
CSL 705 Theory of Computation

Major Exam, Sem II 2010-11, Max 80, Time 2 hrs

Name _____ Entry No. _____ Group _____

Note (i) Write your answers neatly and precisely in the space provided with each question including back of the sheet. You won't get a second chance to explain what you have written.

(ii) You can quote any result covered in the lectures without proof but any other claim should be formally justified.

1. A *shuffle* of two strings $x, y \in \Sigma^*$ denoted by $x||y$ is the set of strings that can be obtained by interleaving the strings x and y in any manner. For example $ab||cd = \{abcd, acbd, acdb, cabd, cadb, cdab\}$. (The strings need not be of the same length.) For two sets of strings A, B , the shuffle is defined as $A||B = \bigcup_{x \in A, y \in B} x||y$. (10 × 2 **marks**)

- (a) If A, B are CFL, is $A||B$ a CFL ?

No- consider two CFLs $L_1 = \{a^i b^i, i \geq 0\}$ and $L_2 = \{c^j d^j, j \geq 0\}$. Then $a^i c^j b^i d^j \in L_1||L_2$. Using the PL for CFL, you can show that either no. of a's \neq no. of b's or no. of c's \neq no. of d's for $i, j \geq n$ where n is the constant of the P.L.

- (b) If A, B are regular is $A||B$ regular ?

$A||B$ is regular based on the following construction of an NFA M . $M = Q_A \times Q_B, \Sigma, F_A \times F_B, \delta$ where $\delta([q_A, q_B], a) = \{[\delta_A(q_A, a), q_B], [q_A, \delta_B(q_B, b)]\}$.

We then argue that w is accepted by this NFA iff $w \in w_1||w_2$ where $w_1 \in A$ and $w_2 \in B$.

2. Rice's theorem for index sets states that any non-trivial property Ψ of r.e. languages is not recursive. To determine if Ψ is r.e., it states three necessary and sufficient conditions (all conditions must be satisfied). Two among these are

(i) If $L \in \Psi$ and $L \subseteq L'$, then $L' \in \Psi$.

(ii) If $L \in \Psi$ and L is infinite then there is some finite subset of L that satisfies Ψ .

For the following properties, use the above criteria to show that they are not r.e. (5 × 2 **marks**)

- (a) The property that $L =$ all even length strings of Σ^* , where Σ denotes the input alphabet. $L \subset \Sigma^*$ and $\Sigma^* \notin \Psi$.

- (b) The property that L is regular

Consider any finite subset of a non-regular language. It violates the 1st property.

3. (a) If $L \in \mathcal{P}$, is $L^* \in \mathcal{P}$? (10 **marks**)

We want to show that $w = w_1 w_2 \cdots w_k$ where $w_i \in L$ can be recognised in polynomial time. Let $w_{i,j} = w_i w_{i+1} \dots w_j$ and we claim that $w_{i,j} \in L^*$ can be determined in polynomial time using a dynamic programming equation similar to CYK parsing algorithm. Without DP, the number of possible partitions of w is exponential and wouldn't suffice.

It is important that there should be some justification that your algorithm achieves polynomial time (even if it is in the RAM model, it is consistent with the definition of \mathcal{P}).

- (b) A language L is logspace reducible to L' iff $x \in L \Leftrightarrow f(x) \in L'$ and $f(x)$ can be computed using a logspace bounded Turing machine. Show that if $L_1 \leq_{\log} L_2$ and $L_2 \leq_{\log} L_3$ then $L_1 \leq_{\log} L_3$. Note that the length of $|f(x)|$ can be much longer than $\log |x|$, so your construction must address this carefully. (10 **marks**)

Let f and g be the logspace reduction function for $L_1 \leq L_2$ and $L_2 \leq L_3$ respectively. From relation between time and space, $|f(x)| \leq |x|^c$ for some constant c . So when g is applied to $f(x)$, then by the property of logspace reduction, it takes $c \log |x|$ space. The problem with this argument is that $f(x)$ is an intermediate string and there is no space to store it - compression techniques

can only achieve constant factor reduction but not exponential reduction (from polynomial to logspace).

The way to handle this is to generate $f(x)$ *on demand*. Since $f(x)$ serves as an input to the reduction function g , in the TM M_g corresponding to g , we keep track of the position of the input head. This takes only logspace (since it has to count upto at most $|f(x)|$). The combined TM for the composition function $g(f)$ computes the required input for M_g by simulating the TM M_f till the required output is generated - again a counter suffices as we do not store the remaining $f(x)$ (the output tape head only moves right).

The most important aspect of this problem was to adhere to the formal model for space complexity

- (c) Show that if there exists a polynomial time algorithm that can approximate the minimum number of colours χ required to color any given graph G within a factor less than $4/3$ (it returns an integer k , such that $\chi \leq k < \chi \cdot 4/3$) then $\mathcal{P} = \mathcal{NP}$. **(10 marks)**

You can reduce the problem of 3 coloring to this problem by noting that a graph is 3 colorable iff the approximate algorithm returns 3. If it returns a number ≥ 4 then the graph needs at least 4 colours. Since a polynomial time algorithm for 3 coloring (any NPC problem) implies $\mathcal{P} = \mathcal{NP}$, the result follows.

4. Consider an \mathcal{RP} algorithm A for a language L that uses an n bit random string for each input of size n . To decrease the probability of error (when A doesn't accept the input), we can repeat A k times to decrease the probability of error to $1/2^k$. Alternately, for $x \in L$, we can define a set of length n strings W_x such that if $x \in L$ then at least for one $y \in W_x$, $A(x, y) = \mathbf{True}$. Now consider $W = \bigcup W_x$, for any input $z \in \Sigma^n$, the algorithm accepts z if $\bigvee_{y \in W} A(z, y) = 1$, (boolean OR) else it doesn't accept. (Note that W_x need not be disjoint.) **(5 + 15 marks)**

- (a) Argue that this makes the algorithm error free and analyse the running time of the algorithm. Note that an RP algorithm never accepts an input $x \notin L$ and by the above scheme, for $x \in L$, it is guaranteed to find a string $y \in W$ for which $A(x, y)$ will be accepted so there is no error in either direction.

Since we may have to try all the strings in W , the running time is $|W| \cdot p(n)$ where $p(n)$ is a polynomial corresponding to the running time of the RP algorithm for input of size n .

- (b) Show the existence of a W such that $|W|$ is polynomial in n . Does this make the above algorithm polynomial time?

Consider a set S of αn strings randomly, each having n bits. From the property of the RP algorithm, for any fixed $x \in \Sigma^n$, the probability that $W_x \cap S = \phi$ is less than $(\frac{1}{2})^{\alpha n} \leq 2^{-\alpha n}$. The probability that S does not contain a witness for any of the (at most) Σ^n strings in L can be bounded by the sum $|\Sigma|^n \cdot 2^{-\alpha n}$. This is less than 1 by choosing $\alpha > \log_2 |\Sigma|$, i.e., the probability that S contains at least one string from W_x for every $x \in L$ is > 0 . Therefore a W exists of size $O(n)$ (i.e. polynomial).

This still does not make the algorithm polynomial time since the cost of constructing W is not accounted for. (It is actually a *non-uniform* polynomial time algorithm).