York University
CSE 2001 – Unit 5.1 Undecidability
Instructor: Jeff Edmonds

Don't cheat by looking at these answers prematurely.

1. Prove that there is an uncomputable computation problem P_{hard}.

   - Answer: Our goal is to prove that there is an uncomputable computation problem P_{hard},
     i.e. one for which each TM M fails to compute,
     because there in an input I_M on which it gives the wrong answer, i.e. $M(I_M) \neq P_{hard}(I_M)$.
     This is stated using the first order logic statement:
     $\exists P_{hard}\ \forall M\ \exists I_M\ M(I_M) \neq P_{hard}(I_M)$
     We prove this using the game.
     Define P_{hard} to be the problem $\neg Problem_{diagonal}$, defined as
     $\neg Problem_{diagonal}(\text{“}M\text{”}) = 0$ iff $M(\text{“}M\text{”}) = 1$, i.e. M on “M” halts and says “yes”
     (assuming “M” is a valid the description of TM M).
     Continuing the game, let M be an arbitrary TM.
     Define input I_M to be the description “M” of TM M.
     We know M does not accept $\neg Problem_{diagonal}$,
     because it gives the wrong answer on input $I_M = \text{“}M\text{”}$,
     i.e. $M(I_M) \neq P_{hard}(I_M)$.
     This completes the proof that there is an uncomputable computation problem.

   - Answer: Reader's Digest:
     Proving the first order logic statement: $\exists P_{hard}\ \forall M\ \exists I_M\ M(I_M) \neq P_{hard}(I_M)$
     Define problem P_{hard} so that $P_{hard}(\text{“}M\text{”})$ is anything different than $M(\text{“}M\text{”})$.
     Let M be an arbitrary TM.
     Define input I_M to be M's nemesis “M”.
     We win because $M(I_M) \neq P_{hard}(I_M)$.
     This completes the proof that there is an uncomputable computation problem.

2. I could ask something to test if you understand both the statement that the halting problem is undecidable, the first order logic, the intuition, or the proof.

3. Accepting/Enumerating problem: Let $P_{\cup TM} = \{\langle \text{“}M_1\text{”}, \text{“}M_2\text{”}\rangle \mid L(M1) \cup L(M2) \neq \{\}\}$.

   (a) Give a deterministic algorithm that accepts/recognizes $P_{\cup TM}$. Explain why your solution is correct.

   - Answer:
     algorithm $M_{\cup TM}(\text{“}M_1\text{”}, \text{“}M_2\text{”})$
     $\langle$***pre−cond***$\rangle$**:** “M_1” and “M_2” are descriptions of TM.
     $\langle$***post−cond***$\rangle$**:** Halt and Accept iff $L(M1) \cup L(M2) \neq \{\}$.
     Equivalently if $\exists I,\ M_1(I) = yes$ and $M_2(I) = Yes$.
     begin
     loop $\langle I, t\rangle$
     if($M_1(I)$ and $M_2(I)$ both halt and accept on I after t time steps)
     halt and accept
     end loop
     end algorithm

$\langle$"M_1", "M_2"$\rangle$ is a Yes instance,
iff $L(M1) \cup L(M2) \neq \{\}$,
iff $\exists I,\ M_1(I) = yes$ and $M_2(I) = Yes$,
iff there is a tuple $\langle I, t\rangle$ such that $M_1(I)$ and $M_2(I)$ both halt and accept on I after t time steps, (This tuple will eventually be reached.)
iff $M_{\cup TM}$ ("M_1", "M_2") halts and accepts.

See the Assignment 5.0 to know how to loop over these tuples.

(b) Give a deterministic algorithm that enumerates $P_{\cup TM}$. Explain why your solution is correct.

- Answer:

algorithm $M_{\cup TM\ Enumerate}$ ()
$\langle$***pre−cond***$\rangle$**:** No inputs
$\langle$***post−cond***$\rangle$**:** Each yes instance $\langle$"M_1", "M_2"$\rangle$ is eventually printed.
    No no instances are outputted.

```
begin
    loop ⟨"M1", "M2", t⟩
        if( M∪TM ("M1", "M2") halts and accepts after exactly t time steps )
            Print(⟨"M1", "M2"⟩)
    end loop
end algorithm
```

$\langle$"M_1", "M_2"$\rangle$ is a Yes instance,
iff there is a tuple $\langle$"M_1", "M_2", $t\rangle$ such that $M_{\cup TM}$ ("M_1", "M_2") halts and accepts after exactly t time steps, (This tuple will eventually be reached.)
iff $\langle$"M_1", "M_2"$\rangle$ is eventually printed.

See the Assignment 5.0 to know how to loop over these tuples.