COMS W3261 — Lecture 10: Making Hard Decisions (Part 1/2).

Teaser: Is the language

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

decidable? (Think high-level.)

Recall: decidable := TM always accepts on strings in language
always rejects on strings not in language.

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

0. reject if the input is not an encoded DFA followed by a string.
1. simulate B on w and accept/reject if B accepts or rejects."

What about A_{NFA}? $= \{\langle B, w\rangle \mid B$ is an NFA and B accepts on $w\}$.

Yes — decidable.

One way: simulate all branches of computation in parallel.

Another way: convert B to a DFA.

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

Yes - decidable.

One way: $R \rightarrow NFA \rightarrow DFA$.

Announcements: HW #5 due 8/2/21 @ 11:59 PM EST.
HW #6 due 8/9/21 @ 11:59 PM EST.
Final 8/10 – 8/11. (See Ed.) Review sessions
in person: 1-4pm Monday CS Lounge
virtually: 5-8pm EST Zoom.

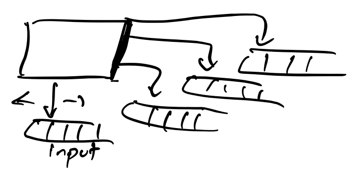
Donuts/bagels on Wednesday!

Readings: Sipser 4.1 (Decidable languages)
Sipser 4.2, 5.1 (Undecidable languages.)

Today: 1. Review
2. More decidable languages
3. Some undecidable (!) languages.

1: Review

- Multitape TMs.



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

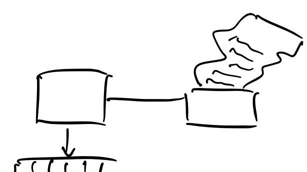
Thm. Every multitape TM has an equivalent single-tape TM.

- Nondeterministic TMs:

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

Thm. Every NTM has an equivalent DTM.

- Enumerators:



TMs with an additional "print" function that doesn't affect computation. They "enumerate" some (possibly infinite) language over the course of computation.

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

Levels of description:

→ Formal description = 7-tuple.

→ Implementation-level description = prose description of tape management and head movement.

→ High-level description = precise prose description of an algorithm, ignoring implementation details. (manipulating finite math objects.)

Church–Turing thesis: Our intuitive notion of an algorithm ("completely specified process") corresponds exactly to what a TM can do.

---

2. Some more decidable languages.

Example. Is $E_{DFA} := \{\langle A\rangle \mid A \text{ is a DFA}, L(A) = \emptyset\}$ decidable?

$T =$ "On input $\langle A\rangle$:

0. reject if $\langle A\rangle$ does not encode some DFA.
1. mark the start state of A.
2. mark all states accessible from A.
3. repeat (2) until we can't find more accessible states.
4. accept if and only if we have marked no accept states; reject otherwise."

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

Idea 1. Try all the strings; reject if they behave differently. X

(*) (Clever) Idea 2. Facts. Given DFAs for A, B, we can construct DFAs for the following languages:

- $A \cup B$. (simulate both and accept if either accepts)
- $A \cap B$.

• $\overline{A}$ (swap accept/reject states)

Thus: Given DFAs A and B, we can build a DFA D such that

$$L(D) = \underbrace{\left(L(A) \cap \overline{L(B)}\right)}_{\text{in } A, \text{ not in } B} \cup \underbrace{\left(\overline{L(A)} \cap L(B)\right)}_{\text{in } L(B), \text{ not in } A.} \quad (*)$$

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

To decide EQ_{DFA}, define

step 0: input check.

F = "On input $\langle A, B \rangle$, where A, B are DFAs:
- Construct D as described.
- Run a TM that decides E_{DFA} on D.
- Accept/reject if our simulation accepts/rejects."

Bonus facts (see section 4.1 of Sipser.)

Fact 1. $A_{CFG} = \{\langle G, w \rangle \mid G \text{ is a CFG that generates } w\}$ is decidable.

Fact 2. $E_{CFG} = \{\langle G \rangle \mid G \text{ is a CFG}, L(G) = \emptyset\}$ decidable.

Fact 3. $EQ_{CFG} = \{\langle G, H \rangle \mid G, H \text{ are CFGs and } L(G) = L(H)\}$ is decidable.

Theorem. Every Context-Free language is decidable.

Proof. Let G be a CFG for A. Let S be a TM that decides A_{CFG}. Define M_G as follows:

M_G = "On input w:

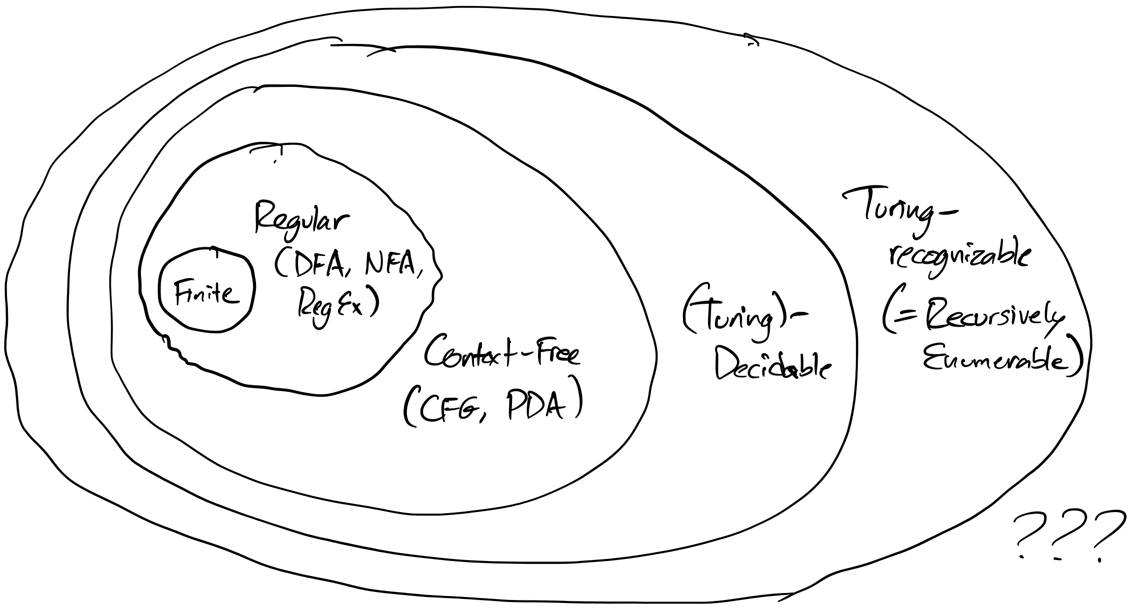
Run S on $\langle G, w \rangle$.

Accept/reject if S accepts/rejects."

G is finite
G is "hard-coded" into M_G.

∎

New picture of the universe:



Next: Countable & Uncountable Sets
Undecidable & Unrecognizable languages.