CS 3100 – Models of Computation – Fall 2011 – Midterm-2
75 mins - 75 points (3 extra points worth of questions provided)

- Note: Right-linear grammars are often easily obtained from NFA.
- In all questions, @ or ε mean "epsilon," the empty string.
- NFA states beginning with `I` are initial states; those beginning with `F` are accepting states; and those beginning with `IF` are both.
- Things such as `0, 1, 2, #, a, b, c` will be considered to be terminals (members of Σ).
- While all regular languages are context-free, when we say "context-free" in a general setting, it always means "context-free and not regular."
- Some languages may be described through regular expressions or CFGs, for convenience (*i.e.* I may not write a set definition for the language, if that is harder to interpret than an RE or a CFG).
- Let $m/2$ mean $m \; div \; 2$.

1 (5 points per question)

Study the languages below. If a language is regular, then say "regular," and write down a right-linear grammar for it. If it is context-free, then write down "context-free," and write down a context-free grammar for it. If it is not context-free, describe (a few sentences) how a Turing machine could recognize strings in the language.

1. $L_1 = 0^* \; 1^*$

   ```
   S -> 0 S | @ | @ A  -- Note: this could also be S -> 0 S | @ | A
   A -> @ | 1 A
   ```

2. $L_2 = \{1^{2n}0^n \mid n > 0\}$

   ```
   S -> 1 1 S 0 | 1 1 0
   ```

3. $L_3 = \{ww \mid w \in \{0,1\}^*\}$

   A TM can scan the string left to right, marking off the symbols in the first w against those in the second w.

   It will need to use one of two methods:

   - A deterministic algorithm where each position is tried as a possible midpoint. This will be a Polynomial-time (anyhow, a cubic-time) algorithm.
   - A non-deterministic algorithm which guesses the middle and matches around it. This is again a non-deterministic Polynomial-time (but non-deterministic quadratic-time) algorithm.

2 (4 points per question)

1. Is $L_4 = L_2 \cup L_1$ regular or context-free? Clearly argue in a few sentences. This is context-free because L_2 and L_1 do not overlap, and L_2 is context-free.

2. Is $L_5 = L_2 \cup L_6$ (where $L_6 = 1^* \; 0^*$) regular or context-free? Clearly argue in a few sentences.

   This is regular because L_6 contains L_2.

3. Is $\overline{L_3}$ regular, context-free, or non-context-free? Explain clearly in a few sentences. Your argument must cover exactly what $\overline{L_3}$ is.

   This is context-free, as was shown in class – including all odd-length strings, plus even length strings *not* of the form ww, for which there is a context-free grammar.

3 (5 points per question)

1. Draw a push-down automaton for L_2 **without using** the CFG to PDA conversion.

   This PDA is quite simple: push 1s, and when 0s arrive, pop two 1s for each 0. When Z surfaces, accept.

2. Simplify the following CFG, calling the simplified grammar G_x. Indicate which non-terminals are generating and which are unreachable from S.

   ```
   S -> a S b S | b S a | T

   T -> @ | U T

   U -> 0 U | 1 V
   ```

   We can remove `U` and `V`, as `U is non-generating`. Every non-terminal is reachable. The simplified grammar is

   ```
   S -> a S b S | b S a | @
   ```

4 Variable points per question

1. (**2 points**) Is the grammar G_x consistent with respect to the language L_{eqab} of strings with equal a's and b's ? If so, argue in a few sentences.

   Yes consistent, as it does not violate the contract of equal as and bs.

2. (**4 points**) Is the grammar G_x complete with respect to the language L_{eqab} ? If so, show it. If not, write down a string in L_{eqab} that cannot be derived from G_x.

   Not complete; no way to generate $baab$.

3. (**2 points**) Reverse G_x, writing down the result as grammar G_r.

   The reversed grammar is

   ```
   Sr -> Sr b Sr a | a Sr b | @
   ```

5 (5 points)

How does this TM accept 00? Write down a sequence of tape contents with the tape head shown. You don't have to show rejecting configurations at all.

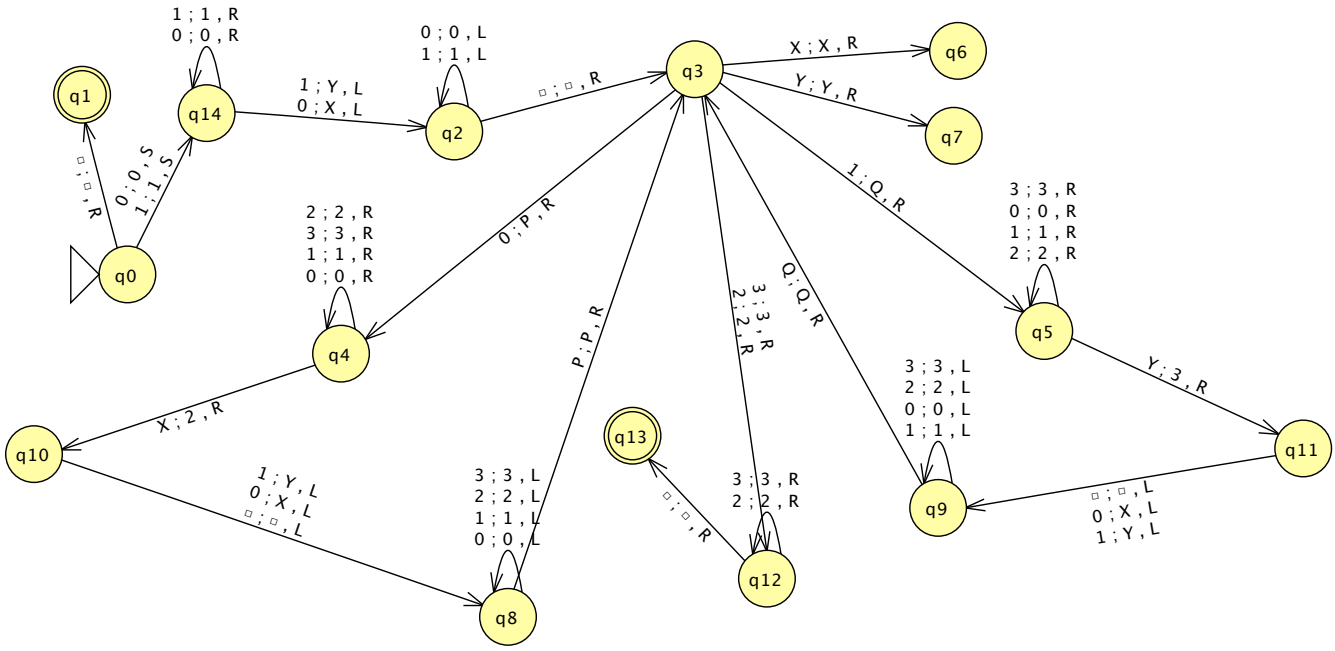

Figure 1: An NDTM for ww

6 (5 points per question)

1. In the PL proof for $L_{neq} = \{0^i 1^j \mid i < j\}$ given an m-state DFA, someone picks the string $0^{m/2}1^{m/2+1}$. They then pick $y = 0$ and show that by pumping, we can arrive at the string $0^{m/2+1}1^{m/2+1}$ which is outside L_{neq}. Is this proof correct? Why or why not?

   Incorrect, because in such an initial string, the "pump" can be either in the 0 region, in the 1 region, or straddle both. Just considering one case ($y = 0$) is insufficient; the contradiction is incompletely demonstrated. The non-demonstrated cases may evade contradiction.

2. If the proof isn't correct, write down a correct proof.

   Suppose we pick a longer string, *i.e.*, $0^m 1^{m+1}$ which is in the language. Then the pump is clearly confined to the 0s. Then regardless of the length of the pump (and we know that it is at least 1), one pump *up* will take the string outside the language.

7 Variable Points

1. Consider the grammar of regular expressions without the Kleene-star and ε (or `@`).

   ```
   RE -> RE + RE | RE RE | (RE) | a | b | c
   ```

2. Which of these strings has an ambiguous parse? For the case with an ambiguous parse, draw the different parse trees. For the case without an ambiguous parse, draw a single parse tree.

   (a) **(3 points)** `a+bc` This has two parse trees, one giving precedence to + and the other to concatenation.

   (b) **(2 points)** `(a+b)c` This has only one parse.

3. **(5 points)** Write a disambiguated RE grammar that is language-equivalent to the above grammar.

   The following grammar gives precedence to concatenation, allowing the ambiguity to be removed.

```
RE -> RE + Term | Term
Term -> Term Factor | Factor
Factor -> a | b | c | ( RE )
```

8 (2 points each)

Provide an answer or say "impossible," justifying why:

1. What are inherently ambiguous languages (one sentence)? Provide an example of such a language (1-2 sentences).

   An inherently ambiguous language is one for which every grammar admits an ambiguous parse for some string.

2. What is the Post Correspondence Problem and why is it of great importance in computer science? (2-3 sentences).

   The PCP encodes a TM's ability to halt on a given input. If an algorithm can answer this "yes/no" (yes, M halts on input w; no it does not), then one can solve many "impossible" problems, including CFG ambiguity.

3. Present an algorithm (in a few sentences) to intersect an NFA and a PDA (2-3 sentences).

   An NFA and a PDA can be intersected in the standard way, barring a few caveats. Basically we need to "match" the moves. The shortest answer: if there are no epsilon based moves in either, then we can intersect similar to a DFA intersection algorithm. Else the algorithm can be a bit more complicated. (I'll try to describe it in another place..)

4. Among the machine types "FA" (NFA/DFA), "PDA" (DPDA/NPDA), "LBA" (DLBA/NLBA), and "TM" (DTM/NDTM), for which machine types (FA, PDA, LBA, TM) does the presence of non-determinism change the *expressive power*? Explain if any of these results are open. (2-3 sentences).

   The power changes with PDA (strictly more expressive); with LBA, it is unknown; with others, it remains the same.