Computational Models 2022aa

Amit Bajar

Question 1

(a) The main idea is to define a NFA that guesses the state q that w will end at, while simultaneously running w' on M_A (when we start from q) to verify that $ww' \in A$ and running M_B on w' to verify that $w' \in B$. Let $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ be the DFA of A and $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ be the DFA of B. Define a NFA for C denoted by $M_C = (Q_C, \Sigma, \delta_C, S, F_C)$ as follows (first state in the vector is the guessed state, the three afterwards are the ones needed for the simultaneous running):
$Q_C := Q_A \times Q_A \times Q_A \times Q_B$, $S := \{(q, q_{0A}, q, q_{0B}) : q \in Q_A\}$, $F_C := \{(q, q, q', q'') : q \in Q_A, q' \in F_A, q'' \in Q_B\}$, $\delta_C((q_1, q_2, q_3, q_4), \sigma) := \{(q_1, \delta_A(q_2, \sigma), \delta_A(q_3, \sigma), \delta_B(q_4, \sigma')) : \sigma' \in \Sigma^*\}$

(b) I will prove that for every $n_1 < n_2 \in \mathbb{N}$ it holds that $[0^{n_1^2}]_L \neq [0^{n_2^2}]_L$ so L has infinitely many equivalence classes and thus by *myhill nerode* we will get that L can't be regular. Assume by contradiction that there are $n_1 < n_2 \in \mathbb{N}$ for which $[0^{n_1^2}]_L = [0^{n_2^2}]_L$, then it must hold that $0^{n_1^2} \cong_L 0^{n_2^2}$, and thus for the word $w = 0^{(n_1+1)^2 - n_1^2}$ we get that $0^{n_1^2} w \in L \Leftrightarrow 0^{n_2^2} w \in L$ which is a contradiction because $0^{n_1^2} w = 0^{(n_1+1)^2} \in L$ whoever $0^{n_2^2} w = 0^{n_2^2 + 2n_1 + 1} \notin L$ because $n_2^2 + 2n_1 + 1$ is not the square of any other natural number: $n_2^2 < n_2^2 + 2n_1 + 1 < n_2^2 + 2n_2 + 1 = (n_2+1)^2$

Question 2

(a) Proof: assume by contradiction that $L_1 \cup L_2 \in R$. Notice that $L_2 = (L_1 \cup L_2) \cap L_1^c) \cup (L_1 \cap L_2)$. whoever because $L_1 \cup L_2, L_1 \cap L_2, L_1 \in R$ and because R is closed under union, intersection and complement we get that $L_2 \in R$ which is a contradiction because $L_2 \notin R$.

(b) Disprove: define $L_1 := ACC, L_2 := \overline{ACC}$. we know that $L_1 \in RE, L_2 \notin RE$ and also $L_1 \cap L_2 = \phi \in R$ whoever $L_1 \cup L_2 = \Sigma^* \in R \subseteq RE$.

Question 3

(a) The motivation would be to use the only polynomial we know, and that is the one that bounds the runtime of our reduction. Let us denote it by $p(x)$, define $L_m := L_2||\{0\}^*$. We can define a reduction from L_1 to L_m as follows: $g(x) = f(x)||\{0\}^{p(|x|)-|f(x)|}$. Now define the following reduction from L_m to L_2: $t(x) = \begin{cases} w, & x = w0^* \wedge w \in L_2 \\ w \notin L_2, & otherwise \end{cases}$

(b) Assume that $L_1 \leq_{sP} L_2$. Due to *cook levin theorem* there exists a circuit ensemble of polynomial size which computes the reduction (because the output is always of the same length if the input is of a certain length, we can actually use circuits). The needed circuit is just a composition of the circuit of the reduction and the circuit of L_2.

Question 4

(a) Take some $L \in coNL$. Notice that $L \leq_L \overline{STCON} \Leftrightarrow \overline{L} \leq_L STCON$. Whoever $L \in coNL \Leftrightarrow \overline{L} \in NL$, and because we showed in class that $STCON \in NLC$ we get that $\overline{STCON}$ is $coNL$ hard. Now we can show that $\overline{STCON} \leq_L L$ with the following reduction: $f(< G, s, t >) =< G, s, t, 0 >$. It is trivial that it is log space computable and there are zero paths from s to t if and only if s and t are not connected by a path.

(b) It is trivial that $P \subseteq RP(1 - 2^{-2^n})$ because every polynomial deterministic TM can easily be converted to a randomized polynomial TM that does not utilize it's randomness. To show that $RP(1-2^{-2^n}) \subseteq P$, i will prove that for large enough n the machine for some $L \in RP(1-2^{-2^n})$ is a decider (does not make an error). Let $p(x)$ be the polynomial for the runtime of L. Because the TM always generates a random string of length $p(n)$ for input of length n we get that the probability for any string w for input of length n is $(\frac{1}{2})^{p(n)}$. If we assume by contradiction that for every input of length n the machine can be wrong it means that for every input of length n there is a string for which it is wrong. whoever this means that the probability of error for each input length n is at least $(\frac{1}{2})^{p(n)}$. this is a contradiction to the fact that the probability of error for length n is bounded by 2^{-2^n} because for large enough n it holds that $2^{-2^n} < (\frac{1}{2})^{p(n)}$ (no matter which polynomial p is). This means that for $n_0 \leq n$ the machine is never wrong for any random string. Thus we can define the following deterministic polynomial machine:

if $|x| < n_0$: then in $O(1)$ time determine if $x \in L$ (constant time, so can fit in the TM definition).
else: simulate the random machine with some string of constant size.