CS 81 (Spring 2018) **Assignment 10** **Name**
Computability and Logic **Due Monday April 16 at 6:00 PM**

Notice: When you submit this assignment, you are certifying therewith that you understand and accept the following policy, which applies to all assignments.

Collaboration Policy: The writeup that you submit must be your own work. You are encouraged to get help from the professor and grutors. You may discuss the problems with classmates, but if you do so, it should be in groups of no more than three. You are not allowed to copy or transcribe solutions from other sources, including the work of other students, the internet, previous solution sets, and images photographed from a whiteboard or blackboard. There is to be no "partnering" where two or more students submit the same writeup. If you get help on a problem, you should say who provided the help on a per-problem basis. Blanket statements such as "worked with John and Mary" are not allowed. Detected infractions may impact your academic career.

Formatting Policy All work must be typeset in electronic media and submitted as a **single pdf file**, one problem on each page as shown in the following pages. Retain this header page. **Handwritten and photographed or scanned work is not allowed.** Do not use inverse video (light typography on dark background). Do not rotate images. **You will not get credit for difficult-to-read submissions.**

Once your document is complete, make a pdf and submit to **Gradescope**.

1. [20 points] Two Turing Machines M and N are equivalent iff they recognize the same language, i.e. $L(M) = L(N)$. Show that the language

$$\{\langle M, N\rangle \mid M \text{ is equivalent to } N\}$$

is neither recognizable nor corecognizable.

[Hint: Choose N appropriately for each case (recognizable, corecognizable) and reduce a known unrecognizable problem to the case in question.]

---

2. [10 points] Show that a language L is decidable iff it is mapping-reducible to some finite language.

---

3. [10 points] Show that if L is a recognizable language, and $L \leq_m L^{\complement}$ (i.e. L is *mapping reducible* to its own complement) then L is decidable.

---

4. [40 points] Classify each of the language below into one of the categories, and state your reason for the classification:

   I. Unrecognizable by Rice's Theorem

   II. Uncorecognizable by Rice's Theorem

   III. Rice's theorem doesn't apply, but it is decidable

   IV. Rice's theorem doesn't apply, and it is not decidable

   The languages are (for brevity, we omit "M is a Turing machine such that" in each case):

$$A = \{\langle M\rangle \mid M \text{ accepts more than 81 strings}\},$$
$$B = \{\langle M\rangle \mid M \text{ accepts not more than 81 strings}\},$$
$$C = \{\langle M\rangle \mid M \text{ has no rejecting control states}\},$$
$$D = \{\langle M\rangle \mid M \text{ uses no more than 81 steps when started on blank tape}\},$$
$$E = \{\langle M\rangle \mid M\ \exists x \in \Sigma^\star,\ M \text{ uses more than 81 steps when started on } x\},$$
$$F = \{\langle M\rangle \mid L(M) \text{ is decidable by some finite-state machine (DFA)}\},$$

   As usual, $L(M)$ is the language recognized by Turing machine M.

---

5. [20 points] Demonstrate by proof of Church's Theorem by converting your Turing machine rules for `tri1` from a08 into Prover9 clauses for resolution and running it with input tape `[1, 1, 1, 1]` in Prover9. It is expected that the tape will contain `[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]` (10 1s) when it halts. Show a second run with a "damaged" version of the machine that does not halt at all. (If you didn't have a correct solution for `tri1`, you may use the one in the provided solutions instead.)

   You may copy the basic simulation clauses from this handout. You will need to make a few changes:

   a. The blank symbol on my rules is set up for `0` as the blank symbol. You might have used a different symbol for blank, in which case you will have to either edit your rules or mine accordingly.

   b. Instead of listing the rules in brackets as in `tm.pro`, each rule is a separate clause, of the form `m(xControl, xRead, yWrite, yControl, yMove).` ending with a period. The rule clauses below give an example.

   c. You should not be using any state or tape symbols that begin with the letters u, v, w, x, y, z as these are recognized as variables in Prover9, and will cause unexpected results.

```
% Turing Machine Simulation in Resolution Logic

% This set of rules is for the Busy-Beaver 3-state machine
% Here 0 is 'blank'.
% (The halting state h does not count in the 3.)

% Problem-Specific Rules
m(a, 0, 1, b, r).
m(b, 0, 0, c, r).
m(c, 0, 1, c, l).
m(a, 1, 1, h, r).
m(b, 1, 1, b, r).
m(c, 1, 1, a, l).

% negated conclusion. You will need to insert your input in the second set of
    brackets.

-reach(s([], a, []), s(x, h, y)).

% End of Problem-Specific Rules

% Transitions of a Turing machine
% non-move rules:
m(xControl, xRead, yWrite, yControl, n) ->
  transition(s(xLeft, xControl, [xRead:xRight]),
             s(xLeft, yControl, [yWrite:xRight])).

% right move rules:
m(xControl, xRead, yWrite, yControl, r) ->
  transition(s(xLeft,          xControl, [xRead:xRight]),
             s([yWrite:xLeft], yControl, xRight)).

% left move rules
m(xControl, xRead, yWrite, yControl, l) ->
  transition(s([xFirst:xLeft], xControl, [xRead:xRight]),
             s(xLeft,          yControl, [xFirst,yWrite:xRight])).

% add more tape on the right:
transition(s(xLeft, xControl, []), s(xLeft, xControl, [0])).
```

```
39
40 % add more tape on the left
41 transition(s([], xControl, xRight), s([0], xControl, xRight)).
42
43 % reachability clauses
44
45 reach(x, x).
46
47 transition(x, y) & reach(y, z) -> reach(x, z).
48
49 x=x. % magic to prevent running out of memory
```
