Computer Science Theory Final Exam

COMS W3261, Summer B 2021

This exam will be available from 12:01 AM EST on Tuesday, 8/10/2021 until Wednesday, 8/11/2021, at 11:59PM EST. You have **12 hours** to complete this exam from the time you download it; you can use any 12 hours in the 48-hour window to work on the exam. The exam is due **Wednesday, 8/11/2021, at 11:59PM EST with NO EXTENSIONS**. Submit to GradeScope (course code: X3JEX4).

Grading policy reminder: LaTeX is preferred, but neatly typed or handwritten solutions are acceptable. Your TAs may dock points for indecipherable writing. Proofs should be complete; that is, include enough information that a reader can clearly tell that the argument is rigorous.

If a question is ambiguous, please state your assumptions. This way, we can give you credit for correct work. (Even better, post on Ed so that we can resolve the ambiguity.)

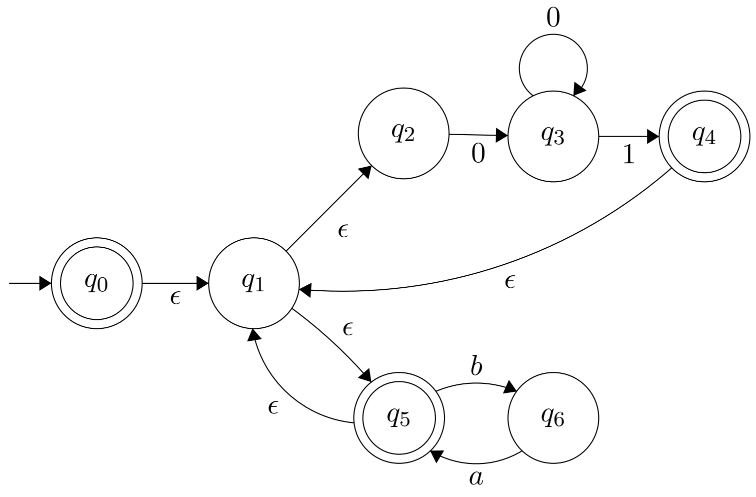
You can consult the following resources during the exam:

1. Your notes and class notes posted on the course webpage.
2. Lecture recordings on YouTube and in-class recordings in the Zoom recordings folder on CourseWorks.
3. Past homework problems, your homework answers, and solutions posted on the course webpage.
4. The textbook.
5. Past questions posted on Ed. (You may post on Ed during the exam, but please do not answer other student's questions or provide any details that might help others solve a problem. Answers from instructors will be limited to basic typo correction and clarifications only.)
6. Resources for LaTeX formatting that are unrelated to the course material.

Any resources not listed here, including all reference sources beyond the textbook, are not permitted.

1 Problem 1 (10 points)

For this problem, consider the NFA state diagram pictured below.



1. (3 points.) Does this NFA accept the empty string ε? What about the string 00101? What about the string ba0011?

   This NFA accepts the empty string ε and the string 00101. It does not accept the string ba0011.

2. (2 points.) In class, we showed that the class of regular languages was closed under Kleene star (*) by demonstrating that, given an NFA N that recognizes any language A, we can convert N into a new NFA N' that recognizes A^*. The NFA pictured above was created by this conversion process. Which transition(s) or state(s) may have been added during conversion?

   To convert N into N', we:

   (a) Add ε-edges from each accept state to the start state.

   (b) Add a new start/accept state with a ε-edge to the old start/accept state.

   This corresponds to the ε edges from q_4 and q_5 to q_1, the old start state. We have also added the new start state q_0 and the ε-edge from q_0 to q_1.

3. (5 points.) Write a regular expression that evaluates to the same language that the NFA recognizes. Explain in a few sentences why the expression and the NFA represent the same language. (Hint: begin by reverse-engineering the conversion process for *.)

   If we remove the ε-edges and state q_0 corresponding to the * modification, we are left with an NFA that accepts a string if (1) a branch of computation takes the ε-edge from q_1 to q_2 and is accepted by the component with states q_2, q_3, and q_4 or (2) a branch of computation takes the ε-edge from q_1 to q_5 and is accepted by the component with states q_5 and q_6.

   In other words, the language recognized by the NFA is the star (*) of the union ($\cup$) of the languages recognizes by these two components. The top component recognizes $00^*1 = 0^+1$

and the bottom component recognizes ba^*. Thus the regular expression is $(0^+1 \cup (ba)^*)^*$. (Equivalent expressions include $(00^*1 \cup (ba)^*)^*$ and $(0^+1 \cup ba)^*$.)

2 Problem 2 (14 points)

For this problem, consider the language

$$L_1 = \{0^i 1^{i+j} 0^j \mid i, j \geq 0\}.$$

1. (7 points.) Use the pumping lemma for regular languages to prove that L is nonregular.

   Assume for contradiction that L_1 is a regular language. By this assumption L_1 satisfies the pumping lemma and there exists some integer p such that any string $s \in L_1$ with $|s| \geq p$ satisfies the following three properties:

   (a) $xy^k z \in L_1$ for any $k \geq 0$,

   (b) $|y| \geq 0$, and

   (c) $|xy| \leq p$.

   Consider the string $s = 0^p 1^{2p} 0^p$, which is in the language and has length $4p \geq p$. We will show that no division of s into substrings x, y, and z satisfies the conditions of the pumping lemma.

   Consider any division that satisfies the conditions $|xy| \leq p$ and $|y| > 0$. This implies that the string y is a nonempty substring of the first substring 0^p. Thus $xy^2 z = 0^{p+|y|} 1^{2p} 0^p \notin L_1$, as $p + |y| + p \neq 2p$. Thus L_1 does not satisfy the pumping lemma and our assumption must be false. L_1 is nonregular.

2. (7 points.) Prove that L is context-free by creating a context-free grammar that generates L. Explain in a few sentences why your CFG generates exactly the strings in L.

   Consider the following context-free grammar G:

$$\begin{aligned} S &\to AB \\ A &\to 0A1 \mid \varepsilon \\ B &\to 1B0 \mid \varepsilon \end{aligned}$$

   Any derivation that begins with the start symbol S first yields the string AB. To complete the derivation, we must then use the first rule for A some number $i \geq 0$ times before replacing A with ε and use the first rule for B some number $j \geq 0$ times before replacing B with ε. This produces the string $0^i 1^{i+j} 0^j$. (There are several other equivalent grammars that might work.)

3 Problem 3 (8 points)

Recall that the XOR operator $\oplus$ evaluates to 1 if exactly one of its inputs is 1: thus $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$. Given two binary strings $w, v \in \{0,1\}^*$ that have the same length, the binary XOR of w and v, written $w \oplus v$, is the string x such that $w_i \oplus v_i = x_i$ for all i. For example,

$$01001 \oplus 11000 = 10001.$$

Consider the language

$$L_2 = \{a\#b\#c \mid a, b, c \in \{0,1\}^* \text{ and } a \oplus b = c\}.$$

1. (8 points.) Prove that L_2 is decidable by writing an implementation-level description of a Turing machine that decides L_2. (Don't forget to check that your input is correctly formatted.)

   We can decide L_2 using a Turing machine M that computes as follows. M will first check to ensure that the input consists of three equal-length binary strings separated by $\#$ symbols, then check that the XOR relation holds.

   $M =$ "On input w :

   1. Scan the input string from left to right and reject unless we see exactly two $\#$ symbols. Accept if a, b, and c are all empty. Return the head to the leftmost tape square.
   2. Shuttle back and forth between the strings a and b, crossing off one symbol from each string in turn and using marked symbols to track our place. Reject unless $|a| = |b|$. Uncross all crossed symbols and return to the beginning of b.
   3. Shuttle back and forth between b and c to check length as in the previous step.
   4. We now know that $|a| = |b| = |c|$. Shuttle from a_1 to b_1 to c_1, crossing out each symbol and rejecting unless $a_1 \oplus b_1 = c_1$. Use dot markers to track our place. Repeat this for a_2, b_2, c_2 etc. until all symbols are crossed off. Accept if we have not yet rejected."

   Other solutions, such as storing the XOR of a and b on a second tape and checking this against c, or merging the length-checking and XOR-checking steps, should be accepted if implemented correctly.

4 Problem 4 (8 points)

1. (4 points.) Prove that the language

$$L_3 = \{a \mid a \text{ is a prime number that is also the sum of two square numbers}\}$$

is decidable by writing a high-level description of a Turing machine that decides L_3.

First, our TM determines if a is prime by checking all numbers smaller than a and rejecting if any divides a. Second, our TM determines if a is the sum of two squares by checking if $b^2 + c^2 = a$ for every pair of integers (b, c) such that $b^2, c^2 \leq a$. Accept if a is prime and the sum of two squares and reject if not.

2. (4 points.) Prove that the language

$$L_4 = \{\langle A \rangle \mid A \text{ is a Turing machine that halts on } \textit{any} \text{ input string } w \in \Sigma^*\}$$

is recognizable by writing a high-level description of a Turing machine that recognizes L_4.

The following is a high-level description of a Turing machine M_4 that decides L_4. Say that $s_1, s_2, s_3, \ldots$ is a list of all possible strings in Σ^*.

```
$M_4$ = "On input $\langle A \rangle$, where $A$ is a TM:
  1. For each integer $i = 1, 2, \ldots$, simulate $A$ on $s_1, s_2, \ldots, s_i$ for $i$ steps.
     Accept if any simulation halts."
```

Note that at any iteration of our loop we simulate A on a finite number of strings for a finite number of steps, which makes the iteration finite. If we try to simulate A on all strings for any finite number of steps, or on one string for a (potentially infinite) number of steps, the recognizer may fail to recognize some TMs in L_4.

5 Problem 5 (10 points)

1. Prove that the language

$$L_5 = \{\langle A, B\rangle \mid A \text{ and } B \text{ are Turing machines and } L(A) \subseteq L(B)\}$$

is undecidable by reducing another undecidable language to L_5. You may not use Rice's theorem. (Hint: it suffices to show that *if* you could build a decider for L_5, you could build a decider for a language that is proved undecidable in lecture, in homework or in the textbook.)

Recall the language

$$E_{TM} = \{\langle A\rangle \mid A \text{ is a TM and } L(A) = \emptyset\},$$

which we proved undecidable in lecture. We'll show that if some decider S exists for the language L_5, we could decide E_{TM}. This is a contradiction and thus it follows that L_5 is undecidable. Assuming the existence of S, we can build the following TM for E_{TM}.

```
$T =$ "On input $\langle A\rangle$, where $A$ is a TM:
  1. Let $B$ be a TM that rejects on all inputs.
  2. Run $S$ on $\langle A, B\rangle$ and accept/reject if $S$ accepts/rejects."
```

To see that T decides E_{TM}, first observe that T always halts (this follows from the fact that S, our hypothetical decider for L_5, always halts.)

If $L(A) = \emptyset$, then $L(A) \subseteq L(B)$ as $\emptyset \subseteq \emptyset$, S accepts, and thus T accepts. If $L(A) \neq \emptyset$, then $L(A) \not\subseteq L(B)$, S rejects, and thus T rejects. Thus we correctly decide E_{TM}.

6 Problem 6 (3 extra credit points)

1. (3 points). Consider the following language:

$$\overline{SELF_{ENUM}} = \{\langle E\rangle \mid E \text{ is an enumerator that never prints out its own description } \langle E\rangle\}.$$

   Recall that a language is Turing-recognizable if and only if some enumerator enumerates it. Show that $\overline{SELF_{ENUM}}$ is not Turing-recognizable by finding a paradox.

   Assume for contradiction that F is an enumerator for $\overline{SELF_{ENUM}}$. Thus F prints out the description of all enumerators that don't enumerate themselves.

   Does F enumerate itself? If it does, then $F \notin \overline{SELF_{ENUM}}$ so this is incorrect. If it does not, then $F \in \overline{SELF_{ENUM}}$ so this is incorrect. Thus the existence of F creates a paradox.