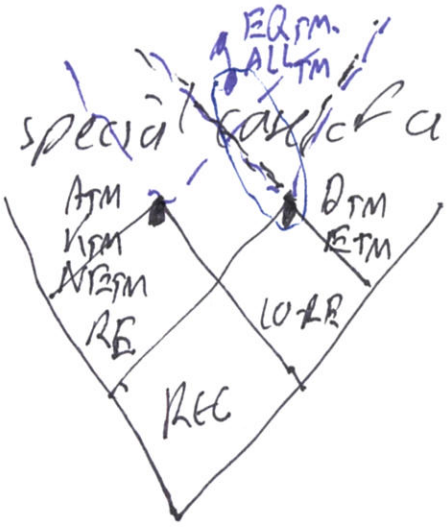
Lecture Thu May 7 (the last) Reductions & Complexity

1. A reduction can be a mapping into a special case of a more general problem.



$E_{TM} = \{\langle M\rangle : L(M) = \emptyset\}$?

$EQ_{TM} = \{\langle M_1, M_2\rangle : L(M_1) = L(M_2)\}$.

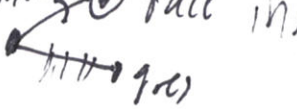
Then $E_{TM} \leq_m EQ_{TM}$ via the mapping $f(\langle M\rangle) = \langle M, M_0\rangle$,

where M_0 = 

$L(M_0) = \emptyset$, so $\langle M, M_0\rangle \in EQ_{TM} \Leftrightarrow \langle M\rangle \in E_{TM}$.

The mapping f is computable since it simply "bolts on" the M_0 code.

Thus $E_{TM} \leq_m EQ_{TM}$, and since E_{TM} is not c.e., neither is EQ_{TM}

In fact $ALL_{TM} \leq_m EQ_{TM}$, using M_1 = 

instead of M_0. "Reduction By Restriction"

Since ALL_{TM} is neither c.e. nor co-c.e., so is EQ_{TM}.

2. Reductions that don't simply "bolt-on" or "drop-in" blocks of code.

5.1

Theorem: There are computable functions f_1 and f_2 that given any one-tape TM M produce context-free grammars G_1, G_2, and G_3 $\{f_1(M) = G_1, f_2(M) = \langle G_2, G_3\rangle\}$ such that:

$$L(M) = \emptyset \overset{①}{\Leftrightarrow} L(G_1) = \Sigma^* \overset{②}{\Leftrightarrow} L(G_2) \cap L(G_3) = \emptyset.$$

① shows $E_{TM} \leq_m ALL_{CFG}$, and ② reduces E_{TM} to "Does $L(G_1) \cap L(G_2) = \emptyset$?"

Proof (sketch): We can write configurations (aka IDs) of a 1-tape TM M in a format specified by the following regular expression.

$M = (Q, \Sigma, \Gamma, \delta, \ldots)$
$\Gamma' = \Gamma \setminus \Sigma \cup \{\sqcup\}$.
assume $[,] \notin \Gamma$
define states $q \in Q$
as alphabet chars
all alphabet: $\Gamma \cup Q \cup \{[,]\}$.

$$r = [\,\cdot\,(\Gamma')^*\cdot Q\cdot\Gamma\cdot(\Gamma')^*\,]$$

e.g. $[s\,w]$ is the start ID of M on input w. (if $w \neq \varepsilon$).

$[\,u\ q\ c\ v\,]$ = "Machine M is currently in state q scanning char c with u to its left, v to its right, and all the rest of the tape is blank."

also: $c = \sqcup$ only if $v = \varepsilon$.

Sequences of IDs have the regular format r^+ (one or more IDs).

The machine M defines a condition for an ID J to follow an ID I by one step of M as coded by an instruction in δ.

$$[\,u\ q\ c\ v\,][\,u'\ r\ c'\ v'\,]$$

If the move is $(q, c/d, S, r)$ then $u' = u$ and $v' = v$, with $c' = d$.
If the move is $(q, c/d, R, r)$ then $u' = ud$, $c' =$ first char of v, $v' =$ the rest of v.
If the move is $(q, c/d, L, r)$, similar. (or etc. if $v = \varepsilon$, $c' = \sqcup$).

= "][" ↑

This condition is like DOUBLEWORD (with # marker)
If we write J backwards, it becomes like PALINDROME, with #.

Define $VH_M = \{I_0 I_1 I_2 \cdots I_t \in L(r^+)$: I_0 = the start ID on some input w, I_t is an accepting ID, and for all j, $1 \le j \le t$, I_j can follow I_{j-1} by $M\}$

Valid Histories

$\widetilde{L(r^+)} \cup$

$\widetilde{VH_M}$ is like $\widetilde{\Delta^* \cdot (\text{DOUBLEWORD}) \cdot \Delta^*}$, and since CFLs are closed under $\cdot$, it is a CFL. $f_1(M) =$ a CFG G_1 for $\widetilde{VH_M}$.

Also define $VHR_M = \{I_0 \# I_1^R \# I_2 \# I_3^R \cdots I_t : I_0 = [s\,w]$ for some w, I_t is accepting, and reverse if t is odd

I_j follows I_{j-1} $\forall j \le t\}$

Handles I_1 follows I_0, I_3 follows I_2, I_5 from I_4 etc.

$\rightsquigarrow$ VHR is like $(MARKEDPAL)^+ \cap (L(r)\#(MARKEDPAL)^*)$.

Handles the odd cases: I_1 followed by I_2, I_3 followed by I_4 etc.

$\therefore$ We can build CFGs G_2 and G_3 s.t. $VHR_M = L(G_2) \cap L(G_3)$.

$\therefore$ $M \in E_{TM} \Leftrightarrow VHR_M = \emptyset \Leftrightarrow L(G_2) \cap L(G_3) = \emptyset \Leftrightarrow f_2(\langle M \rangle)$ call it $\in$ is a yes-instance of the "Does $L(G') \cap L(G) = \emptyset$?" $E_{\cap CFG}$:

Last reduction in §5.1: VH_M and VHR_M can both be decided by special TMs B that never go outside the cells initially occupied by their input $\vec{I}$.

DLBAs are closed under complement.

B is called a def. linear bounded automaton (LBA). Hence both E_{LBA} and ALL_{LBA} are undecidable problems as well.

This happens for ANY kind of deterministic machine that can verify computations & proofs!

3. A little more on Logic and "P" and "NP":

SATISFIABILITY: (SAT)

Instance: A Boolean formula $f(x_1 \cdots x_n)$ like $(x_1 \vee x_2) \wedge \neg(x_3 \wedge x_4) \vee (x_3 \wedge \overline{x_1})$

Is f satisfiable? $\equiv$ $\Leftrightarrow$ Is $\neg f$ not a tautology?

Question: Is there an assignment $a_1, a_2, \ldots, a_n \in \{0,1\}$ that makes $f(a_1, a_2, \ldots, a_n) = true$?

Defn 1: A language B belongs to $\underline{P}$ if there is a deterministic TM M such that $L(M) = B$, and for all inputs $x \in \Sigma^*$, $M(x)$ halts within $q(|x|)$ steps, where q is a fixed polynomial (like $n^K + K$).

M or N can have any number of tapes.

i.e. "t" from VIH_M is $\leq q(|x|)$.

Defn 2: B belongs to $\underline{NP}$, if there is a $\underline{\text{nondeterministic}}$ TM N s.t. $L(N) = B$ and every $\underline{\text{branch}}$ of N's computation halts within $q(n)$ steps, where $n = |x|$ and q is some fixed polynomial.

Examples: All of our multi-tape linear-time languages belong to P.

- All of our decision problems ultimately solved by BFS on E_{DFA} belong to P.
- IH_M
- IHR_M • With the EPS_{CFG} and E_{CFG} algorithms
- IC • Any CFL belongs to P (not trivial! indeed $A_{CFG} \in P$ (really hard!! (not in text!!)
- IO
- $\wedge P$. • $\{\langle f(x_1 \cdots x_n), a_1 a_2 \cdots a_n \rangle : f(a_1 \cdots a_n) = \text{true}\} \in P$

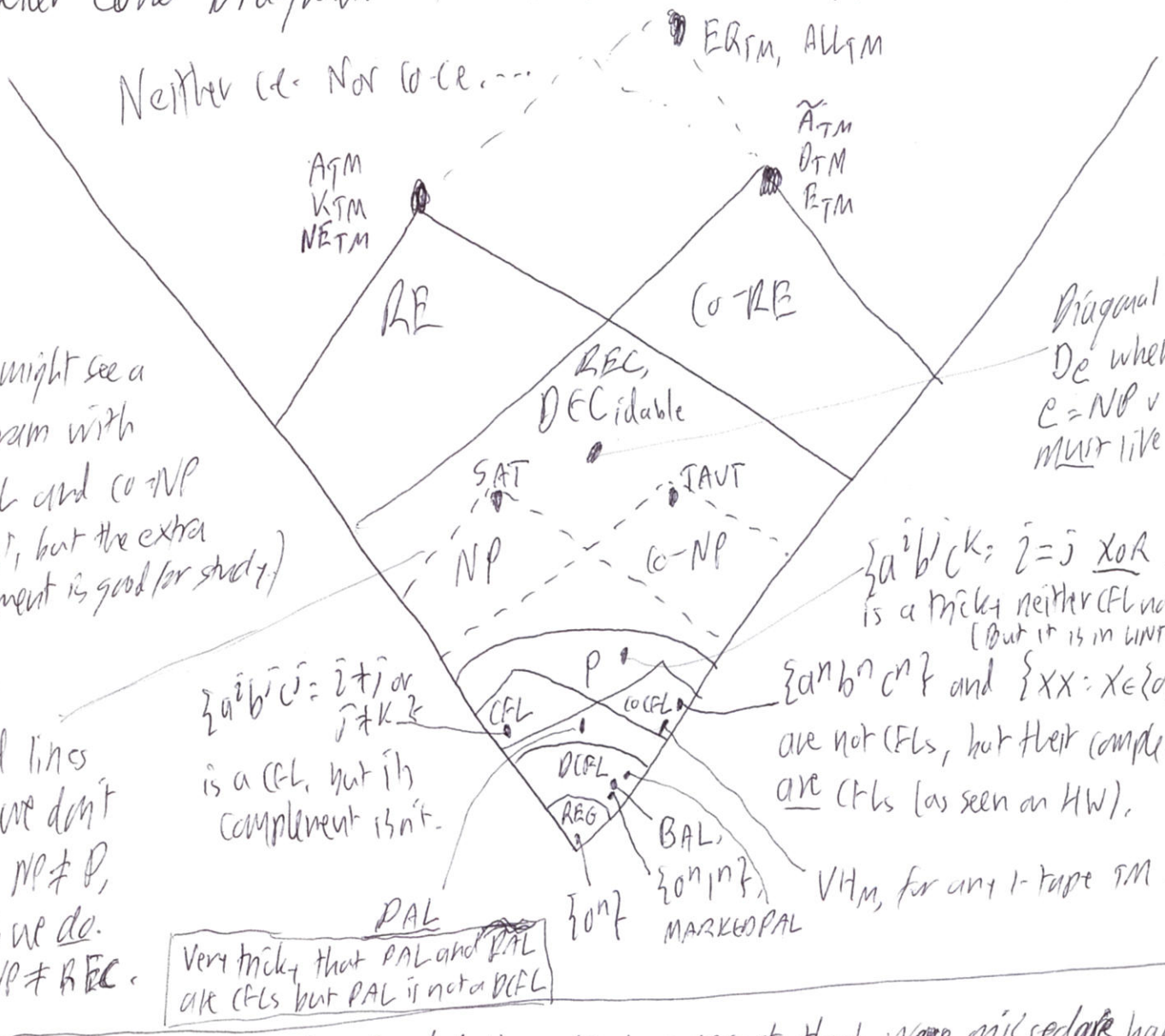
this is value-checking for logical formulas

HOWEVER, $\underline{SAT} = \{\langle f(x_1 \cdots x_n) \rangle : (\exists\, a_1 \cdots a_n) : f(a_1 \cdots a_n) = \text{true}\}$

is only known to belong to $\underline{NP}$, via an NTM that on input f $\underline{\text{guesses}}$ $a_1 \cdots a_n$ and then $\underline{\text{verifies}}$ $f(a_1 \cdots a_n) = \text{true}$.

Theorem: $P = NP \iff SAT \in P$ — but, this is unknown. $NP \stackrel{?}{=} P$

A better "Cone Diagram" than I drew at the end of lecture:



EXTRA: The main & last intended theoretical concepts that were missed are hardness and completeness under polynomial time computable mapping reductions. We write $A \leq_m^p B$ if $A \leq_m B$ via a function $f: \Sigma^* \to \Sigma^*$ such that $y = f(x)$ is computable in time $|x|^{O(1)}$, and of course $x \in A \Leftrightarrow y \in B$ since f is a mapping reduction. Because the composition of two polynomials is a polynomial — like I said $(n^3)^2 = n^6$ in lecture — we get the same kinds of theorems as in §5.3, but with P in place of DEC and NP in place of RE:

If $A \leq_m^p B$ then:
1. $B \in P \Rightarrow A \in P$
2. $B \in NP \Rightarrow A \in NP$
3. $B \in coNP \Rightarrow A \in coNP$

So: $A \notin P \Rightarrow B \notin P$.
So: $A \notin NP \Rightarrow B \notin NP$.
So $\overline{A} \notin NP \Rightarrow \overline{B} \notin NP$
Really the same since $A \leq_m^p B \Leftrightarrow \overline{A} \leq_m^p \overline{B}$.

Finally B is NP-hard if $A \leq_m^p B$ for all $A \in NP$, and NP-complete if also $B \in NP$. So the Cook-Levin Theorem states that SAT is NP-complete. Also A_{TM} is complete for RE under $\leq_m$.