形式语言与计算复杂性

第 3 章 Computability Theory

3.2 Decidability

黄文超

https://faculty.ustc.edu.cn/huangwenchao

$\longrightarrow$ 教学课程 $\longrightarrow$ 形式语言与计算复杂性

3.2 Decidability

Outline

3.2 Decidability

Outline

3.2 Decidability

Introduction

回顾: 问: Why introducing Turing machine?
答: Prove that some problem cannot be *solved algorithmically*, i.e., Hilbert's 10th problem

问: How?
答: Determine the *Decidability*

问: Why studying unsolvability?
答: 2 reasons

- You realize that the *unsolvable* problem *must be simplified* or *altered* before you can find an algorithmic solution.
- Culture: *stimulate* your *imagination* and help you gain an important *perspective* on computation

3.2 Decidability

Introduction

回顾: 问: Why introducing Turing machine?
答: Prove that some problem cannot be *solved algorithmically*, i.e., Hilbert's 10th problem

问: How?
答: Determine the *Decidability*

问: Why studying unsolvability?
答: 2 reasons

- You realize that the *unsolvable* problem *must be simplified* or *altered* before you can find an algorithmic solution.
- Culture: *stimulate* your *imagination* and help you gain an important *perspective* on computation

3.2 Decidability

Introduction

回顾: 问: Why introducing Turing machine?
答: Prove that some problem cannot be *solved algorithmically*, i.e., Hilbert's 10th problem

问: How?
答: Determine the *Decidability*

问: Why studying unsolvability?
答: 2 reasons

- You realize that the *unsolvable* problem *must be simplified* or *altered* before you can find an algorithmic solution.
- Culture: *stimulate* your *imagination* and help you gain an important *perspective* on computation

3.2 Decidability

Outline

问: How to study the Decidability of *problems*?
答: Study the Decidability of the *languages*

问: Then?
答: Firstly, which are decidable?; Then, which are not?

问: Any examples of decidable language?
答: Let's *play* with *Regular language*, *context-free language*, ...

问: How to play?
答: Firstly, formally define the *problem*, i.e., the *language*. Then prove the language is decidable

问: How to study the Decidability of *problems*?
答: Study the Decidability of the *languages*

问: Then?
答: Firstly, which are decidable?; Then, which are not?

问: Any examples of decidable language?
答: Let's *play* with *Regular language*, *context-free language*, ...

问: How to play?
答: Firstly, formally define the *problem*, i.e., the *language*. Then prove the language is decidable

问: How to study the Decidability of *problems*?
答: Study the Decidability of the *languages*

问: Then?
答: Firstly, which are decidable?; Then, which are not?

问: Any examples of decidable language?
答: Let's *play* with *Regular language*, *context-free language*, ...

问: How to play?
答: Firstly, formally define the *problem*, i.e., the *language*. Then prove the language is decidable

问: How to study the Decidability of *problems*?
答: Study the Decidability of the *languages*

问: Then?
答: Firstly, which are decidable?; Then, which are not?

问: Any examples of decidable language?
答: Let's *play* with *Regular language*, *context-free language*, ...

问: How to play?
答: Firstly, formally define the *problem*, i.e., the *language*. Then prove the language is decidable

3.2 Decidability

Outline

定理: Decidability of A_{DFA}

The language A_{DFA} is decidable, where

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

定理: Decidability of A_{DFA}

The language A_{DFA} is decidable, where

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

证明思路 (Proof by Construction)

We simply need to present a TM M that decides A_{DFA}.
$M =$ "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
   - a non-accepting state: reject

定理: Decidability of A_{DFA}

The language A_{DFA} is decidable, where

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

证明 (Proof by Construction)

First, examine the input $\langle B, w\rangle$.

- Proper representation:
  - B is simply a list of its five components: Q, Σ, δ, q_0, and F
- When M receives its input, M first determines *whether it properly represents* a DFA B and a string w.
  - If not, M rejects.

定理: Decidability of A_{DFA}

The language A_{DFA} is decidable, where

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

证明 (Proof by Construction)

First, examine the input $\langle B, w\rangle$.

- Proper representation:
  - B is simply a list of its five components: Q, Σ, δ, q_0, and F
- When M receives its input, M first determines *whether it properly represents* a DFA B and a string w.
  - If not, M rejects.

定理: Decidability of A_{DFA}

The language A_{DFA} is decidable, where

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

证明 (Proof by Construction)

First, examine the input $\langle B, w\rangle$.

- Proper representation:
  - B is simply a list of its five components: Q, Σ, δ, q_0, and F
- When M receives its input, M first determines *whether it properly represents* a DFA B and a string w.
  - If not, M rejects.

定理: Decidability of A_{DFA}

The language A_{DFA} is decidable, where

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

证明 (Proof by Construction)

Then M carries out the *simulation* directly

定理: Decidability of A_{DFA}

The language A_{DFA} is decidable, where

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

证明 (Proof by Construction)

Then M carries out the *simulation* directly

- 思路: It *keeps track* of B' s current *state* and B' s current *position* in the *input* w by *writing* this information down on its *tape*.

定理: Decidability of A_{DFA}

The language A_{DFA} is decidable, where

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

证明 (Proof by Construction)

Then M carries out the *simulation* directly

- *Initially*, B' s current *state* is q_0 and B' s current *input position* is the *leftmost* symbol of w.
- The states and position are *updated* according to the specified *transition function* δ.
- When M *finishes* processing *the last symbol* of w,
  - M *accepts* the input if B is in an *accepting state*
  - M *rejects* the input if B is in a *nonaccepting state*.

定理: Decidability of A_{DFA}

The language A_{DFA} is decidable, where

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

证明 (Proof by Construction)

Then M carries out the *simulation* directly

- *Initially*, B' s current *state* is q_0 and B' s current *input position* is the *leftmost* symbol of w.
- The states and position are *updated* according to the specified *transition function* δ.
- When M *finishes* processing *the last symbol* of w,
  - M *accepts* the input if B is in an *accepting state*
  - M *rejects* the input if B is in a *nonaccepting state*.

定理: Decidability of A_{DFA}

The language A_{DFA} is decidable, where

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

证明 (Proof by Construction)

Then M carries out the *simulation* directly

- *Initially*, B' s current *state* is q_0 and B' s current *input position* is the *leftmost* symbol of w.
- The states and position are *updated* according to the specified *transition function* δ.
- When M *finishes* processing *the last symbol* of w,
  - M *accepts* the input if B is in an *accepting state*
  - M *rejects* the input if B is in a *nonaccepting state*.

定理: Decidability of A_{NFA}

The language A_{NFA} is decidable, where

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

定理: Decidability of A_{NFA}

The language A_{NFA} is decidable, where

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

证明: Proof by Construction

Present a TM N that decides A_{NFA}

定理: Decidability of A_{NFA}

The language A_{NFA} is decidable, where

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

证明: Proof by Construction

Present a TM N that decides A_{NFA}

- 方法 1: Operate like $\underline{M}$
- 方法 2: New idea: *Have N use $\underline{M}$ as a subroutine*
  - *N converts* the *NFA* it receives as input to a *DFA*
  - Pass the DFA and w to $\underline{M}$

定理: Decidability of A_{NFA}

The language A_{NFA} is decidable, where

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

证明: Proof by Construction

Present a TM N that decides A_{NFA}

- 方法 1: Operate like M
- 方法 2: New idea: *Have N use M as a subroutine*
  - N *converts* the *NFA* it receives as input to a *DFA*
  - Pass the DFA and w to M

定理: Decidability of A_{NFA}

The language A_{NFA} is decidable, where

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

证明: Proof by Construction

Present a TM N that decides A_{NFA}

$N =$ "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 $\underline{M}$ on input $\langle C, w\rangle$
3. If $\underline{M}$ accepts, accept; otherwise, reject."

定理: Decidability of A_{REX}

The language A_{REX} is decidable, where

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

定理: Decidability of A_{REX}

The language A_{REX} is decidable, where

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

证明

The following TM P decides A_{REX}

$P =$ "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 NFA A
2. Run TM $\underline{N}$ on input $\langle A, w\rangle$.
3. If $\underline{N}$ accepts, accept; if $\underline{N}$ rejects, reject."

定理: Decidability of E_{DFA}

The language E_{DFA} is decidable, where

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

定理: Decidability of E_{DFA}

The language E_{DFA} is decidable, where

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

证明 (Proof by Construction)

Design a TM T

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

1. *Mark* the *start state* of A
2. Repeat 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."

定理: Decidability of EQ_{DFA}

The language EQ_{DFA} is decidable, where

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

定理: Decidability of EQ_{DFA}

The language EQ_{DFA} is decidable, where

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

准备: Construct a DFA C

定理: Decidability of EQ_{DFA}

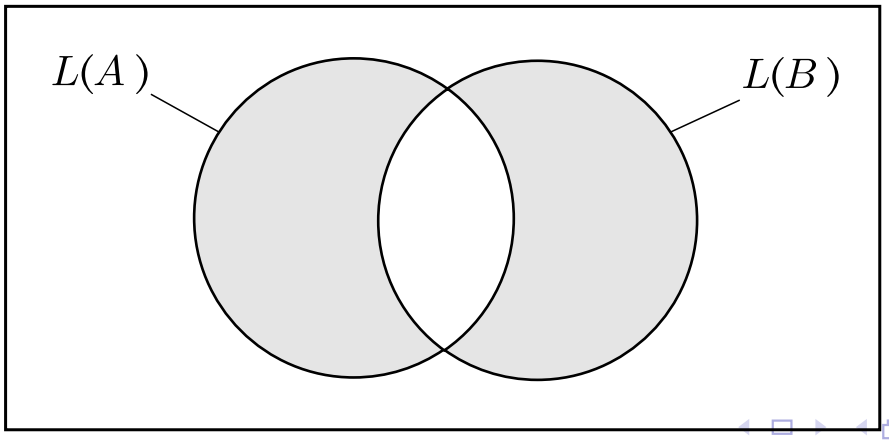
The language EQ_{DFA} is decidable, where

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

准备: Construct a DFA C

- The language of C is

$$L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$$



定理: Decidability of EQ_{DFA}

The language EQ_{DFA} is decidable, where

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

准备: Construct a DFA C

- The language of C is

$$L(C) = \Big(L(A) \cap \overline{L(B)}\Big) \cup \Big(\overline{L(A)} \cap L(B)\Big)$$

- Prove class of regular languages *closed* under complementation, union, and intersection. So C can be a DFA.
- DFA C can be constructed by a TM
- $L(C) = \emptyset$ iff $L(A) = L(B)$

定理: Decidability of EQ_{DFA}

The language EQ_{DFA} is decidable, where

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

准备: Construct a DFA C

- The language of C is

$$L(C) = \Big(L(A) \cap \overline{L(B)}\Big) \cup \Big(\overline{L(A)} \cap L(B)\Big)$$

- Prove class of regular languages *closed* under complementation, union, and intersection. So C can be a DFA.
- DFA C can be constructed by a TM
- $L(C) = \emptyset$ iff $L(A) = L(B)$

定理: Decidability of EQ_{DFA}

The language EQ_{DFA} is decidable, where

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

准备: Construct a DFA C

- The language of C is

$$L(C) = \Big(L(A) \cap \overline{L(B)}\Big) \cup \Big(\overline{L(A)} \cap L(B)\Big)$$

- Prove class of regular languages *closed* under complementation, union, and intersection. So C can be a DFA.
- DFA C can be constructed by a TM
- $L(C) = \emptyset$ iff $L(A) = L(B)$

定理: Decidability of EQ_{DFA}

The language EQ_{DFA} is decidable, where

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

证明 (Proof by Construction)

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

1. Construct DFA C as described
2. Run TM T on input $\langle C\rangle$
3. If T accepts, accept. If T rejects, reject."

定理: Decidability of A_{CFG}

The language A_{CFG} is decidable, where

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

定理: Decidability of A_{CFG}

The language A_{CFG} is decidable, where

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

证明思路

可选方案:

1. use G to go through *all derivations* to determine whether any is a derivation of w.
   - This idea *doesn't work*, as *infinitely* many derivations *may* have to be *tried*
   - If G does not generate w, this algorithm would never halt.
2. ensure that the algorithm tries only *finitely* many derivations
   - Convert G into Chomsky normal form
   - any derivation of w has $2n - 1$ steps

定理: Decidability of A_{CFG}

The language A_{CFG} is decidable, where

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

证明思路

可选方案:

1. use G to go through *all derivations* to determine whether any is a derivation of w.
   - This idea *doesn't work*, as *infinitely* many derivations *may* have to be *tried*
   - If G does not generate w, this algorithm would never halt.
2. ensure that the algorithm tries only *finitely* many derivations
   - Convert G into Chomsky normal form
   - any derivation of w has $2n - 1$ steps

定理: Decidability of A_{CFG}

The language A_{CFG} is decidable, where

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

证明 (Proof by Construction)

The TM S for A_{CFG} follows.

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 $2n - 1$ *steps*, where n is the length of w; except if $n = 0$, then instead list all derivations with *one step*.
3. If any of these derivations generate w, accept; if not, reject."

定理: Decidability of E_{CFG}

The language E_{CFG} is decidable, where

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

定理: Decidability of E_{CFG}

The language E_{CFG} is decidable, where

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

证明思路

可选方案:

定理: Decidability of E_{CFG}

The language E_{CFG} is decidable, where

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

证明思路

可选方案:

use TM $\underline{S}$

- It states that we can test whether a CFG generates some particular string w.
- The algorithm might try going through *all possible* w' s, one by one
- there are *infinitely many* w' s to try

定理: Decidability of E_{CFG}

The language E_{CFG} is decidable, where

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

证明思路

可选方案:
use TM $\underline{S}$

- It states that we can test whether a CFG generates some particular string w.
- The algorithm might try going through *all possible* w' s, one by one
- there are *infinitely many* w' s to try

定理: Decidability of E_{CFG}

The language E_{CFG} is decidable, where

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

证明思路

可选方案:
use TM $\underline{S}$

- It states that we can test whether a CFG generates some particular string w.
- The algorithm might try going through *all possible* w' s, one by one
- there are *infinitely many* w' s to try

定理: Decidability of E_{CFG}

The language E_{CFG} is decidable, where

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

证明思路

可选方案:
use TM $\underline{S}$

- It states that we can test whether a CFG generates some particular string w.
- The algorithm might try going through *all possible* w' s, one by one
- there are *infinitely many* w' s to try

定理: Decidability of E_{CFG}

The language E_{CFG} is decidable, where

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

证明思路

可选方案:

Propagate from *terminal symbols*

- First, the algorithm marks all the terminal symbols in the grammar
- Then, it scans all the rules of the grammar
  - If it ever *finds* a *rule* that permits some *variable* to be *replaced by* some string of symbols, *all of which are already marked*, the algorithm knows that this *variable can be marked*, too

定理: Decidability of E_{CFG}

The language E_{CFG} is decidable, where

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

证明思路

可选方案:

Propagate from *terminal symbols*

- First, the algorithm marks all the terminal symbols in the grammar
- Then, it scans all the rules of the grammar
  - If it ever *finds* a *rule* that permits some *variable* to be *replaced by* some string of symbols, *all of which are already marked*, the algorithm knows that this *variable can be marked*, too

定理: Decidability of E_{CFG}

The language E_{CFG} is decidable, where

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

证明思路

可选方案:

Propagate from *terminal symbols*

- First, the algorithm marks all the terminal symbols in the grammar
- Then, it scans all the rules of the grammar
  - If it ever *finds* a *rule* that permits some *variable* to be *replaced by* some string of symbols, *all of which are already marked*, the algorithm knows that this *variable can be marked*, too

定理: Decidability of E_{CFG}

The language E_{CFG} is decidable, where

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

证明 (Proof by Construction)

$R =$ "On input $\langle G \rangle$, where G is a CFG:

1. Mark all terminal symbols in G
2. Repeat until no new variables get marked:
3. Mark any variable A where G has a rule $A \to U_1 U_2 \cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked
4. If the start variable is not marked, accept; otherwise, reject."

问题: Decidability of EQ_{CFG}

Is the language EQ_{CFG} decidable?

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

答: No!

问: Why not designing a TM similar to TM F ?
答: The design of F depends on closure property under complementation or intersection

问: How to prove that EQ_{CFG} is not decidable?
答: See *Chap 3.3 Reducibility*.

问题: Decidability of EQ_{CFG}

Is the language EQ_{CFG} decidable?

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

答: No!

问: Why not designing a TM similar to TM F ?
答: The design of F depends on closure property under complementation or intersection

问: How to prove that EQ_{CFG} is not decidable?
答: See *Chap 3.3 Reducibility*.

问题: Decidability of EQ_{CFG}

Is the language EQ_{CFG} decidable?

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

答: No!

问: Why not designing a TM similar to TM $\underline{F}$?
答: The design of F depends on closure property under complementation or intersection

问: How to prove that EQ_{CFG} is not decidable?
答: See *Chap 3.3 Reducibility*.

问题: Decidability of EQ_{CFG}

Is the language EQ_{CFG} decidable?

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

答: No!

问: Why not designing a TM similar to TM F ?
答: The design of F depends on closure property under complementation or intersection

问: How to prove that EQ_{CFG} is not decidable?
答: See *Chap 3.3 Reducibility*.

定理

Every context-free language is decidable

定理

Every context-free language is decidable

证明

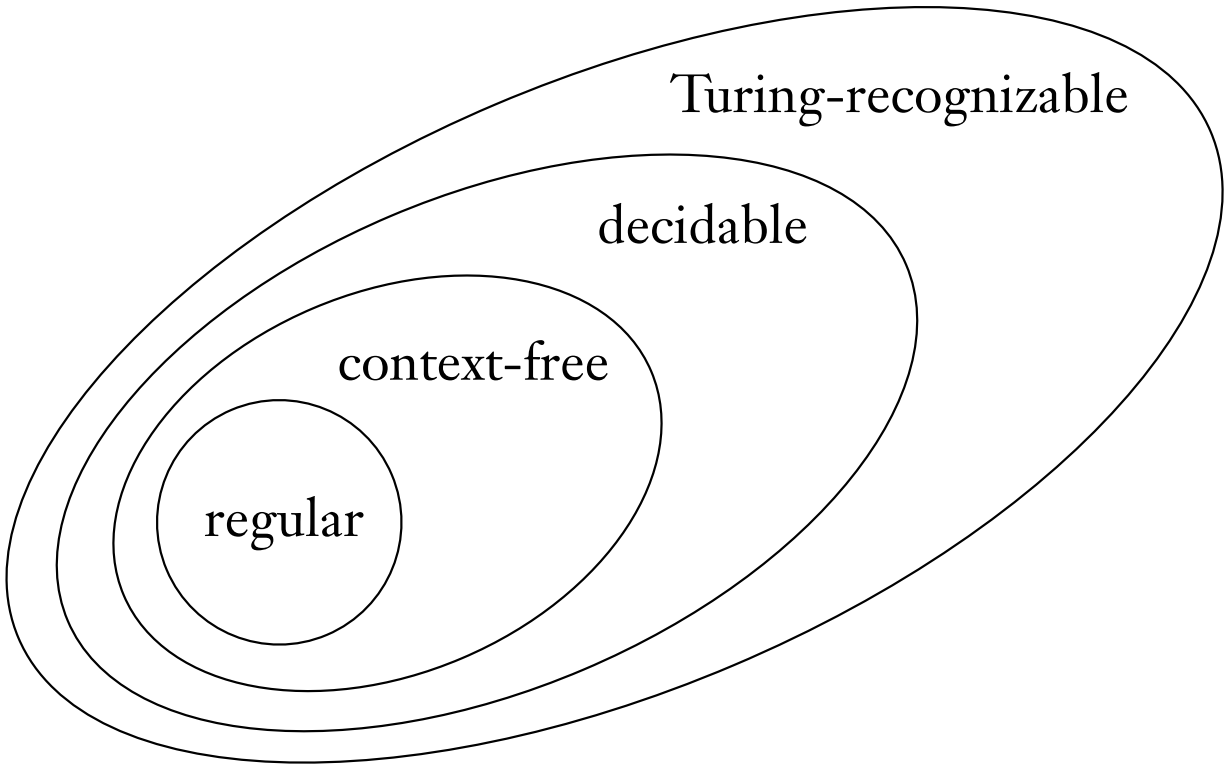
Let A be a CFL.
Let G be a CFG for A.

Design a TM M_G that decides A.
$M_G =$ "On input w:

1. Run TM $\underline{S}$ on input $\langle G, w \rangle$.
2. If this machine accepts, accept; if it rejects, reject."

定理

Every context-free language is decidable



3.2 Decidability

2. Undecidability

问: What sorts of problems are *unsolvable* by computer? Are they esoteric (深奥), dwelling only in the minds of theoreticians?
答: No! *Even* some *ordinary problems* that people want to solve turn out to be computationally *unsolvable*.

问: Any examples?
答: The *general problem* of *software verification* is not solvable by computer.

问: So, why do we study undecidability?
答: 2 points

- help you develop a *feeling* for the *types* of problems that are *unsolvable*
- to learn techniques for *proving* unsolvability.

问: Now, what is *first* undecidable problem to analyze?
答: A_{TM}

3.2 Decidability

2. Undecidability

问: What sorts of problems are *unsolvable* by computer? Are they esoteric (深奥), dwelling only in the minds of theoreticians?
答: No! *Even* some *ordinary problems* that people want to solve turn out to be computationally *unsolvable*.

问: Any examples?
答: The *general problem* of *software verification* is not solvable by computer.

问: So, why do we study undecidability?
答: 2 points

- help you develop a *feeling* for the *types* of problems that are *unsolvable*
- to learn techniques for *proving* unsolvability.

问: Now, what is *first* undecidable problem to analyze?
答: A_{TM}

3.2 Decidability

2. Undecidability

问: What sorts of problems are *unsolvable* by computer? Are they esoteric (深奥), dwelling only in the minds of theoreticians?
答: No! *Even* some *ordinary problems* that people want to solve turn out to be computationally *unsolvable*.

问: Any examples?
答: The *general problem* of *software verification* is not solvable by computer.

问: So, why do we study undecidability?
答: 2 points

- help you develop a *feeling* for the *types* of problems that are *unsolvable*
- to learn techniques for *proving* unsolvability.

问: Now, what is *first* undecidable problem to analyze?
答: A_{TM}

问: What sorts of problems are *unsolvable* by computer? Are they esoteric (深奥), dwelling only in the minds of theoreticians?
答: No! *Even* some *ordinary problems* that people want to solve turn out to be computationally *unsolvable*.

问: Any examples?
答: The *general problem* of *software verification* is not solvable by computer.

问: So, why do we study undecidability?
答: 2 points

- help you develop a *feeling* for the *types* of problems that are *unsolvable*
- to learn techniques for *proving* unsolvability.

问: Now, what is *first* undecidable problem to analyze?
答: A_{TM}

定理

The language A_{TM} is *recognizable* but *undecidable*, where

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

问: How to prove that A_{TM} is *recognizable*?
答: construct a Turing machine 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*."

问: Why is A_{TM} undecidable?
答: M may loop on w

问: How to formally prove the undecidability?
答: ▸ Proof

定理

The language A_{TM} is *recognizable* but *undecidable*, where

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

问: How to prove that A_{TM} is *recognizable*?
答: construct a Turing machine 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*."

问: Why is A_{TM} undecidable?
答: M may loop on w

问: How to formally prove the undecidability?
答: Proof

定理

The language A_{TM} is *recognizable* but *undecidable*, where

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

问: How to prove that A_{TM} is *recognizable*?
答: construct a Turing machine 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*."

问: Why is A_{TM} undecidable?
答: M may loop on w

问: How to formally prove the undecidability?
答: Proof

定理

The language A_{TM} is *recognizable* but *undecidable*, where

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

问: How to prove that A_{TM} is *recognizable*?
答: construct a Turing machine 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*.”

问: Why is A_{TM} undecidable?
答: M may loop on w

问: How to formally prove the undecidability?
答: Proof

定理

The language A_{TM} is *recognizable* but *undecidable*, where

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

问: How to prove that A_{TM} is *recognizable*?
答: construct a Turing machine 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*.”

问: Why is A_{TM} undecidable?
答: M may loop on w

问: How to formally prove the undecidability?
答: ▸ Proof

Outline

问: How to prove the undecidability of a language?
答: Use the technique called Diagonalization (对角论证法)

问: The origin of Diagonalization?
答: Discovered by mathematician Georg Cantor in 1873

- used in the problem of *measuring* the *sizes* of *infinite sets*
  - i.e., how to compare the relative size of 2 infinite sets?
- Idea: two finite sets have the *same size* if the elements of one set can be *paired* with the elements of the other set.

问: How to prove the undecidability of a language?
答: Use the technique called Diagonalization (对角论证法)

问: The origin of Diagonalization?
答: Discovered by mathematician Georg Cantor in 1873

- used in the problem of *measuring* the *sizes* of *infinite sets*
  - i.e., how to compare the relative size of 2 infinite sets?
- Idea: two finite sets have the *same size* if the elements of one set can be *paired* with the elements of the other set.

定义: one-to-one, onto, same size, correspondence

Assume that we have sets A and B and a function f from A to B. Say that f is

- *one-to-one*, or *injective*: $f(a) \neq f(b)$ whenever $a \neq b$
- *onto*, or *surjective*: for every $b \in B$, there is an $a \in A$ such that $f(a) = b$
- *correspondence*, or *bijective*: both *one-to-one* and *onto*

Say that A and B have the *same size* if there is a *correspondence* function $f : A \to B$.

例: same size

Let $\mathcal{N}$ be the set of natural numbers $\{1, 2, 3, \dots\}$
Let $\mathcal{E}$ be the set of even natural numbers $\{2, 4, 6, \dots\}$
$\mathcal{N}$ and $\mathcal{E}$ have the same size

定义: one-to-one, onto, same size, correspondence

Assume that we have sets A and B and a function f from A to B. Say that f is

- *one-to-one*, or *injective*: $f(a) \neq f(b)$ whenever $a \neq b$
- *onto*, or *surjective*: for every $b \in B$, there is an $a \in A$ such that $f(a) = b$
- *correspondence*, or *bijective*: both *one-to-one* and *onto*

Say that A and B have the *same size* if there is a *correspondence* function $f : A \to B$.

例: same size

Let $\mathcal{N}$ be the set of natural numbers $\{1, 2, 3, \dots\}$
Let $\mathcal{E}$ be the set of even natural numbers $\{2, 4, 6, \dots\}$
$\mathcal{N}$ and $\mathcal{E}$ have the same size

定义: countable

A set A is countable if either it is *finite or* it has the *same size* as $\mathcal{N}$

例

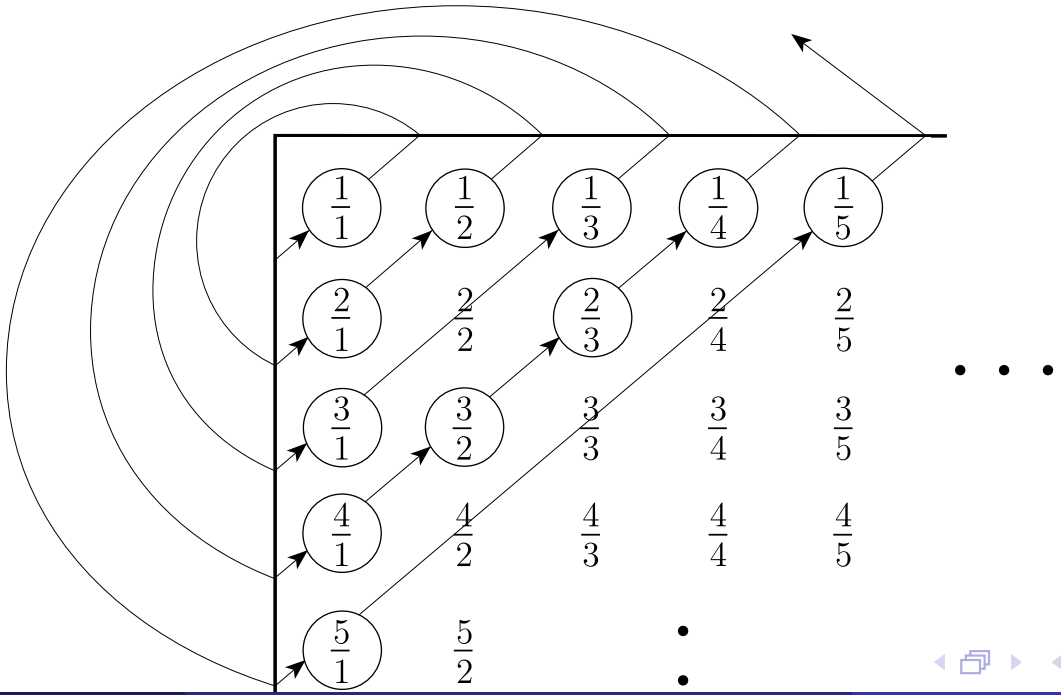
Let $Q = \{\frac{m}{n} \mid m, n \in \mathcal{N}\}$, Q is *countable*

定义: countable

A set A is countable if either it is *finite or* it has the *same size* as $\mathcal{N}$

例

Let $Q = \{\frac{m}{n} \mid m, n \in \mathcal{N}\}$, Q is *countable*



定理: uncountable of $\mathcal{R}$

$\mathcal{R}$ is uncountable,

where $\mathcal{R}$ is the set of real numbers, e.g., $\pi = 3.14\ldots$, $\sqrt{2} = 1.414\ldots$

定理: uncountable of $\mathcal{R}$

$\mathcal{R}$ is uncountable,

where $\mathcal{R}$ is the set of real numbers, e.g., $\pi = 3.14\ldots$, $\sqrt{2} = 1.414\ldots$

证明: (Proof by Contradiction)

定理: uncountable of $\mathcal{R}$

$\mathcal{R}$ is uncountable,

where $\mathcal{R}$ is the set of real numbers, e.g., $\pi = 3.14\ldots$, $\sqrt{2} = 1.414\ldots$

证明: (Proof by Contradiction)

- *Suppose* that a correspondence f existed between $\mathcal{N}$ and $\mathcal{R}$

定理: uncountable of $\mathcal{R}$

$\mathcal{R}$ is uncountable,
where $\mathcal{R}$ is the set of real numbers, e.g., $\pi = 3.14\ldots$, $\sqrt{2} = 1.414\ldots$

证明: (Proof by Contradiction)

- *Suppose* that a correspondence f existed between $\mathcal{N}$ and $\mathcal{R}$
- *Find an* x in $\mathcal{R}$ that is *not paired* with anything in $\mathcal{N}$

定理: uncountable of $\mathcal{R}$

$\mathcal{R}$ is uncountable,
where $\mathcal{R}$ is the set of real numbers, e.g., $\pi = 3.14\ldots$, $\sqrt{2} = 1.414\ldots$

证明: (Proof by Contradiction)

- *Suppose* that a correspondence f existed between $\mathcal{N}$ and $\mathcal{R}$
- In other words, ensure that $x \neq f(n)$ for any n

定理: uncountable of $\mathcal{R}$

$\mathcal{R}$ is uncountable,
where $\mathcal{R}$ is the set of real numbers, e.g., $\pi = 3.14\ldots$, $\sqrt{2} = 1.414\ldots$

证明: (Proof by Contradiction)

- *Suppose* that a correspondence f existed between $\mathcal{N}$ and $\mathcal{R}$
- Let the *nth fractional digit* of x be anything different from the *nth fractional digit* of $f(n)$

n	$f(n)$
1	3.14159...
2	55.55555...
3	0.12345...
4	0.50000...
⋮	⋮

$x = 0.4641\ldots$

定理: uncountable of $\mathcal{R}$

$\mathcal{R}$ is uncountable,
where $\mathcal{R}$ is the set of real numbers, e.g., $\pi = 3.14\ldots$, $\sqrt{2} = 1.414\ldots$

证明: (Proof by Contradiction)

- *Suppose* that a correspondence f existed between $\mathcal{N}$ and $\mathcal{R}$
- So x is different from any $f(n)$

定理: uncountable of $\mathcal{R}$

$\mathcal{R}$ is uncountable,
where $\mathcal{R}$ is the set of real numbers, e.g., $\pi = 3.14\ldots$, $\sqrt{2} = 1.414\ldots$

证明: (Proof by Contradiction)

- *Suppose* that a correspondence f existed between $\mathcal{N}$ and $\mathcal{R}$
- So x is different from any $f(n)$
- Contradiction

推论

Some languages are not Turing-recognizable.

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

- Suppose that every language are Turing-recognizable

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

- Suppose that every language are Turing-recognizable
- The set of *all Turing-machines* has at least the *same size* with the set of *all languages*

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

- Suppose that every language are Turing-recognizable
- The set of *all Turing-machines* has at least the *same size* with the set of *all languages*
- Proof by Contradiction:
  1. The set of *all Turing-machines* is *countable*
  2. The set of *all languages* is *uncountable*

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(1) Prove: The set of *all Turing-machines* is *countable*

- Because each *Turing machine* M has an *encoding* into a *string* $\langle M \rangle$
- The set of all *strings* Σ^* is *countable* for any alphabet Σ
  - We may form a list of Σ^* by writing down all strings of length 0, length 1, length 2, and so on
- (1) is Proved

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(1) Prove: The set of *all Turing-machines* is *countable*

- Because each *Turing machine* M has an *encoding* into a *string* $\langle M \rangle$
- The set of all *strings* Σ^* is *countable* for any alphabet Σ
  - We may form a list of Σ^* by writing down all strings of length 0, length 1, length 2, and so on
- (1) is Proved

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(1) Prove: The set of *all Turing-machines* is *countable*

- Because each *Turing machine* M has an *encoding* into a *string* $\langle M \rangle$
- The set of all *strings* Σ^* is *countable* for any alphabet Σ
  - We may form a list of Σ^* by writing down all strings of length 0, length 1, length 2, and so on
- (1) is Proved

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(1) Prove: The set of *all Turing-machines* is *countable*

- Because each *Turing machine* M has an *encoding* into a *string* $\langle M \rangle$
- The set of all *strings* Σ^* is *countable* for any alphabet Σ
  - We may form a list of Σ^* by writing down all strings of length 0, length 1, length 2, and so on
- (1) is Proved

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2) Prove: The set of *all languages*, denoted as $\mathcal{L}$ over Σ, is *uncountable*

- Let $\mathcal{B}$ be the set of all *infinite binary sequences*.
  - An *infinite binary sequence* is an unending sequence of 0s and 1s.
- Prove:
  - (2.1) $\mathcal{B}$ is *uncountable, similar to the Theorem*
  - (2.2) *Prove Obligation*: There is a correspondence $f : \mathcal{L} \to \mathcal{B}$,
    - i.e., each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2) Prove: The set of *all languages*, denoted as $\mathcal{L}$ over Σ, is *uncountable*

- Let $\mathcal{B}$ be the set of all *infinite binary sequences*.
  - An *infinite binary sequence* is an unending sequence of 0s and 1s.
- Prove:
  - (2.1) $\mathcal{B}$ is *uncountable*, *similar to the Theorem*
  - (2.2) *Prove Obligation*: There is a correspondence $f : \mathcal{L} \to \mathcal{B}$,
    - i.e., each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2) Prove: The set of *all languages*, denoted as $\mathcal{L}$ over Σ, is *uncountable*

- Let $\mathcal{B}$ be the set of all *infinite binary sequences*.
  - An *infinite binary sequence* is an unending sequence of 0s and 1s.
- Prove:
  - (2.1) $\mathcal{B}$ is *uncountable*, similar to the *Theorem*
  - (2.2) *Prove Obligation*: There is a correspondence $f : \mathcal{L} \to \mathcal{B}$,
    - i.e., each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2) Prove: The set of *all languages*, denoted as $\mathcal{L}$ over Σ, is *uncountable*

- Let $\mathcal{B}$ be the set of all *infinite binary sequences*.
  - An *infinite binary sequence* is an unending sequence of 0s and 1s.
- Prove:
  - (2.1) $\mathcal{B}$ is *uncountable*, *similar to the Theorem*
  - (2.2) *Prove Obligation*: There is a correspondence $f : \mathcal{L} \to \mathcal{B}$,
    - i.e., each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2) Prove: The set of *all languages*, denoted as $\mathcal{L}$ over Σ, is *uncountable*

- Let $\mathcal{B}$ be the set of all *infinite binary sequences*.
  - An *infinite binary sequence* is an unending sequence of 0s and 1s.
- Prove:
  - (2.1) $\mathcal{B}$ is *uncountable*, *similar to the Theorem*
  - (2.2) *Prove Obligation*: There is a correspondence $f : \mathcal{L} \to \mathcal{B}$,
    - i.e., each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2) Prove: The set of *all languages*, denoted as $\mathcal{L}$ over Σ, is *uncountable*

- Let $\mathcal{B}$ be the set of all *infinite binary sequences*.
  - An *infinite binary sequence* is an unending sequence of 0s and 1s.
- Prove:
  - (2.1) $\mathcal{B}$ is *uncountable*, *similar to the Theorem*
  - (2.2) *Prove Obligation*: There is a correspondence $f : \mathcal{L} \to \mathcal{B}$,
    - i.e., each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2.2) Prove Obligation: Each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2.2) Prove Obligation: Each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$
Prove: Let $\Sigma^* = \{s_1, s_2, s_3, \dots\}$:

- The ith bit of that sequence (*characteristic sequence* of A, χ_A) is
  - a 1 if $s_i \in A$
  - a 0 if $s_i \notin A$

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2.2) Prove Obligation: Each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$
Prove: Let $\Sigma^* = \{s_1, s_2, s_3, \dots\}$:

- The ith bit of that sequence (*characteristic sequence* of A, χ_A) is
  - a 1 if $s_i \in A$
  - a 0 if $s_i \notin A$

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2.2) Prove Obligation: Each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$
Prove: Let $\Sigma^* = \{s_1, s_2, s_3, \dots\}$: *For simplicity, if* $\Sigma = \{0, 1\}$

- The ith bit of that sequence (*characteristic sequence* of A, χ_A) is
  - a 1 if $s_i \in A$
  - a 0 if $s_i \notin A$

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

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

(2) Prove: The set of *all languages*, denoted as $\mathcal{L}$ over Σ, is *uncountable*

- Let $\mathcal{B}$ be the set of all *infinite binary sequences*.
  - An *infinite binary sequence* is an unending sequence of 0s and 1s.
- Prove:
  - (2.1) $\mathcal{B}$ is *uncountable*, *similar to the Theorem*
  - (2.2) *Prove Obligation*: There is a correspondence $f : \mathcal{L} \to \mathcal{B}$,
    - i.e., each language $A \in \mathcal{L}$ has a unique sequence in $\mathcal{B}$

推论

Some languages are not Turing-recognizable.

证明: (Proof by Contradiction)

- Proof by Contradiction:
  1. The set of *all Turing-machines* is *countable*
  2. The set of *all languages* is *uncountable*

Proved

定理: Undecidability of A_{TM}

◂ Recall

The language A_{TM} is *undecidable*, where

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

证明: (Proof by Contradiction)

定理: Undecidability of A_{TM} ◂ Recall

The language A_{TM} is *undecidable*, where

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

证明: (Proof by Contradiction)

- Suppose that H is a decider for A_{TM}, where

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

定理: Undecidability of A_{TM} ◂ Recall

The language A_{TM} is *undecidable*, where

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

证明: (Proof by Contradiction)

- Suppose that H is a decider for A_{TM}, where

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

- Construct a TM D using H

$D=$ "On *input* $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the *opposite* of what H outputs. That is, if H accepts, *reject*; and if H rejects, *accept*."

定理: Undecidability of A_{TM} ◂ Recall

The language A_{TM} is *undecidable*, where

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

证明: (Proof by Contradiction)

- Suppose that H is a decider for A_{TM}, where

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

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

定理: Undecidability of A_{TM} ◂ Recall

The language A_{TM} is *undecidable*, where

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

证明: (Proof by Contradiction)

- Suppose that H is a decider for A_{TM}, where

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

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

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

定理: Undecidability of A_{TM}

◂ Recall

The language A_{TM} is *undecidable*, where

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

证明: (Proof by Contradiction)

- Suppose that H is a decider for A_{TM}, where

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

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

D rejects $\langle D\rangle$ exactly when D accepts $\langle D\rangle$

定理: Undecidability of A_{TM}

◂ Recall

The language A_{TM} is *undecidable*, where

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

证明: (Proof by Contradiction)

- Suppose that H is a decider for A_{TM}, where

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

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

D rejects $\langle D\rangle$ exactly when D accepts $\langle D\rangle$

Contradiction. Proved.

定理: Undecidability of A_{TM}

◂ Recall

The language A_{TM} is *undecidable*, where

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

分析: Diagonalization:

定理: Undecidability of A_{TM}

Recall

The language A_{TM} is *undecidable*, where

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

分析: Diagonalization:

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

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

定理: Undecidability of A_{TM}

Recall

The language A_{TM} is *undecidable*, where

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

分析: Diagonalization:

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

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

定理: Undecidability of A_{TM}

◂ Recall

The language A_{TM} is *undecidable*, where

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

分析: Diagonalization: D computes the *opposite* of the *diagonal* entries

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$	$\langle D\rangle$	$\cdots$
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*		?	
$\vdots$		$\vdots$					$\ddots$

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

定理: Undecidability of A_{TM} ◂ Recall

The language A_{TM} is *undecidable*, where

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

分析: Diagonalization: D computes the *opposite* of the *diagonal* entries

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\langle M_4\rangle$	$\cdots$	$\langle D\rangle$	$\cdots$
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*		?	
$\vdots$		$\vdots$					$\ddots$

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

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

问: Besides undecidable, is there a language *even not Turing-recognizable*?
答: Yes

问: How to prove it?
答: Define *co-Turing-recognizable* first.

定义: co-Turing-recognizable

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

问: Then?
答: Show a Theorem (加下页), and then prove by using the Theorem

问: Besides undecidable, is there a language *even not Turing-recognizable*?
答: Yes

问: How to prove it?
答: Define *co-Turing-recognizable* first.

定义: co-Turing-recognizable

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

问: Then?
答: Show a Theorem (加下页), and then prove by using the Theorem

问: Besides undecidable, is there a language *even not Turing-recognizable*?
答: Yes

问: How to prove it?
答: Define *co-Turing-recognizable* first.

定义: co-Turing-recognizable

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

问: Then?
答: Show a Theorem (加下页), and then prove by using the Theorem

定理

A language is *decidable*, iff it is Turing-recognizable and co-Turing-recognizable.

定理

A language is *decidable*, iff it is Turing-recognizable and co-Turing-recognizable.

证明 1: If A is decidable

Let M be the decider for A

- If $w \in A$, M accepts w, so A is Turing-recognizable
- If $w \in \overline{A}$, i.e., $w \notin A$, M rejects A, so A is co-Turing-recognizable

定理

A language is *decidable*, iff it is Turing-recognizable and co-Turing-recognizable.

证明 1: If A is decidable

Let M be the decider for A

- If $w \in A$, M accepts w, so A is Turing-recognizable
- If $w \in \overline{A}$, i.e., $w \notin A$, M rejects A, so A is co-Turing-recognizable

定理

A language is *decidable*, iff it is Turing-recognizable and co-Turing-recognizable.

证明 1: If A is decidable

Let M be the decider for A

- If $w \in A$, M accepts w, so A is Turing-recognizable
- If $w \in \overline{A}$, i.e., $w \notin A$, M rejects A, so A is co-Turing-recognizable

定理

A language is *decidable*, iff it is Turing-recognizable and co-Turing-recognizable.

证明 2: If A is Turing-recognizable and co-Turing-recognizable

Let M_1 be the recognizer for A
Let M_2 be the recognizer for $\overline{A}$

定理

A language is *decidable*, iff it is Turing-recognizable and co-Turing-recognizable.

证明 2: If A is Turing-recognizable and co-Turing-recognizable

Let M_1 be the recognizer for A
Let M_2 be the recognizer for $\overline{A}$
Construct the TM M:

定理

A language is *decidable*, iff it is Turing-recognizable and co-Turing-recognizable.

证明 2: If A is Turing-recognizable and co-Turing-recognizable

Let M_1 be the recognizer for A
Let M_2 be the recognizer for $\overline{A}$
Construct the TM M:

$M =$ "On input w:

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

定理

A language is *decidable*, iff it is Turing-recognizable and co-Turing-recognizable.

推论

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

证明: (Proof by Contradiction)

- A_{TM} is Turing-recognizable
- If $\overline{A_{\mathrm{TM}}}$ is Turing-recognizable
  - A_{TM} is co-Turing-recognizable
  - A_{TM} is decidable
  - Contradiction. Proved

定理

A language is *decidable*, iff it is Turing-recognizable and co-Turing-recognizable.

推论

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

证明: (Proof by Contradiction)

- A_{TM} is Turing-recognizable
- If $\overline{A_{\mathrm{TM}}}$ is Turing-recognizable
  - A_{TM} is co-Turing-recognizable
  - A_{TM} is decidable
  - Contradiction. Proved

定理

A language is *decidable*, iff it is Turing-recognizable and co-Turing-recognizable.

推论

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

证明: (Proof by Contradiction)

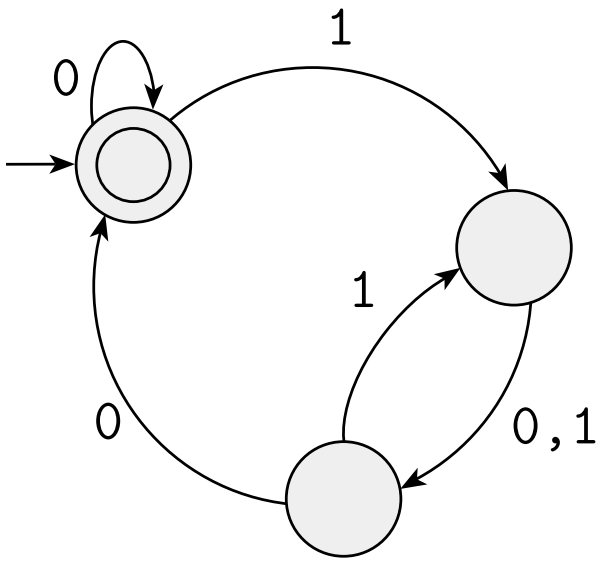
- A_{TM} is Turing-recognizable
- If $\overline{A_{\mathrm{TM}}}$ is Turing-recognizable
  - A_{TM} is co-Turing-recognizable
  - A_{TM} is decidable
  - Contradiction. Proved

总结

- 定义:
  - one-to-one, onto, same size, correspondence
  - countable , co-Turing-recognizable
- 定理:
  - uncountable of $\mathcal{R}$
  - decidable $\equiv$ Turing-recognizable $\wedge$ co-Turing-recognizable
- 推论:
  - Some languages are not Turing-recognizable
- Decidable:
  - A_{DFA}, A_{NFA}, A_{REX}, E_{DFA}, EQ_{DFA}
  - A_{CFG}, E_{CFG}, every context-free language
- Recognizable: A_{TM}
- Undecidable: EQ_{CFG} (Not proved yet), A_{TM}
- Turing-unrecognizable: $\overline{A_{\mathrm{TM}}}$

作业

4.1 Answer all parts for the following DFA M and give reasons for your answers.



a. Is $\langle M, 0100\rangle \in A_{\mathsf{DFA}}$?

b. Is $\langle M, 011\rangle \in A_{\mathsf{DFA}}$?

c. Is $\langle M\rangle \in A_{\mathsf{DFA}}$?

d. Is $\langle M, 0100\rangle \in A_{\mathsf{REX}}$?

e. Is $\langle M\rangle \in E_{\mathsf{DFA}}$?

f. Is $\langle M, M\rangle \in EQ_{\mathsf{DFA}}$?

4.3 Let $ALL_{\mathsf{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \Sigma^*\}$. Show that ALL_{DFA} is decidable.

作业

4.6 Let X be the set $\{1, 2, 3, 4, 5\}$ and Y be the set $\{6, 7, 8, 9, 10\}$. We describe the functions $f: X \longrightarrow Y$ and $g: X \longrightarrow Y$ in the following tables. Answer each part and give a reason for each negative answer.

n	$f(n)$
1	6
2	7
3	6
4	7
5	6

n	$g(n)$
1	10
2	9
3	8
4	7
5	6

[A] **a.** Is f one-to-one?

b. Is f onto?

c. Is f a correspondence?

[A] **d.** Is g one-to-one?

e. Is g onto?

f. Is g a correspondence?

4.7 Let $\mathcal{B}$ be the set of all infinite sequences over $\{0,1\}$. Show that $\mathcal{B}$ is uncountable using a proof by diagonalization.