COMS W3261, Lecture 10:
Making Hard Decisions

Teaser: Is the language

$$A_{DFA} = \{\langle B, w\rangle \mid B \text{ is some DFA that accepts string } w\}$$

<> indicate math objects encoded as strings.

decidable?

→ Decidable = there exists some TM that always accepts on strings in the language and always rejects on strings not in the language.

M_1 = "On input $\langle B, w\rangle$:
0. reject if $\langle B, w\rangle$ doesn't encode a valid TM and string.
1. simulate B on w and accept if B accepts, reject if B rejects."

always terminates.

What about A_{NFA}? $\{\langle B, w\rangle \mid B \text{ is an NFA that accepts } w\}$.

Yes - decidable.

1. TM converts $B \rightarrow$ DFA, then use earlier proof.
2. Do BFS on the tree of computation.

What about A_{REX}? $\{\langle R, w\rangle \mid R \text{ is a regular expression that generates } w.\}$

Yes - decidable.

$R \rightarrow$ NFA $\rightarrow$ DFA.

(These are all examples of high-level descriptions.)

Announcements: HW #5 due today, 8/2/21 @ 11:59 EST
HW #6 due Monday, 8/9/21 @ 11:59 EST

Final: 8/10 and 8/11. Review sessions:
1-4pm in person (CS lounge)
5-8pm on Zoom on 8/9.

Donuts / bagels + coffee wednesday.

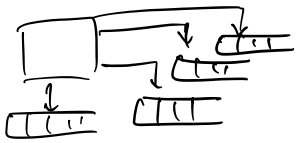
Readings: Sipser 4.1 — Decidable languages
4.2 — Undecidable languages.

Today:
1. Review
2. Decidable Languages, $CF \subseteq$ Turing-Recognizable
3. Undecidable Languages.

---

1. Review.

— Multitape TMs.



$$\delta: Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R, S\}^k$$

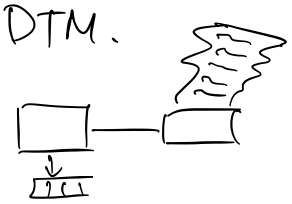
Thm. Every Multitape TM has an equivalent single-tape TM.

— Nondeterministic TMs.

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

Thm. Every NTM has an equivalent DTM.

- Enumerators. TM's with a printer:



Thm: A language is Turing-recognizable if and only if some enumerator enumerates it.

Levels of description:

- Formal descriptions := 7-tuples

↳ Implementation-level descriptions := prose descriptions of tape management & head movement.

↳ High-level descriptions := prose descriptions of algorithms — precise finitely specified process — that ignore implementation details entirely.

Church-Turing thesis: our intuitive notion of algorithm corresponds exactly to what a TM can do.

2. Some more things TMs can do (decidable languages)

Example. Show that $E_{DFA} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ decidable?

T = "On input $\langle A \rangle$, where A is a DFA:

(implicitly step 0: filter out bad input.)

1. Mark the start state of A
2. Mark all states accessible from the start state,
3. Repeat (2) until no more states are accessible.
4. Reject if we've marked any accept state, accept otherwise."

DFA finite: always terminates.

Example: $EQ_{DFA} = \{\langle A, B \rangle \mid A, B \text{ are DFAs and } L(A) = L(B)\}$.

Idea 1: Simulating all strings on A, B and reject if they ever differ.

✗ not a decider.

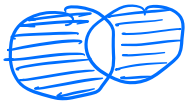
Idea 2: Using closure properties — $L(A) = L(B)$ unless there is some string in A but not B, or some string in B but not A.

Fact: given DFAs for A and B, we can build DFAs that recognize
- $A \cup B$. (Simulate A, B, accept if either accept.)

- $A \cap B$.
- $\overline{A}$, $\overline{B}$ (simulate A and give the opposite result / build a new DFA with switched accept/reject states.)

Thus given A and B, we can build a DFA C that recognizes $L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$

recognized by A, not by B — by B, not by A.



$A \cap B = A = B$ iff $A = B$.

Now: $L(C) = \emptyset$ if and only if $L(A) = L(B)$. We can use our decider for E_{DFA} on C.

brackets? encoding as a string.

F = "On input $\langle A, B \rangle$, where A, B are DFAs:

- Build C as above.
- Simulate a decider for E_{DFA} on C, and accept/reject if the decider accepts/rejects." □

– We don't actually ever write down $L(A) \cap L(B)$.

↳ this set might be infinite!

Some other decidable languages: (proofs in Sipser 4.1)

- $A_{CFG} = \{\langle G, w \rangle \mid G \text{ is a context-free grammar that generates the string } w\}$

(Idea: convert to CNF.)

$A \to B$

$B \to A$.

- $E_{CFG} = \{\langle G \rangle \mid G \text{ is a CF grammar, } L(G) = \emptyset\}$
- ~~$EQ_{CFG} = \{\langle G, H \rangle \mid G, H \text{ are CFGs and } L(G) = L(H)\}$.~~

Erratum: EQ_{CFG} is NOT decidable! See Sipser p. 200 for details.

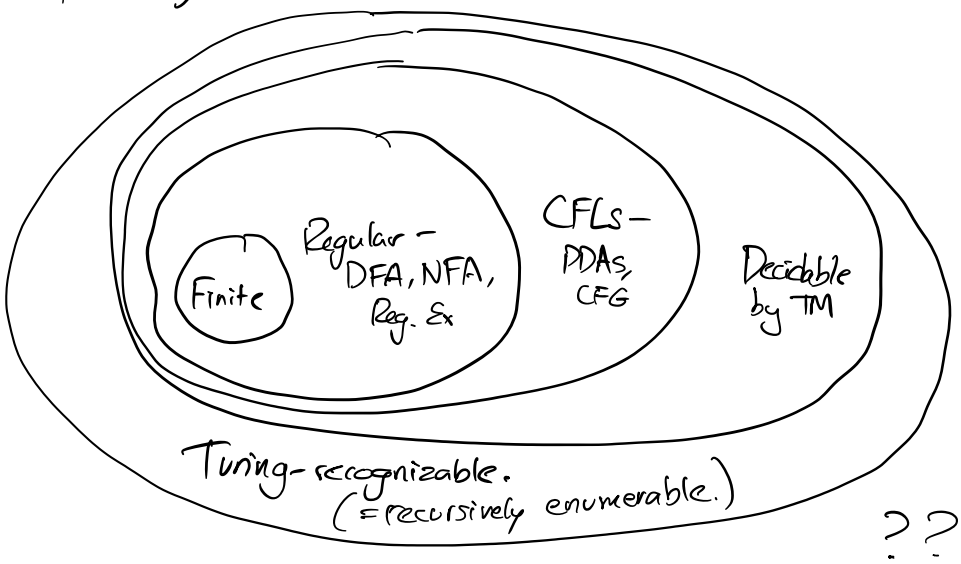
Theorem: Every Context-Free language is decidable.

Proof: Given a CFL A, let G be a CFG for A. Let S

be a TM that decides A_{CFG}. Define this TM:

M_G = "On input w: (hard-coded into M_G)
Run S on $\langle G, w \rangle$,
and accept/reject if S accepts/rejects." □

New picture of the universe:



Break: Back at 11:25.

Next: countable sets, undecidable, unrecognizable languages.

---

2. Countable & Uncountable Sets

Idea: Some problems can't be solved by computers (TMs).

Consider $A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$

($\langle M, w \rangle$: encoded TM.)

We can recognize A_{TM}: simulate M on w and accept if M accepts.
Deciding is harder — how do we know if $M(w)$ is looping or just running for a long time?

($M(w)$: "running M on w")

Recall: Long ago, we proved that $\mathbb{R}$ can't be "listed out" as a sequence programatically. Any such sequence would have to be incomplete.

Suppose $a_1, a_2, a_3, \ldots$ is a sequence that contains every number in $\mathbb{R}$.

$a_1 = 0.00012\ldots$
$a_2 = 1.3684\ldots$
$a_3 = 4.011\ldots$
$a_4 = 4.00000\ldots$
$\vdots$

Using this sequence, we can build a real number n not in the sequence.

$n = 1.421\ldots$

$n \neq a_j$ for any j because the j^{th} digit is different.

Definition: A set is <u>countable</u> if it is finite, or if there exists a one-to-one mapping to the natural numbers $\mathbb{N} = 1, 2, 3, \ldots$

Example.

$\mathbb{Z}$ is countable.

n	f(n)
1	0
2	-1
3	1
4	-2
5	2
⋮	⋮

$2\mathbb{N}$ is countable.

n	f(n)
1	2
2	4
3	6
4	8
⋮	⋮

Exercise: Show $\mathbb{Q} = \frac{m}{n}$, for $m, n \in \mathbb{Z}$, are countable.

But— $\mathbb{R}$ not countable.

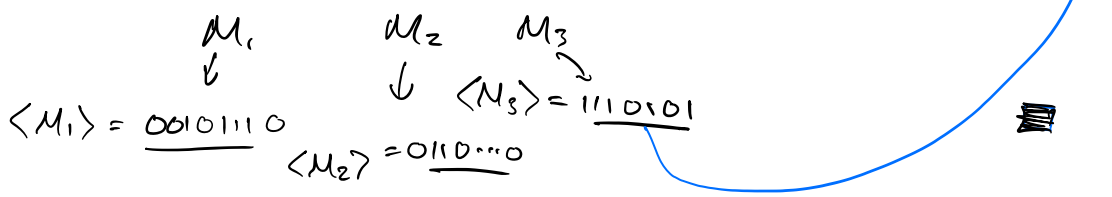
Observation: The set of all TMs is countable.

Proof: TM is a finite object by definition. Consider some encoding $\langle \cdot \rangle$ that encodes TMs as strings over $\{0,1\}$.

$\{0,1\}^*$, the set of binary strings, is countable.

Example: $\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots$

So: Let $f(M)$, for each TM M, map to its order in the list of encoded TMs according to our list of binary strings.



Yes - we can define a total ordering over the set of TMs according to some one-to-one mapping $f: \{\text{all TMs}\} \to \mathbb{N}$.

$$M_1 \prec M_2.$$

Observation: The set of infinite binary strings is uncountable.

$\neq \{0,1\}^*$

Proof: By diagonalization.

$b_1 = 00111010\ldots$
$b_2 = 11010\ldots$
$b_3 = 00010\ldots$
$\vdots$

$n = 101\ldots$

Create n by flipping the i^{th} bit of the i^{th} string. Now $n = b_j$ for any j.

Theorem: Some languages are not Turing-recognizable.

Proof: We can map languages $L \subseteq \Sigma^*$, $L \subseteq \{0,1\}^*$ to infinite binary strings using the following one to one mapping:

$$\Sigma^* = \varepsilon,\ 0,\ 1,\ 00,\ 01,\ 10,\ 11,\ \ldots$$

$$L \to 0\ 1\ 0\ 1\ 0\ 1\ 1\ \ldots$$

$$L = \{0, 00, 10, 11, \ldots\}$$

Map languages to infinite binary strings by making the i^{th} bit of the infinite string a 1 whenever the i^{th} string is in the language.

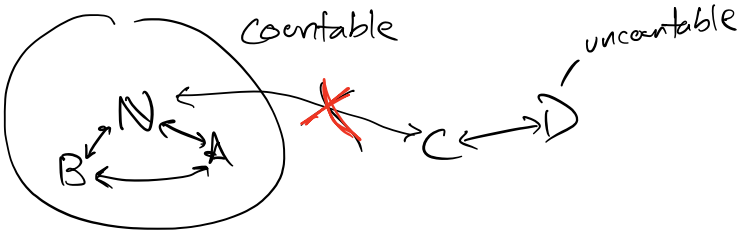
Thus the set of languages over $\{0,1\}$ is uncountable (second observation.)

The set of TMs is countable, so there can be no 1-to-1 mapping between TMs and languages. Specifically, we can't map every language to a TM that recognizes it. ∎

Break: back at 12:02.

---

3. Undecidable and Unrecognizable Languages.



Paradoxes.

Liar's paradox: "This statement is false." < True? No. / False? No.

Russell's paradox: "Consider the set of sets that don't contain themselves. Is S a member of itself?"

S

If $S \in S \Rightarrow S \notin S$.
If $S \notin S \Rightarrow S \in S$.

(Gödel's incompleteness theorem.)

Will show $A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ is undecidable.

Proof. Assume for contradiction that A_{TM} is decidable, and H

is a TM that decides A_{TM}.

$$H(\langle M, w\rangle) = \begin{cases} \text{accept if } M(w) \text{ accepts} \\ \text{reject if } M(w) \text{ rejects or runs forever.} \end{cases}$$

Now that we know H exists, we can build a new TM D that works as follows: D = "On input $\langle M\rangle$, where M is a TM:

1. Run H on $\langle M, \langle M\rangle\rangle$ (encoding of M and an encoding of M)
2. Output the opposite of H."

$$D(\langle M\rangle) = \begin{cases} \text{Accept if } M \text{ does not accept } \langle M\rangle. \\ \text{Reject if } M \text{ accepts } \langle M\rangle. \end{cases}$$

Consider what $D(\langle D\rangle)$ does:

$$D(\langle D\rangle) = \begin{cases} \text{Accept if } D \text{ does not accept } \langle D\rangle. \\ \text{Reject if } D \text{ accepts } \langle D\rangle. \end{cases}$$

So D cannot exist $\longrightarrow$ our assumption is false, A_{TM} is undecidable.

Definition: A language is co-Turing recognizable if its complement is Turing-recognizable.

Ex. A_{TM} is Turing-recognizable, so $\overline{A_{TM}}$ is co-Turing recognizable.

Theorem. A language is decidable if and only if it is recognizable and co-Turing recognizable.

Proof. If A is decidable, then A is recognizable by definition. Moreover, we can simulate a decider for A and accept when the decider rejects to recognize $\overline{A}$.

If A is both Turing-recognizable and co-Turing recognizable, then to decide A:

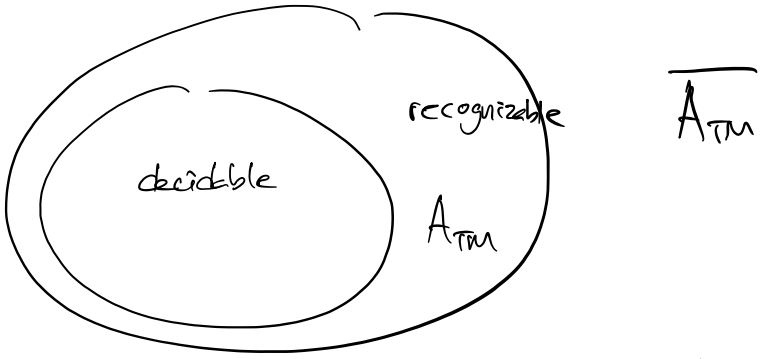
1. Run recognizers for A and for $\overline{A}$ in parallel

(we switch back and forth between simulations every few steps.)

2. If input string $w \in A$, recognizer for A eventually accepts. If $w \notin A$, my recognizer for $\overline{A}$ eventually accepts.

Theorem: $\overline{A_{TM}}$ is not Turing-recognizable.

Proof? If $\overline{A_{TM}}$ is Turing recognizable, then A_{TM} would be both recognizable and co-recognizable $\longrightarrow$ A_{TM} would be decidable. This is a contradiction, so $\overline{A_{TM}}$ is not recognizable.

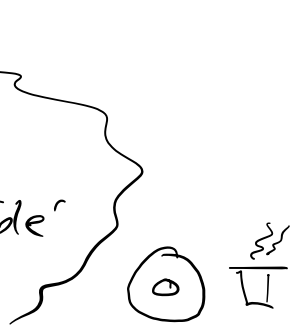


Next time: we'll show $HALT_{TM}$ is undecidable.

Show a bunch of reductions:

"If A decidable, B decidable."

↳ If B undecidable, A can't be decidable"



Time complexity.

Readings: Sipser 4.1 (decidable L's)
Sipser 4.2 (undecidable.)