Lecture 32

Decidable and Undecidable Problems

Here are some examples of decision problems involving Turing machines. Is it decidable whether a given Turing machine

(a) has at least 481 states?

(b) takes more than 481 steps on input ϵ?

(c) takes more than 481 steps on *some* input?

(d) takes more than 481 steps on *all* inputs?

(e) ever moves its head more than 481 tape cells away from the left endmarker on input ϵ?

(f) accepts the null string ϵ?

(g) accepts any string at all?

(h) accepts every string?

(i) accepts a finite set?

(j) accepts a regular set?

(k) accepts a CFL?

(l) accepts a recursive set?

(m) is equivalent to a Turing machine with a shorter description?

Problems (a) through (e) are decidable and problems (f) through (m) are undecidable (proofs below). We will show that problems (f) through (l) are undecidable by showing that a decision procedure for one of these problems could be used to construct a decision procedure for the halting problem, which we know is impossible. Problem (m) is a little more difficult, and we will leave that as an exercise (Miscellaneous Exercise 131). Translated into modern terms, problem (m) is the same as determining whether there exists a shorter PASCAL program equivalent to a given one.

The best way to show that a problem is decidable is to give a total Turing machine that accepts exactly the "yes" instances. Because it must be total, it must also reject the "no" instances; in other words, it must not loop on any input.

Problem (a) is easily decidable, since the number of states of M can be read off from the encoding of M. We can build a Turing machine that, given the encoding of M written on its input tape, counts the number of states of M and accepts or rejects depending on whether the number is at least 481.

Problem (b) is decidable, since we can simulate M on input ϵ with a universal machine for 481 steps (counting up to 481 on a separate track) and accept or reject depending on whether M has halted by that time.

Problem (c) is decidable: we can just simulate M on all inputs of length at most 481 for 481 steps. If M takes more than 481 steps on some input, then it will take more than 481 steps on some input of length at most 481, since in 481 steps it can read at most the first 481 symbols of the input.

The argument for problem (d) is similar. If M takes more than 481 steps on all inputs of length at most 481, then it will take more than 481 steps on all inputs.

For problem (e), if M never moves more than 481 tape cells away from the left endmarker, then it will either halt or loop in such a way that we can detect the looping after a finite time. This is because if M has k states and m tape symbols, and never moves more than 481 tape cells away from the left endmarker, then there are only $482km^{481}$ configurations it could possibly ever be in, one for each choice of head position, state, and tape contents that fit within 481 tape cells. If it runs for any longer than that without moving more than 481 tape cells away from the left endmarker, then it must be in a loop, because it must have repeated a configuration. This can be detected by a machine that simulates M, counting the number of steps M takes on a separate track and declaring M to be in a loop if the bound of $482km^{481}$ steps is ever exceeded.

Problems (f) through (l) are undecidable. To show this, we show that the ability to decide any one of these problems could be used to decide the halting problem. Since we know that the halting problem is undecidable, these problems must be undecidable too. This is called a *reduction.*

Let's consider (f) first (although the same construction will take care of (g) through (i) as well). We will show that it is undecidable whether a given machine accepts ϵ, because the ability to decide this question would give the ability to decide the halting problem, which we know is impossible.

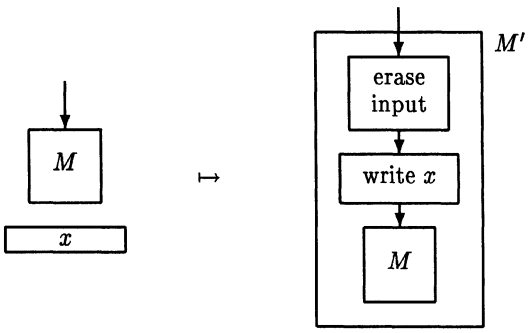
Suppose we could decide whether a given machine accepts ϵ. We could then decide the halting problem as follows. Say we are given a Turing machine M and string x, and we wish to determine whether M halts on x. Construct from M and x a new machine M' that does the following on input y:

(i) erases its input y;

(ii) writes x on its tape (M' has x hard-wired in its finite control);

(iii) runs M on input x (M' also has a description of M hard-wired in its finite control);

(iv) accepts if M halts on x.



Note that M' does the same thing on all inputs y: if M halts on x, then M' accepts its input y; and if M does not halt on x, then M' does not halt on y, therefore does not accept y. Moreover, this is true for every y. Thus

$$L(M') = \begin{cases} \Sigma^* & \text{if } M \text{ halts on } x, \\ \varnothing & \text{if } M \text{ does not halt on } x. \end{cases}$$

Now if we could decide whether a given machine accepts the null string ϵ, we could apply this decision procedure to the M' just constructed, and this would tell whether M halts on x. In other words, we could obtain a decision procedure for halting as follows: given M and x, construct M', then ask whether M' accepts ϵ. The answer to the latter question is "yes" iff M halts

on x. Since we know the halting problem is undecidable, it must also be undecidable whether a given machine accepts ϵ.

Similarly, if we could decide whether a given machine accepts any string at all, or whether it accepts every string, or whether the set of strings it accepts is finite, we could apply any of these decision procedures to M' and this would tell whether M halts on x. Since we know that the halting problem is undecidable, all of these problems must be undecidable too.

To show that (j), (k), and (l) are undecidable, pick your favorite r.e. but nonrecursive set A (HP or MP will do) and modify the above construction as follows. Given M and x, build a new machine M'' that does the following on input y:

(i) saves y on a separate track of its tape;

(ii) writes x on a different track (x is hard-wired in the finite control of M'');

(iii) runs M on input x (M is also hard-wired in the finite control of M'');

(iv) if M halts on x, then M'' runs a machine accepting A on its original input y, and accepts if that machine accepts.

Either M does not halt on x, in which case the simulation in step (iii) never halts and M'' never accepts any string; or M does halt on x, in which case M'' accepts its input y iff $y \in A$. Thus

$$L(M'') = \begin{cases} A & \text{if } M \text{ halts on } x, \\ \varnothing & \text{if } M \text{ does not halt on } x. \end{cases}$$

Since A is neither recursive, CFL, nor regular, and $\varnothing$ is all three of these things, if one could decide whether a given TM accepts a recursive, context-free, or regular set, then one could apply this decision procedure to M'' and this would tell whether M halts on x.