1 Reductions

1.1 Introduction

A reduction is a way of converting one problem into another problem such that a solution to the second problem can be used to solve the first problem. We say the first problem reduces to the second problem.

- Informal Examples: Measuring the area of rectangle reduces to measuring the length of the sides; Solving a system of linear equations reduces to inverting a matrix
- The problem $L_d$ reduces to the problem $A_{TM}$ as follows: “To see if $\langle M \rangle \in L_d$ check if $\langle M, \langle M \rangle \rangle \in A_{TM}$.”

Undecidability using Reductions

**Proposition 1.** Suppose $L_1$ reduces to $L_2$ and $L_1$ is undecidable. Then $L_2$ is undecidable.

**Proof Sketch.**
Suppose for contradiction $L_2$ is decidable. Then there is a $M$ that always halts and decides $L_2$. Then the following algorithm decides $L_1$

- On input $w$, apply reduction to transform $w$ into an input $w'$ for problem 2
- Run $M$ on $w'$, and use its answer.

This can be seen Pictorially as follows.

![Figure 1: Reductions schematically](image)

**The Halting Problem**

**Proposition 2.** The language $HALT = \{ \langle M, w \rangle \mid M \text{ halts on input } w \}$ is undecidable.
Proof. We will reduce $A_{TM}$ to HALT. Based on a machine $M$, let us consider a new machine $f(M)$ as follows:

On input $x$

- Run $M$ on $x$
- If $M$ accepts then halt and accept
- If $M$ rejects then go into an infinite loop

Observe that $f(M)$ halts on input $w$ if and only if $M$ accepts $w$

Suppose HALT is decidable. Then there is a Turing machine $H$ that always halts and $L(H) = \text{HALT}$. Consider the following program $T$

On input $\langle M, w \rangle$

- Construct program $f(M)$
- Run $H$ on $\langle f(M), w \rangle$
- Accept if $H$ accepts and reject if $H$ rejects

$T$ decides $A_{TM}$. But, $A_{TM}$ is undecidable, which gives us the contradiction.

1.2 Definitions and Observations

Mapping Reductions

Definition 3. A function $f : \Sigma^* \to \Sigma^*$ is computable if there is some Turing Machine $M$ that on every input $w$ halts with $f(w)$ on the tape.

Definition 4. A reduction (a.k.a. mapping reduction/many-one reduction) from a language $A$ to a language $B$ is a computable function $f : \Sigma^* \to \Sigma^*$ such that

$$w \in A \text{ if and only if } f(w) \in B$$

In this case, we say $A$ is reducible to $B$, and we denote it by $A \leq_m B$.

Convention

In this course, we will drop the adjective “mapping” or “many-one”, and simply talk about reductions and reducibility.

Reductions and Recursive Enumerability

Proposition 5. If $A \leq_m B$ and $B$ is r.e., then $A$ is r.e.

Proof. Let $f$ be a reduction from $A$ to $B$ and let $M_B$ be a Turing Machine recognizing $B$. Then the Turing machine recognizing $A$ is
On input $w$
Compute $f(w)$
Run $M_B$ on $f(w)$
Accept if $M_B$ accepts, and reject if $M_B$ rejects

\[ \text{Corollary 6. If } A \leq_m B \text{ and } A \text{ is not r.e., then } B \text{ is not r.e.} \]

---

**Reductions and Decidability**

**Proposition 7.** If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.

*Proof.* Let $f$ be a reduction from $A$ to $B$ and let $M_B$ be a Turing Machine deciding $B$. Then a Turing machine that decides $A$ is

On input $w$
Compute $f(w)$
Run $M_B$ on $f(w)$
Accept if $M_B$ accepts, and reject if $M_B$ rejects

\[ \text{Corollary 8. If } A \leq_m B \text{ and } A \text{ is undecidable, then } B \text{ is undecidable.} \]

---

**1.3 Examples**

**The Halting Problem**

**Proposition 9.** The language $\text{HALT} = \{ \langle M, w \rangle \mid M \text{ halts on input } w \}$ is undecidable.

*Proof.* Recall $A_{\text{TM}} = \{ \langle M, w \rangle \mid w \in L(M) \}$ is undecidable. Will give reduction $f$ to show $A_{\text{TM}} \leq_m \text{HALT} \implies \text{HALT undecidable}.$

Let $f(\langle M, w \rangle) = \langle N, w \rangle$ where $N$ is a TM that behaves as follows:

On input $x$
Run $M$ on $x$
If $M$ accepts then halt and accept
If $M$ rejects then go into an infinite loop

$N$ halts on input $w$ if and only if $M$ accepts $w$. i.e., $\langle M, w \rangle \in A_{\text{TM}}$ iff $f(\langle M, w \rangle) \in \text{HALT}$

---

**Emptiness of Turing Machines**

**Proposition 10.** The language $E_{\text{TM}} = \{ \langle M \rangle \mid L(M) = \emptyset \}$ is not r.e.

*Proof.* Recall $L_d = \{ \langle M \rangle \mid M \not\in L(M) \}$ is not r.e.
$L_d$ is reducible to $E_{\text{TM}}$ as follows. Let $f(M) = \langle N \rangle$ where $N$ is a TM that behaves as follows:
On input $x$
- Run $M$ on $\langle M \rangle$
- Accept $x$ only if $M$ accepts $\langle M \rangle$

Observe that $L(N) = \emptyset$ if and only if $M$ does not accept $\langle M \rangle$ if and only if $\langle M \rangle \in L_d$. □

### Checking Regularity

**Proposition 11.** The language $\text{REGULAR} = \{ \langle M \rangle | L(M) \text{ is regular} \}$ is undecidable.

**Proof.** We give a reduction $f$ from $A_{\text{TM}}$ to $\text{REGULAR}$. Let $f(\langle M, w \rangle) = \langle N \rangle$, where $N$ is a TM that works as follows:

- **On input** $x$
  - If $x$ is of the form $0^n1^n$ then accept $x$
  - Else run $M$ on $w$ and accept $x$ only if $M$ does

  If $w \in L(M)$ then $L(N) = \Sigma^*$. If $w \notin L(M)$ then $L(N) = \{0^n1^n | n \geq 0 \}$. Thus, $\langle N \rangle \in \text{REGULAR}$ if and only if $\langle M, w \rangle \in A_{\text{TM}}$. □

### Checking Equality

**Proposition 12.** $\text{EQ}_{\text{TM}} = \{ \langle M_1, M_2 \rangle | L(M_1) = L(M_2) \}$ is not r.e.

**Proof.** We will give a reduction $f$ from $\text{E}_{\text{TM}}$ to $\text{EQ}_{\text{TM}}$. Let $M_1$ be the Turing machine that on any input, halts and rejects i.e., $L(M_1) = \emptyset$. Take $f(M) = \langle M, M_1 \rangle$.

Observe $\langle M \rangle \in \text{E}_{\text{TM}}$ iff $L(M) = \emptyset$ iff $L(M) = L(M_1)$ iff $\langle M, M_1 \rangle \in \text{EQ}_{\text{TM}}$. □