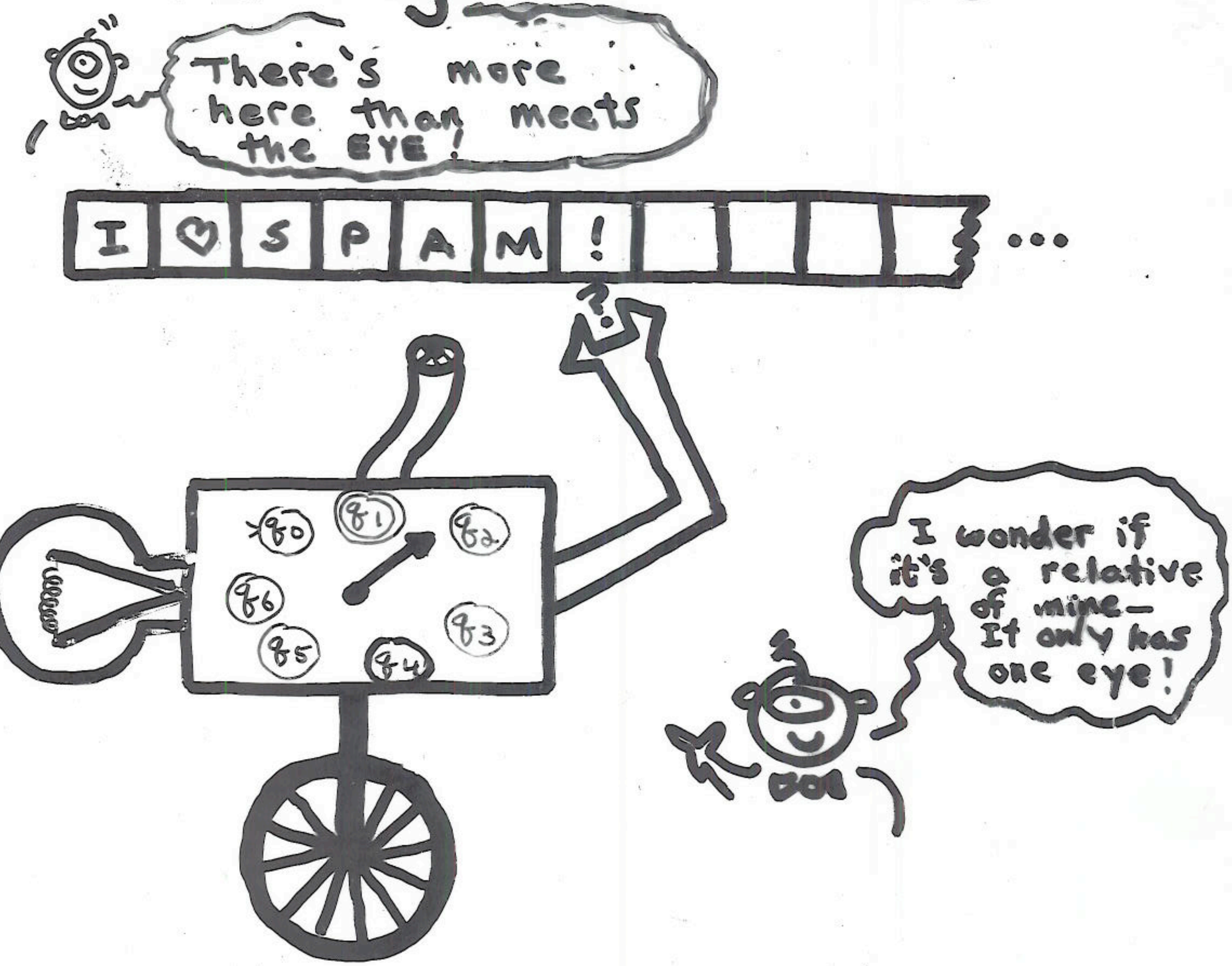
Turing Machines



Features:

1. Read and write on the infinite tape.

2. Read/write head can move left and right.

3. Unique accept and reject states take immediate effect.
4. Deterministic.

Formal Definition

A Turing Machine is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

where Q, Σ, Γ are finite sets and:

1. Q is the set of states.
2. Σ is the input alphabet which may not contain the "$\sqcup$" blank symbol.
3. Γ is the tape alphabet. $\Sigma \subseteq \Gamma$, $\sqcup \in \Gamma$.
4. $\delta : Q \times \Gamma \longrightarrow \overbrace{Q \times \Gamma \times \{L, R\}}$ ← deterministic!
   is the transition function.
5. $q_0 \in Q$ is the start state.
6. $q_{accept} \in Q$ is the accept state.
7. $q_{reject} \in Q$ is the reject state.

Computing with a TM

- The TM receives its input $w \in \Sigma^*$ on the tape beginning at the left end of the tape. Rest of tape blank

- The tape head begins on the left end of the tape.

- Computation proceeds according to δ-function.

- Attempt to go off left edge of tape keeps head at left edge of tape. (ok Sipser, we'll humor you!)

- On q_{accept}, M halts and accepts

- On q_{reject}, M halts and rejects.

- M may run forever!

A Very Important Distinction ...

Definition: A language L is said to be recursively enumerable (r.e.) if there exists a TM that  enters q_{accept} $\forall w \in L$ and either enters q_{reject} or runs forever $\forall w \notin L$.

aka "enumerable" or "semi-decidable"

Definition: A language L is said to be recursive if there exists a TM that enters q_{accept} $\forall w \in L$ and enters q_{reject} $\forall w \notin L$.

aka "decidable"

Question: Are the recursively enumerable languages equal to the recursive languages?

$$L = \{ a^i b^i c^i \mid i \geq 1 \}$$

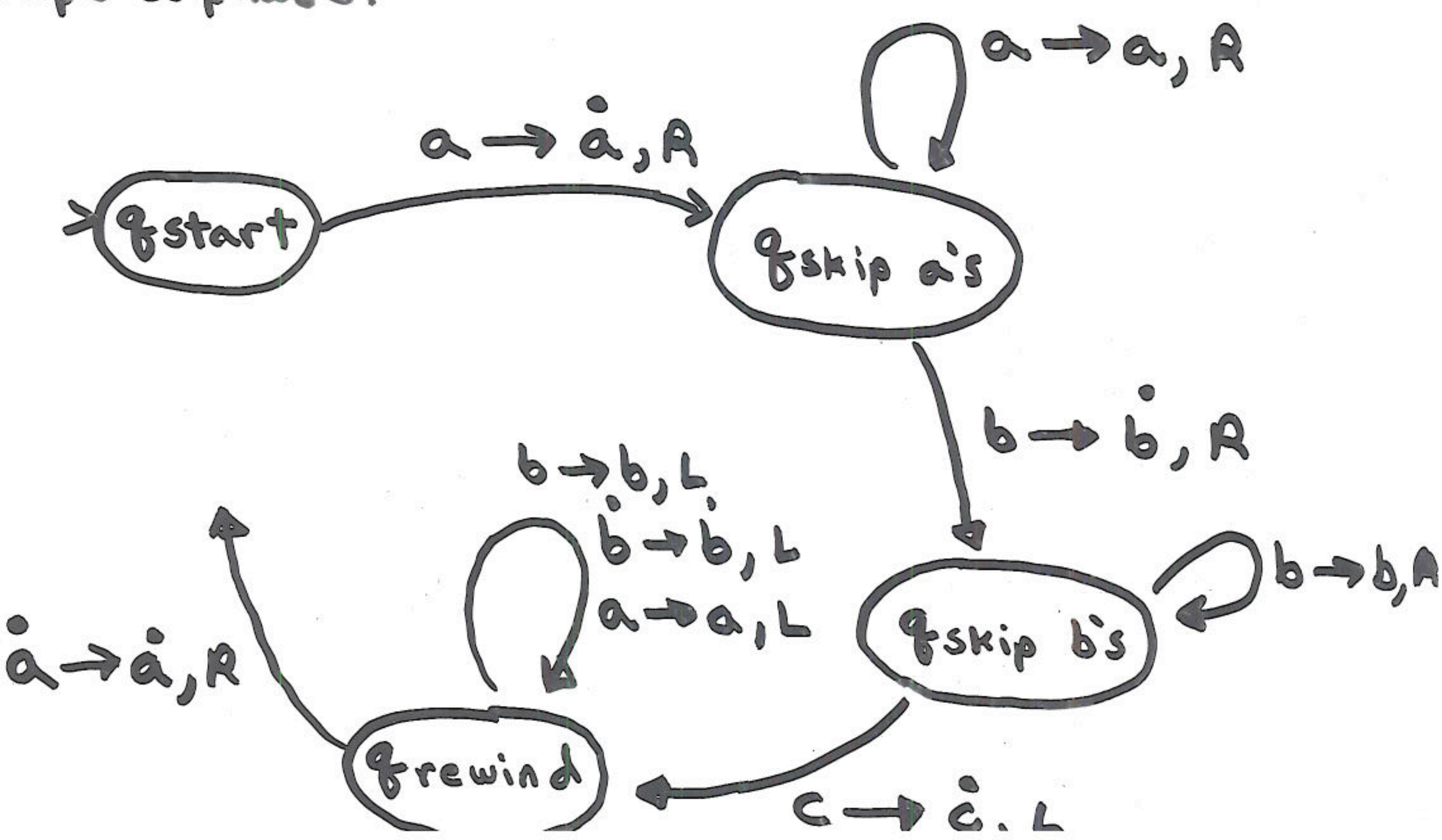
is recursive!

$$\Sigma = \{ a, b, c \}$$

input alphabet

$$\Gamma = \{ a, b, c, \sqcup, \dot{a}, \dot{b}, \dot{c} \}$$

tape alphabet



How About...

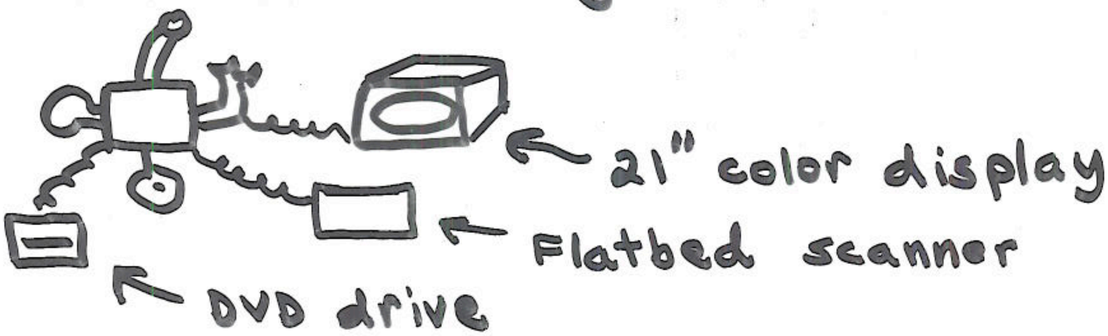
$L_1 = \{ 0^p \mid p \text{ is prime} \}$

$L_2 = \{ b_i \# b_{i+1} \mid b_j \text{ is the binary representation of the number } j \}$

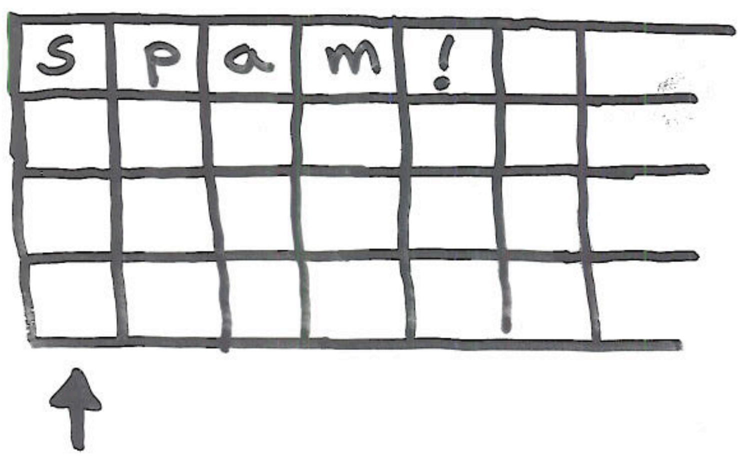
$L_3 = \{ b_i \# b_j \# b_{i \cdot j} \}$

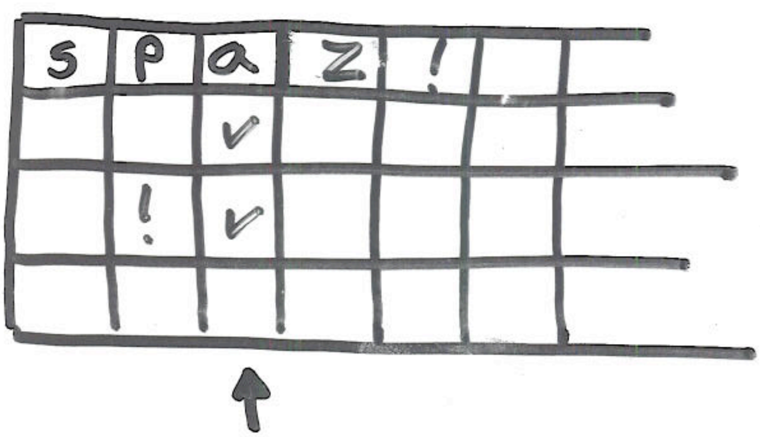
Are all languages recursive?

Adding Features to Turing Machines

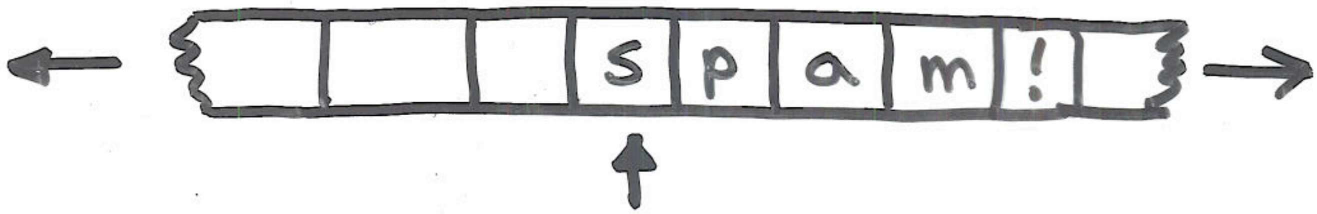


- Multiple Tracks

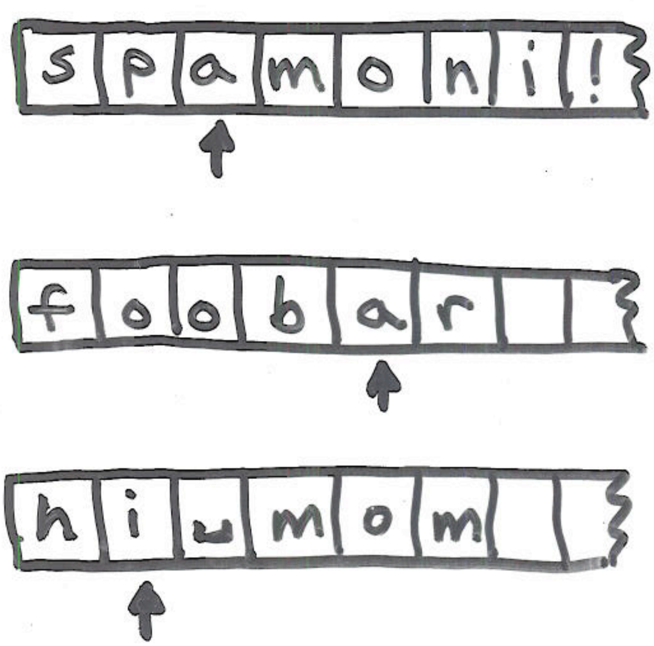




- Two-way infinite tape



- Multiple tapes with independent heads

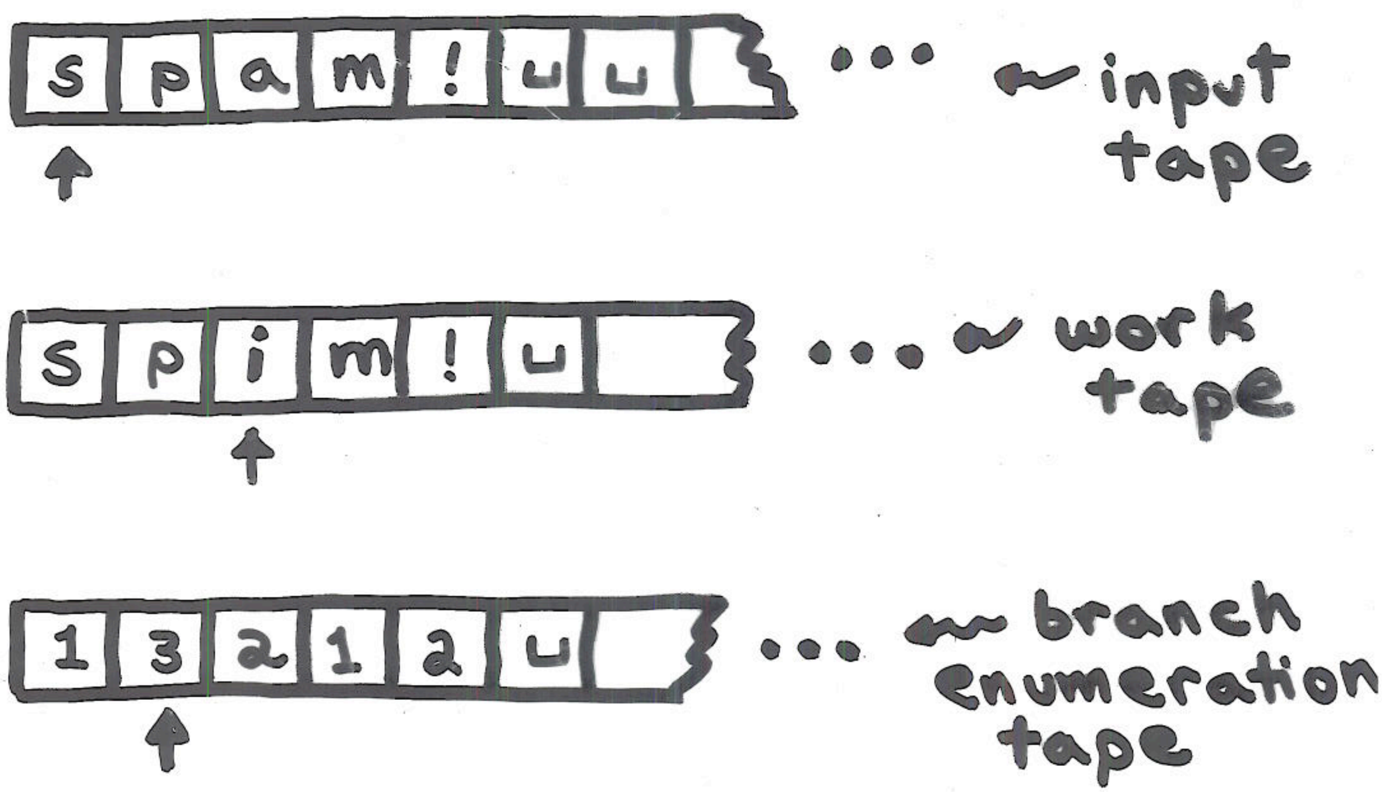


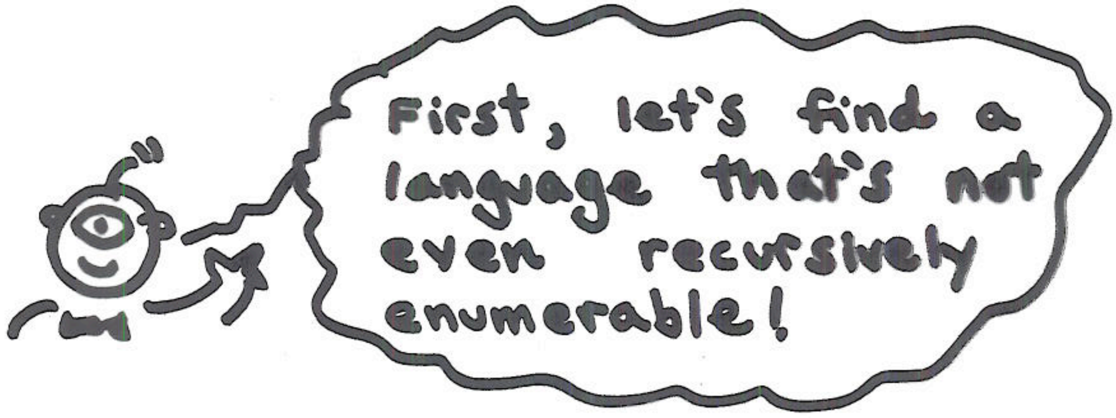
- **Nondeterminism**

(general nondeterminism)
- some computation paths may run forever!

$$\delta(q_i, a) = \{(q_{j_1}, b, L), \ldots, (q_{j_k}, d, R)\}$$

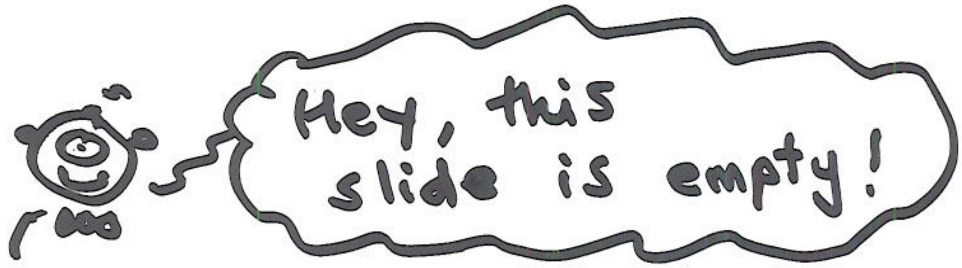
Let r denote the max # of nondet. choices.





- TM encodings
- L_d

TMs as computers of functions and number generators



Question: If L is recursive, is $\overline{L}$ recursive?

Question: If L is r.e., is $\overline{L}$ r.e.?

Question: If L and $\overline{L}$ are r.e. what else can be inferred?

$L_{TM} = \{ \langle M, w \rangle \mid$ TM M accepts string $w \}$

$L_{Halt} = \{ \langle M, w \rangle \mid$ TM M halts on string $w \}$

$L_\emptyset$ for DFA's, PDA's and TM's

$$L_{\emptyset,\mathrm{DFA}} = \{ \langle D \rangle \mid D \text{ is a DFA and } \mathcal{L}(D) = \emptyset \}$$

$$L_{\emptyset,\mathrm{PDA}} = \{ \langle P \rangle \mid P \text{ is a PDA and } \mathcal{L}(P) = \emptyset \}$$

$$L_{\emptyset,\mathrm{TM}} = \{ \langle M \rangle \mid M \text{ is a TM and } \mathcal{L}(M) = \emptyset \}$$

which of these are recursive?

a.k.a. "decidable"

Rice's Theorem

Every nontrivial property of the languages accepted by TM's is undecidable.

How do we formally define "nontrivial property"?

<u>Ex</u>

$L_{infinite} = \{ \langle M \rangle \mid \mathcal{L}(M) \text{ is infinite} \}$

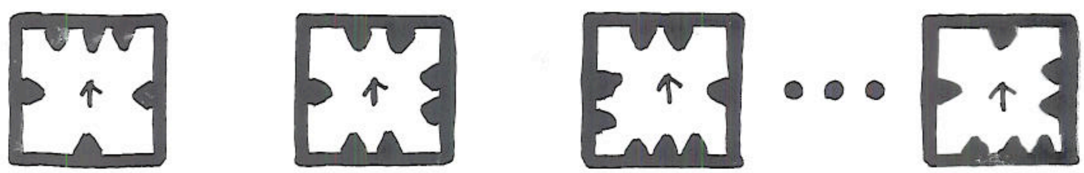
$\mathcal{P} = \{ L_1, L_2, L_3, \ldots \}$

set of all infinite languages accepted by TMs (all infinite r.e. languages)

<u>Def</u> A <u>property</u> of the r.e. languages is any set $\mathcal{P}$ of r.e. languages.

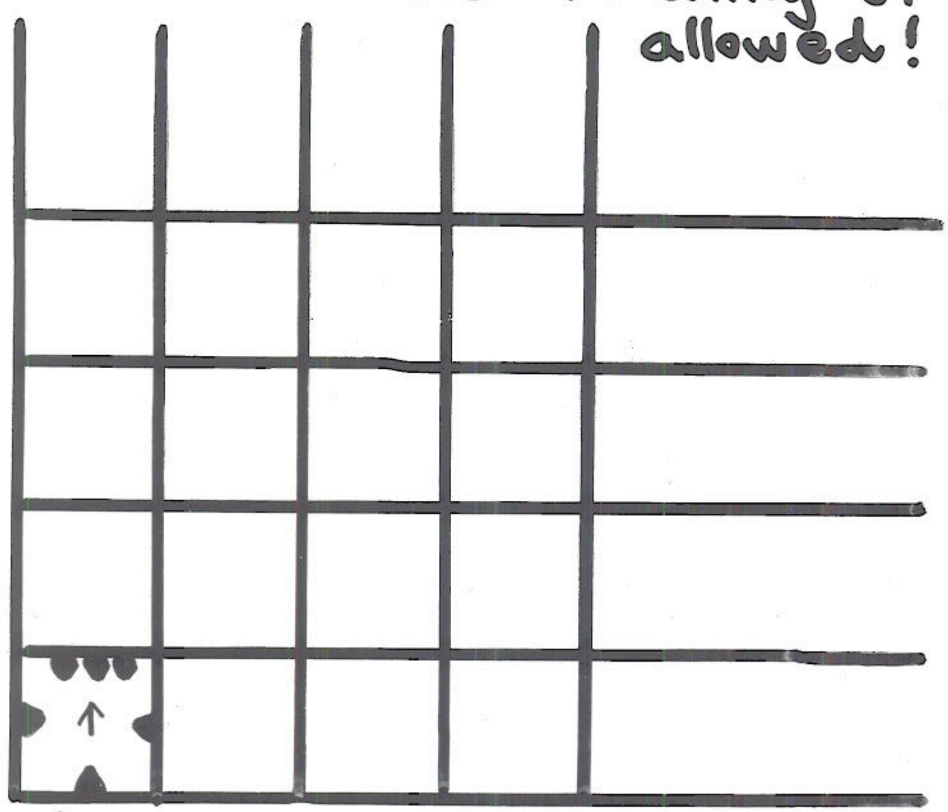
<u>Def</u> Property $\mathcal{P}$ is <u>nontrivial</u> if $\mathcal{P} \neq \emptyset$ and $\mathcal{P} \neq$ all r.e. langs.

An Undecidable Tiling Problem



special origin tile

- Given finite number of different types of tiles.
- Given an infinite number of tiles of each type.
- No rotating of tiles allowed!



An Origin Tile must be placed here

Claim: If we could solve the tiling problem then we could solve the halting problem!

(therefore the tiling problem is undecidable!)

$\langle M, w \rangle$ ← input to halting problem

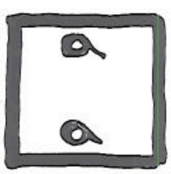
w ← spam

M ← $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

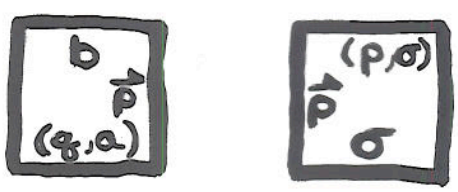
Build the following tile types:

1. 

origin tile

2. 

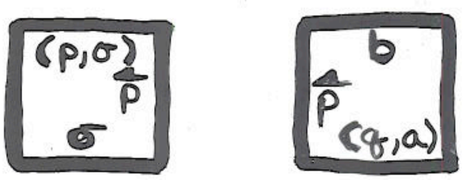
$\forall\ a \in \Gamma$ including blank (B)

3. 

new state, new sym

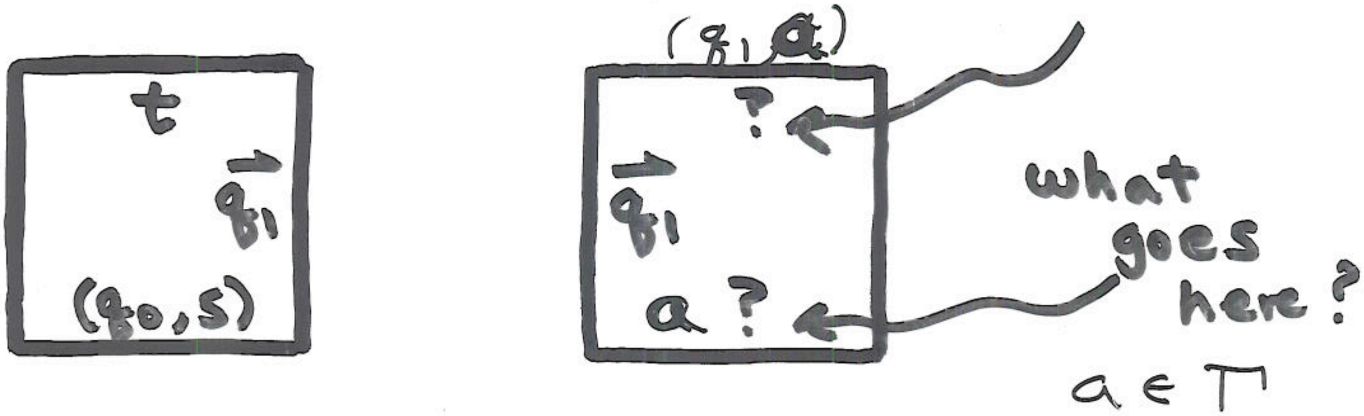
$\delta(q,a) = (p, b, R)$

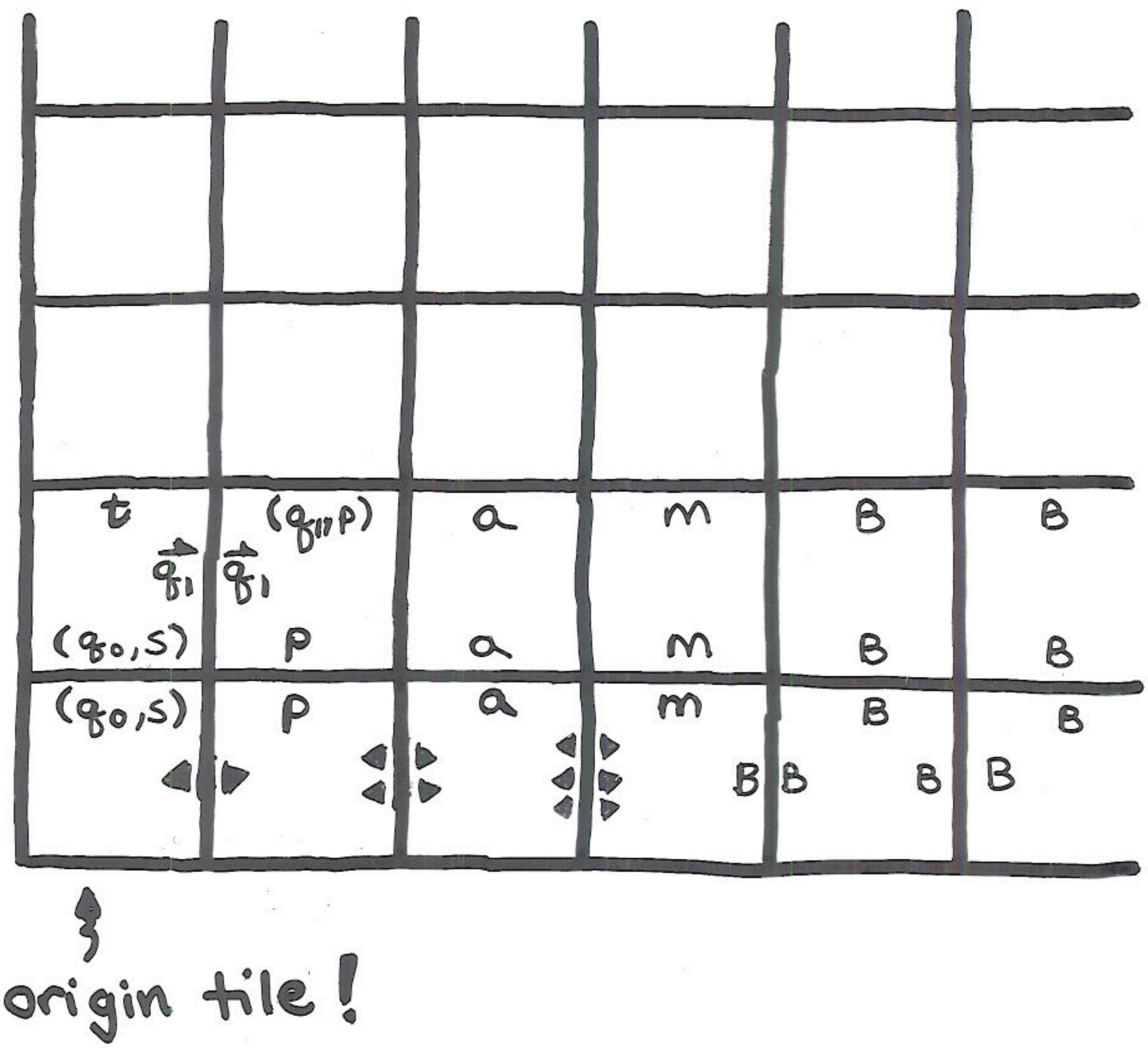
one such tile for every $\sigma \in \Gamma$

4. 

$\delta(q,a) = (p, b, L)$

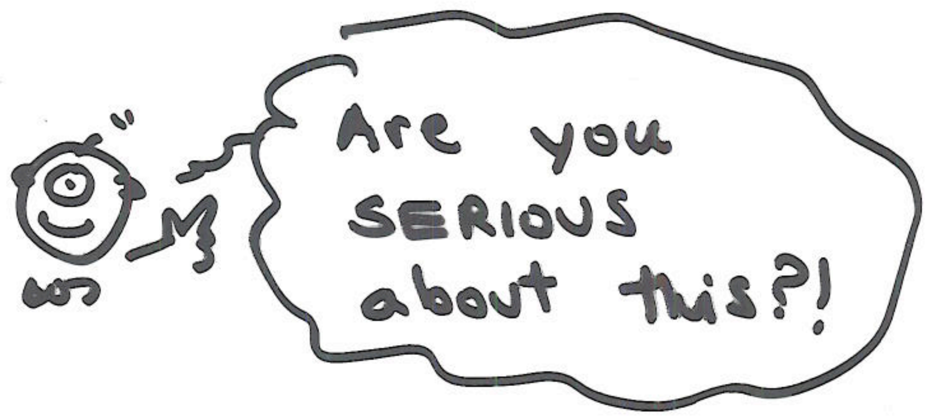
$$\delta(q_0, s) = (q_1, t, R)$$





State Minimization for TMs

Theorem: Every recursively enumerable language is accepted by a TM with only 4 states (5 with explicit "reject") but determining if 3 (4 ↑) or fewer states suffice is undecidable.

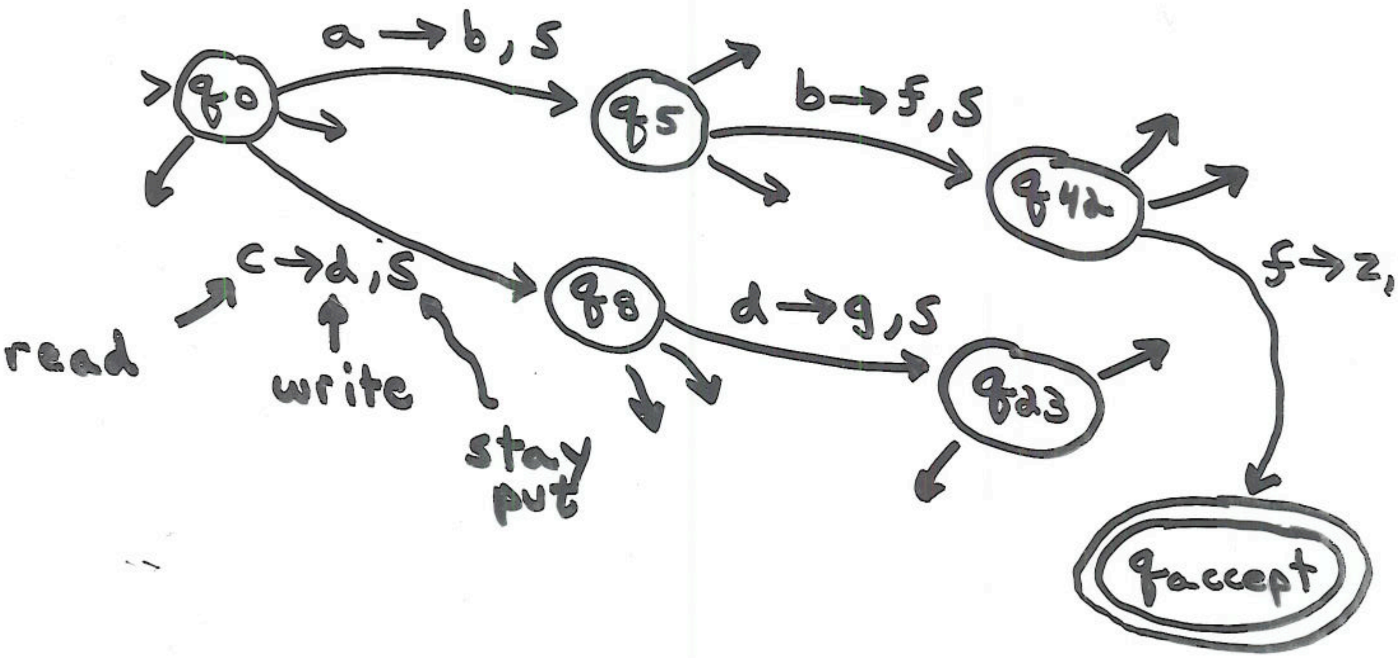


Claim: Every recursively enumerable language is accepted by a TM with only 4 states.

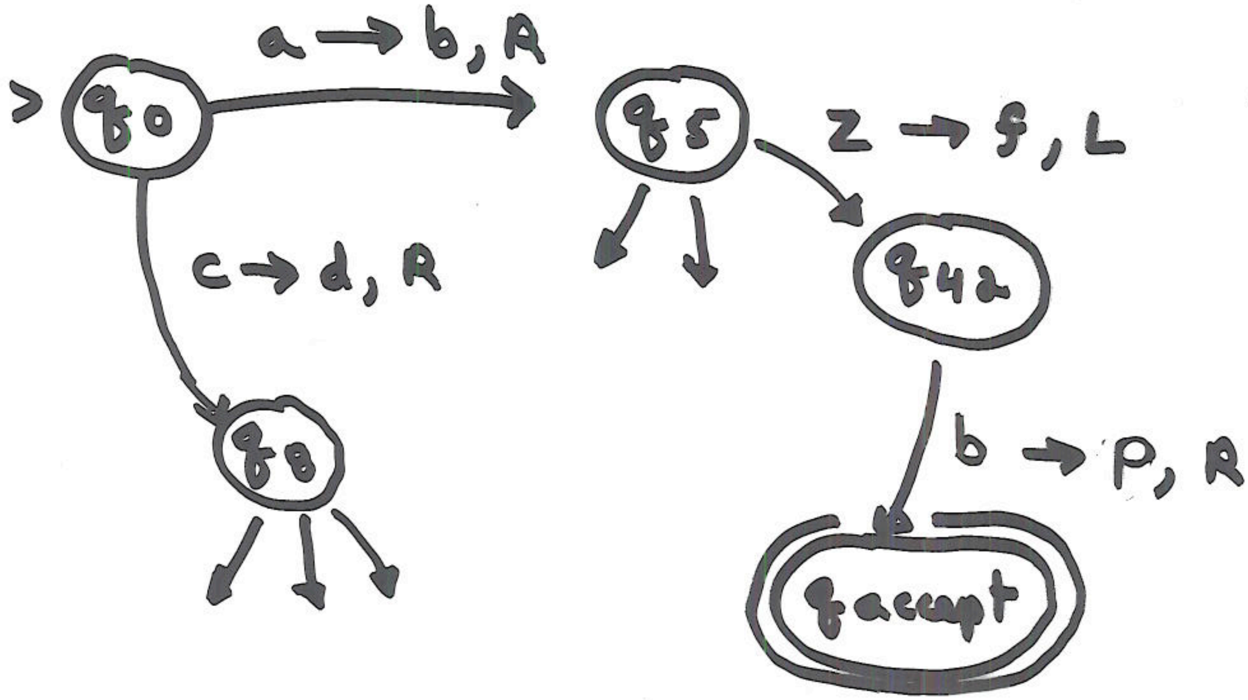
5 with explicit "reject"

Amazing!

Assume that TM's don't move their tape heads...

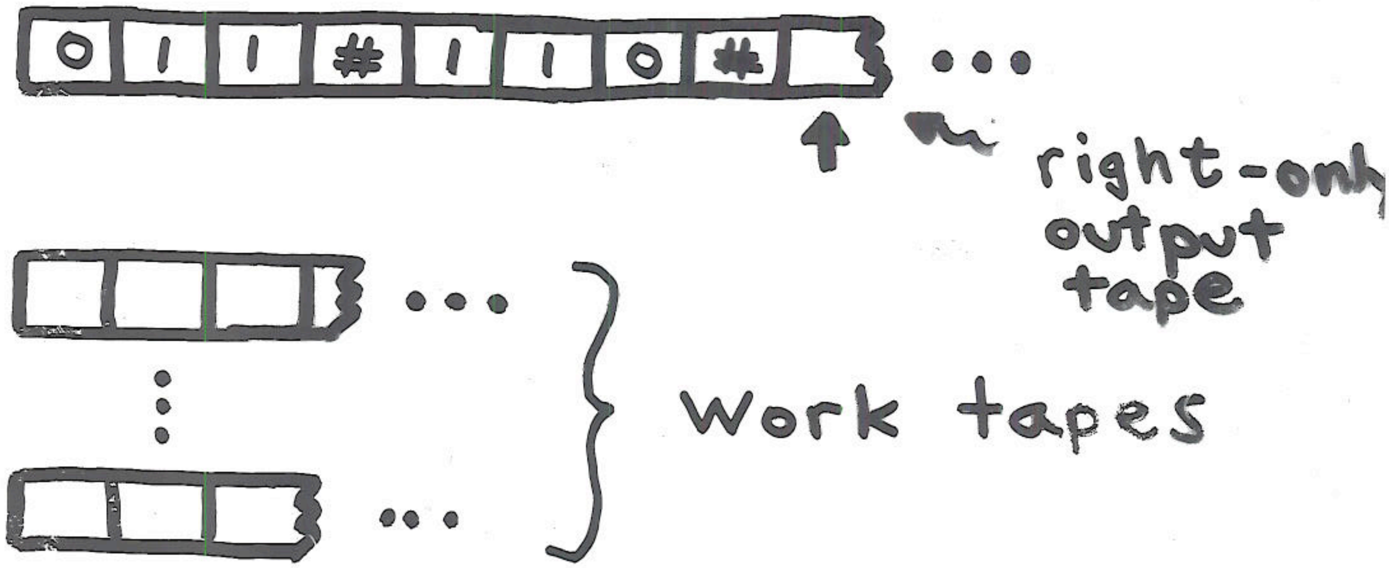


"OK, but TM's can move their heads!"



TMs as Generators

Generator TMs



- Generator halts if language is finite, runs forever if language is infinite.

- Prints out every word in the language in finite time.

Theorem 1:

L is recursive iff

L is generated in lexicographic order by some generating TM.

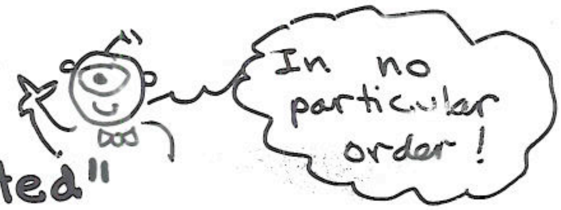
Theorem 2:

L is recursively enumerable

iff

L is generated by some generating TM.

or "enumerated"



The Recursion Theorem!

(How to construct a program that prints itself out, and why it's useful)

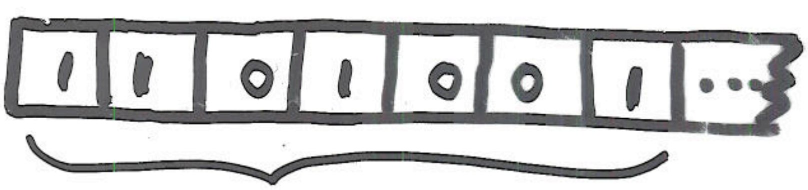
First, the ^Budget Quote-O-Matic!

only $99.95

input: word w

impress your friends!

⋮ quote-o-matic chugs!

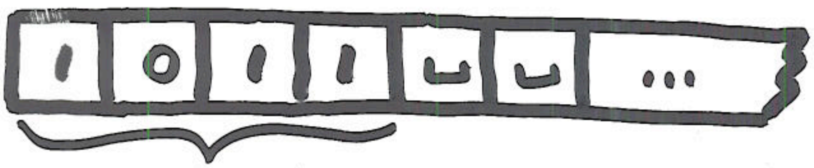


output: code for a TM M s.t. when run, M

1. erases its tape
2. stamps w out on tape
3. halts

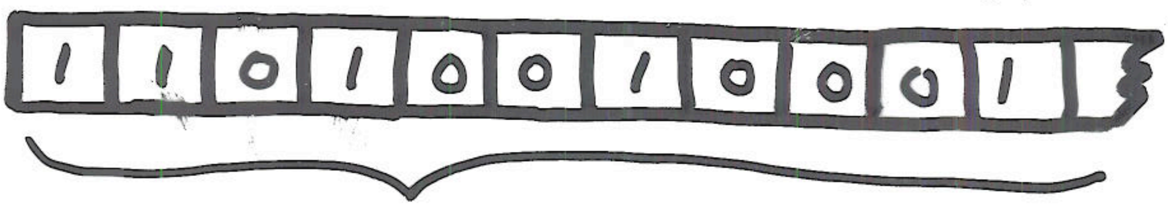
Now, the NEW top-of-the-line Super Quote-and-Shift-O-Matic!

(For serious TM enthusiasts ages 18 to 2^{18})

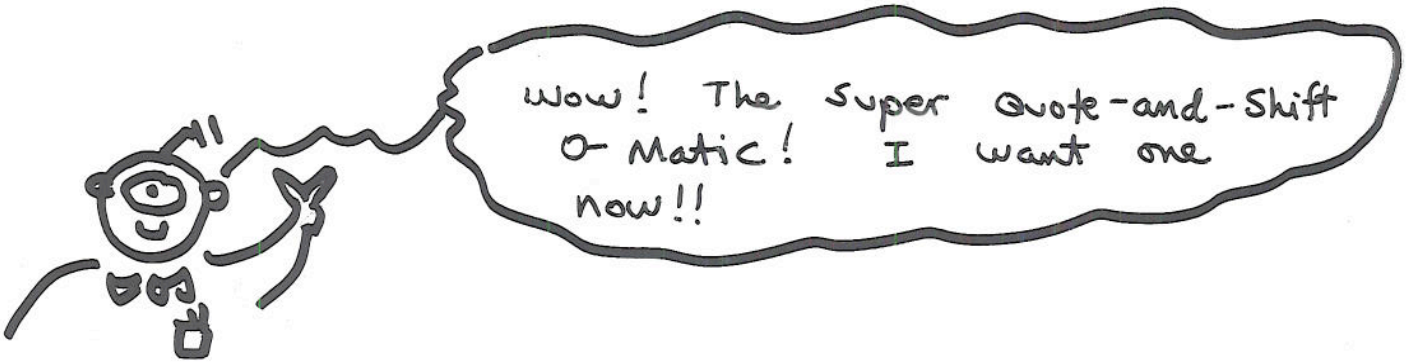


input: word w

⋮ Through the miracle of Turing Machine technology!



output: The previous TM (erases tape, prints w on tape, & halts) followed immediately by w.



A Self-Printing TM !

Objective: Construct a TM that, when run, ignores its input, prints its own TM code on the tape, and halts.

The Recursion Theorem

really $\langle M \rangle, w$

Let T be a TM that takes $\langle M, w \rangle$ as input and computes some function on $\langle M, w \rangle$.

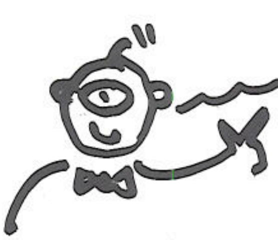
It is possible to construct a TM R s.t. when R is run on input w...

R

1. R prints itself out on the tape and shifts w over. ($\langle R, w \rangle$ now on tape)

   $\langle R \rangle, w$

2. R now computes/performs T on $\langle R, w \rangle$ ← $\langle R \rangle, w$



Why it's useful ...

Example 1

$\langle M \rangle, w$

Recall $L_{TM} = \{ \langle M, w \rangle \mid M$ accepts $w \}$

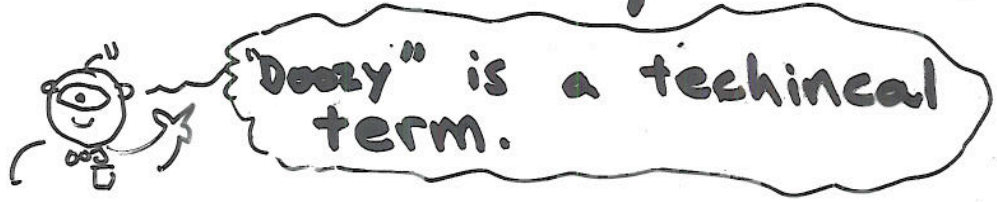
Assume L_{TM} is decidable.
Let M_{TM} be a TM for L_{TM}.

Recall...

$$L_{\emptyset,TM} = \{ \langle M \rangle \mid \mathcal{L}(M) = \emptyset \}$$

$$L_{42} = \{ \langle M \rangle \mid \mathcal{L}(M) \text{ contains exactly 42 words} \}$$

The Doozy!



$MIN_{TM} = \{ \langle M \rangle \mid$ There is no TM with a shorter description that accepts $L(M) \}$

Is MIN_{TM} recursive?

No!! It's not even r.e.!

Assume MIN_{TM} is r.e.

Let G be a generator for it.

- What is a fixed point of a function?

Theorem:

Let $f : \Sigma^* \rightarrow \Sigma^*$ map TM codes to TM codes and be computable. Then $\exists$ a TM M s.t. $t(\langle M \rangle) = \langle M' \rangle$ where M and M' are equivalent.

Unrestricted Grammars

- Recall that

$$L = \{ \circledcirc^{2^i} \mid i \geq 1 \}$$

terminal symbol

is NOT context-free

- consider this weird unrestricted grammar ...

① $S \to A C \circledcirc B$

② $C \circledcirc \to \circledcirc \circledcirc C$

③ $C B \to D B$

④ $C B \to E$

⑤ $\circledcirc D \to D \circledcirc$

⑥ $A D \to A C$

⑦ $\circledcirc E \to E \circledcirc$

⑧ $A E \to \epsilon$

I like being in grammars! It's fun & there are lots of neat variables to meet

One More Amazingly neat thing ...

... about Turing Machines

Claim: The languages generated by unrestricted grammars are exactly the recursively enumerable languages.

$S \rightarrow ABabcD \mid cDe$

$Ba \rightarrow aaB$

$\vdots$

Claim 1: Every unrestricted grammar has a corresponding TM.

may not halt on strings not generated by grammar!

Claim 2: Every TM has a corresponding unrestricted grammar.

Claim 1: Every unrestricted grammar has a corresponding TM

Claim 2: Every TM w/(rec. enum language) has a corresponding unrestricted grammar

Pf: $M = (Q, \Sigma, \Gamma, \delta, q_{start}, q_{accept}, q_{reject})$

input alphabet ($\sqcup \notin \Sigma$)

tape alphabet ($\sqcup \in \Gamma$, $\Sigma \subseteq \Gamma$)