CS 181 Homework 4

Einar Balan

1. Design a TM that recognizes $L = \{x :$ number of 1's in x is at least thrice the number of 0's.$\}$. So for instance $1101, 11110, 011011111 \in L$, whereas $10, 11100 \notin L$. Do not use HOC here but describe the TM in pseudocode as we did in class for Maj. [1 point]

Answer

Pseudocode:

```
1. Scan to the right until a zero is found
2. If no zero is found:
    Clean up the tape and return 1
3. If a zero is found:
    Mark the zero as seen
    Go to start of tape and enter state SearchFirst1
    Scan to the right until a one is found
    If no one is found:
        Clean up the tape and return 0
    If a one is found:
        Mark the one as seen
        Enter state SearchSecond1
        Scan to the right until a one is found
        If no one is found:
            Clean up the tape and return 0
        If a one is found:
            Mark the one as seen
            Enter state SearchThird1
            Scan to the right until a one is found
            If no one is found:
                Clean up the tape and return 0
            If a one is found:
                Mark the one as seen
                Go to start
4. Repeat until all symbols have been seen
```

2. Show that there is a simple constant size TM (or program in your favorite language) *Fermat* such that the program *Fermat* terminates (on any input) if and only if the Fermat's last theorem is false (which we know it's not ... but don't assume that for this problem - just show equivalence). [1 point]

Answer

```
def Fermat(x):
    z = 2
    fermat = True
    while fermat:
        for a in range(1, z):
            if not fermat: break
            for b in range(1, z):
                if not fermat: break
                for c in range(1, z):
                    if not fermat: break
                    for n in range(3, z + 2):
                        if a**n + b**n == c**n:
                            fermat = False
                            break
        z += 1
    return fermat
```

3. Prove that the following function Finite: $\{0,1\}^* \to \{0,1\}$ is uncomputable. On input $P \in \{0,1\}^*$ (the description of the TM), we define Finite(P) = 1 if and only if P is a string that represents a TM such that there are only a finite number of inputs $x \in \{0,1\}^*$ on which the TM outputs 1. [1 point]

   [Hint: One approach is to use a reduction from NOTHALTONZERO.]

Answer

We use KARP reduction, where we use an R which outputs a Turing Machine with the following pseudocode describing it.

Pseudocode:

```
def N(z):
    EVAL(<M>, 0)
    return 1
```

We want to show that for all $\langle M \rangle$, NOTHALTONZERO($\langle M \rangle$) = Finite(R($\langle M \rangle$)).

- if NOTHALTONZERO($\langle M \rangle$) = 1, then N does not halt on any inputs and therefore there is a finite number of inputs for which the TM outputs 1 (there are none). Finite($\langle N \rangle$) = 1
- if NOTHALTONZERO($\langle M \rangle$) = 0, then N has an infinite language since 1 is output for any input and Finite($\langle N \rangle$) = 0

We have shown the reduction holds. So since NOTHALTONZERO is uncomputable (via theorem in class), it must be that Finite is uncomputable.

□

4. Consider the case where $F : \{0,1\}^* \to \{0,1\}$ takes two Turing machines P, M as input and $F(P, M) = 1$ if and only if there is some input x such that P halts on x but M does not halt on x. Prove that F is uncomputable. [1 point]

   [Hint: Use a reduction from HALTONZERO].

Answer

We use KARP reduction, where we use an R which outputs two Turing Machines, N and O, with the following pseudocode.

Pseudocode:

```
def N(z):
    EVAL(<T>, 0)

def O(z):
    while (True): continue
```

We want to show that for all $\langle P \rangle$ and $\langle M \rangle$, HALTONZERO$(\langle T \rangle) = F(R(\langle T \rangle)) = F(\langle N \rangle, \langle O \rangle)$.

- if HALTONZERO$(\langle T \rangle) = 1$ then N halts on all x but O does not halt on all x so $F(\langle N \rangle, \langle O \rangle) = 1$.
- if HALTONZERO$(\langle T \rangle) = 0$ then N does not halt on all x so $F(\langle N \rangle, \langle O \rangle) = 0$.

We have shown the reduction holds. So since HALTONZERO is uncomputable (as shown in class), it must be that F is uncomputable.

□

Practice problems. Do not submit.

1. Design a TM that computes the function $DecbyOne : \{0,1\}^* \to \{0,1\}^*$ that takes the binary representation of an integer as input, and returns the binary representation of the number minus 1. The answer should have the same the number of bits as the input (so you may end up padding with zeros if needed). So for instance, $Decbyone(1100) = 1011$, $Decbyone(0010) = 0001$, $Decbyone(1000) = 0111$. You can assume the input is greater than 0. Do not use HOC here but describe the TM in pseudocode as we did in class for Maj. [1 point]

Answer

```
1. Go to end of tape
2. Scan left for a 1 and at each step:
    If a 1 is found:
        Change 1 to a 0
        Go to end of tape and halt
    If a 0 is found:
        Change 0 to 1
        Continue
    If start of tape reached:
        Go to end of tape and halt
```

2. This problem essentially shows that TMs can actually implement indexable arrays.

   Define a function $Ind : \{0,1\}^* \circ \{\#\} \circ \{0,1\}^* \to \{0,1\}$. That is the input is of the form $i\#x$ where $i \in \{0,1\}^*$, $x \in \{0,1\}^*$. We interpret i as the binary representation of an integer and $Ind(i\#x) = x[i]$. For example, $Ind(0\#101010) = 1$ as the bit in the 0'th position of x is 1. Similarly, $Ind(1\#101010) = 0$, $Ind(11\#101010) = 0$ as the bit in the third position of x is 1. (Remember indexing starts from 0.)

   Give a TM that computes Ind. That is, for instance, when the tape is loaded with $i\#x$, the TM ends with $x[i]$ on its tape. You can assume without loss of generality that the integer whose binary representation is i is less than the length of x (i.e., don't worry about border cases). Do not use HOC here but describe the TM in pseudocode as we did in class for Maj. [1 point]

   [Hint: You can use the idea behind Decbyone as a building ingredient. You can even give a two tape TM if that simplifies your pseudocode. Imagine keeping a separate head at the start of x, how many times do you have to move it to the right to get the right answer?]

Answer

```
1. Scan left for #
2. Once found, copy remaining contents onto second tape
    (i.e. copy x to tape 2)
3. Move tape 2 head to most significant bit on tape
4. Move tape 1 head to least significant bit of i
    (i.e. move until start symbol, then 1 pos right)
5. (Decbyone) On tape 1, scan left for a 1. For each step:
    If a 1 is found:
        Change 1 to a 0
        Go to end of tape and break loop
    If a 0 is found:
        Change 0 to 1
        Continue
6. Move tape 2 head to the right
7. Starting from start of tape 1:
    Scan right for a 1
    If a one is found, repeat from 5
8. Clean up tapes and halt
```

3. Define a function $PowerTwo : 1^* \to 1^*$ that takes as input 1^n and outputs 1^{2^n}. For instance, $PowerTwo(1) = 11$, $PowerTwo(11) = 1111, PowerTwo(111) = 11111111$ and so on. (We do not care about non unary inputs.)

   Describe a TM for computing $PowerTwo$. Describe the TM in pseudocode at a level of detail as we did in class for Maj. You may use multiple tapes.

   [Hint: Try to have two tapes, and every time you move the head of the first tape (that contains the input), you do something on the second tape to double the length.]

Answer

```
1. Two tapes
2. Init tape 1 with input and tape 2 with a single 1
3. Until end of tape 1 is reached:
    (append copy of tape 2 to itself)
    Mark bit on tape 2 as seen
    Go to end of tape 2
    Replace null bit with 0
    Return to most recently seen 1
    Move head 2 right
    Repeat until first 0 is reached
    Replace all 0's with 1's
4. Go to end of tape 2 and halt
```

4. Consider the function $EMPTY : \{0,1\}^* \to \{0,1\}$ that takes a DFA as input and outputs 1 if the language of the DFA is empty. That is, $EMPTY(D) = 1$ if D describes a DFA (under some encoidng - the representation is not important) that does not accept any string. Define $EQUIVALENT : \{0,1\}^* \to \{0,1\}$ as the function that takes two DFAs D, D' and checks their equivalence: that is $EQUIVALENT(D, D') = 1$ if $D(x) = D'(x)$, $\forall x$. Give a reduction from EQUIVALENT to EMPTY. [1 point]

   [You can use high-level programming languages or pseudocode to describe your reduction. By reduction, your goal is to give an algorithm R that takes an input for EQUIVALENT and outputs an input for EMPTY such that the condition for reduction holds: $R((D, D'))$ is a DFA such that EQUIVALENT(D, D') = EMPTY$(R(D, D'))$. As a hint, use the closure properties of DFAs to show that there is a DFA D'' such that D'' is empty if and only if D, D' are equivalent. Then, you can argue that the DFA D" can be produced by an algorithm; you only have to provide high-level explanation for the latter.]

Answer

Use EMPTY to solve EQUIVALENT.

We want EQUIVALENT(D, D') = EMPTY$(R(D, D'))$. Reduction R:

```
1. L = Language(D) - Language(D')
2. Return D for L
```

If D and D' are the same language, then L will always be empty. If they are not the same language, L will contain some .

5. (COMMENTED OUT) Show that computable functions are closed under the "concatenation" operation: If $F, G : \{0,1\}^* \to \{0,1\}$ are two functions, define a new function $H : \{0,1\}^* \to \{0,1\}$ with $H(x) = 1$ if x can be written as $x = x_1 \circ x_2$, $F(x_1) = G(x_2) = 1$; and $H(x) = 0$ if there is no way to break up x as such. Show that if F, G are computable, then so is H. [1 point]

   [Hint: You can give a multi-tape TM for H. That is, you can use two-tapes or three or even four tapes. You don't have to describe how these multi-tape TMs are equivalent to one-tape. Make sure, you give sufficient details to describe how to implement this high-level idea.]

6. (COMMENTED OUT) Consider the function $ToddHalf : \{0,1\}^* \to \{0,1\}$ defined as follows. For a string $x \in \{0,1\}^*$, let $Half(x) = x[0]x[1]\cdots x[\lfloor |x|/2 \rfloor]$; that is, you take the first half (after rounding down) of the string x. Define $ToddHalf(x)$ as follows: If the TM corresponding to x halts on $Half(x)$, then do the opposite; if not, output 0. Formally,

$$ToddHalf(x) = \begin{cases} 1 - \text{ (First bit of } x(Half(x))) & \text{if } x \text{ halts on } Half(x) \\ 0 & \text{otherwise} \end{cases}.$$

   Is $ToddHalf : \{0,1\}^* \to \{0,1\}$ as defined above computable? If yes, describe a Turing machine for it. If not, prove that it is not.