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

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, $co\mathcal{C} = \{L : \overline{L} \in \mathcal{C}\}$.

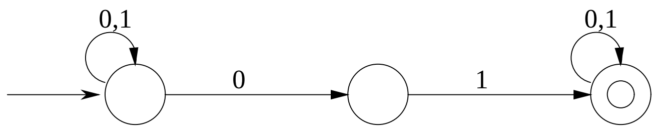
Computational problems:
$SUBSET - SUM = \{(S, t)$: S is a (multi) set of numbers that contains a subset that sums to $t\}$
$HAM - PATH = \{(G, s, t)$: G is a directed graph, that contains a path from s to t, that visits every node in the graph exactly once$\}$
We say that a Boolean formula is $k - cnf$ 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 $k - cnf\}$
$Graph{-}Non{-}Iso = \{(G_1, G_2) : G_1, G_2 are non ismorphic graphs\}$. $TQBF = \{\phi : \phi is fully quanitified boolean formula, and \phi has a true value ***\}$

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.

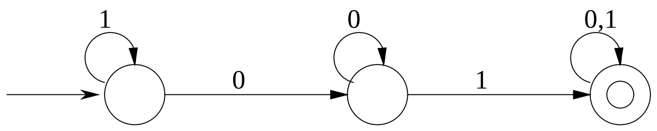


Figure 1: Deterministic Finite Automaton for the language $\Sigma^*01\Sigma^*$

   Correctness: Let w be in the language. Then it contains 01 as a substring. Either the first 01 is preceded by 1's and then we stay in the left state and then move to the accepting state when we read the 01. Or it is preceded by (zero or more) 1's followed by 0's, and then we move to the middle state with the first 0, and then to the accepting state with the appearance of the 1. In the accepting state the computation stays till the end.

   If there is no 01 (namely, the word is not in the language), then we either have only 1's, in this case we stay in the left state. Or we have (zero or more) 1's followed by 0's, and then we end up in the middle state, both are not accepting.

   Minimality: The language defines three distinct equivalence classes: $1^*, 1^*00^*, (0 \cup 1)^*01(0 \cup 1)^*$. Thus By Myhill-Nerode the minimal DFA has three states.

2. (15 points) Given an arbitrary language L over $\Sigma = \{0, 1\}$, define the language $K(L) = \{ww^R \mid w \in L\}$, where w^R is the reverse of w.

Let $L_1 = \Sigma^*$ and $L_2 = \{0^n 1^n \mid n \geq 0\}$.

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

You do not have to prove or explain your answers.

(a) $K(L_1) \in REG$.

Answer: No. $K(L_1) = \{ww^R \mid w \in \Sigma^*\}$ is not regular. There are infinitely many equivalence classes $[0^n 1^n]$ such that $0^n 1^n 1^k 0^k \in K(L_1)$ if and only if $k = n$.

(b) $K(L_1) \in CFL$.

Answer: Yes. The following CFL derives it: $S \to 0S0 \mid 1S1 \mid \epsilon$.

(c) $K(L_2) \in REG$.

Answer: No. $K(L_2) = \{0^n 1^{2n} 0^n | \ n \geq 0\}$ is not regular (pump the word $0^{p_L} 1^{2p_L} 0^{p_L}$)

(d) $K(L_2) \in CFL$.

Answer: No. (pump the word $0^{p_L} 1^{2p_L} 0^{p_L}$).

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) $L = \{\langle M, q\rangle :$ M is a deterministic TM and for every input w, M does not move to the state q during its running $\}$.

Answer: L in $coRE$ but not in RE.

First $L \notin RE$ by the reduction $\overline{A_{TM}} \leq_m L$.

On input M, w, we construct the TM $T_{M,w}$, that on input y, simulates M on w. If M accepts w then $T_{M,w}$ accepts y. Otherwise it rejects.

Now, $(\langle M\rangle, w) \in A_{TM}$ iff $\langle T_{M,w}, q_{acc}\rangle \in \overline{L}$ iff there exists an input w such that $T_{M,w}$ moves to q_{acc}.

Second, $L \in coRE$ by the following algorithm that recognizes $\overline{L}$:

on input $\langle M, q\rangle$, for $i = 1, 2, \ldots$:

run M on $w_1, \ldots, w_i$ (by the lexicographic order) each one for i steps. If for some $j \leq i$, M enters q then accept.

(b) $L = \{\langle M\rangle : M$ is a $TM,\ L(M) \in REG\}$.

Answer: L is not in RE and not in coRE.

First, $A_{TM} \leq_m \overline{L}$, using the following reduction. On input M, w, construct the TM $B_{M,w}$, that on input y, simulates M on w. If M

accepts w and there is $n \geq 0$ such that $y = 0^n 1^n$ then $B_{M,w}$ accepts y. Otherwise it rejects.
Note that $L(B_{M,w})$ is either empty ($\in REG$) or $\{0^n 1^n\}$ ($\notin REG$).
Then $(\langle M \rangle, w) \in A_{TM}$ iff $\langle B_{M,w} \rangle \notin L$. We conclude that $L \notin RE$.
Second, $A_{TM} \leq_m L$, using the following reduction. On input M, w construct the machine $G_{M,w}$, that on input y, simulates M on w for $|y|$ steps. If within $|y|$ steps, M does not accept w and there is $n \geq 0$ such that $y = 0^n 1^n$, then $G_{M,w}$ accepts y. Otherwise it rejects.
If $(\langle M \rangle, w) \in \overline{A_{TM}}$ then $L(G_{M,w})$ is the language $\{0^n 1^n\}$ ($\notin REG$).
If $(\langle M \rangle, w) \notin \overline{A_{TM}}$, then $L(G_{M,w})$ is a language with only finitely many words and so it is regular.
We conclude that $\overline{A_{TM}} \leq_m \overline{L}$, and so $\overline{L} \notin RE$.

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

   $MAJ(L) = \{w_1 \sharp w_2 \sharp \ldots \sharp w_{2k} : k \geq 0,\ w_i \in \Sigma^*$, and for **at least** k indices $w_l \in L\}$

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

   (a) $\{MAJ(L) : L \in REG\}$.
   Answer: CFL.
   Let A be the DFA for L. The pushdown automaton for $MAJ(L)$, 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 or it contains an even number of $+$ (to make sure that the input contains an even number of sub words w_i).
   To see that it is not in REG take $L = \{1\}$. Then $MAJ(L)$ defines infinite number of equivalence classes: $[(1\sharp)^{2n}]$
   Then $(1\sharp)^{2n}(0\sharp)^{2n-1}0 \in MAJ(L)$ and $(1\sharp)^{2(n-1)}(0\sharp)^{2n-1}0 \notin MAJ(L)$.

   (b) $\{MAJ(L) : L \in NP\}$
   Answer: NP. We construct the following machine M, it first checks that the input w is of the form $w_1 \sharp w_2 \sharp \ldots \sharp w_{2k}$ (that is having even number of sub-words) and determines the value of k, if not it rejects.

Then the machine guesses k distinct indices $j_1, j_2, \ldots, j_k$, where $1 \leq j_i \leq 2k$.

Then the machine guesses k witnesses $y_{j_1}, y_{j_2}, \ldots, y_{j_k}$, and verify that $x_i \in L$ using the witness y_{j_i} and the non-deterministic-polynomial-time machine M_L deciding L.

The first step and the guessing steps are polynomial. The verification is bounded by k times the running time of M_L, and hence the running time is polynomial in the length of the input.

To see that it is not in P (unless $NP = P$), take $L \in NP$, then if $MAJ(L) \in P$ we can decide L in **deterministic** polynomial time using the following fact: $x \in L$ if and only if $x\sharp x \in MAJ(L)$.

(c) $\{MAJ(L) : L \in coNP\}$

Answer: $PSPACE$. 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 $MAJ(SAT)$ on $\phi\sharp x_1$. It turns out that $\phi \in \overline{SAT}$ if and only if $\phi\sharp x_1 \in MAJ(SAT)$ (because x_1 is a satisfiable formula).

5. (15 points)

(a) Does the assumption (that we currently can't prove) that $HAM-PATH \notin coNP$ implies that $TQBF \in NP$? (yes/no)

Answer: no. $HAM-PATH \notin coNP$ implies that $NP \neq coNP$, as $HAM-PATH$ is in NP (and in fact an NP-complete language). $PSPACE$ is a deterministic class and is closed under complement, then $PSPACE = coPSPACE$. We also know that $NP \subseteq NPSAPCE = PSPACE$, and thus $coNP \subseteq PSPACE$.

$TQBF$ is $PSAPCE$-complete, and if $TQBF \in NP$, this means that $PSAPCE = NP$. But then it must be the case that $coNP \subseteq PSPACE = NP$. We also know that $coNP \subseteq NP \implies NP \subseteq coNP$. We get that $NP = coNP$, a contradiction.

(b) Does the assumption (that we currently can't prove) that $SUBSET-SUM \leq_p 2-SAT$ implies that $Graph-non-Iso \in NP$? (yes/no)

Answer: yes. We showed that $2-SAT \in P$. If $SUBSET-SUM \leq_p 2-SAT$, that is an NP-complete language is reducible to a language in P, then $NP = P$. The class P is deterministic, then we conclude that $coNP = NP = P$.

Now, $\overline{Graph - non - Iso} \in NP$ (To verify that $\langle G_1, G_2 \rangle \in \overline{Graph - non - Iso}$ one can check in polynomial time that $\pi(G_1) = G_2$). Thus, $Graph - non - Iso \in coNP$. Using the above we get that $Graph - non - Iso \in coNP = NP$.

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

   (a) $PSAPCE = EXP$? (yes/no)

   Answer: yes. We already know that $PSAPCE \subseteq EXP$. To show the equality, we shall show that in this case $EXP \subseteq PSAPCE$.

   We also know that $NP, coNP \subseteq PSAPCE$, and thus $NP \cup coNP \subseteq PSAPCE$. Now, if $EXP = NP \cup coNP$ then $EXP \subseteq NP \cup coNP \subseteq PSAPCE$. And thus $EXP \subseteq PSAPCE$.

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

   Answer: no. By the Hierarchy Theorem $P \neq EXP$.

   If $NP = P$ then $coNP = NP = P$. Now, if in addition $EXP = NP \cup coNP$ then $EXP = NP \cup coNP = P$, a contradiction.

7. (a) Prove that the following language is NP-complete:

   $L = \{\langle \phi, k \rangle : \phi$ has a satisfying assignment in which exactly k variables are assigned true $\}$

   Proof: It is easy to see that $L \in NP$ by the following nondeterministic algorithm: on input $\langle \phi, k \rangle$, choose nondeterministically an assignment with exactly k variables receiving true. If this assignment satisfies ϕ then accept.

   To see that it is NP-hard consider the following reduction from $3 - SAT$: on input ϕ (for $3 - SAT$), let $x_1, \ldots, x_n$ be its variables. Define new variables $y_1, \ldots, y_n$. Let ψ be the formula ϕ where we replace every appearance of x_i with $\neg y_i$, and every appearance of $\neg x_i$ with y_i.

   Let σ be the formula $\phi \wedge \psi$. The output of the reduction is $\langle \sigma, n \rangle$.

   We show that $\phi \in 3 - SAT$ iff $\langle \sigma, n \rangle \in L$.

   Now if $\phi \notin 3 - SAT$ then σ is not satisfiable and hence not in L.

   If $\phi \in 3 - SAT$ then there is a satisfying assignment for ϕ with exactly d true values (for some $0 \leq d \leq n$). In this case the assignment to the variables of σ that gives $x_1, \ldots, x_n$ the same values and to $y_1, \ldots, y_n$ the opposite values (that is, $y_i = true$ if and only if $x_i = false$), satisfies σ. Furthermore, there are exactly d x's that receive true and $n-d$ y's that receive true, altogether exactly n variables of σ receive the value true.

(b) Is it true that the following language is NP-complete:

$$L_k = \{\phi : \phi \text{ has a satisfying assignment in which exactly } k \text{ variables are assigned true }\}$$

Answer: No, unless $NP = P$.

The following is a polynomial-time algorithm for L_k:

On input ϕ, a formula with n variables, for every subset of size k of $x_1, \ldots, x_n$, define the assignment that gives true to exactly these k variables. If one of these assignments satisfies ϕ then accept, otherwise reject.

Clearly the algorithm gives the right answer, and its running time is the size of the formula multiplied by the number of assignments that we check which is $\binom{n}{k} = O(n^k)$. Note that unlike the previous section, here k is a constant and not part of the input, so the running time is polynomial in n.

8. Let $M_1, M_2, \ldots$ be the lexicographic order on Turing machines.

For every $w \in \Sigma^*$, we say that M_i is minimal for w if:

- M_i halts on the input ϵ (the empty string) with the output w. That is, at the end of the computation the tape of the machine contains the word w.
- For every $j < i$, either M_j does not halt on ϵ, or it halts with a different output than w.

Define the following language:

$$L = \{\langle M \rangle : M \text{ is minimal for some } w\}$$

Prove the following claims.

Hint: use the recursion theorem.

(a) $L \notin RE$.

Proof: Assume that $L \in RE$, then there is an enumerator E for the language. Define the following TM S that on input x does the following:

- Print own description $\langle S \rangle$ (this is possible by the recursion theorem).
- Run the enumerator E until it prints a description of a machine M_i that appears after S in the lexicographic order (such M_i exists because L is infinite).
- Run M_i on x and do the same.

Clearly, on every input and in particular on ϵ, S behaves exactly as M_i. Since $M_i \in L$, there is w which is the output of M_i on ϵ. Therefore w will also be the output of S on ϵ. But S comes before M_i in the order which contradicts the fact that M_i is minimal (otherwise it would not be printed by E). We conclude that no such enumerator exists and hence $L \notin RE$.

(b) Let $L' \subseteq L$ be an infinite language, then $L' \notin RE$.

Proof: The proof is similar to the previous section. Note that there we never used the fact that L contains all the minimal machines, we only needed an infinite number of them.

9. Show an interactive proof system for the following language:

$$L = \{\langle G, k \rangle : \text{ the graph } G \text{ contains a clique of size } k \text{ but no clique of size } k{+}1\}$$

Give a short explanation why this is indeed an interactive proof system.

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

Answer: Note that the language

$$L' = \{\langle G, k \rangle : \text{ the graph } G \text{ does not contain a clique of size greater than } k\}$$

is a $coNP$ language. This is because showing the opposite can be done in NP just by choosing (and verifying) a clique of size greater than k. As we showed in class, every language in $coNP$ has an interactive proof system, and in particular L'.

Now the protocol for the language L will be the following, On input $\langle G, k \rangle$:

- The prover sends names of k nodes in the graph, and the verifier checks that they are a clique (otherwise he rejects).
- The prover and the verifier run the protocol for the language L', on the input $\langle G, k \rangle$.

We explain why it is an interactive proof for L:

If $x \in L$, then there is a clique of size k, the prover will send this clique in the first stage. Also there is no clique of size greater than k, and hence by the completeness of the protocol that we showed in class, the verifier will accept the second stage with probability 1.

If $x \notin L$, then either there is no clique of size k, and the verifier will always reject in the first stage. Or there is a clique of size greater than k, and by the soundness of the protocol that we showed in class, the verifier will reject the second stage with probability at least 1/2.