

CS 273: Intro to Theory of Computation
Spring 2007 Final Exam, May 5th, 2007

INSTRUCTIONS (read carefully)

- Print your name and netID here and netID at the top of each other page.

NAME:

NETID:

- The exam contains 12 pages and 11 problems. Make sure you have a complete exam.
- You have three hours.
- The point value of each problem is indicated next to the problem and in the table below.
- It is wise to skim all problems and point values first, to best plan your time. If you get stuck on a problem, move on and come back to it later.
- Points may be deducted for solutions which are correct but excessively complicated, hard to understand, hard to read, or poorly explained.
- This is a closed book exam. No notes of any kind are allowed. Do all work in the space provided, using the backs of sheets if necessary. See the proctor if you need more paper.
- Please bring apparent bugs or unclear questions to the attention of the proctors.

Problem	Possible	Score
1	10	
2	20	
3	8	
4	8	
5	8	
6	6	
7	10	
8	8	
9	8	
10	6	
11	8	
Total	100	

Problem 1: True/False (10 points)

Completely write out “True” if the statement is necessarily true. Otherwise, completely write “False”. Other answers (e.g. “T”) will receive credit only if your intent is unambiguous. For example, “ $x + y > x$ ” has answer “False” assuming that y could be 0 or negative. But “If x and y are natural numbers, then $x + y \geq x$ ” has answer “True”. You do not need to explain or prove your answers.

1. Let M be a DFA with n states such that $L(M)$ is infinite. Then $L(M)$ contains a string of length at most $2n - 1$.

Solution: True.

2. Let $L_w = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$, where w is some fixed string. Then there is an enumerator for L_w .

Solution: True.

3. OMITTED.

4. The set of undecidable languages is countable.

Solution: False.

5. For a TM M and a string w , let $CH_{M,w} = \{x \mid x \text{ is an accepting computation history for } M \text{ on } w\}$. Then $CH_{M,w}$ is decidable.

Solution: True.

6. The language $\{ \langle M, w \rangle \mid M \text{ is a linear bounded automaton and } M \text{ accepts } w \}$ is undecidable.

Solution: False.

7. The language $\{ \langle G \rangle \mid G \text{ is a context-free grammar and } G \text{ is ambiguous} \}$ is Turing-recognizable.

Solution: True.

8. Context-free languages are closed under homomorphism.

Solution: True.

9. The modified Post’s correspondence problem is Turing recognizable.

Solution: True.

10. There is a bijection between the set of Turing-recognizable languages and the set of decidable languages.

Solution: True.

Problem 2: Classification (20 points)

For each language L described below, classify L as

- **R**: Any language satisfying the information must be regular.
- **C**: Any language satisfying the information must be context-free, but not all languages satisfying the information are regular.
- **DEC**: Any language satisfying the information must be decidable, but not all languages satisfying the information are context-free.
- **NONDEC**: Not all languages satisfying the information are decidable. (Some might be only Turing recognizable or perhaps even not Turing recognizable.)

For each language, circle the appropriate choice (**R**, **C**, **DEC**, or **NONDEC**). If you change your answer be sure to erase well or otherwise make your final choice clear. **Ambiguously marked answers will receive no credit.**

1. **Omitted.**

2. **R C DEC NONDEC**

$L = \{ \langle M \rangle \mid M \text{ is a linear bounded automaton and } L(M) = \emptyset \}$.

Solution: NONDEC

3. **R C DEC NONDEC**

$L = \{ ww^R w \mid w \in \{a, b\}^* \}$.

Solution: DEC

4. **R C DEC NONDEC**

$L = \{ w \mid \text{the string } w \text{ occurs on some web page indexed by Google on May 3, 2007} \}$

Solution: R

5. **R C DEC NONDEC**

$L = \{ w \mid w = x \# x_1 \# x_2 \# \dots \# x_n \text{ such that } n \geq 1 \text{ and there is some } i \text{ for which } x \neq x_i \}$.

Solution: C

6. **R C DEC NONDEC**

$L = \{ a^i b^j \mid i + j = 27 \pmod{273} \}$

Solution: R

7. **R C DEC NONDEC**

$L = L_1 \cap L_2$ where L_1 and L_2 are context-free languages

Solution: DEC

8. **R C DEC NONDEC**

$L = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is finite} \}$

Solution: UNDEC

9. **R C DEC NONDEC**

$L = L_1 - L_2$ where L_1 is context-free and L_2 is regular.

Solution: C

10. **R C DEC NONDEC**

$L = L_1 \cap L_2$ where L_1 is regular and L_2 is an arbitrary language.

Solution: NONDEC

Problem 3: Short answer I (8 points)

(a) Give a regular expression for the set of all strings in $\{0, 1\}^*$ that contain at most one pair of consecutive 1's.

Solution:

First, a regular expression for the set of strings that have no consecutive 1's is:

$$R = 0^*. (10.0^*)^*. (\epsilon + 1).$$

The regular expression for the set of strings that contain at most one pair of consecutive 1's is just:

$$R.R = 0^*. (10.0^*)^*. (\epsilon + 1). 0^*. (10.0^*)^*. (\epsilon + 1)$$

since a string w has at most one pair of consecutive 1's iff $w = xy$, for some x, y , where x and y have no pair of consecutive ones.

There are, of course, many solutions to this question; whatever your solution is, be careful to check border cases. For example, check if you allow initial 0's, final 0's, allow 1 to occur in the beginning and the end, allow ϵ , and also test your expression against some random strings, some in the language and some outside it.

(b) Let M be a DFA. Sketch an algorithm for determining whether $L(M) = \Sigma^*$. Do not generate all strings (up to some bound on length) and feed them one-by-one to the DFA. Your algorithm must manipulate the DFA's state diagram.

Solution: If M does not accept all of Σ^* , then there must be a word $w \notin L(M)$. On word w , M would reach a unique state which is non-final. Also, if there is some way to reach a non-final state from the initial state, then clearly the DFA does not accept the word that labels the path to the non-final state. Hence it is easy to see that the DFA M does not accept Σ^* iff there is a path from the initial state to a non-final state (which can be the initial state itself).

The algorithm for detecting whether $L(M) = \Sigma^*$ proceeds as follows:

1. Check whether the input is a valid DFA (in particular, make sure it is complete; i.e. from any state, on any input, there is a transition to some other state).
2. Consider the transition graph of the DFA.
3. Do a depth-first search on this graph, searching for a state that is not final.
4. If any non-final state is reached on this search, report that M does not accept Σ^* ; otherwise report that M accepts Σ^* .

An alternate solution will be to flip the final states to non-final and non-final states to final (i.e. complementing the DFA), and then checking the emptiness of the resulting automaton by searching for a reachable final state from the initial state.

Problem 4: Short answer II (8 points)

(a) Let $\Sigma = \{a, b\}$, let $G = (V, \Sigma, R, S)$ be a CFG, and let $L = L(G)$. Give a grammar G' for the language $L' = \{wxw^R \mid w \in \Sigma^*, x \in L\}$ by modifying G appropriately.

Solution: The grammar $G' = (V', \Sigma, R', S')$ where

- $V' = V \cup \{S'\}$ where S' is a new variable, not in V ;
- R' consists of all rules in R as well as the following rules:
 - One rule $S' \rightarrow aS'a$, for each $a \in \Sigma$
 - The rule $S' \rightarrow S$

Intuitively, S' is the new start variable, and for any word $wxw^R \in L'$, S' generates it by first generating the w and w^R parts, and then calling S to generate x .

(b) Let $P = \left\{ \begin{bmatrix} a \\ c \end{bmatrix}, \begin{bmatrix} a \\ aa \end{bmatrix}, \begin{bmatrix} cba \\ b \end{bmatrix}, \right\}$ be an instance of the Post correspondence problem. Does P have a match? Show a match or explain why no match is possible.

Solution:

Yes, the PCP has a match: $\begin{bmatrix} a \\ aa \end{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix} \begin{bmatrix} cba \\ b \end{bmatrix}, \begin{bmatrix} a \\ aa \end{bmatrix}$

The top word and the bottom word are both $aacbaa$.

Problem 5: Pumping Lemma (8 points)

For each of the languages below you can apply the pumping lemma either to prove that the language is non-regular or to prove that the language is not context-free. Assuming that p is the pumping length, your goal is to give a candidate string for w_p that can be pumped appropriately to obtain a correct proof. You ONLY need to give the string and no further justification is necessary. Assume $\Sigma = \{a, b\}$ unless specified explicitly.

(a) $L = \{w \mid w \text{ is a palindrome}\}$. To show L is not regular using the pumping lemma

Solution: $w_p = a^p b b a^p$

(b) $L = \{w \mid w \text{ contains at least twice as many a's as b's}\}$. To show L is not regular using the pumping lemma

$$w_p = b^p a^{2p}$$

(c) OMITTED

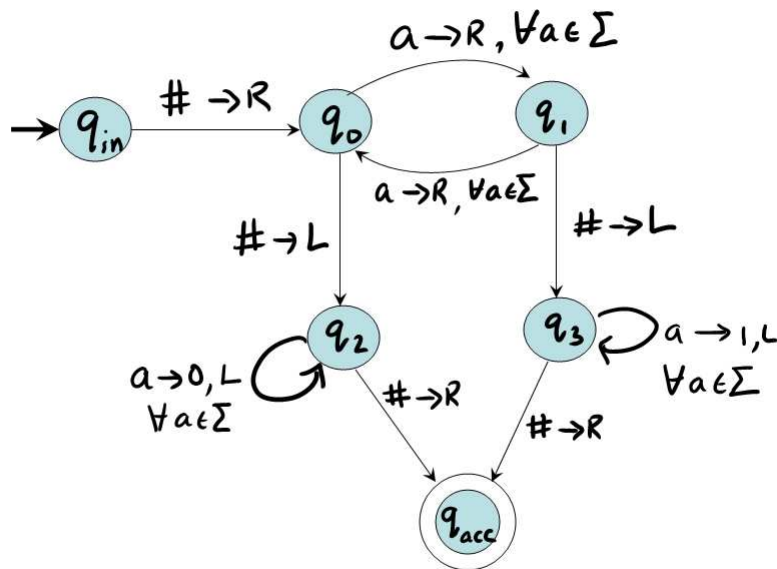
(d) OMITTED

Problem 6: TM design (6 points)

Give the state diagram of a TM M that does the following on input $\#w$ where $w \in \{0,1\}^*$. Let $n = |w|$. If n is even, then M converts $\#w$ to $\#0^n$. If n is odd, then M converts $\#w$ to $\#1^n$. Assume that ϵ is an even length string.

The TM should enter the accept state after the conversion. We don't care where you leave the head at the end of the conversion. The TM should enter the reject state if the input string is not in the right format. However, your state diagram does not need to explicitly show the reject state or the transitions into it.

Solution: The Turing machine's description is:



In the diagram, the initial state is q_{in} , and the accept state is q_{acc} ; the reject state is not shown, and we assume that all transitions from states that are not depicted go to the reject state. We are assuming that the blank tape-symbol is $\#$.

Intuitively, the TM first reads the tape content w , moving right, alternating between states q_0 and q_1 in order to determine whether $|w|$ is even or odd. If $|w|$ is even, it ends in state q_0 , moves left rewriting every letter in w with a 0, till it reaches the first symbol on the tape, and then accepts and halts. If $|w|$ is odd, it does the same except that it rewrites letters in w with 1's.

Problem 7: Subset construction (10 points)

Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA that does not contain any epsilon transitions. The *subset construction* can be used to construct a DFA $M = (Q', \Sigma, \delta', q'_0, F')$ recognizing the same language as N . Fill in the following key details of this construction, using your best mathematical notation:

Solution: $Q' = \mathcal{P}(Q)$

$$q'_0 = \{q_0\}$$

$$F' = \{R \subseteq Q \mid R \cap F \neq \emptyset\}$$

Suppose that P is a state in Q' and a is a character in Σ . Then

$$\delta'(P, a) = \{q \in Q \mid \exists p \in P, q \in \delta(p, a)\}$$

Suppose N has ϵ -transitions. How would your answer to the previous question change?

$$\delta'(P, a) = \{q \in Q \mid \exists p \in P, q \in E(\delta(p, a))\}$$

where

$$E(R) = \{s \mid s \text{ can be reached from some state in } R \text{ using zero or more } \epsilon\text{-edges}\}$$

is the epsilon-closure of the set R .

Problem 8: Writing a proof (8 points)

We have seen that $ALL_{CFG} = \{\langle G \rangle \mid G \text{ is a CFG and } L(G) = \Sigma^*\}$ is undecidable.

(a) Show that $\overline{ALL_{CFG}}$ is Turing-recognizable.

Solution:

$\overline{ALL_{CFG}} = \{\langle G \rangle \mid G \text{ is not the encoding of a CFG, or, } G \text{ is a CFG and } L(G) \neq \Sigma^*\}$.

First, recall that membership of a word in the language generated by a grammar is decidable, i.e., for any grammar G and any word w , we can build a TM that decides whether G generates w .

Also, for any Σ , we can fix some ordering of symbols in Σ , and enumerate all words in Σ^* in the lexicographic ordering. In particular, we can build a TM that can construct the i 'th word in Σ^* for any given i .

We can build a TM recognizing $\overline{ALL_{CFG}}$ as follows:

1. Input is $\langle G \rangle$.
2. Check if $\langle G \rangle$ is a proper encoding of a CFG; if not, halt and accept.
3. Set $i := 1$;
4. while (true) do {
5. Generate the i 'th word w_i in Σ^* , where Σ is the set of terminals in G .
6. Check if w_i is generated by G . If it is not, halt and accept.
7. Increment i ;
8. }

The TM above systematically generates all words in Σ^* and checks if there is any word that is not generated by G ; if it finds one it accepts $\langle G \rangle$.

Note that if $\langle G \rangle$ is either not a well-formed grammar or $L(G) \neq \Sigma^*$, then the TM will eventually halt and accept. If $L(G) = \Sigma^*$, the TM will never halt, and hence will never accept.

(b) Is ALL_{CFG} Turing-recognizable? Explain why or why not.

ALL_{CFG} is not Turing-recognizable. We know that ALL_{CFG} is not Turing-decidable (the universality problem for context-free grammars is undecidable), and we showed above that $\overline{ALL_{CFG}}$ is Turing-recognizable. Also, for any language R , if R and \overline{R} are Turing-recognizable, then R is Turing-decidable. Hence, if ALL_{CFG} was Turing-recognizable, then ALL_{CFG} would be Turing-decidable, which we know is not true. Hence ALL_{CFG} is not Turing-recognizable.

Problem 9: PDA modification (8 points)

Let L be a context-free language on the alphabet Σ . Let $M = (Q, \Sigma, \Gamma, \delta, q, F)$ be a PDA recognizing L . Give a PDA M' recognizing the language $L' = \{xy \mid x \in L \text{ and } y \in \Sigma^* \text{ and } |y| \text{ is even}\}$

Note: If you make assumptions about M , then state them clearly and justify.

(a) Explain the idea behind the construction.

Solution: The idea would be to construct the PDA M' which will essentially simulate M , and from any of the final states of M , nondeterministically jump on an ϵ -transition to a new state that checks whether the rest of the input is of even length.

(b) Give tuple notation for M' .

Solution: $M' = (Q', \Sigma, \Gamma, \delta', q, \{p_0\})$ where $Q' = Q \cup \{p_0, p_1\}$ where p_0 and p_1 are two new states (that are not in Q), and the transition function $\delta' : Q' \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow \mathcal{P}(Q' \times \Gamma_\epsilon)$ is defined as follows:

- For every $q \in Q$, $a \in \Sigma$, $d \in \Gamma_\epsilon$, $\delta'(q, a, d) = \delta(q, a, d)$
- For every $q \in Q$, $d \in \Gamma$, $\delta'(q, \epsilon, d) = \delta(q, \epsilon, d)$
- For every $q \in Q$,
 $\delta'(q, \epsilon, \epsilon) = \delta(q, \epsilon, \epsilon) \cup \{(p_0, \epsilon)\}$, if $q \in F$, and
 $\delta'(q, \epsilon, \epsilon) = \delta(q, \epsilon, \epsilon)$, if $q \notin F$.
- For every $a \in \Sigma$
 $\delta'(p_0, a, \epsilon) = \{(p_1, \epsilon)\}$, and $\delta'(p_1, a, \epsilon) = \{(p_0, \epsilon)\}$.
- For every $a \in \Sigma_\epsilon$, $d \in \Gamma_\epsilon$, where either $a = \epsilon$ or $d \neq \epsilon$,
 $\delta'(p_0, a, d) = \emptyset$, and $\delta'(p_1, a, d) = \emptyset$.

Problem 10: Decidability (6 points)

Show that

$EQINT_{DFA} = \{\langle A, B, C \rangle \mid A, B, C \text{ are DFAs over the same alphabet } \Sigma, \text{ and } L(A) = L(B) \cap L(C)\}$ is decidable.

This question does not require detail at the level of tuple notation. Rather, keep your proof short by exploiting theorems and constructions we've seen in class.

Solution:

We know that for any two DFAs A and B over an alphabet Σ , we can build a DFA D whose language is the intersection of $L(A)$ and $L(B)$. We also know that we can build a TM that can complement an input DFA, and we know that we can build a TM that checks whether the language of a DFA is empty (this TM can, for example, do a depth-first search from the initial state of the DFA, and explore whether any final state is reachable).

Also, note that $L(A) \subseteq L(B)$ iff $L(A) \cap \overline{L(B)} = \emptyset$.

So, we can build a Turing machine that takes as input $\langle A, B, C \rangle$, and first checks whether these are all DFAs (over the same alphabet Σ). Now, if they are, we must first check if $L(A) \subseteq L(B) \cap L(C)$. We can first build a DFA D that accepts $L(B) \cap L(C)$. Now we need to check if $L(A) \subseteq L(D)$, which is the same as checking if $L(A) \cap \overline{L(D)} = \emptyset$. We can do this by first complementing D to get a DFA E , and then building a DFA F that accepts the intersection of the languages of A and E , and then finally checking if the language of F is empty. We do a similar check to verify whether $L(B) \cap L(C) \subseteq L(A)$.

Hence, the decider for $EQINT_{DFA}$ works as follows.

1. Input is $\langle A, B, C \rangle$.
2. Check if A , B and C are DFAs over the same alphabet Σ . If not, reject.
3. Build the automaton D accepting $L(B) \cap L(C)$.
4. Complement D to get DFA E .
5. Build the DFA F that accepts $L(A) \cap L(E)$.
6. Check if $L(F) = \emptyset$. If it is not, then reject (as $L(A) \not\subseteq L(B) \cap L(C)$).
7. Complement A to obtain the DFA G .
8. Construct DFA H accepting $L(G) \cap L(D)$.
9. Check if $L(H) = \emptyset$. If it is, accept, else reject (as $L(B) \cap L(C) \not\subseteq L(A)$).

Problem 11: Reduction (8 points)

Let $ODD_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ does not contain any string of odd length}\}$. Show that ODD_{TM} is undecidable using a reduction from A_{TM} (Turing machine membership). You may not use Rice's Theorem.

Solution:

Let $A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM that accepts } w\}$. We know that A_{TM} is undecidable. Let us reduce A_{TM} to ODD_{TM} to prove that ODD_{TM} is undecidable. In other words, given a decider R for ODD_{TM} , let us show how to build a decider for A_{TM} .

The decider D for A_{TM} works as follows:

1. Input is $\langle M, w \rangle$.
2. Construct the code for a TM $N_{M,w}$ that works as follows:
 - a. Input to $N_{M,w}$ is a word x
 - b. Simulate M on w ; accept x if M accepts w .
3. Feed $N_{M,w}$ to R , the decider of ODD_{TM} .
4. Accept if R rejects; reject if R accepts.

For any TM M and word w , if M accepts w , then $N_{M,w}$ is a TM that accepts all words, and if M does not accept w then $N_{M,w}$ accepts no word. Hence M accepts w iff $L(N_{M,w})$ contains an odd-length word.

Hence D accepts $\langle M, w \rangle$ iff R rejects $\langle N_{M,w} \rangle$ iff $L(N_{M,w})$ contains an odd-length word iff M accepts w . Hence D decides A_{TM} .

Note that D does not simulate M on w ; it simply constructs the code of a TM $N_{M,w}$ (which if run may simulate M on w). So D simply constructs $N_{M,w}$ and runs R on it. Since R is a decider, it halts always, and hence D also always halts.

Since A_{TM} reduces to ODD_{TM} , i.e. any decider for ODD_{TM} can be turned into a decider for A_{TM} , we know that if ODD_{TM} is decidable, then A_{TM} is decidable as well. Since we know A_{TM} is undecidable, ODD_{TM} must be undecidable.