Theory of Languages and Automata

Chapter 5 - Reducibility

Sharif University of Technology

Reducibility

- 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.
- If problem A **reduces** to problem B, we can use a solution to B to solve A.

Undecidable Problems

- Let

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

- Then we have

THEOREM 5.1

$HALT_{\mathsf{TM}}$ is undecidable.

PROOF Let's assume for the purposes of obtaining a contradiction that TM R decides $HALT_{\mathsf{TM}}$. We construct TM S to decide A_{TM}, with S operating as follows.

$S =$ "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*."

Clearly, if R decides $HALT_{\mathsf{TM}}$, then S decides A_{TM}. Because A_{TM} is undecidable, $HALT_{\mathsf{TM}}$ also must be undecidable.

Undecidable Problems (continued)

- Let

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

- Then we have

THEOREM 5.2

E_{TM} is undecidable.

PROOF Let's write the modified machine described in the proof idea using our standard notation. We call it M_1.

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."

This machine has the string w as part of its description. It conducts the test of whether $x = w$ in the obvious way, by scanning the input and comparing it character by character with w to determine whether they are the same.

Putting all this together, we assume that TM R decides E_{TM} and construct TM S that decides A_{TM} as follows.

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*."

Undecidable Problems (continued)

- Let

$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}.$

- Then we have

THEOREM 5.3

$REGULAR_{TM}$ is undecidable.

PROOF We let R be a TM that decides $REGULAR_{\mathsf{TM}}$ and construct TM S to decide A_{TM}. Then S works in the following manner.

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

1. Construct the following TM M_2.

   M_2 = "On input x:

   1. If x has the form $0^n 1^n$, *accept*.
   2. If x does not have this form, run M on input w and *accept* if M accepts w."

2. Run R on input $\langle M_2 \rangle$.
3. If R accepts, *accept*; if R rejects, *reject*."

Undecidable Problems (continued)

- Let

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

- Then we have

THEOREM 5.4

EQ_{TM} is undecidable.

PROOF We 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*."

If R decides EQ_{TM}, S decides E_{TM}. But E_{TM} is undecidable by Theorem 5.2, so EQ_{TM} also must be undecidable.

Reduction via Computation Histories

DEFINITION 5.5

Let M be a Turing machine and w an input string. An ***accepting computation history*** for M on w is a sequence of configurations, $C_1, C_2, \ldots, C_l$, where C_1 is the start configuration of M on w, C_l is an accepting configuration of M, and each C_i legally follows from C_{i-1} according to the rules of M. A ***rejecting computation history*** for M on w is defined similarly, except that C_l is a rejecting configuration.

Linear Bounded Auromata

DEFINITION 5.6

A ***linear bounded automaton*** is a restricted type of Turing machine wherein the tape head isn't permitted to move off the portion of the tape containing the input. If the machine tries to move its head off either end of the input, the head stays where it is, in the same way that the head will not move off the left-hand end of an ordinary Turing machine's tape.

Decidable Problems about LBA

- Let

$$A_{\mathsf{LBA}} = \{\langle M, w\rangle \mid M \text{ is an LBA that accepts string } w\}.$$

- Then we have

LEMMA 5.8

Let M be an LBA with q states and g symbols in the tape alphabet. There are exactly qng^n distinct configurations of M for a tape of length n.

PROOF Recall that a configuration of M is like a snapshot in the middle of its computation. A configuration consists of the state of the control, position of the head, and contents of the tape. Here, M has q states. The length of its tape is n, so the head can be in one of n positions, and g^n possible strings of tape symbols appear on the tape. The product of these three quantities is the total number of different configurations of M with a tape of length n.

Decidable Problems about LBA

- Remember that

$$A_{\mathsf{LBA}} = \{\langle M, w\rangle \mid M \text{ is an LBA that accepts string } w\}.$$

- Finally, we have

THEOREM 5.9 ....

A_{LBA} is decidable.

PROOF The algorithm that decides A_{LBA} is as follows.

$L =$ "On input $\langle M, w \rangle$, where M is an LBA and w is a string:

1. Simulate M on w for qng^n steps or until it halts.
2. If M has halted, *accept* if it has accepted and *reject* if it has rejected. If it has not halted, *reject*."

If M on w has not halted within qng^n steps, it must be repeating a configuration according to Lemma 5.8 and therefore looping. That is why our algorithm rejects in this instance.

Undecidable Problems about LBA

- Let

$$E_{\mathsf{LBA}} = \{\langle M \rangle \mid M \text{ is an LBA where } L(M) = \emptyset\}.$$

- Then, we have

THEOREM 5.10

E_{LBA} is undecidable.

Proof:

- For a TM M and an input w we can determine whether M accepts w by constructing a certain LBA B and then testing whether L(B) is empty.
- The language that B recognizes comprising all accepting computation histories for M on w.

If M accepts w, this language contains one string and so is nonempty. If M does not accept w, this language is empty.

Proof (cont.):

The LBA B works as follows. When it receives an input x, B is supposed to accept if x is an accepting computation for M on w. First, B breaks up x according to the delimiters into strings $C_1, C_2, \ldots, C_l$. Then B determines whether the C_i satisfy the three conditions of an accepting computation history.

1. C_1 is the start configuration for M on w.

2. Each C_{i+1} legally follows from C_i.

3. C_l is an accepting configuration for M.

$$\# \underbrace{\qquad\qquad}_{C_1} \# \underbrace{\qquad\qquad}_{C_2} \# \underbrace{\qquad\qquad}_{C_3} \# \quad \cdots \quad \# \underbrace{\qquad\qquad}_{C_l} \#$$

Proof (cont.)

- Now we are ready to state the reduction of A_{TM} to E_{LBA}. Construct TM S that decides A_{TM} as follows.

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

1. Construct LBA B from M and w as described in the proof idea.
2. Run R on input $\langle B\rangle$.
3. If R rejects, *accept*; if R accepts, *reject*."

Undecidable Problems about CFG

- Let

$$ALL_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \Sigma^*\}.$$

- Then, we have

THEOREM 5.13

ALL_{CFG} is undecidable.

Proof:

- We now describe how to use a decision procedure for ALL_{CFG} to decide A_{TM}.
- For a TM M and an input w we construct a CFG G (or PDA D) that generates all strings if and only if M does not accept w.
- G (or PDA D) generates all strings that

1. do not start with C_1,
2. do not end with an accepting configuration, or
3. where some C_i does not properly yield C_{i+1} under the rule of M.

$$\#\underbrace{\longrightarrow}_{C_1}\#\underbrace{\longleftarrow}_{C_2^{\mathcal{R}}}\#\underbrace{\longrightarrow}_{C_3}\#\underbrace{\longleftarrow}_{C_4^{\mathcal{R}}}\#\ \cdots\ \#\underbrace{\qquad\quad}_{C_l}\#$$

Post Correspondence Problem (PCP)

We can describe this problem easily as a type of puzzle. We begin with a collection of dominos, each containing two strings, one on each side. An individual domino looks like

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

and a collection of dominos looks like

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

The task is to make 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. This list is called a ***match***. For example, the following list is a match for this puzzle.

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

An instance of the PCP

is a collection P of dominos:

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

and a match is a sequence $i_1, i_2, \ldots, i_l$, where $t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$. The problem is to determine whether P has a match. Let

$$PCP = \{\langle P\rangle \mid P \text{ is an instance of the Post correspondence problem with a match}\}.$$

Main Theorem

THEOREM 5.15

PCP is undecidable.

MPCP

$MPCP = \{\langle P \rangle \mid P$ is an instance of the Post correspondence problem with a match that starts with the first domino$\}$.

Proof:

PROOF We let TM R decide the PCP and construct S deciding A_{TM}. Let

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}}),$$

where Q, Σ, Γ, and δ, are the state set, input alphabet, tape alphabet, and transition function of M, respectively.

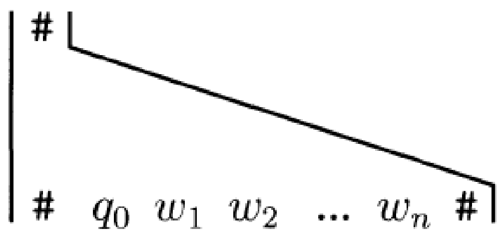
In this case S constructs an instance of the PCP P that has a match iff M accepts w. To do that S first constructs an instance P' of the MPCP. We describe the construction in seven parts, each of which accomplishes a particular aspect of simulating M on w. To explain what we are doing we interleave the construction with an example of the construction in action.

Part 1

Part 1. The construction begins in the following manner.

$$\text{Put } \left[\frac{\#}{\#q_0w_1w_2\cdots w_n\#}\right] \text{ into } P' \text{ as the first domino } \left[\frac{t_1}{b_1}\right].$$

Because P' is an instance of the MPCP, the match must begin with this domino. Thus the bottom string begins correctly with $C_1 = q_0w_1w_2\cdots w_n$, the first configuration in the accepting computation history for M on w, as shown in the following figure.



Part 2 and 3

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

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

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

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

Part 4

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

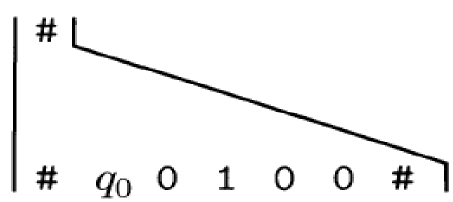
$$\text{put } \left[\frac{a}{a}\right] \text{ into } P'.$$

Now we make up a hypothetical example to illustrate what we have built so far. Let $\Gamma = \{0, 1, 2, \sqcup\}$. Say that w is the string 0100 and that the start state of M is q_0. In state q_0, upon reading a 0, let's say that the transition function dictates that M enters state q_7, writes a 2 on the tape, and moves its head to the right. In other words, $\delta(q_0, 0) = (q_7, 2, \mathrm{R})$.

Part 1 places the domino

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

in P', and the match begins:



Example (cont.)

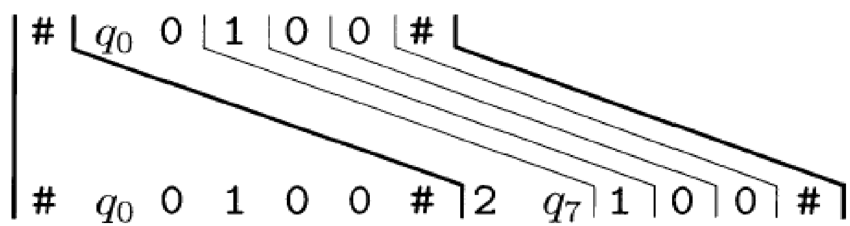
In addition, part 2 places the domino

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

as $\delta(q_0, 0) = (q_7, 2, \mathrm{R})$ and part 4 places the dominos

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

in P', as 0, 1, 2, and $\sqcup$ are the members of Γ. That, together with part 5, allows us to extend the match to



Part 5

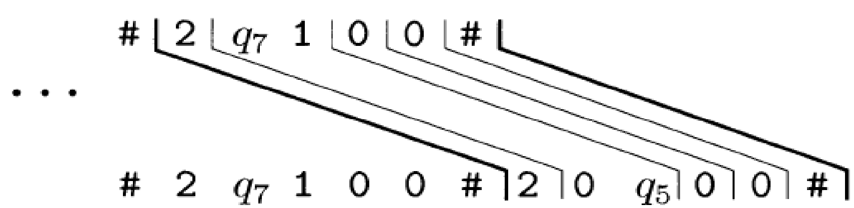
Part 5.

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

Continuing with the example, let's say that in state q_7, upon reading a 1, M goes to state q_5, writes a 0, and moves the head to the right. That is, $\delta(q_7, 1) = (q_5, 0, \mathrm{R})$. Then we have the domino

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

So the latest partial match extends to

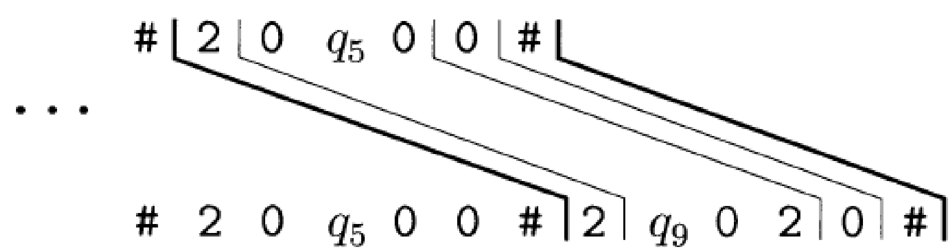


Example (cont.)

Then, suppose that in state q_5, upon reading a 0, M goes to state q_9, writes a 2, and moves its head to the left. So $\delta(q_5, 0) = (q_9, 2, \mathrm{L})$. Then we have the dominos

$$\left[\frac{0q_50}{q_902}\right], \left[\frac{1q_50}{q_912}\right], \left[\frac{2q_50}{q_922}\right], \text{ and } \left[\frac{\sqcup q_50}{q_9\sqcup 2}\right].$$

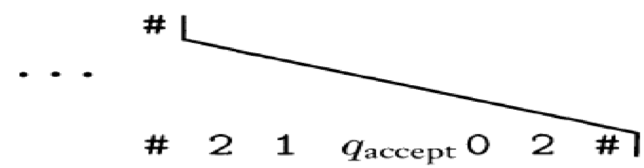
The first one is relevant because the symbol to the left of the head is a 0. The preceding partial match extends to



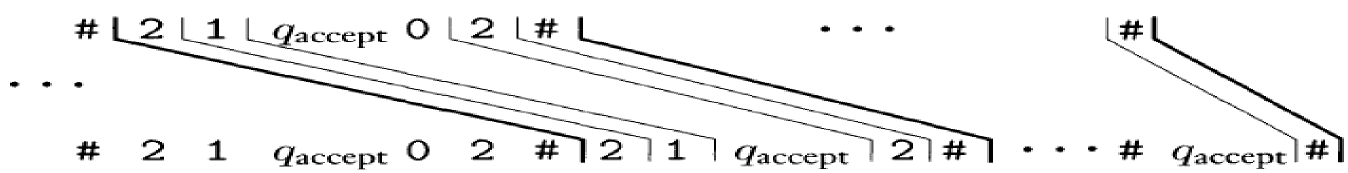
Part 6

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

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



The dominos we have just added allow the match to continue:



Part 7

Part 7. Finally we add the domino

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

and complete the match:

$$\cdots \quad \begin{array}{cccc} \# & q_{\text{accept}} & \# & \# \\ \\ \# & q_{\text{accept}} & \# & \# \end{array}$$

Conversion of P′ to P

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

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

To convert P' to P, an instance of the PCP, we do the following. If P' were the collection

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

we let P be the collection

$$\left\{ \left[\frac{\star t_1}{\star b_1 \star}\right], \left[\frac{\star t_1}{b_1 \star}\right], \left[\frac{\star t_2}{b_2 \star}\right], \left[\frac{\star t_3}{b_3 \star}\right], \ldots, \left[\frac{\star t_k}{b_k \star}\right], \left[\frac{* \diamond}{\diamond}\right] \right\}.$$

Computable Functions

DEFINITION 5.17

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

Mapping Reducibility

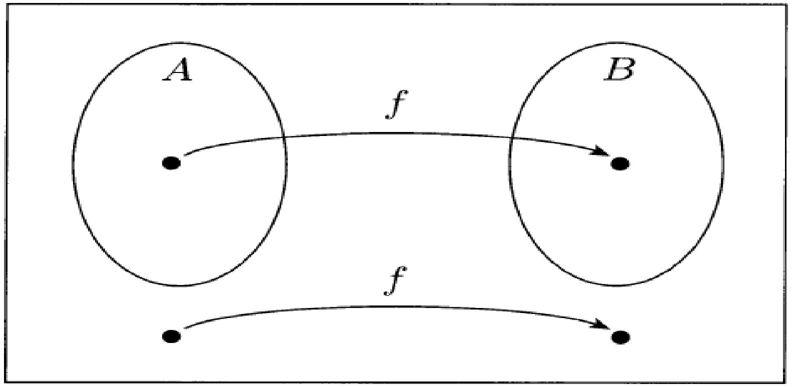
DEFINITION 5.20

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

$$w \in A \Longleftrightarrow f(w) \in B.$$

The function f is called the ***reduction*** of A to B.

The following figure illustrates mapping reducibility.



Theorem

THEOREM 5.22

If $A \leq_m B$ and B is decidable, then A is decidable.

PROOF 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."

Clearly, if $w \in A$, then $f(w) \in B$ because f is a reduction from A to B. Thus M accepts $f(w)$ whenever $w \in A$. Therefore N works as desired.

Corollary

COROLLARY 5.23

If $A \leq_m B$ and A is undecidable, then B is undecidable.

Theorem

THEOREM 5.28

If $A \leq_m B$ and B is Turing-recognizable, then A is Turing-recognizable.

Corollary

COROLLARY 5.29

If $A \leq_m B$ and A is not Turing-recognizable, then B is not Turing-recognizable.

Theorem

THEOREM 5.30

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

PROOF First we show that EQ_{TM} is not Turing-recognizable. We do so by showing that A_{TM} is reducible to $\overline{EQ_{\mathsf{TM}}}$. The reducing function f works as follows.

$F =$ "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 $\langle M_1, M_2\rangle$."

Proof (cont.)

To show that $\overline{EQ_{\mathsf{TM}}}$ is not Turing-recognizable we give a reduction from A_{TM} to the complement of $\overline{EQ_{\mathsf{TM}}}$—namely, EQ_{TM}. Hence we show that $A_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$. The following TM G computes the reducing function g.

$G =$ "The input is $\langle M, w \rangle$ 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 $\langle M_1, M_2 \rangle$."