Midterm Exam #2

INSTRUCTIONS:

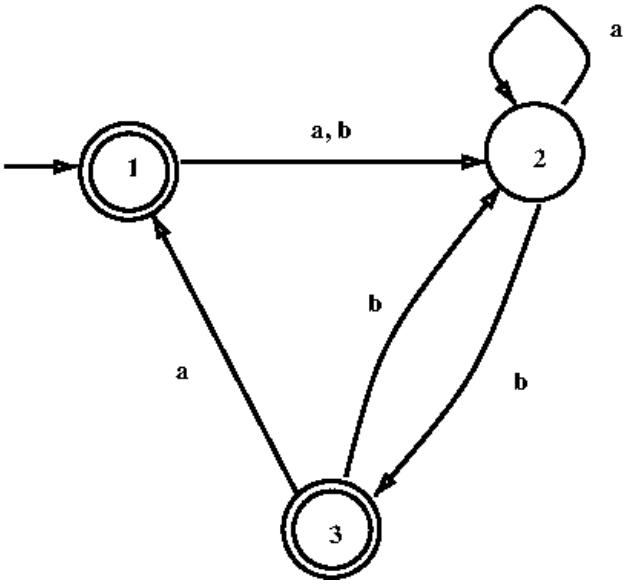
- Put your NAME and SBU ID # on this exam booklet in the space provided.
- This is a CLOSED-BOOK exam, which TERMINATES AT 12:50PM (80 minutes). NO ELECTRONIC DEVICES, including calculators, may be used during the exam.
- Please place ALL ANSWERS IN THIS BOOKLET, on the sheet where the corresponding question is printed.
- THINK BEFORE YOU WRITE. A partial solution can get you partial credit, but too much extraneous information can prevent me from finding your correct solution.
- SOME QUESTIONS ARE HARDER THAN OTHERS, and you might not have time to answer all questions completely. LOOK OVER ALL THE QUESTIONS BEFORE STARTING, and work first on those that will get you the most credit fastest. Use the number of points listed for each question as a guide.

Question:	1	2	3	4	5	6	7	Total
Points:	10	10	10	10	10	10	20	80
Score:								

Note: Point values have been assigned so that you should expect to be answering roughly one point per minute.

NAME and SBU ID#: _____

1. (10 points) Find a regular expression equivalent to the following NFA. For full credit, you must include some kind of plausible description of how you obtained your result.



Solution: Use the procedure in Sipser Lemma 1.60. The first step is to add a new initial state and accept state. After that, the other states are eliminated one by one. The order in which the states are eliminated will affect the result. For example:

1. Eliminate 1, then 2, then 3:

$$\epsilon \cup (a \cup b)\{a^*b(a(a \cup b) \cup b)a^*b)^*\}(\epsilon \cup a)$$

2. Eliminate 1, then 3, then 2:

$$\epsilon \cup (a \cup b)\{(a \cup b(a(a \cup b) \cup b))^*b\}(\epsilon \cup a)$$

   Note that the terms in curly braces are equivalent.

2. (10 points) Prove that the following language is not regular:

$$L = \{0^m 1^n \mid m \neq n\}.$$

Solution: Suppose, for the purpose of obtaining a contradiction, that it were regular. Then its complement $\overline{L}$ would also be regular, as would the intersection $\overline{L} \cap 0^*1^*$. But $\overline{L} \cap 0^*1^*$ is the language $\{0^m 1^n \mid m = n\}$; *i.e.* $\{0^n 1^n \mid n \geq 0\}$, which we have previously shown by the Pumping Lemma is not regular. This is a contradiction, and we conclude that the given language L cannot be regular.

3. (10 points) Consider the following context-free grammar G:

$$\begin{aligned} S &\to A \\ S &\to SS \\ A &\to (S) \\ A &\to \epsilon \end{aligned}$$

Give a grammar G' that is equivalent to G, but which is in Chomsky Normal Form.

Solution:

1. Add new start variable:

$$\begin{aligned} S_0 &\to S \\ S &\to A \\ S &\to SS \\ A &\to (S) \\ A &\to \epsilon \end{aligned}$$

2. Remove ϵ rule ($A \to \epsilon$):

$$\begin{aligned} S_0 &\to S \\ S &\to A \\ S &\to \epsilon \\ S &\to SS \\ A &\to (S) \end{aligned}$$

3. Remove ϵ rule ($S \to \epsilon$):

$$\begin{aligned} S_0 &\to S \\ S_0 &\to \epsilon \\ S &\to A \\ S &\to SS \\ A &\to (S) \\ A &\to () \end{aligned}$$

4. Remove unit rule ($S \to A$):

$$\begin{aligned} S_0 &\to S \\ S_0 &\to \epsilon \\ S &\to (S) \\ S &\to () \\ S &\to SS \\ A &\to (S) \\ A &\to () \end{aligned}$$

5. Remove unit rule ($S_0 \to S$):

$$\begin{aligned} S_0 &\to \epsilon \\ S_0 &\to (S) \\ S_0 &\to () \\ S_0 &\to SS \\ S &\to (S) \\ S &\to () \\ S &\to SS \\ A &\to (S) \\ A &\to () \end{aligned}$$

6. Put rules in proper form and drop rules for variable A which is unreachable from S_0:

$$\begin{aligned} S_0 &\to \epsilon \\ S_0 &\to LS_1 \\ S_1 &\to SR \\ S_0 &\to LR \\ S_0 &\to SS \\ S &\to LS_1 \\ S &\to LR \\ S &\to SS \\ L &\to (\\ R &\to) \end{aligned}$$

Final Result:

$$
\begin{aligned}
S_0 &\rightarrow \epsilon \\
S_0 &\rightarrow LS_1 \\
S_1 &\rightarrow SR \\
S_0 &\rightarrow LR \\
S_0 &\rightarrow SS \\
S &\rightarrow LS_1 \\
S &\rightarrow LR \\
S &\rightarrow SS \\
L &\rightarrow (\\
R &\rightarrow)
\end{aligned}
$$

4. Consider the following CFG:

$$S \rightarrow \epsilon \mid aS \mid aSbS$$

(a) (4 points) Give a *leftmost* derivation for the following string: *aababbab*.

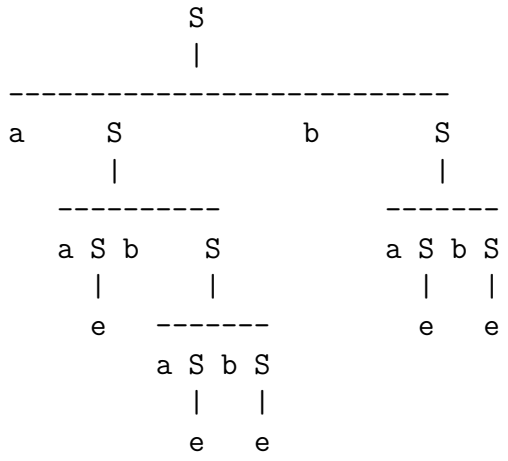
Solution:

$$S \Rightarrow aSbS \Rightarrow aaSbSbS \Rightarrow aabSbS \Rightarrow aabaSbSbS \Rightarrow aababSbS$$

$$\Rightarrow aababbS \Rightarrow aababbaSbS \Rightarrow aababbabS \Rightarrow aababbab$$

(b) (6 points) Draw the parse tree corresponding to your derivation:

Solution:



5. (10 points) Draw the transition diagram for a pushdown automaton that recognizes the language generated by the following grammar:

$$\begin{aligned} S &\to \epsilon \\ S &\to AS \\ A &\to aSb \\ A &\to a \end{aligned}$$

Solution: Use the construction in the proof of Lemma 2.21.

- One transition from q_{start} to q_{loop}: $\epsilon, \epsilon \to S\$$.
- Several transition loops that start and end on q_{loop}:
  - $\epsilon, S \to \epsilon$
  - $\epsilon, S \to S$ $\quad \epsilon, \epsilon \to A$
  - $\epsilon, A \to b$ $\quad \epsilon, \epsilon \to S$ $\quad \epsilon, \epsilon \to a$
  - $\epsilon, A \to a$
  - $a, a \to \epsilon$
  - $b, b \to \epsilon$
- One transition from q_{loop} to q_{end}: $\epsilon, \$ \to \epsilon$.

6. Example 2.36 in Sipser shows that the language $\{a^n b^n c^n \mid n \geq 0\}$ is not context-free.

   (a) (4 points) Use the languages $A = \{a^m b^n c^n \mid m, n \geq 0\}$ and $B = \{a^n b^n c^m \mid m, n \geq 0\}$ together with Sipser Example 2.36 to show that the class of context-free languages is not closed under intersection.

Solution: The languages A and B are both context-free. For example, a CFG that generates A is:

$$\begin{aligned} S &\rightarrow AX \\ A &\rightarrow \epsilon \mid aA \\ X &\rightarrow \epsilon \mid bXc \end{aligned}$$

A similar CFG generates B. The intersection $A \cap B$ is the language $\{a^n b^n c^n \mid n \geq 0\}$, which Example 2.36 shows is not context-free. Since A and B are context-free but their intersection is not, this shows that the class of context-free languages is not closed under intersection.

(b) (6 points) Use part (a) and DeMorgan's law to show that the class of context-free languages is not closed under complementation.

Solution: Suppose the class of context-free languages were closed under complementation. Then since A and B are context-free it would follow that their complements, $\overline{A}$ and $\overline{B}$, are also context-free. Using the already proved fact that the class of context-free languages is closed under union, it would follow that $\overline{A} \cup \overline{B}$ and hence $\overline{(\overline{A} \cup \overline{B})}$ are context-free. But DeMorgan's law says that $\overline{(\overline{A} \cup \overline{B})} = A \cap B$, which would show that $A \cap B$ is context-free, a contradiction with part (a). We conclude that the class of context-free languages cannot be closed under complementation.

7. Let $\Sigma = \{0, 1\}$. Let $WW_k = \{ww \mid w \in \Sigma^*$ and w is of length $k\}$.

   (a) (10 points) Show that for each k, no DFA with fewer than 2^k states can recognize WW_k.

   Note, the exam handed out had a typo and said "fewer than k states". It is actually easier to prove that no DFA with fewer than k states can recognize WW_k than it is to prove that no DFA with fewer than 2^k states can recognize WW_k, so the typo doesn't really affect this part of the problem, other than to make it easier. I was not looking just for "the DFA needs k states to determine where w ends"; rather, I was looking for a rigorous proof, which would use the Pigeonhole Principle.

   Solution: Suppose M is a DFA that recognizes WW_k. If M has fewer than 2^k states, then since there are 2^k strings in Σ^k, by the Pigeonhole Principle there must exist distinct x, $y \in \Sigma^k$ such that x and y take M from the initial state to the same state q. But then since M accepts xx and yy, which are both in WW_k, and since M reaches the same state q after reading x as after reading y, then it must also accept xy, which is not in WW_k. We conclude that M must have at least 2^k states.

(b) (10 points) Describe a much smaller NFA for $\overline{WW}_k$, the complement of WW_k. (**Note:** A clear description, including an analysis of the number of states, is required for full credit. It is not necessary to give a formal description.)

This part of the problem is more difficult, and I did not expect that very many people would be able to do it. Note that the typo from part (a) does affect this part to some extent, since it is hard to imagine how one might create an NFA to recognize $\overline{WW}_k$ that is "much smaller" than k states. However, very few people had any kind of correct characterization of $\overline{WW}_k$ or a description of an NFA that came anywhere even close to recognizing that language, so I don't think it really made much difference in the end. It is certain that a fixed number of states (not depending on k) will not work, even though many people drew state diagrams for NFAs with small fixed numbers of states and claimed that they worked.

Solution: The language $\overline{WW}_k$ consists of all strings that are not of the form ww for some $w \in \Sigma^k$. Equivalently, a string is in $\overline{WW}_k$ if and only if either it does not have length $2k$, or else it is of the form xy with $|x| = |y|$, but $x \neq y$. That is, the language $\overline{WW}_k$ is the union of the language $L_1 = \{w \in \Sigma^* \mid |w| \neq 2k\}$ and the language $L_2 = \{xy \in \Sigma^* \mid |x| = |y| = k \text{ and } x \neq y\}$. It is straightforward to construct a DFA M_1 with $2k + 1$ states that recognizes L_1. We can also construct an NFA N_2 that recognizes L_2 by guessing the index at which strings x and y will disagree and then checking that this is in fact the case. In more detail, N_2 will read its first k input symbols and nondeterministically choose at one point to remember the current input position i and whether the symbol read at that position was 0 or 1. It will then read the remaining input, accepting only if the length of the remaining input is exactly k and the symbol at the ith position in the second half differs from that at the ith position in the first half. The NFA N_2 can be constructed using $O(k^2)$ states: one factor of k is used to count through the first k input symbols and remember the position of the guessed difference, another factor of k is used to count through the second k input symbols. There is an additional factor of 2 (covered by the "big-O") required to remember whether the symbol at the guessed position in the first half was 0 or 1. The DFA M_1 can now be combined with the NFA N_2, using a technique shown in the textbook, to obtain an NFA N that recognizes the union $L_1 \cup L_2 = \overline{WW}_k$. The NFA N will have $O(k + k^2) = O(k^2)$ states, which is asympotically much smaller than 2^k states.

(scratch paper)

(scratch paper)