA to a DFA.

3. Use the equivalence class argument to minimize the DFA.

3 Regular Expressions (15 min)

1. (5 min) Ignoring the 3-step conversion procedure of the last problem, directly create a regular expression and a deterministic finite automaton (circles and arrows) for the language: "All strings of letters from the set $\{a, b, c\}$ such that every b is preceded by an a." (e.g. Accept ab, ccc, aa, ccaabab. Reject b, ba, ccb, abb, abcabb.)

2. (10 min) Likewise, see if you can directly write down a deterministic finite automaton for the language: "All strings of 0's and 1's that either have exactly two 1's or end in 1." E.g. 1010 is good (exactly two 1's), 111 and 101010101 and 00001 are good (end in 1), 1001 is good (meets both conditions), 101010 and 0000 fail (not two 1's, don't end in 1). [hint: looked at in class: my first attempt to make the FA in class was wrong: I now think I have a 7-state solution].

4 Parsing (10 min)

Here's a CFG grammar with starting symbol S, nonterminals S,A,B,C,D,E; terminals b,c,d,e; λ the null symbol.

S $\rightarrow$ AB
A $\rightarrow$ CD
A $\rightarrow$ CE
B $\rightarrow$ CB
B $\rightarrow$ b
C $\rightarrow$ c
C $\rightarrow$ λ
D $\rightarrow$ d
D $\rightarrow$ λ
E $\rightarrow$ e

1. Is this an LL(1) grammar? If not why not and how do you fix it so it is?
2. (Doesn't depend on first part) What are the FIRST and FOLLOW sets of nonterminal A?
3. (Doesn't depend on first part) Is this grammar ambiguous? Show why or why not.

5 A Promotion Opportunity? (10 min)

Your new boss at Google is a mathematics PhD who wants you to design a finite state machine (or regular expression) that accepts the strings $a^n b^{2n}, n = (0, 1, 2, ...)$.

Give him the machine (Regular expression, FA diagram, or (S, S_0, C, F, T) sets), or explain to him why you can't. (He'll want a solid argument. Better, a proof).

6 Quotation Correction++ (10 min)

Here's a quote from Rosen, (one of our texts): *Discrete Mathematics and its Applications,* p. 789.

> One grammar that generates the set $\{0^n1^n2^n \mid n = 0, 1, 2, ...\}$ is $G = (V, T, S, P)$ with [symbols] $V = \{0, 1, 2, S, A, B, C\}$, [terminals] $T = \{0, 1, 2\}$; starting state S; and productions [with λ = the null symbol]
>
> S $\rightarrow$ C
> C $\rightarrow$ 0CAB
> S $\rightarrow \lambda$
> BA $\rightarrow$ AB
> 0A $\rightarrow$ 01
> 1A $\rightarrow$ 11
> 1B $\rightarrow$ 12
> 2B $\rightarrow$ 22

1. There's a simple and obvious (one letter) misprint in the productions [hint: left-hand side]. Fix it.
2. Using your fixed grammer, give a derivation tree for `001122`.
3. What are the most specific technical names of the grammar's type, the associated language's type, and the associated automaton's type?