21 April 2021 — Undecidable problems

Announcements

Wellness Days 4/23–4/26.
No lectures, office hours, or recitations on these days.
Problem Set 8 due 4/29.
Problem Set 9 (last one) will be due 5/13.
Reminder: lowest homework grade will be dropped (elevated to perfect score).

The Halting Problem.

Given input string in the form $x\#y$ where x is a description of a TM M_x and y is a description of an input to x, decide whether M_x halts (i.e. either accepts or rejects) when processing input y.

algorithm should accept $x\#y$ if M_x halts on y

It should reject $x\#y$ if M_x loops on y.

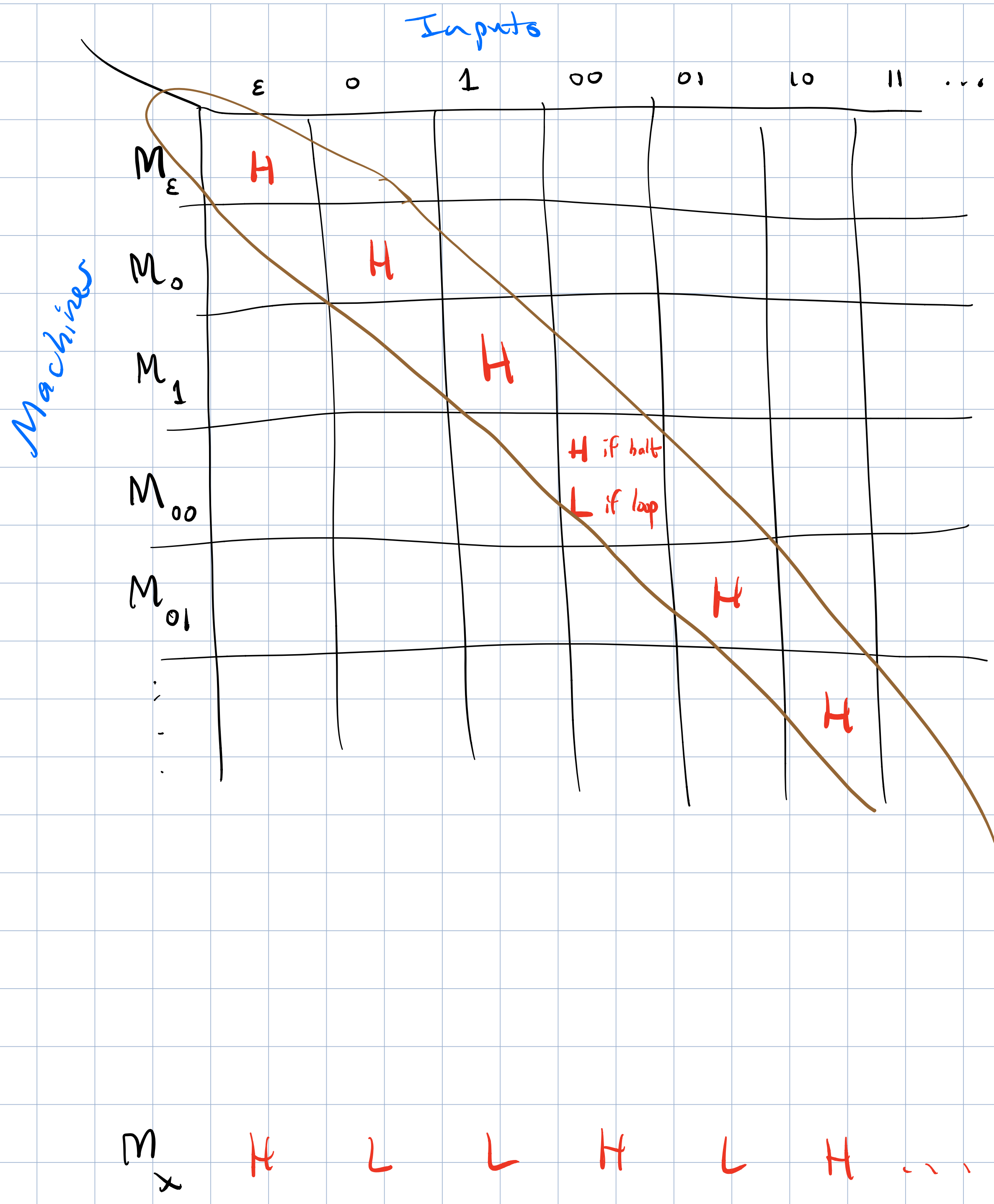
To prove HP (halting problem) cannot be solved by a Turing machine we will use <u>diagonalization</u>.

Idea will be a proof by contradiction: if there is a Turing machine K that decides HP we will construct another TM, M, whose behavior contradicts itself. (There is an input, y, such that M must halt on input y but it must also loop on input y.)

To describe this M it helps to visualize an infinite 2D table with rows corresponding to all the Turing machines.

M_x is the Turing machine whose description is the string x.

Convention: if x is not a TM descrip. then M_x is a trivial machine with one state that immediately rejects.



Inputs
ε 0 1 00 01 10 11 ...
Machines
M_ε
M_0
M_1
M_{00}
M_{01}
H
H
H
H if halt
L if loop
H
H
M_x H L L H L H ...

The string of H's and L's on the diagonal of this table has an interesting property:

If I "flip" each symbol of the string (change H to L, L to H) I obtain an infinite string that does not appear as any row of the table.

If FD denotes this "flipped diagonal string" the first symbol of FD differs from the first symbol of Row 1. The 2nd symbol of FD differs from 2nd symbol of row 2.

⋮

The r^{th} symbol of FD differs from the r^{th} symbol of row r, for all r.

Now what I want to do is construct a machine M, whose "halting vs. looping" behaviour is described by the infinite string FD.

In other words M halts or loops on input y if and only if the symbol in position y of the FD string is H or L, respectively.

This means that on input y, M must:

a. halt if M_y loops on input y

b. loop if M_y halts on input y.

Here's how M works: given input y, first write $y\#y$. Then run Turing machine K on input $y\#y$. (Recall K accepts every $x\#y$ s.t. M_x halts on y, rejects every other $x\#y$.)

→ If K accepts $y\#y$, M enters a state where it loops forever.

→ If K rejects $y\#y$, M accepts y.

To see why this is contradictory: let x be the description of M. Let's ask: does M halt or loop on input x?

If M halts on x, then K must accept $x\#x$. But then, by def'n of M, we know M must loop.

If M loops on x, then K must reject $x\#x$. By def'n of M, this means M accepts x.

In both cases we reach a contradiction.

<u>Conclusion.</u> There is no Turing machine that decides the HP.

<u>Recall:</u> A set of strings, L, is recursively enumerable (r.e.) if $\exists$ a TM M s.t.

$$M \text{ accepts } x \iff x \in L.$$

We say L is recursive if $\exists$ TM M that halts on every input and accepts $x \iff x \in L$.

The set $\{x\#y \mid M_x \text{ halts on input } y\}$ is <u>not</u> <u>recursive</u>. (We just showed that.)

But it <u>is</u> r.e.

If U denotes a universal Turing machine then modify U to a machine V that is the same as U except when U transitions to its reject state, V transitions to accept.

Then

$$\begin{aligned} V \text{ accepts } x\#y &\iff U \text{ accepts or rejects } x\#y \\ &\iff M_x \text{ accepts or rejects } y \\ &\iff M_x \text{ halts on } y. \end{aligned}$$

Summary. HP is r.e. but not recursive.

Theorem. Let $\text{co-HP} = \{ x\#y \mid x\#y \notin HP \}$.

Then co-HP is not r.e.

This is actually a corollary of a stronger theorem....

Theorem A set of strings, L, is recursive if and only if L and its complement are r.e.

Proof. If L is recursive, and M is a Turing machine that decides L, let M' be a Turing machine that's same as M but with accept, reject states swapped.

Then
$x \in L \iff M$ accepts x
$x \notin L \iff M$ rejects x
$\iff M'$ accepts x

L is r.e.
$\bar{L}$ is r.e.

Suppose L and $\bar{L}$ are both r.e.
So we have M_0, M_1 s.t.
$x \in L \iff M_0$ accepts x
$x \notin L \iff M_1$ accepts x.

Design M that always halts, and accepts $x \iff x \in L$.

Pseudocode for M

for $t = 1, 2, 3, \ldots$

simulate M_0 running for t steps on input x

simulate M_1 running t steps on x

if M_0 accepts x within t steps:

accept

if M_1 accepts x within t steps:

reject

Since HP is r.e. but not recursive, we know

HP and co-HP aren't <u>both</u> r.e.

$\therefore$ co-HP cannot be r.e.

<u>Summary.</u> (1) HP is r.e., not recursive.

(2) co-HP is not r.e.