CS 4820, Spring 2018 **Homework 10, Problem 1**
Name: Rongguang Wang
NetID: rw564
Collaborators: Siyuao Liu (sl2928); Yihao Chen (yc2288)

(1) *(10 points)* Recall that an instance of the halting problem is a string of the form $x;y$ and the goal is to decide if the Turing machine M_x encoded by the string x halts on input y. If M_x halts on y, then $x;y$ is a YES instance. Otherwise, $x;y$ is a NO instance. Let us say that a Turing machine M *fails to solve* the halting problem for instance $x;y$ if it never terminates or if it produces the wrong answer, i.e., it rejects in case that $x;y$ is a YES instance or it accepts in case that $x;y$ is a NO instance.

Because the halting problem is undecidable, we know that for every Turing machine M there exists an instance $x;y$ of the halting problem such that M fails to solve $x;y$. Describe an algorithm to find such an instance. The input to your algorithm is a description of a Turing machine M. For every such input, your algorithm should run for a finite number of steps and output a halting problem instance $x;y$ such that M fails to solve $x;y$.

Solution:

The problem is aiming to design an algorithm which output a halting problem instance $x;y$ such that a Turing Machine M described by x fails to solve $x;y$. The algorithm should take in a description of a Turing Machine M as input and run in a finite number of steps.

By contradiction, M is a Turing Machine and set F has finite element that consisting of all halting problem instances which M fails to solve. We introduce another Turing Machine M' takes $x;y$ as input which obey the following rules. M' first checks whether $x;y$ belongs to F. It accepts or rejects $x;y$ according to whether M_x halts on input y or not. If so, M' outputs the correct halting problem solution for $x;y$. Otherwise, M' simulates M on input $x;y$ and accepts or rejects $x;y$ based on M. If M accepts $x;y$, then M' accepts $x;y$ and vice verse. The checking is realized by encode the set F and the list of correct halting problem solutions for every input $x;y \in F$ into the state set and transition rules of M'. The encoding is possible since F is a finite set. We define that M' accepts all strings $x;y$ that are YES instances of the halting problem and rejects all others, which means M' decides the halting problem. For strings $x;y$ that belongs to F, M' correctly decides the halting problem on $x;y$ because the correct answer is encoded into M'; For strings $x;y$ that do not belong to F, M' correctly decides the halting problem on $x;y$ because M does so.

Suppose that the inputs allows M have no YES transitions for every state but still output YES on all inputs. Thus, the algorithm is simulating M on $x;y$ and allows wrong outputs. The inputs that have corresponding YES output with the given state are not in F set. M halts and continue to operate for the next input until a NO output appears which leads to the finite execution steps. In this case, M enters infinite loops and M' operates to halts and return a YES output. Therefore, we can find at least one instance which is a halting problem instance $x;y$ such that M fails to solve $x;y$. Since all machines fails to decide the halting problem, there exists an instance of the halting problem $x;y$ such that M fails to solve $x;y$.

Pseudo-code for M':

```
Given input string x;y

if M_x halts on y

  output x;y

else

  M’ simulate M on input x;y

    if M accepts x;y

       M’ accepts x;y

    else if M rejects x;y

       M’ rejects x;y
```