CS 70 Discrete Mathematics and Probability Theory
Summer 2019 James Hulett and Elizabeth Yang

DIS 4A

1 Countability and the Halting Problem

Prove the Halting Problem using the set of all programs and inputs

a) Show that the set of all programs are countable.

b) Show that the set of all inputs are countable.

c) Assume that you have a program that tells you whether or not it halts. Since the set of all programs and the set of all inputs are countable, we can enumerate them and construct the following table.

	x_1	x_2	x_3	x_4	...
p_1	H	L	H	L	...
p_2	L	L	L	H	...
p_3	H	L	H	L	...
p_4	L	H	L	L	...
⋮	⋮	⋮	⋮	⋮	⋱

Now write a program that is not within the set of programs in the table above.

d) Find a contradiction in part a and part b to show that the halting problem can't be solved.

Solution:

a) Notice that all programs can be represented by a set of finite length binary strings. The set of finite length binary strings are countably infinite. Therefore the set of all programs are countable.

b) Notice that all inputs can be represented by a set of finite length binary strings. The set of finite length binary strings are countably infinite. Therefore the set of all inputs are countable.

c)
```
procedure P'(x_j)
    if P_j(x_j) halts then
        loop
    else
        halt
    end if
end procedure
```

d) If the program you wrote in part c) exists, it must occur somewhere in our list of programs, P_n. This cannot be. Consider for the sake of contradiction that the P' program exists in the set of all our programs (and thus we can solve the Halting problem). If the P' program exists, then for whatever input we take in, we can call it and P' will do the opposite of what the program P_j would take in. Now consider when P' is some the program for some arbitrary row j. In this sense, we have following, which just degenerates to the Halting Problem.

```
procedure P'(x_j)
    if P'_j(x_j) halts then
        loop
    else
        halt
    end if
end procedure
```

2 Computability

Decide whether the following statements are true or false. Please justify your answers.

(a) The problem of determining whether a program halts in time 2^{n^2} on an input of size n is undecidable.

(b) There is no computer program `Line` which takes a program P, an input x, and a line number L, and determines whether the L^{th} line of code is executed when the program P is run on the input x.

Solution:

(a) False. You can simulate a program for 2^{n^2} steps and see if it halts.

Generally, we can always run a program for any fixed *finite* amount of time to see what it does. The problem of undecidability arises when no bounds on time are available.

(b) True.

We implement `Halt` which takes a program P, an input x and decides whether $P(x)$ halts, using `Line` as follows. We take the input P and modify it so that each exit or return statement jumps to a particular new line. Call the resulting program P'. We then hand that program to `Line` along with the input x and the number of the new line. If the original program halts than `Line` would return true, and if not `Line` would return false.

This contradicts the fact that the program `Halt` does not exist, so `Line` does not exist either.

There are two ways to show undecidability: 1. Use your program as a subroutine to solve a problem we know is undecidable or to do a diagonalization proof like we did for `Halt`. The former is natural for computer programmers and flows from the fact that you are given P as text! Therefore you can look at it and modify it. This is what the solution above does.