Computation, Machines and Formal Languages - Exam (spring 2003) Moed A

Lecturer: Prof. Michael Ben-Or.

Course number: 67521

Time: 3 hours.

Write your answers on the Hebrew exam sheet. If you wish, you can write your answers in (readable) English.

If the answer is yes/no, **write down the answer that you chose**.

Answer all the questions 1-6 and **one** of the questions 7-9.

Unless stated otherwise, you have to add a short explanation (within the designated area for each question) for your answers. Answers that will not be accompanied by explanations will not be credited. The explanation should include the main steps in the proof, do not get into technical details, notations etc. If your explanation is given by a contradicting example, you have to state the example together with a short explanation of the contradiction. If you have an example that contradicts an unproven but a reasonable assumption (such as $P \neq NP$), state the example, the reasonable assumption, and a short explanation of the contradiction.

Definitions and Notations: (identical to those we used in class)

For a language $\mathcal{L}$, we denote by $\overline{\mathcal{L}}$ the language $\Sigma^* \setminus \mathcal{L}$ (i.e. the complement of $\mathcal{L}$).

Complexity Classes:
REG is the class of regular languages.
CFL is the class of context-free languages.
R is the class of languages that are decidable by Turing Machines (TM).
RE is the class of languages that are recognizable by TM's.
P is the class of languages that are decidable by deterministic TM's in polynomial time.
NP is the class of languages that are decidable by non-deterministic TM's in polynomial time.
EXP is the class of languages that are decidable by deterministic TM's in exponential time.
$PSPACE$ is the class of languages that are decidable by deterministic TM's in polynomial space.
For a class of languages $\mathcal{C}$, we denote by $co\mathcal{C}$ the class of languages that their complements are in $\mathcal{C}$. that is, $co\mathcal{C} = \{L : \overline{L} \in \mathcal{C}\}$.

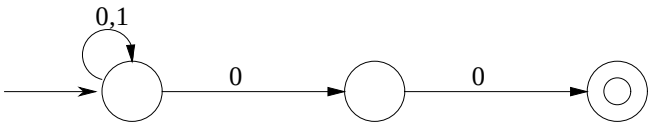
Computational problems:
$SUBSET - SUM = \{(S, t) : S$ is a (multi) set of numbers that contains a subset that sums to $t\}$
$HAM - PATH = \{(G, s, t) : G$ is a directed graph, that contains a path from s to t, that visits every node in the graph exactly once$\}$
We say that a Boolean formula is $k - cnf$ if it is of the form:

$$\bigwedge_i (a_1^i \vee a_2^i \vee \ldots \vee a_k^i)$$

where a_j^i is a variable or its negation.
$k - SAT = \{\phi : \phi$ is a satisfiable $k - cnf\}$

Reductions:
For two languages $\mathcal{L}_1, \mathcal{L}_2$, $\mathcal{L}_1 \leq_p \mathcal{L}_2$ denotes that there is a polynomial-time mapping reduction from $\mathcal{L}_1$ to $\mathcal{L}_2$.

1. (15 points) You are given the following nondeterministic finite automaton (see fig. 1).



   (a) Does the string 1101000 belong to the language of the automaton? (yes/no)
   You do not have to prove or explain your answer.
   Answer: yes.

   (b) Draw a **Deterministic** finite automaton with a minimal number of states for the same language.
   Give a short explanation of the correctness and the minimality.

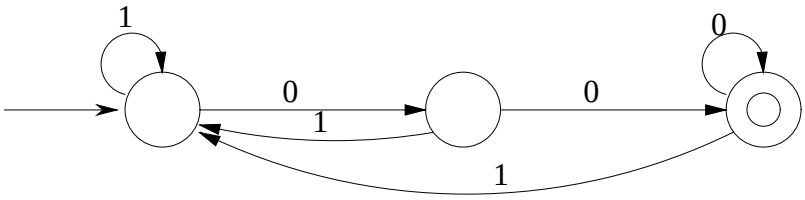


Figure 1: Deterministic Finite Automaton for the language Σ^*00

Correctness: From every state, when reading 1 we return to the starting state, from there only two 0's bring us to the accepting state. When in the accepting state, 0's don't take us out of it.

Minimality: The language defines three distinct equivalence classes: $\Sigma^*1 \cup \epsilon$, $\Sigma^*10 \cup 0$, $\Sigma^*100 \cup 00$. Thus By Myhill-Nerode the minimal DFA has three states.

2. (15 points) Let us define the following languages over $\Sigma = \{a, b, c, d\}$:

$$L_1 = \{a^n b^k c^n d^k : n, k \geq 0\}$$

$$L_2 = \{a^n b^k | n \leq k \leq 2n\}$$

For each one of the following languages, state whether it is in CFL or not.

You do not have to prove or explain your answers.

(a) L_1. **Answer:** L_1 is not a CFL (pump the word $a^{p_L}b^{p_L}c^{p_L}d^{p_L}$).

(b) $\overline{L_1}$. **Answer:** $\overline{L_1}$ is a CFL. $\overline{L_2} = K_1 \cup K_2 \cup K_3$, where $K_1 = \{w \mid w \notin a^*b^*c^*d^*\}$, $K_2 = \{a^*b^lc^*d^k \mid l \neq k\}$ and $K_3 = \{a^lb^*c^kd^* \mid l \neq k\}$. Clearly, $K_1 \in REG,\ K_2, K_3 \in CFL$ (the PDA checks that k and l are distinct). Then $\overline{L_1} \in CFL$ as a finite union of CFLs.

(c) L_2. **Answer:** L_2 is a CFL (with the CFG: $S \to aSbB|\epsilon;\ B \to b|\epsilon$).

(d) $\overline{L_2}$. **Answer:** $\overline{L_2}$ is a CFL. $\overline{L_2} = K_1 \cup K_2 \cup K_3$, where $K_1 = \{w \mid w \notin a^*b^*\}$, $K_2 = \{a^nb^k \mid n > k\}$ and $K_3 = \{a^nb^k \mid 2n < k\}$.
Clearly, $K_1 \in REG,\ K_2 \in CFL$ (with the CFG: $S \to aASb|\epsilon;\ A \to aA|a$), $K_3 \in CFL$ (with the CFG: $S \to aSBbb|b;\ B \to bB|\epsilon$).

3. (15 points) For each one of the following languages, write the smallest class that contains it out of the following list: R, RE, $coRE$, or none of these classes.

(a) $L = \{\langle M\rangle : M \text{ is a TM and } 011 \in L(M)\}$
Answer: L is not in R, by Rice theorem. It is in RE since given a TM M, we can simulate M on the input 011. If M accepts 011 then $\langle M\rangle \in L$.

(b) $L = \{\langle M\rangle : M \text{ is a TM, L(M) contains infinite number of words}\}$.
Answer: L is not in RE and not in coRE.
First, $A_{TM} \leq_m L$, using the following reduction. On input M, w, we construct the TM $B_{M,w}$, that on input y, simulates M on w. If M accepts w then $B_{M,w}$ accepts y. Otherwise it rejects. Then $(\langle M\rangle, w) \in A_{TM}$ iff $\langle B_{M,w}\rangle \in L$. Now,$A_{TM} \leq_m L$ iff $\overline{A_{TM}} \leq_m \overline{L}$, and so $\overline{A_{TM}} \notin RE$ implies $\overline{L} \notin RE$. Thus $L \notin coRE$
Second, $\overline{A_{TM}} \leq_m L$, using the following reduction. On input M, w we construct the machine $B_{M,w}$, that on input y, simulates M on w for $|y|$ steps. If within $|y|$ steps, M does not accept w, then $B_{M,w}$ accepts y. Otherwise it rejects.
If $(\langle M\rangle, w) \in \overline{A_{TM}}$ then $L(B_{M,w}) = \Sigma^*$.
If $(\langle M\rangle, w) \notin \overline{A_{TM}}$, then $L(B_{M,w})$ is a language with only finitely many words (only words with length $\leq k$, where k is the number of steps that takes M to accept w). Thus $\overline{A_{TM}} \leq_m L$, and $L \notin RE$.

4. (15 points) Given a language L over an alphabet Σ that does not contain the symbol $\sharp$, we define the language:

$EXACT - HALF(L) = \{w_1 \sharp w_2 \sharp \ldots \sharp w_{2k} : k \geq 0, w_i \in \Sigma^*$, and for **exactly** k indices $w_i \in L\}$

What is the smallest class from: $REG, CFL, P, NP, PSPACE$, that contains the following set of languages:

(a) $\{EXACT - HALF(L) : L \in REG\}$.

Answer: CFL. Let A be the DFA for L. The pushdown automaton for $EXACT-HALF(L)$, runs A on every w_i, and compares the number of strings in L to those not in L. This is done by pushing the symbol + to the stack when detecting $w_i \in L$ (and when the stack is empty or has + in it), and popping + whenever $w_i \notin L$. If the stack is empty or contains $-$, then we push $-$ when detecting $w_i \notin L$ and popping $-$ when $w_i \in L$. At the end we accept if the stack is empty.

To see that it is not in REG take $L = \{1\}$. Then $EXACT-HALF(L)$ defines infinite number of equivalence classes: $1, 1\sharp 1, 1\sharp 1\sharp 1, \ldots$

(b) $\{EXACT - HALF(L) : L \in P\}$

Answer: P. Let A be the polynomial-time algorithm deciding L. We run it on every w_i and we maintain a counter that is incremented when $w_i \in L$ and decremented when $w_i \notin L$. We accept if the counter is 0.

The running time is $2k$ times the running time of A, and hence polynomial in the length of the input.

To see that it is not in CFL. Take $L = a^n b^n c^n$. then by the pumping lemma we cannot pump the word $a^p b^p c^p \sharp a$ and stay in the language $EXACT - HALF(L)$.

(c) $\{EXACT - HALF(L) : L \in NP\}$

Answer: $PSPACE$. This is a generalization of the language $XOR-SAT$ that we saw in one of the exercises. It is $PSAPACE$ because we can try all the (polynomial-long) witnesses for each w_i and decide whether it is in L or not, and then we can simply count the number of strings in and not in the language.

It is not in NP (unless $NP = coNP$), because otherwise we would have a poly-time nondeterministic algorithm for $\overline{SAT}$: by taking the input formula ϕ and run the poly-time nondeterministic algorithm for $EXACT - HALF(SAT)$ on $\phi \sharp x_1$. It turns out that $\phi \in \overline{SAT}$ if and only if $\phi \sharp x_1 \in EXACT - HALF(SAT)$ (because x_1 is a satisfiable formula).

5. (15 points) Does the assumption (that we currently can't prove) that $HAM-PATH \in P$ implies:

   (a) $NP = coNP$? (yes/no)

   Answer: yes. $HAM - PATH \in P$ implies that $NP = P$, as $HAM - PATH$ is NP-complete language. P is closed under complement and so $NP = coNP$.

   (b) $SUBSET - SUM \leq_p 2 - SAT$? (yes/no)

   Answer: yes. $HAM - PATH \in P$ implies that $NP = P$. Also, we showed that $2-SAT \in P$. Let L_1, L_2 be arbitrary nontrivial languages in P. Then we know that $L_1 \leq_p L_2$.

6. (15 points) Under the assumption (that we currently can't prove) that $NP = EXP$, is it true that:

   (a) $NP = coNP$? (yes/no)

   Answer: yes. EXP is a deterministic class, and so it is closed under complement. If $NP = EXP$ then $coNP = EXP$. In particular $NP = coNP$.

   (b) $NP = P$? (yes/no)

   Answer: no. By definition, $P \subseteq NP \subseteq EXP$. By the Hierarchy Theorem $P \neq EXP$. And so if $NP = EXP$ it must be the case that $P \neq NP$.

7. (10 points) Prove that the following language is not in RE nor in $coRE$.

   $\{\langle M \rangle : M$ is a TM and $L(M) \in DTIME(n^3)\}$

   Hint: Show reductions from the halting problem and its complement. The output of the reductions should be a TM that either accepts a trivial language or a hard language.

   First, $\overline{A_{TM}} \leq_m L$, using the following reduction. The machine $B_{M,w}$, on input y:

   - Simulates M on w for $|y|$ steps. If within $|y|$ steps, M does not accept w, then $B_{M,w}$ accepts y.
   - If M accepts w within $|y|$ steps and if y is of the form $\langle N \rangle, x$, then $B_{M,w}$ simulates N on x. $B_{M,w}$ accepts y if N accepts x.

If $(\langle M \rangle, w) \in \overline{A_{TM}}$ then $L(B_{M,w}) = \Sigma^*$. Clearly Σ^* is in REG, and so can be decided in linear time.

If $(\langle M \rangle, w) \notin \overline{A_{TM}}$, then $L(B_{M,w}) = A_{TM} \setminus K$, where K is the language of words with length $\leq k$, where k is the number of steps that takes M to accept w. Note that $A_{TM} \setminus K$ is essentially equivalent to A_{TM} (we might concatenate k redundant states to the encoding of a given machine) and thus not in R. In this case, $L(B_{M,w}) \in RE \setminus R$, and in particular $L(B_{M,w}) \in RE \setminus DTIME(n^3)$.

Now, $(\langle M \rangle, w) \in \overline{A_{TM}}$ iff $L(B_{M,w}) \in DTIME(n^3)$, thus $\overline{A_{TM}} \leq_m L$, and so $L \notin RE$.

Similarly, $\overline{A_{TM}} \leq_m \overline{L}$, using the following reduction. The machine $G_{M,w}$, on input y:

- Simulates M on w for $|y|$ steps. If within $|y|$ steps, M does not accept w, and if y is of the form $\langle N \rangle, x$, then $G_{M,w}$ simulates N on x, and accepts y if N accepts x.
- If M accepts w within $|y|$ steps, then then $G_{M,w}$ accepts y.

If $(\langle M \rangle, w) \in \overline{A_{TM}}$, then $L(B_{M,w}) = A_{TM}$.

If $(\langle M \rangle, w) \notin \overline{A_{TM}}$ then $L(B_{M,w})$ is a complement of a finite a language and thus regular.

Now, $(\langle M \rangle, w) \in \overline{A_{TM}}$ iff $L(B_{M,w}) \in \overline{L}$, thus $\overline{A_{TM}} \leq_m \overline{L}$, and so $L \notin coRE$.

8. (10 points) Prove that for every enumerator E that enumerates the descriptions of all the Turing Machines in some order $M_1, M_2, \ldots$, there must exist an index i such that,

$$L(M_i) = L(M_{i+1})$$

Hint: Use the recursion theorem.

Answer: Consider the TM S, that on input x:

(a) Print its own description $\langle S \rangle$ (by the recursion theorem).

(b) Run E until it prints the description of S.

(c) Run E to obtain the next machine M after S.

(d) Run M on x and give the same answer.

Let i be the location of S in the order that E defines. Then M is in location $i + 1$ and clearly $L(S) = L(M)$.

9. (10 points) We say that two Boolean formulas, ϕ, ϕ', on the same variables are equivalent, if every assignment that satisfies ϕ also satisfies ϕ' and vice versa. We say that ϕ is minimal if there is no equivalent formula ϕ' to ϕ, that appears before it in the lexicographic order.
Show an interactive proof system for the following language:
$Non - Minimal - Formula = \{\phi : \phi$ is **not** minimal $\}$

Give a short explanation why this proof system has all the requirements.

Hint: Use the interactive proof system for $\sharp SAT$ that was given in class (you do not have to describe the proof system itself).

Answer: Consider the following protocol for $Non-Minimal-Formula$:

Input: A Boolean formula ϕ.

The protocol:

(a) The prover sends a Boolean formula ϕ' that appears before ϕ in the lexicographic order.

(b) Run the protocol for $\sharp SAT$ on the input $(\neg(\phi \Leftrightarrow \phi'), 0)$. (Note that $\phi \Leftrightarrow \phi'$ can be written as a Boolean formula with $\wedge, \vee, \neg$. Alternatively see the remark below).

Completeness: If ϕ is not minimal then indeed there is ϕ' that comes before it and is equivalent to ϕ. The prover sends this formula. Now, for two equivalent formulas, the formula $\neg(\phi \Leftrightarrow \phi')$ is a contradiction and there is no satisfying assignment for it. Hence, with probability 1, the verifier will accept the protocol in stage (b)

Soundness: If ϕ is minimal then there is no ϕ' that comes before it and is equivalent to ϕ. If the prover sends a formula that is not before ϕ in the order then the verifier will reject in the first stage. Otherwise every ϕ' that appears before ϕ is not equivalent to it, and therefore there is a satisfying assignment to $\neg(\phi \Leftrightarrow \phi')$. It follows that with probability at least 1/2, the verifier will reject in the second stage.

Remark: For those that did not see that $\phi \Leftrightarrow \phi'$ can be written as a "standard" Boolean formula. We could have instead of stage (b) do the following: the prover sends a number k (which is supposed to be the number of satisfying assignments for ϕ and ϕ') and then the verifier and the prover run the protocol for $\sharp SAT$ three times on the inputs: $(\phi, k), (\phi', k), ((\phi \wedge \phi'), k)$. And the verifier accepts if and only if it accepts on all three protocols.