CS 70 Discrete Mathematics and Probability Theory
Spring 2026 Sinclair, Song

DIS 6A

1 Computability Intro

Note 11

Computability: The main focus is on the Halting problem, and programs that provably cannot exist.

The *Halting problem* is the problem of determining whether a program P run on input x ever halts, or whether it loops forever. It turns out that there does not exist any program that solves this problem.

Using this information, we can prove that other problems also cannot be solved by a computer program, through the use of *reductions*. The main idea is to show that if a given problem can be solved by a computer program `TestX`, then the Halting problem can also be solved by a computer program `TestHalt` that uses `TestX` as a subroutine.

Here is an example of a template that we will often use for reductions. Suppose we want to show that a program `TestX` does not exist, where `TestX(P', y)` tries to determine whether a program P' on input y does some task $\mathscr{X}$ (i.e. it outputs "True" if $P'(y)$ does the task $\mathscr{X}$, and it outputs "False" if $P'(y)$ does not do the task $\mathscr{X}$). We can define `TestHalt` as follows (in pseudocode):

```
def TestHalt(P, x):
    def P'(y):
        run P(x)
        do 𝒳
    return TestX(P', y)  # for some given y
```

Then we will show that if `TestX` exists, then `TestHalt` would correctly solve the halting problem.

Note that reductions in CS70 will generally look like this template. but more complex reductions will require more sophisticated programs—you'll learn more about this in classes like CS170 and CS172.

(a) Consider the reduction template given above. Let's break down what it's doing.

We follow an argument by contradiction—we assume that there is a program `TestX(P', y)` that is able to determine whether another program P' on input y does some task $\mathscr{X}$.

There are two cases: either $P(x)$ halts, or it loops forever. We'd like to show that `TestHalt` as defined above returns the correct answer in both of these cases.

(i) Suppose $P(x)$ halts. What does `TestHalt` return, and why?

(ii) Suppose $P(x)$ loops forever. What does `TestHalt` return, and why?

(iii) What does this tell us about the existence of `TestX`? Briefly justify your answer.

2 Hello World!

Note 11

Determine the computability of the following tasks. If it's not computable, write a reduction or self-reference proof. If it is, write the program. Throughout this problem, you are allowed to execute programs while surpressing their print statements.

(a) You want to determine whether a program P on input x prints "Hello World!". Is there a computer program that can perform this task? Justify your answer.

(b) You want to determine whether a program P prints "Hello World!" while or before running the kth line in the program. Is there a computer program that can perform this task? Justify your answer.

(c) You want to determine whether a program P prints "Hello World!" in the first k steps of its execution. Is there a computer program that can perform this task? Justify your answer.

3 Code Reachability

Note 11

Consider triplets (M, x, L) where

- `M` is a Java program
- `x` is some input
- `L` is an integer

and the question of: if we execute $M(x)$, do we ever hit line L?

Prove that this problem is undecidable.