CECS 419-519, Writing Assignment 8, Due 8:00 am, April 19th, 2024, Dr. Ebert

Directions

Make sure name is on all pages. Order pages (front and back) so that solutions are presented in their original numerical order. **Please no staples or folding of corners (your papers won't get lost). A paper clip is OK** Show all necessary work and substantiate all claims. Avoid plagiarism.

Problems

1. Prove that there is some constant c such that, for every strings x and y,

$$K(xy) \leq 2K(x) + K(y) + c,$$

where xy denotes the concatenation of x with y. Hint: self-terminating code. (20 pts)

Solution. The idea is that, when representing program P as a binary string s and which outputs xy, within s must be two substrings: s_1 which encodes a minimum-length program P_1 that outputs x, and s_2 which encodes a minimum-length program P_2 that outputs y. Thus, we must encode these substrings as a pair $\langle s_1, s_2 \rangle$ in order to be able to parse s and extract both s_1 and s_2. The rest of s encodes a program that first simulates P_1 to get x, followed by simulating P_2 in order to get y, and then outputs xy. Finally, if we use a self-terminating code to encode $\langle s_1, s_2 \rangle$, then this encoding has length $2|s_1| + |s_2|$, and we have

$$K(xy) \leq 2K(x) + K(y) + c,$$

where $c > 0$ is the constant number of bits needed to encode the rest of s, and is indenpendent of x and y (in the sense that the two programs that get simulated could be any two arbitrary programs). □

2. With the help of the Turing-machine model of computation, we may provide formal definitions for the complexity classes $\texttt{P}$ and $\texttt{NP}$. For example language $L \in \texttt{P}$ iff there exists a DTM M and a polynomial $p(n)$ for which i) $L(M) = L$ and ii) for an input x of size $n \geq 0$, the computation $M(x)$ requires at most $p(n)$ steps. Similarly, $L \in \texttt{NP}$ iff there exists an NTM N and a polynomial $p(n)$ for which i) $L(N) = L$ and ii) for an input x of size $n \geq 0$, each branch of the computation tree $T(N, x)$ has a length not exceeding $p(n)$. Show that any language L that satisfies the latter definition is a member of $\texttt{NP}$ in accordance with the definition for $\texttt{NP}$ provided in the Computational Complexity lecture.

   (a) For a given instance x of L describe a certificate in relation to x. Hint: you may assume that, for the δ transition function of the NTM N that decides L, $|\delta(q, s)| \leq 2$, for all $q \in Q$ and $s \in \Gamma$. (10 pts)

Solution. Without loss of generality, we may assume that N's δ-transition function causes any computation branch to split into at most two subbranches per computational step. In other words, the computational tree $T(N, x)$ is binary. A certificate is then a binary string s of length $cp(|x|)$, for some constant $c > 0$, and where $cp(|x|)$ is an upper bound on the longest computational branch of $T(N, x)$. Note: if $T(N, x)$ is not binary, then we must increase the length of s by only a constant factor, since O(1) bits will be needed to encode which branch to follow at any nondeterministic step of N.

(b) Provide a semi-formal verifier algorithm that takes as input i) an instance x of L, ii) a certificate for x as defined in part a, and decides if the certificate is valid for x. (10 pts)

Solution. On inputs x and s, simulate N on input x. During the simulation, for the i th step of the computation for which the δ-transition function provides two possible next configurations, the simulator chooses configuration s_i, i.e. the first (respectively, second) choice provided by δ in case $s_i = 0$ (respectively, $s_i = 1$). Thus, certificate s allows for the simulation to follow a single branch of $T(N, x)$. Moreover, the verifier returns 1 iff the branch is an accepting branch.

(c) **Solution.** The running time is $O(p(n))$, since polynomial $p(n)$ provides an upper bound on the length of any branch of $T(N, x)$, and simulating a single step of a branch may be accomplished in O(1) steps (see the proof of the Time Hierarchy Theorem for an explanation of how this is accomplished).

3. For each decision problem L defined below, provide a complexity class C for which the statement $L \in C$ is true. Moreover, C should be chosen so that it is the best possible, meaning that it is the least complex of all possible valid choices. For example, if L is in both `P` and `NP`, then `P` is the better choice. The choices for C are `P`, `NP`, `co-NP`, Σ_2^p, and Π_2^p. Provide a predicate-logic-formula characterization for each (define all certificate sets and predicate functions). (5 pts each)

(a) An instance of `Sum-Free Hamilton Path` consists of a simple graph $G = (V, E)$ and a nonnegative integer t. Moreover, each of the n vertices of G is labeled with a positive integer. The problem is to decide if i) G has a Hamilton path $P = v_1, v_2, \ldots, v_{n-1}, v_n$ for which the set of labels $S = \{\text{label}(v_1), \ldots, \text{label}(v_{n/2})\}$ has no subset that sums to t.

Solution. Either Π_2^p or Σ_2^p. Note: The problem should have labeled the *edges* and not the vertices. This would make it Σ_2^p: "Does *there exist* a set of edges that make a Hamilton Path such that *for every* subset of those edges the labels of those edges do not sum to t?".

(b) An instance of `Satisfied Max Cut` is a simple graph $G = (V, E)$ and a nonnegative integer $k \geq 0$, where each vertex v of G is labeled with a Boolean formula F_v. The problem is to decide if, for every cut of size k (See the `Max Cut` problem from Writing Assignment 6), there is at least one blue vertex and one red vertex each of which is labeled with a satisfiable formula.

Solution. Π_2^p: "*For every* subset of vertices B (the blue vertices), *there exists* two vertices u and v and two assignments α and β such that i) $u \in B$, ii) $F(u)$ is satisfied by α, iii) $v \in V - B$ and iv) $F(v)$ is satisfied by β".

(c) An instance of `Unsolvable Quadratic Diophantine` consists of three positive integers $a, b, c > 0$ and the problem is to decide if the equation $ax^2 + by = c$ has no positive integer solutions, meaning there is no pair (x, y) of positive integers that satisfies the equation.

Solution. `co-NP`. "*For every* pair of positive integers (x, y), $|x|, |y| \leq |c|$, $ax^2 + by \neq c$."

(d) Recall the Tseytin transformation f : `SAT` $\rightarrow$ `3SAT` used to map reduce `SAT` to `3SAT`. An instance of decision problem L is a Boolean formula F and a nonnegative integer k. The problem is to decide if $f(F)$ has at least k clauses.

Solution. `P`: the Tseytin transformation is polynomial-time computable. One can simply count the number of clauses upon completion of the transformation.

4. For the CFG provided in Example 2.3 of the Space Complexity lecture, provide a derivation of the expression

$$((a \times a)) + (a + a \times a).$$

(10 pts)

5. Provide a CFG G for which $L(G)$ equals all words of the form $u\#v$, where $u, v \in \{a, b\}^*$ and $u \neq v$. Defend your solution by explaining why u can never equal v, and that all the words that should be in $L(G)$ can in fact be generated. (20 pts)

Solution. The main goal is to form a word of the form $xcA\#ydA$, where $x, y \in \{a, b\}^*$ satisfy $|x| = |y|$, $c, d \in \{a, b\}$, and $c \neq d$. The A variable can then be used to create suffices of any length, since characters c and d have already established that the left side does not equal the right side. The key insight towards achieving this goal is to derive dA *before* deriving y, and to derive cA *after* deriving x. This involves two cases (and hence two sets of rules): $c = a, d = b$, and $c = b, d = a$. The rules are shown as follows.

$$S \rightarrow DaA \mid D'bA$$

$$D \rightarrow aDa \mid aDb \mid bDa \mid bDb \mid bA\#$$
$$D' \rightarrow aD'a \mid aD'b \mid bD'a \mid bD'b \mid aA\#$$
$$A \rightarrow AA \mid a \mid b \mid \varepsilon$$

For example, to derive $babab\#baaab$ we have

$$S \Rightarrow DaA \Rightarrow bDaaA \Rightarrow baDbaaA \Rightarrow babA\#baaA \overset{*}{\Rightarrow} babab\#baaab. \quad \square$$