20 April 2018 Undecidability, continued

Three entities that relate to one another.

Turing machine — String that describes M — Set of all inputs M accepts

$M \xrightarrow{\text{one-to-one}} \langle M \rangle \xrightarrow{\text{not one-to-one}} L(M)$

One thing that is done in lecture notes:

$$L(x) = \begin{cases} \emptyset & \text{if } x \text{ is not a TM description} \\ L(M) & \text{if } x \text{ is the description of a TM, } M. \end{cases}$$

via this mapping, every string x describes a set of strings, and all r.e. sets have such a description.

In this notation, $D \triangleq \{ x \mid x \notin L(x) \}$.

"=" set of all Turing machines that don't accept their own description.

There is no M such that $D = L(M)$ because if there were such M, and its description were $\langle M \rangle$, then

M accepts $\langle M \rangle \Rightarrow M$ accepts its own descript. $\Rightarrow \langle M \rangle \notin D \Rightarrow D \neq L(M)$

M doesn't accept $\langle M \rangle \Rightarrow M$ doesn't " " " " $\Rightarrow \langle M \rangle \in D \Rightarrow D \neq L(M)$

Halting Problem is Undecidable

Recall $H = \{ \langle x; y \rangle \mid x \text{ is a descript of TM, } M, \text{ and } M \text{ halts on input } y \}$.

Reduce "accept z if and only if $z \in D$" to "decide whether $\langle x; y \rangle \in H$."

Terminate and say yes if $z \in D$. Do anything else, possibly run forever, if $z \notin D$.

Always terminate. Say yes if $\langle x; y \rangle \in H$, no if $\langle x; y \rangle \notin H$.

An algorithm to solve the problem on the left, given a machine M_H that decides the set H, is the following:

On input z, pass the string $\langle z; z \rangle$ to M_H and wait for answer.

If $M_H(\langle z; z \rangle) = \text{no}$, then $z \notin L(z) \Rightarrow z \in D$, answer "yes."

If $M_H(\langle z; z \rangle) = \text{yes}$, then simulate machine $M(z)$ that z describes on input z, answer "yes" if and only if it doesn't terminate in the "yes" state.

Another undecidability example

symbol that programs are not supposed to overwrite.

Consider Turing machines with alphabet $\Sigma = \{\emptyset, 1, *\}$.

Call M "insecure" if $\exists$ input string z containing at least one $*$ such that when M runs on input z, it overwrites at least one $*$ with $\emptyset$ or 1.

↳ at any time during execution. ↳ That was present in the input.

Problem: Given x, is x the description of an insecure Turing machine?

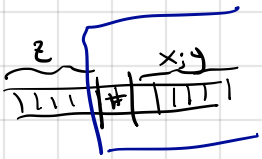
This decision problem is undecidable, by reduction from the halting problem!

Assume (by way of contradiction) that $\exists$ Turing machine M_I s.t. $M_I(x) = yes$ if x is a descr. of insecure TM,
$M_I(x) = no$ otherwise.

Derive contradiction by building M_H to decide H using M_I.

Road map: decide whether $x;y \in H$ (i.e. whether $M(x)$ halts on y) by checking whether a machine that simulates $U(x;y)$ and then overwrites its tape with $\emptyset$'s can ever overwrite a $*$.

Let M_H be a machine that does the following.
On input $x;y$ it writes a description of a Turing machine, M, whose behavior can be described as follows.



1. On any input z, find first $\sqcup$ symbol after z.
2. Append separator character, #, followed by the string $x;y$.
3. Run universal Turing machine U, modified to treat # as if it were $\triangleright$.
   U is also modified so that {halt, yes, no} are remapped to a special "done" state of M.
4. When M reaches "done", it goes back to $\triangleright$, and overwrites every symbol on tape with $\emptyset$.

Checks if the description of M is accepted by M_I.