CECS 329, Exam 2 Spring 2024, Dr. Ebert

Directions: Solve AT MOST SIX problems. Closed Notes but you may use a non-programmable scientific calculator. Solve each problem on a separate sheet of paper. For example, if you decide to solve all six problems, then you should submit six pages, one for each problem. Make sure your name is on each page.

Unit-2 LO's

LO5. Solve the following.

a. What does it mean for a unary function $f(x)$ to be URM-computable? Explain. (5 pts)
Solution. There is a URM program P for which $P(x) = f(x)$ for all $x \in \mathcal{N}$.

b. Provide the instructions of a URM program that computes the function $f(x, y) = x \bmod y$. You may assume that $y \geq 1$. (15 pts)
Solution.

i. $J(1, 3, 9)$
ii. $J(2, 4, 7)$
iii. $J(1, 3, 9)$
iv. $S(3)$
v. $S(4)$
vi. $J(1, 1, 2)$
vii. $Z(4)$
viii. $J(1, 1, 1)$
ix. $T(4, 1)$

c. Describe the role each register plays in computing $f(x)$. (5 pts)
Solution. R_1 and R_2 store the inputs x and y. R_3 counts up to x. R_4 counts up to y and resets each time the count reaches y. Once R_3 counts up to x, whatever is in R_4 should be output, since this represents the remainder.

LO6. Solve the following problems.

a. Provide the URM program P whose Gödel number equals

$$2^{30} + 2^{74} + 2^{112} + 2^{137} - 1.$$

Show all work. (10 pts)
Solution. $P = T(4, 1), J(1, 1, 6), S(10), Z(7)$

b. Let P be a URM program that accepts two inputs x and y. What does it mean for P to be universal? (5 pts)
Solution. P is universal iff $P(x, y) = P_x(y)$ for all $x, y \in \mathcal{N}$.

c. A universal program P_U is simulating a program that has 113 instructions and whose Gödel number is

$$x = 2^3 + 2^{179} + 2^{191} + 2^{196} + 2^{224} + 2^{268} + \cdots + 2^{c_{113}} - 1.$$

If the current configuration of the computation of P_x on some input has encoding

$$\sigma = 2^5 + 2^{14} + 2^{16} + 2^{23} + 2^{28} - 1,$$

then provide the next configuration of the computation *and* its encoding. (10 pts)

Solution. Current configuration: $c = (5, 8, 1, 6, 4)$. $\beta(I_4) = 4$ and so $I_4 = Z(2)$ and next configuration is

$$c_{\text{next}} = (5, 0, 1, 6, 5)$$

which has encoding

$$2^5 + 2^6 + 2^8 + 2^{15} + 2^{21} - 1.$$

LO7. Answer the following. Note: correctly answering two of the three constitutes a pass.

a. Describe what it means to be a positive instance of the `Self Accept` decision problem. (5 pts)

Solution. See Section 5 of the "Undecidability and Diagonalization Method" lecture.

b. In applying the diagonalization method towards proving the undecidability of `Self Accept`, we defined a computable function $g(x)$ in terms of $f(x)$, the decision function for `Self Accept`, which we assume to be total computable. Provide the formula for computing $g(x)$ and describe what needs to be established about $g(x)$ in order for the diagonalization method to be successful. (10 pts)

Solution. See Section 5 of the "Undecidability and Diagonalization Method" lecture. We must show that g is different from each ϕ_i, $i \geq 0$, which contradicts g's own computability.

c. Suppose URM program P_{60} computes $\phi_{60}(x)$ and that

$$\phi_{60}(x) = \begin{cases} \lfloor x/3 \rfloor & \text{if } x \text{ is divisible by 3} \\ \uparrow & \text{otherwise} \end{cases}$$

What is the value of $g(60)$? Show that $g \neq \phi_{60}$. (10 pts)

Solution. Since 60 is divisible by 3, $\phi_{60}(60) = 20$ and so P_{60} halts on its own Gödel number 60. Thus $g(60) = \uparrow$ (see the lecture for a solution to part b). Thus, $g \neq \phi_{60}$ since they differ at input 60.

LO8. An instance of `Self Output` is a Gödel number x and the problem is to decide if P_x outputs its own Gödel number x, i.e. there is an input y for which $P_x(y) = x$. Consider the function

$$g(x) = \begin{cases} 1 & \text{if } P_x \text{ outputs its own Gödel number } x \\ 0 & \text{otherwise} \end{cases}$$

a. Evaluate $g(16)$, $g(78)$, and $g(81)$, where (3 pts each)

i. $\phi_{16}(y) = y^2$.
Solution. $g(16) = 1$ since $\phi_{16}(4) = 16$ and so P_{16} outputs its own Gödel number.

ii. $\phi_{78}(y) = 2y$.
Solution. $g(78) = 1$ since $\phi_{78}(39) = 78$ and so P_{78} outputs its own Gödel number.

iii. $\phi_{81}(y) = 5y$.
Solution. $g(81) = 0$ since there is no y for which $\phi_{81}(y) = 5y = 81$. This is true since 5 does not divide evenly into 81.

b. Prove that the function $g(x)$ is not URM computable. In other words, there is no URM program that, on input x, always halts and either outputs 1 or 0 as output, depending on whether or not P_x outputs its own Gödel number x. Do this by writing a program P that uses g and makes use of the `self` programming construct. (10 pts)

Solution.

Input y.
If $g($`self`$) = 1$, then loop forever.
Else $//g($`self`$) = 0$
    Return `self`.

c. Use a proof by cases to show that your program P from part b behaves inconsistently with how g evaluates P's Gödel number. Conclude that g is not total computable and hence `Self Output` is undecidable. (6 pts)

Solution. Let e denote the Gödel number of P.

Case 1: $g(e) = 1$. Then P outputs e for some input, but this contradicts the fact that P loops forever on *every* input.

Case 2: $g(e) = 0$. Then P never outputs e on some input which contradicts the fact that P outputs e on *every* input.

Therefore, `Self Output` must be undecidable since assuming otherwise leads to a contradiction.

Advanced Problems

A1. Solve/answer the following.

(a) Formally state Kleene's 2nd Recursion Theorem. (10 pts)
Solution. See Page 3 of Self Referencing Programs lecture.

(b) Explain the main application of Kleene's 2nd Recursion Theorem in relation to computer programming. (5 pts)
Solution. The theorem legitimizes and supports the use of the `self` programming construct which allows programming statements that access the Gödel number of the program being written.

(c) The proof of Kleene's 2nd Recursion Theorem involves defining a URM program $P = ABC$, where A, B, and C are subprograms that are concatenated together to form P. Explain the role played by each subprogram. Please include information on the effect each has on the machine's registers R_1 and R_2 (assuming the URM model of computation). (15 pts)
Solution. See Page 5 of Self Referencing Programs lecture.

A2. Answer the following.

(a) Prove that there is a total computable function $k(x)$ for which $\phi_{k(x)}(y) = x$. (10 pts)

Solution. We know that $g(x, y) = x$ is a computable function. Thus, by the SMN Theorem, there is a total computable function $k(x)$ for which

$$\phi_{k(x)}(y) = x.$$

(b) Use the program encoding functions to provide a mathematical formula for the value of $k(x)$ on some input x. Hint: you may find useful the geometric series formula

$$1 + a + a^2 + \cdots + a^k = (a^{k+1} - 1)/(a - 1). \text{ (20 pts)}$$

Solution. Given x, the simplest program that can compute the function $f(y) = x$ is $P = Z(1), S(1), S(1), \ldots, S(1)$, where there are x $S(1)$ statements. Thus,

$$\gamma(P) = 2^0 + 2^2 + 2^4 + \cdots + 2^{2x} = 1 + 4 + 4^2 + \cdots + 4^x = \frac{4^{x+1} - 1}{3}.$$

A3. An instance of `Empty` is a Gödel number x and the problem is to decide if P_x has an empty domain, meaning that it never halts on any input. Alicia wants to use the diagonalization method the prove that `Empty` is undecidable. Letting d_{Empty} denote the decision function for `Empty`, she defines the antagonist function $g(x)$ as

$$g(x) = \begin{cases} 1 & \text{if } d_{\text{Empty}}(x) = 1 \\ \uparrow & \text{otherwise} \end{cases}$$

(a) For a diagonalization proof to work, what must be achieved by $g(x)$? (10 pts)

Solution. We must show that $g(x) \neq \phi_i(x)$ for every $i \geq 0$.

(b) Does $g(x)$ succeed? If yes, provide a proof. Otherwise, provide an example where g fails. (15 pts)

Solution. Suppose $\phi_i(x)$ is undefined on all inputs. Then $\phi_i(i) = \uparrow$ while $g(i) = 1$. Hence, $g \neq \phi_i$. On the other hand, assume $\phi_i(x)$ is *not* undefined for every input x. In other words, $d_{\text{Empty}}(i) = 0$ and so $g(i) = \uparrow$. But it is possible that $\phi_i(i) = \uparrow$ since ϕ_i does not necessarily have to be undefined at $x = i$. Therefore, it is possible that $g = \phi_i$ in which case a contradiction cannot be guaranteed.

A4. Consider the function $m(x)$ which, on input x, returns the least y for which P_y computes function ϕ_x. In other words, P_y is a minimal program for ϕ_x. Prove that $m(x)$ is not a computable function. Hint: assume $m(x)$ is computable and write a program that uses the self-programming construct to create a contradiction. (30 pts)

Solution. Consider the following program P.

Input z.

For each $x = 0, 1, 2, \ldots,$

Compute $y = m(x)$.

If $y >$ `self`, then break.

Simulate P_y on input z and return whatever is returned by P_y on input z.

Note that, since there are an infinite number of minimal programs (why?) there must be some y that exceeds P's Gödel number. Moreover, $P(z) = P_y(z)$ for every z which means that there is a program that computes the same function that is computed by P_y and whose Gödel number is less than y which contradicts the minimality of P_y. Therefore, m is not total computable.

Unit-1 LO's

LO1. Solve the following.

(a) Provide the definition of what it means to be a mapping reduction from decision problem A to decision problem B.

Solution. See Definition 2.1 of Mapping Reducibility lecture.

(b) Let $S = \{7, 12, 15, 19, 21, 35, 47, 48\}$ be an instance of `Set Partition (SP)`. Provide $f(S)$, where $f : \texttt{SP} \rightarrow \texttt{SS}$ is the mapping reduction from `SP` to `Subset Sum` provided in lecture.

Solution. $f(S) = (S, t = M/2) = (S, t = 102)$, where $M = \sum_{s \in S} s$.

(c) Using the problem instances in part b, verify that f maps a positive instance of `SP` to a positive instance of `SS`.

Solution. $A = \{15, 19, 21, 47\}$ and $B = \{7, 12, 35, 48\}$ forms the desired set partion of S, while A's members sum to $t = 102$. Hence, both S and $(S, t = M/2 = 102)$ are positive instances of their respective decision problems.

LO2. An instance of `Composite` is a positive integer $n \geq 2$ and the problem is to decide if there is a number m, such that $2 \leq m < n$ and for which m divides evenly into n.

(a) For a given instance n of `Composite`, describe a certificate in relation to n.

Solution. Certificate m is an integer in the interval $[2, n-1]$.

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

Solution. Return $n \bmod m = 0$.

(c) Which is a better size parameter for instance n: n itself or parameter b that represents the number of bits needed to represent n? Explain.

Solution. b is better since it reflects the actual size of n in terms of number of bits needed to represent n.

LO3. Recall the mapping reduction $f(\mathcal{C}) = (S, t)$, where f maps an instance of `3SAT` to an instance of the `Subset Sum` decision problem. Given `3SAT` instance

$$\mathcal{C} = \{(x_1, \overline{x}_2, x_3), (x_2, x_3, \overline{x}_4), (x_1, x_2, x_4), (\overline{x}_1, \overline{x}_3, \overline{x}_4)\}$$

answer the following questions about $f(\mathcal{C})$. Hint: to answer these questions you are *not* required to draw the table, but you might find it helpful.

(a) What is the value of t?

Solution. $m = |\mathcal{C}| = 4$ and $n = 4$ is the number of variables. Therefore, $t = 11,113,333$ ($n = 4$ 1's followed by $m = 4$ 3's).

(b) How many numbers (counting repeats) are in S? What is the largest (in terms of numerical value) number in S?

Solution. There are $2m + 2n = 16$ (not necessarily distinct) numbers in S. The largest is $y_1 = 10,001,010$

(c) Determine a satisfying assignment for $\mathcal{C}$ and use it to identify a subset A of S that sums to t. List all the members of A. Hint: there are multiple possible answers, but the subset you choose must correspond with your chosen satisfying assignment.

Solution. Solutions may vary. One possible satisfying assignment is $\alpha = (x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1)$. Hence, $f(\mathcal{C})$ is a positive instance of `Subset Sum` via subset $S = \{y_1, y_2, z_3, y_4, g_1, h_1, g_2, h_2, g_4, h_4\}$.

LO4. Answer the following questions. Correctly answering at least two of the three is sufficient for passing LO4.

(a) Provide the definition of what it means for a decision problem to be NP-complete. (6 pts)

Solution. See Complexity Lecture Definition 5.1.

(b) Describe the three main steps that must be completed in order to establish that a decision problem L is a member of NP. Clearly define all technical terms. Hint: your three steps should make reference to two different technical terms that need defining.

Solution. See Complexity Lecture Definition 4.1.

(c) Provide the chain of mapping reductions that is needed to establish that the `Vertex Cover (VC)` decision problem is NP-complete, assuming that IS $\leq_m^p$ VC is one of the reductions, and the other reductions all appeared as examples in either the Mapping Reducibility or Computational Complexity lecture.

Solution. We have

$$\texttt{SAT} \leq_m^p \texttt{3SAT} \leq_m^p \texttt{Clique} \leq_m^p \texttt{IS} \leq_m^p \texttt{VC}.$$