Last time I covered an algorithm for the problem.

**INSTANCE:** A DFA $M$  
**QUESTIONS:** Is $L(M) \neq \emptyset$? Both the problem and the language.

A fixed encoding whose details don't matter.

$\mathsf{NE} \mathsf{DFA} = \{ \langle M \rangle : M \text{ is a DFA and } L(M) \neq \emptyset \}$

$\mathsf{NE} \mathsf{NFA} = \{ \langle N \rangle : N \text{ is an NFA and } L(N) \neq \emptyset \}$

**Theorem:** $\mathsf{NE} \mathsf{NFA}$ is decidable (pretty efficiently!)

**Proof:** Given an NFA $N$, $L(N) \neq \emptyset$ if and only if there is a path from its start state $s$ to some final state $f$. The chain (and $\epsilon$s) along that path (concatenated together into an $x \in L(N)$). We can decide the existence of a path by doing Breadth-First Search starting at $s$. (Unlike with $\mathsf{NFA} \rightarrow \mathsf{DFA}$, this search is only in the NFA graph, so there is no possible "exponential explosion." )
ALL DFA

\[ \text{INST: A DFA } M = (Q, \Sigma, \delta, s, F) \]

\[ \text{QUES: Is } L(M) = \Sigma^*? \]

**Thm:** \text{ALL DFA is decidable (and efficiently)}

**Algm:** Given \( M \), 1. Construct the complement DFA \( M' = (Q, \Sigma, \delta, s, Q \setminus F) \).

Thus \( L(M') = \Sigma^* \setminus L(M) \), so \( L(M) = \emptyset \iff L(M') = \emptyset \).

2. Run our algm for \text{NE DFA} on \( \langle M' \rangle \).
3. Answer yes about \( M \) iff the algm says no on \( M' \).

※ We have "reduced" \text{ALL DFA} to the (complemented) problem \text{EQ DFA}:

\[ \text{EQ DFA} = \{ \langle N' \rangle : N' \text{ is a DFA and } L(N') = \emptyset \} \]

\[ \text{ALL DFA} = \{ \langle N \rangle : N \text{ is an NFA and } L(N) = \Sigma^* \} \]

**Thm:** \text{ALL NFA is decidable (but not so efficiently as)}

**Proof:** Convert \( N \) to \text{DFAM} st. \( L(M) = L(N) \), run \text{ALL DFA} on \( M \).

Not always "efficient." Later we will see that \text{ALL NFA} is an NP-hard problem.

"EQ NFA, regexp": \text{INST: An NFA } N \text{ and a regexp } \( r \)

\[ \text{QUES: Is } L(N) = L(r)? \]

Note that \text{ALL NFA} is the special case where we're given \( r = (a+b)^* \).

So if this problem had an efficient decider, so would \text{ALL NFA}.
Algorithm: 1. Convert $r$ into an NFA $N_r$ s.t. $L(N_r) = L(r)$.
2. Convert both $N$ and $N_r$ into equivalent DFAs $M_2$ and $M_2$.

Then $L(N) = L(r) \iff L(N) = L(N_r)$. We have reduced the goal to an instance of DFA.

$I_2$: We can build a DFA $M_3$ such that $L(M_3) = L(M_1) \Delta L(M_2) = \emptyset$ via the (Cartesian Product) construction.

$L(M_3) = \emptyset$ when $M_3$ is obtained via "Contr. Pred. for $\Delta$".

$L(M_3) = \emptyset$ is a yes-instance of the EDFA problem.

3. Convert $M_1$ and $M_2$ into a DFA $M_3$ s.t. $L(M_3) = L(M_1) \Delta L(M_2)$.

Note: This is fairly efficient: time $\alpha \cdot (\text{size of } M_1) \cdot (\text{size of } M_2)$.

4. Run the EDFA alg on $M_3$, and accept $(N, r)$ iff the alg says "yes, $L(M_3) = \emptyset$".

Grammars: Let's compare two algorithms.

1. $\text{NULL} = \emptyset$
   
   bool changed = false;

   while (changed) {
     changed = false;
     for (each rule $A \rightarrow X$) {
       if (A \notin \text{NULL}) {
         if ($X \in \text{NULL}^*$) {
           \text{NULL} = \text{NULL} \cup \exists A \forall \}
         } else { $\text{NULL}^*$ }
         changed = true;
       }
   }

   accept if $\exists$ \text{NULL} \land \text{true} \iff \{G \in \text{EPS} \mid \text{NE}(G = \{G\}) \land (G \not\in \text{EPS}) : S \Rightarrow \epsilon \}

2. $\text{LIVE} = \Sigma$
   
   bool changed = false;

   while (changed) {
     changed = false;
     for (each rule $A \rightarrow X$ with $A \notin \text{LIVE}$) {
       if ($X \in \text{LIVE}^*$) {
         \text{LIVE} = \text{LIVE} \cup \exists A \forall \}
       } else { $\text{LIVE}^*$ }
       changed = true;
     }

   accept if $\exists$ \text{LIVE} \land \text{true} \iff \{G \in \text{EPS} \mid \text{NE}(G = \{G\}) \land (G \not\in \text{EPS}) : S \Rightarrow \epsilon \}$.
Hence these problems are decidable:

\[ \text{NE}(G) = \{ G \mid G \text{ is a CFG and } L(G) \neq \emptyset \} \]

\[ \text{EPS}(G) = \{ G \mid G \text{ is a CFG and } S \Rightarrow \varepsilon \} \]

How about:

\[ \text{ALL}(G) = \{ G = (V, \Sigma, R, S) \mid G \text{ is a CFG and } L(G) = \Sigma^* \} \]

\[ \text{EQ}(G_1, G_2) = \{ \langle G_1, G_2 \rangle \mid L(G_1) = L(G_2) \} \]

\[ \text{EQ}(G, \text{regexp}) = \{ \langle G, r \rangle \mid L(G) = L(r) \} \]

\textbf{Fact:} These problems are \underline{undecidable}. There is no algorithm that halts and answers correctly in all cases. Take, for example, when \( G \) is "aback": TM computable, 6.5.2. skim.

\[ \text{Key Fact: If a language } L \text{ is not acceptable by a TM at all, i.e. not in \( \text{RE} \), then it is not in } \text{DEC}, \text{ hence it is undecidable (too).} \]
DEF of the "Diagonal Problem" for any class $C$ of machine:

$D_C$ = \{$M \mid R(M, \langle M \rangle) \neq L(M) \}$

PROOF: Just run $M$ on $\langle M \rangle \uparrow$. Then:

$D_C$ is decidable. 
$D_\text{DFA}$ is decidable. 

$D_\text{NFA}$ = $Z(N)$: the NFA $N$ does not accept $\langle N \rangle$. 

$D_CFG$ = $\{ \langle G \rangle \mid \text{the CFG $G$ does not generate $\langle G \rangle$, \text{or} \}$

Let $L(G) = \{ \langle G \rangle \}$. Then $\langle G \rangle \in L(G) \iff G'$ derives it in $2n-1$ steps. 

Thm: $D_\text{TM} = \{ \langle M \rangle \mid M \text{ is a dec TM and $M$ does not accept } \langle \langle M \rangle \rangle \}$ is not in RE.

PROOF: Suppose there were a TM $Q$ s.t. $L(Q) = D_\text{TM}$. Then $\langle Q \rangle$ would be in code over $\Sigma$. (say $\Sigma = \{0,1\}$). Then

$\langle Q \rangle \in D_\text{TM} \iff Q \text{ does not accept } \langle \langle Q \rangle \rangle \uparrow$ by def. of $D_\text{TM}$

$\iff Q \text{ does accept } \langle Q \rangle \uparrow$ by def. of $L(Q)$ in $D_\text{TM}$

A logical statement can never be $\iff$ to its own negation. Hence the supposition is wrong. $Q$ does not exist, so $D_\text{TM}$ is not Turing-reducible.