Computation - Solution of Moed B

1. You are given the pushdown automaton over the alphabet $\{a, b, c\}$, described in Figure 1.

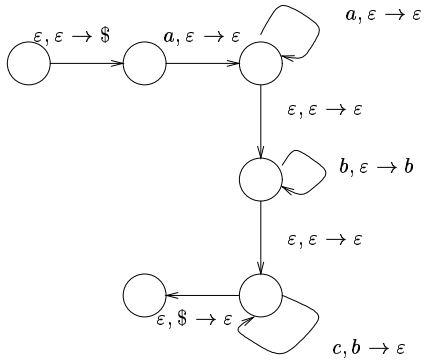

Figure 1: PDA for the question 1

   (a) What is the language that the automaton accepts? (6 points)
   Answer: $L = \{a^m b^n c^n : m \geq 1, n \geq 0\}$.

   (b) Write the context free grammar of this language. (6 points)
   Answer: $S \to AB$. $A \to aA|a$. $B \to bBc|\varepsilon$.

2. In the following questions the alphabet is $\{a, b, c\}$.

   (a) Is the language $L_1 = \{w : w$ does not contain aa as a substring$\}$ regular? **yes** (4 points)
   Answer: The complementary language $\bar{L_1}$ is regular - see the finite automaton for it in Figure 2. Since the class of regular languages is closed under complementation, L_1 is also regular.

   (b) Is the language $L_2 = \{w :$ in w, $\#a = \#b = \#c\}$ regular?
   no (4 points) **Answer:** Assume the contrary, that is, L_2 is regular. Let

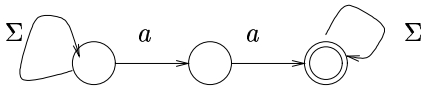

Figure 2: Automaton that accepts $\bar{L_1}$

p be the pumping length, and let $w = a^p b^p c^p$. By Pumping Lemma, there exist x, y, z such that $w = xyz$ and for all $i \geq 0$, $xy^i z \in L_2$. Note, that both x and y should contain only a's. Let $y = a^l$. Then $xy^2z = a^{p+l}b^p c^p \notin L_2$, and we reached a contradiction.

(c) What is the number of states in the minimal deterministic finite automaton that accepts the language $L_3 = \{w : |w| \leq 3\}$?
(4 points) **Answer:** The minimal deterministic automaton that accepts L_3 has 5 states, by the number of equivalence classes of L_3 : the first equivalence class contains ε, the second contains all words of length 1, the third class contains all words of length 2, the fourth class contains all words of length 3, and the fifth class contains all words of length 4 or more.

3. Define the operation $Parity$ on a language L over the alphabet Σ:

$Parity(L) = \{w_1\$w_2\$...\$w_n$: for every i, $w_i \in \Sigma^*$, and the number of i's such that $w_i \in L$ is even$\}$

Remark: it is given that $\$ \notin \Sigma$.

We say that a class of languages C is closed under the operation $Parity$ if for every language $L \in C$, $Parity(L) \in C$.
Is it known that the following classes are closed under $Parity$?

(a) REG
yes (5 points)
Answer: Let $L \in REG$. Let A be the DFA that accepts L. In Figure 3 we describe an NFA that accepts $Parity(L)$. Intuitively, the automaton "remembers" whether the current number of sub-words that belong to L is even or odd, and on each sub-word runs the automaton A.

(b) P
yes (5 points)
Answer: Let $L \in P$, and let M be the DTM that decides L. We

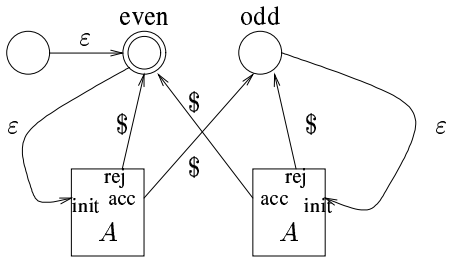

Figure 3: Automaton that accepts $Parity(L)$

define M_{Parity} in the following way. M_{Parity} has two tapes. On the first tape there is an input, and on the second tape it writes the number of sub-words of the input that belong to L. On each sub-word M_{Parity} runs M, and if M accepts, it increases the number on the second tape. Finally, after $blank$ symbol encountered (end of the input), M_{Parity} checks whether the number on the second tape is even or odd. The time complexity of M_{Parity} is $O((\text{complexity of } M \text{ on each subword}) * (\text{number of sub-words}))$, which is polynomial in the size of the input.

(c) NP

no (5 points)

Answer: Assume the contrary. Let M be the NTM that decides $Parity(SAT)$ in polynomial time. Now, $S\bar{A}T \subseteq Parity(SAT)$ (because a word in $S\bar{A}T$ contains 0 occurrences of words in SAT, which is even), thus, given an input to $S\bar{A}T$ we can run M. Thus, $S\bar{A}T$ is in NP, in contradiction to the fact that $S\bar{A}T$ is NP-complete.

(d) NL

yes (5 points)

Answer: Let $L \in NL$, and let M be the NTM that decides L in logarithmic space. Note, that $NL = co{-}NL$, thus there also exists M' that decides $\bar{L}$ in logarithmic space. Now, we construct $M_{Parity(L)}$ by running M and M' in parallel on each subword of the input. $M_{Parity(L)}$ also keeps a bit that indicates the parity of the number of sub-words that belong to L.

(e) $NPSPACE$

yes (5 points)

Answer: By Savitch Theorem, $NPSPACE = PSPACE$, and is closed under complementation. Let $L \in PSPACE$, and let M be the DTM that decides L in polynomial space. Now we construct

$M_{Parity(L)}$ by running M on each sub-word, and counting the number of sub-words of the input that belong to L. Clearly, $M_{Parity(L)}$ decides $Parity(L)$ in polynomial space.

4. Are the following languages in NP? And are they NP-complete ?

   (a) $MONOTONE - SAT = \{\phi : \phi \in 3 - cnf$, ϕ does not contain negations and it is satisfiable$\}$
   (5 points)
   Answer: $MONOTONE - SAT \in P$ (thus, in particular, it is not known to be NP-complete. All we have to do is to check that the formula is really in $3-cnf$ form and that it does not contain negations. If this is the case then we know that the assignment that gives $true$ to all the variables is satisfying. If it was NP-complete we would get that $NP = P$.

   (b) $4 - COLOR$
   (5 points)
   Answer: $4 - COLOR$ is NP-complete. It is in NP because we can non-deterministically choose a coloring of the graph in 4 colors and check that it is valid. This checkup takes time that is linear in the number of edges (or quadratic in the number of nodes). For hardness in NP we show a reduction from the NP-complete language $3 - COLOR$. Given a graph G we output the graph G' which is identical to G with the addition of a new node that is connected to all the other nodes in the graph. In every valid coloring of G', the new node must get a color which is different to all the other nodes. Therefore the coloring number of G' is exactly the coloring number of G plus 1. So G is three colorable if and only if G' is four colorable.

   (c) $\{\phi$: the formula ϕ does not have a satisfying assignment$\}$
   (5 points)
   Answer: The language is not known to be in NP. This language is the complement of the NP-complete language SAT. As we showed in class, if the complement of an NP-complete language is in NP then $NP = co - NP$ and this is not known.

5. (a) Is the following language NL-complete?
   $MULTI - PATH = \{(G, s, t)$: in the graph G there are at least two different paths from s to $t\}$
   yes/no (5 points)
   Answer: Yes. To see that the language is in NL we run the normal

algorithm for $PATH$ twice in parallel (i.e. we walk on the graph where each time the next step is chosen non-deterministically). In addition we keep a bit which is initialized to 0, and is set to 1 in the first time that the two paths differ. If at the end the two walks get to t and the bit is 1 then we accept. The amount of memory that is required is twice the amount that is needed for the algorithm to $PATH$ plus 1 which is still $O(log(n))$. For hardness in NL we show a reduction from $PATH$. Given (G, s, t) we output (G', s, t) where G' is identical to G with the addition of a path from s to t through a new node. Clearly if there is a path in G from s to t then with the new path there are at least two paths in G'. Otherwise there is exactly one path in G'.

(b) Is the language $\{w : \#a = \#b = \#c\}$ in L ? yes/no (5 points)
Answer: Yes. We keep a binary counter for each one of the symbols. We go once through the input and count the number of a's, b's, and c's. At the end we compare the counters. The amount of memory is $O(log(n))$ because the value of the binary counters is n at the most.

6. For each one of the following languages state whether it is in R, RE, or not in RE:

(a) $L_1 = \{\langle M_1, M_2\rangle : L(M_1) = \overline{L(M_2)}\}$
(6 points)
Answer: $L_1 \notin RE$. We saw that $ALL_{TM} \notin RE$ therefore a reduction from this language will complete the proof. The reduction: given an input $\langle M\rangle$ to ALL_{TM}, the output of the reduction will be $\langle M, M'\rangle$ where M' is a machine that always rejects, that is, $L(M') = \emptyset$, and $L(\overline{M'}) = \Sigma^*$. It is easy to see that $\langle M, M'\rangle \in L_1 \Leftrightarrow \langle M\rangle \in ALL_{TM}$.

(b) $L_2 = \{\langle M_1, M_2\rangle$: there exists $x \in \Sigma^*$ such that M_1 and M_2 halt when they are given x as input$\}$
(6 points)
Answer: $L_2 \in RE$ and $L_2 \notin R$. It is in RE because we can run the two machines on each input (in a lexicographic order) until we find x on which both machines halt. In order not to get stuck forever on one input we run the first input one step and then the first and the second two steps and so on.To see that it is not in R we show a reduction from $HALT^{\epsilon}_{TM}$ (the language of TM that halt on the empty string) which we proved not to be in R. The reduction: on input $\langle M\rangle$ to $HALT^{\epsilon}_{TM}$, the output will be $\langle M, M'\rangle$, where M' is a machine that only halts on ϵ. Clearly $\langle M\rangle \in HALT^{\epsilon}_{TM} \Leftrightarrow \langle M, M'\rangle \in L_2$.

(c) $L_3 = \{\langle M \rangle : L(M) \subseteq a^*\}$
(6 points)
Answer: $L_3 \notin RE$. By Rice theorem we know that $L_3 \notin R$. To see that it is not in RE we show that its complement is in RE. The last two facts imply that $L_3 \notin RE$ because otherwise we will have that both L_3 and $\overline{L_3}$ are in RE which implies that $L_3 \in R$. $\overline{L_3}$ is in RE because we can run over all the inputs that contain at least one symbol which is not a, in the same manner that we did in section 1, until we find such input that the machine accepts. In this case we know that the language of the machine is not contained in a^*.

7. You are given the following protocol between a prover P and a verifier V:

Input: A pair of graphs (G_0, G_1), where the number of nodes in each graph is n.

P: Choose two permutations $\pi_0, \pi_1 \in S_n$ (where the choice is uniformly distributed and independent) and send to V the graphs: $H_0 = \pi_0(G_0)$, $H_1 = \pi_1(G_1)$.
V: Choose uniformly and independently $b \in \{0, 1\}$, and send it to P.
P: Send to V two permutations $\tau_0, \tau_1 \in S_n$.
V: Accept if and only if: $H_b = \tau_0(G_0)$ and $H_b = \tau_1(G_1)$.

Is this protocol a zero-knowledge interactive proof system for the language $GRAPH - ISOMORPHISM$? Answer each condition separately:

(a) The running time of V is polynomial in the length of the input.
yes/no (4 points)
Answer: Yes. V has to choose a random bit and as a probabilistic machine it can do it in constant time, and then it has to check the equality of two pairs of graphs on n nodes and this can be done in time which is linear in the number of edges in the graphs, which is $O(n^2)$.

(b) If the graphs are isomorphic then V accepts with probability 1.
yes/no (4 points)
Answer: Yes. If the graphs are isomorphic, then G_0 and G_1 are isomorphic to H_0 and H_1. Therefore, for both values of b, P (which has unbounded computational power) can find the isomorphisms between H_b and each of the graphs G_0 and G_1.

(c) If the graphs are not isomorphic then V accepts with probability 1/2 at the most.

yes/no (4 points)
Answer: Yes. If the graphs are not isomorphic then regardless of what P is sending in the first stage, and regardless of the value of b, there cannot be an isomorphism between H_b and both G_0 and G_1. Otherwise, a composition of the two isomorphisms would give an isomorphism between G_0 and G_1. Therefore, in this case, with probability 0 (and in particular less then 1/2) V will accept.

(d) The zero-knowledge condition holds (no V^* can learn from the proof any additional information).
yes/no (4 points)
Answer: No. In the last stage of the protocol, V has isomorphisms between H_b and both G_0 and G_1. A simple operation (that can be done in polynomial-time) of composing the two isomorphisms gives V the isomorphism between G_0 and G_1. If a simulator (that runs in probabilistic polynomial-time) could simulate this protocol, it would also find (in probabilistic polynomial-time) an isomorphism between G_0 and G_1 and such an algorithm is not known to exist.