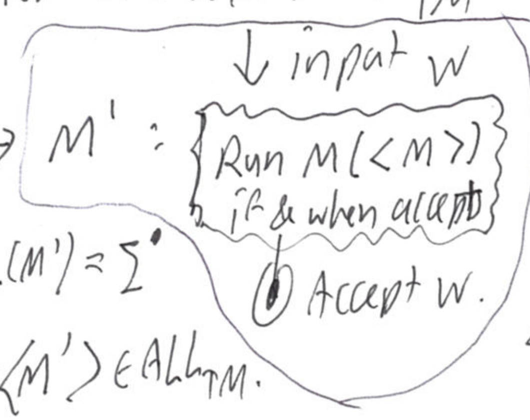
[Showed "Collatz 3n+1 Conjecture" Turing Machine.]
We believe the language is $(0 \cup 1)^*$ but don't even know whether it's decidable! We can add ϵ to make the language equal $(0 \cup 1)^*$. Then if we could decide ALL_{TM} for this 8-state machine, we could solve the conjecture.

But Theorem: ALL_{TM} is not even recognizable

Note: Earlier we showed $A_{TM} \leq_m ALL_{TM}$. Here is $K_{TM} \leq_m ALL_{TM}$

$M \xrightarrow{f} M'$:

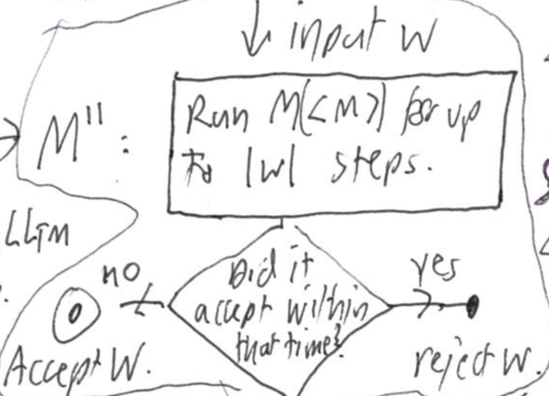


So $\langle M\rangle \in K \iff L(M') = \Sigma^*$
$\iff f(\langle M\rangle) = \langle M'\rangle \in ALL_{TM}$.

"All or Nothing Switch":
$\langle M\rangle \in K \implies M$ accepts $\langle M\rangle$
$\implies \forall w\ M'$ accepts $w \implies L(M') = \Sigma^*$
$\langle M\rangle \notin K \implies M$ does not acc $\langle M\rangle$
$\implies (\forall w)\ M'$ does not acc $w \implies L(M') = \emptyset$

Now show $D_{TM} \leq_m ALL_{TM}$ via the "Delay Switch". To reduce we need:

$M \xrightarrow{f'} M''$:



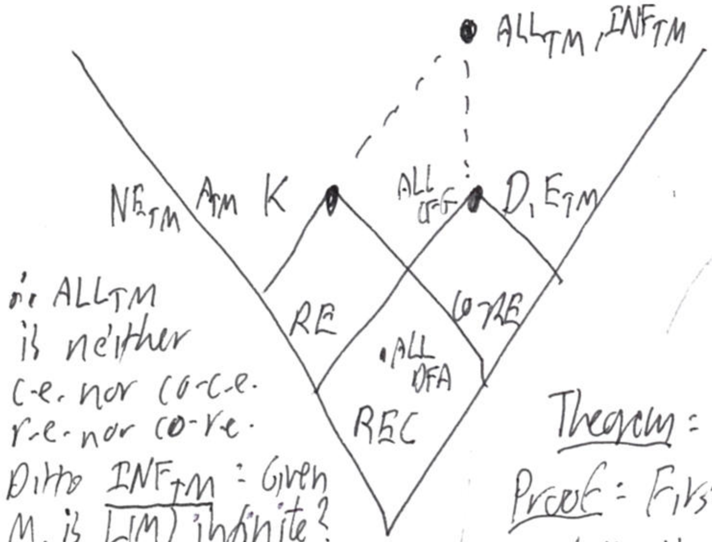
$\langle M\rangle \in D_{TM} \equiv M$ does not accept $\langle M\rangle$
$\iff \langle M''\rangle \in ALL_{TM} \equiv L(M'') = \Sigma^*$
So:
$\langle M\rangle \in D_{TM} \implies (\forall w)\ M$ does not accept $\langle M\rangle$ in $|w|$ steps (or at all)
$\implies (\forall w)\ M''$ accepts $w \implies \langle M''\rangle \in ALL_{TM}$
$\langle M\rangle \notin D_{TM} \implies M$ does accept $\langle M\rangle \implies$ there is a $t \geq 1$ s.t. M accepts $\langle M\rangle$ in t steps $\implies$ for all w s.t. $|w| \geq t$, $M''(w)$ rejects $\implies L(M'') \neq \Sigma^*$.

Note:
$\langle M\rangle \in D_{TM} \implies \langle M''\rangle \in ALL_{TM}$
$\implies L(M'')$ is infinite.
$\langle M\rangle \notin D_{TM} \implies (\exists t > 0)$
$L(M'') \subseteq \Sigma^{\leq t}$. (all $x \in L(M'')$ have $|x| \leq t$)
$\implies L(M'')$ is finite! So $D_{TM} \leq_m INF_{TM}$ too.

Defn: A language B is *hard* for a class $\mathcal{C}$ of languages *under* $\leq_m$ if:

$\boxed{\text{For all } A \in \mathcal{C},\ A \leq_m B.}$

B is *complete* for $\mathcal{C}$ (under $\leq_m$) if *also* $B \in \mathcal{C}$.



ie ALL_{TM} is neither c.e. nor co-c.e. r.e. nor co-r.e.
Ditto $\underline{INF_{TM}}$: Given M, is $\underline{L(M)}$ infinite?

Theorem: A_{TM} is complete for RE under $\leq_m$.

Proof: First, $A_{TM} \in$ RE. Let any $A \in$ RE be given. By def^n there is a det^c TM M such that $A = L(M)$. M is *fixed*, so the function $f(x) = \langle M, x \rangle$ is computable. And $x \in A \iff M$ accepts $x \iff \langle M, x\rangle \in A_{TM} \iff f(x) \in A_{TM}$. So $A \leq_m A_{TM}$, and since $A \in$ RE is arbitrary, A_{TM} is complete.

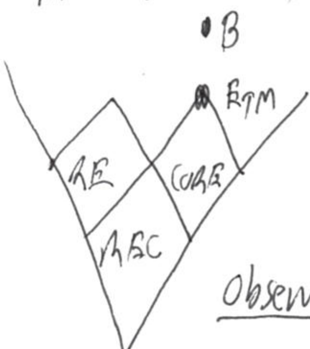
Also $A_{TM} \leq_m K$ since taking $g \equiv$ All or Nothing Switch from A_{TM} last Thu

$\langle M, x \rangle \xrightarrow{g} M'$ [↓ input w; Sim $M(x)$; if acc → Accept w.]

$\langle M, x\rangle \in A_{TM} \Rightarrow L(M') = \Sigma^* \Rightarrow M'$ *does* accept its own code $\langle M' \rangle$ (*whatever* that is) $\Rightarrow g(M, x) \in K$.

$\langle M, x\rangle \notin A_{TM} \Rightarrow L(M') = \emptyset \Rightarrow M'$ does *not* accept its own code or anything $\Rightarrow \langle M' \rangle \notin K$ ie $g(\langle M, x\rangle) \notin K$.

So A_{TM} reduces back to K_{TM}, and since $K_{TM} \in$ RE and $\leq_m$ is *transitive*, K too is *RE-complete*. Ditto NE_{TM}...



Since E_{TM} is not even recognizable, then if $E_{TM} \leq_m B$, the B is not recognizable either, let alone decidable.

Observe: $\langle M \rangle \in E_{TM} \iff L(M) = \emptyset \iff M$ has *no* accepting computations $\iff$ the language $\underline{V_M} =_{def} \{\langle I_0, \ldots, I_t \rangle :$ this is a valid accepting comp^n $\} = \emptyset$.

Now for any fixed M, V_M *is* decidable — in fact, decidable in $\overline{\text{linear time}}^{\,O(N)}$

The length $N = |\langle I_0(x), I_1, \ldots, I_t \rangle| \leq (1+t) \cdot$ max length of any ID I_j $\leq$ some constant $\cdot \max(n, t)$.

because $|I_0(x)| = n+1$ In t steps, ID can grow by at most t chars.

Added Note: For Thu's last lecture, this shows that the IDs can fit into a $(t+1) \times (t+1)$ grid, i.e. *tableau*

Moreover, the decider just needs to check $(\forall j : 1 \leq j \leq t)\ I_{j-1} \vdash_M I_j$ and $I_0 = sx$ for some x and $I_j = q_{acc}$ ⊥ (by "good housekeeping" for M).

It can do this without using any tape besides the input chain of IDs.

Hence the decider can be a Linear Bounded Automaton (LBA).

$\therefore\ E_{TM} \leq_m E_{LBA}$ via the conversion from M to an LBA for V_M.

The simplest LBA I know emulates having 2 tape heads that check $I_0 \overset{?}{\vdash_M} I_1$, $I_1 \overset{?}{\vdash_M} I_2$, $I_2 \overset{?}{\vdash_M} I_3$, $I_3 \overset{?}{\vdash} I_4 \cdots I_{t-1} \overset{?}{\vdash_M} I_t$ in tag-team fashion. Hence emptiness E_{Tmac} for this ~~type~~ "Tmac" of machines is undecidable. What other machines or tag-teams or complements can check computations?

One ~~DPDA~~ cannot: Text shows how we can help it by writing every odd ID in reverse: $I_0 \# I_1^R \# I_2 \# I_3^R \# I_4 \cdots I_t^{(R?)}$

And NPDAs can do no better, since $E_{NPDA} \equiv E_{CFG}$ is decidable!

A DPDA can push on I_0, pop-compare on I_1, but then has nothing left on the stack by which to test $I_1 \overset{?}{\vdash_M} I_2$.

However, V_M does equal $L(P_1) \cap L(P_2)$ for two DPDAs that "tag-team" checking the odd and even pairs of IDs.

Moreover, an NPDA N can recognize when $I_0 \cdots I_t$ is invalid by guessing which pair fails.

Thus these two problems are both undecidable: E_{TM} reduces to both of them.

① INST: CFGs G_1 and G_2
QUES: Is $L(G_1) \cap L(G_2) = \emptyset$?

② ALL_{CFG}: $L(G) = \Sigma^* \iff L(N) = \Sigma^* \iff L(M) = \emptyset$.

Nevertheless, ALL_{DFA} is decidable, indeed in time

$O(|M|^2) = O(|Q|^2)$ ignoring log factors, by Breadth-First Search

FYI $\boxed{\tilde{O}(t(n)) \equiv O(t(n)(\log t(n))^{O(1)})}$

$\boxed{\text{Added: The log factors come from labels on states by binary integers. In CSE331 sometimes you ignore such labels on graph nodes, treating each node visit as unit time. Or write } \tilde{O}(|Q|^2) \text{ with a tilde}}$

Defn: A language L belongs to the class P of problems decidable in deterministic polynomial time if there is a polynomial $p(n)$ and a det^c TM M s.t. for all $x \in \Sigma^*$,

M has an accepting comp^n of length $\leq p(|x|)$ and $L = L(M)$.

Defn: $L \in NP$ if we allow a nondet^c TM N in place of M.

$x \in L \iff N$ has some ~~an~~ acc comp^n of length $\leq p(|x|)$.

P-computable fns f are defined similarly, and $A \leq_m^p B$ iff there is an f computable in poly time s.t. $\forall x: x \in A \iff f(x) \in B$

Defn: B is NP hard $\equiv$ for all $A \in NP$, $A \leq_m^p B$.

and B is NP-complete $\equiv$ $B \in NP$ and B is NP-hard.

ALL_{NFA} is NP-hard. (Added). The reason converting the given NFA N into an equivalent DFA M is no good for P is that it runs in $\sim 2^{|Q|}$ time in worst case. The easier complementary problem, "Is there some string x of length $\leq |Q|$ that N fails to accept?" does belong to NP (your HW) and is in fact NP-complete, as Thursday will finish showing.

Footnotes: The "Sipser Naming Scheme" extends to other Kinds of Question subscripted by other Types of Machine or Formal Object ("Tmac") Let "T" stand for a 'Tmac', could be TM or FA or regexp or CFG etc. or a "tandem" or "complement" of a Tmac(s), etc.

A_{Tmac}: Given T and an $x \in \Sigma^*$, is $x \in L(T)$?
E_{Tmac}: Given a T, is $L(T) = \emptyset$?
ALL_{Tmac}: Given a T, is $L(T) = \Sigma^*$?
EQ_{Tmac}: Given T_1 and T_2, is $L(T_1) = L(T_2)$?
$HALT_{Tmac}$: Given T and x, does T run on x halt?
PT_{Tmac} aka TOT_{Tmac}: Given a T, is T total, ie $\forall x\, T(x)\downarrow$?
$DISJ_{Tmac}$: Given T_1 and T_2, is $L(T_1) \cap L(T_2) = \emptyset$?

just for ALL_{CFG} in §5.1

ALL_{Tmac} is not named generally in te[xt]. It is the restriction of $EQ_{Tmac}(T_1, T_2)$ where T_2 is a fixed argument such that $L(T_2) = \Sigma^*$

PT for "Program Termination" or TOT for "total". Much as $HALT_{TM} \equiv_m A_{TM}$, also $TOT_{TM} \equiv_m ALL_{TM}$

Q \ Tmac	DFA	NFA Regexp ①	DPDA	NPDA CFG	DLBA NLBA	Ptime TMs	DTM NTM	Boolean Formulas ϕ	
A	Dec, in P	Dec, in P	Dec, in P	Dec, in P	Dec, NPH	Dec, $\notin$P ⑦	RE-complete	Is $\phi(x)$ = True?	Dec - in P
E	Dec, in P	Dec, in P	Dec, in P	Dec, in P	CREC	CREC	CREC	Is $\forall x\ \phi(x)$ = false?	co-NPC ⑤
ALL	Dec, in P	Dec, NP-hard	Dec, in P ③	⑥ CREC	CREC	CREC	Neither re nor co-re. ④	ϕ = tautology?	co-NPC
EQ	Dec, in P	Dec, NP-hard	Dec, NPH	CREC	CREC	CREC	Neither-Nor	Is $\phi_1 \leftrightarrow \phi_2$?	co-NPC
HALT	Always	Always	Dec, in P	Dec/ in P	Dec, NPH	Always	RE-complete	n-a.	ALL_{TM}, EQ_{TM}, PT_{TM}, INF_{TM}
PT	Always	Always	Dec, in P	Dec/ in P	Dec, NPH	Always	Neither-Nor	n-a.	
DISJ	Dec, in P	Dec, in P	UNDEC CREC ②	CREC	CREC	CREC	CREC	$\equiv E(\phi_1 \wedge \phi_2)$	co-NPC

Know all

① Formally, regular expressions disallow integer powers like $(0 \cup 1)^{37}$. If they are allowed, the complexity gets worse.

② CREC is short for "Co-RE complete", ie $\equiv_m D_{TM}$.

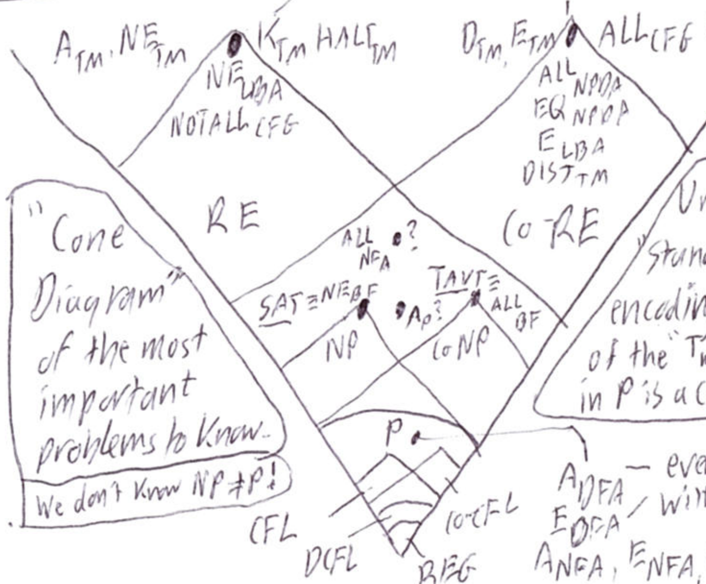
③ NPH is short for NP-hard

④ $ALL_{TM} \equiv_m EQ_{TM} \equiv_m PT_{TM}$ — all complete for a class called Π_2

⑤ co-NPC means complete for co-NP — ie, the complements NE_{BF} etc. are all NP-complete. $NE_{BF} \equiv \{\phi : (\exists x)\ \phi(x) = true\} \equiv SAT$ which is NPC.

⑥ EQ_{DPDA} was only proved decidable a decade ago. It is a crazy hard problem — just FYI — "off the map" but in REC.

⑦ The diagonal language $D_P = \{$P-machines that don't accept their own code$\}$ is decidable but not in P. Try proving this yourself! Hence A_P ditto.



"Under standard encodings", none of the "Tmac problems" in P is a CFL or co-CFL

A_{DFA}, E_{DFA} — everything with DFAs. A_{NFA}, E_{NFA}, $DIST_{NFA}$, A_{CFG}, E_{CFG}. Halting and totality for DPDAs is part of a "good housekeeping" conversion

Sipser's text only proves that every CFL is in P. $A_{CFG} \in P$ follows a different way.