21 April 2021  Undecidable problems

Announcements

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. An 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 diagonalization.

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.
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 rth symbol of FD differs from the rth 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 $x \neq x$. But then, by def'n of $M$, we know $M$ must loop.

If $M$ loops on $x$, then $K$ must reject $x \neq x$.

By def'n of $M$, this means $M$ accepts $x$.

In both cases we reach a contradiction.

Conclusion: There is no Turing machine that decides the HP.

Recall: A set of strings, $L$, is recursively enumerable (r.e.) if

$\exists$ a TM $M$ s.t.

$M$ accepts $x \iff x \in L$.

We say $L$ is recursive if

$\exists$ a TM $M$ that halts on every input and accepts $x \iff x \in L$.

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

But it is 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

$V$ accepts $x \# y \iff U$ accepts or rejects $x \# y \iff M_x$ accepts or rejects $y \iff M_x$ halts on $y$.

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

Theorem. Let $\text{co-}VHP = \{ x \# y \mid x \# y \notin HP \}$. Then $\text{co-}HP$ is not r.e.

This is actually a corollary of a stronger theorem: ...
\textbf{Theorem.} A set of strings, \( L \), is recursive if and only if \( L \) and its complement are r.e.

\textbf{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 \( x \in L \iff M \text{ accepts } x \)

\( L \) is r.e.,

\( x \notin L \iff M \text{ rejects } x \)

\( L \) is r.e.,

\( x \notin L \iff M' \text{ accepts } x \)

Suppose \( L \) and \( \overline{L} \) are both r.e.,

So we have \( M_0, M_1 \) s.t.

\( x \in L \iff M_0 \text{ accepts } x \)

\( x \notin L \iff M_1 \text{ accepts } x \).

Design \( M \) that always halts,

and accept \( x \iff x \in L \).
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 both r.e.

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

Summary:
(1) $HP$ is r.e., not recursive.
(2) co-$HP$ is not r.e.