Reducibility
(Based on [Sipser 2006])

Yih-Kuen Tsay

Department of Information Management
National Taiwan University

Introduction

- A *reduction* is a way of converting one problem into another problem in such a way that a solution to the second problem can be used to solve the first problem.

- If a problem A reduces (is reducible) to another problem B, we can use a solution to B to solve A.

- *Reducibility* says nothing about solving A or B alone, but only about the solvability of A in the presence of a solution to B.

- Reducibility is the primary method for proving that problems are computationally unsolvable.

- Suppose that A is reducible to B. If B is decidable, then A is decidable; equivalently, if A is undecidable, then B is undecidable.

The Halting Problem

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

Theorem (5.1)

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

- The idea is to reduce the acceptance problem A_{TM} (shown to be undecidable) to $HALT_{\text{TM}}$.
- Assume toward a contradiction that a TM R decides $HALT_{\text{TM}}$.
- We could then construct a decider S for A_{TM} 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*; it M has rejected, *reject*."

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

Theorem (5.2)

E_{TM} is undecidable.

🌐 Assuming that a TM R decides E_{TM}, we construct a decider S for A_{TM} as follows.

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

1. Construct the following TM M_1.
   $M_1 = $ "On input x:
   1.1 If $x \neq w$, *reject*.
   1.2 If $x = w$, run M on input w and *accept* if M accepts w."

2. Run R on input $\langle M_1 \rangle$.

3. If R accepts, *reject*; if R rejects, *accept*."

Undecidable Problems (cont.)

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

Theorem (5.3)

$REGULAR_{\mathrm{TM}}$ is undecidable.

- Assuming that a TM R decides $REGULAR_{\mathrm{TM}}$, we construct a decider S for A_{TM} as follows.

Undecidable Problems (cont.)

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

1. Construct the following TM M_2.
   $M_2 =$ "On input x:
   1.1 If x has the form $0^n 1^n$, *accept*.
   1.2 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*."

Rice's Theorem

Theorem

Any "nontrivial" property about the languages recognized by Turing machines is undecidable.

- Note 1: The theorem considers only properties that do not distinguish equivalent Turing machine descriptions.
- Note 2: A property is *nontrivial* if it is satisfied by some, but not all, Turing machine descriptions.

Undecidable Problems (cont.)

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

Theorem (5.4)

EQ_{TM} *is undecidable.*

- Assume that a TM R decides EQ_{TM}.
- We construct a decider S for E_{TM} as follows.
- $S =$ "On input $\langle M \rangle$:
  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*."

Computation Histories

Definition (5.5)

An *accepting computation history* for M on w is a sequence of
configurations $C_1, C_2, \cdots, C_l$, where

1. C_1 is the start configuration,
2. C_l is an accepting configuration, and
3. C_i yields C_{i+1}, $1 \leq i \leq l-1$.

A *rejecting computation history* for M on w is defined similarly,
except that C_l is a rejecting configuration.

- Computation histories are finite sequences.
- Deterministic machines have at most one computation history
  on any given input.

Definition (5.6)

A *linear bounded automaton* (LBA) is a restricted type of Turing machine wherein the tape head is not permitted to move off the portion of the tape containing the input.

🔵 So, an LBA is a TM with a limited amount of memory. It can only solve problems requiring memory that can fit within the tape used for the input.

(Note: Using a tape alphabet larger than the input alphabet allows the available memory to be increased up to a constant factor.)
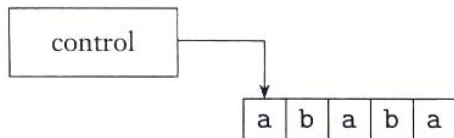
Linear Bounded Automata (cont.)

FIGURE **5.7**
Schematic of a linear bounded automaton

Source: [Sipser 2006]

Linear Bounded Automata (cont.)

Despite their memory constraint, LBAs are quite powerful.

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.

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

Theorem (5.9)

A_{LBA} *is decidable.*

- $L =$ "On input $\langle M, w\rangle$, an encoding of an LBA M and a string w:
    1. Simulate M on input 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 M has not halted, *reject*."

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

Theorem (5.10)

E_{LBA} *is undecidable.*

- Assuming that a TM R decides E_{LBA}, we construct a decider S for A_{TM} as follows.

- $S =$ "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:
    1. Construct an LBA B from $\langle M, w \rangle$ that, on input x, decides whether x is an accepting computation history for M on w.
    2. Run R on input $\langle B \rangle$.
    3. If R rejects, *accept*; if R accepts, *reject*."
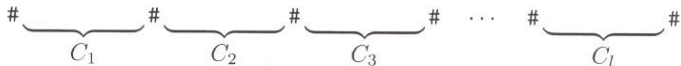
Undecidable Problems about LBAs (cont.)

FIGURE **5.11**
A possible input to B

Source: [Sipser 2006]

Three conditions of an accepting computation history:

- 🔵 C_1 is the start configuration.
- 🔵 C_l is an accepting configuration.
- 🔵 C_i yields C_{i+1}, for every i, $1 \leq i < l$.

Undecidable Problems about LBAs (cont.)
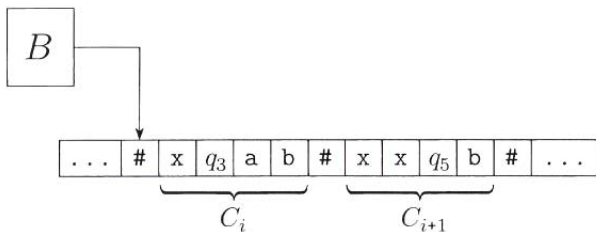
FIGURE **5.12**
LBA B checking a TM computation history

Source: [Sipser 2006]

Undecidable Problems about CFGs

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

Theorem (5.13)

ALL_{CFG} *is undecidable.*

● For a TM M and an input w, we construct a CFG G (by first constructing a PDA) to generate all strings that are *not* accepting computation histories for M on w.

● That is, G generates all strings if and only if M does not accept w.

● If ALL_{CFG} were decidable, then A_{TM} would be decidable.

Undecidable Problems about CFGs (cont.)

The PDA for recognizing computation histories that are not accepting works as follows.

🌐 The input is regarded as a computation history of the form:

$$\#C_1 \# C_2^R \# C_3 \# C_4^R \# \cdots \# C_l \#$$

where C_i^R denotes the reverse of C_i.

🌐 The PDA nondeterministically chooses to check if one of the following conditions holds for the input:

☀ C_1 is not the start configuration.

☀ C_l is not an accepting configuration.

☀ C_i does not yield C_{i+1}, for some i, $1 \leq i < l$.

🌐 It also accepts an input that is not in the proper form of a computation history.

Undecidable Problems about CFGs (cont.)

FIGURE **5.14**
Every other configuration written in reverse order

Source: [Sipser 2006]

The Post Correspondence Problem

- Consider a collection of dominos such as follows:

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

- A *match* is a list of these dominos (repetitions permitted) where the string of symbols on the top is the same as that on the bottom. Below is a match:

$$\left[\frac{a}{ab} \right] \left[\frac{b}{ca} \right] \left[\frac{ca}{a} \right] \left[\frac{a}{ab} \right] \left[\frac{abc}{c} \right]$$
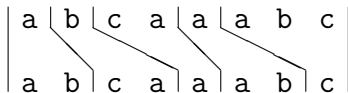
- The Post correspondence problem (PCP) is to determine whether a collection of dominos has a match.

- More formally, an instance of the PCP is a collection of dominos:

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

- A *match* is a sequence $i_1, i_2, \cdots, i_l$ such that $t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$.

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

Theorem (5.15)

PCP is undecidable

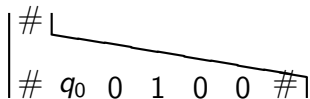- 🌐 The proof is by reduction from A_{TM} via accepting computation histories.

- 🌐 From any TM M and input w we can construct an instance P where a match is an accepting computation history for M on w.

- 🌐 Assume that a TM R decides *PCP*.

- 🌐 A decider S for A_{TM} constructs an instance of the PCP that has a match if and only if M accepts w, as follows.

Undecidability of the PCP (cont.)

1. Add $\left[\dfrac{\#}{\# q_0 w_1 w_2 \cdots w_n \#}\right]$ as $\left[\dfrac{t_1}{b_1}\right]$.

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

$$\text{if } \delta(q, a) = (r, b, R), \text{ add } \left[\dfrac{qa}{br}\right].$$

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

$$\text{if } \delta(q, a) = (r, b, L), \text{ add } \left[\dfrac{cqa}{rcb}\right].$$

4. For every $a \in \Gamma$, add $\left[\dfrac{a}{a}\right]$.

5. Add $\left[\dfrac{\#}{\#}\right]$ and $\left[\dfrac{\#}{\sqcup\#}\right]$.

A start configuration (by Part 1):

$$\left| \begin{array}{c} \# \_ \\ \# \; q_0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; \# \end{array} \right.$$

Suppose $\delta(q_0, 0) = (q_7, 2, R)$. With Parts 2-5, the match may be extended to:

$$\left| \begin{array}{c} \# \_ q_0 \_ 0 \_ 1 \_ 0 \_ 0 \_ \# \_ \\ \# \; q_0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; \# \_ 2 \; q_7 \_ 1 \_ 0 \_ 0 \_ \# \end{array} \right.$$
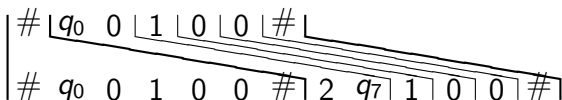
Undecidability of the PCP (cont.)

6. For every $a \in \Gamma$, add $\left[\dfrac{a q_{\text{accept}}}{q_{\text{accept}}} \right]$ and $\left[\dfrac{q_{\text{accept}} a}{q_{\text{accept}}} \right]$.

$$\# \lfloor 2 \lfloor 1 \lfloor q_\text{a} \ 0 \lfloor 2 \lfloor \# \lfloor \quad \cdots \quad \lfloor \# \lfloor$$
$$\# \ 2 \ 1 \ q_\text{a} \ 0 \ 2 \ \# \rceil 2 \rceil 1 \rceil q_\text{a} \rceil 2 \rceil \# \rceil \cdots \# \ q_\text{a} \rceil \# \rceil$$

7. Add $\left[\dfrac{q_{\text{accept}} \# \#}{\#} \right]$.

$$\# \lfloor q_\text{a} \ \# \ \# \rceil$$
$$\# \ q_\text{a} \ \# \rceil \# \rceil$$
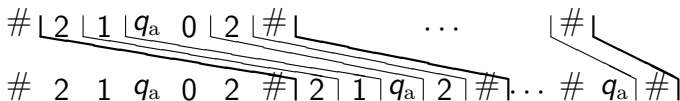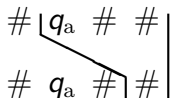
Undecidability of the PCP (cont.)

To ensure that a match starts with $\left[\dfrac{t_1}{b_1}\right]$,

S converts the collection $\left\{\left[\dfrac{t_1}{b_1}\right],\left[\dfrac{t_2}{b_2}\right],\cdots,\left[\dfrac{t_k}{b_k}\right]\right\}$ to

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

where

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

Computable Functions

🔵 A Turing machine computes a function by starting with the
input to the function on the tape and halting with the output of
the function on the tape.

Definition (5.17)

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

🔵 For example, all usual arithmetic operations on integers are
computable functions.

🔵 Computable functions may be transformations of machine
descriptions.

Mapping (Many-One) Reducibility

Definition (5.20)

Language A is **mapping reducible** (many-one reducible) to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \longrightarrow \Sigma^*$, where for every w, $w \in A \Longleftrightarrow f(w) \in B$.
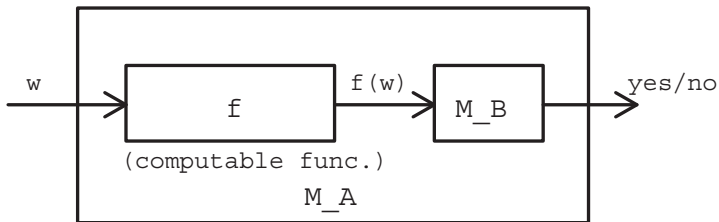
🌐 This provides a way to convert questions about membership testing in A to membership testing in B.
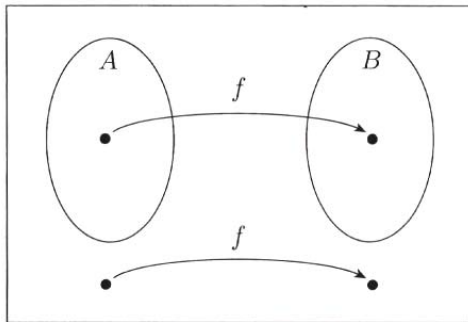
Mapping (Many-One) Reducibility (cont.)

FIGURE **5.21**
Function f reducing A to B

Source: [Sipser 2006]

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

Reducibility and Decidability

Theorem (5.22)

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

🌀 Let M be a decider for B and f a reduction from A to B. A decider N for A works as follows.

🌀 $N =$ "On input w:
   1. Compute $f(w)$.
   2. Run M on input $f(w)$ and output whatever M outputs."

Corollary (5.23)

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

Reducibility and Decidability (cont.)

Theorem

$HALT_{\mathrm{TM}}$ *is undecidable.*

🌐 We show that $A_{\mathrm{TM}} \leq_m HALT_{\mathrm{TM}}$, i.e., a computable function f exists (as defined by F below) such that

$$\langle M, w \rangle \in A_{\mathrm{TM}} \iff f(\langle M, w \rangle) \in HALT_{\mathrm{TM}}.$$

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

Reducibility and Recognizability

Theorem (5.28)

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

Corollary (5.29)

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

Corollary

If $A \leq_m B$ (i.e., $\overline{A} \leq_m \overline{B}$) and A is not co-Turing-recognizable, then B is not co-Turing-recognizable.

Note: "A is not co-Turing-recognizable" is the same as "$\overline{A}$ is not Turing-recognizable".

Reducibility and Recognizability (cont.)

Theorem (5.30 Part One)

EQ_{TM} is not Turing-recognizable.

- We show that A_{TM} reduces to $\overline{EQ_{\mathrm{TM}}}$, i.e., $\overline{A_{\mathrm{TM}}}$ reduces to EQ_{TM}.

- Since $\overline{A_{\mathrm{TM}}}$ is not Turing-recognizable, EQ_{TM} is not Turing-recognizable.

- $F =$ "On input $\langle M, w \rangle$:
    1. Construct the following two machines M_1 and M_2.
       $M1 =$ "On any input: *reject*."
       $M2 =$ "On any input: Run M on w. If it accepts, *accept*."
    2. Output $\langle M_1, M_2 \rangle$."

Theorem (5.30 Part Two)

EQ_{TM} is not co-Turing-recognizable.

- We show that A_{TM} reduces to EQ_{TM}.

- Since A_{TM} is not co-Turing-recognizable, EQ_{TM} is not co-Turing-recognizable.

- $G =$ "On input $\langle M, w \rangle$:
    1. Construct the following two machines M_1 and M_2.
       $M1 =$"On any input: *accept*."
       $M2 =$"On any input: Run M on w. If it accepts, *accept*."
    2. Output $\langle M_1, M_2 \rangle$."