Last time I covered an algorithm for the problem

INSTANCE: A DFA M

QUESTION: Is $L(M) \neq \emptyset$?

"NE_{DFA}" names both the problem and the language

A fixed encoding whose details don't matter

$NE_{DFA} = \{\langle M \rangle : M \text{ is a DFA } \& \ L(M) \neq \emptyset\}$

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

Theorem: NE_{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 chars (and ϵs) along that path concat 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 NFA → DFA, this search is only in the NFA graph, so there is no possible "exponential expansion.") ☒.

ALL_{DFA}: INST: A DFA $M = (Q, \Sigma, \delta, s, F)$
QUES: Is $L(M) = \Sigma^*$?

Thm: ALL_{DFA} is decidable (and efficiently)

Alg^m: Given M, 1. Construct the complemented DFA $M' = (Q, \Sigma, \delta, s, Q \setminus F)$.

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

2. Run our alg^m for NE_{DFA} on $\langle M' \rangle$.
3. Answer <u>yes</u> about M iff the alg^m says <u>no</u> on M'.

☆ We have "reduced" ALL_{DFA} to the complemented problem E_{DFA}
$E_{DFA} = \{\langle M' \rangle : M'$ is a DFA and $L(M') = \emptyset\}$.

$ALL_{NFA} = \{\langle N \rangle : N$ is an NFA and $L(N) = \Sigma^*\}$

Thm: ALL_{NFA} is decidable (but not so efficiently as far as we know)

Proof: Convert N to DFA M s.t. $L(M) = L(N)$, run ALL_{DFA} on M.

Not always "efficient." Later we will see that ALL_{NFA} is an <u>NP-hard</u> problem

"$EQ_{NFA, regexp}$": INST: An NFA N and a regexp r
QUES: Is $L(N) = L(r)$?

Note that ALL_{NFA} is the special case where we're given $\boxed{r = (a+b)^*}$
So if this problem had an <u>efficient</u> decider, so would ALL_{NFA}.

3

Algorithm: 1. Convert r into an NFA N_r s.t. $L(N_r) = L(r)$.

⚠ nonefficient!

2. Convert both N and N_r into equivalent DFAs M_1 and M_2.

Then $L(N) = L(r) \iff L(N) = L(N_r)$

$\iff L(M_1) = L(M_2)$ ← EQ_{DFA}: I: DFAs M_1, M_2; Q: Is $L(M_1) = L(M_2)$?

We have reduced the goal to an instance of

$\iff L(M_1) \Delta L(M_2) = \emptyset$.

$\iff L(M_3) = \emptyset$ where M_3 is obtained via "Cart. Prod for Δ".

$\iff \langle M_3 \rangle$ is a yes-instance of the E_{DFA} problem.

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

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 $\approx$ (size of M_1) $\cdot$ (size of M_2).

4. Run the E_{DFA} algm on M_3, and accept $\langle N, r \rangle$ iff the algm says "yes, $L(M_3) = \emptyset$".

Grammars: Let's compare two algorithms.

1. $NULL = \emptyset$
```
bool changed = true;
  while (changed) {
    changed = false;

    for (each rule A -> X
         where A ∉ NULL) {
      if (X ∈ NULL*) {
        NULL = NULL ∪ {A}
        changed = true;
} } }
accept iff S ∈ NULL
```
← true iff $\langle G \rangle \in EPS_{CFG} = \{\langle G \rangle : S \Rightarrow^* \varepsilon\}$.

2. $LIVE = \Sigma$
```
bool changed = true;
while (changed) {
  changed = false;
  for (each rule A -> X with A ∉ LIVE) {
    if (X ∈ LIVE*) {
      LIVE = LIVE ∪ {A}
      changed = true;
} } }
accept iff S ∈ LIVE.
```
← Pretty efficient since outer loop runs at most $|V|$ times

new variable

$\equiv \langle G \rangle$ is in $NE_{CFG} = \{\langle G \rangle : L(G) \neq \emptyset\}$

Hence these problems are decidable:

$NE_{CFG} = \{\langle G\rangle : G \text{ is a CFG and } L(G) \neq \emptyset\}$.

$EPS_{CFG} = \{\langle G\rangle : G \text{ is a CFG and } S \overset{*}{\Rightarrow} \varepsilon\}$

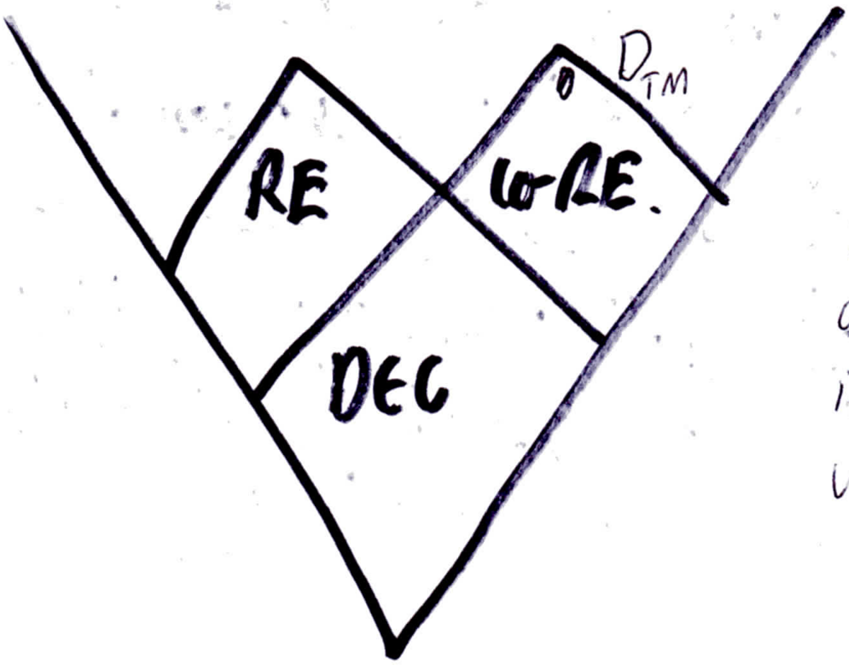
How about

$ALL_{CFG} = \{\langle G\rangle : G=(V,\Sigma,R,S) \text{ is a CFG and } L(G)=\Sigma^*\}$?

$EQ_{CFG} = \{\langle G_1,G_2\rangle : L(G_1)=L(G_2)\}$.

$EQ_{CFG,Regexp} = \{\langle G_1, r\rangle : L(G_1)=L(r) \ \ r \text{ is a regexp}\}$.

<u>Fact</u>: These problems are <u>undecidable</u>. There is "no algorithm that halts and answers correctly in all cases.
"Takes root" when G is "about" TM computations, §5.2 <u>skim</u>.



<u>Key Fact</u>: If a language L is not acceptable by a TM at all, i.e. not in RE, then it is not in DEC, hence is undecidable (too).

Defn of the "Diagonal Problem" for any class of machine:

D_{DFA}: INST: A DFA $M = (Q, \Sigma, \delta, s, F)$, coded as a string $\langle M \rangle$ over Σ.
QUES: Does M not accept $\langle M \rangle$?

Language: $D_{DFA} = \{ \langle M \rangle : M$ is a DFA and $\langle M \rangle \notin L(M) \}$.

Theorem: D_{DFA} is decidable. Proof: Just run M on $\langle M \rangle$. ⊠

$D_{NFA} = \{ \langle N \rangle :$ the NFA N does not accept $\langle N \rangle \}$. Decide by first converting N into a DFA M, or track $N(\langle N \rangle)$ directly.

$D_{CFG} = \{ \langle G \rangle :$ the CFG G does not generate $\langle G \rangle \}$.
Decide by first converting G into a ChNF grammar G' s.t. $L(G') = L(G)$.
Let $n = |\langle G \rangle|$. Then $\langle G \rangle \in L(G) \Leftrightarrow G'$ derives it in $2n-1$ steps.
We can decide this by trying all derivations for $2n-1$ steps.

Thm: The language $D_{TM} = \{ \langle M \rangle : M$ is a det^c TM and M does not accept $\langle M \rangle \}$ is not in RE.

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

$\langle Q \rangle \in D_{TM} \Leftrightarrow Q$ does not accept $\langle Q \rangle$ by defn of D_{TM}

$\Leftrightarrow Q$ does accept $\langle Q \rangle$ by defn of $L(Q) = D_{TM}$

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