Reducibility

So far, we have seen many ways to reduce on problem to another.

Reductions appear in

- decidability theory (familiar)
- complexity theory (to come)

Time to get formal.

Computable Functions

A TM computes a function

$$f : \Sigma^* \longrightarrow \Sigma^*$$

if it

- starts with input w, and
- halts with only $f(w)$ on tape.

Computable Functions

Claim: All the usual arithmetic functions are computable.

Exercise: Design a TM that on input $\langle m, n \rangle$, halts with $\langle m + n \rangle$ on tape.

Computable Functions

One useful class of functions transforms TM descriptions.

On input w:

- if $w = \langle M \rangle$ for some TM,
  - construct $\langle M' \rangle$, where
  - $L(M') = L(m)$, but
  - M' never tires to move off LHS of tape.
- otherwise write ε and halt.

Left as an exercise.

Computable Functions

Definition: Language A is *mapping reducible* to B, written

$$A \leq_m B,$$

if there is a computable function

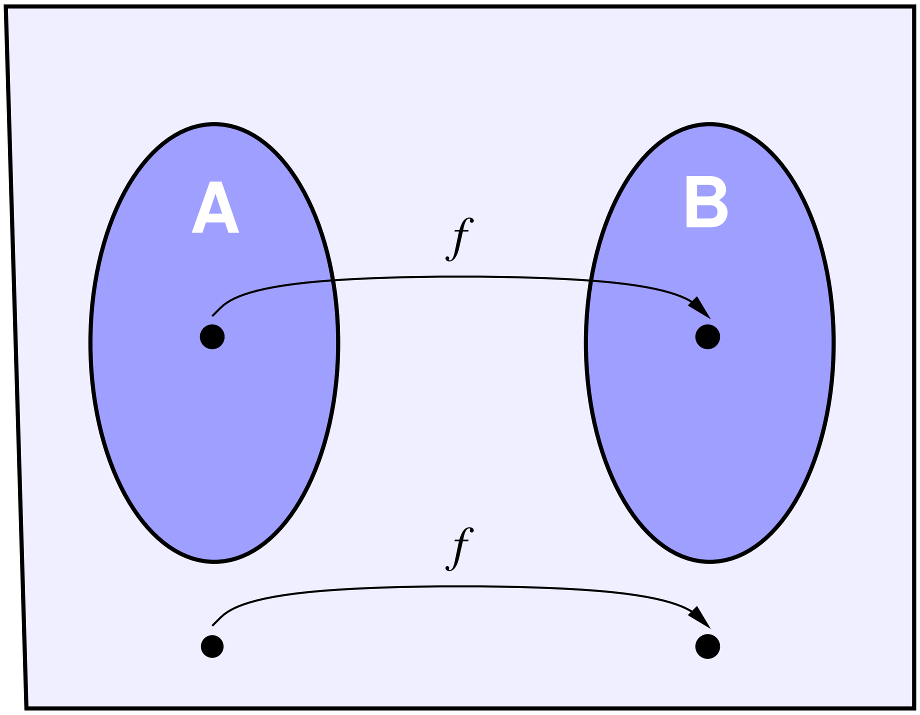
$$f : \Sigma^* \longrightarrow \Sigma^*$$

such that, for every w,

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

Function f is called the *reduction* of A to B.

Computable Functions



A mapping reduction converts questions about membership in A to membership in B

Computable Functions

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

Proof: Let

- M be the decider for B, and
- f the reduction from A to B.

Define

N: on input w

1. compute $f(w)$

2. run M on input $f(w)$ and output whatever M outputs.

Computable Functions

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

This has been our principal tool for proving undecidability.

Example: Halting

Recall that

$$\begin{aligned} A_{\mathsf{TM}} &= \{\langle M, w\rangle | \mathsf{TM}\ M \text{ accepts input } w\} \\ H_{\mathsf{TM}} &= \{\langle M, w\rangle | \mathsf{TM}\ M \text{ halts on input } w\} \end{aligned}$$

Earlier we proved that

- H_{TM} undecidable
- by reduction from A_{TM}.

Let's reformulate this.

Example: Halting

Define computable function f:

- input of form $\langle M, w \rangle$
- output of form $\langle M', w' \rangle$
- where $\langle M, w \rangle \in A_{\mathsf{TM}} \iff \langle M', w' \rangle \in H_{\mathsf{TM}}$.

Example: Halting

The following machine computes this function f.

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

- Construct the following machine M'.

  M': on input x

  - run M on x
  - If M accepts, *accept*.
  - if M rejects, enter a loop.
- output $\langle M', w \rangle$

Example: Post Correspondence Problem

Proof of PCP uses two mapping reductions.

- $A_{\mathsf{TM}} \leq_m$ MPCP
- MPCP $\leq_m$ PCP

Because mapping reduction is transitive (why?),

$$A_{\mathsf{TM}} \leq_m \mathsf{PCP}.$$

Enumerability

Theorem: If $A \leq_m B$ and B is enumerable, then A is enumerable.

Proof is same as before, using accepters instead of deciders.

Enumerability

Corollary: If $A \leq_m B$ and A is not enumerable, then B is not enumerable.

TM Equality

Theorem: EQ_{TM} is neither enumerable no co-enumerable.

First, we prove that EQ_{TM} is not enumerable.

We show that A_{TM} is reducible to

- $\overline{\text{EQ}_{\text{TM}}}$
- EQ_{TM}.

TM Equality

Claim: A_{TM} is reducible to $\overline{\mathsf{EQ}_{\mathsf{TM}}}$.

$f : A_{\mathsf{TM}} \longrightarrow \overline{\mathsf{EQ}_{\mathsf{TM}}}$ works as follows:

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

- Construct machine M_1: on any input, *reject*.
- Construct machine M_2: on any input, run M on w. If it accepts, *accept*.
- Output $\langle M_1, M_2 \rangle$.

Note

- M_1 accepts nothing
- if M accepts w then M_2 accepts everything, and otherwise nothing.
- $\langle M, x \rangle \in A_{\mathsf{TM}} \iff \langle M_1, M_2 \rangle \in \overline{\mathsf{EQ}_{\mathsf{TM}}}$

TM Equality

Claim: A_{TM} is reducible to $\mathsf{EQ}_{\mathsf{TM}}$.

$f : A_{\mathsf{TM}} \longrightarrow \mathsf{EQ}_{\mathsf{TM}}$ works as follows:

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

- Construct machine M_1: on any input, *accept*.
- Construct machine M_2: on any input, run M on w. If it accepts, *accept*.
- Output $\langle M_1, M_2 \rangle$.

Note

- M_1 accepts everything
- if M accepts w then M_2 accepts everything, and otherwise nothing.
- $\langle M, x \rangle \in A_{\mathsf{TM}} \iff \langle M_1, M_2 \rangle \in \mathsf{EQ}_{\mathsf{TM}}$