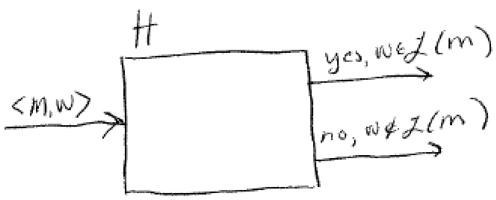
Theory of Computation, CSCI 438 spring 2021
Undecidability, 201-202, March 22nd

1. Prove that A_{TM} = {<M, w> | M is a TM, w is a string in the alphabet of the TM and $w \in \mathcal{L}(M)$} is not decidable? (I.e. is the acceptance problem for TM decidable?)

Proof:
Suppose, by way of contradiction, that $\mathbf{A_{TM}}$ is decidable. Then there is a Turing machine, H, that decides it. That is for input <M,w>

H accepts if $w \in \mathcal{L}(M)$ and

H rejects if $w \notin \mathcal{L}(M)$.



Consider the new Turing machine D, which takes a single input <M> that encodes a Turing machine, feeds two copies of <M> into H, and then flips the result, so that when H accepts, D rejects; and when H rejects, D accepts.

Turing-machine D:
D= "On input <M> which encodes a TM
1. Make another copy of <M> on the tape to feed <M,M> into H.
2. If H accepts, reject. If H rejects, accept."

Clearly, D is a legal Turing machine.
(Draw D, with H inside.)

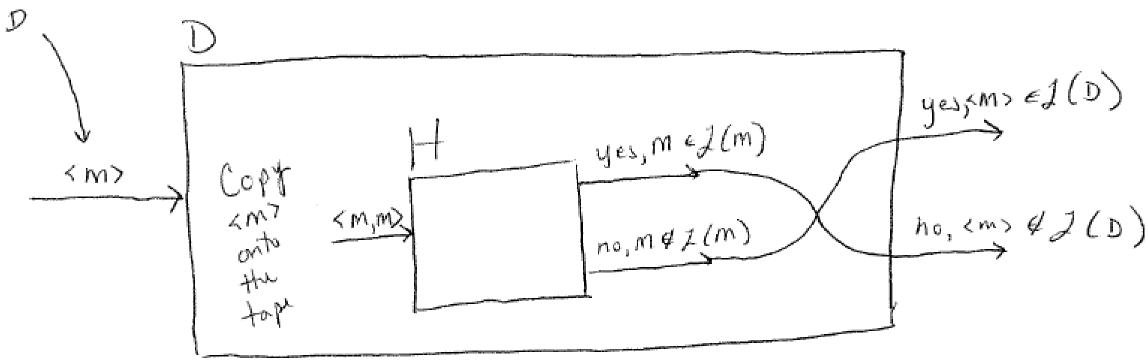
However, if the encoding of D, is fed into D, we get:

<D> $\in \mathcal{L}(D)$ when <D> $\notin \mathcal{L}(D)$ and

<D> $\notin \mathcal{L}(D)$ when <D> $\in \mathcal{L}(D)$.

This is impossible. Thus, $\mathbf{A_{TM}}$ must not have been decidable.

Show this by using another color marker, and show that for at least one input of my new machine D, the input <D>, the machine doesn't make sense.



In a world of saints and thieves (or Greeks and Cretans) where saint always tell the truth and liars always lie, you meet someone who says "I am a liar". Are they a saint or a thief? They can't be either. It is nonsensical in the same idea as the above.

2. Prove that A_{TM} is Turing-recognizable.

A_{TM} = {<M,w> | M is a TM, w is a string in the alphabet of M, and M accepts w}

Proof: The following TM that recognizes A_{TM}.

M = "On input <M,w> that encodes a TM followed by a string in the alphabet of M

1. Run M on w.
2. If M accepts, accept; otherwise reject or loop forever."

Exercise 4.10

Let $INFINITE_{DFA}$ = { <A> | A is a DFA and $\mathcal{L}(A)$ is an infinite language}. Show that $INFINITE_{DFA}$ is a decidable language.

This language is decidable.

If a state is visited twice (which can be determined by a marking algorithm) it seems like the language would be infinite. However, the twice visited state may not lead to an accepting state. So a marking algorithm doesn't work.

Another idea is the following:

M = "On input <A> that encodes a DFA

1. Let k be the number of states in A
2. Generate all strings in Σ^{k+1}, one by one. For each:
   a. Simulate the DFA on the string. If the DFA accepts the string; M accepts; otherwise continue.
3. If no string causes an accept; reject"

However, there may be no accepted string of length k+1, but there is an accepted string longer than k+1.

The following Turing machine decides $INFINITE_{DFA}$:

M = "On input <A> that encodes a DFA

1. Let k be the number of states in A
2. Construct a DFA D that accepts all strings of length k or more. (Note that this is a mechanical process accomplishable by a Turning machine.)
3. Construct a DFA for the intersection of A and the new DFA, D. (Note that this is a mechanical process accomplishable by a Turing machine.)
4. If the intersection is empty, reject; otherwise, accept. (If there are no strings in the intersection, the language is finite. If there is at least one string in the intersection, this string is pumpable and the language is infinite.)"

Another Turing machine that decides $INFINITE_{DFA}$:

M = "On input <A> that encodes a DFA

1. Convert the DFA to a regular expression.
2. Accept if the regular expression contains a Kleene closure; otherwise reject.

Thank you Jake Michelotti!

Problem 4.21 (page 212)

4.21 Let S = {<M> | M is a DFA that accepts w^R whenever it accepts w}. Show that S is decidable.

Note that how S is defined it must be true that $\mathcal{L}(S) = \mathcal{L}(S)^R$.

The following decides S.

D = "On input <M> where M is a DFA

1. Construct an NFA M' which recognizes the same language as M except has a single accept state
2. Construct an NFA M'' which recognizes strings which are the reverse of the strings recognized by M and M' by making the single accept state be the start state, the start state be the only accept state, and reversing all of the transitions
3. Construct a DFA M''' which recognizes the same strings as M'' (using the algorithm we used in class back when studying NFAs)
4. Run the decider for $EQ_{M,M'''}$
5. If the decider accepts, accept; if it rejects/reject.