CSE 105
THEORY OF COMPUTATION

"Winter" 2018

http://cseweb.ucsd.edu/classes/wi18/cse105-ab/

Today's learning goals

Sipser Ch 5.1

- Define and explain core examples of computational problems, include A_{**}, E_{**}, EQ_{**}, $HALT_{TM}$ (for ** either DFA or TM)
- Explain what it means for one problem to reduce to another
- Use reductions to prove undecidability (or decidability)

Decidable	Undecidable
A_{DFA}	A_{TM}
E_{DFA}	A_{TM}^{C}
EQ_{DFA}	$HALT_{TM}$

Using reduction to prove undecidability

Claim: Problem X is undecidable

Proof strategy: Show that A_{TM} reduces to X.

Alternate Proof strategy: Show that $HALT_{TM}$ reduces to X.

etc.

In each of these, have access to Genie which can answer questions about X.

E_{TM} = { <M> | M is TM, L(M) is empty}

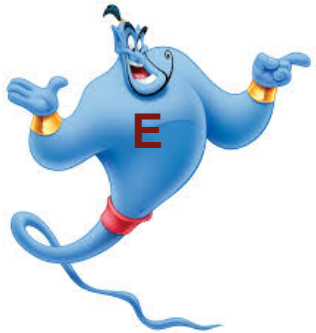
A. Decidable
B. Undecidable
C. No way to tell

Give an example of a string in E_{TM} , and a string not in E_{TM}

Claim: A_{TM} is no harder than E_{TM}

In other words: we could use E_{TM} to solve A_{TM}

Given: Turing machine M, string w, magic genie for E_{TM}



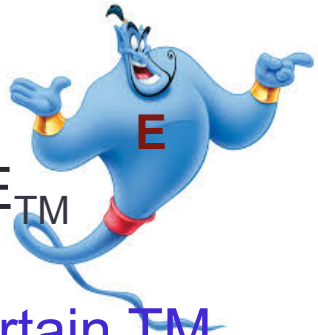
Goal: Accept if w is in L(M); Reject if w is not in L(M).

If we could prove this claim…

Since A_{TM} is undecidable, we conclude that E_{TM} is too.

Claim: A_{TM} is no harder than E_{TM}

In other words: we could use E_{TM} to solve A_{TM}



Given: Turing machine M, string w, magic genie for E_{TM}

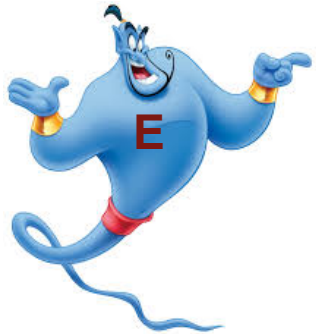
We can ask the magic genie "Is the language of a certain TM empty?" Genie will **magically** give correct yes/no answer.

We can ask the genie as many of these questions as we'd like, about *any* TM.

Goal: Accept if w is in L(M); Reject if w is not in L(M).

Claim: A_{TM} is no harder than E_{TM}

In other words: we could use E_{TM} to solve A_{TM}



Given: Turing machine M, string w, magic genie for E_{TM}

"On input <M,w>

1. Ask Genie about M.
2. If Genie says no, then reject; if Genie says yes, run M on w.
   a. If this computation accepts, accept.
   b. If this computation rejects, reject."

Goal: Accept if w is in L(M); Reject if w is not in L(M) ?

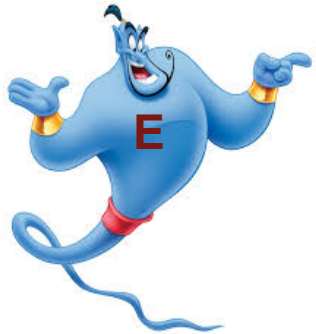
Genies are not all-powerful

Genies can answer **their specific question** magically

To exploit their magic, we sometimes need to build **new "auxiliary" Turing machines** which hardcode information about M,w and then can check the properties of these new machines with our Genie.

Claim: A_{TM} is no harder than E_{TM}

In other words: we could use E_{TM} to solve A_{TM}



Given: Turing machine M, string w, magic genie for E_{TM}

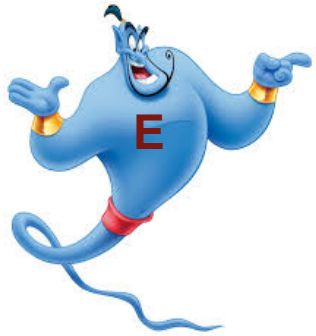
"On input <M,w>

1. Build a new Turing machine X:
   X = "On input x, 1. Run M on w. If accepts, accept; if rejects, reject."
2. Ask Genie about X.
3. If Genie says no, then _____; if Genie says yes, then _____.

Goal: Accept if w is in L(M); Reject if w is not in L(M) ?

Claim: $HALT_{TM}$ is no harder than E_{TM}

In other words: we could use E_{TM} to solve $HALT_{TM}$



Given: Turing machine M, string w, magic genie for E_{TM}

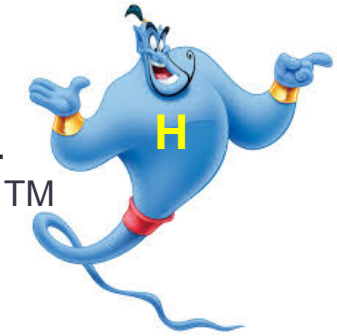
"On input <M,w>

1. Build a new Turing machine X:
   X = "On input x, 1. Run M on w. _____"
2. Ask Genie about X.
3. If Genie says no, then _____; if Genie says yes, then _____.

Goal: Accept if M halts on input w; Reject if M loops on input w?

Claim: E_{TM} is no harder than $HALT_{TM}$

In other words: we could use $HALT_{TM}$ to solve E_{TM}



Given: Turing machine M, string w, magic genie for $HALT_{TM}$

"On input <M>

1. Build a new Turing machine X:
   X = "On input x, _____"
2. Ask Genie about X.
3. If Genie says no, then _____; if Genie says yes, then _____.

Goal: Accept if L(M) is empty; Reject if L(M) is nonempty?

Reduction does not mean reduction

Caution: Section 5.2, 5.3 won't be covered in CSE 105.

Mapping reducibility from Section 5.3 is ***different*** from the reductions we see in Section 5.1.

The results from 5.3 do not necessarily carry over to the reductions from 5.1.

Next time

Pre-class reading Theorems 5.3, 5.4