Academic Year:	2022/2023	Semester:	Fall
School:	Information Technology	Course Code:	CSAI 335
Date:	January 22nd, 2023	Course Title:	Theory of Computing
Time:	3 hours	Full Mark:	100

Final Exam (MODEL ANSWERS)

Answer ALL Questions. You can have three A4-sized cheat sheets.

Question A: True/False Statements [10 marks]

State whether the following statements are **True** or **False** (Don't correct the false ones).

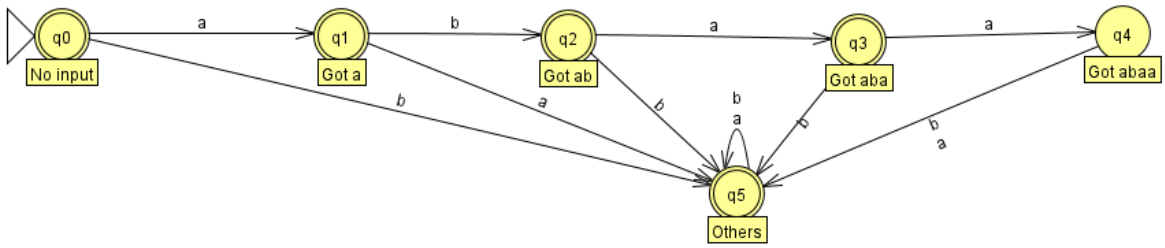
	Statement	T/F?
A.1	A finite automaton (FA) can have exactly one accept state.	**F**
A.2	A deterministic FA can have exactly one transition exiting every state for each possible input symbol.	**T**
A.3	The empty language contains the empty string.	**F**
A.4	If the start state of a FA is an accept state, the empty string ε is not accepted.	**F**
A.5	The intersection operation is a binary operation over regular languages.	**T**
A.6	Every context-free language is regular.	**F**
A.7	A string w is derived ambiguously in context-free grammar G if it has a unique rightmost derivation.	**F**

A.8	The pumping lemmas for the regular and context-free languages are based on the pigeonhole principle.	**T**
A.9	The language $\{0^i 1^j \mid 0 \leq i \leq j\}$ is non-regular.	**T**
A.10	The CFG variables are the symbols that appear on the left-hand side of the rules and the terminals are the remaining symbols.	**T**
A.11	In a Turing machine, the tape alphabet contains the input alphabet.	**T**
A.12	A non-deterministic single-tape Turing machine is more powerful than a deterministic one.	**F**
A.13	Any Turing-recognizable language is decidable.	**F**
A.14	The set of negative rational numbers is countable.	**T**
A.15	There are more Turing machines than languages.	**F**
A.16	The Church-Turing Thesis indicates that computability is dependent on the model of computation.	**F**
A.17	NP is the set of languages where we can test membership quickly.	**F**
A.18	The CLIQUE problem is NP-complete.	**T**
A.19	A problem that is polynomial-time reducible to a problem in *P* is not in *P*.	**F**
A.20	A problem with a polynomial-time algorithm has a polynomial-space algorithm.	**T**

B. Regular Languages (35 marks)

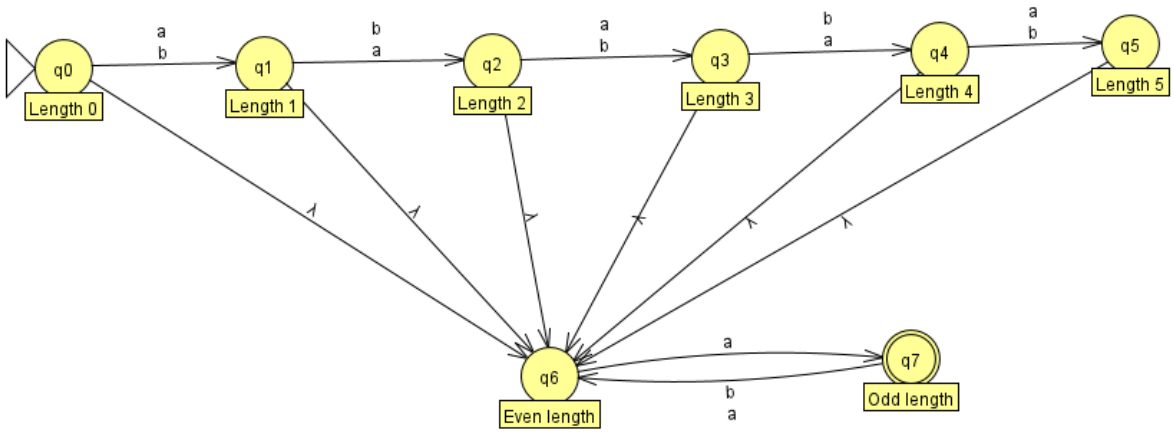
B.1 Draw a deterministic finite automaton (DFA) for a password verification module that rejects the password string '$aaba$' and accepts all other strings over the alphabet $\{a, b\}^*$.

Answer:

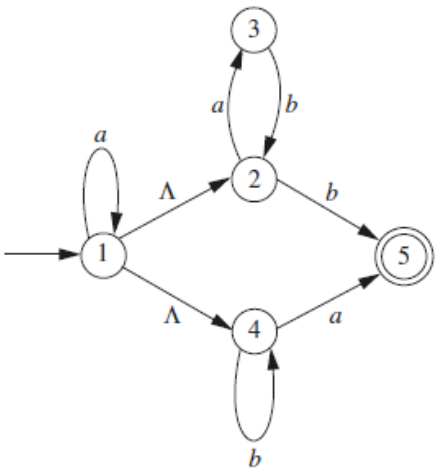


B.2 Give the state diagram of a nondeterministic finite automaton (NFA) recognizing the concatenation of the languages $L_1 = \{w | w \in \{a, b\}^*$ and the length of w is at most $5\}$ and $L_2 = \{w | w \in \{a, b\}^*$ and every odd position of w is an $a\}$.

Answer:

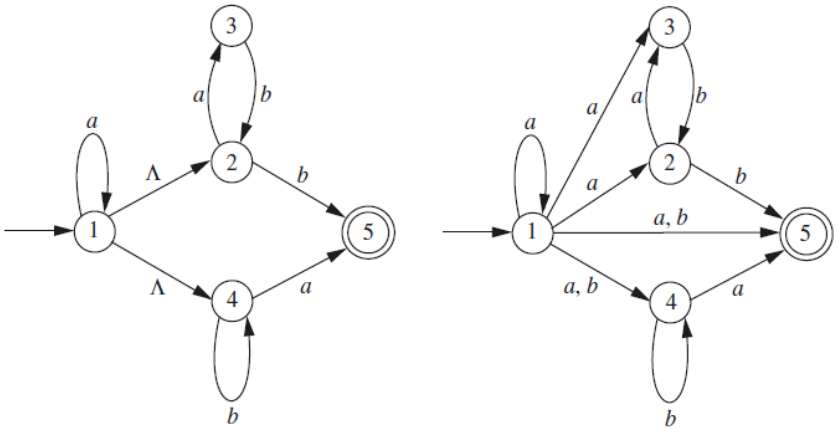


B.3 Convert the following nondeterministic finite automaton (NFA) to an equivalent deterministic finite automaton (DFA) (*Hint:* Build the NFA transition table to define the minimum number of DFA states). The symbol Λ denotes the empty string.



Answer:

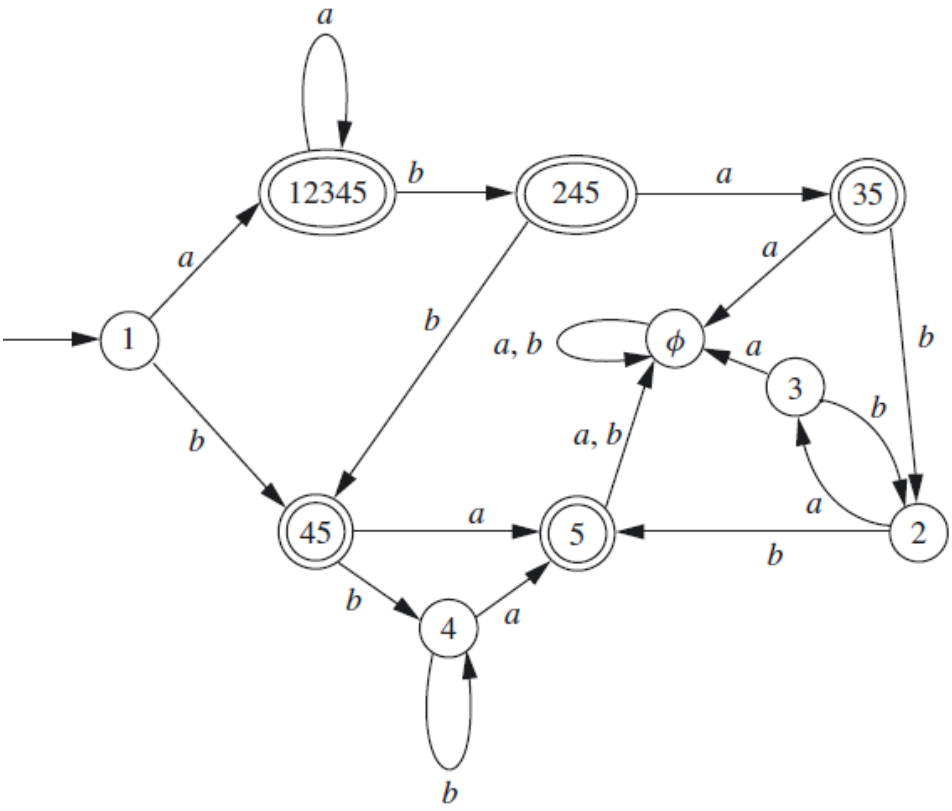
First, remove the Λ transitions:



Second, construct the state transition table:

q	$\delta(q, a)$	$\delta(q, b)$	$\delta(q, \Lambda)$	$\delta^*(q, a)$	$\delta^*(q, b)$
1	{1}	Ø	{2, 4}	{1, 2, 3, 4, 5}	{4, 5}
2	{3}	{5}	Ø	{3}	{5}
3	Ø	{2}	Ø	Ø	{2}
4	{5}	{4}	Ø	{5}	{4}
5	Ø	Ø	Ø	Ø	Ø

Finally, create the DFA diagram:



B.4 Give a nondeterministic finite automaton (NFA) recognizing the language generated by the regular expression $(((00)^*(11))\cup 01)^*$.

Sketch of Answer:

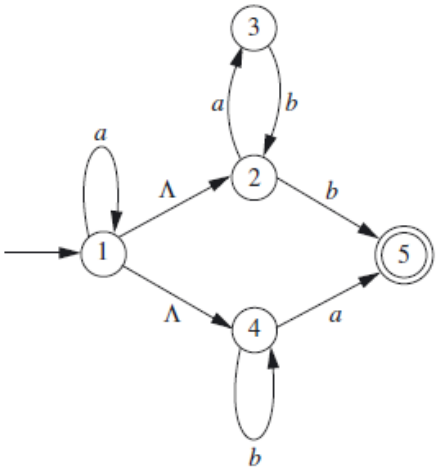
Build the NFA following Examples 1.56 (page 68) and 1.58 (page 69) in the textbook (Sipser 2013).

B.5 Give a regular expression generating the language of strings in $\{a, b\}^*$ ending with b and every a is followed immediately by bb.

Answer:

$$(abb \cup b)^*(b \cup abb) = (abb \cup b)^+.$$

B.6 Convert the following finite automata to a regular expression:



Answer (Fill in the details for the conversion process):

a*((ab)*b ∪b*a)

B.7 Show that the language $L = \{w: w \in \{a,b\}^*, n_a(w) = n_b(w)\}$ is not a regular language (Hint: Provide a proof by contradiction based on the string $s = a^p b^p$ where p denotes the pumping length).

Answer:

Assume that *L* is a regular language and obtain a contradiction. Let *p* be the pumping length given by the pumping lemma. We use the suggested string $s = a^p b^p$. Based on the pumping lemma, the string can be written as $s = xyz$, where

1. **for each $i \geq 0, xy^i z \in L$,**
2. **$|y| > 0$, and**
3. **$|xy| \leq p$.**

We assume that Conditions 2 and 3 are satisfied and show that Condition 1 couldn't be satisfied. Since $|xy| \leq p$, both x, y contain only the 'a' alphabet. In fact, a general expression for the string would be $s = xyz = (a^k)(a^l)(a^{p-(k+l)}b^p), |y| = l > 0, |xy| = k + l \leq p$. Thus, $xy^0z = (a^k)(a^{p-(k+l)}b^p) = a^{p-l}b^p, p - l \leq p$. Clearly, xy^0z doesn't have the structure of the strings in L, and thus s can't be a member of L. That violates Condition 1 of the lemma and is thus a contradiction. So, the assumption that *L* is regular must be false. Hence, we have proved that *L* is not regular.

C. Context-Free Languages (20 marks)

C.1 For this set of context-free grammar (CFG) rules, specify the following:

$S \rightarrow S_1 \mid S_2$
$S_1 \rightarrow \text{if } (E)\ S_1 \text{ else } S_1 \mid OS$
$S_2 \rightarrow \text{if } (E)\ S \mid \text{if } (E)\ S_1 \text{ else } S_2$

- the variables, ***{S, S_1, S_2}***
- the start variable, ***S***
- the terminals, ***{if (E), else, OS}***
- one nonacceptable string, and ***if (E)***
- one acceptable string (with the corresponding parse tree). ***S → S_1 → OS***

In our notation, *E* is short for <expression>, *S* for <statement>, and *OS* for <otherstatement>.

C.2 Give a context-free grammar (CFG) generating the set of even-length strings over the alphabet $\{a, b\}$ whose first and last symbols are all the same.

Answer:

Rule	**Explanation**
S → A|B|ε	**Create two types of 'a'-bounded or 'b'-bounded strings or none at all**
A → aTa	**Strings with '*a*' at the first and last positions**
B → bRb	**Strings with '*b*' at the first and last positions**
T → aTa | aTb | bTa | bTb | ε	**For strings with '*a*' at the first and last positions, add any arrangement of 'a' and 'b' in pairs. The last inserted symbol**

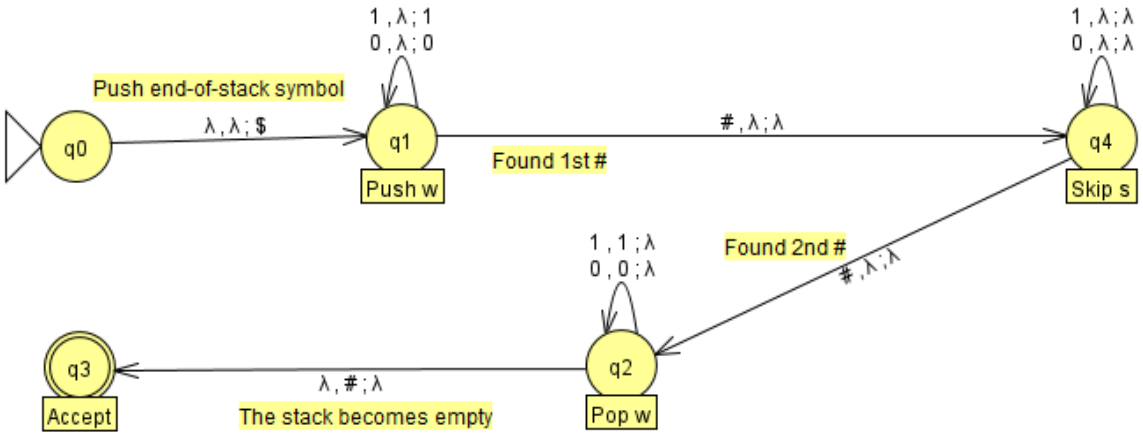
	should be 'ε' to ensure the string length is even.
R → aRa | aRb | bRa | bRb | ε	**For strings with 'b' at the first and last positions, add any arrangement of 'a' and 'b' in pairs. The last inserted symbol should be 'ε' to ensure the string length is even.**

C.3 For the following set of CFG rules, construct an equivalent push-down automata (PDA).

$$S \rightarrow \varepsilon \mid aB \mid bA$$
$$A \rightarrow aS \mid bAA$$
$$B \rightarrow bS \mid aBB$$

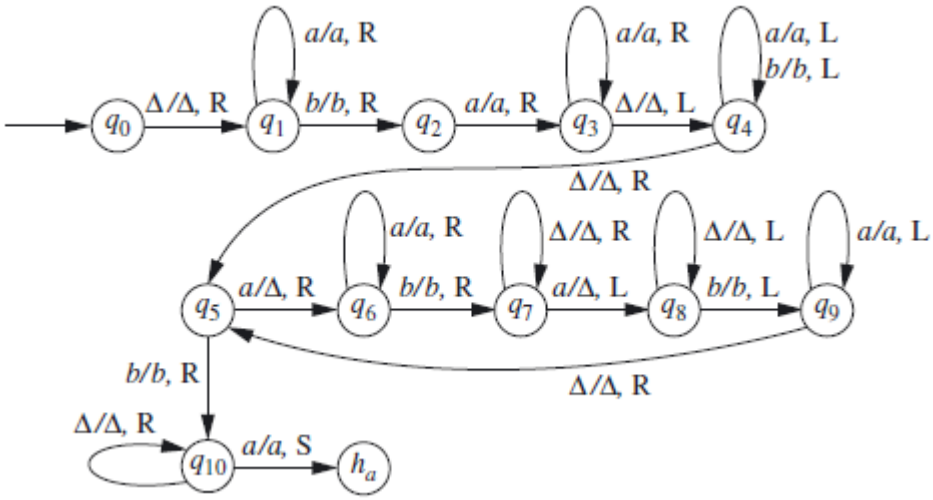
C.4 For a string w, let w^R denote the string written backwards. Construct a push-down automata (PDA) for the language $\{w\#s\#w^R : w, s \in \{a, b\}^*\}$.

Answer:



D. Turing Machines (15 marks)

D.1 Here is the state diagram for a Turing machine M accepting $L = \{a^i b a^j \mid 0 \leq i < j\}$, where Δ denotes the empty string.



Give the sequence of configurations that M enters when started on the input string $abaa$. Describe what this machine does in plain English.

Answer:

Here is a trace of the computation for the input string *abaa*:

$$
\begin{aligned}
q_0\Delta abaa &\vdash \Delta q_1 abaa \vdash \Delta a q_1 baa \vdash \Delta ab q_2 aa\Delta \vdash \Delta aba q_3 a \\
&\vdash \Delta abaa q_3\Delta \vdash \Delta aba q_4 a \vdash^* q_4\Delta abaa \vdash \Delta q_5 abaa \\
&\vdash \Delta\Delta q_6 baa \vdash \Delta\Delta b q_7 aa \vdash \Delta\Delta q_8 b\Delta a \vdash \Delta q_9 \Delta b\Delta a \\
&\vdash \Delta\Delta q_5 b\Delta a \vdash \Delta\Delta b q_{10}\Delta a \vdash \Delta\Delta b\Delta q_{10} a \vdash \Delta\Delta b h_a a \quad \text{(accept)}
\end{aligned}
$$

The machine makes a preliminary pass through the entire string to make sure that it has the form *a*baa. If it doesn't, it will be rejected; if it does, this TM also carries out a sequence of iterations in which an *a* preceding the *b* and another one after the *b* are erased, but the *a* it erases in each portion will be the leftmost *a*.**

D.2 Show that a 3-tape deterministic Turing machine is equivalent to a 2-tape deterministic Turing machine.

Sketch of answer:

Follow the line of the proof of Theorem 3.13 (p. 177, Sipser 2013) to show the that each machine can be simulated by the other.

E. Decidability and Undecidability (10 marks)

E.1 Let EQ_{3DFA} = {$< A, B, C >$| A, B, and C are DFAs and $L(A) = L(B) = L(C)$}. Show that EQ_{3DFA} is a decidable language (Hint: Construct a new DFA D from A, B, and C where D accepts only those strings that are accepted by either A, B, or C but not all of them).

Answer:

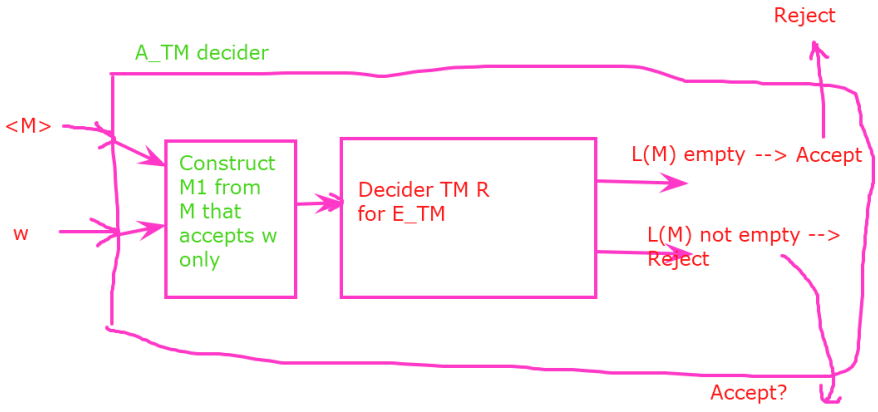
Give TM $D_{\text{EQ-3DFA}}$ that decides EQ_{3DFA}.

$D_{\text{EQ-3DFA}}$ = "On input $\langle A, B, C\rangle$ [IDEA: Make DFA D that accepts w where A and B disagree, A and C disagree, or B and C disagree.]

1. **Construct DFA D where $L(D) = \left(\left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)\right) \cup \left(\left(L(A) \cap \overline{L(C)}\right) \cup \left(\overline{L(A)} \cap L(C)\right)\right) \cup \left(\left(L(B) \cap \overline{L(C)}\right) \cup \left(\overline{L(B)} \cap L(C)\right)\right)$.**
2. **Run $D_{\text{E-DFA}}$ on $\langle D\rangle$.**
3. ***Accept* if $D_{\text{E-DFA}}$ accepts.**
   ***Reject* if $D_{\text{E-DFA}}$ rejects."**

E.2 Show that E_{TM} = {$<M>$ | M is a TM and $L(M)$ = ∅}, the emptiness problem of a Turing machine, is undecidable (Hint: Show that A_{TM}, the acceptance problem of a Turing machine, is reducible to E_{TM}).

Answer:



F. Complexity (10 marks)

F.1 Let G represent an undirected graph. Also, let $SPATH$ = {$< G, a, b, k >|$ G contains a simple path of length at most k from a to b}, where a simple path is a path that doesn't repeat any nodes. Show that $SPATH \in P$.

Answer:

Starting from a, we check at most n-1 unvisited neighbors for b. If b isn't found, we check at most n-2 unvisited neighbors of neighbors for b. We continue this for at most k steps where in the last step we check at most n-k unvisited neighbors for b. This takes a complexity of O(n.(n-1).(n-2)….(n-k)) = O(n^k).

F.2 A triangle in an undirected graph is a 3-clique. Show that $TRIANGLE \in PSPACE$ (It has a polynomial-space algorithm), where $TRIANGLE$ = {$< G >|$ G contains a triangle}.

Answer:

Each triple of nodes can be examined to find if they form a triangle in polynomial space (and the same space can be used for all triples).

Alternatively, we construct a TM *M* that decides *TRIANGLE* in polynomial time, and thus conclude that it takes polynomial space (space can be reused while time can't).

***M* = "On input <*G*> where *G* is a graph:**
1. For each triple of nodes *v*1, *v*2, *v*3 in *G*:
2. If (*v*1, *v*2), (*v*1, *v*3), and (*v*2, *v*3), are edges of *G*, *accept*.
3. No triangle has been found in *G*, so *reject*."

Graph *G* has at most *n* nodes so it has at most

$$\binom{m}{3} = \frac{m!}{3!(m-3)!} = O(m^3)$$

triples of nodes. Hence stage 2 will repeat at most *O*(*m*3) times. Each stage runs in polynomial time. Therefore, *TRIANGLE* ∈ P, and thus *TRIANGLE* ∈ PSPACE.