# T065001: Introduction to Formal Languages

## Lecture 11: Decidability

*Chapter 4 in Sipser's textbook*

2025-06-30

(Lecture slides by Yih-Kuen Tsay)

# Decidability/Solvability

- We shall demonstrate certain problems that can be solved algorithmically and others that cannot.
- Our objective is to explore the limits of algorithmic solvability.
- Why should you study unsolvability?
  - Knowing when a problem is algorithmically unsolvable is useful because then you realize that the problem must be simplified or altered before you can find an algorithmic solution.
  - A glimpse of the unsolvable can stimulate your imagination and help you gain an important perspective on computation.

First, some examples of **solvable** problems.
In terms of Turing machines: **decidable** languages.

# (From Chapter 3.1)

## Definition (3.5)

A language is **Turing-recognizable** (also called *recursively enumerable*) if some Turing machine recognizes it.

- A Turing machine can fail to accept an input by entering the $q_{\mathrm{reject}}$ state and rejecting, or by looping (not halting).
- A machine is called a *decider* if it halts on all inputs. A decider that recognizes some language is said to *decide* the language.

## Definition (3.6)

A language is **Turing-decidable**, or simply **decidable** (also called *recursive*), if some Turing machine decides it.

An example:

- For example, $B = \{w\#w \mid w \in \{0,1\}^*\}$; that is whether the string comprises two identical strings separated by a $\#$ symbol.

Another example:

- Let $A$ be the language consisting of all strings representing undirected graphs that are connected.

$$A = \{< G >\mid G \text{ is a connected undirected graph}\}.$$

Both of these languages are decidable; see Lectures 9 and 10, respectively.

# Decidable Languages/Problems

- $A_{\mathrm{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts } w\}$.
- This is the *acceptance problem* (membership problem) for DFAs formulated as a language.

# Decidable Languages/Problems

- $A_{\mathrm{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts } w\}$.
- This is the *acceptance problem* (membership problem) for DFAs formulated as a language.

### Theorem (4.1)

*$A_{\mathrm{DFA}}$ is a decidable language.*

# Decidable Languages/Problems

- $A_{\text{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts } w\}$.
- This is the *acceptance problem* (membership problem) for DFAs formulated as a language.

**Theorem (4.1)**

*$A_{\text{DFA}}$ is a decidable language.*

- $M =$ "On input $\langle B, w\rangle$, where $B$ is a DFA and $w$ is a string:
  1. Simulate $B$ on input $w$.
  2. If the simulation ends in an accept state, *accept*; otherwise, *reject*."

The input to the TM $M$ is a representation of a DFA $B$ and a string $w$. $M$ uses **three tapes**. Tape 1 stores $(Q, \Sigma, \delta, q_0, F)$ for $B$; tape 2 stores $w$; tape 3 keeps track of $B$'s state during the simulation (initially, $q_0$).

According to Theorem 3.13, $M$ has an equivalent single-tape TM.

# Decidable Languages/Problems (cont.)

- $A_{\mathrm{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts } w\}$.

# Decidable Languages/Problems (cont.)

- $A_{\text{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts } w\}$.

**Theorem (4.2)**

*$A_{\text{NFA}}$ is a decidable language.*

- $N =$ "On input $\langle B, w\rangle$, where $B$ is an NFA and $w$ is a string:
  1. Convert NFA $B$ to an equivalent DFA $C$.
  2. Run TM $M$ for deciding $A_{\text{DFA}}$ (as a "procedure") on input $\langle C, w\rangle$.
  3. If $M$ accepts, *accept*; otherwise, *reject*."

In step 1, use the construction from the proof of Theorem 1.39.
In step 2, use $M$ from the proof of Theorem 4.1.

# Decidable Languages/Problems (cont.)

- $A_{\mathrm{REX}} = \{\langle R, w\rangle \mid R$ is a regular expression that represents $w\}$.

# Decidable Languages/Problems (cont.)

- $A_{\mathrm{REX}} = \{\langle R, w \rangle \mid R \text{ is a regular expression that represents } w\}$.

**Theorem (4.3)**

*$A_{\mathrm{REX}}$ is a decidable language.*

- $P =$ "On input $\langle R, w \rangle$, where $R$ is a regular expression and $w$ is a string:
  1. Convert regular expression $R$ to an equivalent DFA $A$.
  2. Run TM $M$ for deciding $A_{\mathrm{DFA}}$ on input $\langle A, w \rangle$.
  3. If $M$ accepts, *accept*; otherwise, *reject*."

In step 1, use the constructions from the proofs of Lemma 1.55 and Theorem 1.39.
In step 2, use $M$ from the proof of Theorem 4.1.

# Decidable Languages/Problems (cont.)

- $E_{\mathrm{DFA}} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$.

# Decidable Languages/Problems (cont.)

- $E_{\mathrm{DFA}} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$.

**Theorem (4.4)**

*$E_{\mathrm{DFA}}$ is a decidable language.*

- $T =$ "On input $\langle A \rangle$, where $A$ is a DFA:
  1. Mark the start state of $A$.
  2. Repeat Step 3 until no new states get marked.
  3. Mark any state that has a transition coming into it from any state that is already marked.
  4. If no accept state is marked, *accept*; otherwise, *reject*."

# Decidable Languages/Problems (cont.)

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

# Decidable Languages/Problems (cont.)

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

**Theorem (4.5)**

*$EQ_{\mathrm{DFA}}$ is a decidable language.*

- $F =$ "On input $\langle A, B \rangle$, where $A$ and $B$ are DFAs:
  1. Construct DFA $C$ such that

     ?

# Decidable Languages/Problems (cont.)

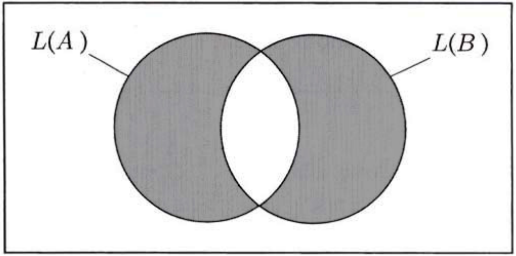



**FIGURE 4.6**
The symmetric difference of $L(A)$ and $L(B)$

# Decidable Languages/Problems (cont.)

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

**Theorem (4.5)**

*$EQ_{\mathrm{DFA}}$ is a decidable language.*

- $F =$ "On input $\langle A, B \rangle$, where $A$ and $B$ are DFAs:
  1. Construct DFA $C$ such that
     $L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$, i.e.,
     $C$ accepts strings that are accepted by one of $A$ and $B$ only.
  2. Run TM $T$ for deciding $E_{\mathrm{DFA}}$ on input $\langle C \rangle$.
  3. If $T$ accepts, *accept*; otherwise, *reject*."

In step 2, use $T$ from the proof of Theorem 4.4.

## Decidable CFL Properties

- $E_{\mathrm{CFG}} = \{\langle G \rangle \mid G$ is a CFG and $L(G) = \emptyset\}$.

# Decidable CFL Properties

- $E_{\mathrm{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$.

**Theorem (4.8)**

*$E_{\mathrm{CFG}}$ is a decidable language.*

- $R =$ "On input $\langle G\rangle$, where $G$ is a CFG:
  1. Mark all terminals in $G$.
  2. Repeat Step 3 until no new variables get marked.
  3. Mark any variable $A$ where $A \to U_1 U_2 \cdots U_k$ is a rule in $G$ and each symbol $U_1, U_2, \cdots, U_k$ has already been marked.
  4. If the start symbol is not marked, *accept*; otherwise, *reject*."

# Decidable CFL Properties (cont.)

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

## (From Chapter 2.1)

- When working with context-free grammars, it is often convenient to have them in simplified form.

### Definition (2.8)

A context-free grammar is in **Chomsky normal form** if every rule is of the form

$$\begin{array}{lcl} A & \rightarrow & BC \quad \text{or} \\ A & \rightarrow & a \end{array}$$

where $a$ is any terminal and $A$, $B$, and $C$ are any variables, except that $B$ and $C$ cannot be the start variable.

In addition,

$$S \rightarrow \varepsilon$$

is permitted if $S$ is the start variable.

# Decidable CFL Properties (cont.)

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

**Theorem (4.7)**

*$A_{\mathrm{CFG}}$ is a decidable language.*

- $S =$ "On input $\langle G, w\rangle$, where $G$ is a CFG and $w$ is a string:
  1. Convert $G$ to an equivalent grammar in Chomsky normal form.
  2. List all derivations with $2|w| - 1$ steps.
  3. If any of these derivations generate $w$, *accept*; otherwise, *reject*."

In step 1, use the construction from the proof of Theorem 2.9.
In step 2, we know from Exercise 2.26 that a derivation of $w$ (if one exists) will always use exactly $2|w| - 1$ steps.

## Decidability of CFLs

Theorem 4.7 ("$A_{\mathsf{CFG}}$ is a decidable language.") has the following important consequence:

### Theorem (4.9)

*Every context-free language is decidable.*

- Let $L$ be any context-free language. By definition, there exists a context-free grammar $G$ with $L(G) = L$.
  Then the following Turing machine $M_G$ decides $L$:
- $M_G =$ "On input $w$:
  1. Run TM $S$ for deciding $A_{\mathrm{CFG}}$ on input $\langle G, w \rangle$.
  2. If $S$ accepts, *accept*; otherwise, *reject*."

In step 1, use $S$ from the proof of Theorem 4.7.

# Classes of Languages

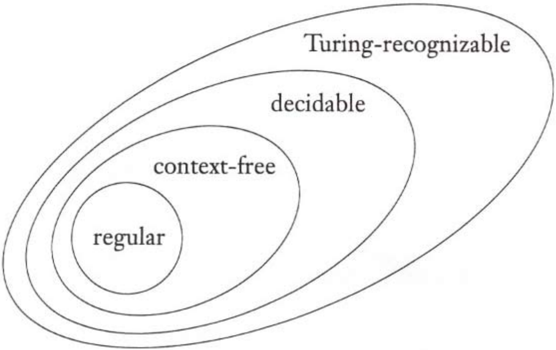



FIGURE **4.10**
The relationship among classes of languages

# Classes of Languages (cont.)

| Chomsky Hierarchy | Grammar | Language | Computation Model |
|---|---|---|---|
| Type-0 | Unrestricted | R.E. | Turing Machine |
| N/A | (no common name) | Recursive | Decider |
| Type-1 | Context-Sensitive | Context-Sensitive | Linear Bounded |
| Type-2 | Context-Free | Context-Free | Pushdown |
| Type-3 | Regular | Regular | Finite |

- Recall that Recursively Enumerable (R.E.) $\equiv$ Turing-recognizable and Recursive $\equiv$ Decidable (Turing-decidable).

**Remark 1:** A context-sensitive grammar is defined like a context-free grammar, but may also have substitution rules of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$.

**Remark 2:** A linear bounded automaton is a restricted type of Turing machine with a limited amount of memory; more precisely, the tape head is not allowed to move off the portion of the tape containing the input.

# Undecidability

We will first prove that undecidable problems do indeed exist by using a technique called diagonalization, invented by mathematician Georg Cantor in 1873.

Cantor was interested in the problem of measuring the sizes of **infinite sets**.

- If we have two infinite sets, how can we tell whether one is larger than the other or whether they are of the same size?
- For finite sets, of course, answering these questions is easy. We simply count the elements in a finite set, and the resulting number is its size.
- But if we try to count the elements of an infinite set, we will never finish!
- For example, take the set of even integers and the set of all strings over $\{0, 1\}$. Both sets are infinite and thus larger than any finite set, but is one of the two larger than the other?

# Countable vs. Uncountable Sets

## Definition (4.12)

Let $f$ be a function from $A$ to $B$.

- We say that $f$ is *one-to-one* if $f(a) \neq f(b)$ whenever $a \neq b$.
- Say that $f$ is *onto* if, for every $b \in B$, there is an $a \in A$ such that $f(a) = b$.
- A function that is both one-to-one and onto is called a *correspondence*.
- Two sets are considered to have the same size if there is a correspondence between them.

## Definition (4.14)

A set $A$ is **countable** if either it is finite or it has the same size as $\mathcal{N} = \{1, 2, 3, \cdots\}$; it is **uncountable**, otherwise.

# Countable vs. Uncountable Sets (cont.)

**EXAMPLE 4.13**

Let $\mathcal{N}$ be the set of natural numbers $\{1, 2, 3, \ldots\}$ and let $\mathcal{E}$ be the set of even natural numbers $\{2, 4, 6, \ldots\}$. Using Cantor's definition of size, we can see that $\mathcal{N}$ and $\mathcal{E}$ have the same size. The correspondence $f$ mapping $\mathcal{N}$ to $\mathcal{E}$ is simply $f(n) = 2n$. We can visualize $f$ more easily with the help of a table.

| $n$ | $f(n)$ |
|---|---|
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| $\vdots$ | $\vdots$ |

Of course, this example seems bizarre. Intuitively, $\mathcal{E}$ seems smaller than $\mathcal{N}$ because $\mathcal{E}$ is a proper subset of $\mathcal{N}$. But pairing each member of $\mathcal{N}$ with its own member of $\mathcal{E}$ is possible, so we declare these two sets to be the same size.

# Countable vs. Uncountable Sets (cont.)

**Example 4.15:**

Let $\mathcal{Q} = \{\frac{m}{n} \mid m, n \in \mathcal{N}\} =$ the set of positive rational numbers.
$\mathcal{Q}$ is countable because there exists a correspondence between $\mathcal{N}$ and $\mathcal{Q}$:

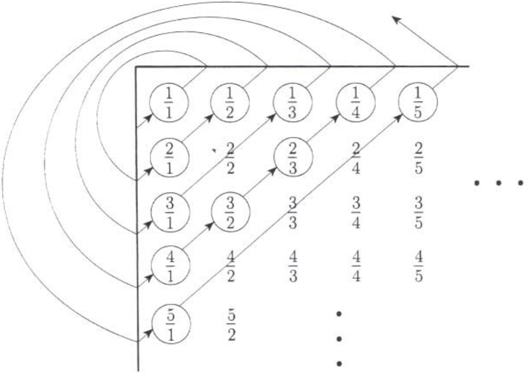



FIGURE 4.16
A correspondence of $\mathcal{N}$ and $\mathcal{Q}$

## Uncountable Sets

- A real number is one that has a (possibly infinite) decimal representation.
- Let $\mathcal{R}$ be the set of real numbers.

**Theorem (4.17)**

*$\mathcal{R}$ is uncountable.*

# Uncountable Sets (cont.)

- Assume that a correspondence $f$ existed between $\mathcal{N}$ and $\mathcal{R}$.

| $n$ | $f(n)$ |
|---|---|
| 1 | $3.\underline{1}4159\cdots$ |
| 2 | $55.5\underline{5}555\cdots$ |
| 3 | $0.12\underline{3}45\cdots$ |
| 4 | $0.500\underline{0}0\cdots$ |
| $\vdots$ | $\vdots$ |

- We can find an $x$, $0 < x < 1$, so that the $i$-th digit following the decimal point of $x$ is different from that of $f(i)$; for example, $x = 0.4641\cdots$ is a possible choice.
  But then there is no $n$ such that $x = f(n)$ holds, i.e., $x$ is not anywhere in the list. Contradiction. Therefore, $\mathcal{R}$ is uncountable.

# Uncountable Sets (cont.)

- Assume that a correspondence $f$ existed between $\mathcal{N}$ and $\mathcal{R}$.

| $n$ | $f(n)$ |
|---|---|
| 1 | $3.\underline{1}4159\cdots$ |
| 2 | $55.5\underline{5}555\cdots$ |
| 3 | $0.12\underline{3}45\cdots$ |
| 4 | $0.500\underline{0}0\cdots$ |
| $\vdots$ | $\vdots$ |

- We can find an $x$, $0 < x < 1$, so that the $i$-th digit following the decimal point of $x$ is different from that of $f(i)$; for example, $x = 0.4641\cdots$ is a possible choice.
  But then there is no $n$ such that $x = f(n)$ holds, i.e., $x$ is not anywhere in the list. Contradiction. Therefore, $\mathcal{R}$ is uncountable.
- This proof technique is called *diagonalization*, discovered by Georg Cantor in 1873.

**Exercise:**

Let $B$ be the set of all infinite binary sequences. Prove that $B$ is uncountable.

**Exercise:**
Let $B$ be the set of all infinite binary sequences. Prove that $B$ is uncountable.

**(Proof by contradiction)**

Suppose that $B$ is countable. Then there exists a correspondence $f : \mathcal{N} \rightarrow B$.
Use $f$ to define an infinite binary sequence $X = x_1, x_2, x_3, \ldots$ as follows.

**Exercise:**
Let $B$ be the set of all infinite binary sequences. Prove that $B$ is uncountable.

**(Proof by contradiction)**

Suppose that $B$ is countable. Then there exists a correspondence $f : \mathcal{N} \rightarrow B$. Use $f$ to define an infinite binary sequence $X = x_1, x_2, x_3, \ldots$ as follows.

For every positive integer $i$:

- If the $i$th bit in the sequence $f(i)$ is 0 then let $\boxed{x_i = 1}$.
- If the $i$th bit in the sequence $f(i)$ is 1 then let $\boxed{x_i = 0}$.

**Exercise:**
Let $B$ be the set of all infinite binary sequences. Prove that $B$ is uncountable.

**(Proof by contradiction)**

Suppose that $B$ is countable. Then there exists a correspondence $f : \mathcal{N} \to B$.
Use $f$ to define an infinite binary sequence $X = x_1, x_2, x_3, \ldots$ as follows.

For every positive integer $i$:

- If the $i$th bit in the sequence $f(i)$ is 0 then let $\boxed{x_i = 1}$.
- If the $i$th bit in the sequence $f(i)$ is 1 then let $\boxed{x_i = 0}$.

We get a sequence $X$ that cannot be equal to $f(n)$ for any positive integer $n$ (because the $n$th bit of $X$ is the opposite of the $n$th bit of $f(n)$).

**Exercise:**
Let $B$ be the set of all infinite binary sequences. Prove that $B$ is uncountable.

**(Proof by contradiction)**

Suppose that $B$ is countable. Then there exists a correspondence $f : \mathcal{N} \to B$.
Use $f$ to define an infinite binary sequence $X = x_1, x_2, x_3, \ldots$ as follows.

For every positive integer $i$:

- If the $i$th bit in the sequence $f(i)$ is 0 then let $\boxed{x_i = 1}$.
- If the $i$th bit in the sequence $f(i)$ is 1 then let $\boxed{x_i = 0}$.

We get a sequence $X$ that cannot be equal to $f(n)$ for any positive integer $n$ (because the $n$th bit of $X$ is the opposite of the $n$th bit of $f(n)$).

This means there is no integer $n$ such that $f(n) = X$, so $f$ is not a correspondence.

Contradiction. Therefore, $B$ is uncountable.

## Unrecognizability

Using the result from the previous exercise, we will now prove that there are many more languages than possible Turing machines:

### Corollary (4.18)

*Some languages are not Turing-recognizable.*

- To show that the set of all Turing machines is countable, we first observe that the set of all strings $\Sigma^*$ is countable for any alphabet $\Sigma$.
- With only finitely many strings of each length, we may form a list of $\Sigma^*$ by writing down all strings of length 0, length 1, length 2, and so on. ("Shortlex order")
- The set of all Turing machines is countable because each Turing machine $M$ has an encoding into a string $< M >$. If we simply omit those strings that are not legal encodings of Turing machines, we can obtain a list of all Turing machines.

## Unrecognizability

Using the result from the previous exercise, we will now prove that there are many more languages than possible Turing machines:

### Corollary (4.18)

*Some languages are not Turing-recognizable.*

- To show that the set of all Turing machines is countable, we first observe that the set of all strings $\Sigma^*$ is countable for any alphabet $\Sigma$.
- With only finitely many strings of each length, we may form a list of $\Sigma^*$ by writing down all strings of length 0, length 1, length 2, and so on. ("Shortlex order")
- The set of all Turing machines is countable because each Turing machine $M$ has an encoding into a string $< M >$. If we simply omit those strings that are not legal encodings of Turing machines, we can obtain a list of all Turing machines.

In contrast, the set of all languages is uncountable, as shown on the next slide.

# Unrecognizability (cont.)

(By the exercise, the set $B$ of all infinite binary sequences is uncountable.)

- Let $L$ be the set of all languages over alphabet Σ. We show that $L$ is uncountable by giving a correspondence with $B$, thus showing that the two sets are the same size.

- Let $\Sigma^* = \{s_1, s_2, s_3, \cdots\}$. Each language $A \in L$ has a unique sequence in $B$. The $i$th bit of that sequence is a 1 if $s_i \in A$ and is a 0 if $s_i \notin A$, which is called the characteristic sequence of $A$.

  **Example:**

$$
\begin{array}{rcccccccccccl}
\Sigma^* = \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \cdots & \} & ; \\
A = \{ & & 0, & & 00, & 01, & & & 000, & 001, & \cdots & \} & ; \\
\chi_A = & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \cdots & & .
\end{array}
$$

# Unrecognizability (cont.)

(By the exercise, the set $B$ of all infinite binary sequences is uncountable.)

- Let $L$ be the set of all languages over alphabet Σ. We show that $L$ is uncountable by giving a correspondence with $B$, thus showing that the two sets are the same size.

- Let $\Sigma^* = \{s_1, s_2, s_3, \cdots\}$. Each language $A \in L$ has a unique sequence in $B$. The $i$th bit of that sequence is a 1 if $s_i \in A$ and is a 0 if $s_i \notin A$, which is called the characteristic sequence of $A$.

  **Example:**

$$
\begin{array}{rcccccccccccl}
\Sigma^* = \{ & \varepsilon, & 0, & 1, & 00, & 01, & 10, & 11, & 000, & 001, & \cdots & \} & ; \\
A = \{ & & 0, & & 00, & 01, & & & 000, & 001, & \cdots & \} & ; \\
\chi_A = & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \cdots & & .
\end{array}
$$

- The function $f : L \to B$, where $f(A)$ equals the characteristic sequence of $A$, is one-to-one and onto, and hence is a correspondence.

- Therefore, as $B$ is uncountable, $L$ is uncountable as well.

- Thus we have shown that the set of all languages cannot be put into a correspondence with the set of all Turing machines. We conclude that some languages are not recognized by any Turing machine.

# Unrecognizability (cont.)

We have just seen that there exist languages which aren't Turing-recognizable. Our next objective:

- We shall prove that *there is a specific problem that is algorithmically unsolvable*.
- This result demonstrates that computers are limited in a very fundamental way.
- Unsolvable problems are not necessarily esoteric. Some ordinary problems that people want to solve may turn out to be unsolvable.
- For example, the general problem of software verification is not solvable by computer.
- The specific problem that we will prove algorithmically unsolvable is the one of *testing whether a Turing machine accepts a given input string*.

# The Acceptance Problem

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

## The Acceptance Problem

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

**Theorem (4.11)**

*$A_{\mathrm{TM}}$ is undecidable.*

Proof by contradiction as follows.

# Undecidability of the Acceptance Problem

- Suppose $H$ is a decider for $A_{\mathrm{TM}}$:

$$H\big(\langle M, w\rangle\big) = \begin{cases} \mathit{accept} & \text{if } M \text{ accepts } w \\ \mathit{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

## Undecidability of the Acceptance Problem

- Suppose $H$ is a decider for $A_{\mathrm{TM}}$:

$$H\big(\langle M, w\rangle\big) = \begin{cases} \textit{accept} & \text{if } M \text{ accepts } w \\ \textit{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

- Let $D =$ "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* and if $H$ rejects, *accept*."

  In other words:

$$D\big(\langle M\rangle\big) = \begin{cases} \textit{accept} & \text{if } M \text{ does not accept } \langle M\rangle \\ \textit{reject} & \text{if } M \text{ accepts } \langle M\rangle. \end{cases}$$

## Undecidability of the Acceptance Problem

- Suppose $H$ is a decider for $A_{\mathrm{TM}}$:

$$H\big(\langle M, w\rangle\big) = \begin{cases} \mathit{accept} & \text{if } M \text{ accepts } w \\ \mathit{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

- Let $D =$ "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* and if $H$ rejects, *accept*."

  In other words:

$$D\big(\langle M\rangle\big) = \begin{cases} \mathit{accept} & \text{if } M \text{ does not accept } \langle M\rangle \\ \mathit{reject} & \text{if } M \text{ accepts } \langle M\rangle. \end{cases}$$

- When $D$ takes itself, namely $\langle D \rangle$, as input:

$$D\big(\langle D\rangle\big) = \begin{cases} \mathit{accept} & \text{if } D \text{ does not accept } \langle D\rangle \\ \mathit{reject} & \text{if } D \text{ accepts } \langle D\rangle. \end{cases}$$

# Undecidability of the Acceptance Problem

- Suppose $H$ is a decider for $A_{\mathrm{TM}}$:

$$H(\langle M, w\rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ accepts } w \\ \textit{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

- Let $D =$ "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* and if $H$ rejects, *accept*."

  In other words:

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

- When $D$ takes itself, namely $\langle D\rangle$, as input:

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

  $D$ is forced to do the opposite of what $D$ does. Contradiction.
  Thus, neither $D$ nor $H$ can exist. $\Rightarrow$ $A_{\mathrm{TM}}$ is undecidable.

# Undecidability of the Acceptance Problem

| | $\langle M_1 \rangle$ | $\langle M_2 \rangle$ | $\langle M_3 \rangle$ | $\langle M_4 \rangle$ | $\cdots$ | $\langle D \rangle$ | $\cdots$ |
|---|---|---|---|---|---|---|---|
| $M_1$ | *accept* | *reject* | *accept* | *reject* | | *accept* | |
| $M_2$ | *accept* | *accept* | *accept* | *accept* | $\cdots$ | *accept* | $\cdots$ |
| $M_3$ | *reject* | *reject* | *reject* | *reject* | | *reject* | |
| $M_4$ | *accept* | *accept* | *reject* | *reject* | | *accept* | |
| $\vdots$ | | $\vdots$ | | | $\ddots$ | | |
| $D$ | *reject* | *reject* | *accept* | *accept* | | ? | |
| $\vdots$ | | $\vdots$ | | | | | $\ddots$ |

**FIGURE 4.21**
If $D$ is in the figure, a contradiction occurs at "?"

# The Acceptance Problem (cont.)

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

**Remark:** Although $A_{\mathsf{TM}}$ is undecidable by Theorem 4.11, $A_{\mathsf{TM}}$ is Turing-recognizable because there exists a TM $U$ with $L(U) = A_{\mathsf{TM}}$:

- $U =$ "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*."

## The Acceptance Problem (cont.)

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

**Remark:** Although $A_{\mathsf{TM}}$ is undecidable by Theorem 4.11, $A_{\mathsf{TM}}$ is Turing-recognizable because there exists a TM $U$ with $L(U) = A_{\mathsf{TM}}$:

- $U =$ "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*."

Note that $U$ recognizes $A_{\mathsf{TM}}$, but $U$ does not decide $A_{\mathsf{TM}}$.
The reason is that if $M$ loops on $w$ then $U$ loops on input $\langle M, w\rangle$.

# The Acceptance Problem (cont.)

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

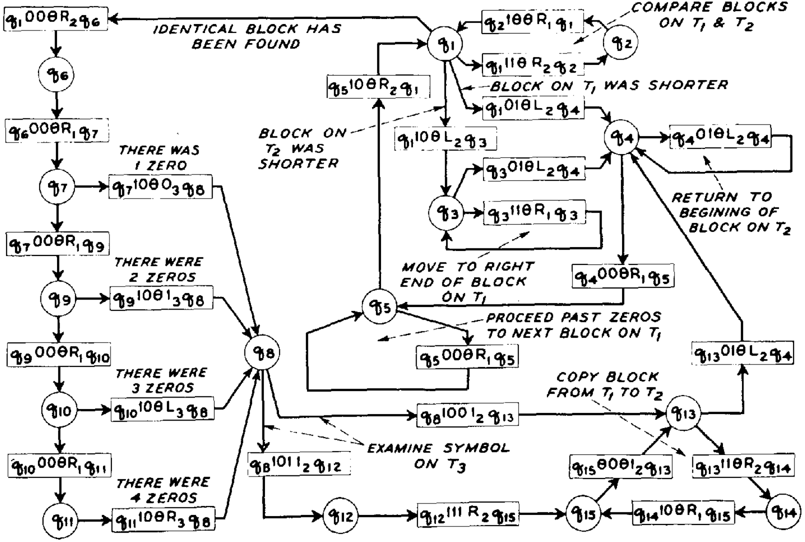
**Remark:** Although $A_{\mathsf{TM}}$ is undecidable by Theorem 4.11, $A_{\mathsf{TM}}$ is Turing-recognizable because there exists a TM $U$ with $L(U) = A_{\mathsf{TM}}$:

- $U =$ "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*."

Note that $U$ recognizes $A_{\mathsf{TM}}$, but $U$ does not decide $A_{\mathsf{TM}}$.
The reason is that if $M$ loops on $w$ then $U$ loops on input $\langle M, w\rangle$.

$U$ is a **universal Turing machine**, first proposed by Alan Turing in 1936. It is called "universal" because it can simulate any other Turing machine from the description of that machine. For an example of what $U$ can look like, see the next slide (don't worry about the details!). Universal TMs played a key role in the development of stored-program computers.

# 15-STATE UNIVERSAL TURING MACHINE




[Figure from E. F. Moore: *Proceedings of the 1952 ACM National Meeting*, pp. 50–55, 1952.]

# Classes of Languages

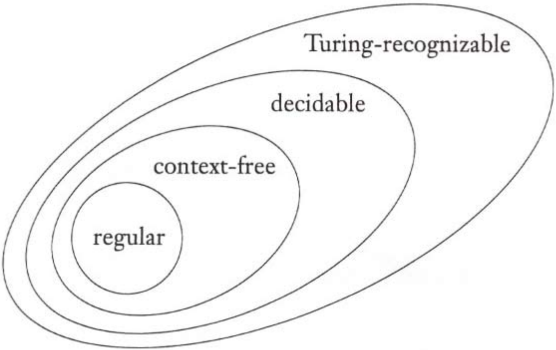



**FIGURE 4.10**
The relationship among classes of languages

# A Turing-Unrecognizable Language

We have seen that $A_{\mathsf{TM}}$ is undecidable but Turing-recognizable.

Our last objective for today:
Find an example of a language that is not even Turing-recognizable.

We'll need the following definition:

- A language is *co-Turing-recognizable* if it is the complement of a Turing-recognizable language.

# A Turing-Unrecognizable Language (cont.)

**Theorem (4.22)**

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

$\Rightarrow$) Suppose $A$ is decidable. Then by definition, $A$ is Turing-recognizable. Let $M$ be a TM with $L(M) = A$ that always halts, and let $M'$ be $M$ with the accept and reject states switched. Since $L(M') = \overline{A}$, this means that $\overline{A}$ is Turing-recognizable and thus $A$ is co-Turing-recognizable, too.

# A Turing-Unrecognizable Language (cont.)

## Theorem (4.22)

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

$\Rightarrow$) Suppose $A$ is decidable. Then by definition, $A$ is Turing-recognizable. Let $M$ be a TM with $L(M) = A$ that always halts, and let $M'$ be $M$ with the accept and reject states switched. Since $L(M') = \overline{A}$, this means that $\overline{A}$ is Turing-recognizable and thus $A$ is co-Turing-recognizable, too.

$\Leftarrow$) Let $M_1$ be a recognizer for $A$ and $M_2$ be a recognizer for $\overline{A}$.

$M =$ "On input $w$:

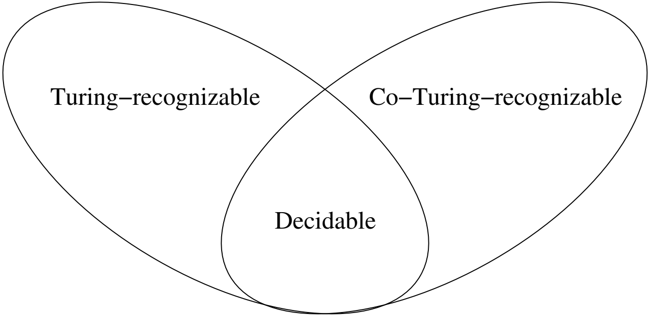
1. Run both $M_1$ and $M_2$ on input $w$ in parallel. ($M$ takes turns simulating one step of each machine until one of them accepts.)
2. If $M_1$ accepts, *accept* and if $M_2$ accepts, *reject*."

For any string $w$, either $M_1$ or $M_2$ will accept $w$, so $M$ is a decider.

- *Decidable language:*
  There exists a TM that, for any input string *w*, tells us whether *w* **is or is not** in the language.

- *Turing-recognizable language*:
  There exists a TM that, for any input string *w*, verifies if *w* is in the language.

- *Co-Turing-recognizable language*:
  There exists a TM that, for any input string *w*, verifies if *w* is not in the language.



## A Turing-Unrecognizable Language (cont.)

- $\overline{A_{\mathrm{TM}}} = \{\langle M, w\rangle \mid M$ is a TM and $M$ does not accept $w$ or $\langle M, w\rangle$ is not an encoding of a TM and an input string $\}$

# A Turing-Unrecognizable Language (cont.)

- $\overline{A_{\mathrm{TM}}} = \{\langle M, w\rangle \mid M$ is a TM and $M$ does not accept $w$ or $\langle M, w\rangle$ is not an encoding of a TM and an input string $\}$

**Corollary (4.23)**

*$\overline{A_{\mathrm{TM}}}$ is not Turing-recognizable.*

- $A_{\mathrm{TM}}$ is Turing-recognizable, but not decidable.
- From Theorem 4.22, $A_{\mathrm{TM}}$ must not be co-Turing-recognizable.
- Therefore, $\overline{A_{\mathrm{TM}}}$ is not Turing-recognizable.