Exam 3. December 6, 2021

CS181: Fall 2021

Guidelines:

- The exam is closed book and closed notes. Do not open the exam until instructed to do so.
- Write your solutions clearly and when asked to do so, provide complete proofs. Unless explicitly asked not to use a specific result, you may use results and theorems from class without proofs or details as long as you state what you are using.
- I recommend taking a quick look at all the questions first and then deciding what order to tackle to them in. Even if you don't solve the problems fully, attempts that show some understanding of the questions and relevant topics will get partial credit.
- You can use extra sheets for scratch work, but you can only use the white space (it should be more than enough) on the exam sheets for your final solutions. The exams will be scanned into gradescope so for each problem, only the corresponding white space will be used for grading.
- Most importantly, make sure you adhere to the policies for academic honesty set out on the course webpage. The policies will be enforced strictly and any cheating reported with the score automatically becoming zero.
- Write clearly and legibly. All the best!

Problem	Points	Maximum
1		5
2		3
3		3
4		3
5		3
6		3
7		3
8		3
Total		26

Name	
UID	

1 Problem

The answers to the following should fit in the white space below the question.

1. Which of the following statements are true? No need for explanations. [2 points]

   (a) Define the function $F : \{0,1\}^* \to \{0,1\}$ as $F(M) = 1$ if and only if there are no smaller-size Turing machines equivalent to it. The function F is semantic.

   Solution. False. Let M_0 be the smallest TM that computes $\{0,1\}^*$. Let M_1 be the same as M_0, but with a dummy state and arbitrary transition from it added. Then, $F(M_0) = 1$, $F(M_1) = 0$ but the two machines compute the same language, so F is not semantic. □

   (b) The language $L \subseteq \{0,1\}^*$ defined below is computable:

   $$L = \{\langle M \rangle : \langle M \rangle \text{ has length at most 100 and encodes a TM that halts on zero}\}.$$

   Solution. True. There are finitely many descriptions of TMs of length at most 100. Let $w_1, \ldots, w_m$ be all the short (length at most 100) descriptions of TMs that halt on zero. Then, a TM that just checks whether its input w equals some w_i computes L. □

   (c) If we have a reduction from a function F to HALTONZERO then F is uncomputable.

   Solution. False. Take $F(x) = 1$ if and only if $x = 1$. Then we can map string 1 to a TM that immediately halts on any input, and all the other strings will be mapped to a TM that just loops on any input. Then, the constructed mapping is a reduction but F is clearly computable. *Note:* to show that F is uncomputable we need a reduction from any uncomputable function (e.g., HALTONZERO) to F. □

   (d) If we have reductions from a function F to G and from a function G to H, then we also have a reduction from F to H.

   Solution. True. Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be reductions from F to G and from G to H, respectively. Let $\mathcal{R} = R_2 \circ R_1$ be a composition of the two reductions. Then, $F(x) = 1$ if and only if $G(\mathcal{R}_1(x)) = 1$ if and only if $H(\mathcal{R}_2(\mathcal{R}_1(x))) = 1$, so $\mathcal{R}(x) = \mathcal{R}_2(\mathcal{R}_1(x))$ is a reduction from F to H. □

2. Write a QIS that is equivalent to the *Goldbach Conjecture*, i.e., the statement "Every even number greater than two is the sum of two primes." [1.5 point]

   (You can assume that you have access to a formula $Prime(p)$ that returns true if and only if p is a prime as in the homework.)

   Solution. Let $Prime(p) = (p > 1) \wedge \neg\exists m \exists n : p = m \cdot n$ be a formula for primality. Then, the following statement is equivalent to the Goldbach conjecture: Or you can also do:

   $$\forall m \ (m \leq 1) \vee (\exists \, p, q \ ((2 \cdot m = p + q) \wedge Prime(p) \wedge Prime(q))).$$

   (There are many other possibilities.) □

3. Call a TM M *egotist* if on any input it just prints its own encoding three times (i.e., ignores inputs and just talks about itself ...). That is for any input w, $M(w) = \langle M \rangle \langle M \rangle \langle M \rangle$. Show that there exists an egotist Turing machine. [1.5 points]

   Solution. [Solution based on KRT] Define $T : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ as the TM that does the following $T(\langle N \rangle, w) = \langle N \rangle \langle N \rangle \langle N \rangle$. Now, by Kleene's recursion theorem applied to T, there exists a machine K such that $K(w) = T(\langle K \rangle, w) = \langle K \rangle \langle K \rangle \langle K \rangle$ for all w. That is, K is an egotist Turing machine. □

   Solution. [Direct Construction] Let B be the Turing machine that, on input $w = \langle M \rangle$, first computes $q(\langle M \rangle)$ (which is a description of a TM $P_{\langle M \rangle}$ that prints $\langle M \rangle$ and halts) and then outputs *three copies* of the description of the following TM:

   (a) On input x, run $P_{\langle M \rangle}$

   (b) Pass the output to M

   (c) Run M

   Then let $A = P_{\langle B \rangle}$ be a machine that outputs the description of B. Analogously to the construction for $SELF$, a machine that runs A, passes the output to B, and runs B, is an egotist TM. □

2 Problem

Call a TM M unbounded if there are an infinite number of inputs x such that $M(x) = 1$.

Consider the function $F : \{0,1\}^* \to \{0,1\}$ defined as $F(M) = 1$ if and only if M is an unbounded TM.

Prove that F is uncomputable. **Do not** use Rice's theorem for this problem but give a reduction from any of the uncomputable problems we discussed in class. [3 points]

Solution. We reduce $HALTONZERO$ to F (in fact, the same reduction we used in class for NOTEMPTY works here as well). For a TM M, let $\mathcal{R}(M)$ be a TM N defined as folllows:

1. Run $Eval(M, 0)$

2. Return 1.

Notice that if M halts on zero, then $N(x) = 1$ for infinitely many inputs x (actually, all of them). On the other hand, if M does not halt on 0, then N will keep simulating M, so N does not return 1 on any input. Hence, $\mathcal{R}$ is a reduction from an uncomputable problem, so F is also uncomputable. □

3 Problem

Call a function $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ that takes two Turing machines as input *doubly-semantic* if $F(A, B) = F(A', B')$ whenever TM A is equivalent to A' and TM B is equivalent to B'. Call a doubly-semantic function non-trivial if it is not the constant *zero* or constant *one* function.

Prove that every non-trivial doubly-semantic function is uncomputable. [3 points]

Solution. Let F be any non-trivial doubly-semantic function. Since F is non-trivial, there exists A_0, B_0, A_1, B_1 such that $F(A_1, B_1) = 1$, and $F(A_0, B_0) = 0$. Consider the following cases.

1. $F(A_0, B_1) = 0$. Then, $G(M) = F(M, B_1)$ is a non-trivial semantic function. It is semantic because $G(M_1) = G(M_2)$ whenever M_1 is equivalent to M_2, and it is non-trivial because $G(A_0) = F(A_0, B_1) = 0$ and $G(A_1) = F(A_1, B_1) = 1$. Hence, by Rice's theorem, G is not computable. However, if F were computable, then G would also be computable. Therefore, F is uncomputable.

2. $F(A_0, B_1) = 1$. A similar argument can be applied to $G(M) = F(A_0, M)$.

In each case, F is uncomputable. □

4 Problem

Show that the language $L = \{x : x \text{ contains more 1's than 0's}\}$ is context-free. [3 points]

Solution. The following grammar generates L.

$$S \to 1S \mid 1P \mid 0P1S$$
$$P \to PP \mid 1P0 \mid 0P1 \mid \varepsilon,$$

where P is a grammar that generates strings which contain equally many 1's and 0's. Indeed, if a string x starts with 1, then $x = 1x'$ where x' has more ones than zeros (so it can be generated by S) or the number of ones in x' is the same as the number of zeros in x' (so it can be generated by P). On the other hand, if x starts with a zero, then there exists some prefix with equally many ones and zeros (because the quantity $\#1s - \#0s$ changes by one with each new symbol, it is negative after the first symbol, and is positive in the end of the word). Hence, x is of the form $0x'1x''$ where x' has equally many ones and zeros, and x'' has strictly more ones than zeros, so x', x'' are generated by P, S, respectively. □

5 Problem

For a string $x \in \{0,1\}^*$, let $Reverse(x)$ denote the string backwards (e.g., $Reverse(000110) = 011000$) and let $Complement(x)$ denote the string with all bits flipped (e.g., $Complement(000110) = 111001$).

Show that the language $L = \{u \circ Reverse(Complement(u)) : u \in \{0,1\}^*\}$ is context-free. [3 points]

As examples, $001011, 10110010, 111000$ are in L as defined above (you must have u followed by the string obtained by reversing the complement of u), whereas $00101, 11100$ are not.

Solution. L is generated by the following grammar.

$$S \to 1S0 \mid 0S1 \mid \varepsilon.$$

Indeed, L is essentially a language of even-length palindromes with each bit on the right is changed to the opposite bit. □

6 Problem

Show that the language $L = \{10^n10^{2^n} : n > 0\}$ is not context-free. For instance $10100, 10010000, 1000100000000$ are in L.

Solution. Assume L is context-free. Let $p > 0$ be given by the PL. Choose $x = 10^p10^{2^p} \in L$. Suppose $x = abcde$ such that

1. $|bcd| \leq p$,
2. $|bd| > 0$,
3. $ab^icd^ie \in L$ for every $i = 0, 1, 2, \ldots$.

Neither b nor d contains 1's. Otherwise, ab^2cd^2e would contain at least 3 ones, and would not be in L. Hence, b and d contain only zeros. Consider the following cases:

1. b and d contain zeros only from one of the two blocks of zeros. Then, ab^2cd^2e is clearly not in L because we changed the number of zeros in exactly one of the blocks.
2. b and d contain k_1 zeros from the first block of zeros and k_2 zeros from the second block, and $k_1, k_2 \geq 1$. For $x' = ab^2cd^2e$ to be in L, we should have $2^p + k_2 = 2^{p+k_1}$. But $k_2 \leq p$, so

$$2^p + k_2 \leq 2^p + p < 2^p + 2^p = 2^{p+1}.$$

But $k_1 \geq 1$, so $2^{p+k_1} \geq 2^{p+1} > 2^p + k_2$ which is a contradiction, so $x' \notin L$.

In any case, $x' \notin L$ which contradicts the PL, so L is not CF. □

7 Problem

Recall that we call a Turing machine M *not-empty* if and only if there is some input string w such that $M(w) = 1$.

Consider the language $L = \{x \in \{0,1\}^* : x \text{ is the encoding of a Turing machine that is not-empty}\}$. Prove that there is a effective, sound, complete verifier for L. [3 points]

For full-credit you have to describe the verifier (in any high-level programming language or pseudo-code) and write a sentence or two explaining why your verifier satisfies the three properties for L.

Solution. Let V be a TM that, on input $(x = \langle M \rangle, w)$, does the following.

1. For all strings $w_1, w_2, \ldots, w_m$ of length at most $|w|$, run M on w_i for at most $|w|$ steps.
2. If, for some input w_i, M halts and accepts, return 1. Otherwise, return 0.

V is an effective, sound and complete verifier for L.

- There are finitely many strings of length at most $|w|$, and for each of those, we run M for fixed number of steps, so V always halts and outputs 0 or 1, so V is effective.
- Suppose M accepts some input w^* in t steps. Let $k = \max\{|w^*|, t\}$. Then, $V(M, 0^k) = 1$, because V simulates M on all inputs of length at most k (including w^*) for $k \geq t$ steps, so V will finish simulating M on w^* and accept. Hence, V is complete.
- Suppose M doesn't accept any input. Then, $V(M, w) = 0$ no matter what w is, since V accepts only if it finds a string that is accepted by M. Therefore, V is sound.

Alternate solution: It is also possible to define the verifier as follows. Let V be a TM which takes input $(x = \langle M \rangle, w = (y, t))$ and does the following.

1. Run M on input y for t steps.
2. If M halts and accepts y after at most t steps, then return 1. Otherwise, return 0.

A similar proof shows that V is effective, sound, and complete. □

8 Problem

Consider a TM $F : \{0,1\}^* \to \{0,1\}^*$ that always halts which takes an encoding of a TM M as input and produces another encoding TM as some output (i.e., F takes one program as input and then returns another program as output after doing some processing).

Prove that for any such TM manipulator as above, there exists a *fixed point*, that is, there is a TM M such that the TM $F(M)$ (the TM whose encoding is given by the output of F on M) is equivalent to M. [3 points]

Solution. We can use Kleene's recursion theorem for a suitably defined T. Let $T(\langle M\rangle, w) = F(\langle M\rangle)(w)$, i.e., on input (M, w), T transforms M to M' according to F, and then runs M' on w. By the Kleene's recursion theorem, there is a TM K such that $K(w) = T(\langle K\rangle, w)$ for all w. Now, this equation says that $K(w) = F(\langle K\rangle)(w)$.

Thus the Turing machine K is equivalent to the Turing machine $F(K)$. In other words, K is a fixed point of F.

Alternate solution: Let U denote the universal TM and define T to be a TM such that $T(\langle M\rangle, w) = U(F(\langle M\rangle), w)$. That is, T first runs $F(\langle M\rangle)$ to get another TM, M', then simulates M' on input w, using U. Note that $U(F(\langle M\rangle), w) = F(\langle M\rangle)(w)$ for all w (by definition of the universal TM). By KRT there exists K such that

$$K(w) = T(\langle M\rangle, w) = U(F(\langle M\rangle), w) = F(\langle M\rangle)(w)$$

for all w. Now, plug in K for M and observe that the above equality implies

$$K(w) = F(\langle K\rangle)(w)$$

for all w. Thus K is a fixed point for F. □