The University of British Columbia

Final Exam, December 5, 2017

CPSC 421

Closed book examination | Time: 2.5 hours

Last Name ____________________ **First** ______________ **Signature** ____________________

Student Number ____________________

Special Instructions:

Two two-sided 8.5 x 11 sheets of notes allowed.

1		16
2		15
3		15
4		15
5		15
Total		76

[16] **1**. Circle either T (true) or F (false).
Circle either T for true, or F for false, for each of the statements below:

The set of functions $\{0,1\} \to \mathbb{Z}$ is countable. T F

True: this set of functions is in bijection with $\mathbb{Z}^2 = \mathbb{Z} \times \mathbb{Z}$, which is in bijection with $\mathbb{N}^2$, which we know is countable.

The set of functions $\mathbb{Z} \to \{0,1\}$ is countable. T F

False: this set of functions is in bijection with $\mathrm{Power}(\mathbb{Z})$, which is in bijection with $\mathrm{Power}(\mathbb{N})$, which we know is uncountable.

If $L' \leq_{\mathrm{P}} L$ then $L' \in \mathrm{P}^L$. T F

True: one can test membership in L' by applying the reduction from L' to L (which is poly time) and then making a single oracle query to L.

If L is undecidable and recognizable, then L's complement is unrecognizable. T F

True: otherwise L and its complement would both be recognizable, and hence L (and its complement) would be decidable (by running the two algorithms for recognizability in parallel).

If L and L' are in PSPACE then $L^* \cap L'$ is also in PSPACE. T F

True: PSPACE is closed under the star operation and under intersection. (For the star operation one can use dynamic programming; for intersection one can run the two algorithms, one after the other.) Hence L^* is in PSPACE, and hence $L^* \cap L'$ is in PSPACE.

If L is PSPACE-complete and $L' \in \mathrm{NP}$, then $L' \leq_P L$. T F

True: NP is contained in PSPACE; hence L' is in PSPACE, and hence L' can be reduced to L in poly time.

If $L \in \mathrm{P}$, then there are polynomial size circuits for L. T F

True (see [Sip], Chapter 9), although not covered in 2019.

[15] **2.** Give a Turing machine that takes as input, $x \in \{0,1\}^*$, and (1) accepts x if x begins with a 0 and x has exactly two more 0's than it has 1's, and (2) rejects x otherwise. You must **explain how your machine works**, and **explicitly write** your choice of $Q, \Sigma, \Gamma, \delta, q_0, q_{\mathrm{accept}}, q_{\mathrm{reject}}$. To describe δ, you may (1) list its values, or (2) use a diagram as used in Sipser's textbook (and class), or (3) give a table of its values as done in class and the solutions to Homework 10.

SOLUTION

See similar examples on the homework, on Midterm 2019, and on other previous exams.

[15] **3.** Let L be the language of descriptions of a sequence of positive integers $n_1, \ldots, n_k, m, t$ such that there is an $I \subset [k] = \{1, \ldots, k\}$ and a $J \subset [k]$ for which

$$\left(\sum_{i\in I} n_i\right) \ + \ m\left(\sum_{j\in J} n_j\right) = t.$$

Show that L is NP-complete; you may use the fact that SUBSET-SUM and PARTITION are known to be NP-complete.

SOLUTION

L **is in NP:** Non-deterministically pick subsets I, J of $[k]$ and check if

$$\left(\sum_{i\in I} n_i\right) \ + \ m\left(\sum_{j\in J} n_j\right) = t.$$

Since the subsets I, J each require k choices, and k is less than the size of the input, the choosing and checking can be done in polynomial time.

Any language in NP can be reduced to L**:** Any language in NP can be reduced to SUBSET-SUM. So given an instance

$$\langle n_1, \ldots, n_k, t\rangle$$

of SUBSET-SUM, it suffices to describe a polynomial time reduction to L. To do this compute $t+1$ and write down the string

$$\langle n_1, \ldots, n_k, t+1, t\rangle$$

as an instance of L (i.e., we take $m = t+1$ for the variable m above); this clearly can be done in poly time. This string lies in L iff there are I, J with

$$\left(\sum_{i\in I} n_i\right) \ + \ (t+1)\left(\sum_{j\in J} n_j\right) = t;$$

but this equation can only hold if J is empty (since the n_i's are non-negative), and hence this equation holds iff

$$\left(\sum_{i\in I} n_i\right) = t,$$

which holds iff $\langle n_1, \ldots, n_k, t\rangle$ lies in SUBSET-SUM.

[15] **4.** Fix an integer $k \in \mathbb{N}$. Let L be the language of strings over $0, 1$ whose k-th last symbol is a 1 (and whose length is therefore at least k), i.e.,

$$L = \left\{ x1y \;\middle|\; x, y \in \{0,1\}^* \text{ and } |y| = k-1 \right\}.$$

(a) Write an NFA that recognizes L and has at most $k+1$ states, and explain how your NFA works.

SOLUTION

We non-deterministically wait at the initial state, q_0, until we see a 1 and then see exactly $k-1$ remaining characters, at which point we accept (since the input is of the form $x1y$ with $|y| = k-1$. Hence the alphabet is $\Sigma = \{0,1\}$ with initial state q_0 and transitions

$$\delta(q_0, \epsilon) = \emptyset, \quad \delta(q_0, 1) = \{q_0, q_1\} \quad \delta(q_0, 0) = \{q_0\};$$

the state q_1 is reached upon (a non-deterministic amount of waiting followed by) reading a 1. Hence we set

$$\delta(q_i, 0) = \delta(q_i, 1) = \{q_{i+1}\}, \quad \delta(q_i, \epsilon) = \emptyset$$

for $i = 1, \ldots, k-1$, with q_k the unique final state and no transitions leaving q_k (i.e., $\delta(q_k, \sigma) = \emptyset$ for $\sigma = 0, 1, \epsilon$).

Hence the NFA is $(Q, \Sigma, q_0, \delta, F)$ where $\Sigma = \{0,1\}$, $Q = \{q_0, q_1, \ldots, q_k\}$ (which has $k+1$ states), $\delta \colon Q \times \Sigma_\epsilon \to Q$ is described above, and $F = \{q_k\}$.

(b) Prove that any DFA recognizing L has at least 2^k states.

SOLUTION

We claim that $\mathrm{AccFut}_L(s)$ are all distinct as s ranges over all elements of $\{0,1\}^k$. Indeed, if $s = \sigma_1 \ldots \sigma_k$ and $s' = \sigma'_1 \ldots \sigma'_k$, then for some $i \in [k]$ we have $\sigma_i \neq \sigma'_i$ (i.e., the i-th symbol of s, s' are different). In this case all strings of length $k-i$ lie in one of $\mathrm{AccFut}_L(s), \mathrm{AccFut}_L(s')$ (according to whether, respectively $\sigma_i = 1$ or $\sigma'_i = 1$) and all do not lie in the other.

Hence there are at least 2^k distinct values of $\mathrm{AccFut}_L(s)$ as s varies over all strings, and hence, by the Myhill-Nerode theorem, any DFA recognizing L has at least 2^k states.

[18] **5**. **Short Problems**. Each question is worth 3 points. Answer each question and **justify your anwer. No credit will be given for a simple yes or no.**

(a) Let B be any PSPACE-complete language. Is $P^B \subset$ PSPACE?

SOLUTION Yes: given a polynomial time algorithm for a Turing machine with oracle B, each oracle query to B queries a string of size at most polynomial in n, the input length (since the algorithm runs in polynomia time). Since B is PSPACE-complete, B lies in PSPACE and hence one can simulate any oracle query to B with a (deterministic) algorithm that uses space at most polynomial in the string queried, which is therefore polynomial in n, the input length. The polynomial time algorithm with simulated oracle queries runs in (1) polynomial space for the algorithm without the oracle queries (since a polynomial time algorithm uses at most polynomial space), plus (2) an extra polynomial space for each oracle query (this space can be reused upon each query); hence this algorithm runs in polynomial space.

(b) Is NP contained in $\mathrm{P}^{\mathrm{3SAT}}$?

SOLUTION

Yes: any language, L, in NP has a polynomial time reduction to 3SAT; running this reduction and then making a single oracle query to 3SAT (accepting the input if the oracle query says "yes," rejecting the input if "no") decides the language L in polynomial time.

(c) Let L be the language of descriptions $\langle M, w, q\rangle$ of a (deterministic) Turing machine M, an input w to M, and a state q of the Turing machine such that q is reached during the computation of M on input w. Is L recognizable?

SOLUTION

Yes: using a universal Turing machine, simulate M on input w and accept if the simulated computation ever reaches q. This algorithm accepts $\langle M, w, q\rangle$ iff q is reached during the computation of M on input w; hence L is recognized by this algorithm.

(d) Let SNEAKY-PSPACE be the descriptions $\langle M, w, 1^s \rangle$ where M accepts w within space 1^s. Prove that if $L \in$ PSPACE, then L has a polynomial time reduction to SNEAKY-PSPACE.

SOLUTION

Since L is in PSPACE, there is a polynomial $p(n)$ and a Turing machine, M, such that on input w, M decides w within space $p(|w|)$. So given a description of an instance $\langle w \rangle$ of L (i.e., a word in the alphabet of L), consider the algorithm that produces the string $\langle M, w, 1^{p(|w|)} \rangle$. This algorithm takes time polynomial in the length of $\langle w \rangle$, since M is fixed and p is a fixed polynomial. Furthermore $w \in L$ iff $\langle M, w, 1^{p(|w|)} \rangle$ lies in SNEAKY-PSPACE (which in 2019 we called PSPACE-SNEAKY). Hence this algorithm is a poly time reduction from L to SNEAKY-PSPACE.

(e) If L is in P and L' is any language, then is $\{s \mid st \in L \text{ for some } t \in L'\}$ necessarily in P? [Hint: Consider $\{0^n 1^n\}$.]

SOLUTION

The set P is countable (each language in P is decided by some Turing machine, and there are countably many Turing machines).

Let $L = \{0^n 1^n\}$. For any subset $N \subset \mathbb{N}$, let

$$L' = \{01^n \mid n \in N\}.$$

Then

$$S(L, L') = \{s \mid st \in L \text{ for some } t \in L'\} = \{0^{n-1} \mid n \in N\}.$$

It follows that any language over the alphabet $\{0\}$ can occur as a set $S(L, L')$; hence the set of languages of the form $S(L, L')$ is uncountable. Since P is countable, some language of the form $S(L, L')$ is not in P.

The End