形式语言与计算复杂性

第 3 章 Computability Theory

3.3 Reducibility

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

3.3 Reducibility

Outline

3.3 Reducibility

Outline

3.3 Reducibility

Introduction

问: Other method of proving undecidability?
答: A primary method, called *Reducibility*

问: What is "reducibility" or "reduction"?
答: A *reduction* is a way of *converting* one *problem* to another *problem* in such a way that a *solution* to the *second problem* can be *used* to solve the *first problem*.

问: An example of *reducing* problem A to problem B?
答:

- A: find your way around a new city; B: obtain a map of the city
- A: traveling from Boston to Paris; B: buying a plane ticket
- A: measuring the area of a rectangle; B: measuring its length and width
- A: solving a system of linear equations; B: the problem of inverting a matrix;

3.3 Reducibility

Introduction

问: Other method of proving undecidability?
答: A primary method, called *Reducibility*

问: What is "reducibility" or "reduction"?
答: A *reduction* is a way of *converting* one *problem* to another *problem* in such a way that a *solution* to the *second problem* can be *used* to solve the *first problem*.

问: An example of *reducing* problem A to problem B?
答:

- A: find your way around a new city; B: obtain a map of the city
- A: traveling from Boston to Paris; B: buying a plane ticket
- A: measuring the area of a rectangle; B: measuring its length and width
- A: solving a system of linear equations; B: the problem of inverting a matrix;

3.3 Reducibility

Introduction

问: Other method of proving undecidability?
答: A primary method, called *Reducibility*

问: What is "reducibility" or "reduction"?
答: A *reduction* is a way of *converting* one *problem* to another *problem* in such a way that a *solution* to the *second problem* can be *used* to solve the *first problem*.

问: An example of *reducing* problem A to problem B?
答:

- A: find your way around a new city; B: obtain a map of the city
- A: traveling from Boston to Paris; B: buying a plane ticket
- A: measuring the area of a rectangle; B: measuring its length and width
- A: solving a system of linear equations; B: the problem of inverting a matrix;

3.3 Reducibility

Introduction

问: Relationship between Reducibility and Decidability:
答: If A is reducible to B, solving A *cannot be harder* than solving B. so

- If A is reducible to B and B is decidable, A also is decidable
- If A is undecidable and reducible to B, B is *undecidable*.

问: How to use reducibility to *prove undecidability* of a problem B?
答:

1. find some other problem A already known to be undecidable, e.g., A_{TM}
2. Prove that A can be reduced to B

定理: Undecidability of A_{TM}

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

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

3.3 Reducibility

Introduction

问: Relationship between Reducibility and Decidability:
答: If A is reducible to B, solving A *cannot be harder* than solving B. so

- If A is reducible to B and B is decidable, A also is decidable
- If A is undecidable and reducible to B, B is *undecidable*.

问: How to use reducibility to *prove undecidability* of a problem B?
答:

1. find some other problem A already known to be undecidable, e.g., A_{TM}
2. Prove that A can be reduced to B

定理: Undecidability of A_{TM}

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

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

3.3 Reducibility

Outline

定理: Undecidability of $HALT_{\mathrm{TM}}$, i.e, Halting problem

$HALT_{\mathrm{TM}}$ is undecidable, where

$$HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

证明: (Proof by Contradiction)

Prove the undecidability of $HALT_{\mathrm{TM}}$ by reducing A_{TM} to $HALT_{\mathrm{TM}}$.

定理: Undecidability of $HALT_{\mathrm{TM}}$, i.e, Halting problem

$HALT_{\mathrm{TM}}$ is undecidable, where

$$HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

证明: (Proof by Contradiction)

Prove the undecidability of $HALT_{\mathrm{TM}}$ by reducing A_{TM} to $HALT_{\mathrm{TM}}$.

That is, assume that we have a TM R that decides $HALT_{\mathrm{TM}}$

Use R to construct S, i.e., a TM that decides A_{TM}

定理: Undecidability of $HALT_{\mathrm{TM}}$, i.e, Halting problem

$HALT_{\mathrm{TM}}$ is undecidable, where

$$HALT_{\mathrm{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$$

证明: (Proof by Contradiction)

Prove the undecidability of $HALT_{\mathrm{TM}}$ by reducing A_{TM} to $HALT_{\mathrm{TM}}$.

That is, assume that we have a TM R that decides $HALT_{\mathrm{TM}}$

Use R to construct S, i.e., a TM that decides A_{TM}

$S =$ "On input $\langle M, w\rangle$, an encoding of a TM M and a string w:

1. Run TM R on input $\langle M, w\rangle$.
2. If R rejects, *reject*.
3. If R accepts, simulate M on w until it halts.
4. If M has accepted, *accept*; if M has rejected, *reject*."

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

First try: run R on input $\langle M \rangle$ and see *whether it accepts*

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

First try: run R on input $\langle M \rangle$ and see *whether it accepts*

- If it does, we know that $L(M)$ is empty and therefore that M does not accept w

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

First try: run R on input $\langle M \rangle$ and see *whether it accepts*

- *But* if R rejects $\langle M \rangle$, all we know is that $L(M)$ is not empty and therefore that M accepts *some string*
  - but we still *do not know* whether M *accepts* the *particular* string w

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

First try: run R on input $\langle M \rangle$ and see *whether it accepts*

- *But* if R rejects $\langle M \rangle$, all we know is that $L(M)$ is not empty and therefore that M accepts *some string*
  - but we still *do not know* whether M *accepts* the *particular* string w

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

First try: run R on input $\langle M \rangle$ and see *whether it accepts*

- *But* if R rejects $\langle M \rangle$, all we know is that $L(M)$ is not empty and therefore that M accepts *some string*
  - but we still *do not know* whether M *accepts* the *particular* string w

Prove failed, try another way!

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

Right way: run R on a *modification* of $\langle M \rangle$, i.e., M_1

- M_1 rejects the input x, if $x \neq w$
- M_1 accepts the input x, if $x = w$ and M *accepts* w

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

Right way: run R on a *modification* of $\langle M \rangle$, i.e., M_1

- M_1 rejects the input x, if $x \neq w$
- M_1 accepts the input x, if $x = w$ and M *accepts* w

So, the language of M_1 will be nonempty,
iff M accepts w,

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

Right way: run R on a *modification* of $\langle M \rangle$, i.e., M_1

- M_1 rejects the input x, if $x \neq w$
- M_1 accepts the input x, if $x = w$ and M *accepts* w

So, the language of M_1 will be nonempty, (i.e., R *rejects* $\langle M_1 \rangle$)
iff M accepts w,

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

Right way: run R on a *modification* of $\langle M \rangle$, i.e., M_1

- M_1 rejects the input x, if $x \neq w$
- M_1 accepts the input x, if $x = w$ and M *accepts* w

So, the language of M_1 will be nonempty, (i.e., R *rejects* $\langle M_1 \rangle$)
iff M accepts w, (i.e., S *accepts* $\langle M, w \rangle$)

定理: Undecidability of E_{TM}

E_{TM} is undecidable, where

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

证明: (Proof by Contradiction)

Assume that E_{TM} is decidable, prove that $\underline{A_{\mathrm{TM}}}$ is decidable

That is, *assume* that we have a TM R that *decides* E_{TM}

Use R to *construct* S, i.e., a TM that *decides* A_{TM}

$M_1 =$ "On input x:

1. If $x \neq w$, *reject*.
2. If $x = w$, run M on input w and *accept* if M does."

$S =$ "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

1. Use the description of M and w to construct the TM M_1 just described.
2. Run R on input $\langle M_1 \rangle$.
3. If R accepts, *reject*; if R rejects, *accept*."

定理: Undecidability of EQ_{TM}

EQ_{TM} is undecidable, where

$$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

定理: Undecidability of EQ_{TM}

EQ_{TM} is undecidable, where

$$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

证明: (Proof by Contradiction)

定理: Undecidability of EQ_{TM}

EQ_{TM} is undecidable, where

$$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

证明: (Proof by Contradiction)

Assume that EQ_{TM} is decidable, prove that $\underline{E_{\mathrm{TM}}}$ is decidable

定理: Undecidability of EQ_{TM}

EQ_{TM} is undecidable, where

$$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

证明: (Proof by Contradiction)

Assume that EQ_{TM} is decidable, prove that $\underline{E_{\mathrm{TM}}}$ is decidable

Let TM R decide EQ_{TM} and construct TM S to decide E_{TM} as follows.

定理: Undecidability of EQ_{TM}

EQ_{TM} is undecidable, where

$$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

证明: (Proof by Contradiction)

Assume that EQ_{TM} is decidable, prove that E_{TM} is decidable

Let TM R decide EQ_{TM} and construct TM S to decide E_{TM} as follows.

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

1. Run R on input $\langle M, M_1\rangle$, where M_1 is a TM that rejects all inputs.
2. If R accepts, *accept*; if R rejects, *reject*."

3.3 Reducibility

Outline

问: Any other undecidable problems?
答: Yes, the *Post Correspondence Problem*, or *PCP*

问: Is PCP useful?
答: Yes, one of the proof for undecidability of first-order logic.

问: Then, what is PCP?
答: Begin with a collection of dominos

问: Any other undecidable problems?
答: Yes, the *Post Correspondence Problem*, or *PCP*

问: Is PCP useful?
答: Yes, one of the proof for undecidability of first-order logic.

问: Then, what is PCP?
答: Begin with a collection of dominos

问: Any other undecidable problems?
答: Yes, the *Post Correspondence Problem*, or *PCP*

问: Is PCP useful?
答: Yes, one of the proof for undecidability of first-order logic.

问: Then, what is PCP?
答: Begin with a collection of dominos

定义: A domino

A domino contains two strings: one on each side.

An individual domino looks like

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]$$

定义: A domino

A domino contains two strings: one on each side.
An individual domino looks like

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]$$

A *collection* of dominos looks like

$$\left\{\left[\frac{\mathtt{b}}{\mathtt{ca}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right], \left[\frac{\mathtt{ca}}{\mathtt{a}}\right], \left[\frac{\mathtt{abc}}{\mathtt{c}}\right]\right\}$$

定义: A domino

A domino contains two strings: one on each side.
An individual domino looks like

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]$$

A *collection* of dominos looks like

$$\left\{\left[\frac{\mathtt{b}}{\mathtt{ca}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right], \left[\frac{\mathtt{ca}}{\mathtt{a}}\right], \left[\frac{\mathtt{abc}}{\mathtt{c}}\right]\right\}$$

定义: Match

A list of these dominos (*repetitions permitted*) so that the string we get by reading off the symbols on the *top* is the *same* as the string of symbols on the *bottom*

定义: A domino

A domino contains two strings: one on each side.
An individual domino looks like

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]$$

A *collection* of dominos looks like

$$\left\{\left[\frac{\mathtt{b}}{\mathtt{ca}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right], \left[\frac{\mathtt{ca}}{\mathtt{a}}\right], \left[\frac{\mathtt{abc}}{\mathtt{c}}\right]\right\}$$

定义: Match

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]\left[\frac{\mathtt{b}}{\mathtt{ca}}\right]\left[\frac{\mathtt{ca}}{\mathtt{a}}\right]\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]\left[\frac{\mathtt{abc}}{\mathtt{c}}\right]$$

定义: A domino

A domino contains two strings: one on each side.

An individual domino looks like

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]$$

A *collection* of dominos looks like

$$\left\{\left[\frac{\mathtt{b}}{\mathtt{ca}}\right], \left[\frac{\mathtt{a}}{\mathtt{ab}}\right], \left[\frac{\mathtt{ca}}{\mathtt{a}}\right], \left[\frac{\mathtt{abc}}{\mathtt{c}}\right]\right\}$$

定义: Match

$$\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]\left[\frac{\mathtt{b}}{\mathtt{ca}}\right]\left[\frac{\mathtt{ca}}{\mathtt{a}}\right]\left[\frac{\mathtt{a}}{\mathtt{ab}}\right]\left[\frac{\mathtt{abc}}{\mathtt{c}}\right] \qquad \begin{array}{|cccccccc|} \mathtt{a} & \mathtt{b} & \mathtt{c} & \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{b} & \mathtt{c} \\ \mathtt{a} & \mathtt{b} & \mathtt{c} & \mathtt{a} & \mathtt{a} & \mathtt{a} & \mathtt{b} & \mathtt{c} \end{array}$$

定义: An instance of the PCP

An *instance* of the PCP is a collection P of dominos:

$$P = \left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots, \left[\frac{t_k}{b_k}\right] \right\}$$

定理: Undecidability of PCP

PCP is undecidable, where
PCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match*$\}$

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right]$ $\}$

定义: An instance of the PCP

An *instance* of the PCP is a collection P of dominos:

$$P = \left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots, \left[\frac{t_k}{b_k}\right] \right\}$$

定理: Undecidability of PCP

PCP is undecidable, where
PCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match*$\}$

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right]$ $\}$

定义: An instance of the PCP

An *instance* of the PCP is a collection P of dominos:

$$P = \left\{ \left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \cdots, \left[\frac{t_k}{b_k}\right] \right\}$$

定理: Undecidability of PCP

PCP is undecidable, where
PCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match*$\}$

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right]$ $\}$

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $[\frac{t_1}{b_1}]\ \}$

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $[\frac{t_1}{b_1}]$ $\}$

证明 Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right] \}$

证明 Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right] \}$

证明 Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7

Part 1: The construction begins in the following manner

$$\text{Put } \left[\frac{\#}{\#q_0 w_1 w_2 \cdots w_n \#}\right] \text{ into } P' \text{ as the first domino } \left[\frac{t_1}{b_1}\right]$$

where the bottom string $C_1 = q_0 w_1 w_2 \cdots w_n$ is the first configuration in the accepting computation history for M *on* w

定义: MPCP

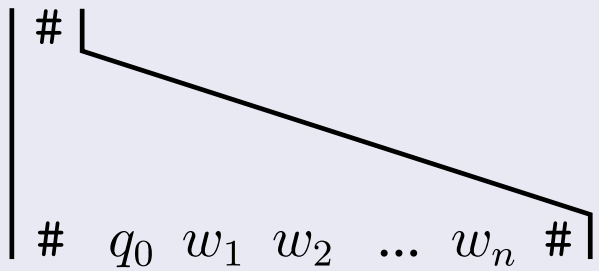
MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right] \}$

证明 Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7

Part 1: The construction begins in the following manner



定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right] \}$

证明 Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7

Part 2: For every $a, b \in \Gamma$ and every $q, r \in Q$ where $q \neq q_{\mathrm{reject}}$

$$\text{if } \delta(q, a) = (r, b, \mathrm{R}), \text{ put } \left[\frac{qa}{br}\right] \text{ into } P'$$

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right] \}$

证明 Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7

Part 3: For every $a, b, c \in \Gamma$ and every $q, r \in Q$ where $q \neq q_{\mathrm{reject}}$

$$\text{if } \delta(q, a) = (r, b, \mathrm{L}), \text{ put } \left[\frac{cqa}{rcb}\right] \text{ into } P'$$

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right] \}$

证明 Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7

Part 4: For every $a \in \Gamma$,

$$\text{put } \left[\frac{a}{a}\right] \text{ into } P'$$

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right] \}$

证明 Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7

Part 5:

$$\text{Put } \left[\frac{\#}{\#}\right] \text{ and } \left[\frac{\#}{\sqcup\#}\right] \text{ into } P'$$

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right] \}$

证明 Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7

Part 6: For every $a \in \Gamma$,

$$\text{put } \left[\frac{a\ q_{\text{accept}}}{q_{\text{accept}}}\right] \text{ and } \left[\frac{q_{\text{accept}}\ a}{q_{\text{accept}}}\right] \text{ into } P'$$

定义: MPCP

MPCP $= \{\langle P \rangle \mid P$ is an *instance* of the Post Correspondence Problem with a *match that starts with the first domino*, i.e., $\left[\frac{t_1}{b_1}\right] \}$

证明 Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7

Part 7: Finally, we add the domino

$$\left[\frac{q_{\mathrm{accept}}\#\#}{\#}\right]$$

Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$

Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$

Part1: places the domino in P'

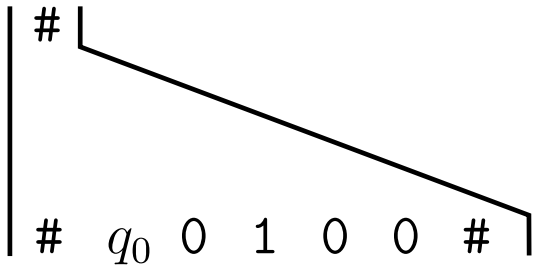
$$\left[\frac{\#}{\#q_0 0100\#}\right] = \left[\frac{t_1}{b_1}\right]$$

Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$

Part1: places the domino in P' and the match begins:

$$\left[\frac{\#}{\#q_0 0100\#}\right] = \left[\frac{t_1}{b_1}\right]$$



Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$

Part1: places the domino in P'

$$\left[\frac{\#}{\#q_0 0100\#}\right] = \left[\frac{t_1}{b_1}\right]$$

Part 2: places the domino in P'

$$\left[\frac{q_0 0}{2 q_7}\right]$$

Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$

Part1: places the domino in P'

$$\left[\frac{\#}{\#q_0 0100\#}\right] = \left[\frac{t_1}{b_1}\right]$$

Part 2: places the domino in P'

$$\left[\frac{q_0 0}{2 q_7}\right]$$

Part 4: places the dominos:

$$\left[\frac{0}{0}\right], \left[\frac{1}{1}\right], \left[\frac{2}{2}\right], \text{ and } \left[\frac{\sqcup}{\sqcup}\right]$$

Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$

Part1: places the domino in P'

$$\left[\frac{\#}{\#q_0 0100\#}\right] = \left[\frac{t_1}{b_1}\right]$$

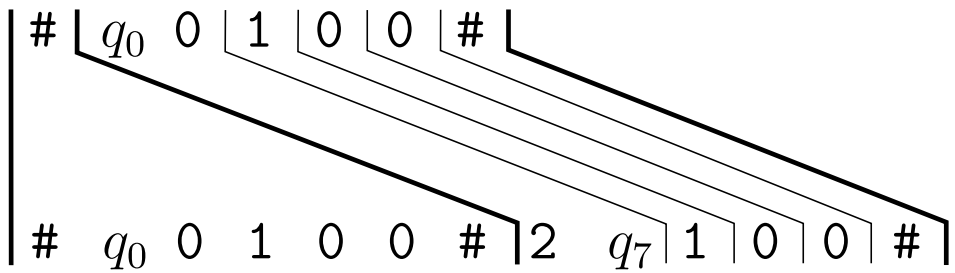
Part 2: places the domino in P'

$$\left[\frac{q_0 0}{2 q_7}\right]$$

Part 4: places the dominos:

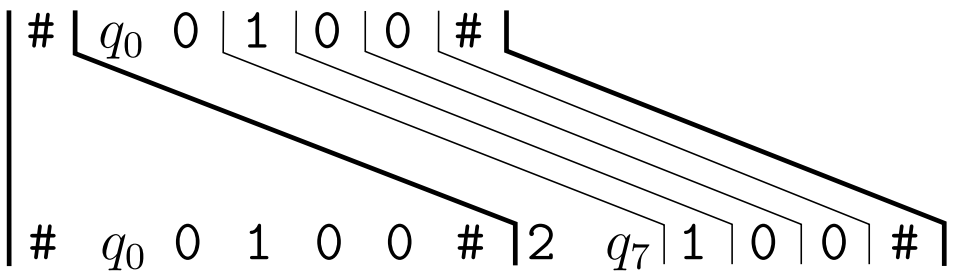
$$\left[\frac{0}{0}\right], \left[\frac{1}{1}\right], \left[\frac{2}{2}\right], \text{ and } \left[\frac{\sqcup}{\sqcup}\right]$$

Together with *Part 5*, that allows us to extend the match to



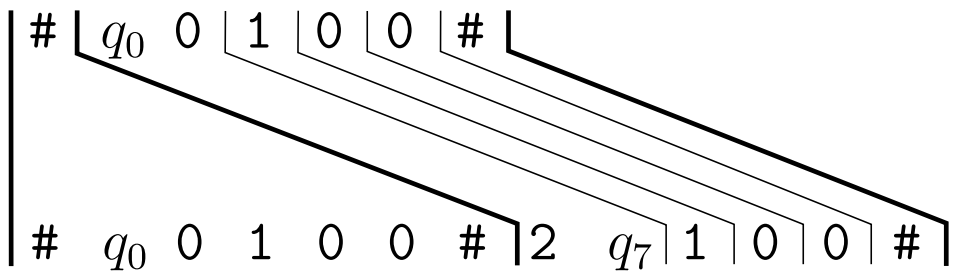
Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$



Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

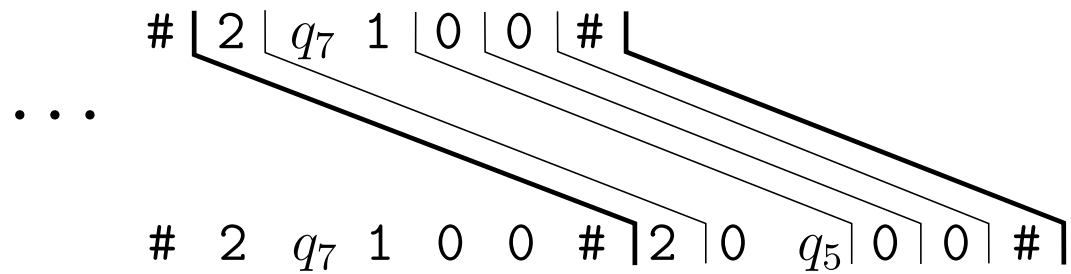
例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$



Let $\delta(q_7, 1) = (q_5, 0, R)$

$\left[\frac{q_7 1}{0 q_5}\right]$ in P'

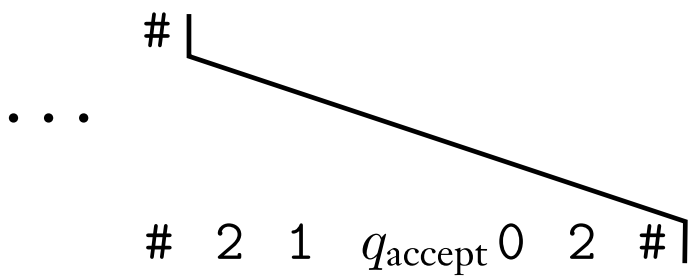
the latest partial match extends to



Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$

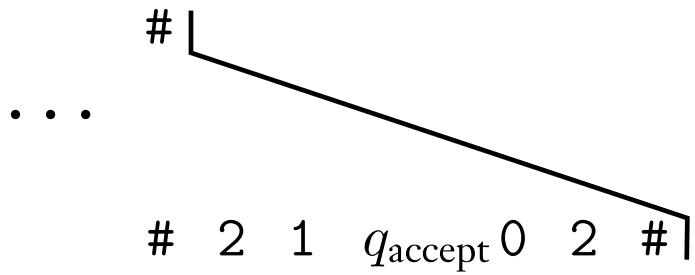
If the partial match up to the point when the machine halts in the accept state is



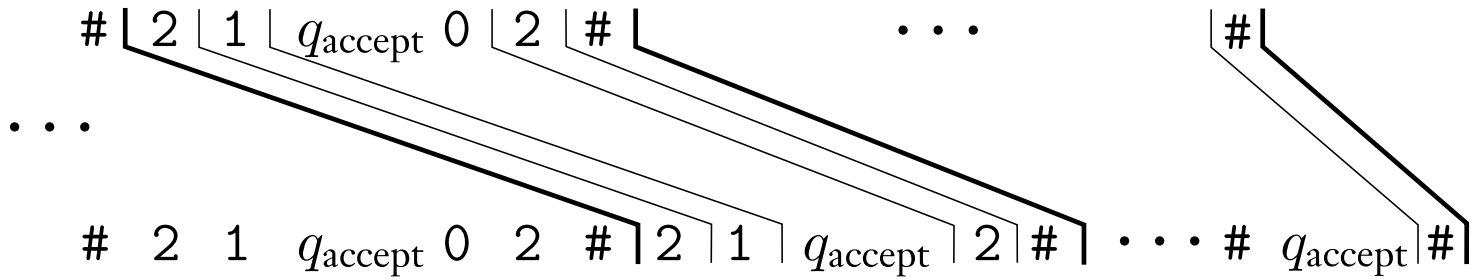
Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$

If the partial match up to the point when the machine halts in the accept state is

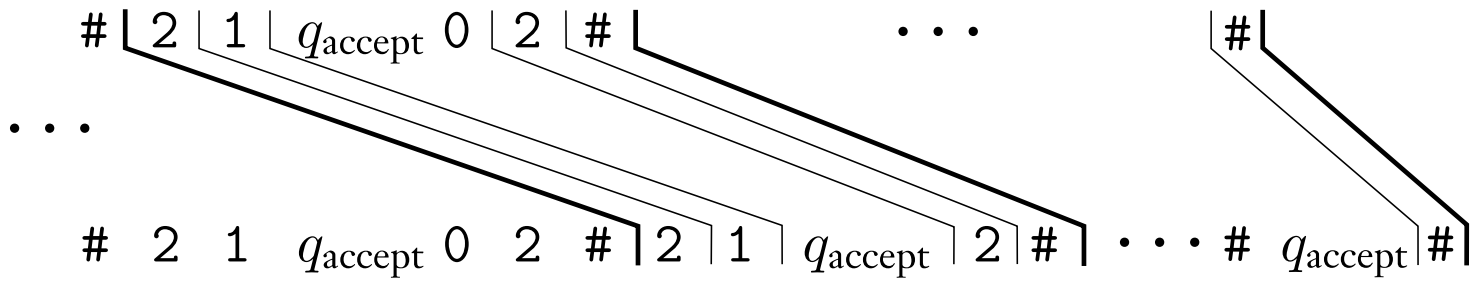


The dominos we added in *Part 6* allow the match to continue



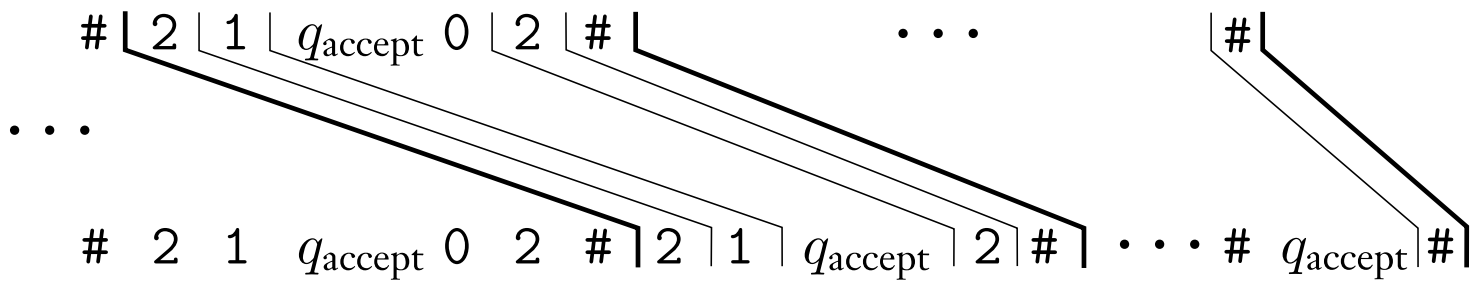
Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$

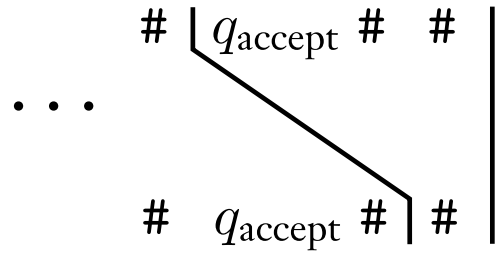


Links: TM R, configuration, A_{TM}, Part of P' 1 2 3 4 5 6 7

例: Let $\Gamma = \{0, 1, 2, \sqcup\}$, w is the string 0100, $\delta(q_0, 0) = (q_7, 2, R)$



The dominos we added in *Part 7* allow the match to continue



证明: Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7
   - So, P' is an instance of the MPCP whereby the match simulates the computation of M on w
2. Convert P' to P, an instance of the PCP that still simulates M on w

证明: Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7
   - So, P' is an instance of the MPCP whereby the match simulates the computation of M on w
2. Convert P' to P, an instance of the PCP that still simulates M on w

证明: Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7
   - So, P' is an instance of the MPCP whereby the match simulates the computation of M on w
2. Convert P' to P, an instance of the PCP that still simulates M on w

证明: Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7
   - So, P' is an instance of the MPCP whereby the match simulates the computation of M on w
2. Convert P' to P, an instance of the PCP that still simulates M on w

Let $u = u_1 u_2 \cdots u_n$ be any string of length n. Define $\star u$, $u\star$, and $\star u\star$ to be the three strings

$$\begin{aligned} \star u &= *u_1 * u_2 * u_3 * \cdots * u_n \\ u\star &= u_1 * u_2 * u_3 * \cdots * u_n * \\ \star u\star &= *u_1 * u_2 * u_3 * \cdots * u_n * \end{aligned}$$

证明: Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7
   - So, P' is an instance of the MPCP whereby the match simulates the computation of M on w
2. Convert P' to P, an instance of the PCP that still simulates M on w

$$\begin{aligned} \star u &= *u_1*u_2*u_3*\cdots*u_n \\ u\star &= u_1*u_2*u_3*\cdots*u_n* \\ \star u\star &= *u_1*u_2*u_3*\cdots*u_n* \end{aligned}$$

If P' were the collection:

$$\left\{\left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \left[\frac{t_3}{b_3}\right], \cdots, \left[\frac{t_k}{b_k}\right]\right\}$$

证明: Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7
   - So, P' is an instance of the MPCP whereby the match simulates the computation of M on w
2. Convert P' to P, an instance of the PCP that still simulates M on w

$$\begin{aligned} \star u &= *u_1*u_2*u_3*\cdots*u_n \\ u\star &= u_1*u_2*u_3*\cdots*u_n* \\ \star u\star &= *u_1*u_2*u_3*\cdots*u_n* \end{aligned}$$

If P' were the collection: $\left\{\left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \left[\frac{t_3}{b_3}\right], \cdots, \left[\frac{t_k}{b_k}\right]\right\}$

Let P be the collection: $\left\{\left[\frac{\star t_1}{\star b_1\star}\right], \left[\frac{\star t_1}{b_1\star}\right], \left[\frac{\star t_2}{b_2\star}\right], \left[\frac{\star t_3}{b_3\star}\right], \cdots, \left[\frac{\star t_k}{b_k\star}\right], \left[\frac{*\diamond}{\diamond}\right]\right\}$

证明: Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7
   - So, P' is an instance of the MPCP whereby the match simulates the computation of M on w
2. Convert P' to P, an instance of the PCP that still simulates M on w

$$\begin{aligned} \star u &= *u_1*u_2*u_3*\cdots*u_n \\ u\star &= u_1*u_2*u_3*\cdots*u_n* \\ \star u\star &= *u_1*u_2*u_3*\cdots*u_n* \end{aligned}$$

If P' were the collection: $\left\{\left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \left[\frac{t_3}{b_3}\right], \cdots, \left[\frac{t_k}{b_k}\right]\right\}$

Let P be the collection: $\left\{\left[\frac{\star t_1}{\star b_1\star}\right], \left[\frac{\star t_1}{b_1\star}\right], \left[\frac{\star t_2}{b_2\star}\right], \left[\frac{\star t_3}{b_3\star}\right], \cdots, \left[\frac{\star t_k}{b_k\star}\right], \left[\frac{*\diamond}{\diamond}\right]\right\}$

the only domino of the *first* possible match $\left[\frac{\star t_1}{\star b_1\star}\right]$

the only domino of the *last* possible match $\left[\frac{*\diamond}{\diamond}\right]$

证明: Undecidability of PCP

Let TM R decide the MPCP and construct S deciding A_{TM}

1. S constructs an instance P' of the MPCP: 1 2 3 4 5 6 7
   - So, P' is an instance of the MPCP whereby the match simulates the computation of M on w
2. Convert P' to P, an instance of the PCP that still simulates M on w

$$\begin{aligned} \star u &= *u_1*u_2*u_3*\cdots*u_n \\ u\star &= u_1*u_2*u_3*\cdots*u_n* \\ \star u\star &= *u_1*u_2*u_3*\cdots*u_n* \end{aligned}$$

If P' were the collection: $\left\{\left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \left[\frac{t_3}{b_3}\right], \cdots, \left[\frac{t_k}{b_k}\right]\right\}$

Let P be the collection: $\left\{\left[\frac{\star t_1}{\star b_1\star}\right], \left[\frac{\star t_1}{b_1\star}\right], \left[\frac{\star t_2}{b_2\star}\right], \left[\frac{\star t_3}{b_3\star}\right], \cdots, \left[\frac{\star t_k}{b_k\star}\right], \left[\frac{*\diamond}{\diamond}\right]\right\}$

Now Proved

Outline

问: How to use reducibility in more refined way?
答: *Formally* define reducibility.

问: How?
答: Several ways, a *simple type* of reducibility called *mapping reducibility*

问: Roughly speaking, what does mapping reducibility (e.g., reduce problem A to problem B) mean?
答: A *computable function exists* that converts instances of problem A to instances of problem B

- If we have such a conversion function, called a *reduction*, we can solve A with a solver for B, by
  - first using the reduction to convert it to an instance of B
  - then applying the solver for B

问: How to use reducibility in more refined way?
答: *Formally* define reducibility.

问: How?
答: Several ways, a *simple type* of reducibility called *mapping reducibility*

问: Roughly speaking, what does mapping reducibility (e.g., reduce problem A to problem B) mean?
答: A *computable function exists* that converts instances of problem A to instances of problem B

- If we have such a conversion function, called a *reduction*, we can solve A with a solver for B, by
  - first using the reduction to convert it to an instance of B
  - then applying the solver for B

问: How to use reducibility in more refined way?
答: *Formally* define reducibility.

问: How?
答: Several ways, a *simple type* of reducibility called *mapping reducibility*

问: Roughly speaking, what does mapping reducibility (e.g., reduce problem A to problem B) mean?
答: A *computable function exists* that converts instances of problem A to instances of problem B

- If we have such a conversion function, called a *reduction*, we can solve A with a solver for B, by
  - first using the reduction to convert it to an instance of B
  - then applying the solver for B

定义: Computable function

A function $f : \Sigma^* \to \Sigma^*$ is a *computable function* if some *Turing machine* M, on every input w, halts with just $f(w)$ on its tape.

例: Arithmetic operations on integers

A machine that takes input $\langle m, n \rangle$ and returns $m + n$

例: Transformations of machine descriptions

One computable function f takes input w and returns the description of a Turing machine $\langle M' \rangle$, if $w = \langle M \rangle$ is an encoding of a Turing machine M.

- M' is a machine that recognizes the same language as M
- The function returns ε if w is not a legal encoding of a Turing machine.

定义: Computable function

A function $f : \Sigma^* \to \Sigma^*$ is a *computable function* if some *Turing machine* M, on every input w, halts with just $f(w)$ on its tape.

例: Arithmetic operations on integers

A machine that takes input $\langle m, n \rangle$ and returns $m + n$

例: Transformations of machine descriptions

One computable function f takes input w and returns the description of a Turing machine $\langle M' \rangle$, if $w = \langle M \rangle$ is an encoding of a Turing machine M.

- M' is a machine that recognizes the same language as M
- The function returns ε if w is not a legal encoding of a Turing machine.

定义: Computable function

A function $f : \Sigma^* \to \Sigma^*$ is a *computable function* if some *Turing machine* M, on every input w, halts with just $f(w)$ on its tape.

例: Arithmetic operations on integers

A machine that takes input $\langle m, n \rangle$ and returns $m + n$

例: Transformations of machine descriptions

One computable function f takes input w and returns the description of a Turing machine $\langle M' \rangle$, if $w = \langle M \rangle$ is an encoding of a Turing machine M.

- M' is a machine that recognizes the same language as M
- The function returns ε if w is not a legal encoding of a Turing machine.

定义: Mapping Reducibility

Language A is *mapping reducible* to language B, written $A \leq_{\mathrm{m}} B$, if there is a *computable function* $f : \Sigma^* \to \Sigma^*$, where for every w,

$$w \in A \Leftrightarrow f(w) \in B$$

The function f is called the *reduction* from A to B

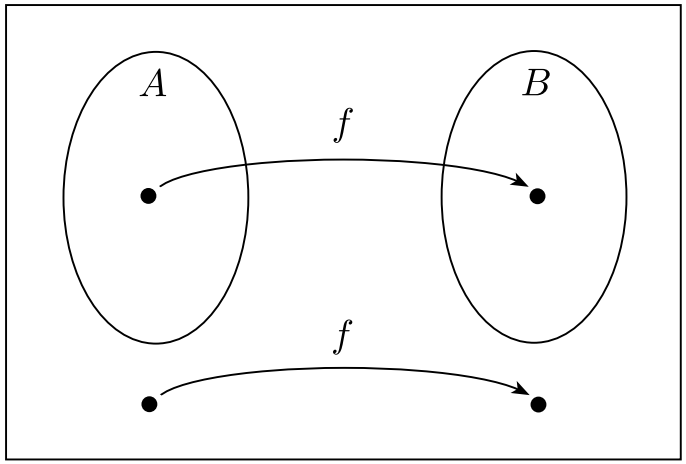
定义: Mapping Reducibility

Language A is *mapping reducible* to language B, written $A \leq_{\mathrm{m}} B$, if there is a *computable function* $f : \Sigma^* \to \Sigma^*$, where for every w,

$$w \in A \Leftrightarrow f(w) \in B$$

The function f is called the *reduction* from A to B

To test whether $w \in A$, we use the reduction f to map w to $f(w)$ and test whether $f(w) \in B$



定理

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable

定理

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable

证明

We let M be the decider for B and f be the reduction from A to B.
We describe a decider N for A as follows.

$N =$ "On input w:

1. Compute $f(w)$.
2. Run M on input $f(w)$ and output whatever M outputs."

定理

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable

推论

If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable

定理

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable

推论

If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable

例:

- $\underline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} \underline{HALT_{\mathrm{TM}}}$ Old proof Def. MapReduce

$F =$ "On input $\langle M, w \rangle$:

1. Construct the following machine M'.
   $M' =$ "On input x:
   1. Run M on x.
   2. If M accepts, *accept*.
   3. If M rejects, enter a loop."
2. Output $\langle M', w \rangle$."

定理

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable

推论

If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable

例:

$A_{\mathrm{TM}} \leq_{\mathrm{m}}$ MPCP, MPCP $\leq_{\mathrm{m}}$ PCP
So, $A_{\mathrm{TM}} \leq_{\mathrm{m}}$ PCP,

定理

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable

推论

If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable

例:

$\underline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} \underline{\overline{E_{\mathrm{TM}}}}$,
$\underline{E_{\mathrm{TM}}} \leq_{\mathrm{m}} \underline{EQ_{\mathrm{TM}}}$

定理

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable

推论

If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable

定理

If $A \leq_{\mathrm{m}} B$ and B is Turing-recognizable, then A is Turing-recognizable

定理

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable

推论

If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable

定理

If $A \leq_{\mathrm{m}} B$ and B is Turing-recognizable, then A is Turing-recognizable

推论

B isn't Turing-recognizable, if $A \leq_{\mathrm{m}} B$ and A isn't Turing-recognizable

定理

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable

推论

If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable

定理

If $A \leq_{\mathrm{m}} B$ and B is Turing-recognizable, then A is Turing-recognizable

推论

B isn't Turing-recognizable, if $A \leq_{\mathrm{m}} B$ and A isn't Turing-recognizable

推论

B isn't Turing-recognizable, if $\underline{A_{TM} \leq_m \overline{B}}$

- Since $\underline{A \leq_{\mathrm{m}} B \Leftrightarrow \overline{A} \leq_{\mathrm{m}} \overline{B}}$, and $\overline{A_{\mathrm{TM}}}$ is not Turing-recognizable

定理

EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

定理

EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

证明

(1) To prove EQ_{TM} is not Turing-recognizable, prove $\underline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} \overline{EQ_{\mathrm{TM}}}$

Def. MapReduce

定理

EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

证明

(1) To prove EQ_{TM} is not Turing-recognizable, prove $\underline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} \overline{EQ_{\mathrm{TM}}}$

Def. MapReduce

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

1. Construct the following two machines, M_1 and M_2.
   $M_1 =$ "On any input:
      1. *Reject*."
   $M_2 =$ "On any input:
      1. Run M on w. If it accepts, *accept*."
2. Output $\langle M_1, M_2 \rangle$."

定理

EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

证明

(2) To prove $\overline{EQ_{\mathrm{TM}}}$ is not Turing-recognizable, prove $\underline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$

Def. MapReduce

定理

EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

证明

(2) To prove $\overline{EQ_{\mathrm{TM}}}$ is not Turing-recognizable, prove $\underline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$

Def. MapReduce

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

1. Construct the following two machines, M_1 and M_2.
   $M_1 =$ "On any input:
      1. *Accept*."
   $M_2 =$ "On any input:
      1. Run M on w.
      2. If it accepts, *accept*."
2. Output $\langle M_1, M_2 \rangle$."

总结

- How to prove undecidability using reducibility?
- Prove undecidability
  - $HALT_{\mathrm{TM}}$, E_{TM}, EQ_{TM}, PCP
- Map Reducibility
  - Prove decidability by Map Reducibility
  - Prove undecidability by Map Reducibility
  - Prove recongnizability by Map Reducibility
  - Prove unrecongnizability by Map Reducibility
  - $A_{\mathrm{TM}} \leq_{\mathrm{m}} HALT_{\mathrm{TM}}$
  - $A_{\mathrm{TM}} \leq_{\mathrm{m}}$ MPCP, MPCP $\leq_{\mathrm{m}}$ PCP, $A_{\mathrm{TM}} \leq_{\mathrm{m}}$ PCP
  - $A_{\mathrm{TM}} \leq_{\mathrm{m}} \overline{E_{\mathrm{TM}}}$, $E_{\mathrm{TM}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$
  - $A_{\mathrm{TM}} \leq_{\mathrm{m}} \overline{EQ_{\mathrm{TM}}}$, $A_{\mathrm{TM}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$
- Turing-unrecognizable: EQ_{TM}, $\overline{EQ_{\mathrm{TM}}}$

作业

5.1 Show that EQ_{CFG} is undecidable.

5.3 Find a match in the following instance of the Post Correspondence Problem.

$$\left\{\left[\frac{\texttt{ab}}{\texttt{abab}}\right], \left[\frac{\texttt{b}}{\texttt{a}}\right], \left[\frac{\texttt{aba}}{\texttt{b}}\right], \left[\frac{\texttt{aa}}{\texttt{a}}\right]\right\}$$

[A]**5.7** Show that if A is Turing-recognizable and $A \leq_{\mathrm{m}} \overline{A}$, then A is decidable.

Turing Machine

A Turing machine is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the *input* alphabet *not* containing the *blank symbol* $\sqcup$
3. Γ is the *tape* alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function.
5. $q_0 \in Q$ is the start state
6. $q_{\text{accept}} \in Q$ is the accept state
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$

定义: Configuration

As a Turing machine computes, changes occur in

- the *current state*
- the *current tape contents*
- the *current head location*

A setting of *these three items* is called a *configuration* of the Turing machine.

例: $1011q_701111$ represents the configuration when the *tape* is 101101111, the *current state* is q_7, and the *head* is currently on the second 0.