CS 341: Foundations of CS II

Marvin K. Nakayama
Computer Science Department
New Jersey Institute of Technology
Newark, NJ 07102

Chapter 4
Decidability

Contents

Decidable Languages

- We now tackle the question:

  What can and can't computers do?

- We consider the questions:

  Which languages are
  1. Turing-decidable
  2. Turing-recognizable
  3. neither?

- Assuming the Church-Turing thesis,
  - these are fundamental properties of languages and algorithms.
- Why study decidability?
  - Certain problems are unsolvable by computers.
  - You should be able to recognize these.

Describing TM Programs

- Three Levels of Describing Algorithms:
  - Formal (state diagrams, CFGs, etc.)
  - Implementation (pseudo-code)
  - High-level (coherent and clear English)
- Describing input/output format:
  - TMs allow only strings over some alphabet as input.
  - If our input X and Y are of another form (graph, TM, polynomial),
    - then we use $\langle X, Y \rangle$ to denote some kind of **encoding** as a string over some alphabet.
- When defining TM, **make sure to specify its input!**
- If TM M decides language L, then M
  - always gives correct answer (YES/NO, accept/reject)
  - never loops forever on any input.

Acceptance Problem for DFAs

Decision problem: Does a given DFA B accept a given string w?

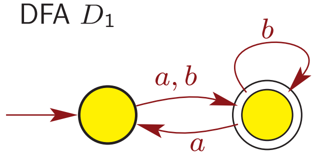
- **Instance** is a particular pair $\langle B, w\rangle$ of a DFA B and a string w.
- **Universe** comprises **every** possible instance

$$\Omega = \{ \langle B, w\rangle \mid B \text{ is a DFA and } w \text{ is a string} \}$$

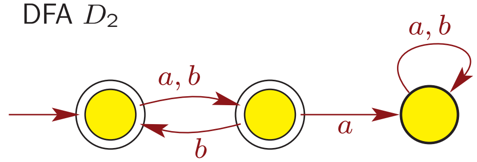
- **Language** comprises all YES instances

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

DFA D_1



DFA D_2



- $\langle D_1, abb\rangle \in A_{\text{DFA}}$ and $\langle D_2, \varepsilon\rangle \in A_{\text{DFA}}$ are YES instances.
- $\langle D_1, \varepsilon\rangle \notin A_{\text{DFA}}$ and $\langle D_2, aab\rangle \notin A_{\text{DFA}}$ are NO instances.

Acceptance Problem for DFAs is Decidable

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

Theorem 4.1
A_{DFA} is a decidable language.

Remarks:

- Recall universe for Acceptance Problem for DFAs

$$\Omega = \{ \langle B, w\rangle \mid B \text{ is a DFA and } w \text{ is a string} \}.$$

- To prove A_{DFA} is decidable, need to show $\exists$ TM M that decides A_{DFA}.
- For TM M to decide A_{DFA}, TM must
  - take any instance $\langle B, w\rangle \in \Omega$ as input
  - halt and **accept** if $\langle B, w\rangle \in A_{\text{DFA}}$
  - halt and **reject** if $\langle B, w\rangle \notin A_{\text{DFA}}$

Proof: TM M that Decides A_{DFA}

$M =$ "On input $\langle B, w\rangle \in \Omega$, where

- $B = (Q, \Sigma, \delta, q_0, F)$ is a DFA
- $w = w_1 w_2 \cdots w_n \in \Sigma^*$ is input string to process on B.

0. Check if $\langle B, w\rangle$ is 'proper' encoding. If not, *reject*.
1. Simulate B on w with the help of two pointers, q and i:
   - $q \in Q$ points to the current state of DFA B.
     - Initially, $q = q_0$, the start state of B.
   - $i \in \{1, 2, \ldots, |w|\}$ points to the current position in string w.
   - While i increases from 1 to $|w|$,
     - $q = \delta(q, w_i)$; i.e., transition function δ determines next state from current state q and input symbol w_i.
2. If B ends in state $q \in F$, then M *accepts*; otherwise, *reject*."

Acceptance Problem for NFAs is Decidable

Decision problem: Does a given NFA B accept a given string w?

$$A_{\text{NFA}} = \{ \langle B, w\rangle \mid B \text{ is NFA that } \textbf{accepts} \text{ string } w \}$$
$$\subseteq \{ \langle B, w\rangle \mid B \text{ is NFA, } w \text{ is string} \} \equiv \Omega$$

Theorem 4.2
A_{NFA} is a decidable language.

Proof. TM: "On input $\langle B, w\rangle \in \Omega$

- $B = (Q, \Sigma, \delta, q_0, F)$ is NFA
- $w \in \Sigma^*$ is input string for B.

0. If input $\langle B, w\rangle$ is not proper encoding of NFA B and string w, *reject*.
1. Use algo in Thm. 1.39 to transform NFA B into equivalent DFA C.
2. Run TM decider M for A_{DFA} (Theorem 4.1) on input $\langle C, w\rangle$.
3. If M accepts $\langle C, w\rangle$, *accept*; otherwise, *reject*."

Proof **reduces** A_{NFA} to A_{DFA}.

Acceptance Problem for Regular Expressions is Decidable

Decision problem: Does a reg exp R generate a given string w?

$$A_{\text{REX}} = \{ \langle R, w \rangle \mid R \text{ is regular expression that } \textbf{generates} \text{ string } w \}$$
$$\subseteq \{ \langle R, w \rangle \mid R \text{ is regular expression and } w \text{ is string} \} \equiv \Omega.$$

Example: For regular expressions $R_1 = a^*b$ and $R_2 = ba^*b^*$,

$$\langle R_1, aab \rangle \in A_{\text{REX}}, \quad \langle R_1, ba \rangle \notin A_{\text{REX}}, \quad \langle R_2, aab \rangle \notin A_{\text{REX}}.$$

Theorem 4.3
A_{REX} is a decidable language.

Proof. "On input $\langle R, w \rangle \in \Omega$:

0. Check if $\langle R, w \rangle$ is proper encoding of regular expression and string. If not, *reject*.
1. Convert R into DFA B using algos in Lemma 1.55 and Thm 1.39.
2. Run TM decider for A_{DFA} (Theorem 4.1) on input $\langle B, w \rangle$ and give same output."

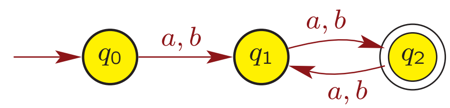
Proof **reduces** A_{REX} to A_{DFA}.

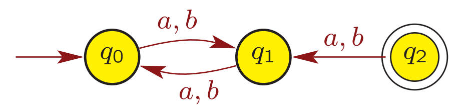
Emptiness Problem for DFAs

Decision problem: Does a DFA recognize the empty language?

$$E_{\text{DFA}} = \{ \langle B \rangle \mid B \text{ is a DFA and } L(B) = \emptyset \}$$
$$\subseteq \{ \langle B \rangle \mid B \text{ is a DFA} \} \equiv \Omega.$$

Examples: DFA C DFA D





Note that $\langle C \rangle \notin E_{\text{DFA}}$ and $\langle D \rangle \in E_{\text{DFA}}$.

Theorem 4.4
E_{DFA} is a decidable language.

Proof Idea:

- Check if any accept state is reachable from start state.
- If so, then *reject*; otherwise, *accept*.

Proof that E_{DFA} is Decidable

On input $\langle B \rangle \in \Omega$, where $B = (Q, \Sigma, \delta, q_0, F)$ is a DFA:

0. If $\langle B \rangle$ is not a proper encoding of a DFA, *reject*.
1. Define S as set of states reachable from q_0. Initially, $S = \{q_0\}$.
2. Repeat $|Q|$ times:
   (a) If S has an element from F, then *reject*.
   (b) Otherwise, add to S the elements that can be reached from S using transition function δ, i.e.,
   - If $\exists\ q_i \in S$ and $\ell \in \Sigma$ with $\delta(q_i, \ell) = q_j$, then add q_j to S.
3. If $S \cap F = \emptyset$, then *accept*; otherwise, *reject*.

Remark: TM just tests whether any accepting state is reachable from start state (**transitive closure**).

DFA Equivalence Problem is Decidable

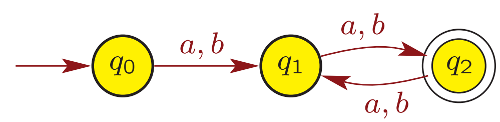
Decision problem: Are 2 given DFAs equivalent?

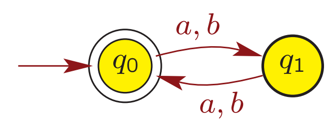
$$EQ_{\text{DFA}} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \}$$
$$\subseteq \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs} \} \equiv \Omega.$$

- For DFAs A and B with same input alphabet Σ, $\langle A, B \rangle \in EQ_{\text{DFA}}$ iff A and B agree on every string in Σ^*.

Example:

DFA A_1 DFA B_1



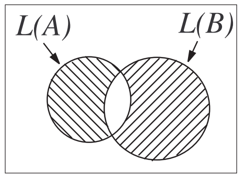


DFAs A_1 and B_1 don't recognize same language, so $\langle A_1, B_1 \rangle \notin EQ_{\text{DFA}}$.

Theorem 4.5
EQ_{DFA} is a decidable language.

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

- Given DFAs A and B, construct new DFA C such that C accepts any string accepted by A or B but not both:
$$L(C) = \left[L(A) \cap \overline{L(B)} \right] \cup \left[\overline{L(A)} \cap L(B) \right]$$
- $L(C)$ is the **symmetric difference** of $L(A)$ and $L(B)$.



- Note that $L(A) = L(B)$ if and only if $L(C) = \emptyset$.
- Construct DFA C using algorithms for DFA complements (slide 1-15), intersections (slide 1-34), and unions (Thm 1.25).
- DFA C can be constructed with one big TM.

Proof that EQ_{DFA} is Decidable

On input $\langle A, B \rangle \in \Omega$, where A and B are DFAs:

0. Check if $\langle A, B \rangle$ is a proper encoding of 2 DFAs. If not, *reject*.
1. Construct DFA C such that
$$L(C) = \left[L(A) \cap \overline{L(B)} \right] \cup \left[\overline{L(A)} \cap L(B) \right]$$
using algorithms for DFA complements (slide 1-15), intersections (slide 1-34), and unions (Thm 1.25).
2. Run TM decider for E_{DFA} (Theorem 4.4) on input $\langle C \rangle$.
3. If $\langle C \rangle \in E_{\text{DFA}}$, *accept*;
If $\langle C \rangle \notin E_{\text{DFA}}$, *reject*.

Acceptance, Emptiness and Equivalence Problems for CFGs

$$\begin{aligned} A_{\text{CFG}} &= \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}, \\ E_{\text{CFG}} &= \{ \langle G \rangle \mid G \text{ is a CFG with } L(G) = \emptyset \}, \\ EQ_{\text{CFG}} &= \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs with } L(G) = L(H) \}. \end{aligned}$$

Example:

- Consider CFGs
  - G_1 with rules $S \rightarrow aSb \mid \varepsilon$, so $L(G_1) = \{ a^k b^k \mid k \geq 0 \}$,
  - G_2 with rules $S \rightarrow aSb$, so $L(G_2) = \emptyset$.
- $\langle G_1, aabb \rangle \in A_{\text{CFG}}$, $\langle G_1, aab \rangle \notin A_{\text{CFG}}$, and $\langle G_2, aabb \rangle \notin A_{\text{CFG}}$, .
- $\langle G_1 \rangle \notin E_{\text{CFG}}$ and $\langle G_2 \rangle \in E_{\text{CFG}}$.
- $\langle G_1, G_2 \rangle \notin EQ_{\text{CFG}}$.

Acceptance Problem for CFGs is Decidable

- **Decision problem:** Does a CFG G generate a string w?
$$\begin{aligned} A_{\text{CFG}} &= \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \} \\ &\subseteq \{ \langle G, w \rangle \mid G \text{ is a CFG and } w \text{ a string} \} \equiv \Omega. \end{aligned}$$
- For any specific pair $\langle G, w \rangle \in \Omega$ of a CFG G and string w,
  - $\langle G, w \rangle \in A_{\text{CFG}}$ if G generates w, i.e., $w \in L(G)$.
  - $\langle G, w \rangle \notin A_{\text{CFG}}$ if G doesn't generate w, i.e., $w \notin L(G)$.

Theorem 4.7
A_{CFG} is a decidable language.

Bad Idea for Proof:

- Design a TM M that takes input $\langle G, w \rangle$, and enumerates all derivations using CFG G to see if any generates w.
- **Problem:** M might **recognize** A_{CFG} but does not **decide** it. Why?
  - If $w \notin L(G)$ and $|L(G)| = \infty$, then TM M never halts.

Better Approach: Use Chomsky Normal Form

- **Recall:** A context-free grammar $G = (V, \Sigma, R, S)$ is in **Chomsky normal form** if each rule is of the form

$$A \to BC \quad \text{or} \quad A \to x \quad \text{or} \quad S \to \varepsilon$$

  - variable $A \in V$
  - variables $B, C \in V - \{S\}$
  - terminal $x \in \Sigma$.
- Every CFG can be converted into Chomsky normal form (Theorem 2.9).
- CFG G in Chomsky normal form is easier to analyze.
  - Can show that for any string $w \in L(G)$ with $w \neq \varepsilon$, derivation $S \overset{*}{\Rightarrow} w$ takes exactly $2|w| - 1$ steps.
  - $\varepsilon \in L(G)$ iff G includes rule $S \to \varepsilon$.

Proof that A_{CFG} is Decidable

On input $\langle G, w \rangle \in \Omega$, where G is a CFG and w is a string,

0. Check if $\langle G, w \rangle$ is proper encoding of CFG and string; if not, *reject*.
1. Convert G into equivalent CFG G' in Chomsky normal form.
2. If $w = \varepsilon$, check if $S \to \varepsilon$ is a rule of G'.
   If so, *accept*; otherwise, *reject*.
3. If $w \neq \varepsilon$, list all derivations with $2n - 1$ steps, where $n = |w|$.
4. If any generates w, *accept*;
   otherwise, *reject*.

Remarks:

- # derivations with $2n - 1$ steps is finite, so TM is a decider.
- We consider a more efficient algorithm in Chapter 7.

Emptiness Problem for CFGs is Decidable

Decision problem: Is a CFG's language empty?

$$E_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG with } L(G) = \emptyset \}$$
$$\subseteq \{ \langle G \rangle \mid G \text{ is a CFG} \} \equiv \Omega$$

Theorem 4.8
E_{CFG} is decidable.

Proof. On input $\langle G \rangle \in \Omega$, where G is a CFG,

0. Check if $\langle G \rangle$ is a proper encoding of a CFG $G = (V, \Sigma, R, S)$;
   if not, *reject*.
1. Define set $T \subseteq V \cup \Sigma$ such that $u \in T$ iff $u \overset{*}{\Rightarrow} w$ for some $w \in \Sigma^*$.
   Initially, $T = \Sigma$, and iteratively add to T.
2. Repeat $|V|$ times:
   - Check each rule $B \to X_1 \cdots X_k$ in R.
   - If $B \notin T$ and each $X_i \in T$, then add B to T.
3. If $S \in T$, then *reject*; otherwise, *accept*.

Are Two CFGs Equivalent?

- **Decision problem:** Are two CFGs equivalent?

$$EQ_{\text{CFG}} = \{ \langle G, H \rangle \mid G, H \text{ are CFGs and } L(G) = L(H) \}$$
$$\subseteq \{ \langle G, H \rangle \mid G, H \text{ are CFGs} \} \equiv \Omega.$$

- For DFAs, used emptiness decision procedure to solve equality problem.
  - Try to construct CFG C from CFGs G and H such that

$$L(C) = \left[L(G) \cap \overline{L(H)} \right] \cup \left[\overline{L(G)} \cap L(H) \right]$$

    and check if $L(C)$ is empty using TM decider for E_{CFG}.
- We can't define CFG C for symmetric difference. Why?
  - Class of CFLs **not closed** under complementation nor intersection.
- **Fact:** EQ_{CFG} is **not** a decidable language.
  - We'll prove this later (HW 9).

CFLs are Decidable

Theorem 4.9
Every CFL L is a decidable language.

Bad Idea for Proof:

- Convert PDA for L directly into a TM.
  - Can do this by using TM tape to simulate PDA stack.
- Nondeterministic PDA yields nondeterministic TM (NTM).
- NTM can be converted into deterministic TM (DTM).
- Problem:
  - Some branch of PDA might run forever.
  - Some branch of NTM might run forever.
  - Corresponding DTM **recognizes** L,
    - but **does not decide** L since it may not halt on every input.

Proof that Every CFL L is Decidable

- Let L be a CFL with alphabet Σ, so $L \subseteq \Sigma^*$
  - G' be a CFG for language L
  - S be a TM from Theorem 4.7 that decides

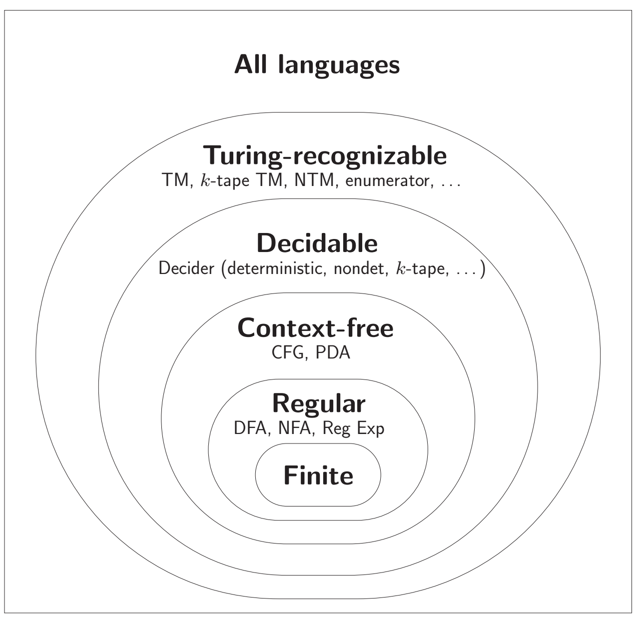
$$A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}$$

- Construct TM $M_{G'}$ for language L having CFG G' as follows:

  $M_{G'} =$ "On input $w \in \Sigma^*$:
  1. Run TM decider S on input $\langle G', w \rangle$.
  2. If S accepts, *accept*;
     otherwise, *reject*."

- How do TMs S and $M_{G'}$ differ?
  - TM S has input $\langle G, w \rangle$ for **any** CFG G and string w.
  - TM $M_{G'}$ has input w for **fixed** G'.

Hierarchy of Languages (so far)



Examples

???

???

$\{ 0^n 1^n 2^n \mid n \geq 0 \}$

$\{ 0^n 1^n \mid n \geq 0 \}$

$(0 \cup 1)^*$

$\{ 110, 01 \}$

The Universal TM U

- Is one TM capable of simulating all other TMs?
- Given an encoding $\langle M, w \rangle$ of a TM M and input w,
  - can we simulate M on w?
- We can do this via a **universal TM** U:

  $U =$ "On input $\langle M, w \rangle$, where M is a TM and w is a string:

  1. Simulate M on input w.

  2. If M ever enters its accept state, *accept*;
  if M ever enters its reject state, *reject*."

- Can think of U as an **emulator**.

Acceptance Problem for TMs is Turing-Recognizable

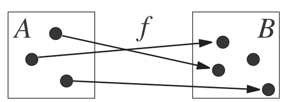
- **Decision problem:** Does a given TM M accept a given string w?
- **Instance:** $\langle M, w\rangle$, where M is TM, w is a string.
- **Universe:** $\Omega = \{ \langle M, w\rangle \mid M$ is TM and w is string $\}$.
- **Language:**
  $A_{\mathsf{TM}} = \{ \langle M, w\rangle \mid M$ is TM that **accepts** string $w \} \subseteq \Omega$.
- For a specific pair $\langle M, w\rangle \in \Omega$ of TM M and string w,
  - $\langle M, w\rangle \in A_{\mathsf{TM}}$ if M accepts w
  - $\langle M, w\rangle \notin A_{\mathsf{TM}}$ if M does not accept w.
- Universal TM U
  - U **recognizes** A_{TM}, so A_{TM} is Turing-recognizable.
  - U **does not decide** A_{TM}.
    - If M loops on w, then U loops on $\langle M, w\rangle$.
- But can we also decide A_{TM}?
  - We will see later that A_{TM} is **undecidable**.

Unsolvable Problems

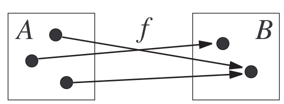
- Computers (and computation) are limited in a very fundamental way.
- Common, every-day problems are unsolvable (i.e., undecidable)
  - Does a program sort an array of integers?
  - Both program and specification are precise mathematical objects.
  - One might think that it is then possible to develop an algorithm that can determine if a program matches its specification.
  - However, this is impossible.
- To show this, we need to introduce some new ideas.

Mappings and Functions

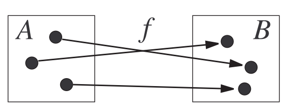
- Consider fcn $f : A \to B$ mapping objects in one set A to another B.
- **Definition:** f is **one-to-one** (aka **injective**) if every $x \in A$ has a unique image $f(x)$:
  - If $f(x) = f(y)$, then $x = y$.
  - Equivalently, if $x \neq y$, then $f(x) \neq f(y)$.



- **Definition:** f is **onto** (aka **surjective**) if every $z \in B$ is "hit" by f:
  - If $z \in B$, then there is an $x \in A$ with $f(x) = z$.



- **Definition:** f is a **correspondence** (aka **bijection**) if it both one-to-one and onto.
  - Inverse fcn $f^{-1} : B \to A$ then exists.
  - A way to pair elements from A with elements from B.



Example: $f : \mathcal{R} \to \mathcal{R}$ with $f(x) = e^x$ is

- *one-to-one* since $x \neq y$ implies $e^x \neq e^y$.
- *not onto* since $e^x > 0$ for all $x \in \mathcal{R}$.

Example: $f : \mathcal{R} \to \mathcal{R}$ with $f(x) = x^2$ is

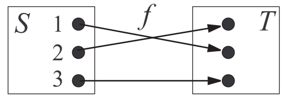
- *not one-to-one* since $3^2 = (-3)^2 = 9$.
- *not onto* since $x^2 \geq 0$ for all $x \in \mathcal{R}$.

Example: $f : \mathcal{R} \to \mathcal{R}$ with $f(x) = x^3$ is

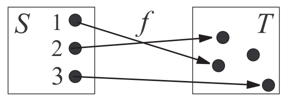
- *one-to-one* since $x \neq y$ implies $x^3 \neq y^3$.
- *onto* since for any $z \in \mathcal{R}$, letting $x = z^{1/3}$ yields $f(x) = (z^{1/3})^3 = z$.
- Thus, f is a *correspondence* between $A = \mathcal{R}$ and $B = \mathcal{R}$.

Cardinality

- Set T has $|T| = k$ iff $\exists$ *correspondence* between $\{1, 2, \ldots, k\}$ and T, in which case $\{1, 2, \ldots, k\}$ and T are of the **same size**.
  - **Ex:** $|T| = 3$.



- If $\exists$ *one-to-one* mapping from set S to set T, then T is **at least as big** as S, i.e., $|T| \geq |S|$.
  - **Ex:** $|T| \geq 3$.



- **Defn:** Two sets S and T, possibly infinite, are of the **same size** if there is a *correspondence* between them.
- If $\exists$ *one-to-one* fcn from S to T but $\nexists$ *correspondence* from S to T, then T is **strictly bigger** than S.

Countable Sets

- Let $\mathcal{N} = \{1, 2, 3, \ldots\}$ be the set of natural numbers.
- Set T is **infinite** if there exists a **one-to-one** function $f : \mathcal{N} \to T$.
  - "The set T is at least as big as the set $\mathcal{N}$."
- Set T is **countable** if it is finite or has the same size as $\mathcal{N}$.
  - Can *enumerate* all elements in T in (possibly infinite) list.
  - each element is eventually listed.

Fact: $\mathcal{N} = \{1, 2, 3, \ldots\}$ and $\mathcal{E} = \{2, 4, 6, \ldots\}$ have same size.

Proof. Define correspondence between $\mathcal{N}$ and $\mathcal{E}$ by function $f(i) = 2i$.

Remark: Set T and a proper subset of T can have the same size!

Set of Rational Numbers is Countable

Fact: The set of rational numbers

$$\mathcal{Q} = \left\{ \frac{m}{n} \;\middle|\; m, n \in \mathcal{N} \right\}$$

is countable.

Proof.

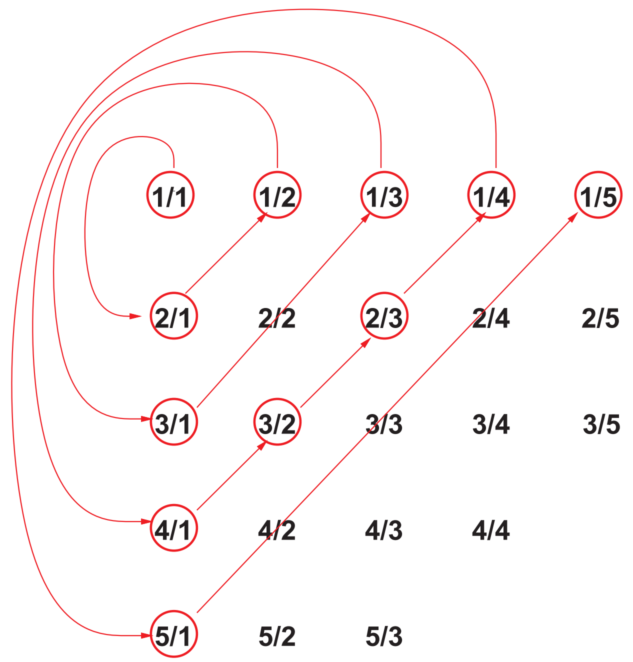
- Write out elements in $\mathcal{Q}$ as an infinite 2-dimensional array:

$$\begin{array}{cccccc}
1/1 & 1/2 & 1/3 & 1/4 & 1/5 & \cdots \\
2/1 & 2/2 & 2/3 & 2/4 & 2/5 & \cdots \\
3/1 & 3/2 & 3/3 & 3/4 & 3/5 & \cdots \\
4/1 & 4/2 & 4/3 & 4/4 & 4/5 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

- If we try to
  - first list all elements in first row,
  - then list all elements in second row,
  - and so on,

  then we will never get to the second row because the first row is infinitely long.

- Instead,
  - enumerate elements along Southwest to Northeast diagonals,
  - skip duplicates.



More Countable Sets

Examples: $\exists$ correspondence between $\mathcal{N} = \{1, 2, 3, \dots\}$ and each of

- $\mathcal{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$
- $\mathcal{N}^2 = \{\, (i, j) \mid i, j \in \mathcal{N} \,\}$
- $\{a\}^*$
- Σ^*, for any alphabet Σ; e.g., $\Sigma = \{a, b\}$.
  - Simply enumerate strings in Σ^* in *string order*.

$\mathcal{N}$	1	2	3	4	5	6	7	...
$\mathcal{Z}$	0	+1	−1	+2	−2	+3	−3	...
$\mathcal{N}^2$	(1, 1)	(2, 1)	(1, 2)	(3, 1)	(2, 2)	(1, 3)	(4, 1)	...
$\{a\}^*$	ε	a	aa	aaa	$aaaa$	$aaaaa$	$aaaaaa$	...
$\{a, b\}^*$	ε	a	b	aa	ab	ba	bb	...

So is every infinite set countable?

Uncountable Sets

Definition: A set is **uncountable** if it is not countable.

Remark: Uncountable sets are (much) larger than countable sets.

Definition: A **real number** is a number with a (possibly infinite) decimal representation.

- $\pi = 3.1415926\dots$
- $\sqrt{2} = 1.4142136\dots$
- $2 = 2.0000\dots$

Theorem 4.17
The set $\mathcal{R}$ of all real numbers is uncountable.

Set $\mathcal{R}$ of All Real Numbers is Uncountable

Proof.

- Suppose that there is a correspondence between $\mathcal{N}$ and $\mathcal{R}$:

n	$f(n)$
1	3.14159...
2	0.55555...
3	40.00000...
4	15.20361...
⋮	⋮

- Since correspondence exists, enumerated list is supposed to contain every real number.
- Each number is written as an infinite decimal expansion.
- We now construct a number x between 0 and 1 that is not in the list using Cantor's **diagonalization** method

Diagonalization Method

- Let $x = 0.\, d_1\, d_2\, d_3\, \ldots$, where
  - d_n is nth digit after decimal point in decimal expansion of x
  - d_n differs from the nth digit in the nth number in the list.

n	$f(n)$
1	$3.\underline{1}4159\ldots$
2	$0.5\underline{5}555\ldots$
3	$40.00\underline{0}00\ldots$
4	$15.203\underline{6}1\ldots$
$\vdots$	$\vdots$

- For example, can take $x = 0.2617\ldots$.
- $\forall\, n$, x differs from nth number $f(n)$ in the list in at least position n,
  - so x is not in the list,
  - contradiction since list is supposed to contain all of $\mathcal{R}$, including x.
- Thus, $\nexists$ correspondence $f : \mathcal{N} \to \mathcal{R}$, so $\mathcal{R}$ is uncountable.

Set of All TMs is Countable

Fact: If $S \subseteq T$ and T is countable, then S is countable.

Proof. In enumeration of T, skip elements in $T - S$ to enumerate S.

Fact: For any (finite) alphabet Ψ, the set Ψ^* is countable.

Proof. Enumerate strings in string order.

Fact: The set of all TMs is countable.

Proof.

- Every TM has a finite description, e.g., as 7-tuple or source code.
  - Can describe TM M using encoding $\langle M \rangle$
  - Encoding is a finite string of symbols over some alphabet Ψ.
- So just enumerate all strings over Ψ
  - omit any that are not legal TM encodings.
- Since Ψ^* is countable,
  - there are only a countable number of different TMs.

Set of All Languages is Uncountable

Fact: The set $\mathcal{B}$ of all *infinite* binary sequences is uncountable.

Proof. Use diagonalization argument as in proof that $\mathcal{R}$ is uncountable.

Fact: The set $\mathcal{L}$ of all languages over alphabet Σ is uncountable.

Proof.

- **Idea:** show $\exists$ *correspondence* χ between $\mathcal{L}$ and $\mathcal{B}$, so $\mathcal{L}$ has same size as uncountable set $\mathcal{B}$.
- Each language $A \in \mathcal{L}$ has $A \subseteq \Sigma^*$, so $\mathcal{L} = \mathcal{P}(\Sigma^*)$.
- Language's **characteristic sequence** defined by correspondence

$$\chi : \mathcal{L} \to \mathcal{B}$$

  - Write out elements in Σ^* in string order: $s_1, s_2, s_3, \ldots$.
  - Each language $A \in \mathcal{L}$ has a unique sequence $\chi(A) \in \mathcal{B}$.
  - The nth bit of $\chi(A)$ is 1 if and only if $s_n \in A$

- **Recall:** Each language $A \in \mathcal{L}$ has a unique sequence $\chi(A) \in \mathcal{B}$
  - nth bit of $\chi(A)$ is 1 if and only if $s_n \in A$.
  - $\chi(A)$ specifies which strings from Σ^* are or aren't in A.
- **Example:** For $\Sigma = \{0, 1\}$,

$$\begin{array}{rcccccccccccc} \Sigma^* & = & \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & \ldots & \} \\ A & = & \{ & & 0, & & 00, & 01, & & & 000, & \ldots & \} \\ \chi(A) & = & & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & \ldots & \end{array}$$

- The mapping $\chi : \mathcal{L} \to \mathcal{B}$ is a **correspondence** because it is
  - **one-to-one**: different languages A_1 and A_2 differ for at least one string s_i, so the ith bits of $\chi(A_1)$ and $\chi(A_2)$ differ;
  - **onto**: for each sequence $b \in \mathcal{B}$, $\exists$ language A for which $\chi(A) = b$.
- Thus, $\mathcal{L}$ is **same size** as uncountable set $\mathcal{B}$,
  - so $\mathcal{L}$ is also uncountable.

Some Languages are not Turing-Recognizable

- Each TM recognizes some language.
- Set of all TMs is countable.
- Set of all languages is uncountable.
- Since uncountable sets are larger than countable ones,
  - $\exists$ more languages than there are TMs that can recognize them.

Corollary 4.18
Some languages are not Turing-recognizable.

- What kind of languages are not Turing-recognizable?
  - We'll see some later ...

Revisit Acceptance Problem for TMs

- **Decision problem:** Does a TM M accept string w?

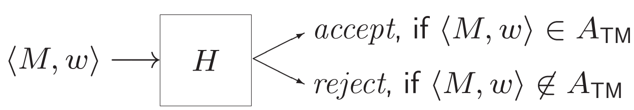
$$\begin{aligned} A_{\text{TM}} &= \{ \langle M, w \rangle \mid M \text{ is a TM that } \textbf{accepts} \text{ string } w \} \\ &\subseteq \{ \langle M, w \rangle \mid M \text{ is a TM and } w \text{ is a string} \} \equiv \Omega \end{aligned}$$

- Universe Ω of instances
  - contains all possible pairs $\langle M, w \rangle$ of TM M and string w
  - not just one specific instance.
- For a specific TM M and string w,
  - if M accepts w, then $\langle M, w \rangle \in A_{\text{TM}}$ is a YES instance
  - if M doesn't accept w (rejects or loops), then $\langle M, w \rangle \notin A_{\text{TM}}$ is a NO instance.

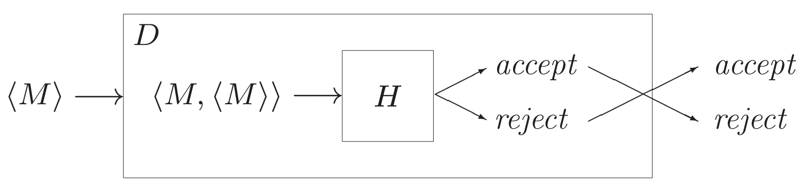
Theorem 4.11
A_{TM} is undecidable.

Outline of Proof by Contradiction

- Suppose A_{TM} is decided by some TM H, with input $\langle M, w \rangle \in \Omega$.



- Use H as subroutine to define another TM D, with input $\langle M \rangle$.



- What happens when we run D with input $\langle D \rangle$?
  - D accepts $\langle D \rangle$ iff D doesn't accept $\langle D \rangle$, which is impossible.

Proof by Contradiction that A_{TM} is Undecidable

- Suppose there exists a TM H that decides A_{TM}.
  - TM H takes input $\langle M, w \rangle \in \Omega$, where M is a TM and w a string.
  - H accepts $\langle M, w \rangle \in A_{\text{TM}}$; i.e., if M accepts w.
  - H rejects $\langle M, w \rangle \notin A_{\text{TM}}$; i.e., if M does not accept w.
- Consider language $L = \{ \langle M \rangle \mid M \text{ is TM that doesn't accept } \langle M \rangle \}$.
- Using TM H as subroutine, we can construct TM D that decides L:

  $D =$ "On input $\langle M \rangle$, where M is a TM:
  1. Run H on input $\langle M, \langle M \rangle \rangle$.
  2. If H accepts, *reject*. If H rejects, *accept*."

- What happens when we run D with input $\langle D \rangle$?
  - Stage 1 of D runs H on input $\langle D, \langle D \rangle \rangle$.
  - D accepts $\langle D \rangle$ iff D doesn't accept $\langle D \rangle$, which is impossible.
- So TM H must not exist, i.e., A_{TM} is undecidable.

Another View of Proof

Remark: The proof implicitly used diagonalization ...

- Since the set of all TMs is countable, we can enumerate them:

$$M_1,\ M_2,\ M_3,\ M_4,\ \ldots$$

- Construct table of acceptance behavior of TM M_i on input $\langle M_j \rangle$:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	accept		accept		$\cdots$
M_2	accept	accept	accept	accept	$\cdots$
M_3					$\cdots$
M_4	accept	accept			$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

  - Blank entries are reject or loop.

Another View of Proof

- Another table
  - entry (i, j) is value of "acceptance function" H on input $\langle\, M_i,\, \langle M_j \rangle\, \rangle$:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	accept	reject	accept	reject	$\cdots$
M_2	accept	accept	accept	accept	$\cdots$
M_3	reject	reject	reject	reject	$\cdots$
M_4	accept	accept	reject	reject	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Another View of Proof

- Diagonal entries swapped for output of D on $\langle M_i \rangle$.
- D is a TM, so it must appear in the enumeration $M_1, M_2, M_3, \ldots$
- Contradiction occurs when evaluating D on $\langle D \rangle$:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$	$\langle D \rangle$	
M_1	<u>accept</u>	reject	accept	reject	$\cdots$	accept	$\cdots$
M_2	accept	<u>accept</u>	accept	accept	$\cdots$	accept	$\cdots$
M_3	reject	reject	<u>reject</u>	reject	$\cdots$	reject	$\cdots$
M_4	accept	accept	reject	<u>reject</u>	$\cdots$	accept	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$		
D	reject	reject	accept	accept	$\cdots$	<u>?</u>	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$		$\ddots$

Another View of the Problem

- "Self-referential paradox"
  - occurs when we force the TM D to disagree with itself.
- D knows what it is going to do on input $\langle D \rangle$ by H,
  - but then D does the opposite instead.
- You cannot know for sure what you will do in the future.
  - If you could, then you could change your actions and create a paradox.
- The diagonalization method implements the self-reference paradox in a mathematical way.
- In logic this approach often used to prove that certain things are impossible.
- Kurt Gödel gave a mathematical equivalent of the statement "This sentence is not true" or "I am lying."

Co-Turing-Recognizable Languages

$$A_{\mathsf{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts string } w \}$$

- A_{TM} is **not Turing-decidable**, but is **Turing-recognizable**.
  - Use universal TM U to simulate TM M on string w.
    - If M accepts w, then U *accepts* $\langle M, w \rangle \in A_{\mathsf{TM}}$.
    - If M rejects w, then U *rejects* $\langle M, w \rangle \notin A_{\mathsf{TM}}$.
    - If M loops on w, then U *loops* on $\langle M, w \rangle \notin A_{\mathsf{TM}}$.
- What about a language that is **not** Turing-recognizable?
- Recall that complement of language A over alphabet Σ is

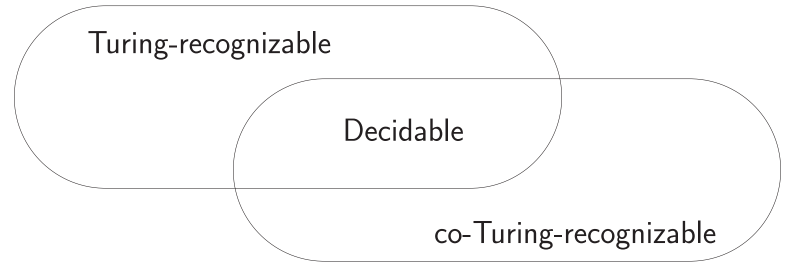
$$\overline{A} \;=\; \Sigma^* - A \;=\; \Omega - A$$

Definition: Language A is **co-Turing-recognizable** if its complement $\overline{A}$ is Turing-recognizable.

Decidable $\iff$ Turing- and co-Turing-recognizable

Theorem 4.22
A language is decidable if and only if it is both

- Turing-recognizable and
- co-Turing-recognizable.



Decidable $\Rightarrow$ TM-recognizable and co-TM-recognizable

- Suppose language A is **decidable**.
- Then A is **Turing-recognizable**.
- Also, since A is decidable, $\exists$ TM M that
  - always halts
  - correctly *accepts* strings $w \in A$
  - correctly *rejects* strings $w \notin A$
- Define TM M' same as M except swap *accept* and *reject* states.
  - M' *rejects* when M accepts,
  - M' *accepts* when M rejects.
- TM M' always halts since M always halts, so M' decides $\overline{A}$.
  - Thus, $\overline{A}$ is also Turing-recognizable
  - i.e., A is **co-Turing-recognizable**.

TM-recognizable and co-TM-recognizable $\Rightarrow$ Decidable

- Suppose A is both **TM-recognizable** and **co-TM-recognizable**.
- Then there exists
  - TM M recognizing A
  - TM M' recognizing $\overline{A}$.
- For any string $w \in \Sigma^*$, either $w \in A$ or $w \notin A$ (but not both), so either M or M' accepts w (but not both).
- Construct another TM D from M and M' as follows:

  $D =$ "On input $w \in \Sigma^*$:
  1. Alternate running one step on each of M and M' both on input w. Wait for M or M' to accept.
  2. If M accepts, *accept*; if M' accepts, *reject*."
- Note that D **decides** A, so A is **decidable**.

$\overline{A_{\text{TM}}}$ is not Turing-recognizable

Remarks:

- $A_{\text{TM}} = \{ \langle M, w \rangle \mid M$ is a TM that accepts string $w \}$ is **Turing-recognizable** (by UTM) but **not decidable** (Thm 4.11).
- Theorem 4.22: Decidable $\Leftrightarrow$ Turing-recog and co-Turing-recognizable.
- $\overline{A_{\text{TM}}} = \{ \langle M, w \rangle \mid M$ is a TM that does **not** accept string $w \}$.

Corollary 4.23
$\overline{A_{\text{TM}}}$ is not Turing-recognizable.

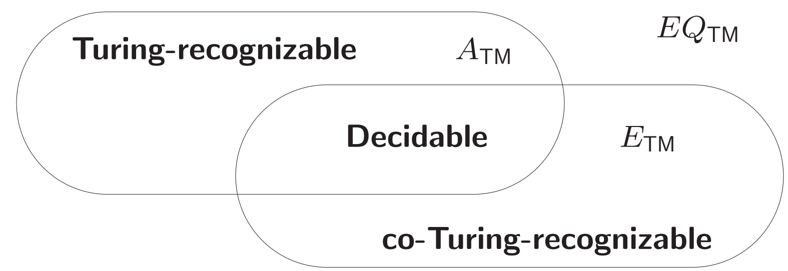
Proof.

- If $\overline{A_{\text{TM}}}$ were Turing-recognizable, then A_{TM} would be both **Turing-recognizable** and **co-Turing-recognizable**.
- But then Theorem 4.22 would imply A_{TM} is **decidable**, which is a **contradiction**.

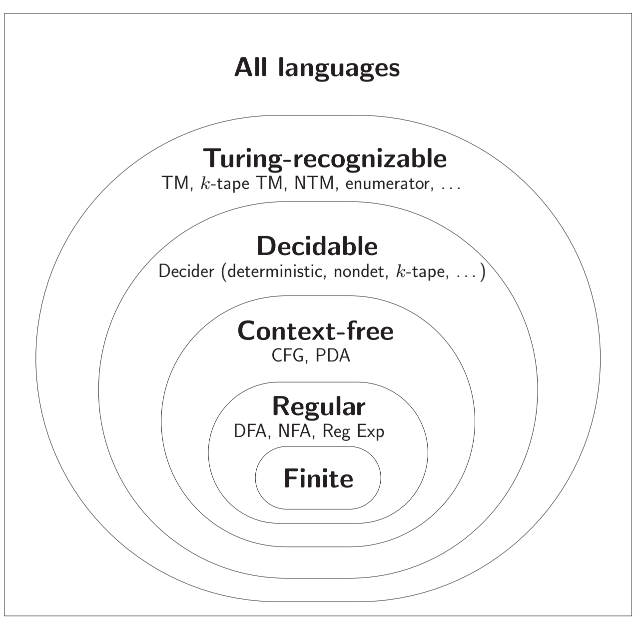
Some Other Non-Turing-Recognizable Languages

We'll later show the following languages are also not Turing-recognizable:

- $E_{\text{TM}} = \{ \langle M \rangle \mid M$ is a TM with $L(M) = \emptyset \}$, which is co-Turing-recognizable.
- $EQ_{\text{TM}} = \{ \langle M, N \rangle \mid M$ and N are TMs with $L(M) = L(N) \}$, which is not even co-Turing-recognizable.



Hierarchy of Languages



Examples

$\overline{A_{\text{TM}}}$

A_{TM}

$\{ 0^n 1^n 2^n \mid n \geq 0 \}$

$\{ 0^n 1^n \mid n \geq 0 \}$

$(0 \cup 1)^*$

$\{ 110, 01 \}$

Summary of Chapter 4

- Decidable languages: A_{DFA}, A_{NFA}, A_{REX}, E_{DFA}, EQ_{DFA}, A_{CFG}, E_{CFG}, CFL
- Universal TM (UTM): can simulate any given TM on given string
- A_{TM} (acceptance problem for TM) is Turing-recognizable but undecidable.
- Countable and uncountable sets
  - Diagonalization method used to prove certain sets are uncountable
  - Set of all TMs is countable
  - Set of all languages is uncountable
  - So some languages not Turing-recognizable, e.g., $\overline{A_{\text{TM}}}$.
- Language is co-Turing-recognizable if its complement is Turing-recognizable.
- Decidable $\Longleftrightarrow$ Turing-recognizable and co-Turing-recognizable.