Theory of Computing
Undecidability

Ming-Hsien Tsai

Department of Information Management
National Taiwan University

Spring 2019

(original created by Bow-Yaw Wang, some slides from Yih-Kuen Tsay)

Languages and Computational Problems

- In this course, we are working on models of computation.
  - finite automata, pushdown automata, Turing machines.
- Now consider the following computational problem P:
  Given a graph G and two nodes s, t on G, check if there is a path connecting s and t.
- How do we formulate this problem in the terminology of machines?
  - Recall that a machine recognizes a language.
  - We therefore formulate a computational problem as a language.
- Consider the following language A:

$$A = \{\langle G, s, t\rangle : \text{ there is a path connecting } s \text{ and } t \text{ in } G\}.$$

- To find an algorithm that solves the computational problem P is to find a TM that decides the language A.

Acceptance Problem for DFA's

- The acceptance problem for DFA's is to test whether a given deterministic finite automaton accepts a given input string.
- Consider the following language:

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

Theorem 1

A_{DFA} is a decidable language.

Proof.

We need to give a TM M that decides A_{DFA}. Consider
$M =$ "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."

□

Acceptance Problem for NFA's

- The acceptance problem for NFA's is also solvable. Consider

$$A_{\text{NFA}} = \{\langle B, w \rangle : B \text{ is an NFA that accepts input string } w\}.$$

Theorem 2

A_{NFA} is a decidable language.

Proof.

We need to give a TM that decides A_{NFA}. Consider
N = "On input $\langle B, w \rangle$ where B is an NFA and w is a string:

1. Convert the NFA B into a DFA C.
2. Run TM M from Theorem 1 on input $\langle C, w \rangle$.
3. If M accepts, accept; otherwise, reject." □

- Can we simulate an NFA by an NTM directly? Why not?

Acceptance Problem for Regular Languages

- Consider

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

Theorem 3

A_{REX} is a decidable language.

Proof.

Consider
$P =$ "On input $\langle R, w\rangle$ where R is a regular expression and w is a string:

1. Convert R into an NFA A.
2. Run TM N (Theorem 2) on the input $\langle A, w\rangle$.
3. If N accepts, accept; otherwise, reject." □

Emptiness Problem for DFA's

- The emptiness problem for DFA's is to test whether the language recognized by a given DFA A is empty or not.
- Consider

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

Theorem 4

E_{DFA} is a decidable language.

Proof.

Consider

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

1. Mark the start state of A.
2. Repeat until no new state is marked:
    1. Mark any state that has a transition from a marked state to it.
3. If an accept state is marked, reject; otherwise, accept." □

Language Equivalence Problem for DFA's

- The language equivalence problem for DFA's is to test whether two given DFA's recognize the same language.
- Consider

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

Language Equivalence Problem for DFA's

Theorem 5

EQ_{DFA} is a decidable language.

Proof.

$L(A)$ and $L(B)$ are regular languages. Recall that regular languages are closed under complementation. Consider

$F =$ "On input $\langle A, B \rangle$ where A and B are DFA's:

1. Construct a DFA C that recognizes

$$(L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B)).$$

2. Run TM T (Theorem 4) on the input $\langle C \rangle$.
3. If T accepts, accept; otherwise, reject." □

Acceptance Problem for Context-Free Grammars

- Consider

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

Lemma 6

Let G be a CFG in Chomsky normal form and $w \neq \epsilon$ generated by G. Any derivation of w has $2|w| - 1$ steps.

Proof.

We show any derivation $A \stackrel{*}{\Longrightarrow} u$ has $2|u| - 1$ steps by induction on $|u|$.

- $|u| = 1$. Since G is in Chomsky normal form, the only possible rule is $A \longrightarrow a$. Hence $A \stackrel{*}{\Longrightarrow} u$ in 1 step.
- $|u| = k + 1$. Consider a derivation $A \Longrightarrow BC \stackrel{*}{\Longrightarrow} u$. Let $B \stackrel{*}{\Longrightarrow} u_1$ and $C \stackrel{*}{\Longrightarrow} u_2$ where $u = u_1u_2$. By IH, $B \stackrel{*}{\Longrightarrow} u_1$ in $2|u_1| - 1$ steps and $C \stackrel{*}{\Longrightarrow} u_2$ in $2|u_2| - 1$ steps. Thus $A \stackrel{*}{\Longrightarrow} u$ in $1 + (2|u_1| - 1) + (2|u_2| - 1) = 2(|u_1| + |u_2|) - 1 = 2|u| - 1$ steps. □

Acceptance Problem for Context-Free Grammars

Theorem 7

A_{CFG} is a decidable language.

Proof.

Consider

S = "On input $\langle G, w \rangle$ where G is a CFG and w is a string:

1. Convert G into Chomsky normal form.
2. If $w = \epsilon$, check all derivations with 1 step.
3. If $w \neq \epsilon$, check all derivations with $2|w| - 1$ steps.
4. If any of these finite derivations generates w, accept; otherwise, reject."

□

- Can we simply check all derivations of G without converting it into Chomsky normal form? Why not?

Emptiness Problem for Context-Free Grammars

- Consider

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

Theorem 8

E_{CFG} is a decidable language.

Proof.

A CFG G generates a non-empty language iff there is a derivation for a string from its start variable. We mark symbols that generate a string.
R = "On input $\langle G\rangle$ where G is a CFG:

1. Mark all terminal symbols in G.
2. Repeat until no variable is marked:
   1. Mark any variable A that has a rule $A \longrightarrow U_1U_2\cdots U_k$ in G where U_i are marked for $i = 1, \ldots, k$.
3. If the start variable is not marked, accept; otherwise, reject." □

Context-Free Languages are Decidable

Theorem 9

Every context-free language is decidable.

Proof.

Let A be a context-free language. We need to come up with a TM that decides A. Let G be a CFG for A. Consider

M_G = "On input w:

1. Run TM S (Theorem 7) on the input $\langle G, w \rangle$.
2. If S accepts, accept; otherwise, reject." □

- Let A be a context-free language and P a pushdown automaton recognizing A.
- Can we use an NTM to simulate P? Why not?

Relationship among Languages

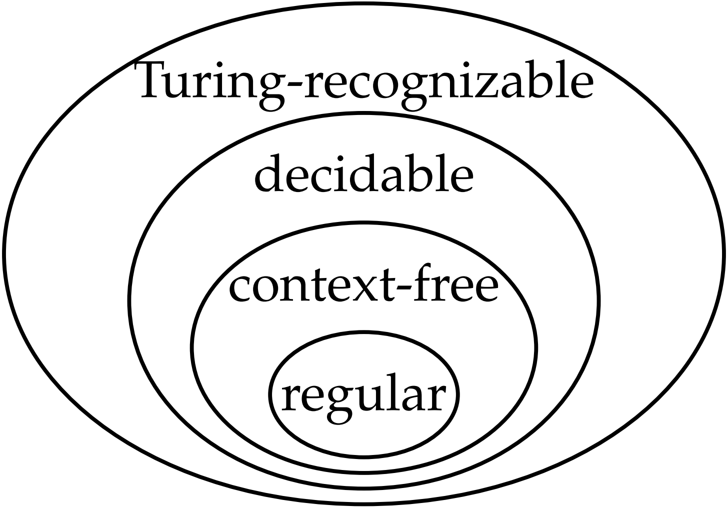

Figure: Relationship among Different Languages

Exercise

- Consider the problem of determining whether a DFA and a regular expression are equivalent. Express this problem as a language and show that it is decidable.
- Let $E_{TM} = \{\langle M \rangle \mid M \text{ is a } TM \text{ and } L(M) = \emptyset\}$. Show that $\overline{E_{TM}}$, the complement of E_{TM}, is Turing-recognizable.

Acceptance Problem for TM's

- Consider

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

- Consider the following TM:
  U = "On input $\langle M, w\rangle$ where M is a TM and w is a string:
  1. Simulate M on the input w.
  2. If M enters its accept state, accept; if M enters its reject state, reject."
- Does U decide A_{TM}? Why not?
- The TM U is called the universal Turing machine, which inspired "stored-program" computers.

Countable vs. Uncountable Sets

Definition 10

Let f be a function from A to B.

- We say that f is one-to-one (injective) if $f(a) \neq f(b)$ whenever $a \neq b$.
- Say that f is onto (surjective) 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 (bijection).
- Two sets are considered to have the same size if there is a correspondence between them.

Definition 11

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.)

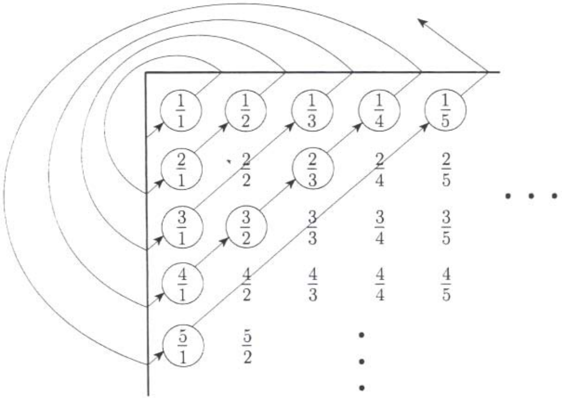

FIGURE 4.16
A correspondence of $\mathcal{N}$ and $\mathcal{Q}$

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 12

$\mathcal{R}$ 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.
- This proof technique is called diagonalization, discovered by Georg Cantor in 1873.

Exercise

- Let $T = \{(i, j, k) \mid i, j, k \in \mathbb{N}\}$. Show that T is countable (Hint: any subset of $\mathbb{N}$ is countable).
- Let B be the set of all infinite sequence over $\{0, 1\}$. Show that B is uncountable using a proof by diagnoalization.

Counting Arguments

- Recall that $|\mathbb{N}| = |\mathbb{Z}| = |\Sigma^*| = \aleph_0$ (Σ is finite).
- Also recall that $|\mathcal{P}(\Sigma^*)| > \aleph_0$.
  - Consult your textbook or my notes on discrete mathematics if you are not sure.

Corollary 13

Some languages are not Turing-recognizable.

Proof.

The set of all Turing machines is countable since each TM M has an encoding $\langle M \rangle$ in Σ^*.
The set of all languages over Σ is $\mathcal{P}(\Sigma^*)$ and hence is uncountable.
Hence some languages are not Turing-recognizable. □

- There are in fact uncountably many languages that cannot be recognized by Turing machines.
- Can we find a concrete example?

Undecidability of the Acceptance Problem for TM's

Theorem 14

$A_{TM} = \{\langle M, w\rangle : M \text{ is a TM and } M \text{ accepts } w\}$ *is not a decidable language.*

Proof.

Suppose there is a TM H deciding A_{TM}. That is,

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

Consider the following TM:
D = "On input $\langle M\rangle$ where M is a TM:

1. Run H on the input $\langle M, \langle M\rangle\rangle$.
2. If H accepts, reject. If H rejects, accept."

Consider

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

A contradiction. □

A Turing-unrecognizable Language

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

Theorem 15

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

Proof.

If A is decidable, then A and $\overline{A}$ are both recognizable. Since $\overline{\overline{A}} = A$, A is Turing-recognizable and co-Turing-recognizable.
Now suppose A and $\overline{A}$ are Turing-recognizable by M_1 and M_2 respectively. Consider
$M =$ "On input w:

1. Run both M_1 and M_2 on the input w in parallel.
2. If M_1 accepts, accept; if M_2 accepts; reject." □

A Turing-unrecognizable Language

Corollary 16

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

Proof.

A_{TM} is Turing-recognizable. If $\overline{A_{\text{TM}}}$ is Turing-recognizable, A_{TM} is both Turing-recognizable and co-Turing-recognizable. By Theorem 15, A_{TM} is decidable. A contradiction. □