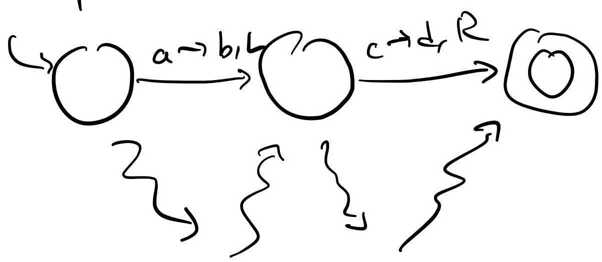
$\overline{E_{TM}} = \{ \langle M \rangle \mid M \text{ is a TM that accepts at least 1 string} \}$.

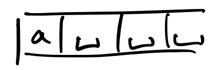
<u>Puzzle</u>: Can we build a TM that <u>recognizes</u> this language?

(always halt+accept if $\langle M \rangle \in \overline{E_{TM}}$
can reject or loop if $\langle M \rangle \notin \overline{E_{TM}}$)

— check if input is an encoded TM, reject otherwise.

Idea: look at TM
"see if a path exists to accept state"





— Check alphabet. Say $\Sigma = \{0,1\}$.

— Enumerate all strings over Σ: ε, 0, 1, 00, 01, 10, 11, ...

— Call these strings $s_1, s_2, s_3, \ldots$

TM 1:

— Start by simulating $M(s_1)$.
 If M accepts, accept
 If M rejects, run $M(s_2)$,
 etc...

worry: what if $M(s_1)$ runs forever,
$M(s_2)$ accepts?

$M_{\overline{E_{TM}}}$: "On input $\langle M \rangle$:

- Let $s_1, s_2, \ldots$ be an enumeration of all strings over Σ.

(*) — For $i = 1, 2, 3, \ldots$

- Simulate $M(s_1), M(s_2), \dots M(s_i)$ for i steps each.
- Accept if any simulation accepts."

Imagine M accepts s_j after running for k steps.
We'll accept when simulating $M(s_j)$ in the $\max(j,k)$th iteration of the loop.

(If $M \notin \overline{E_{TM}}$, our machine runs forever/loops.)

Today:

1. Reducing variant TMs to regular TMs.
2. NTMs. $(P \overset{?}{=} NP)$
3. Enumerator.
4. More undecidable/unrecognizable L's — proof by reduction.

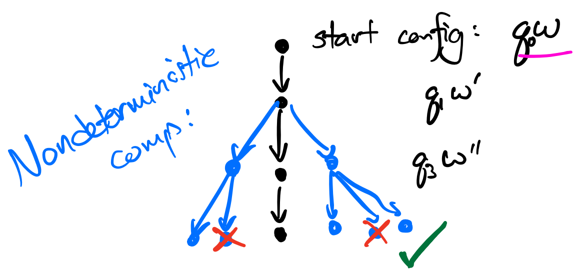
2. NTMs

Defined just like TMs, except with the transition function

$$\delta : Q \times \Gamma \longrightarrow \underline{\mathcal{P}(Q \times \Gamma \times \{L, R\})}$$

Accept if and only if some branch reaches q_{accept}.

Theorem: Every NTM has an equivalent TM.
(TM $\to$ NTM trivial.)



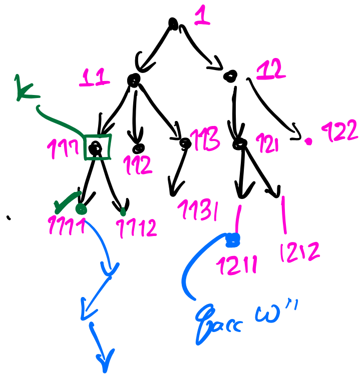
Idea: explore this "tree" of computation, and accept if we find an accepting branch.

Proof. Given an NTM M_N, build an equivalent TM M_D as follows. W.l.o.g., I'll assume that M_D has 3 tapes. (We've shown that k-tape TMs are equivalent to 1-tape TMs.)

Tape 1: store the input string w.

Tape 2: store the "address" of a location in the tree of computation.

Tape 3: "simulation" tape for computation.



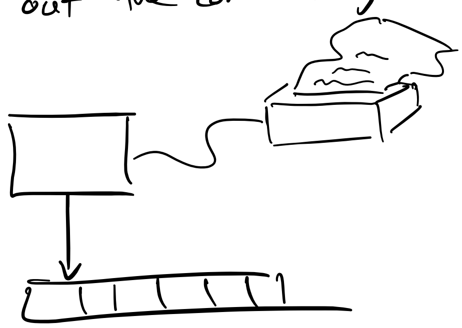
M_D = " On input w:

- write w to the input tape
- start at address 1, the root of the tree
- perform a breadth-first search on the tree.
  - if we're at address k: (111)
    - find the next unexplored child. (1112).
    - reach that address by simulating all computation steps that got us to address k, then following the next (second) untaken transition.
- accept if any branch reaches an accept state."

(Note: BFS ≫ DFS, because we avoid loops.)

3. Enumerator.

A TM with an attached printer that can write down and print out the contents of its tape with a special operation.



Theorem. A language is (Turing-) recognizable if and only if some enumerator enumerates it. (Turing-recognizable = RE.)

└ prints out all strings in the language, or if infinite, eventually will print out any given string.

Proof.

1. If an enumerator E enumerates some language, there exists a TM T that recognizes that language.

$T =$ "On input w:
- Simulate a hard-coded copy of E. On a "print" operation, check the "printed" string against w and accept if they are equal."

2. If a TM T recognizes a language L over Σ, there exists an enumerator E that enumerates L.

(Let $s_1, s_2, \cdots$ be an infinite list of strings over Σ.)
(ε, 0, 1, 00, 01, ...)

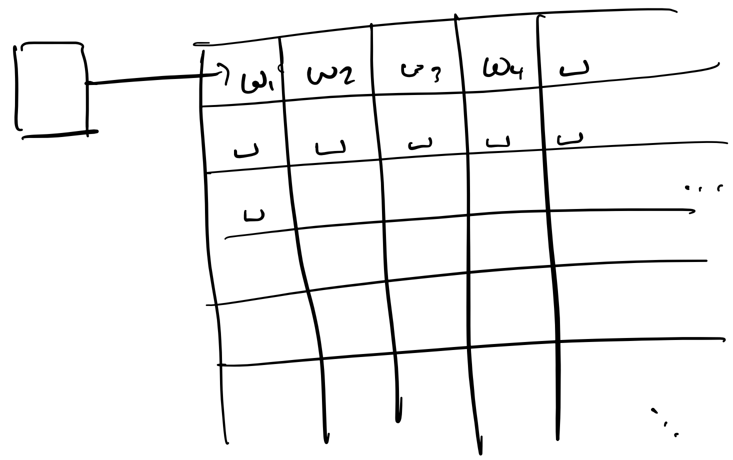
$E =$ "Repeat the following for $i = 1, 2, 3, \cdots$:
– Simulate $T(s_1), T(s_2), \dots T(s_i)$ for i steps each. If any computation accepts, print out the corresponding input string."

Claim: Let $s_j \in L$. T eventually accepts on s_j by definition, say after k steps. Then E prints s_j on loop number $\max(j,k)$.
($s_j \notin L$: $T(s_j)$ never accepts, so s_j never prints.)

Conclusion: E enumerates L. □

——— Back at 2:16 ———

2DTM: A TM with a 2-dimensional tape, extending infinitely to the right and down.



To reduce 2DTM to TM: $\qquad \delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, U, D\}$

Simulate by storing multiple "rows" of my 2D tape on the tape of my regular TM, separated by delimiters.

w_1	w_2	w_3	w_4	#	$\sqcup$	#	$\sqcup$	# …

Two additional subroutines:

- make space on one "row" by moving everything R.
- add new "rows" as our 2DTM reaches them for the first time.

3. Proving Undecidability & Unrecognizability by Reduction

<u>Fact</u>: USPTO has a special policy against patents for perpetual motion machines.

<u>Corollary</u>: If you apply for a patent on a device that would let you build a perp. motion machine, you get denied.

We know $A_{TM} = \{\langle M, w\rangle \mid M(w) \text{ accepts}\}$ is undecidable.

$\overline{A_{TM}}$ is unrecognizable.

We'll show $HALT_{TM}$ decidable $\Longrightarrow$ A_{TM} decidable, which is a contradiction.

Prop. $HALT_{TM} = \{\langle M, w\rangle \mid M(w) \text{ halts}\}$.

Proof. We'll show $HALT_{TM}$ decidable $\Longrightarrow$ A_{TM} decidable.

Let T be some decider for $HALT_{TM}$ (assumption).

always halts.

M_1 = "On input $\langle M, w\rangle$:

- Use a hard-coded copy of T to see if $M(w)$ halts. If $\langle M, w\rangle \notin HALT_{TM}$, reject.
- Otherwise, we know $M(w)$ halts. Simulate $M(w)$ and accept if and only if $M(w)$ accepts."

Claim: M_1 is a decider, because T is a decider (always halts.)

Thus M_1 decides A_{TM}, contradicts the fact that A_{TM} is undecidable.

So our assumption is false: $HALT_{TM}$ is undecidable. □

EQ_{TM}, ALL_{TM}, ...

$\downarrow$

dec. E_{TM}

Decide $HALT_{TM}$

$\downarrow$

Decide A_{TM}

$\downarrow$

Impossible.

Proposition. $E_{TM} = \{\langle M\rangle \mid M$ is a TM that never accepts any string.$\}$ is undecidable.

Proof. Assume for contradiction that H decides E_{TM}. We'll use H to build a decider for A_{TM}, which is a contradiction.

$$A_{TM} = \{\langle M, w\rangle \mid M(w) \text{ accepts}\}.$$

$T =$ "On input $\langle M, w\rangle$:

// Goal: make N s.t. $\langle N\rangle \in E_{TM}$ if and only if $M(w)$ accepts.

(1) Define a new TM N that

- rejects all strings $x \neq w$
- on input w, simulates $M(w)$ and accepts if $M(w)$ accepts.

// If $M(w)$ accepts, $L(N) = \{w\}$.
else, $L(N) = \emptyset$.

finite $\{$ (2) Simulate $H(\langle N\rangle)$.

If $H(\langle N\rangle)$ accepts, $M(w)$ doesn't accept, so reject.
If $H(\langle N\rangle)$ rejects, $M(w)$ accepts, so accept."

T decides A_{TM}, which is a contradiction. Thus our assumption is false and E_{TM} is undecidable.

Prop. E_{TM} is unrecognizable.

Proof. $\overline{E_{TM}}$ is recognizable. (Weds' intro puzzle.)

If E_{TM} was recognizable, we could run recognizers for E_{TM} and $\overline{E_{TM}}$ "simultaneously", and accept/reject when one of the two accepts.

run each simulation for one step each, go back and forth.

This lets me decide E_{TM}, which is a contradiction. Thus our assumption is false and E_{TM} is unrecognizable.

Back at 3:15

$ALL_{TM} = \{\langle M\rangle \mid M \text{ is a TM that accepts all strings.}\}$

Show ALL_{TM} decidable $\Rightarrow \underline{A_{TM}}$ decidable.

$\{\langle M, w\rangle \mid M(w) \text{ accepts}\}$

Assume some TM T decides ALL_{TM}.

A_{TM} decider: "On input $\langle M, w\rangle$:

- Let N be a TM that accepts $x \neq w$, and simulates $M(w)$ on w, accepts if $M(w)$ accepts.
- Accept if $T(\langle N\rangle)$ accepts."

$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1, M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

Show EQ_{TM} decidable $\Rightarrow \underline{E_{TM}}$ decidable.

$\{\langle M\rangle \mid M \text{ doesn't accept any string.}\}$

Assume some TM R decides EQ_{TM}

A_{TM} decider: "On input $\langle M, w\rangle$:

- Let M_2 be a TM that accepts w, rejects $x \neq w$.
  $(L(M_2) = \{w\})$.
- Let M_3 be a TM that rejects $x \neq w$, and simulates M on w.
  $(L(M_3) = \{w\}$ if $M(w)$ accepts, $\emptyset$ otherwise.)
- Run $R(\langle M_2, M_3\rangle)$. $L(M_2) = L(M_3)$ if and only if $M(w)$ accepts.
  Accept if and only if $R(\langle M_2, M_3\rangle)$ accepts."

E_{TM} decider: "On input $\langle M\rangle$:

- Let M_{NO} be a TM that rejects everything
- Simulate $R(\langle M, M_{NO}\rangle)$, and accept if and only if R accepts."

(** bonus: $REGULAR_{TM} = \{\langle M\rangle \mid M \text{ is a TM, } L(M) \text{ is regular}\}$
Show $REGULAR_{TM}$ decidable $\Rightarrow A_{TM}$ decidable.)

Sipser p. 219.

Rice's Theorem:

Consider any language of the form

$$P_X = \{\langle M\rangle \mid M \text{ is a Turing Machine and } L(M) \text{ has property } X\}.$$

- If P_X includes at least one, but not all TMs, and
- the property X depends only on $L(M)$,

then P_X is undecidable.

Punchline: all nontrivial properties of TMs are undecidable!

Today:

- NTMs, 2DTMs = TMs.
- Enumerators enumerate the Turing-recognizable languages
- "If A were decidable, then A_{TM}/$HALT_{TM}$/E_{TM} or etc. would be decidable" $\Longrightarrow$ A undecidable.

Next time:

move from computability to complexity.

Q: using regular operations to help show EQ_{TM} undecidable?

N s.t. L(N) = this

$$L(M_1) = L(M_2) \text{ iff } \underbrace{L(M_1) \cap \overline{L(M_1)}}_{} = \emptyset$$

$$\text{and } \overline{L(M_1)} \cap L(M_2) = \emptyset.$$

("E_{TM} undecidable $\Rightarrow$ EQ_{TM} undecidable.")