Sipser's Problem Naming Scheme:

A: "Does $M$ accept $x$?"
- accept
E: "Is $L(M) = \emptyset$?"
NE: "Is $L(M) \neq \emptyset$?"
ALL: "Is $L(M) = \Sigma^*$?"

Two givens:
E: for empty?

Has just one given: an encoding of a machine $M$

NE for non-emptiness: $(\exists x) x \in L(M)$?

$x$ is quantified: not a give

Empty $\emptyset$: Given $M_1$ and $M_2$, is $L(M_1) \cap L(M_2) = \emptyset$?

EQUAL: Given $M_1$ and $M_2$, is $L(M_1) = L(M_2)$?

The above problems talk only about accept/reject and can be worded the same:

to be about regexps $r$ or grammars $G$.

More specific to machines, we ask:

HALT (HP): Given $M$ and $x$, does $M(x) \downarrow$?

TOT: Given just $M$, does $M$ halt for all inputs? $(\forall x) M(x) \downarrow$?

How Problems are Specified:

A DFA $M$ and an $x \in \Sigma^*$

Input: A DFA $M$ and an $x \in \Sigma^*$

QUERY: Does $M$ accept $x$?

Name of problem: often synonymous with the language $L(M)$ of the machine

Language is

$L = \{ z = \langle M, x \rangle : \text{the answer is yes, i.e. } M \text{ accepts } x \}$
Algorithm to decide A DFA problem/ling.

0. Given DFA M and \( x \in \Sigma^* \) (M = (Q, \Sigma, \delta, s, F))

1. Run M on \( x \). M must halt within \( |x|+1 \) of its own steps.

2. If M accepts \( x \), you accept \( \langle M, x \rangle \). If not, you reject \( \langle M, x \rangle \).

Since your algorithm is total, the A DFA language is decidable.

The code in both cases can be the "Turing Kit" Turing Kit is not a decider — not total — but does show that the AOA language is computably enumerable.

We will see next week that AOA is not in \( \text{DEC} \).

But A DFA is decidable, because when M is a DFA, both M and Turing's Kit on M are guaranteed to halt.
APDA: Input: A DPDA $M = (Q, \Sigma, \Gamma, \delta, \epsilon, q_0, F)$ and an $x \in \Sigma^*$.

Question: Does $M$ accept $x$?

ANFA:

Instance: An NFA $N$, an $x \in \Sigma^*$.

Question: Does $N$ accept $x$?

Can convert $N$ to an equivalent DFA $M$, then run $M(x)$.

But this can incur exponential blowup in time.

Can solve in $\text{poly}(|x|)$ time by simulating $N(x)$ directly. Keeping track of which states are currently lit.

This is how UNIX `grep` and scripting langs solve DFA.

ENFA:

Instance: An NFA $N$.

Question: Is $L(N) \neq \emptyset$?

$L(N) \neq \emptyset \iff N$ has a path from $s$ to some state in $F$.

Algorithm:

- Do Breadth-First Search starting from $s$.
- This is a decider. Must halt within $|Q|$ iterations, $m = |Q|$.
- Accept $\langle N \rangle$ iff some state in $F$ is found.
How about ALL DFA? \[ \frac{I: \text{ A DFA } M=(Q, \Sigma, \delta, s, F)}{Q: \text{ asks } L(M) = \Sigma^* ?} \]

**Decider:**

1. Given \( M \), first convert \( M \) into \( M' \) s.t. \( L(M') = \sim L(M) \) i.e. \( \Sigma' \setminus L(M) \).
   - Hence \( L(M) = \Sigma' \Leftrightarrow L(M') = \emptyset \Leftrightarrow \langle M' \rangle \notin \text{NBDFA} \)

2. Feed \( \langle M' \rangle \) to your algorithm for \( \text{NBDFA} \). If it accepts, you reject.

So this is a correct decider. (Idea for we reduced ALL DFA chs. to the \( \text{NBDFA} \) problem)

How about ALL NFA?

Possible decider in flowchart form.

Decider, but not in \( \text{poly}(n) \) time.

How about ALL DPOA?

How about ALL NPDA and ALL CFG?

In fact, \text{ THERE IS NO DECIDER! }