CSC 2/486

Lane A. Hemaspaandra

Do lion packs (prides) and tennis tournaments have a role in our understanding of the nature of semi-feasible algorithms?

What is computable?

What is feasibly computable?

Do one-way functions exist?

Are all undecidable sets Turing-equivalent in complexity?

What notions do we have of low information content sets, and can any such set be NP-complete? NP-hard?

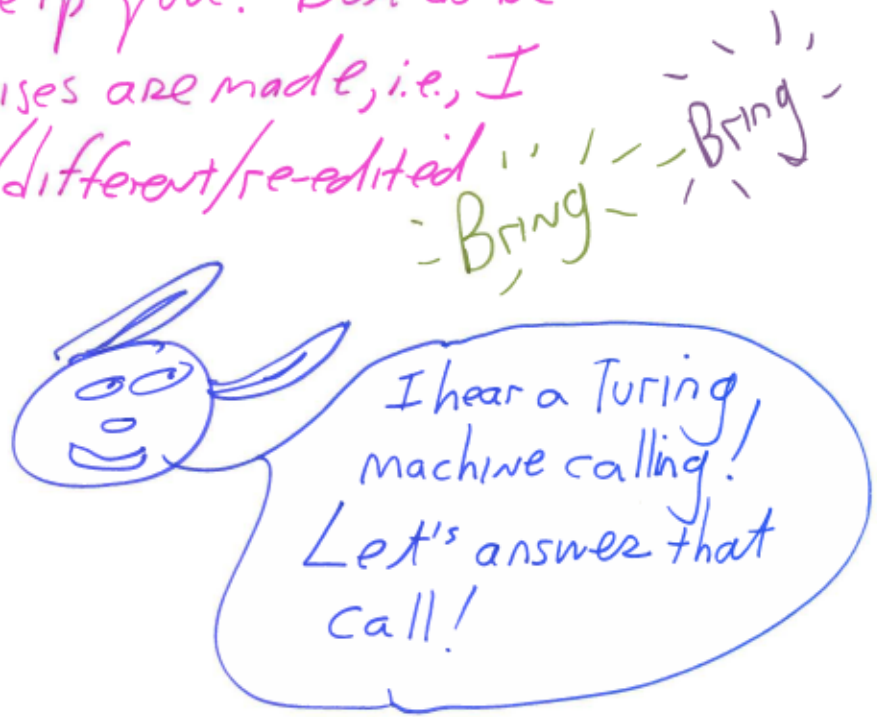
Are computability and complexity theory related? How closely?

These slides are not for redistribution

© Lane A Hemaspaandra.

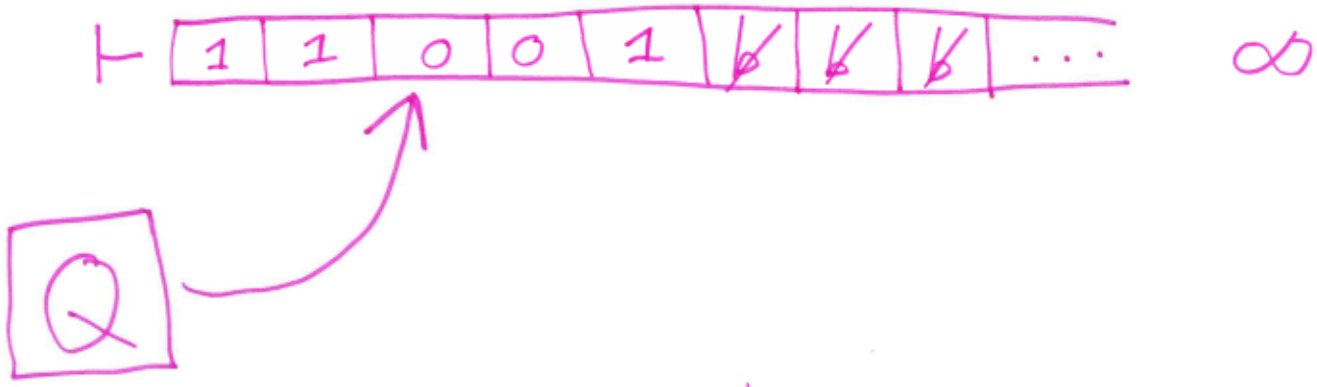
Regarding the pdf I'll put up of this set of slides at the start of the term:

I'm putting it up to help you. But do be aware that no promises are made, i.e., I may use more/less/different/re-edited slides in the lectures.



The Turing Machine

Nov. 12, 1936 Turing presents a paper to the London Math. Society



- can move left or right
- can read and write

- tape is ∞ length to right (semi-infinite tape)

$\{a^N b^N c^N \mid N \geq 1\}$ is not a CFL, but is a piece of cake on a Turing machine.

A TM (det. or nondet.) is $M = (Q, \Sigma, \Gamma, \vdash, \not b, s, F, \delta)$

Q, Σ, s, F are as usual.

- Q: states (finite set)
- Σ: input alphabet (finite set)
- s: start state
- F: set of final states

Γ finite tape-alphabet $\Sigma \subseteq \Gamma$

$\vdash \in \Gamma - \Sigma$ left end marker

$\not b \in \Gamma - \Sigma$ blank symbol

What does δ look like?

TM in one step:

Read tape symbol and then based on that symbol and the current state ($\in Q$) do:

(1) print a symbol on the tape.

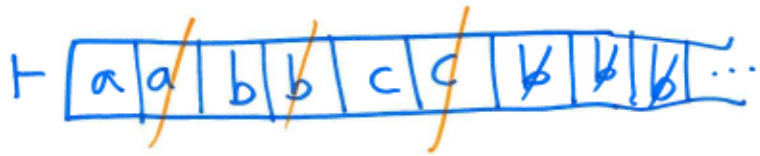
(2) move right or left. $\Sigma_{\vdash}^{?}$

(3) enter a state (perhaps the same).

Example:

Example

$\{a^N b^N c^N \mid N \geq 1\}$ $\quad \Sigma = \{a, b, c\}$

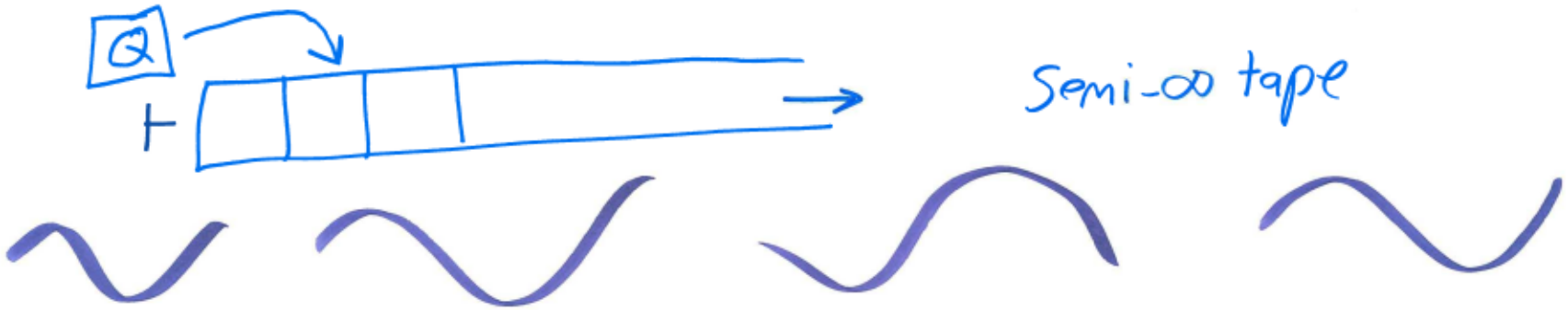


Loosely:

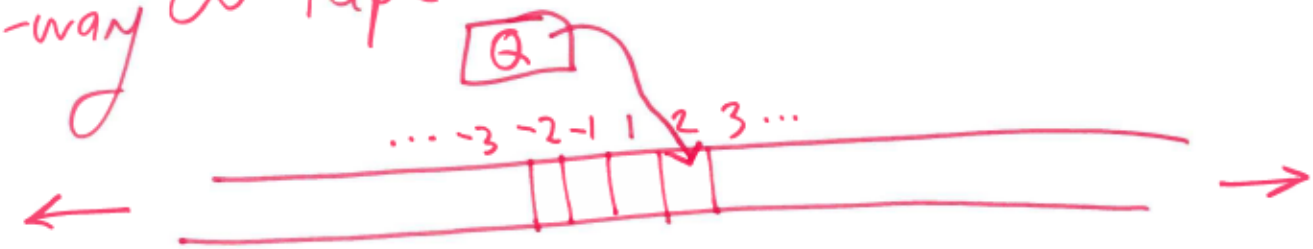
- w = scan right until first b (then write/change to "b" and t and move left)
- t = scan left until you see a "c" (then write/change to "c" and u and move left)
- u = scan left until you see a "b" (" " " " " v " " ")
- v = scan left until you see an "a" (" " " " " w " " " right)

halt (ϕ reject) if we see wrong symbols during a left scan.

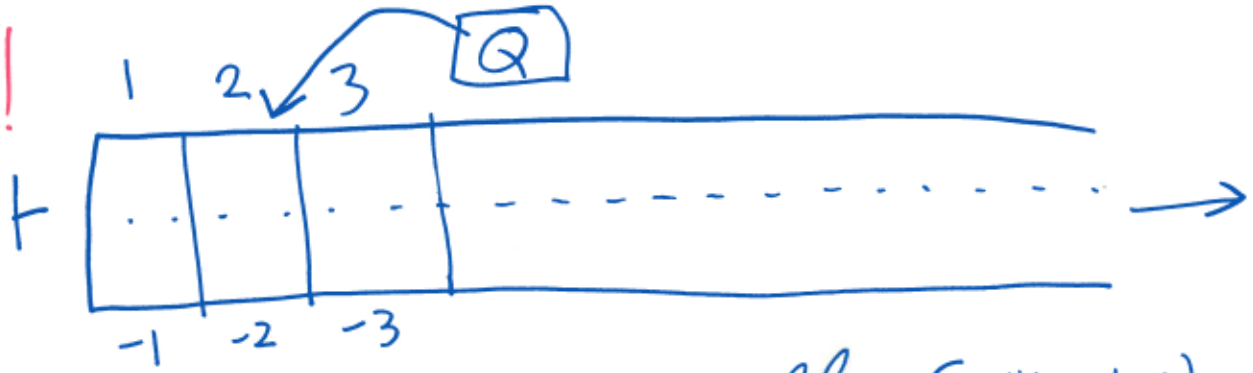
Accept if we hit ⊢ during state t.



Two-way ∞ tape

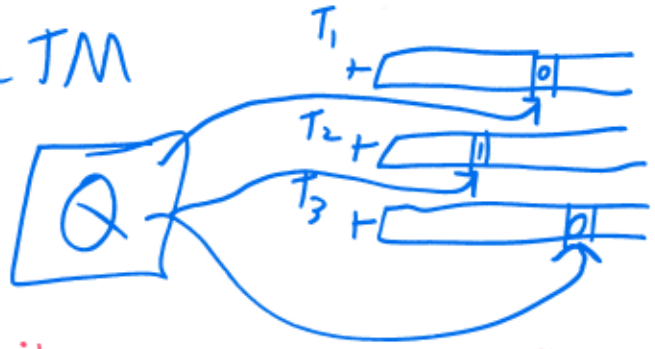


"fold"!!

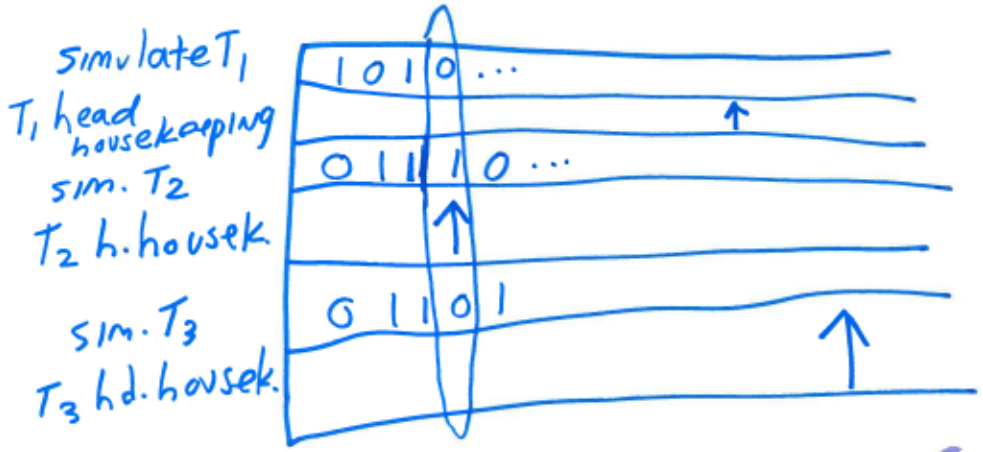


- "double-up" the info in each cell. (2 "tracks")
- appropriately modify the finite control to do the "right" thing. E.g., our state will hold whether we are "on" the pos. or the neg. part, and moving left into ⊢ is our "around the fold" excitement.

K-tape TM



simulate it using a 2K-track TM:



Nerdy note:

Chosen to avoid any startup grief (assuming the input comes in on the lower track)

Note: And 1-track can easily simulate ℓ tracks.

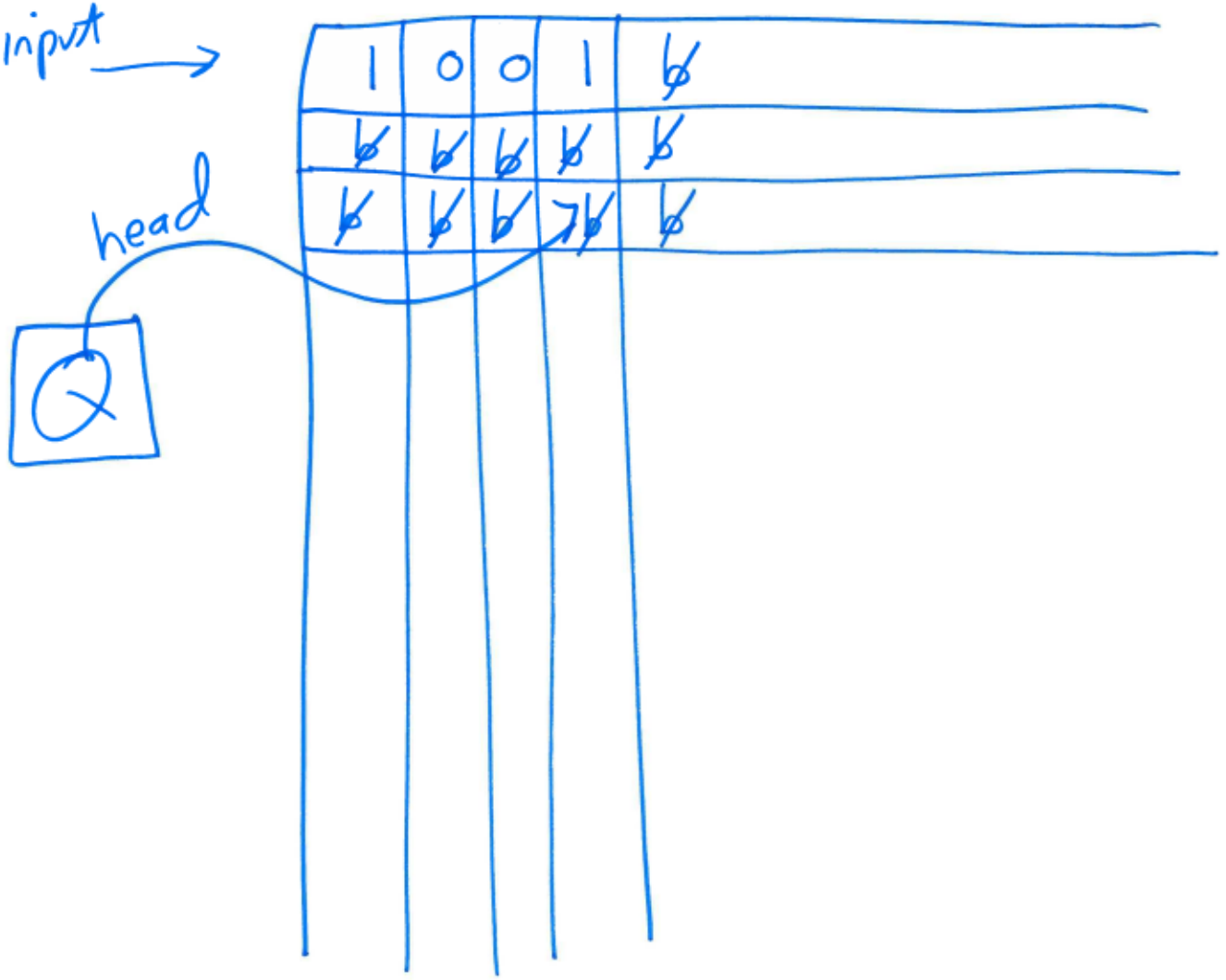
$\left\{ \begin{array}{l} \begin{smallmatrix}\not b\\ \not b\end{smallmatrix} \to \not b \quad \begin{smallmatrix}\not b\\ 0\end{smallmatrix} \to 0 \quad \begin{smallmatrix}\not b\\ 1\end{smallmatrix} \to 1 \\ \begin{smallmatrix}0\\ \not b\end{smallmatrix} \to 3 \quad \begin{smallmatrix}0\\ 0\end{smallmatrix} \to 4 \quad \begin{smallmatrix}0\\ 1\end{smallmatrix} \to 5 \\ \begin{smallmatrix}1\\ \not b\end{smallmatrix} \to 6 \quad \begin{smallmatrix}1\\ 0\end{smallmatrix} \to 7 \quad \begin{smallmatrix}1\\ 1\end{smallmatrix} \to 8 \end{array} \right.$

E.g.

1	0	1	0	0	$\not b$	$\not b$	$\not b$	...
1	0	0	1	0	0	$\not b$	$\not b$	...

8	4	7	5	4	0	$\not 0$	$\not b$

2-dim. TM



$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{North, East, South, West\}$$

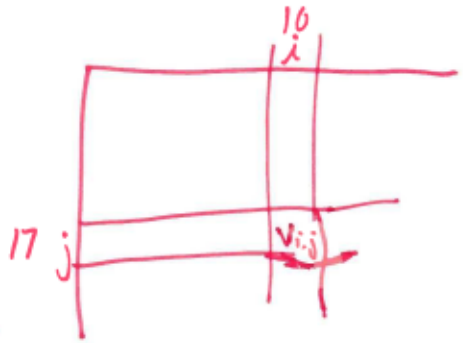
$\mathbb{N}^2 \to \mathbb{N}$

1-1

onto.

("pairing function")

$$f(i,j) = j + \frac{(i+j+1)(i+j)}{2}$$

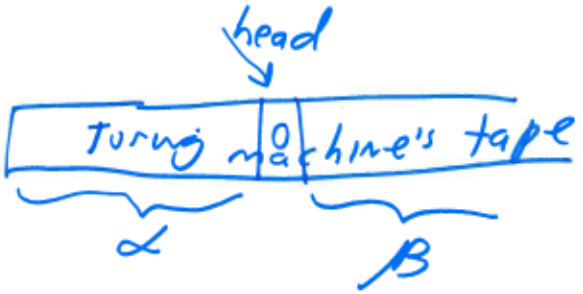


simulation flavor:

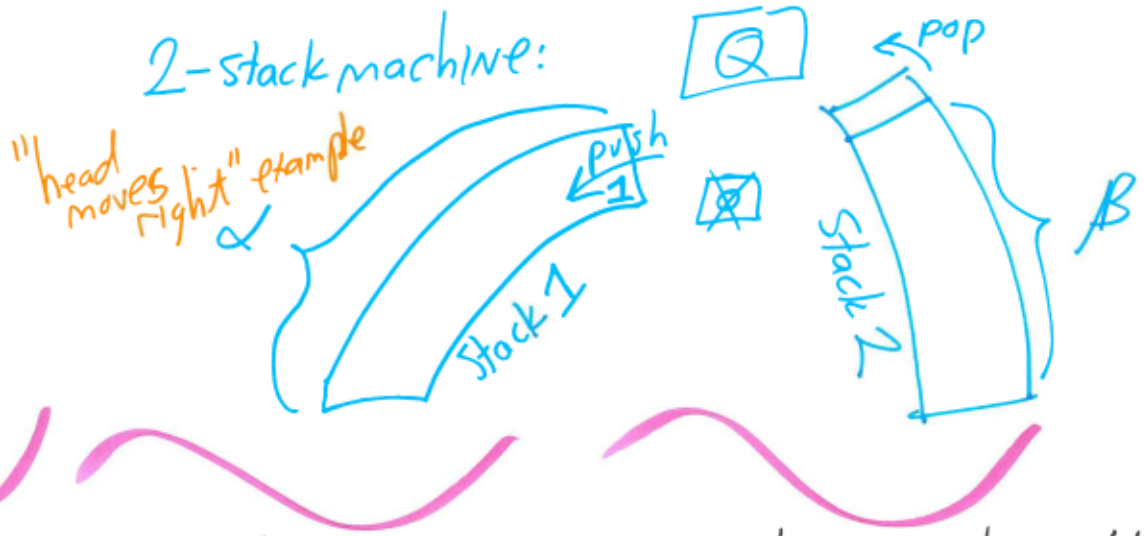
Nerdy note: The input comes in, we will assume, on our bottom tape; we may need to start by appropriately seeding those bits on our "memory" tape — the top tape, of course.

"Weaker Models"

We can simulate at TM by a 2-stack machine:

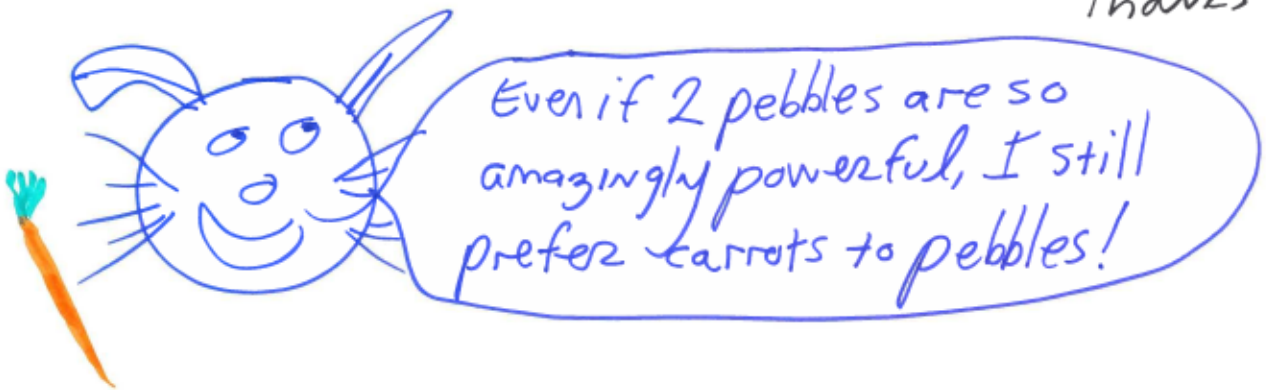


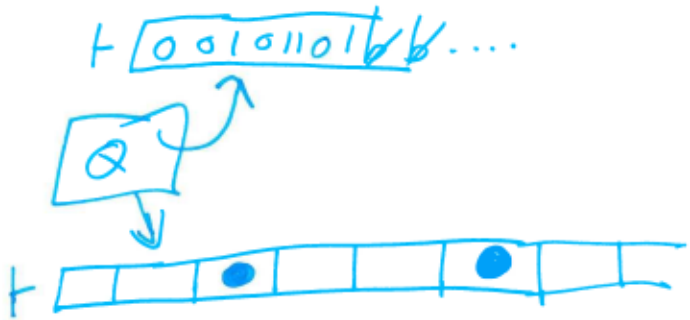
2-stack machine:



- We can simulate a 2-stack machine with a 4-counter machine.
- " " " " 4-counter machine with a 2- " " .
- We " " " " " " 3 pebbles (2 in our model, thanks to the left end marker).

separate read-only input tape, for these models





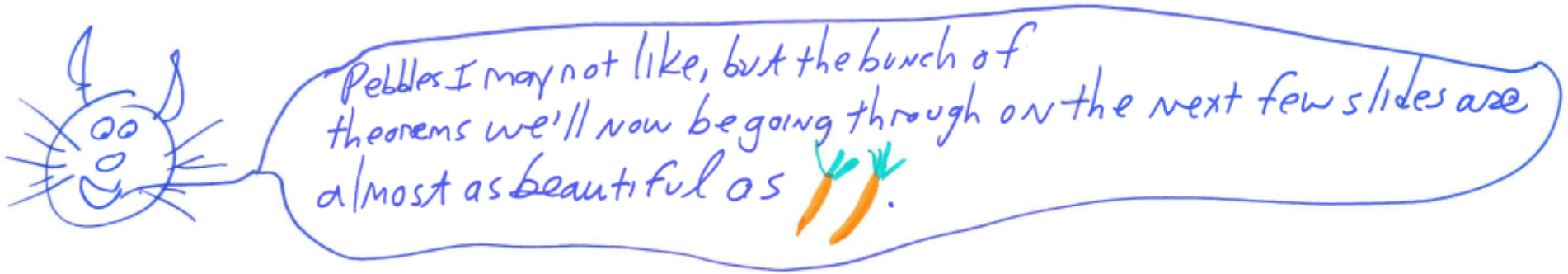
"L is an RE set" means $\exists$ TM M such that $L = L(M)$.

Recall: M accepts x exactly if M halts in an accepting state on input x.
Otherwise, M is said to reject x. Fine print

L is a recursive set exactly if $\exists$ TM M such that $L = L(M)$ and M halts on every input.

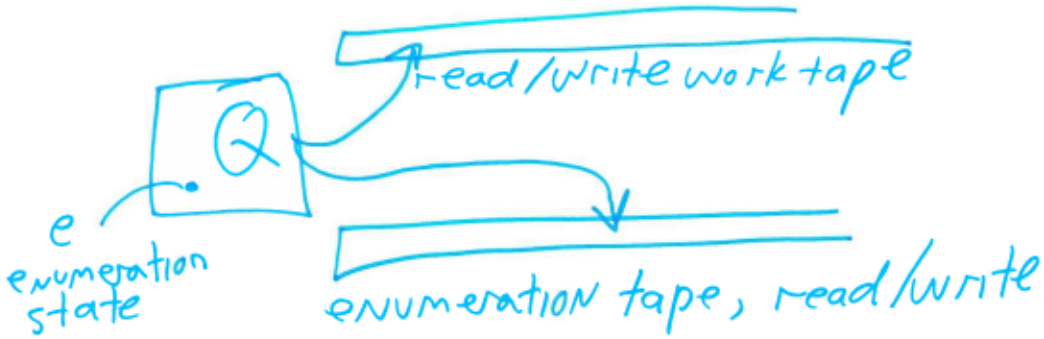
So, being sloppy as there is a machine under the hood of a $\exists$ quantifier, things look like this:

	$x \in L$	$x \notin L$
recursive	halt & accept	halt & reject
R.E.	halt & accept	Reject (possibly by running forever!)



Def. (Recall) A set A is called **recursively enumerable** (RE) if $(\exists \text{TM } M)[A = L(M)]$.

TURING: "(the / a) enumeration machine" — NO INPUT.



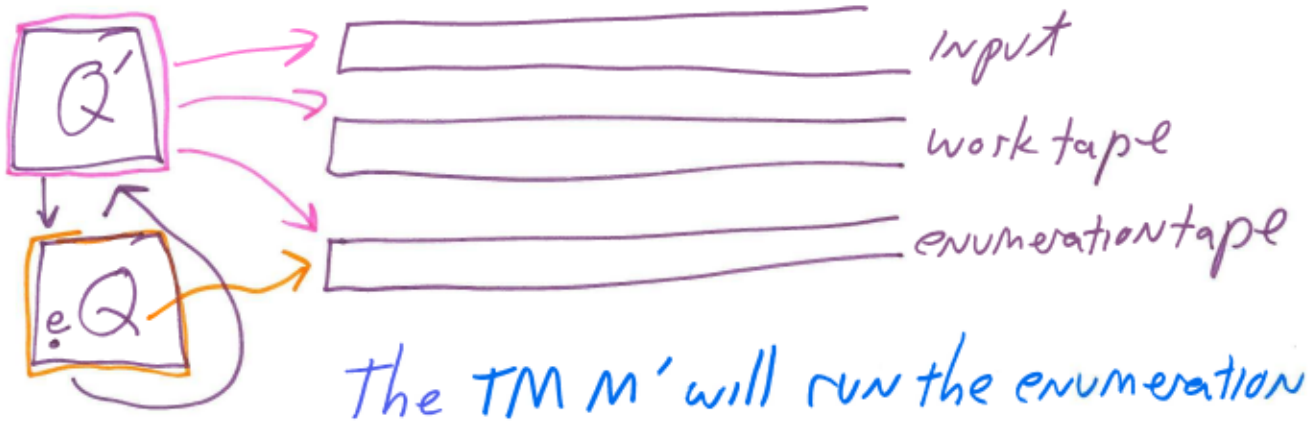
"x is enumerated" (by enumeration machine M) iff the enumeration machine M ever enters into state "e" with x written on its enumeration tape. The machine never halts!!... just continuously enumerates (well, runs, and perhaps enumerates).

$E(M)$ = the set of words enumerated by enumeration machine M.

Theorem $\{L \mid L \text{ is RE (i.e., acceptable; i.e., recursively enumerable)}\}$

$= \{L \mid (\exists \text{ enumerating TM } M)[L = E(M)]\}$.

Pf: $\supseteq$: Given enumeration machine M_j, our goal is to construct a TM M' such that $E(M_j) = L(M')$.



The TM M' will run the enumeration machine as a subroutine, while our input is of course sitting on the input tape. Whenever the enumeration machine enters state "e," control is passed back to M', which checks to see if what is on the input tape is equal to what's on the enumeration tape. If so, ACCEPT, otherwise, continue our simulation of the enum. machine.

$\subseteq$ Given a TM M, build an enumeration machine M' accepting the same language: $E(M') = L(M)$.

I've got this one solved! Simply run, in order, $M(\epsilon)$, then $M(0)$, then $M(1)$, then $M(00)$, and so on, enumerating those x that are accepted!

...

Oh wait!! This is a fatally bad approach!! If $L(M) = \{0000\}$ and M on input ϵ runs forever, my M' will have $E(M') = \emptyset$, since M' will fall into a black hole simulating $M(\epsilon)$ and so won't ever get to simulating $M(0000)$. Horrors! Maybe my friend the dove can help me out here!?!

So... solution: Let s_i be the ith string lexicographically: $s_0 = \epsilon, s_1 = 0, s_2 = 1, s_3 = 00, \ldots$

Our enumeration machine will:

Ply 1 → Run $M(s_0)$ for 1 step.

Ply 2 → Run $M(s_0), M(s_1)$ for 2 steps.

Ply 3 → Run $M(s_0), M(s_1), M(s_2)$ for 3 steps.

Etc.

this type of interleaving is the sort often called dovetailing

And during this process, every time an $M(s_k)$ accepts, we enumerate s_k.

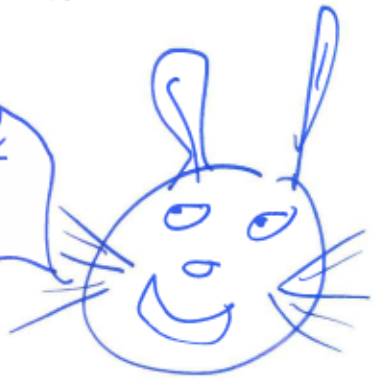
Why is this correct?

. If $x \in L(M)$ then M accepts x in some finite # of steps, so M' will eventually enumerate x. (Can you describe in which ply this will happen?)

. If $x \notin L(M)$ then this scheme clearly will never enumerate x.

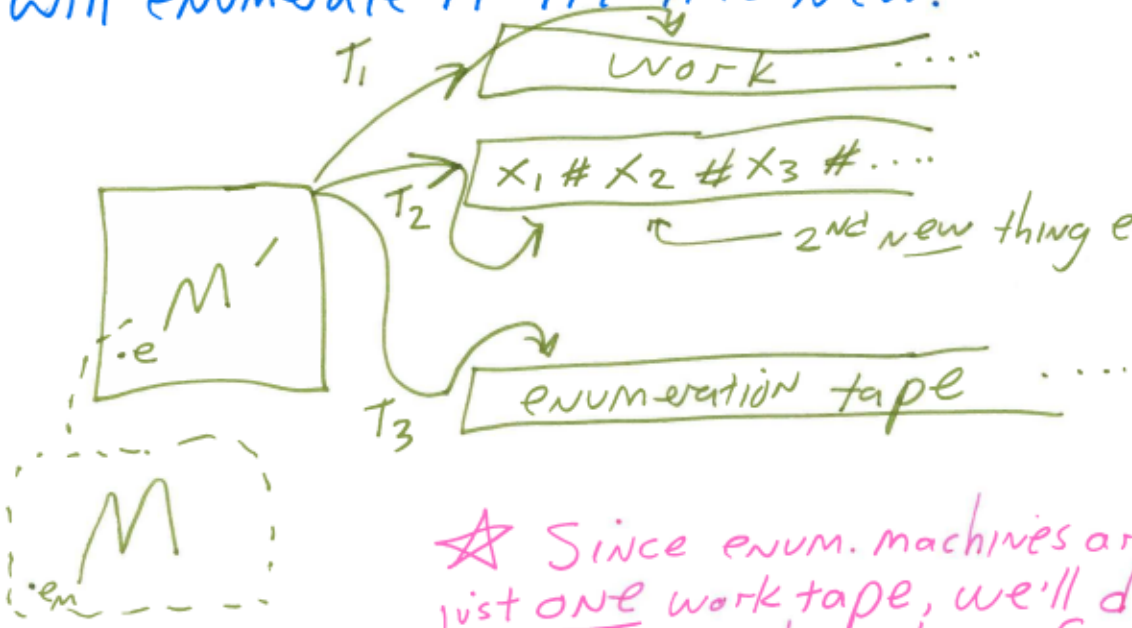
QED

Theorem If L is R.E. then L is enumerated by an enumeration machine that never repeats elements.



Proof Sketch L is R.E., so by the previous theorem it has an enumeration machine M, i.e., $E(M)=L$. Consider M', which simulates M but keeps track, on an extra tape, ☆ of all strings that M has enumerated. Every time M enumerates something, M' will enumerate it iff it is new.

□



☆ Since enum. machines are allowed (an enum. tape and) just one work tape, we'll do T_1 & T_2 on one tape using the enum. mach. analgue of our reg. TM "2 tape → 2 tracks → 1 tape" work.

Theorem If L is enumerated in order, from smallest strings onward, then L is recursive, and vice versa.

Pf $\Rightarrow$ Case 1: L is finite: Then L is recursive.

Case 2: L is infinite: Let M be the "in-order" enumeration machine for L (that the "$\Rightarrow$" hypothesis gives).

Note that this lets us build a total* TM for L:

$x \notin L$ iff M outputs some string longer than x at a moment when it has not yet output x.

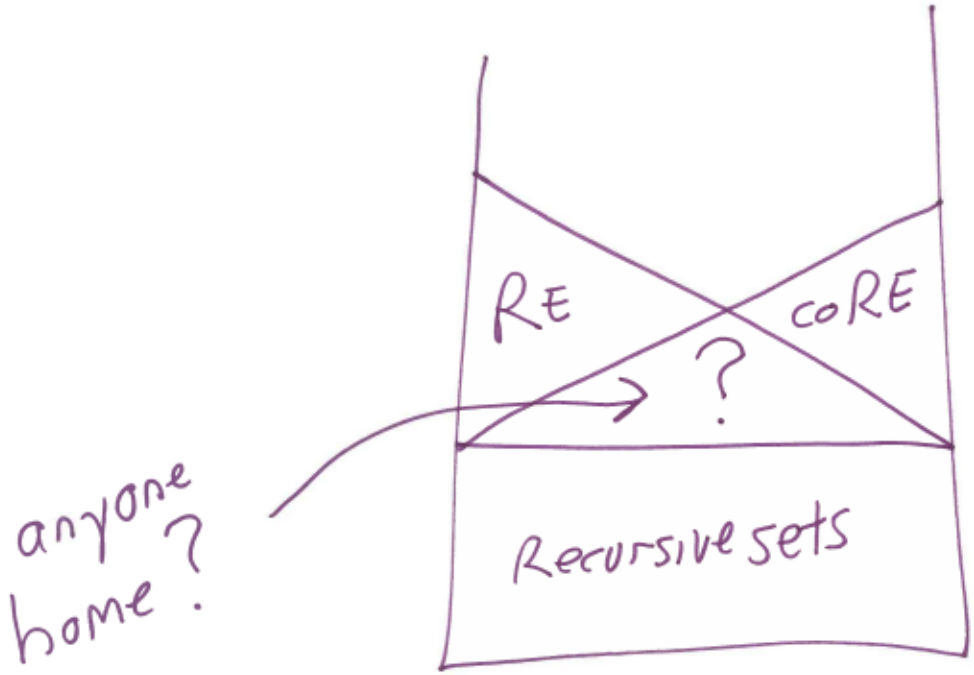
Example: $x = 110000$

00
01
11010
11101
111111110

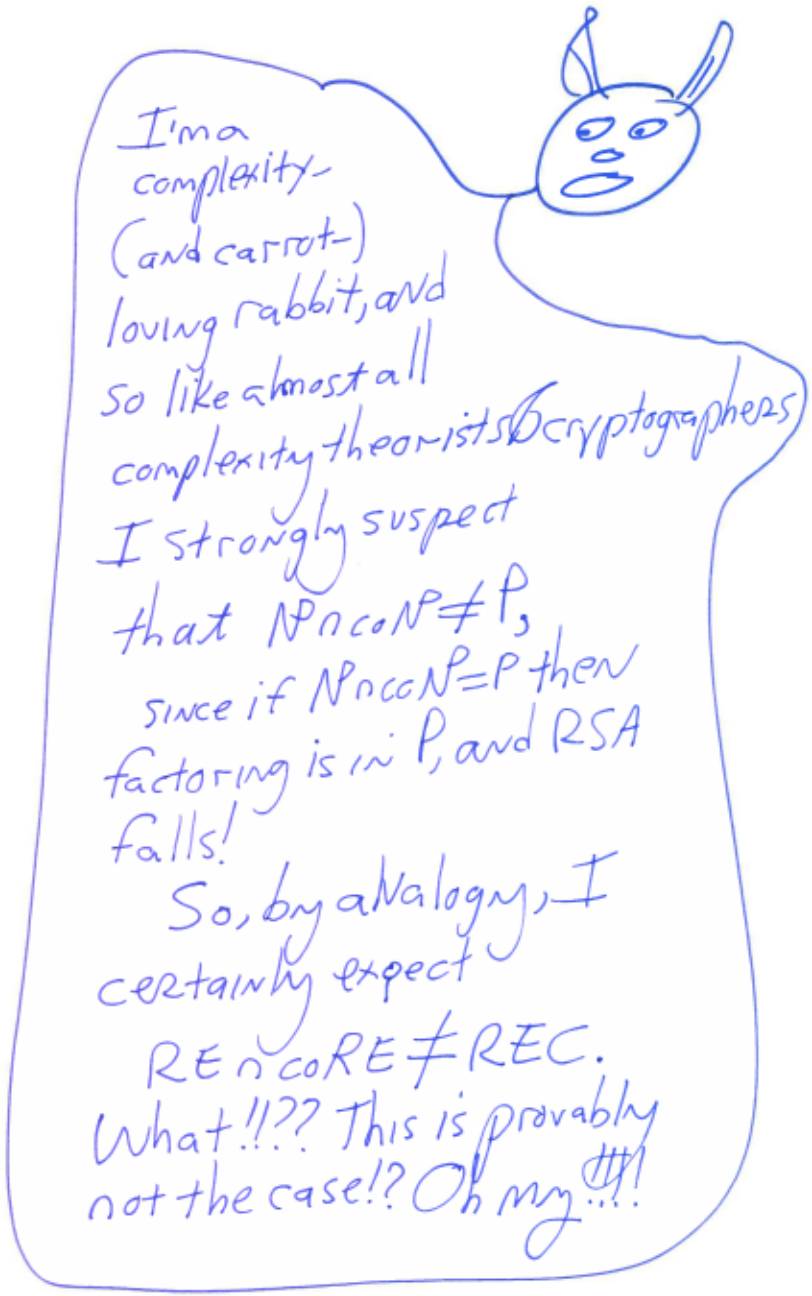
$\Leftarrow$ Run the $(a)_{\Sigma^*}$ total machine for L on, in sequence (no need* for dovetailing!), $\epsilon, 0, 1, 00, 01, 10, 11, 000, \ldots$, and enumerate each as it accepts.

QED

* A TM is total iff $(\forall x)$[M halts on input x].
* Indeed, it would do us harm by mucking up the order!



Def. L is coRE iff $\overline{L}$ is RE.



Theorem If L is R.E. and $\bar{L}$ is R.E., then L is recursive.

Pf. sketch

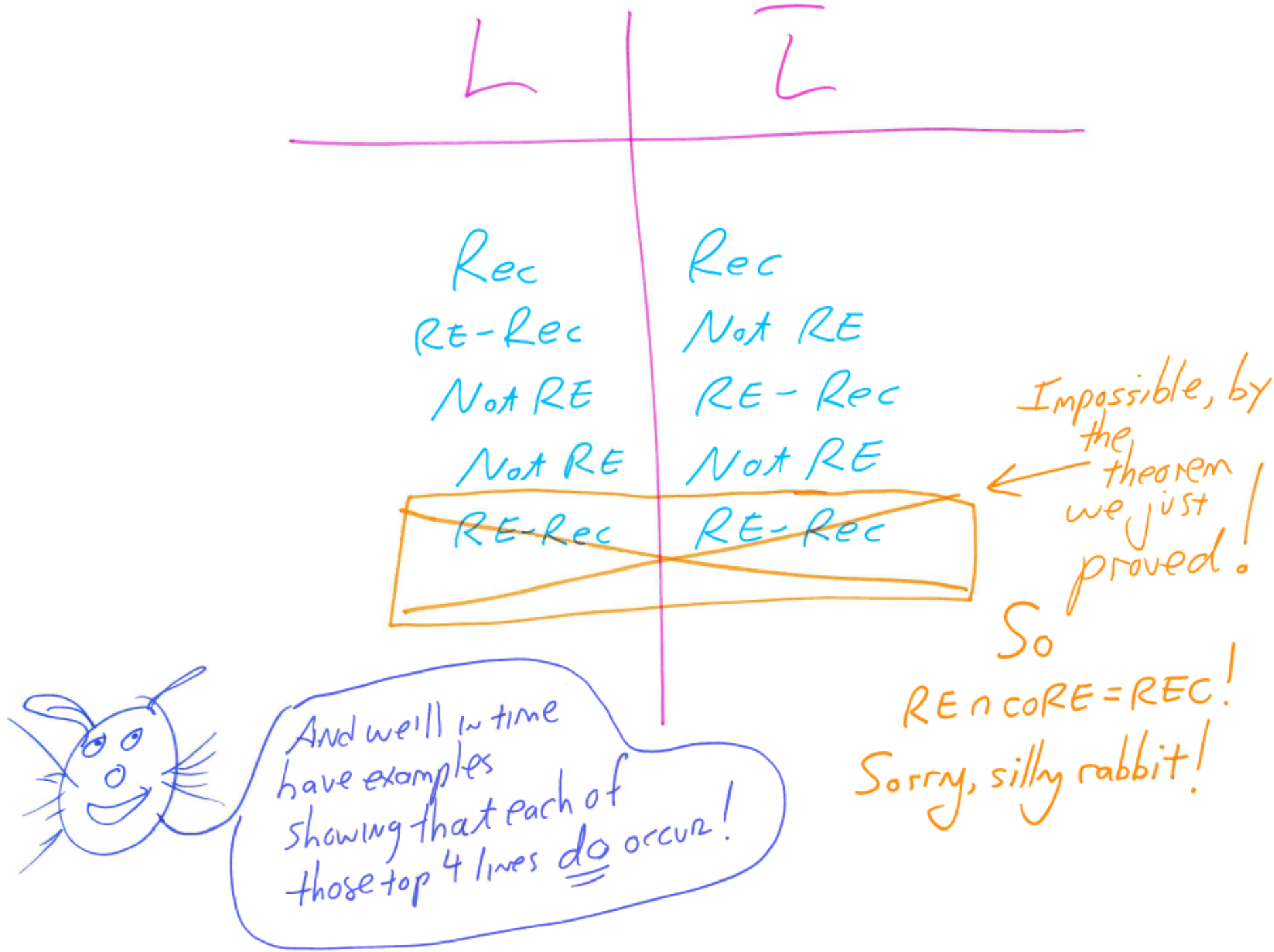
E_L → [enumerator of L] → 01010, 0, 111, etc.

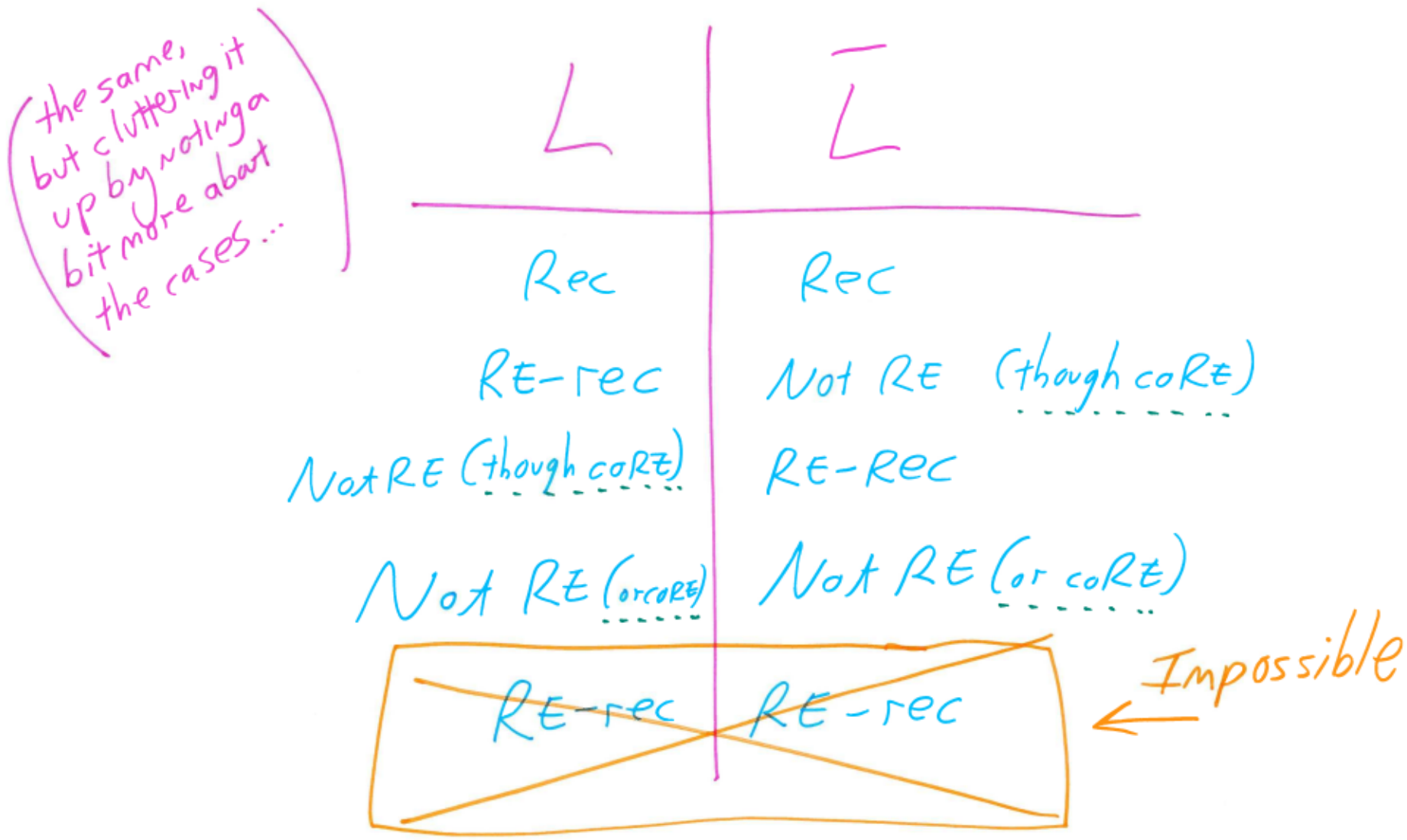
$E_{\bar{L}}$ → [enumerator of $\bar{L}$] → 11, 1011110, 010000, etc.

Here is a "program" (that always halts!) for L:

On input x, run both E_L and $E_{\bar{L}}$ in (sigh) "parallel" (really: do a type of dovetailing, e.g. E_L and $E_{\bar{L}}$ alternate being run for 1 more step), and wait until one outputs x (one eventually will) (why?), and if that one is E_L, then accept, but if that one is $E_{\bar{L}}$ then reject. (Why don't we have to worry that the "other" one would later have enumerated x, suggesting the opposite answer?)

QED

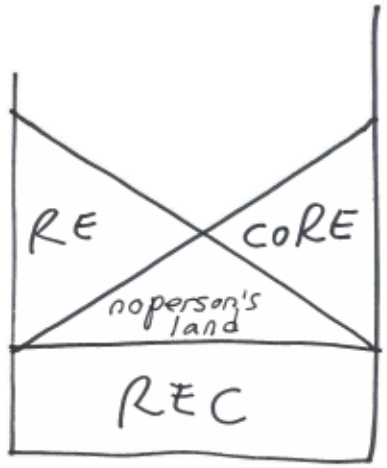

L
$\overline{L}$
Rec
Rec
RE-Rec
Not RE
Not RE
RE-Rec
Not RE
Not RE
RE-Rec
RE-Rec
Impossible, by the theorem we just proved!
So
RE ∩ coRE = REC!
Sorry, silly rabbit!
And we'll in time have examples showing that each of those top 4 lines do occur!


the same, but cluttering it up by noting a bit more about the cases...
L
$\bar{L}$
Rec
Rec
RE-rec
Not RE (though coRE)
Not RE (though coRE)
RE-Rec
Not RE (or coRE)
Not RE (or coRE)
RE-rec
RE-rec
Impossible

Let $M_1, M_2, \ldots$ be a (std., nice) list of all TM descriptions. Σ^*_{HU}

Universal TM:

$U(i\#x)$ simulates $M_i(x)$.



$HP = \{x \mid M_x(x) \text{ accepts}\}$. $\Sigma^*_{N \leftrightarrow \Sigma^*, \text{"}\_\_\text{"}}$

$\overline{HP} = \{x \mid M_x(x) \text{ rejects}\}$.

HP is r.e. but not recursive.

$\overline{HP}$ is not even RE (it however is coRE).

Recursive $\equiv$ decidable.

HP is undecidable.

How?

Technique: Diagonalization (Cantor, 1890s).

The next slide almost makes my head explode!

It is amazing that such a

HUGE

result can have such a short (and somewhat magically delightful, slight-of-hand-y) proof.

Assume (hoping for a contradiction) that $\overline{HP}$ is r.e.

So, since $\overline{HP}$ is assumed r.e., it is accepted by some TM, say M_k. But

If $M_k(k)$ accepts then $k \notin \overline{HP}$ (by def. of $\overline{HP}$).

If $M_k(k)$ accepts then $k \in \overline{HP}$ (by our assump. that $L(M_k) = \overline{HP}$).

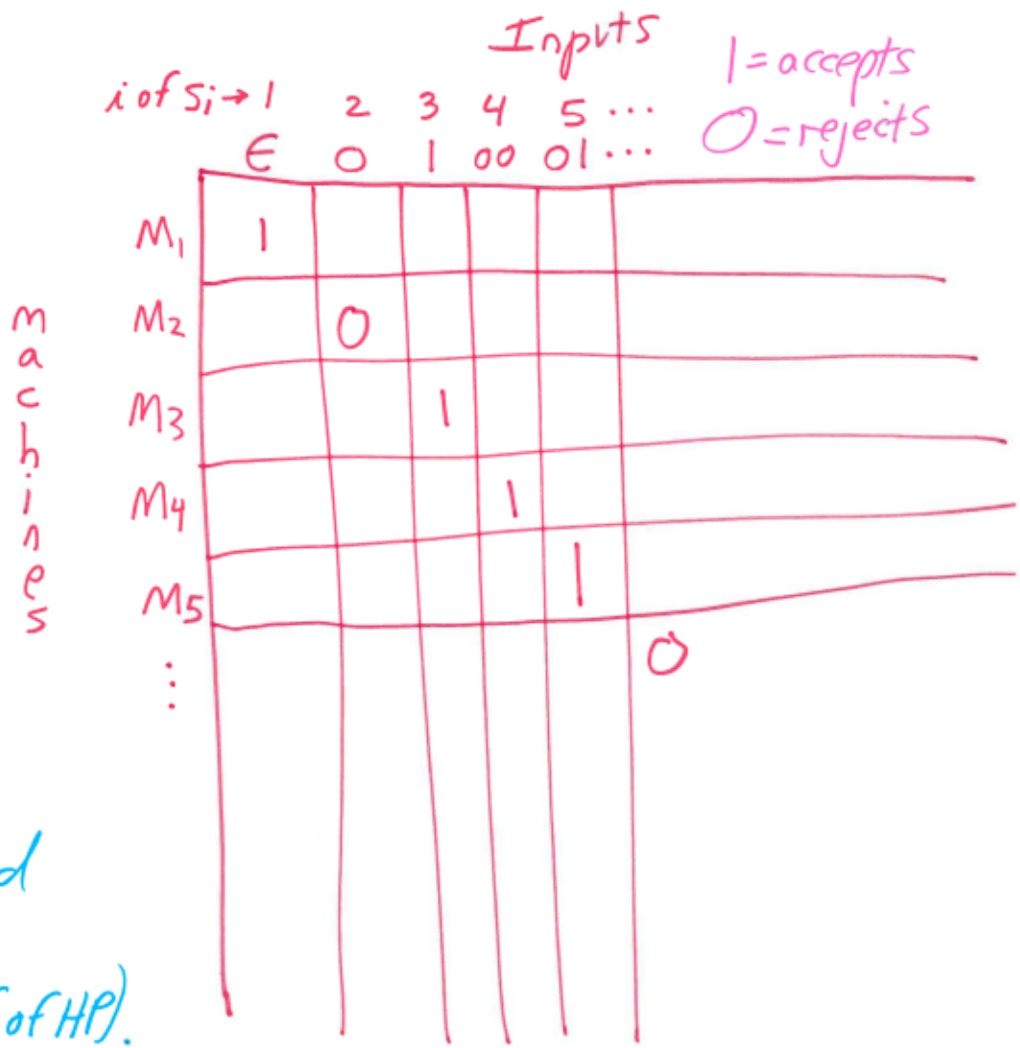
So if $M_k(k)$ accepts we have a contradiction.

But if $M_k(k)$ rejects, a contradiction similarly holds. (Try it! Or, read on:

If $M_k(k)$ rejects then $k \notin HP$ (by def. of HP).

If $M_k(k)$ rejects then $k \in HP$ (by our assump. that $L(M_k) = \overline{HP}$).)

Thus no M_k accepts $\overline{HP}$. So $\overline{HP}$ is not r.e.!



We've just shown:

(★) $\overline{HP}$ is not R.E.

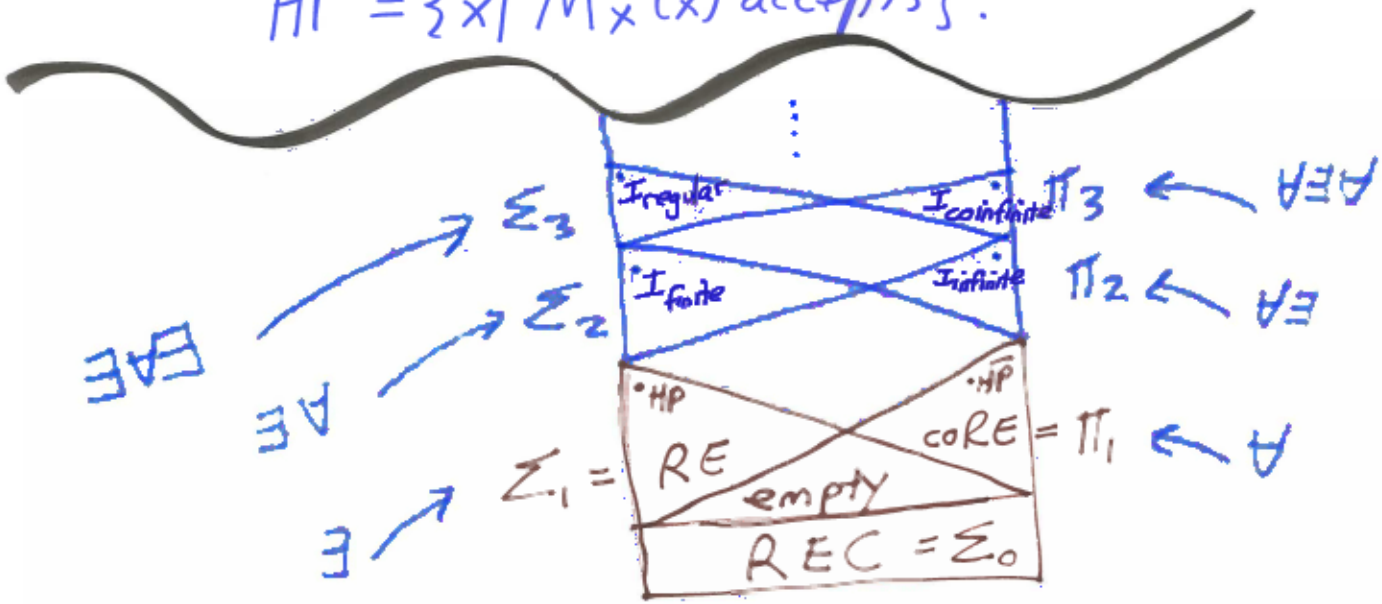
What about HP?

Well, (★) $\Rightarrow$ HP is not recursive. (Why? $L \in REC \Rightarrow \overline{L} \in REC$.)

Indeed, (★) $\Rightarrow$ HP is not coRE. (Why? It is just definition-caused that $(\forall A)[A \in RE \Leftrightarrow \overline{A} \in coRE]$ Is Def. of coRE. So $\overline{HP} \notin RE \Leftrightarrow HP \notin coRE$. Just equival. things!)

Claim HP is r.e.

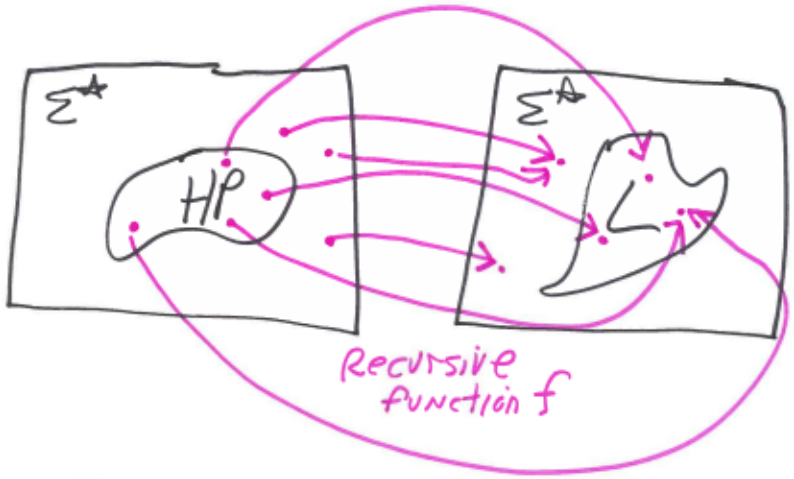
$HP = \{x \mid M_x(x) \text{ accepts}\}$.



"Tool A"

Theorem (A tool for proving $L \notin REC$)

If there is a recursive function f such that



$(\forall x)\,[x \in HP \text{ iff } f(x) \in L]$, ☆

Then L is undecidable.

Pf. Suppose L is decidable. Then so is HP. Why?

$x \in HP \iff \underbrace{f(x) \in L}_{\text{decidable}}$.

QED

Nerdy note: The theorem is "leaving money on the table"...
but we'll soon grab that loot too...

☆ **Def.** $A \leq_m B$ iff $(\exists$ rec. fcn $f)(\forall x)[x \in A \iff f(x) \in B]$.
(recursive) many-one reduction
So this says "If $HP \leq_m L$, then L is undecidable."

Theorem $L = \{\langle i, x\rangle \mid M_i(x) \text{ halts}\}$ is r.e. but not recursive.

That is not HP though maybe it should be... and in some places...)

Pf L is clearly r.e. Let us show it is not recursive by applying Tool A.

Goal Construct a recursive reduction σ such that $(\forall x)[x \in HP \text{ iff } \sigma(x) \in L]$.

$\sigma(x)$: outputs $\langle i_x, y\rangle$.

What is i:

i_x { (The index of) A Turing machine that (on arb. (i.e., all) input, z) does:
① erase its input tape
② write x on its input tape.
③ Simulate M_x on the input tape, except, if M_x is about to halt and reject, go into an infinite loop.

y { Fred$_{\text{binary}}$.

It is easy to see that, for each x, $x \in HP$ iff $\sigma(x) \in L$ (i.e. $\langle i_x, \text{Fred}\rangle \in L$).

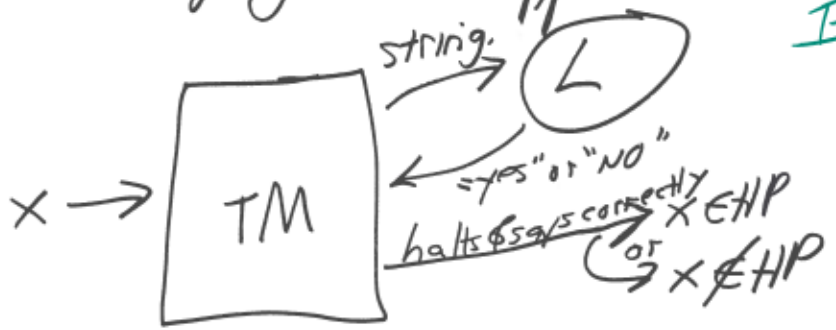
Thus by Tool A, L is not recursive. QED

Nerdy note: We'll see that L (and HP) are "RE-complete" later.

"Tool B"

Theorem (An even more flexible tool for proving $L \notin REC$)

Let L be a language and suppose



I: multiple queries ok

(i.e., $HP \leq_T L$)

Then L is undecidable.

Pf. (See it!) I: do QED

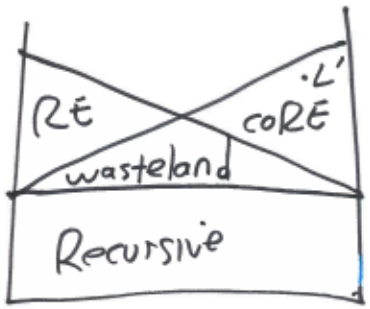
$A \leq_m A' \Rightarrow A \leq_T A'$. (Why?)

Can we prove $\Leftarrow$? No, not in general! Because we know:

$HP \leq_T \overline{HP}$ (why?)

but $HP \not\leq_m \overline{HP}$. (why?)

Theorem $L' = \{i \mid M_i \text{ accepts no strings}\} = \{i \mid L(M_i) = \emptyset\}$ is coRE but not recursive.



Claim L' is coRE.

I.e. $\overline{L'}$ is RE. $\overline{L'} = \{i \mid (\exists x)[M_i(x) \text{ accepts}]\}$.

Reason 1 There is an enumeration machine for $\overline{L'}$.

(Can you see it directly, rather than just by combining Reason 2 with our earlier thm. about enum. TM's covering exactly the RE sets?) (Hint: Big, fat clove!) ✡

Reason 2 There is a TM, M, so $L(M) = \overline{L'}$.

(Namely: $M(i)$: In a dovetailed fashion, run $M_i(\epsilon), M_i(0), M_i(1), \ldots$ and accept as soon as any of these accept.

✡ Which is actually what one would get by unwinding the proof/construction implicit in the "rather than"....

Recall: $L' = \{i \mid L(M_i) = \emptyset\}$

Let us show L' is not decidable. We'll connect it to HP's hardness!

Consider this recursive function σ:

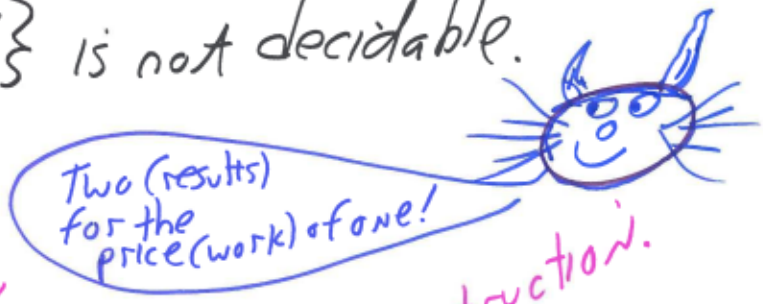
$\sigma(x)$: Output a TM M such that on input y, M simulates $M_x(x)$.
If $M_x(x)$ accepts then M accepts.
Otherwise (perhaps because $M_x(x)$ diverges [P.]) M rejects.

Claim $x \in HP \iff \sigma(x) \notin L'$. (Can you see/check this?)

Thus by Tool B, L' is undecidable.

Nerdy note: We could for this set L' have made an indirect argument using Tool A, namely, using Tool A (by exactly the above construction!!) to show $HP \leq_m L'$ (as the above constr. shows exactly that) and so by Tool A, $\overline{L'} \notin REC$, and so since $REC = co\text{-}REC$, $L' \notin REC$. [P.] Friedberg-Muchnik 1950's (solved Post's problem; $\exists \geq 2$ T-degrees in Σ_1^0) led to Young (Rogers p. 177) separating for Σ_1^0 btt-comp from m-comp.; so Tool B has its day... even within Σ_1^0.

Theorem $L'' = \{ i \mid L(M_i) = \Sigma^* \}$ is not decidable.



Pf. See previous page!

$x \in HP$ iff $\sigma(x) \in L''$.

↑ the σ from the previous page's construction.

QED

A Much Stronger Claim About L'':

Thm L'' is neither RE nor coRE.

Question: How do we prove things to be non RE?

(Then we can RETURN and prove this!)

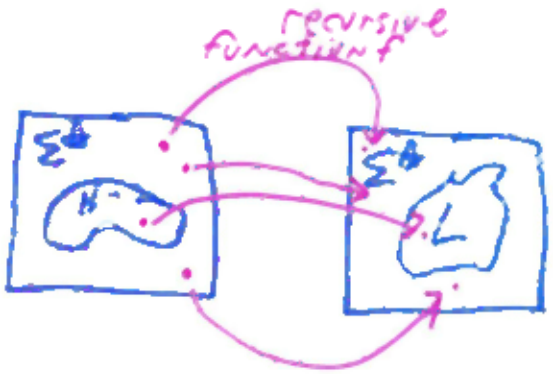
the underpinning of the tool

Theorem (A tool for proving sets are $\notin RE$, and it can also be used on complements to prove that sets are $\notin coRE$.)

If there is a recursive function f such that

$$(\forall x)[x \in H \text{ iff } f(x) \in L]$$

(i.e., if $H \leq_m L$),



THEN "if L is r.e. then H is r.e."

(why?)

How does this help us prove things to be non RE? Like this:

Tool C

Corollary If $\overline{HP} \leq_m L$, then we can conclude that L is not r.e.

Pf. Set $H = \overline{HP}$ in the thm. above, so the above says $(\overline{HP} \leq_m L \wedge L \in RE) \Rightarrow \overline{HP}$ is RE. But we have proven already that $\overline{HP} \notin RE$. So $\overline{HP} \leq_m L \Rightarrow L \notin RE$. QED

Goal $\overline{HP} \leq_m L'' = \{i \mid L(M_i) = \Sigma^*\}$.

By Tool C, showing this yields $L'' \notin RE$.

Recall: $\overline{HP} = \{i \mid M_i(i) \text{ does not accept}\}$.

(Recursive function) $\sigma(x)$: Outputs a machine M with the following behavior. On input y, M accepts iff $M_x(x)$ does not accept within $1y_{binary}$ steps. I.e. exam

Claim $x \in \overline{HP}$ iff $\sigma(x) \in L''$. (Thus L'' is not r.e.)

$x \in \overline{HP} \Rightarrow M_x(x)$ rejects.

$\Rightarrow (\forall y)[M_x(x)$ does not accept within $1y$ steps$]$.

$\Rightarrow L(M_{\sigma(x)}) = \Sigma^*$.

$\Rightarrow \sigma(x) \in L''$.

$x \notin \overline{HP} \Rightarrow M_x(x)$ accepts.

$\Rightarrow (\exists y_0)(\forall y >_{lex} y_0)[M_x(x)$ accepts within $1y$ steps$]$.

$\Rightarrow L(M_{\sigma(x)}) \neq \Sigma^*$. (TANGERINE)

$\Rightarrow \sigma(x) \notin L''$.

Thus $x \in \overline{HP}$ iff $\sigma(x) \in L''$. QED for the claim.

Warning: Never use "$A \Rightarrow B \Rightarrow C$" to mean "$A \Rightarrow B \wedge B \Rightarrow C$," as I do above and on some other slides, as interpreted mechanically it does not mean that.

Goal Prove that L'' is not coRE.

$$L'' = \{i \mid L(M_i) = \Sigma^*\}.$$

How? Our above goal is equivalently stated as $\overline{L''}$ is not RE.

aHA! So by Tool C,

proving $\overline{HP} \leq_m \overline{L''}$ (equivalently $HP \leq_m L''$)

will suffice!

Let us do that:

$\sigma(x)$: Output a machine M with the following behavior:

On input y, M will simulate $M_x(x)$. If $M_x(x)$ accepts then M accepts. I: straight-out simulation, wrt rej. & ∞ loop..., inherited; "Immersive"

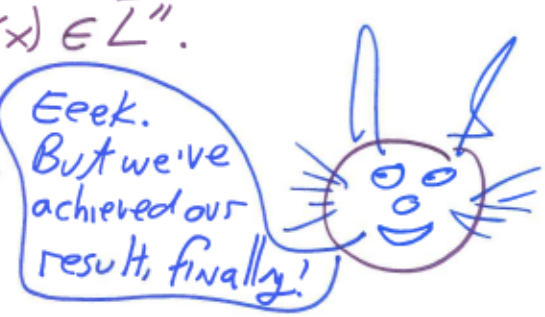
Note that (pardon the "$\Rightarrow$" chaining naughtiness ☹):

$x \in \overline{HP} \Rightarrow L(M_{\sigma(x)}) = \phi \Rightarrow \sigma(x) \notin L'' \Rightarrow \sigma(x) \in \overline{L''}$.

And $x \notin \overline{HP} \Rightarrow L(M_{\sigma(x)}) = \Sigma^* \Rightarrow \sigma(x) \notin \overline{L''}$.

So, as σ is a rec. fcn., we have shown $\overline{HP} \leq_m \overline{L''}$.

And so L'' is not coRE.



So $L'' \notin$ REUcoRE is now established, by 2 uses of Tool C.

Let us do another example!

Theorem $L = \{i \mid L(M_i) \text{ is } \infty\}$ is not r.e. & not coRE.

Claim: We'll get this almost for free — as the previous proof's constructions yield what we need! That is, it is a "cor. to the proof" — though not a "cor. to the thm."

Proof $\overline{HP} \leq_m \overline{L}$ by the σ of the previous slide (because ϕ is finite and Σ^* is infinite). So by Tool C, $\overline{L}$ is not RE (i.e., L is not coRE).

$\overline{HP} \leq_m L$ by the σ of <u>two</u> slides ago (because Σ^* is infinite and the set in line "TANGERINE" that on that slide is merely asserted to $\neq \Sigma^*$ in fact is proven by what comes one line before it to even be finite). So by Tool C, L is not RE.

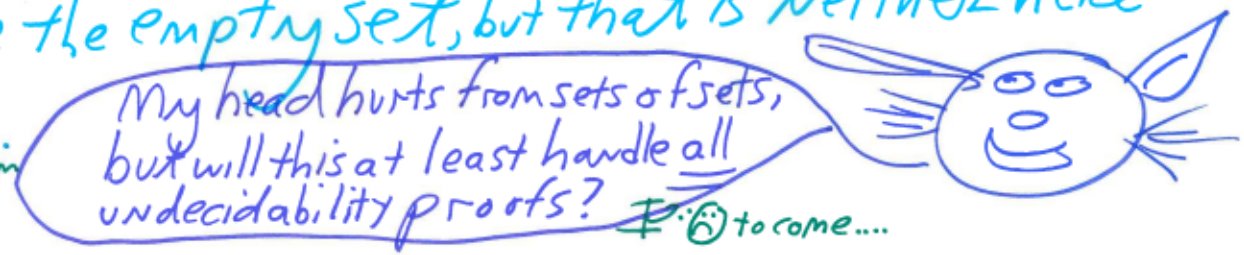
QED

Theorem (Rice's Theorem) Any nontrivial property of the r.e. languages is undecidable.

Definitions Let $\mathcal{C}$ be a set of r.e. languages, each a subset of $\{0,1\}^*$.

- $\mathcal{C}$ is said to be a property of the r.e. languages. E.g. examples, beware "nRE" $\subseteq \{0,1\}^*$
- We say a language L has property $\mathcal{C}$ iff $L \in \mathcal{C}$.
- $\mathcal{C}$ is a trivial property of the r.e. languages iff: $\mathcal{C} = \emptyset$ or $\mathcal{C}$ is all r.e. sets.
- $\mathcal{C}$ is said to be (a) decidable (property of the r.e. languages) iff $\{i \mid L(M_i) \in \mathcal{C}\}$ is decidable.

Careful! $\mathcal{C} = \emptyset$ means $\mathcal{C}$ is empty. So $\mathcal{C} = \{\emptyset\}$ is NOT a trivial property of the r.e. languages, since $\mathcal{C} = \{\emptyset\}$ is not empty... it contains one set (that itself happens to be the empty set, but that is neither here nor there). E.g. idiom



Proof 1 of Rice's Theorem (a clean, natural proof!)

We're given $\mathcal{C}$, a nontrivial prop. of the RE languages.

W.l.o.g., ϕ does not have property $\mathcal{C}$ (if not, consider RE − $\mathcal{C}$)

Since $\mathcal{C}$ is nontrivial, there is an i_0 such that $L(M_{i_0})$ has property $\mathcal{C}$. (★★)

Goal Show $HP \leq_m \{i \mid L(M_i) \in \mathcal{C}\}$.

$\sigma(x)$: Outputs a TM M with the following behavior:

$M(y)$ simulates $M_x(x)$ until $M_x(x)$ accepts. (If $M_x(x)$ does not accept, $M(y)$ will not accept.) If $M_x(x)$ accepts, then do an immersive simulation of $M_{i_0}(y)$.

Note that σ is a rec. fcn., and $(\forall x)[x \in HP \iff \sigma(x) \in \{i \mid L(M_i) \in \mathcal{C}\}]$.

So $\{i \mid L(M_i) \in \mathcal{C}\}$ is undecidable. QED

Hold on! We just established that $HP \leq_m \{i \mid L(M_i) \in \mathcal{C}\}$. So by Tool C, we can conclude that every nontrivial prop. of the RE sets is not coRE. Please fork over my Turing Award! And my Fields Medal (as $\{i \mid L(M_i) = \phi\} \in$ coRE... kaBoom!).

Is our rabbit right??

Proof 2 of Rice's Theorem (proof by a powerful, opaque tool)

Timeout

Let $M_1, M_2, \ldots$ be our standard (nice) enum. of TMs. ? even any acceptable enum.

Theorem (The Recursion Theorem) Let f be any recursive function. Then there is an i such that $L(M_i) = L(M_{f(i)})$.

We will give the famously opaque proof soon.

Time in

Let $\mathcal{C}$ be a nontrivial prop. of the r.e. sets. Say $L(M_{i_0})$ has property $\mathcal{C}$ and $L(M_{i_1})$ does not have property $\mathcal{C}$.

Assume (hoping for a contradiction) that $\mathcal{C}$ is decidable (i.e., $\{i \mid L(M_i) \in \mathcal{C}\}$ is decidable). Consider the map (which is rec. under our assumption)

$$f(i) = \begin{cases} i_0 & \text{if } L(M_i) \text{ does not have property } \mathcal{C} \\ i_1 & \text{if } L(M_i) \text{ has property } \mathcal{C}. \end{cases}$$

But recursive fcn. f has no fixed point in the rec. thm. sense (i.e., $(\forall i)\,[L(M_i) \neq L(M_{f(i)})]$ — no "fixed point"). This is impossible by the recursion thm. So $\mathcal{C}$ must in fact be decidable. QED

Theorem (The Recursion Theorem) Let f be any recursive function. Then there is an i such that $L(M_i) = L(M_{f(i)})$.

Proof

Let $g(i)$ be the index of the TM that, on input x, runs the $M_i(i)$th TM on input x. So $M_{g(i)}(x) = M_{M_i(i)}(x)$, $\forall i, x$.

(☆)

- Sip, Du-Ko seek nonopaque
- $M_i(i)$ viewed as fcn, partial
- $M_{\downarrow}(x)$ itself $\downarrow$. Still, creepy, iffy. (HU)

Note that g is total recursive (even when $M_i(i)$ diverges).

Consider $f \cdot g$, i.e. $f(g(\cdots))$, which clearly is recursive.

(☆☆) Let j be the index of $f \cdot g$.

Then we claim $g(j)$ is the fixed point! Why?

$$M_{g(j)}(x) \underset{\text{by ☆}}{=} M_{M_j(j)}(x)$$

$$\underset{\text{by ☆☆}}{=} M_{f \cdot g(j)}(x)$$

$$= M_{f(g(j))}(x)$$

Let $g(j) = i$. We have shown $M_i(x) = M_{f(i)}(x)$ for all x.

QED

There are many, many problems that Rice's Theorem does not speak to at all. It is very focused as to what it covers: language properties of the r.e. sets.

Example where Rice does not help

- $\{i \mid M_i$ on some input, runs for an odd number of steps$\}$.

Theorem (Rice II) Let $\mathcal{C}$ be a property of the r.e. languages.
$\mathcal{C}$ is r.e. (by which we mean $\{i \mid L(M_i) \in \mathcal{C}\}$ is r.e.) iff 1 & 2 & 3 hold.

(1) If L has $\mathcal{C}$, and $L \subseteq L'$, and L' is r.e., then L' has $\mathcal{C}$.

(2) If L is an infinite language in $\mathcal{C}$, then there is a finite subset of L having prop. $\mathcal{C}$.

(3) There is a TM that enumerates (coded versions of the string-sets of) all finite languages in $\mathcal{C}$.

Reductions Between Problems

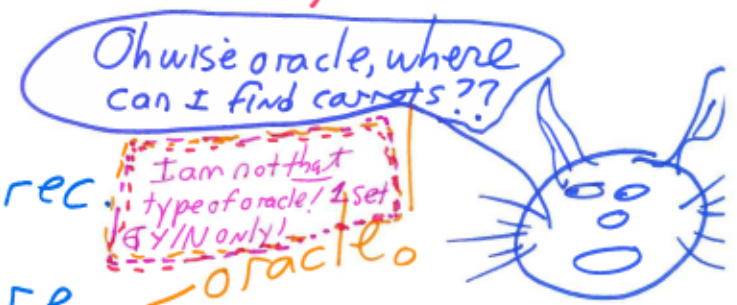
(Recall:) Def. $A \leq_m B$ iff there is a recursive fcn. f such that

$(\forall x)[x \in A$ iff $f(x) \in B]$. "many-one reduction"

Theorem (shuffled versions of our tools!)[*]

a) If $A \leq_m B$ and B is rec., then A is rec.

b) If $A \leq_m B$ and B is r.e., then A is r.e.

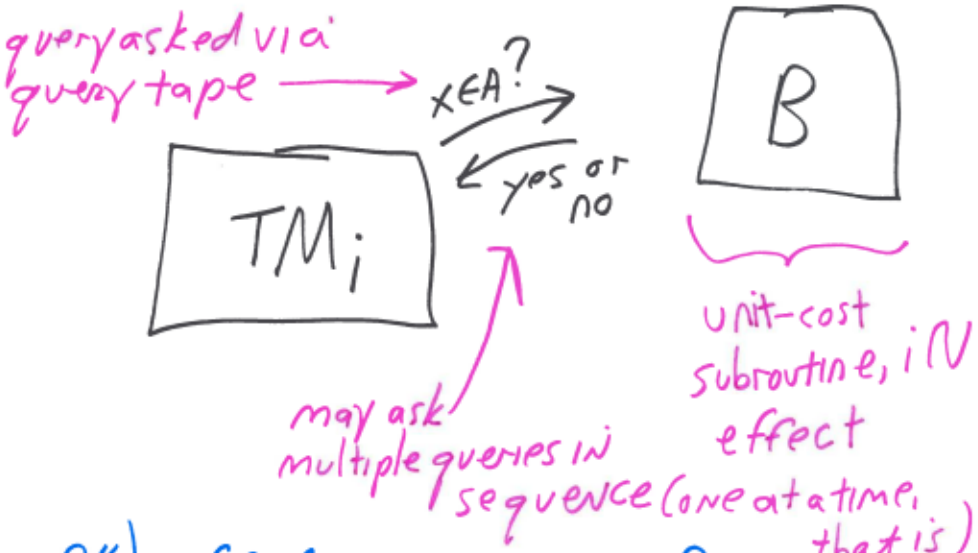


Def. A is recursive in B iff $A = L(M_i^B)$ for some i such that M_i^B always halts.

Oracles A TM with oracle B ($B \subseteq \Sigma^*$) is a TM with three special new states, $q_?$, q_y, and q_n, and a query tape, and when $q_?$ is entered, by magic we enter q_y if the string on the query tape is in A, and q_n if the string on the query tape is not in A.

[*] (a) is a more general version of the idea used in the proof of Tool A, and (b) is exactly the "underpinning" theorem stated just before Tool C.

Picture of oracle setup: M_i^B (informal)



Def. $A \leq_T^B$ ("A Turing reduces to B") iff A is recursive in B

(i.e., $(\exists i)[L(M_i^B) = A \wedge (\forall x)[M_i^B(x) \text{ halts}]]$).

Fact If $A \leq_m B$ then $A \leq_T B$.

Theorem (more general version of an idea used in Post/B's proof)

If $A \leq_T B$ and B is rec. then A is rec.

NOT TRUE: If $A \leq_T B$ and B is r.e. then A is r.e.

For example, HP is r.e. and $\overline{HP}$ is not r.e., yet

$\overline{HP} \leq_T HP$.

Complete Sets

Def. A set B is $\leq_m$-complete for a class $\mathcal{C}$ (often written "B is $\mathcal{C}$-complete") iff (1) $(\forall D \in \mathcal{C})[D \leq_m B]$, and (2) $B \in \mathcal{C}$.

Theorem (an RE-complete set) $U_{\Sigma_1} = \{\langle i, x\rangle \mid M_i(x) \text{ accepts}\}$ is Σ_1-complete. $I^0_{\Sigma_1 = RE}$

Proof $U_{\Sigma_1} \in RE$ is clear.

Say we're given an arb. RE set L. So $L = L(M_{\hat{i}})$, for some $\hat{i}$. For that $\hat{i}$, note that (yes, again, don't write iff chains as I'm about to!) $x \in L$ iff $x \in L(M_{\hat{i}})$ iff $\langle \hat{i}, x\rangle \in U_{\Sigma_1}$. So the rec. fcn $\sigma(x) = \langle \hat{i}, x\rangle$ shows $L \leq_m U_{\Sigma_1}$.

QED

Theorem $HP = \{i \mid M_i(i) \text{ accepts}\}$ and $L = \{i \mid M_i(i) \text{ halts}\}$ (sometimes called K) are Σ_1-complete.

Each is clearly in Σ_1, and you (try it!) can easily write a $\leq_m$-reduction from U_{Σ_1} to HP, and from U_{Σ_1} to L.

The Arithmetical Hierarchy (a.k.a. Kleene Hierarchy)

$\Sigma_0 = \Pi_0 =$ the recursive sets

$\Sigma_1 =$ the r.e. sets $\quad \Pi_1 =$ the co-r.e. sets

$\Sigma_{i+1} = \{L \mid (\exists \text{TM } M_j)(\exists A \in \Sigma_i)[L = L(M_j^A)]\}$.

$\Pi_{i+1} = \{L \mid \overline{L} \in \Sigma_{i+1}\}$.

Lovely fact: Although defined by machines/oracles, these classes are crisply captured by quantifiers (logic).

Examples

- $\Sigma_1 = \{L \mid$ for some recursive predicate R, $L = \{x \mid \overbrace{(\exists y)}^{\text{E}}[R(x,y)]\}\}$. (Can you see how to prove this?)
- $\Pi_3 = \{L \mid$ for some recursive predicate R, $L = \{x \mid \overbrace{(\forall y)(\exists z)(\forall w)}^{\text{AEA}}[R(x,y,z,w)]\}\}$.
- And so on, e.g. Σ_6 is about "AEAEAE" and Π_8 is about "EAEAEAEA."

Let us sketch the proof of one such case.

Recall: $\Sigma_2 =_{def} \{L \mid (\exists A \in RE)(\exists i)[L = L(M_i^A)]\}$.

Claim $\Sigma_2 = \{L \mid$ for some rec. predicate R, $L = \{x \mid (\exists y)(\forall z)[R(x,y,z)]\}\}$.

Proof Sketch $\supseteq$: Easy. (On x, M_i steps through each $\hat{y}$ in turn, asking the "coRE" query "$(\forall z)[R(x,\hat{y},z)]$". Except instead ask the "RE" query "$(\exists z)[$NOT $R(x,\hat{y},z)]$" (as Σ_2 is allowed an RE oracle, not a coRE oracle) and flip the oracle answer!) oops!! #2

$\subseteq$:

A is RE; say $A = L(M_j)$

$(\exists t)(\forall t' : t' > t)[M_i^{L(M_j)}(x)$, t-simulated, accepts, and $M_i^{L(M_j)}(x)$ has the same behavior whether t-simulated or t'-simulated$]$.

(really (using M_j in our t-sims))

I: t-sim, $M_i \neq M_j$

will do k-sims in more detail later on

I: same oracle queries & answers

1 Actual set: $\{\langle x,y\rangle \mid (\forall z)[R(x,y,z)]\}$.

2 Actual set: $A = \{\langle x,y\rangle \mid (\exists z)[\neg R(x,y,z)]\}$.

QED

All levels of the arithmetical hierarchy have complete sets.
"Universal" sets make this a piece of cake (and then one can reduce from them to prove more and more sets complete).

Universal set for Σ_2 (as an example):

Theorem $U_{\Sigma_2} =_{\text{def.}} \{\langle i,j,x\rangle \mid x \in L(M_i^{L(M_j)})\}$. U_{Σ_2} is Σ_2-complete.

Proof (1) $U_{\Sigma_2} \in \Sigma_2$ is clear. (Why? Hint: U_{Σ_1} might be a good oracle choice.)

(2) $(\forall L \in \Sigma_2)[L \leq_m U_{\Sigma_2}]$: Let $L \in \Sigma_2$. So $(\exists A \in \Sigma_1)(\exists i)[L = L(M_i^A)]$.

Call one such i, "$\hat{i}$." But $A \in \Sigma_1$, so $(\exists j)[A = L(M_j)]$. Call one such j, "$\hat{j}$."

So $x \in L$ iff $\langle \hat{i}, \hat{j}, x\rangle \in U_{\Sigma_2}$.

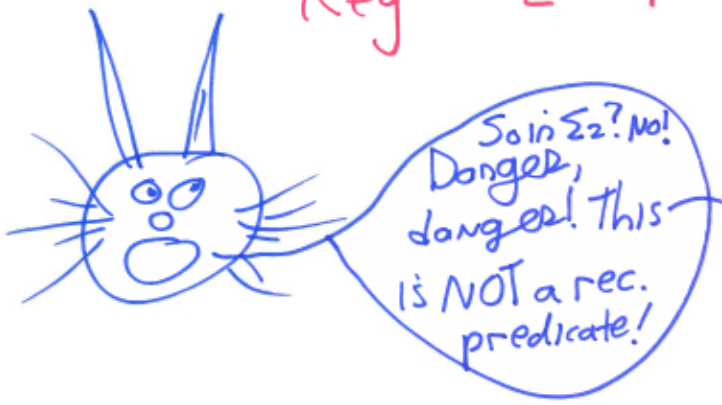
So indeed $L \leq_m U_{\Sigma_2}$.

QED

Claim $\{i \mid L(M_i) \text{ is r.e.}\}$ is Σ^* (or $\mathbb{N}$ if one prefers), which is recursive!

Claim $L_{Reg} =_{def} \{i \mid L(M_i) \text{ is regular}\}$ is in Σ_3.

$$L_{Reg} = \{i \mid (\exists \text{ FA } f)$$
$$(\forall x)$$
$$[x \in L(M_i) \text{ iff } x \in L(f)]\}.$$



$$\exists \forall [(\exists \Rightarrow REC) \wedge (REC \Rightarrow \exists)]$$
$$\exists \forall [(\forall \vee REC) \wedge (REC \vee \exists)]$$

ϕ can write out now as $\exists \forall \exists$.

$L_{inf} = \{ i \mid L(M_i) \text{ is } \infty \}$.

Claim L_{inf} is in Π_2!

Why? Form is $\forall\exists$.

$L_{inf} = \{ i \mid (\forall x)(\exists y, t)[|y| > |x| \wedge M_i(y) \text{ accepts within } t \text{ steps}] \}$.

Claim $L_{finite} =_{def.} \{ i \mid L(M_i) \text{ is finite} \}$ is in Σ_2.

Follows from the above claim, but let us see it directly.

$L_{finite} = \{ i \mid (\exists x)(\forall y)(\forall t)$

$\{ [|y| > |x| \Rightarrow M_i(y) \text{ does not accept within } t \text{ steps}]. \}$

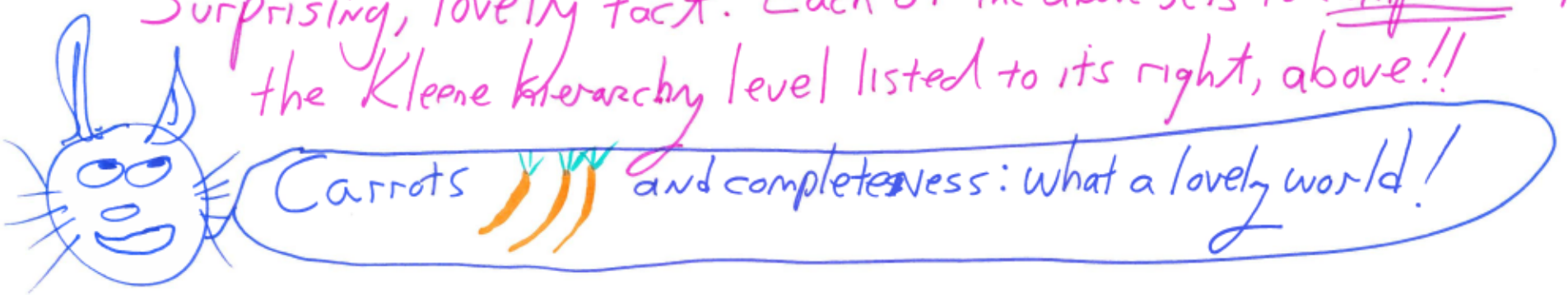
This is a recursive predicate (in effect).

$(\forall y)(\forall t)$
↓ do as, under the hood
$(\forall q)$ and then interpret q as $q = \langle y, t \rangle$.

Kleene Hierarchy
Arithmetical Hierarchy

$\{i \mid L(M_i)$ is	$\neq \phi\}$	$\exists$		Σ_1
	$= \phi\}$	$\forall$		Π_1
	finite$\}$ has	$\exists\forall$	IS	Σ_2
	infinite$\}$ the	$\forall\exists$		Π_2
	$= \Sigma^*\}$ form	$\forall\exists$		Π_2
	regular$\}$	$\exists\forall\exists$		Σ_3
	recursive$\}$	$\exists\forall\exists$	IN	Σ_3
	a CFL$\}$	$\exists\forall\exists$		Σ_3
	cofinite$\}$	$\exists\forall\exists$		Σ_3
	coinfinite$\}$	$\forall\exists\forall$		Π_3
$\{i \mid M_i$ is total$\}$		$\forall\exists$		Π_2

Surprising, lovely fact: Each of the above sets is complete for the Kleene hierarchy level listed to its right, above!!



Let us do an upper bound for a 3rd-level set.

Theorem $L_{cofinite} =_{def} \{ i \mid L(M_i) \text{ is cofinite} \}$ is in Σ_3.

$$L_{cofinite} = \{ i \mid (\exists x)(\forall y : |y| > |x|)(\exists t) [M_i(y) \text{ accepts within } t \text{ steps}] \}.$$

$\exists \forall \exists$

Thus Σ_3.

What about proving completeness (for natural sets — universal sets are easy to show complete) at levels such as $\Pi_2, \Sigma_2, \Pi_3, \Sigma_3, \ldots$?

Doing so is often far more involved than proving Σ_1- and Π_1-completeness results. But, let us prove a Σ_2-completeness Result as an example (but don't feel bad if you don't immediately fully follow it).

Theorem $L_{finite} =_{def} \{ i \mid L(M_i) \text{ is finite} \}$ is $\leq_m$-complete for Σ_2.

Proof sketch

(1) We've already shown $L_{finite} \in \Sigma_2$. Good!

(2) As to showing that $(\forall L \in \Sigma_2)[\, L \leq_m L_{finite} \,]$....

Well, we already proved that

$$U_{\Sigma_2} =_{def} \{ \langle i, j, x \rangle \mid x \in L(M_i^{L(M_j)}) \}$$

is Σ_2-complete.

So *if* we prove $U_{\Sigma_2} \leq_m L_{finite}$, we are done, by the transitivity of $\leq_m$ (if $L \in \Sigma_2$ then $(L \leq_m U_{\Sigma_2} \wedge U_{\Sigma_2} \leq L_{finite})$; so $L \leq L_{finite}$). (This is the std. tool/trick to obtain completeness results.)

So, "all" (!!!!) we have left is to show $U_{\Sigma_2} \leq_m L_{finite}$.

We'll do this in a following Proposition, so: QED (but must prove the proposition!)

Proposition $U_{\Sigma_2} \leq_m L_{finite}$.

Proof First, we need some groundwork.

Define a k-simulation of $M_q(x)$ to be the following:

Run $M_q^{[k]}(x)$. Do what M_q does... and if it is still running after k steps, pretend it rejects.

Define a k-simulation of $M_q^{M_r}(x)$ as the following:

k-simulate M_q & k-simulate all oracle calls.

M_r ← $L(M_r)$ is the oracle, but our k-sim. works w.r.t. actual machines.

End of Groundwork!

Let us build/describe our recursive reduction from U_{Σ_2} to L_{finite}.

So, we're given as input (to the reduction) $\langle i, j, x \rangle$.

Consider the index $\sigma(\langle i,j,x\rangle)$ of the TM that on input y, $1y_{\text{binary}}$-simulates $M_i^{M_j}(x)$ and that accepts iff the simulation (i) rejects or (ii) accepts in a new way (i.e., the sequence of queries and/or their answers changed (relative to a $(1y_{\text{binary}}-1)$-simulation).

Claim σ: $\langle i,j,x\rangle \to \sigma\langle i,j,x\rangle$ many-one reduces U_{Σ_2} to L_{finite}.

Case 1: $z=\langle i,j,x\rangle \in U_{\Sigma_2}$. Let $t = 1 + \max$ (the number of steps M_i takes to accept in the run of $M_i^{L(M_j)}(x)$, the maximum number of steps taken by M_j to accept on any of the accepting oracle calls made during the run of $M_i^{L(M_j)}(x)$).

Crucially, note that for each string w such that $1w_{\text{binary}} \geq t$, $M_{\sigma(\langle i,j,x\rangle)}(w)$ rejects. Thus $L(M_{\sigma(\langle i,j,x\rangle)})$ is finite. So $\sigma(\langle i,j,x\rangle) \in L_{\text{finite}}$.

Case 2: $z=\langle i,j,x\rangle \notin U_{\Sigma_2}$.

Case 2.1 Consider $0,1,2,\ldots$ simulations of $M_i^{M_j}(x)$. Suppose ∞ # of these reject. Then by (i) of the def. of σ, clearly $L(M_{\sigma(\langle i,j,x\rangle)})$ is ∞. So $\sigma(\langle i,j,x\rangle) \notin L_{\text{finite}}$

Case 2.2 a.e. k-simulation ($k=0,1,2,\ldots$) accepts.

Case 2.2.1 Suppose a.e. accepts in the same way. Paradoxical, since that implies $z \in U_{\Sigma_2}$.

Case 2.2.2 Accepts i.o. in new ways. Thus by (ii) of def. of σ, $L(M_{\sigma(\langle i,j,x\rangle)})$ is ∞. Thus $\sigma(\langle i,j,x\rangle) \notin L_{\text{finite}}$.

QED (of Claim, and Proposition, and so our Σ_2-completeness theorem is established)

that was tough!! But I now have the hang of things, and think that, with some effort, I can avoid being fooled by this Kleene Hierarchy — I can prove upper bounds, & with hardwork, matching lower bounds, as we just did for L finite.

Really? Then riddle me this & What is $\hat{L} = \{\langle i,j\rangle \mid L(M_i^{L(M_j)}) \text{ is } \Sigma_2\text{-complete}\}$ complete for?

Nice try, sneaky carrot. But I'm on top of sussing out quantifiers! And this set is really about:

$\{\langle i,j\rangle \mid (\forall L \in \Sigma_2, \text{i.e., } \forall \langle \hat{i},\hat{j}\rangle \text{ as speaking of } L(M_{\hat{i}}^{L(M_{\hat{j}})}))(\exists \text{ fcn. } f)$
$[f \text{ is total} \wedge (\forall x)[x \in L(M_{\hat{i}}^{L(M_{\hat{j}})}) \Leftrightarrow f(x) \in L(M_i^{L(M_j)})]]\}$.

So: $\forall\exists[(\forall\exists) \wedge \forall[(\exists\forall) \Leftrightarrow (\exists\forall)]]$. Which unwinds to $\forall\exists\forall\exists\forall$, so Π_5!

Well, Π_5 is indeed the natural upper bound. But you'd <u>never</u> be able to prove a matching lower bound — i.e., prove $U_{\Pi_5} \leq_m \hat{L}$. Why? This set, by a clever trick, is even in Σ_4! Namely: In your Π_5 upper bound, you used your leading $\forall$ to quantify over all $L \in \Sigma_2$, as what you needed to be able to reduce to $L(M_i^{L(M_j)})$. But note that all you need to show is that $U_{\Sigma_2} \leq_m L(M_i^{L(M_j)})$ and you're done, as U_{Σ_2} is Σ_2-comp., & $\leq_m$ is transitive. So: Σ_4

CRUNCH! (Such cases where the natural upper bound is misleading are rare, and I was hungry! Of course, I still have not proven a Σ_4 lower bound: oh come back, wise carrot!!)

Consider $L = \{i \mid L(M_i) \text{ is cofinite}\}$.

Goal Show that L is not RE.

We now have a new approach available (though it is no picnic!).

Method 1 (new) (hard, and overkill) Prove that L is Σ_3-complete, and then use the fact that $\Sigma_0 \subsetneq \Sigma_1 \subsetneq \Sigma_2 \subsetneq \Sigma_3 \subsetneq \cdots$.

Method 2 Use Rice II.

Method 3 Direct (sort of). Prove $\overline{HP} \leq_m L$. And we sooo know how to build such reductions. For example, this is such a reduction:

$\sigma(x)$: Output a machine with the behavior: On input y, run $M_x(x)$ for $1y$ binary steps. If $M_x(x)$ accepts within that number of steps then reject and if "does not accept" "number of steps then accept.

So σ is rec.; $x \in \overline{HP}$ gives $L(M_{\sigma(x)}) = \Sigma^*$ which is cofinite; and $x \in HP$ gives "$L(M_{\sigma(x)})$ is finite" and so $L(M_{\sigma(x)})$ is not cofinite. So $\overline{HP} \leq_m L$ via σ.

Let us do more examples of showing upper bounds in the Kleene Hierarchy!

$\{i \mid M_i(i) \text{ accepts within } i \text{ steps}\} \in \Sigma_0$

$\{i \mid M_i(i) \text{ accepts (i.e. ever having visited) within } 2^{2^{2^i}} \text{ tape cells}\} \in \Sigma_0$

$\{i \mid M_i(i) \text{ never moves its head left}\} \in \Sigma_0$ (Runaway train!?)

$\{i \mid L(M_i) \subseteq 0(0+1)^*\}$

$(\forall x)[x \in L(M_i) \Rightarrow x \text{ starts with } 0]$

$(\forall x)[\exists \Rightarrow \cdot]$

$(\forall)[\overline{\exists} \vee \cdot]$

$\forall \quad \forall$

$\forall$

Π_1.

$$\{ \langle i,j \rangle \mid L(M_i) \neq L(M_j) \}.$$

$$(\exists x)\left[(x \in L(M_i) - L(M_j)) \vee (x \in L(M_j) - L(M_i)) \right]$$

$$(\exists)\left[(\exists \wedge \forall) \quad \vee (\exists \quad \wedge \forall) \right]$$

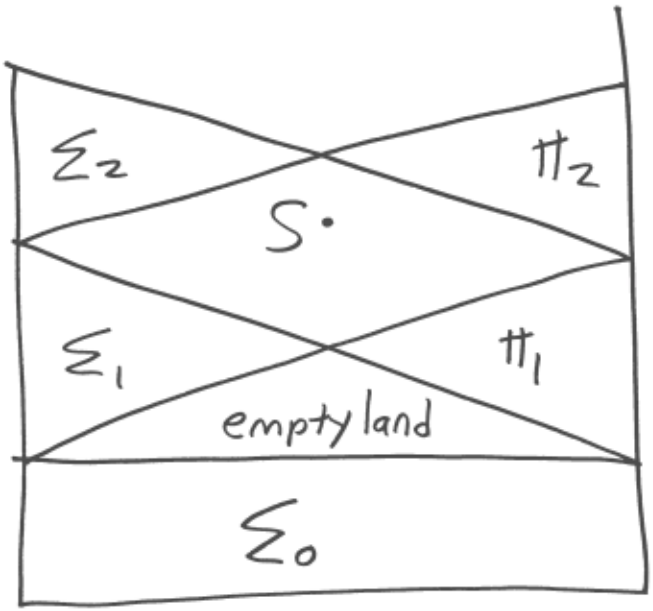
So $\exists\forall$

So Σ_2.

$$S = \{1i \mid i \in L(M_i)\} \cup \{0i \mid i \notin L(M_i)\}.$$

"join"
"disjoint union"
"marked union"

$$S \in (\Sigma_2 \cap \Pi_2) - (\Sigma_1 \cup \Pi_1)$$



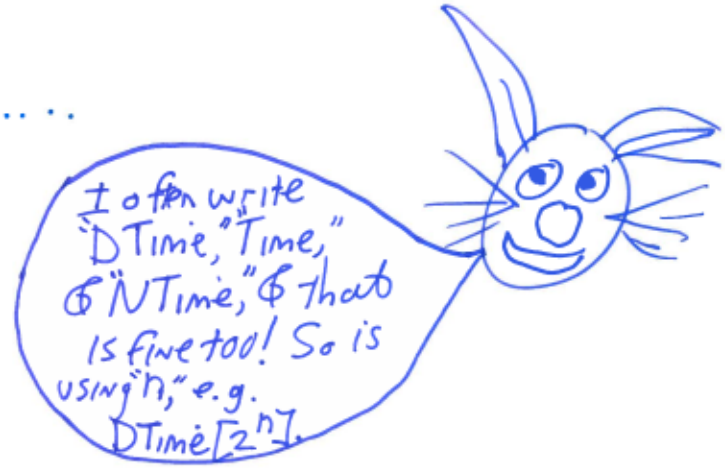
Nerdy note: $\Sigma_2 \cap \Pi_2 = \{L \mid L \leq_T U_{\Sigma_1}\}$, basically by relativizing our RE ∩ coRE = REC proof, with any Σ_1-complete oracle.

COMPUTATIONAL COMPLEXITY!

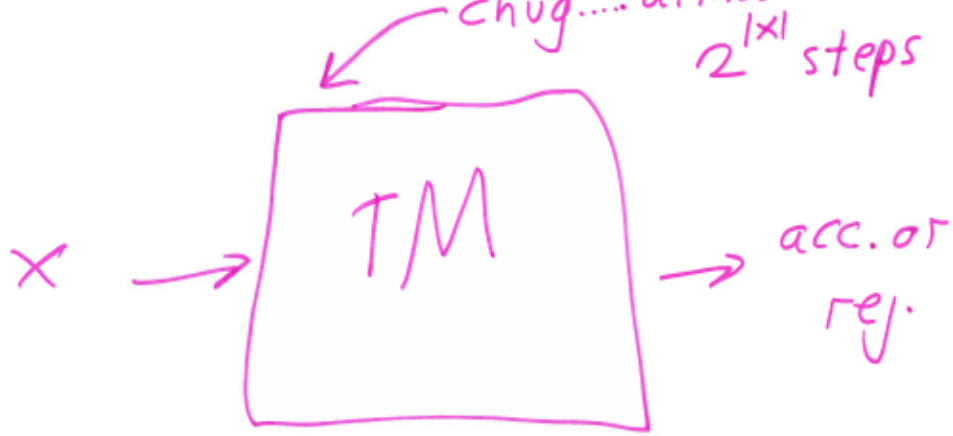
Informal Def.

$\begin{matrix} DTIME \\ TIME \end{matrix}[f(N)]$ is the class of languages accepted by a deterministic TM that always runs within $f(N)$ steps on inputs of length N.

$NTIME[f(N)]$ .... nondeterministic ....



Example $Time[2^N]$



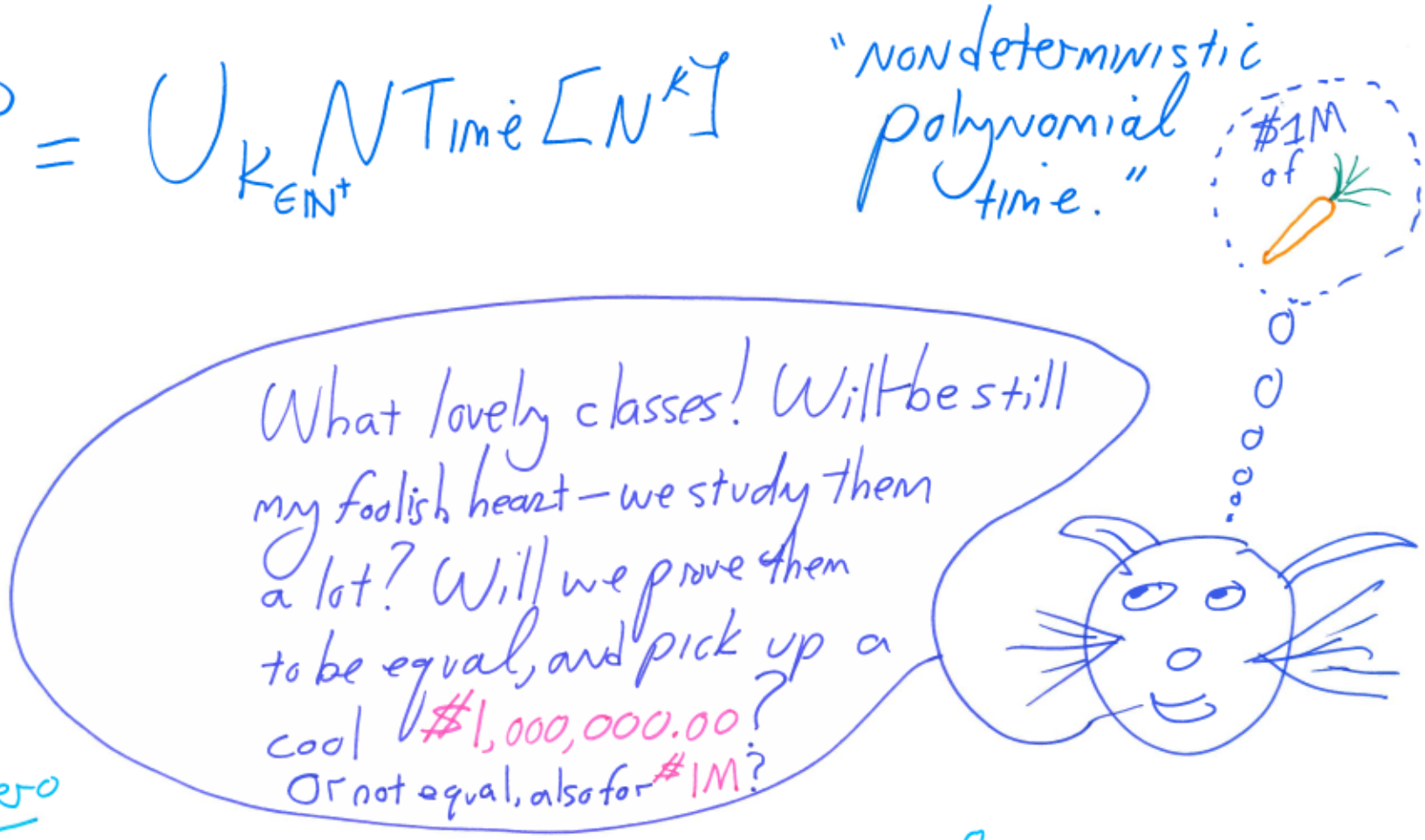
Claim $Time[2^N] \subseteq \Sigma_0 =$ the recursive sets.

Needy notes: (1) For time, $f(N)$ is shorthand for $\max(N+1, \lceil f(N) \rceil)$.
(2) " space, " " " " " $\max(1, \lceil f(N) \rceil)$.

$$P = \bigcup_{k \in \mathbb{N}^+} \text{Time}[N^k]$$

$$NP = \bigcup_{k \in \mathbb{N}^+} \text{NTime}[N^k]$$

"nondeterministic polynomial time."



Fun-with-Zero

Nerdy notes: (1) $\mathbb{N}^+$ rather than $\mathbb{N}$ to avoid 0^0.
(2) For $N=0$, recall last page's Nerdy Notes....
or...

P = "polynomial time" $= \bigcup_{i \geq 1} \text{Time}[N^i + i]$.

NP = "nondeterministic polynomial time" $= \bigcup_{i \geq 1} \text{NTime}[N^i + i]$.

Note: This alternate definition works even without the "$= f(N)$ means $\max(N+1, \lceil f(N) \rceil)$" assumed tweak.

- By convention, we say that problems in P are tractable.

P: cat, mouse, piano

- Gar-Joh's book (6 decades of study) show that many useful problems are in NP.

GOAL $P = NP$. ← This is the holy grail of theoretical computer science!

P:

Claim $P \subseteq NP$.

Pf. Immed. □

What's to come?

- parallels between NP (PH) and Σ_1 (Kleene hierarchy).
- Reductions, & complete sets for NP.
- the polynomial hierarchy
- Cook's (C-Karp-Levin) Theorem
- NP ∩ coNP and the Borodin-Demers Theorem
- Lots of looking into the nature/structure of NP; for example, can NP-complete sets:
  - be sparse?
  - have "small circuits" (equiv., be in P^{SPARSE})?
  - have semi-feasible algorithms (be in P-sel)?
  - ....

Why/how is NP like Σ_1?

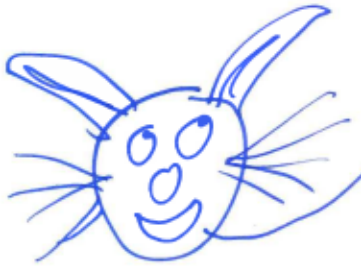
Σ_1 : $\exists$ Rec. predicate

NP : $\exists_p$ poly-time predicate

$\updownarrow$

Σ_1^p $\quad NP = \{L \mid (\exists k \in \mathbb{N}^+)(\exists$ polynomial-time computable predicate $R)$

$[\, L = \{x \mid (\exists y)[|y| \leq |x|^k + k \wedge R(x,y)]\}\}$.



Secret siblings!? This is like the plot twists on a soap opera!
Is there more in this family's closet? Yes....

Σ_3^p $\quad \exists_p \forall_p \exists_p$ poly-time predicate

The Polynomial Hierarchy

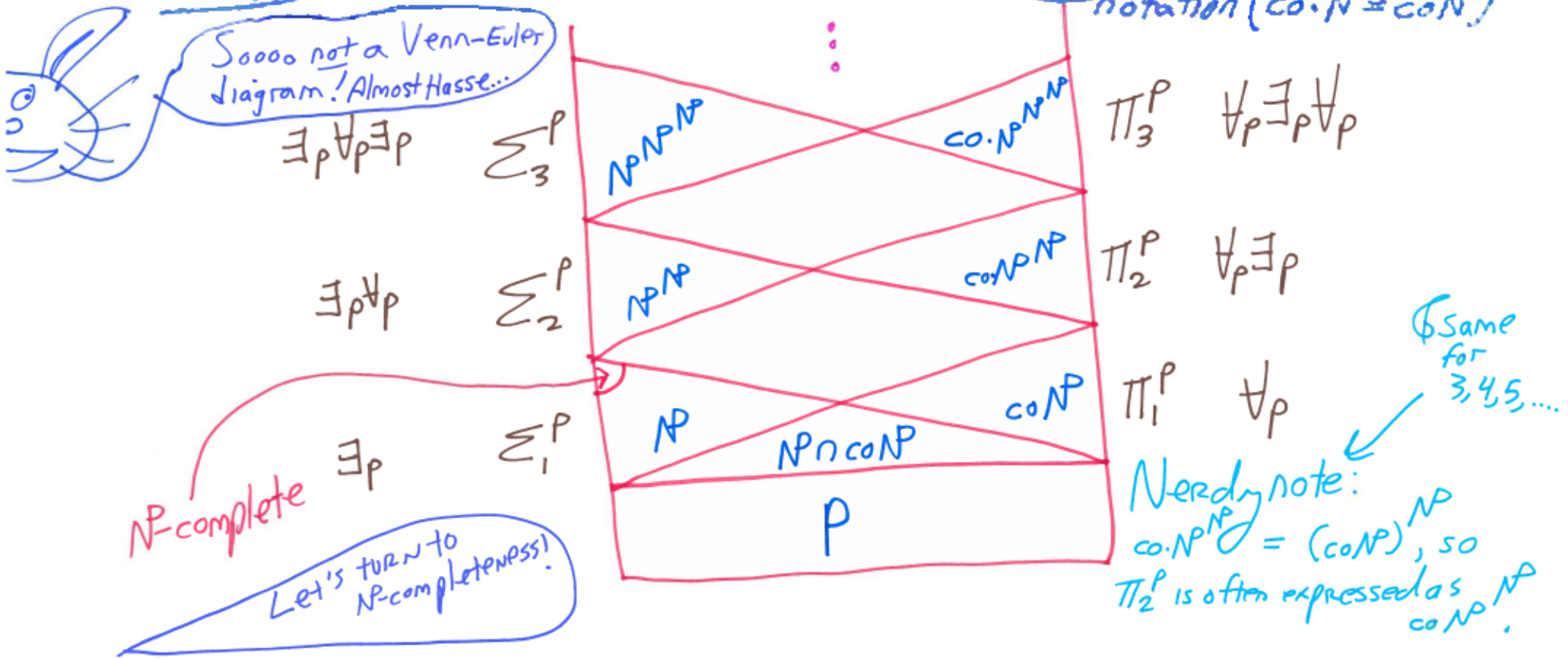
$\Sigma_0^P = \Pi_0^P = \Delta_0^P = \Delta_1^P = P.$

$\Sigma_{i+1}^P = NP^{\Sigma_i^P}, \quad i \geq 0.$ E.g., $\Sigma_1^P = NP^P = NP$. So $\Sigma_1^P = NP$.

$\Pi_{i+1}^P = co \cdot \Sigma_{i+1}^P =_{\text{def. (of the "co" operator)}} \{L \mid \bar{L} \in \Sigma_{i+1}^P\}, \quad i \geq 0.$

Examples: $\Sigma_2^P = NP^{NP}$. $\Pi_1^P = \{L \mid \bar{L} \in NP\} = coNP$.

notation ($co \cdot NP = coNP$)

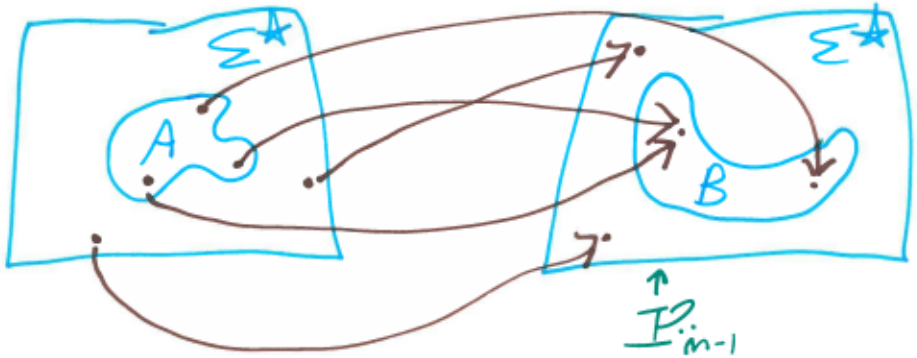


Def. L_1 is **NP-complete** exactly if: ① $L_1 \in NP$ and ② $(\forall \hat{L} \in NP)[\hat{L} \leq_m^p L_1]$.

Nerdy note: ② defines "NP-hard." I?: was "T" in old days

Def. (many-one polynomial-time reduction's) $A \leq_m^p B$ iff there exists a polynomial-time computable function f such that $(\forall x \in \Sigma^*)[x \in A \Leftrightarrow f(x) \in B]$.

Picture:



Claim Let $N_1, N_2, N_3, \ldots$ be std. enumeration of NPTMs. Wlog., let N_i run within time $n^i + i$ on inputs of length n. Then

$U = \{ N_i \# x \# 1^k \mid N_i(x) \text{ accepts within } k \text{ steps} \}$ is NP-complete.

I?:

Why? Let $L \in NP$. Say $L = L(N_j)$.

Consider: $x \in L(N_j)$? $\xrightarrow[\text{reduction from } x]{\text{poly}}$ $N_j \# x \# 1^{|x|^j + j} \in U$? So $L \leq_m^p U$.

Does $L \leq_m^p U$ hold even without the 1^k being in U's definition??

I?: 88

$SAT = \{ f \mid f \text{ is a satisfiable Boolean formula} \}$.

"true" — $T \in SAT$

"false" — $F \notin SAT$

$x_1 \in SAT$

$x_1 \wedge \bar{x}_1 \notin SAT$

$x_1 \wedge x_2 \in SAT$

etc.

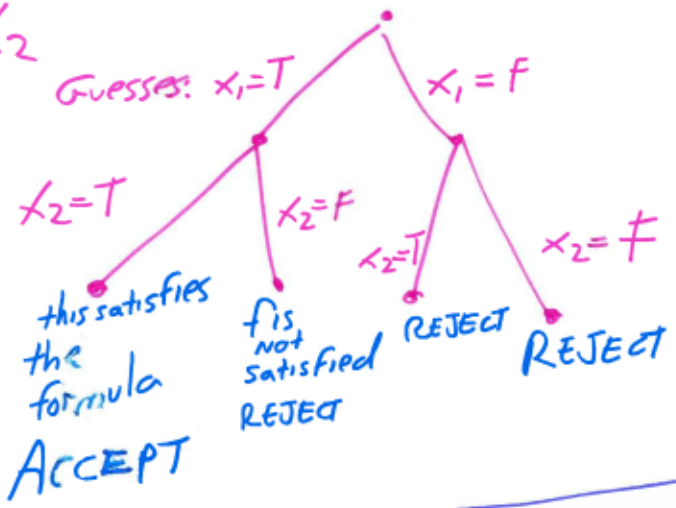
looks easy....

.... but it is <u>not</u> (we believe — it is a $1M question!)

Cook's Theorem (a.k.a. the Cook-Karp-Levin Theorem) SAT is NP-complete.

Claim $SAT \in NP$

$f = x_1 \wedge x_2$



I remember how acceptance of NPTMs is defined: If any path accepts on the given input, the machine is said to have accepted that input. So this is an ACCEPT!

Cook's Theorem

Pf 1 Hop-Ull

Pf 2 Gar-Joh

Proof's core insight/achievement is:

Given N_i and x (N_i our std. enum of NPTMs, so wlog N_i runs in time $n^i + i$),

we can construct a formula $f(i, x)$ such that

$$x \in L(N_i) \Leftrightarrow f(i, x) \text{ is satisfiable;}$$

and we can perform the formula construction in time polynomial in $|N_i| + |x|^i$.

Galil (-Cook) observed that one can ensure that $f(i, x)$ displays N_i and x in an obvious fashion. We'll discuss (and use!) Galil's insight soon.... but before we do that, let's spend a good bit of time discussing NP-complete sets, and how to prove sets to be NP-complete.

Some NP-Complete Sets

Clique

$\{(G, k) \mid$ Graph G has a clique of size $k\}$

← 4-clique

← 3-clique

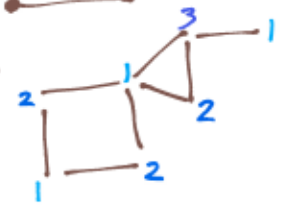
Graph Coloring:

$\{(G,k) \mid G$ is k-colorable$\}$

I.e., there is a way of associating each vertex with the colors $1, 2, \ldots, k$, such that no edge has both its ends touching nodes of the same color.

Example:

is 3-colorable



but is not 2-colorable.

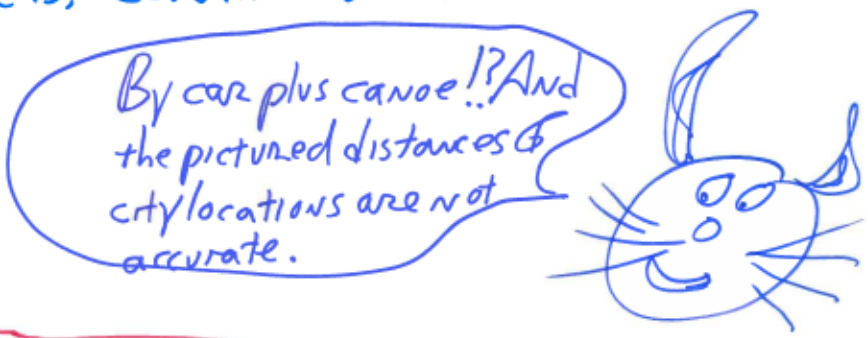
is 2 colorable

but is not 1-colorable

Nerdy note: Testing 2-colorability (equivalently, bipartiteness, equiv., having no odd length cycle) is in P. But even 3-colorability testing is already NP-complete.

Some NP-Complete Sets, continued

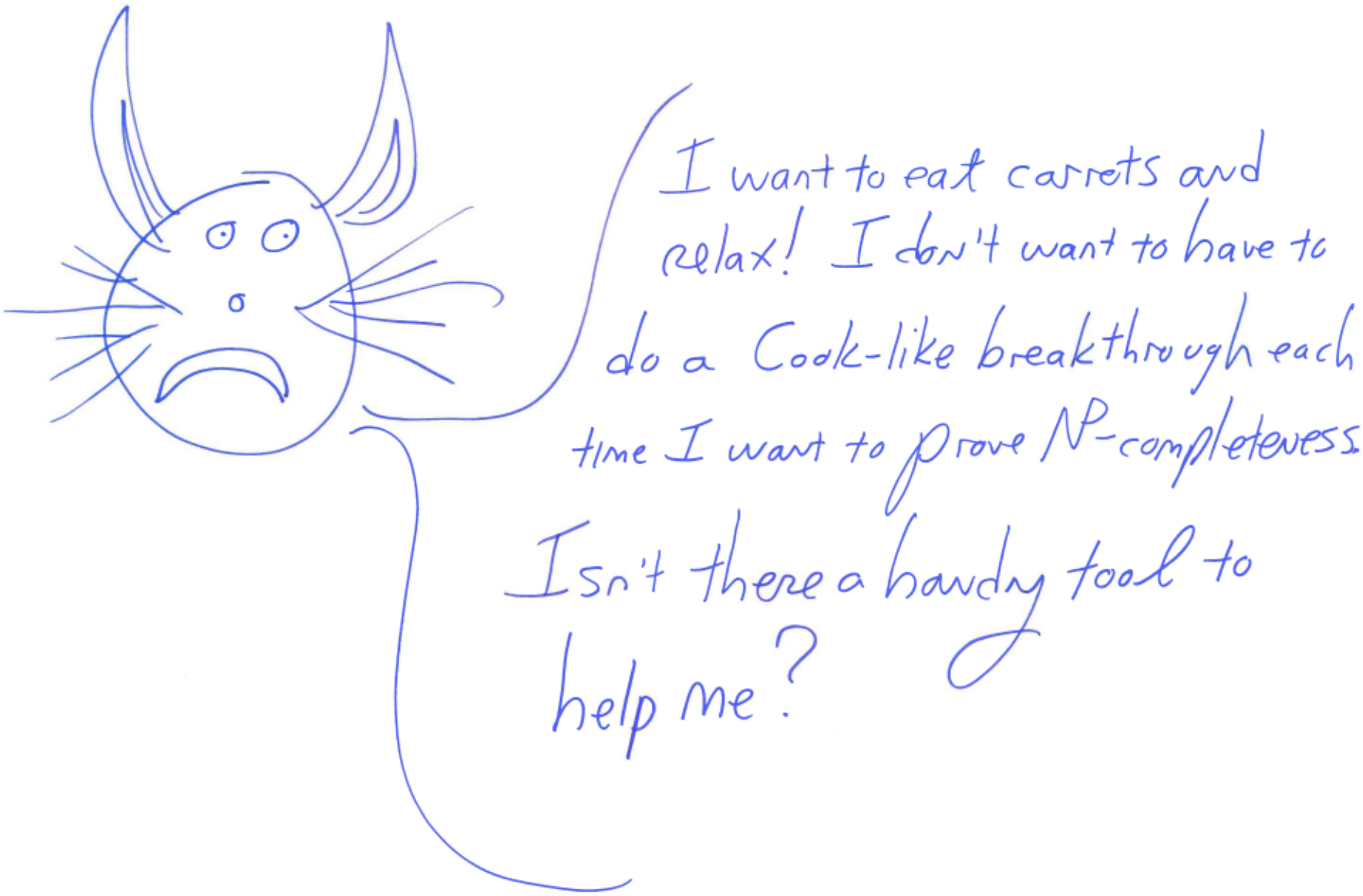
The Traveling Salesperson Problem



$$\left\{ \left(\begin{array}{c}\text{directed pairwise distance table for a collection of cities} \\ \uparrow \\ \text{"map", informally}\end{array}, k \right) \;\middle|\; \text{There is a tour of length} \leq k \right\}.$$

Tour: (each city appears 1 time here) City 4 → City 7 → ⋯ → City 3

Nerdy note: The "symmetric" TSP (dist(a,b) = dist(b,a)) remains NPC; thanks goes to Hamiltonian cycle!


I want to eat carrots and relax! I don't want to have to do a Cook-like breakthrough each time I want to prove NP-completeness.
Isn't there a handy tool to help me?

Key Tool If

(1) L is NP-complete, and

(2) $L' \in NP$, and

(3) $L \leq_m^P L'$,

then L' is NP-complete.

Proof $L' \in NP$ by (1).

We must show that every NP problem reduces to L'.

Let $\hat{L} \in NP$. $\hat{L} \leq_m^P L$, since L is NP-complete.

So $\hat{L} \leq_m^P L \leq_m^P L'$. Thus $\hat{L} \leq_m^P L'$, by the p-time reduction function $g(f(\cdot))$. □

say by f, p-time fcn.

say by g, p-time fcn.

Nerdy note: Yes, that was an on-the-fly proof that $\leq_m^P$ is transitive... embedded in our proof of the Tool.

Taking Stock (Plus some facts)

Recall:

- $SAT = \{f \mid f \text{ is a satisfiable Boolean formula}\}$ is NP-complete (w.r.t. $\leq_m^P$ reductions).
- KEY TOOL: (L is NP-complete AND $L' \in NP$ AND $L \leq_m^P L'$) $\Rightarrow L'$ is NP-complete (aka NPC).

Fact (comes for free from Cook's Theorem's proof):

$CNF_{SAT} = \{f \mid f \in SAT \wedge f \text{ is in conjunctive normal form}\}$ is NPC.

conjunctive normal form: $\underbrace{(x_1 \vee x_{17} \vee \overline{x_{27}})}_{\text{clause}} \wedge \underbrace{(x_{17} \vee \overline{x_{92}} \vee \overline{x_1} \vee x_7)}_{\text{clause}} \wedge \ldots$

Fact $3CNF_{SAT}$ is NPC. ($3CNF_{SAT}$ is the same as CNF_{SAT} except each clause can have at most three literal occurrences.)

$(x_1 \vee \overline{x_4} \vee \overline{x_{17}}) \wedge (\overline{x_2} \vee \overline{x_9} \vee \overline{x_{11}}) \wedge (x_2 \vee x_5 \vee x_8) \wedge \ldots$

Nerdy note: x_1 is a literal. So is $\overline{x_1}$. } holds for each variable x_1.

$L_{log} = \{f \mid f$ is in 3CNF-form $\wedge$ $f \in SAT \wedge$ the number of variables that appear complemented in f is less than or equal to $\log_2(|f|)\}$.

$L_{\sqrt{\ }} = \{f \mid \ldots \ldots \ldots \ldots \sqrt{|f|}\}$.

Question: How hard are L_{log} and $L_{\sqrt{\ }}$?

Claim $L_{log} \in P$.

Why? $(x_1 \vee x_{17} \vee x_{27}) \wedge (x_3 \vee x_5 \vee \overline{x_{11}}) \wedge \ldots$

Palgorithm: (1) Check for 3CNF-form. Check that # complemented vars. $\leq \log_2(|f|)$. Reject if either is violated.

(2) Instantiate the $\leq \log_2(|f|)$ complemented variables every possible way.

ways is $\leq 2^{\log_2(|f|)} = |f|$

$\rightarrow x_1=T \; x_2=T \; \ldots \; x_{\text{"}\leq \log(|f|)\text{"}}=T$

$\rightarrow x_1=T \; x_2=T \; \ldots \; x_{\text{"}\leq \ldots\text{"}}=F$

$\rightarrow \vdots$

$\rightarrow x_1=F \; x_2=F \; \ldots \; = F$

If any way satisfies f then accept, else reject.

□

Claim $L_{\ulcorner\urcorner}$ is NPC.

Part 1 $L_{\ulcorner\urcorner}$ is in NP. Pf:

Part 2 $3CNF_{SAT} \leq_m^p L_{\ulcorner\urcorner}$. We'll build a $\leq_m^p$-red. f so that

$(\forall x)[x \in 3CNF_{SAT} \Leftrightarrow f(x) \in L_{\ulcorner\urcorner}]$.

$$q \longrightarrow \begin{cases} F & \text{if } q \text{ is not in 3CNF-form.} \\ q \wedge new_1 \wedge new_2 \wedge \cdots \wedge new_{|q|^{486}} & \text{otherwise.} \end{cases}$$

(wlog new_i variables are not used in q)

So $q \in 3CNF_{SAT} \Leftrightarrow f(q) \in L_{\ulcorner\urcorner}$. (Do you see why?)

□

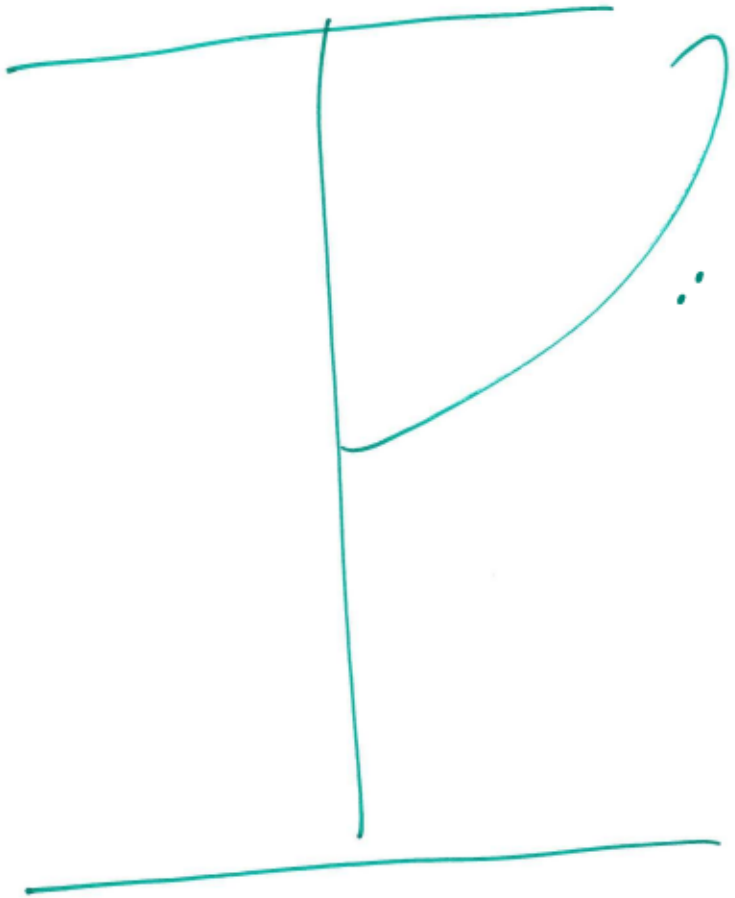
Thm $L_{=} = \{f \mid$ (each variable appearing in f appears the same # of times) $\wedge f \in SAT\}$ is NPC.

[Count x_i and $\bar{x}_i$ as same variable for the above purpose.]

E.g., $x_1 \vee \bar{x}_1 \vee x_2 \vee x_2 \in L_{=}$.

$x_1 \vee x_2 \vee x_3 \vee \bar{x}_3 \notin L_{=}$.

Ex: → Can you see how to prove this Thm?



Recall:

Cook [P.]

(N_i, x) ["Does $N_i(x)$ accept?"]

$\downarrow$ whooosh... fast!

$F_{i,x}$

(N_i w/log runs within $|x|^i + i$ steps)

① $N_i(x)$ accepts $\iff$ $F_{i,x}$ is satisfiable.

② We can produce $F_{i,x}$, given N_i and x, in time polynomial in $|N_i| + |x|^i + i$.

We have $N_{i,x} \xrightarrow{\text{fast}} F_{i,x}$.

Let us seek to achieve (thank you, Zvi Galil!)

$N_{i,x} \xrightarrow{\text{fast}} \hat{F}_{i,x}$.

("I come from N_i and x") $\wedge$ $(F_{i,x})$

Let $z = \langle N_i, x \rangle$.

So say z as a binary string is $b_1 b_2 \cdots b_k$.

Let r be a variable that does not occur in $F_{i,x}$.

$$\hat{F}_{i,x} = \underbrace{(r_{b_1} \vee r_{b_2} \vee \cdots \vee r_{b_k} \vee T)}_{\text{this}} \wedge (F_{i,x}), \text{ where}$$

by/in this, we take/write: r_{b_i} as r if $b_i = 1$

r_{b_i} as $\bar{r}$ if $b_i = 0$.

E.g. If $z = 1001$, then $\hat{F}_{i,x}$ will be of the form:

$$(z \vee \bar{z} \vee \bar{z} \vee z \vee z \vee T) \wedge (F_{i,x}).$$

This has all the magic of Cook's reduction.... plus, the output formula openly reveals the machine and input that it is about.

Cool.... but can we put this to work to see something surprising?

POLL:

Likely T or F: $(\exists A)[A \in P \wedge A \subseteq SAT \wedge$ "no poly-time machine can find solutions to all of A"$]$.

(H_1:)

i.e. $(\forall f \in FP)(\exists q \in A)[f(q)$ is NOT a satisfying assignment of $q]$.

"obviously satisfiable but not obvious how they are satisfied"

Likely T or F: (H_2:) $NP \cap coNP \neq P$

Which is "more likely" to hold?

informal (sticky)

Thm. (the Borodin-Demers Theorem) $NP \cap coNP \neq P$

$\Downarrow$

$(\exists A)[A \in P \wedge A \subseteq SAT \wedge$ "no poly-time machine can find solutions to all of A" $]$.

i.e. $(\forall$ poly-time comp. fcn $f)$
$(\exists q \in A)[$ $f(q)$ is not a satisfying assignment of q $]$.

Note: $(\exists A)[A^{\infty} \wedge A \in P \wedge A \subseteq SAT]$ hold unconditionally,
($\|A\| = \infty$)
e.g., $A = \{T, TVT, TVTVT, \ldots\}$.

It is just due to also having the 3rd Conjunct that this theorem has (very) sharp teeth.

Proof of the Borodin-Demers Theorem:

We assumed $NP \cap coNP \neq P$. So, let $L \in (NP \cap coNP) - P$.

Let N_k be an NPTM (from the std. enum.) accepting L.

Let $N_{\hat{k}}$ " " " " " " accepting $\overline{L}$.

Set: $S = \{ f \mid (\exists x)[f \text{ is } (\hat{F}_{k,x}) \vee (\hat{F}_{\hat{k},x})] \}$.

This completes the construction!!—it took just a few lines!!!

Claim $S \in P$.

Claim $S \subseteq SAT$.

Pf. $(\forall x)[x \in L \vee x \notin L]$.

So, $(\forall x)[(\hat{F}_{k,x}) \vee (\hat{F}_{\hat{k},x})$ is satisfiable$]$.

Claim If you could efficiently (Ptime) find satisfying assignments for S, then you could determine membership in L in poly time. (Why?)

(But that i.e., α yields a contradiction, as $L \notin P$ was assumed.)

Q ∈ D