Chapter 5: Reducibility

Decidability

A Language L is **Turing-decidable** if there is a TM M that decides it: M accepts every string in L and rejects every string in $\overline{L}$.

A ***recursive*** language is a decidable language.

Recognizability

A Language L is **Turing-recognizable** if there is a TM M that recognizes it: M accepts every string in L and either rejects or fails to halt on every string in $\overline{L}$.

A ***recursively enumerable*** language is a recognizable language.

Co-Recognizability

A Language L is **co-recognizable** if there is a TM M that recognizes $\overline{L}$: M accepts every string in $\overline{L}$ and either rejects or fails to halt on every string in L.

A **co-r.e.** language is a co-recognizable language.

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.

Proving L undecidable by reducing an undecidable problem to L.

Example 1: $HALT_{TM}$ is undecidable

Let

$HALT_{TM}$ = {<M, w>| M is a TM and M halts on input w}

Then $HALT_{TM}$ is undecidable.

Proof: (by reduction)

We will show that

$$A_{TM} \leq HALT_{TM}$$

$HALT_{TM}$ is undecidable

- Let's assume for the purpose of obtaining a contradiction that TM R decides $HALT_{TM}$.

  We construct TM S to decide A_{TM}:

  S = "On input <M, w>,

  1. Run TM R on <M, w>
  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."

So if R decides $HALT_{TM}$, then S decides A_{TM}. Because A_{TM} is undecidable, so $HALT_{TM}$ must be undecidable.

Example 2: E_{TM} is undecidable

Let

E_{TM} = {<M> | M is a TM and M accepts no inputs}

Then E_{TM} is not decidable.

Proof: (by mapping reduction)

We will show that

$$A_{TM} \leq E_{TM}$$

Example 2: E_{TM} is undecidable (cont'd)

Proof:

Let's assume for the purpose of obtaining a contradiction that TM R decides E_{TM}. Based on R, we can construct TM S to decide A_{TM}:

S = "on input <M, w>:

1. Construct a TM M1:

   M1 = "on input x:

   1. If x ≠ w, reject;
   2. If x = w, run M on w and accept x if M accepts w; otherwise reject x."

2. Run R on <M1>
3. If R accepts, reject; if R rejects, accept."

Example 3: $REGULAR_{TM}$

- $REGULAR_{TM}$={<M>| M is a TM and L(M) is a regular language}.
- Proof: Assume TM R decides $REGULAR_{TM}$. Based on R, we can construct TM S to decide A_{TM}:

S = "on input <M, w>:

1. construct TM M2:

   M2="on input x:

   1. If x has the form 0^n1^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 <M2>
3. If R accepts, *accept*; if R rejects, *reject*."

(if M accepts w, L(M2)= Σ^*, otherwise L(M2)= $\{0^n1^n\}$)

Computable Function

A function *f* is **computable** iff some Turing machine M, on input *w*, halts with *f(w)* on its tape.

Mapping Reduction

Language A is mapping reducible to language B, written $A \leq_m B$, if there is a computable function *f*, where for every *w*,

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

Notation

$$L_1 \text{ reduces to } L_2 \longleftrightarrow L_1 \leq_m L_2$$

Theorem

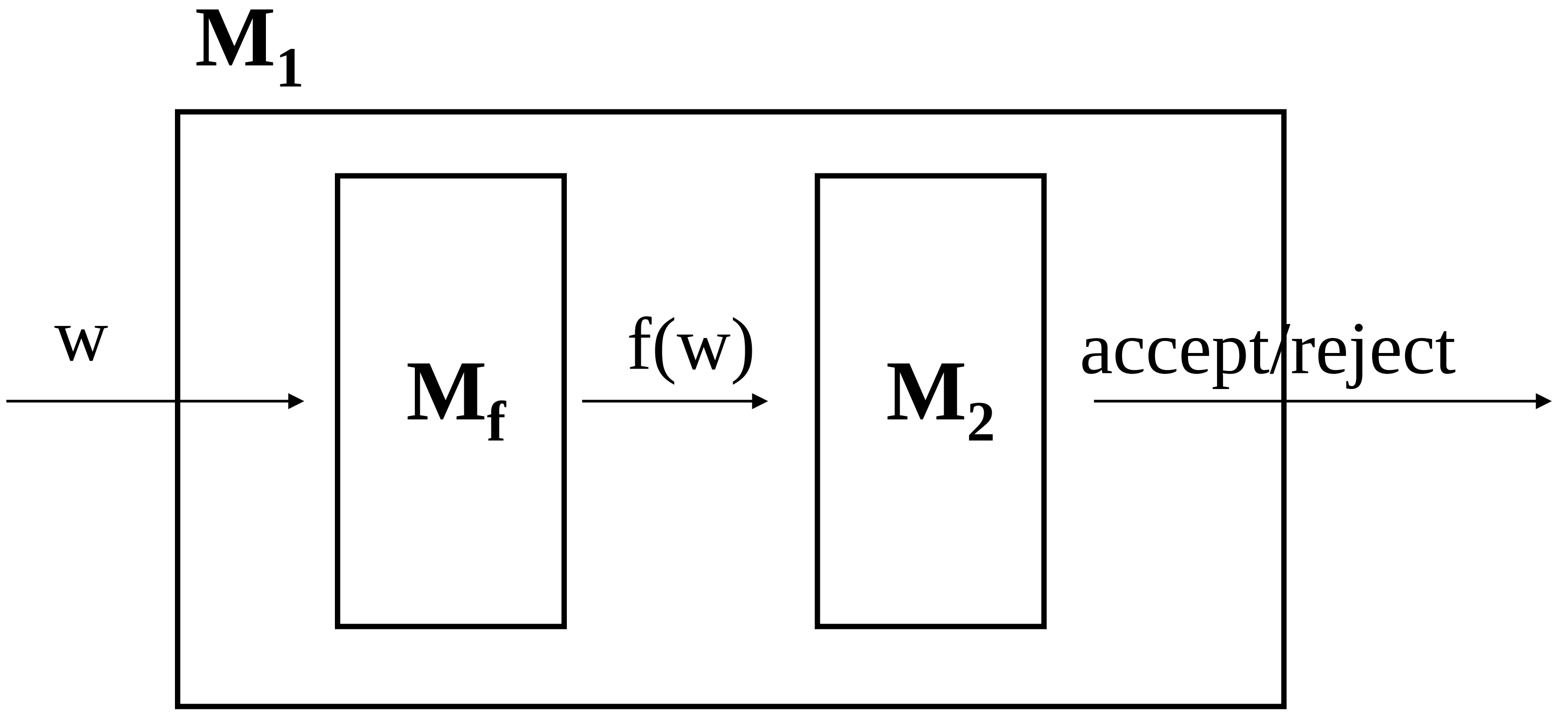
$\forall$ L_1 , L_2 , if $L_1 \leq_m L_2$ and L_2 is decidable, then L_1 is decidable.

Proof

- Let f be the computable function for the reduction from L_1 to L_2
- Let TM M_2 decide L_2

M_1 = "On input w,
1. Compute f(w)
2. Run M_2 on f(w) and output whatever M_2 outputs."

Proof idea: construct a TM for L_1 based on the TM for L_2



Proof

- Let M_f be a Turing Machine that reduces L_1 to L_2
- Let M_2 decide L_2

M_1 = "On input w,
1. Run M_f
2. run M_2 on f(w)
3. Accept if M_2 accepts; reject if M_2 rejects.

Corollary

If $L_1 \leq_m L_2$ and L_1 is undecidable, then L_2 is undecidable.

Theorems

Assume that $L_1 \leq_m L_2$,

- If L is (un)decidable, then $\overline{L}$ is (un)decidable
- If L_1 is undecidable, then L_2 is undecidable
- If L_1 is unrecognizable, then L_2 is unrecognizable
- If L_1 is not co-recognizable, then L_2 is not co- recognizable
- If L_2 is decidable, then L_1 is decidable
- If L_2 is recognizable, then L_1 is recognizable
- If L_2 is co-recognizable, then L_1 is co-recognizable

Example 1: $HALT_{TM}$ is undecidable

$HALT_{TM}$ = {<M, w>| M is a TM and M halts on input w}

We prove by reduction from $A_{TM} \leq_m HALT_{TM}$.

To prove it, we need to find a computable function F that takes input of the form <M, w> and returns output of the form <M', w'>, such that

$$\langle M, w\rangle \in A_{TM} \Leftrightarrow \langle M', w'\rangle \in HALT_{TM}.$$

Example 1 (cont’d)

F =“ On input <M, w>,

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 <M’, w>.”

Example 2: E_{TM} is undecidable

E_{TM} = {<M> | M is a TM and M accepts no inputs}

We prove by reduction from $A_{TM} \leq_m \overline{E_{TM}}$.

To prove it, we need to find a computable function F that takes input of the form <M, w> and returns output of the form <M'>, such that

$$\langle M, w\rangle \in A_{TM} \Leftrightarrow \langle M'\rangle \in \overline{E_{TM}}.$$

Example 2 (cont'd)

F =" On input <M, w>,

1. Construct the following machine M'.

   M' = "On input x:

   1. If $x \neq w$, reject
   2. If $x = w$, run M on x
   3. If M accepts, accept."

2. Output <M'>."

Example 3: EQ_{TM} = {<M1, M2> | M1 and M2 are TMs, and L(M1)=L(M2)} is undecidable

- Need to show that EQ_{TM} is neither Turing recognizable nor co-Turing-recognizable.
- Proof: First show that EQ_{TM} is not Turing-recognizable by showing that $A_{TM} \leq_m \overline{EQ_{TM}}$. The reduction function F works as follows:

F = "On input <M, w>, where M is a TM and w is a string:

1. Construct the following two machines M1 and M2.

    M1= "on any input: reject."

    M2= "On any input: Run M on w. If it accepts, accept."

2. Output <M1, M2>."

Example 3 (cont'd)

- Second show that $\overline{EQ_{TM}}$ is not Turing-recognizable by showing that $A_{TM} \leq_m EQ_{TM}$. The reduction function G works as follows:

G = "On input <M, w> where M is a TM and w is a string:

1. Construct the following two machines M1 and M2.

   M1= "On any input: accept."

   M2= "On any input: Run M on w. If it accepts, accept."

2. Output <M1, M2>."

Reference

- www.cs.unm.edu/~gemmell/500.html by Dr. Peter Gemmell.