

Decidability

(Based on [Sipser 2006,2013])

Yih-Kuen Tsay

Department of Information Management
National Taiwan University

- 🌐 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.

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_{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*.”

Decidable Languages/Problems (cont.)

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

Theorem (4.2)

A_{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_{DFA} (as a “procedure”) on input $\langle C, w \rangle$.
 3. If M accepts, *accept*; otherwise, *reject*.”

Decidable Languages/Problems (cont.)

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

Theorem (4.3)

A_{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_{DFA} on input $\langle A, w \rangle$.
3. If M accepts, *accept*; otherwise, *reject*.”

Decidable Languages/Problems (cont.)


$$\text{E}_{\text{DFA}} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}.$$

Theorem (4.4)

E_{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.)

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

Theorem (4.5)

EQ_{DFA} is a decidable language.

-  $F =$ “On input $\langle A, B \rangle$, where A and B are DFAs:
1. Construct DFA $C = (A \cap \bar{B}) \cup (\bar{A} \cap B)$.
 2. Run TM T for deciding E_{DFA} on input $\langle C \rangle$.
 3. If T accepts, *accept*; otherwise, *reject*.”

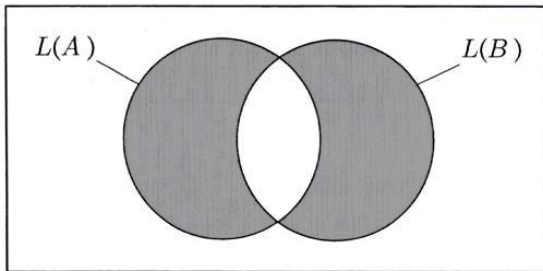


FIGURE 4.6

The symmetric difference of $L(A)$ and $L(B)$

Source: [Sipser 2006]

Decidable CFL Properties

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

Theorem (4.7)

A_{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*.”

Decidable CFL Properties (cont.)

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

Theorem (4.8)

E_{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 $A \rightarrow U_1 U_2 \cdots U_k$ 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*.”

Decidability of CFLs

Theorem (4.9)

Every context-free language is decidable.

- Let G be a CFG for the given language A and design a TM M_G that decides A .
- $M_G =$ “On input w :
 - Run TM S for deciding A_{CFG} on input $\langle G, w \rangle$.
 - If S accepts, *accept*; otherwise, *reject*.”

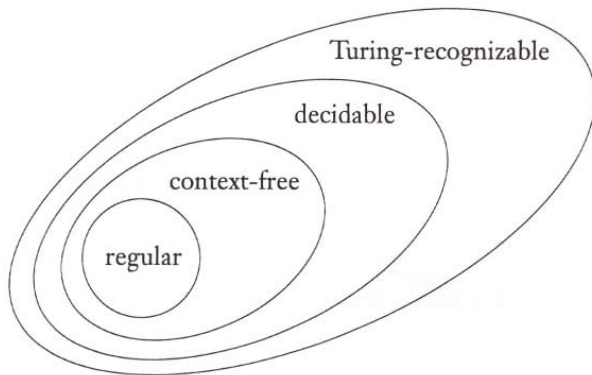


FIGURE 4.10

The relationship among classes of languages

Source: [Sipser 2006]

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).
- 🌐 Linear Bounded Automata will be introduced later.

Undecidability

- 🌐 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

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

Theorem (4.11)

A_{TM} is undecidable.

- 🌐 We will prove this fundamental result later.
- 🌐 On the other hand, A_{TM} is Turing-recognizable.

The Acceptance Problem (cont.)

- 🌐 $U =$ “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*.”
- 🌐 If we had (actually not) some way to determine that M was not *halting* on w , then we could turn the recognizer U into a decider.

Note: The Turing machine U is an example of the *universal Turing machine*, as it is capable of simulating any other Turing machine from the description of that machine. The universal Turing machine inspired “stored-program” computers.

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 $\mathcal{N} = \{1, 2, 3, \dots\}$; it is **uncountable**, otherwise.

Countable vs. Uncountable Sets (cont.)

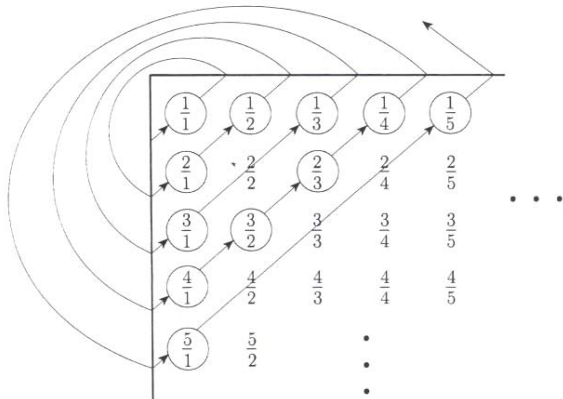


FIGURE 4.16
 A correspondence of \mathcal{N} and \mathcal{Q}

Source: [Sipser 2006]

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. <u>1</u> 4159...
2	55.5 <u>5</u> 555...
3	0.12 <u>3</u> 45...
4	0.500 <u>0</u> 0...
\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, $x = 0.4641\dots$ is a possible choice.
- This proof technique is called *diagonalization*, discovered by Georg Cantor in 1873.

Unrecognizability

Corollary (4.18)

Some languages are not Turing-recognizable.

- 🌐 The set of all Turing machines is **countable** because each Turing machine M has an encoding into a string $\langle M \rangle$.
- 🌐 Let \mathcal{L} be the set of all languages over alphabet Σ .
- 🌐 We can show that there is a correspondence between \mathcal{L} and the **uncountable** set \mathcal{B} of all infinite binary sequences.
 - ☀ Let $\Sigma^* = \{s_1, s_2, s_3, \dots\}$.
 - ☀ Each language $A \in \mathcal{L}$ has a unique sequence in \mathcal{B} , where the i -th bit is a 1 if and only if $s_i \in A$.

Undecidability of the Acceptance Problem

Suppose H is a decider for A_{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*.”

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

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

Undecidability of the Acceptance Problem (cont.)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	\dots
M_1	<i>accept</i>		<i>accept</i>		
M_2	<i>accept</i>	<i>accept</i>	<i>accept</i>	<i>accept</i>	
M_3					\dots
M_4	<i>accept</i>	<i>accept</i>			
\vdots			\vdots		

FIGURE 4.19

Entry i, j is *accept* if M_i accepts $\langle M_j \rangle$

Source: [Sipser 2006]

Undecidability of the Acceptance Problem (cont.)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$...
M_1	<i>accept</i>	<i>reject</i>	<i>accept</i>	<i>reject</i>	
M_2	<i>accept</i>	<i>accept</i>	<i>accept</i>	<i>accept</i>	...
M_3	<i>reject</i>	<i>reject</i>	<i>reject</i>	<i>reject</i>	
M_4	<i>accept</i>	<i>accept</i>	<i>reject</i>	<i>reject</i>	
\vdots			\vdots		

FIGURE 4.20

Entry i, j is the value of H on input $\langle M_i, \langle M_j \rangle \rangle$

Source: [Sipser 2006]

Undecidability of the Acceptance Problem (cont.)

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

FIGURE 4.21

If D is in the figure, a contradiction occurs at “?”

Source: [Sipser 2006]

A Turing-Unrecognizable Language

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

Theorem (4.22)

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

- 🌐 Let M_1 be a recognizer for A and M_2 be a recognizer for \bar{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 halts.)
 2. If M_1 accepts, *accept* and if M_2 accepts, *reject*.”

A Turing-Unrecognizable Language (cont.)

🌐 $\overline{A_{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ does not accept } w\}$.

Corollary (4.23)

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

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