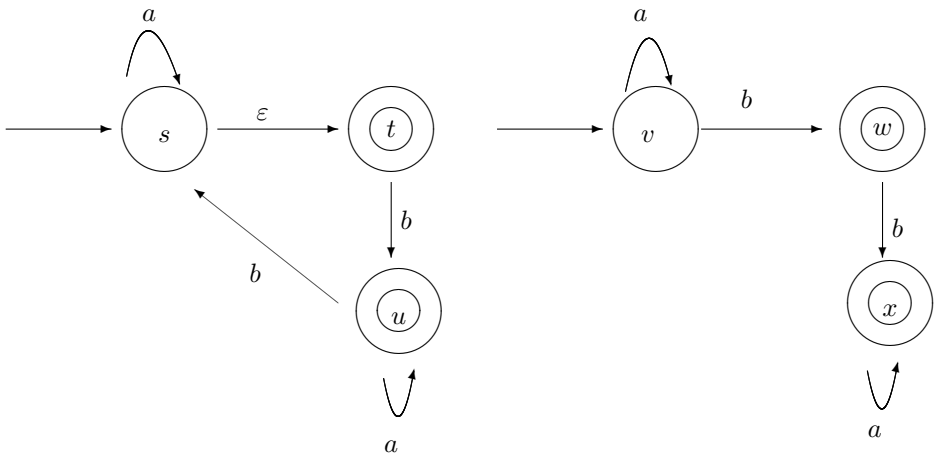
CS 420 Spring 2019
Final Exam Solutions

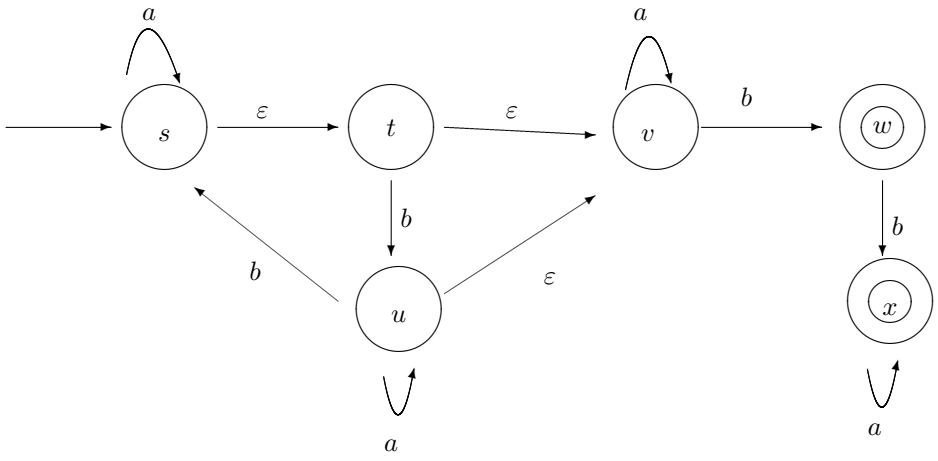
Name: _____

Put all your answers on the test itself. Be sure to put your name above.

1. Using the method from class (which is the same as the method in Theorem 1.47), give the state diagram for an NFA that recognizes the concatenation of the language recognized by the first automaton below with the language recognized by the second automaton below.:



Answer:



[10 points]

2. Apply the method from class that decides E_{CFG} to the following CFG and answer the questions below.

$$\begin{aligned} S &\to AbS|AW \\ T &\to TS|VWY \\ A &\to AT|SVa \\ V &\to aYbW|AT \\ W &\to aWX|aY \\ X &\to aX|\varepsilon \\ Y &\to bY|a \end{aligned}$$

(a) List the terminals and variables you mark in the order they get marked. (List each terminal and variable only the first time you mark it. There is more than one possible order.)
$a, b|X, Y|W|V|T$

(b) Does the CFG belong to E_{CFG}? Yes

(c) How does your answer to (b) follow from your answer to (a)? (You must refer specifically to your answer in (a) when you answer this question.)
S is not marked

[10 points]

3. Let G be the following Chomsky Normal Form grammar.

$$\begin{aligned} S &\rightarrow AB|BC|b \\ A &\rightarrow BA|CB|a|b \\ B &\rightarrow BA|c \\ C &\rightarrow CB|a \end{aligned}$$

Here is a partially filled in table for the algorithm described in Theorem 7.16 to determine if a particular string belongs to $L(G)$.

		j				
		1	2	3	4	5
	1	A, C	A, C, S	A, C, S	?	?
i	2	–	B	A, B, S	A, B, S	?
	3	–	–	A, C	A, C, S	A, C, S
	4	–	–	–	B	A, B
	5	–	–	–	–	A, S

(a) What is the string w for this table? (For this grammar, you can tell just by looking at the table.)
acacb

(b) What are the three missing entries in the table? Give complete answers.

$$\begin{aligned} table(1,4) &= A, C, S \\ table(2,5) &= A, B, S \\ table(1,5) &= A, C, S \end{aligned}$$

(c) Is w in $L(G)$? Yes

(d) How does your answer to (c) follow from your answer to (b)?
$S \in table(1,5)$

[10 points]

4. Let G be the grammar

$$\begin{aligned} S &\rightarrow WbT \\ T &\rightarrow aWbT|bVaT|\varepsilon \\ W &\rightarrow aWbW|\varepsilon \\ V &\rightarrow bVaV|\varepsilon \end{aligned}$$

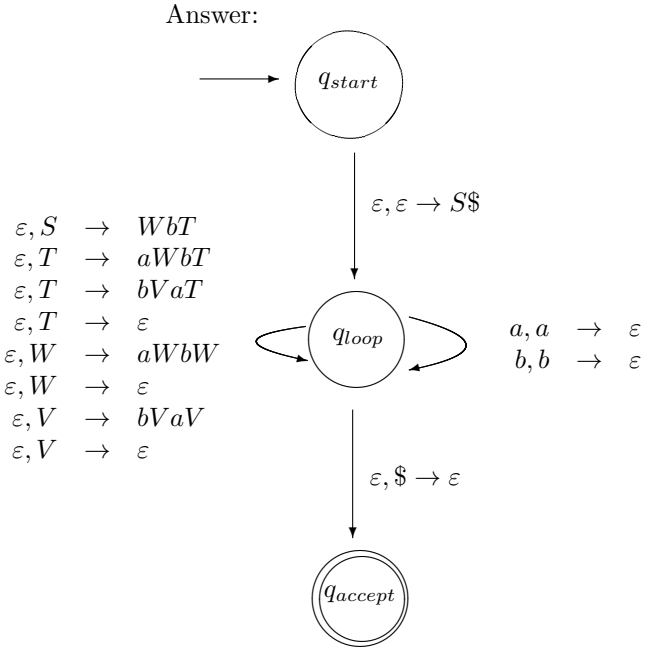
(a) Give a leftmost derivation of the string $aabbbbaab$ in G. (It may help you to know that $L(G)$ is the set of strings w over $\{a, b\}$ that have exactly one more b than a, that T generates the set of strings with the same number of a's as b's, Wgenerates the set of strings with the same number of a's as b's that also have the property that every prefix of the string has at least as many a's as b's, and V generates the strings with the same number of a's as b's such that every prefix of the string has at least as many b's as a's.) Do not combine steps. Your derivation should have 11 steps.

Answer:

$S \Rightarrow WbT \Rightarrow aWbWbT \Rightarrow aaWbWbWbT \Rightarrow aabWbWbT$
$\Rightarrow aabbWbT \Rightarrow aabbbT \Rightarrow aabbbbVaT \Rightarrow aabbbbaT$
$\Rightarrow aabbbbaaWbT \Rightarrow aabbbbaabT \Rightarrow aabbbbaab$

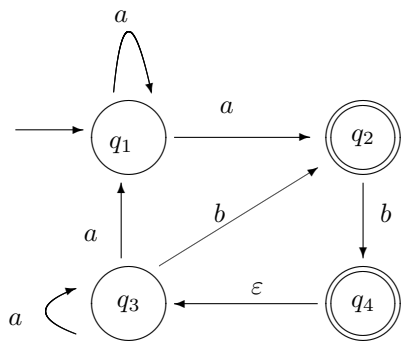
(b) Using the procedure given in Theorem 2.20, give a PDA equivalent to G. (You may use moves that push more than one symbol onto the stack.)

Answer:



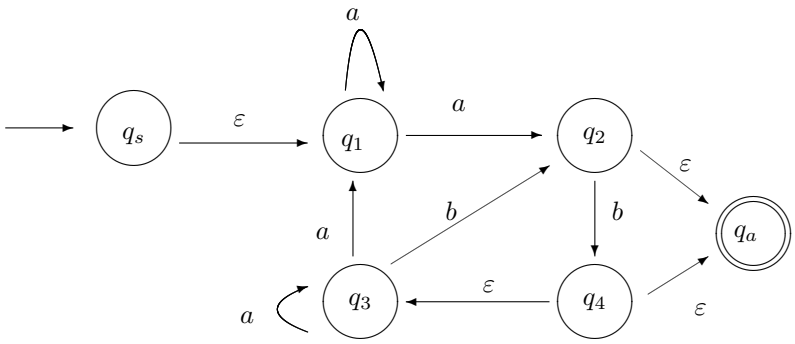
[10 points]

5. Let M be the NFA given by

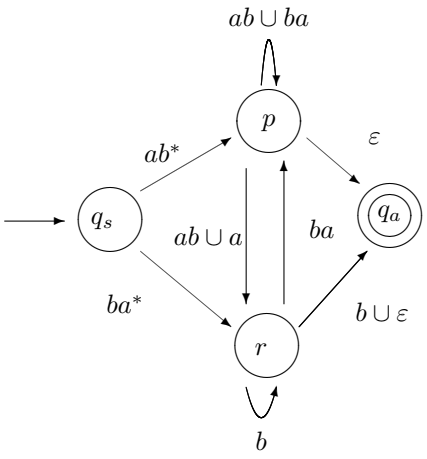


(a) If you want to transform M into an equivalent regular expression, you have to first modify M into a GNFA M' *before* you start eliminating states. Give the state diagram of this GNFA M'. (You are *not* being asked to eliminate any states. Just give the GNFA M' that you first transform M into.) Do not give transitions labeled with $\emptyset$.

Answer:

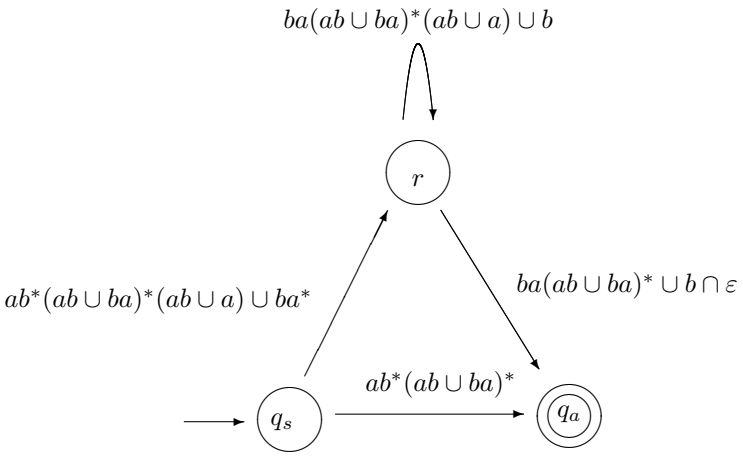


(b) Suppose that at some point while transforming an NFA into a regular expression you have the following GNFA.



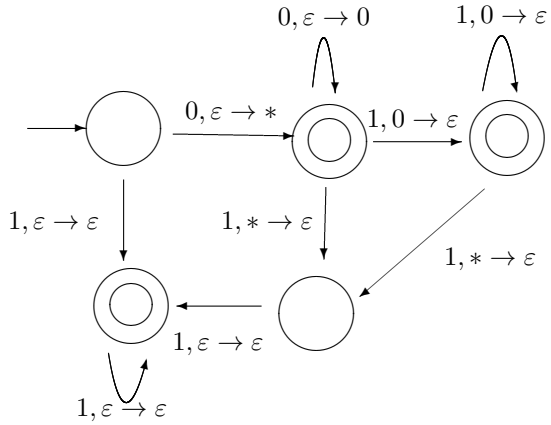
Show the GNFA you would get from this one by eliminating state p. (You are not being asked to convert the GNFA into a regular expression. Just eliminate the state p.)

Answer:



[10 points]

6. Let M be the PDA given by:



(a) Does M accept the string 11? Yes

(b) Does M accept the string 001? Yes

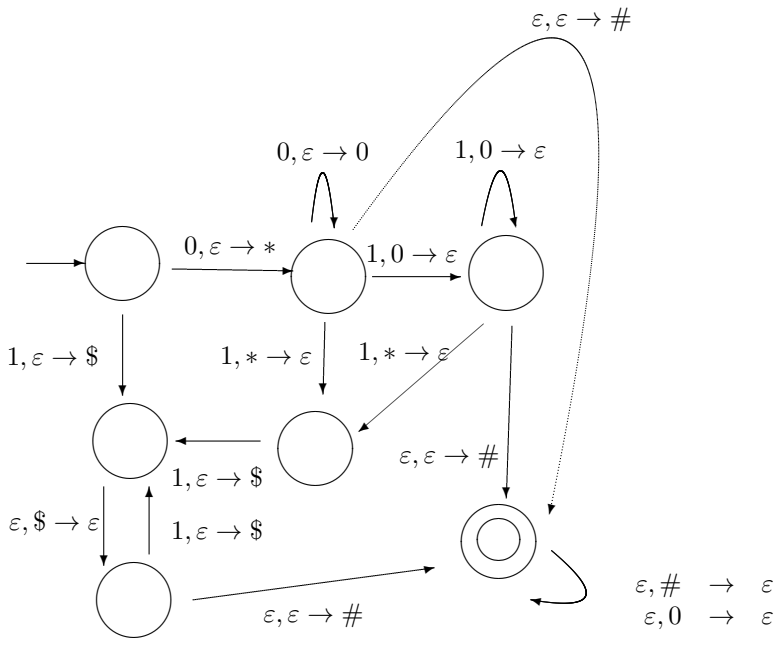
(c) Does M accept the string 0011? No

(d) Does M accept the string 00111? Yes

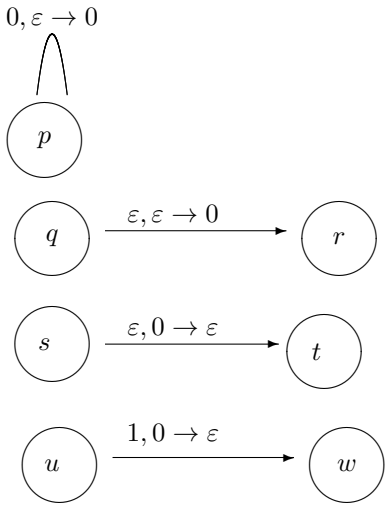
(e) What is the language recognized by M?
Answer:
$\{0^n 1^m | n \neq m\}$

(f) If you want to convert the PDA M into a context-free grammar, then the first thing you have to do is transform M into another PDA M'. Give this PDA M'. (You are *not* being asked to transform M into a CFG. Just give the PDA M'.)

Answer:



(g) Suppose you are converting a PDA into a CFG and the only moves in the PDA that involve the stack symbol 0 are the following:



The grammar will contain four rules corresponding to the different ways to push and pop a 0. What are these four rules?

Answer:

$$\begin{aligned} A_{pt} &\to 0A_{ps} \\ A_{pw} &\to 0A_{pu}1 \\ A_{qt} &\to A_{rs} \\ A_{qw} &\to A_{ru}1 \end{aligned}$$

[13 points]

7. Show that the following grammar is ambiguous. (The grammar generates the same language as the grammar given in Problem 4)

$$\begin{aligned} S &\rightarrow TbT \\ T &\rightarrow aTbT|bTaT|\varepsilon \end{aligned}$$

Answer:

The string *bab* has the following two leftmost derivations, so the grammar is ambiguous.

$$S \Rightarrow TbT \Rightarrow bTaTbT \Rightarrow baTaT \Rightarrow babT \Rightarrow bab$$

and

$$S \Rightarrow TbT \Rightarrow bT \Rightarrow baTbT \Rightarrow babT \Rightarrow bab$$

[7 points]

8. Let A be the language $\{u \in \{a,b\}^* | u$ has more than twice as many a's as b's$\}$. The following Turing machine T decides the language A. Some parts of the definition are left blank and you will be asked to fill in these parts.

   $T =$"On input w

   1. Repeat as long as there are __Ⓐ__ left on the tape.
   2. Scan the tape and cross off __Ⓑ__.
   3. If there are __Ⓒ__ left on the tape *accept*, else *reject*."

   (a) Fill in each of the blanks in the definition of T, in such a way that T decides A in time $O(n^2)$.

   Ⓐ At least two a's and one b

   Ⓑ Two a's and one b

   Ⓒ At least one a and no b's

(b) Give an implementation-level description of a multi-tape Turing machine that decides A in time $O(n)$. (You do not need to explain why the running time is $O(n)$.)

Answer:

M = "On input w

1. Scan tape 1. For each a read, write one a on tape 2 and for each b read, write two b's on tape 3.
2. Return to the first cells on tapes 2 and 3.
3. Scan tapes 2 and 3. If a blank is reached on both tapes at the same time, *accept*, else *reject*."

[10 points]

9. A path in an undirected graph is called *simple* if there are no repeated vertices on the path. Let $LONGEST - PATH = \{\langle G, s, t, k\rangle | G$ is an undirected graph, s and t are vertices of G and there is a simple path from s to t of length $\geq k\}$.

   Show that $LONGEST - PATH$ is in NP.

   Answer:

   $LONGEST - PATH$ is decided by the following nondeterministic Turing machine:

   $N =$"On input $\langle G, s, t, k\rangle$ where G is an undirected graph, s and t are vertices of G and k is a number:

   1. Let G have m vertices.
   2. If $k \geq m$, *reject*.
   3. Guess a sequence $p_1, p_2, \ldots p_n$ of vertices of G, where $k < n \leq m$.
   4. Check that there are no repeated vertices in the list.
   5. Check that $p_1 = s$ and $p_n = t$.
   6. Check that for each i, $1 \leq i < n$, (p_i, p_{i+1}) is an edge of G.
   7. If all checks are passed, *accept*, else *reject*."

   Each step is executed once. In Step 6, checking each pair (p_i, p_{i+1}) requires at most examining each edge of the graph once, and there are no more than $m - 1$ pairs to check, so this step can be implemented in polynomial time, as can all the others, so N runs in polynomial time. [6 points]

10. In class, we gave an algorithm to convert a non-deterministic finite automaton N into a deterministic finite automaton M. (This is the same algorithm as the one in the proof of Theorem 1.39.)

    (a) Does this algorithm run in polynomial time? (Assume that the algorithm has to produce the whole DFA M defined in the algorithm, not just the reachable part.)

    Yes _____
    No X

    (b) Explain your answer in (a).

    Answer:

    If N has m states, then the full DFA M has 2^m states, so M cannot be produced in polynomial time from N.

[6 points]

11. Prove that if the language $\{0^k1^k|k \geq 0\}$ is NP complete, then $P = NP$. [You may use results in the book without proof.]

Answer:

Let $L = \{0^k1^k|k \geq 0\}$ and suppose that L is NP complete. Since L is a context-free language, by Theorem 7.16, L is in P. But then, by Theorem 7.35, $P = NP$.

[4 points]

12. Let $S = \{f : \mathcal{N} \to \mathcal{N} |$ for all N, $f(n)$ diivides $f(n+1)$ evenly$\}$. ("$f(n)$ divides $f(n+1)$ evenly" means that for some natural number k, $f(n+1) = kf(n)$. The number k can be different for different n.)

Use diagonalization to show that S is an uncountable set.

Answer:

Let $f_1, f_2, f_3, \ldots$ be a sequence of elements of S. Define a function d by

$d(1) = f_1(1) + 1$

and

$$d(n+1) = \begin{cases} d(n) & \text{if } d(n) \neq f_{n+1}(n+1) \\ 2d(n) & \text{if } d(n) = f_{n+1}(n+1) \end{cases}$$

Then for all n, $d(n+1)$ is either $d(n)$ or $2d(n)$, so $d(n)$ divides $d(n+1)$ evenly for all n, which means that d is in S.

Since we do not consider 0 to be a natural number, $2d(n) \neq d(n)$ no matter what natural number $d(n)$ is, so $d(n+1) \neq f_{n+1}(n+1)$ for all n, and $d(1) \neq f_1(1)$, so d is different from all the f_n's. This proves that S is uncountable. [4 points]