Consider the following program that aims to compute the Toddler function:

```
def infant(M):
     z = Eval(M,M)
     if z[0] == 0:
         return 1
     else
         return 0
```

[Here, $$M$$ is taken as the representation of a TM just as in the definition of the Toddler function from class.]

Does the program above compute the Toddler function? If yes, prove that it does. If not, explain why not and give an example of an input where the output is wrong. (You can use high-level programming languages to write down the input $$M$$.).

|files|

|____|

↳Add **Subquestion**

✚ Add **Question 12**

Save Assignment

Q1 Regexp
1 Point

Write a regular expression to describe the language $L = \{x \in \{0,1\}^* : \text{x does not contain two consecutive zeroes}\}$.

That is, your regular expression should match only those strings that do not contain two consecutive zeroes (e.g., $01, 101, 101101, 010, 1010, \ldots$, etc.,).

Type your expression in the answer field below followed by short (one or two sentences) description of your logic. No need for proof.

answer

Q2 NFA
1 Point

Let M be a NFA and let L be the language accepted by M. Let M' be the NFA with the same transitions and start state as M but whose accept states are the complement of those of M: that is, a state s is an accepting state for M' if and only if it is *not* an accepting state in M.

Let L' be the language accepted by M'. Is L' the complement of L (i.e., is $L' = \{0,1\}^* \setminus L$?)? Explain your answer in one or two sentences.

Q3 Configuration?
1 Point

Define a Turing machine's configuration to contain all the information needed to capture the current status of its computation (i.e., if I paused my computation then wrote the configuration down, I could resume the computation from where I left off using what I wrote down alone). List three necessary components of a Turing machine's configuration.

Q4 Universality
1 Point

Describe the $EVAL$ function we defined in class for Turing machines and state the universality theorem we stated in class.

[Just the definition and theorem. Please type the answer.]

Q5 Reduction
1 Point

Suppose we have a function $F : \{0,1\}^* \to \{0,1\}$ that we want to show is uncomputable. Which of the following two strategies should we use to show that F is uncomputable? Explain your answer in a sentence or two.

- ○ Use a machine that supposedly computes $HALT$ to compute F.
- ◉ Use a machine that supposedly computes F to compute $HALT$.

Q6 Rice's theorem
2 Points

Q6.1 Rice's theorem
1 Point

State a complete definition of semantic properties.

State Rice's theorem.

Q6.2 Semantic?
1 Point

Which of the following functions $F : \{0,1\}^* \to \{0,1\}$ are semantic? [Here, we associate a binary string M with the corresponding TM as in class.]

- ☐ $F(M) = 1$ if M outputs 1 on input $M \circ M$.
- ☑ $F(M) = 1$ if M halts on all inputs of length with at most 2020 bits.
- ☐ $F(M) = 1$ if M on input x computes the parity of x in at most $|x| + 1$ steps.
- ☑ $F(M) = 1$ if the output of M on any input x has more than $|x|$ bits.

Q7 Automaton
3 Points

Let L_1, L_2 be two regular languages recognized by DFAs M_1, M_2 respectively.

Q7.1 Concatenation
1 Point

Describe how you can design a NFA to recognize the concatenation of L_1, L_2, that is $L = \{x : x = x_1 \circ x_2, \text{ where } x_1 \in L_1, x_2 \in L_2\}$.

No files uploaded

Q7.2 Union
2 Points

Describe how you can design a NFA to recognize the union of the two languages, that is $L = L_1 \cup L_2$.

To get full-credit, the number of states in your NFA should be within a constant factor of (number of states for recognizing L_1) + (number of states for recognizing L_2).

[So the reduction we did in class to get a DFA for the union would not work.]

No files uploaded

Q8 Regular?
2 Points

Prove or disprove that the following language is regular: $L = \{0^m 1 0^n 1 0^{mn} | m \geq 1, n \geq 1\}$.

No files uploaded

Q9 Regular?
2 Points

Prove or disprove that the following language is regular: $L = \{0^m 1^n | m \geq n\}$.

No files uploaded

Q10 Design TM
3 Points

Define $Half : \{0,1\}^* \to \{0,1\}^*$ as the first half, rounded-down, of the input. That is for $x \in \{0,1\}^*$, $Half(x) = x[0]x[1] \cdots x[\lfloor |x|/2 \rfloor]$.

So for example, $Half(1001) = 10$, $Half(100) = 1$ $Half(011010) = 011$, $Half(01100) = 01$.

Describe a TM that computes $Half$.

[You can design a one-tape or two-tape TM. Your description should be at a level of detail comparable to what we used in lectures or in homework solutions. But do not just use high-level programming language here. Recall that the output of a TM is what is written to the left of the head including the head on the first tape (excluding the start symbol).]

No files uploaded

Q11 Toddler?
3 Points

Consider the following program that aims to compute the Toddler function:

```
def infant(M):
    z = Eval(M,M)
    if z[0] == 0:
        return 1
```

```
    else
        return 0
```

[Here, M is taken as the representation of a TM just as in the definition of the Toddler function from class.]

Does the program above compute the Toddler function? If yes, prove that it does. If not, explain why not and give an example of an input where the output is wrong. (You can use high-level programming languages to write down the input M.).

No files uploaded