Theory of Computing
Reducibility

Ming-Hsien Tsai

Department of Information Management
National Taiwan University

Spring 2019

(original created by Bow-Yaw Wang)

Reducibility

- In mathematics, many problems are solved by "reduction."
- Recall the reduction from Eulerian path to Eulerian cycle.
  - Suppose $EC(G)$ returns true iff G has a Eulerian cycle.
  - Let s, t be nodes of a graph G.
  - To check if there is a Eulerian path from s to t in G.
  - Construct a graph G' that is identical to G except an additional edge between s and t.
  - If $EC(G')$ returns true, there is a Eulerian path from s to t.
  - If $EC(G')$ returns false, there is no Eulerian path from s to t.
- Instead of inventing a new algorithm for finding Eulerian paths, we use $EC(G)$ as a subroutine.
- We say the Eulerian path problem is reduced to the Eulerian cycle problem.

Reducibility

- Let us say A and B are two problems and A is reduced to B.
- If we solve B, we solve A as well.
  - If we solve the Eulerian cycle problem, we solve the Eulerian path problem.
- If we can't solve A, we can't solve B.
- To show a problem P is not decidable, it suffices to reduce A_{TM} to P.
- We will give examples in this chapter.

The Halting Problem for Turing Machines

- The halting problem is to test whether a TM M halts on a string w.
- As usual, we first give a language-theoretic formulation.

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

Theorem 1

$HALT_{TM}$ is undecidable.

Proof.

We would like to reduce the acceptance problem to the halting problem. Suppose a TM R decides $HALT_{\text{TM}}$. Consider
$S =$ "On input $\langle M, w\rangle$ where M is a TM and w is a string:

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

□

Emptiness Problem for Turing Machines

- Consider $E_{\text{TM}} = \{\langle M \rangle : M \text{ is a TM and } L(M) = \emptyset\}$.

Theorem 2

E_{TM} is undecidable.

Proof.

We reduce the acceptance problem to the emptiness problem. Let the TM R decides E_{TM}. Consider
$S =$ "On input $\langle M, w \rangle$ where M is a TM and w a string:

1. Use $\langle M \rangle$ to construct
   $M_1 =$ "On input x:
   1. If $x \neq w$, reject.
   2. If $x = w$, run M on the input x. If M accepts x, accept."
2. Run R on the input $\langle M_1 \rangle$.
3. If R accepts, reject; otherwise, accept."

□

Regularity Problem for Turing Machines

- Consider

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

Theorem 3

$REGULAR_{TM}$ is undecidable.

Proof.

Let R be a TM deciding $REGULAR_{\text{TM}}$. Consider
S = "On input $\langle M, w \rangle$ where M is a TM and w a string:

1. Use $\langle M \rangle$ to construct
   M_2 = "On input x:
   1. If x is of the form $\mathtt{0}^n\mathtt{1}^n$, accept.
   2. Otherwise, run M on the input w. If M accepts w, accepts."
2. Run R on the input $\langle M_2 \rangle$.
3. If R accepts, accept; otherwise, reject." □

Rice's Theorem

Theorem 4

Let P be a language consisting of TM descriptions such that

1. *P is not trivial ($P \neq \emptyset$ and there is a TM M with $\langle M \rangle \notin P$);*
2. *If $L(M_1) = L(M_2)$, $\langle M_1 \rangle \in P$ iff $\langle M_2 \rangle \in P$.*

Then P is undecidable.

Proof.

Let R be a TM deciding P. Let $T_\emptyset$ be a TM with $L(T_\emptyset) = \emptyset$. WLOG, assume $\langle T_\emptyset \rangle \notin P$. Moreover, pick a TM T with $\langle T \rangle \in P$. Consider

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

1. Use $\langle M \rangle$ to construct

   M_w = "On input x:

   1. Run M on w. If M halts and rejects, reject.
   2. If M accepts w, run T on x."

2. Run R on $\langle M_w \rangle$.
3. If R accepts, accept; otherwise, reject."

□

Language Equivalence Problem for Turing Machines

- Consider

$$EQ_{\mathrm{TM}} = \{\langle M_1, M_2\rangle : M_1 \text{ and } M_2 \text{ are TM's with } L(M_1) = L(M_2)\}.$$

Theorem 5

EQ_{TM} is undecidable.

Proof.

We reduce the emptiness problem to the language equivalence problem this time. Let the TM R decide EQ_{TM} and TM M_1 with $L(M_1) = \emptyset$. Consider

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

1. Run R on $\langle M, M_1\rangle$.
2. If R accepts, accept; otherwise, reject."

□

Computation History

Definition 6

Let M be a TM 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
- C_i yields C_{i+1} in M for $1 \leq i < l$.

A rejecting computation history for M on w is similar, except C_l is a rejecting configuration.

- Note that a computation history is a finite sequence.
- A deterministic Turing machine has at most one computation history on any given input.
- A nondeterminsitic Turing machine may have several computation histories on an input.

Linear Bounded Automaton

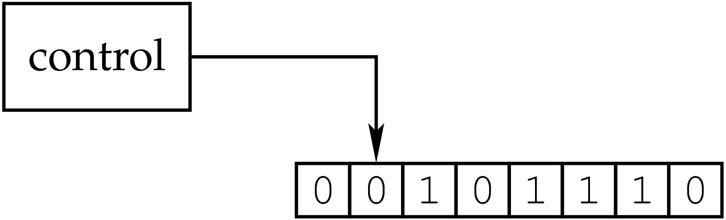


Figure: Schematic of Linear Bounded Automata

Definition 7

A linear bounded automaton is a Turing machine whose tape head is not allowed to move off the portion of its input. If an LBA tries to move its head off the input, the head stays.

- With a larger tape alphabet than its input alphabet, an LBA is able to increase its memory up to a constant factor.

Acceptance Problem for Linear Bounded Automata

- Consider

$$A_{\text{LBA}} = \{\langle M, w\rangle : M \text{ is an LBA and } M \text{ accepts } w\}.$$

Lemma 8

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

- An LBA has only a finite number of different configurations on an input.
- Many langauges can be decided by LBA's.
  - For instance, $A_{\text{DFA}}, A_{\text{CFG}}, E_{\text{DFA}}$, and E_{CFG}.
- Every context-free langauges can be decided by LBA's.

Acceptance Problem for Linear Bounded Automata

Theorem 9

A_{LBA} is decidable.

Proof.

Consider

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

1. Simulate M on w for qng^n steps or until it halts. (q, n, and g are obtained from $\langle M \rangle$ and w.)
2. If M does not halt in qng^n steps, reject.
3. If M accepts w, accept; if M rejects w, reject." □

- The acceptance problem for LBA's is decidable. What about the emptiness problem for LBA's?

$$E_{\text{LBA}} = \{\langle M \rangle : M \text{ is an LBA with } L(M) = \emptyset\}.$$

Emptiness Problem for Linear Bounded Automata

Theorem 10

E_{LBA} is undecidable.

Proof.

We reduce the acceptance problem for TM's to the emptiness problem for LBA. Let R be a TM deciding E_{LBA}. Consider
$S =$ "On input $\langle M, w \rangle$ where M is a TM and w a string:

1. Use $\langle M \rangle$ to construct the following LBA:
   $B =$ "On input $\langle C_1, C_2, \ldots, C_l \rangle$ where C_i's are configurations of M:
   1. If C_1 is not the start configuration of M on w, reject.
   2. If C_l is not an accepting configuration, reject.
   3. For each $1 \leq i < l$, if C_i does not yield C_{i+1}, reject.
   4. Otherwise, accept."
2. Run R on $\langle B \rangle$.
3. If R rejects, accept; otherwise, reject."

□

Universality of Context-Free Grammars

- Consider a problem related to the emptiness problem for CFL's

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

- Let x be a string. Write x^R for the string x in reverse order.
  - For example, $\texttt{100}^R = \texttt{001}$, $\texttt{level}^R = \texttt{level}$.
  - Another example,

乾隆： 客上天然居 居然天上客
紀曉嵐： 人過大鐘寺 寺鐘大過人

- Let $C_1, C_2, \ldots, C_l$ be the accepting configuration of M on input w. Consider the following string in the next theorem:

$$\#\langle C_1\rangle\#\langle C_2\rangle^R\#\cdots\#\langle C_{2k-1}\rangle\#\langle C_{2k}\rangle^R\#\cdots\#\langle C_l\rangle\#$$

Universality of Context-Free Grammars

Theorem 11

ALL_{CFG} is undecidable.

Proof.

We reduce the acceptance problem for TM's to the universalty problem. We construct a nondeterministic PDA D that accepts all strings if and only if M does not accept w. The input and stack alphabets of D contain symbols to encode M's configurations.

$D =$ "On input $\#x_1\#x_2\# \cdots \#x_l\#$:

1. Do one of the following branches nondeterministically:
   - If $x_1 \neq \langle C_1 \rangle$ where C_1 is the start configuration of M on w, accept.
   - If $x_l \neq \langle C_l \rangle$ where C_l is a accepting configuration of M, accept.
   - Choose odd i nondeterministically. If $x_i \neq \langle C \rangle$, $x_{i+1}^R \neq \langle C' \rangle$, or C does not yield C' (C, C' are configurations of M), then accept."
   - Choose even i nondeterministically. If $x_i^R \neq \langle C \rangle$, $x_{i+1} \neq \langle C' \rangle$, or C does not yield C' (C, C' are configurations of M), then accept."

M accepts w iff the accepting computation history of M on w is not in $L(D)$ iff $CFG(D) \notin ALL_{CFG}$. □

Post Correspondence Problem (PCP)

- A domino is a pair of strings: $\left[\frac{t}{b}\right]$
- A match is a sequence of dominos $\left[\frac{t_1}{b_1}\right]\left[\frac{t_2}{b_2}\right]\cdots\left[\frac{t_k}{b_k}\right]$ such that $t_1 t_2 \cdots t_k = b_1 b_2 \cdots b_k$.
- The Post correspondence problem is to test whether there is a match for a given set of dominos.

$$PCP = \{\langle P \rangle : P \text{ is an instance of the PCP with a match}\}$$

- Consider

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

- A match in P:

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

The Modified Post Correspondence Problem

- The modified Post correspondence problem is a PCP where a match starts with the first domino. That is,

$$MPCP = \{\langle P \rangle : \quad P \text{ is an instance of the PCP with a match starting with the first domino}\}$$

Theorem 12

PCP is undecidable.

Proof idea.

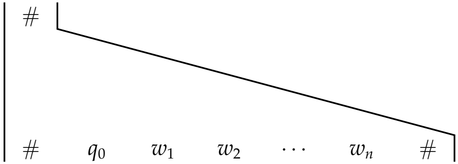
We reduce the acceptance problem for TM's to PCP. Given a TM M and a string w, we first construct an MPCP P' such that $\langle P' \rangle \in MPCP$ if and only if M accepts w. The MPCP P' encodes an accepting computation history of M on w. Finally, we reduce MPCP P' to PCP P.

The Post Correspondence Problem

Proof.

Let the TM R decide *MPCP*. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be the given TM and $w = w_1 w_2 \cdots w_n$ the input. The set P' of dominos has

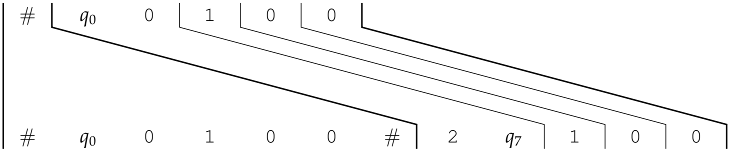
- $\left[\frac{\#}{\#q_0 w_1 w_2 \cdots w_n \#}\right]$ as the first domino. Begin with the start configuration (bottom).



The Post Correspondence Problem

Proof (cont'd).

- $\left[\frac{qa}{br}\right]$ if $\delta(q, a) = (r, b, R)$ with $q \neq q_{\text{reject}}$. Reads a at state q (top); writes b and moves right (bottom).
- $\left[\frac{cqa}{rcb}\right]$ if $\delta(q, a) = (r, b, L)$ with $q \neq q_{\text{reject}}$. Reads a at state q (top); writes b and moves left (bottom).
- $\left[\frac{a}{a}\right]$ if $a \in \Gamma$. Keeps other symbols intact.

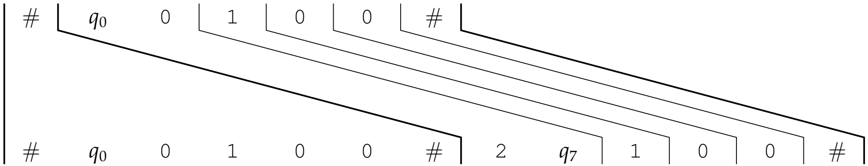


$$\delta(q_0, 0) = (q_7, 2, R)$$

The Post Correspondence Problem

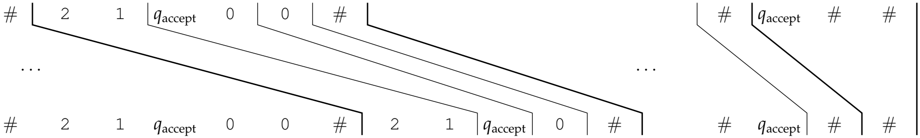
Proof (cont'd).

- $\left[\frac{\#}{\#}\right]$ and $\left[\frac{\#}{\sqcup\#}\right]$ Matches previous # (top) with a new # (bottom). Adds $\sqcup$ when M moves out of the right end.



Proof (cont'd).

- $\left[\frac{aq_{\text{accept}}}{q_{\text{accept}}}\right]$ and $\left[\frac{q_{\text{accept}}a}{q_{\text{accept}}}\right]$ if $a \in \Gamma$. Eats up tape symbols around q_{accept}.
- $\left[\frac{q_{\text{accept}}\#\#}{\#}\right]$. Completes the match.



The Post Correspondence Problem

Proof (cont'd).

So far, we have reduced the acceptance problem of TM's to MPCP. To complete the proof, we need to reduce MPCP to PCP.
Let $u = u_1 u_2 \cdots u_n$. Define

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

Given a MPCP P':

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

Construct a PCP P:

$$\left\{ \left[\frac{\star t_1}{\star b_1 \star} \right], \left[\frac{\star t_2}{b_2 \star} \right], \ldots, \left[\frac{\star t_k}{b_k \star} \right], \left[\frac{* \Diamond}{\Diamond} \right] \right\}$$

Any match in P must start with the domino $\left[\frac{\star t_1}{\star b_1 \star} \right]$. □

Computable Functions

Definition 13

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

- Usual arithmetic operations on integers are computable functions. For instance, the addition operation is a computable function mapping $\langle m, n \rangle$ to $\langle m + n \rangle$ where m, n are integers.

Mapping Reducibility

Definition 14

A language A is mapping reducible (or many-one reducible) to a languate B (written $A \leq_m B$) if there is a computable function $f : \Sigma^* \to \Sigma^*$ such that

$$w \in A \text{ if and only if } f(w) \in B, \text{ for every } w \in \Sigma^*.$$

f is called the reduction of A to B.

Properties of Reducibility

Theorem 15

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

Proof.

Let the TM M decide B and f the reduction of A to B. Consider
$N =$ "On input w:

1. Construct $f(w)$.
2. Run M on $f(w)$.
3. If M accepts, accept; otherwise reject. □

Corollary 16

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

Example 17

Give a mapping reduction of A_{TM} to $HALT_{\mathrm{TM}}$.

Proof.

We need to show a computable function f such that $\langle M, w\rangle \in A_{\mathrm{TM}}$ if and only if $\langle M', w'\rangle \in HALT_{\mathrm{TM}}$ whenever $\langle M', w'\rangle = f(\langle M, w\rangle)$.

Consider

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

1. Use $\langle M\rangle$ and w to construct
   M' = "On input x:
   1. Run M on x.
   2. If M accepts, accept.
   3. If M rejects, loop."
2. Output $\langle M', w\rangle$."

□

Examples

Example 18

Give a mapping reduction from E_{TM} to EQ_{TM}.

Proof.

The proof of Theorem 5 gives such a reduction. The reduction maps the input $\langle M \rangle$ to $\langle M, M_1 \rangle$ where M_1 is a TM with $L(M_1) = \emptyset$. □

Transitivity of Mapping Reductions

Lemma 19

If $A \leq_m B$ and $B \leq_m C$, $A \leq_m C$.

Proof.

Let f and g be the reductions of A to B and B to C respectively. $g \circ f$ is a reduction of A to C. □

Example 20

Give a mapping reduction from A_{TM} to PCP.

Proof.

The proof of Theorem 12 gives such a reduction. We first show $A_{\text{TM}} \leq_m MPCP$. Then we show $MPCP \leq_m PCP$. □

More Properties about Mapping Reductions

Theorem 21

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

Proof.

Similar to the proof of Theorem 15 except that M and N are TM's, not deciders. □

Corollary 22

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

More Properties about Mapping Reductions

- Observe that $A \leq_m B$ if and only if $\overline{A} \leq_m \overline{B}$.
  - The same reduction applies to $\overline{A}$ and $\overline{B}$ as well.
- Recall that $\overline{A_{\text{TM}}}$ is not Turing-recognizable.
- In order to show B is not Turing-recognizable, it suffices to show $A_{\text{TM}} \leq_m \overline{B}$.
  - $A_{\text{TM}} \leq_m \overline{B}$ implies $\overline{A_{\text{TM}}} \leq_m \overline{\overline{B}}$. That is, $\overline{A_{\text{TM}}} \leq_m B$.

Equivalence Problem for TM's (revisited)

Theorem 23

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

Proof.

We first show $A_{TM} \leq_m \overline{EQ_{TM}}$. Consider
$F =$ "On input $\langle M, w \rangle$ where M is a TM and w a string:

1. Construct
   $M_1 =$ "On input x:
   1. Reject."

   $M_2 =$ "On input x:
   1. Run M on w. If M accepts, accept."
2. Output $\langle M_1, M_2 \rangle$."

Equivalence Problem for TM's (revisited)

Proof (cont'd).

Next we show $A_{\text{TM}} \leq_m EQ_{\text{TM}}$. Consider
G = "On input $\langle M, w \rangle$ where M is a TM and w a string:

1. Construct
   M_1 = "On input x:
   1. Accept."

   M_2 = "On input x:
   1. Run M on w.
   2. If M accepts w, accept."
2. Output $\langle M_1, M_2 \rangle$."

□