Lecture 18

Q2: show (i) $\Rightarrow$ (ii) & (ii) $\Rightarrow$ (i)

To show (i) $\Rightarrow$ (ii) [show if $\exists$ an onto fxn $g: \mathbb{N} \to S$ then $\exists$ 1-1 fxn $f: S \to \mathbb{N}$]

3 steps

[(1) Define f

[(2) Show your f is well-defined
($\forall s \in S$, $f(s)$ = one element in $\mathbb{N}$)

[(3) Show f is 1-1

Recap from Last week

① $D = \{\langle M\rangle \mid M(\langle M\rangle) \text{ does not accept}\}$ is not r.e.

Diagonal Language

Proof by diagonalization

② $\bar{D} = \{\langle M\rangle \mid M(\langle M\rangle) \text{ halts and accepts}\}$

is r.e. but not recursive

($\forall L \subseteq \Sigma^*$
If L recursive
then $\bar{L}$ also recursive)

③ $A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$

is r.e. but not recursive

$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$

D not r.e.
$\overline{D}$ is re, no rec
A_{TM} is re, not recursive

- We saw that A_{TM} is r.e./recognizable.

Pf that A_{TM} is not decidable:

Assume for sake of contradiction there is a decider N for A_{TM}.

We will use M to construct a decider N' for $\tilde{D}$:

N':

> on input $\langle M\rangle$:
> check if input is a legal encoding of a TM. If not reject
> Otherwise Run N on $\langle M, \langle M\rangle\rangle$
> If N accepts $\longrightarrow$ accept
> If N rejects $\longrightarrow$ reject

Since N always halts, N' always halts.
Also N' accepts $\tilde{D}$. Contradiction since $\tilde{D}$ is not decidable
$\therefore$ A_{TM} is not decidable

TM Reductions

The previous proof showing that A_{TM} is not decidable is a **reduction**; we showed:

a decider for A_{TM} $\Longrightarrow$ a decider for $\tilde{D}$

Since $\tilde{D}$ not decidable, A_{TM} also not decidable.

So we showed $\tilde{D} \leq_T A_{TM}$

Defn Language A is TM-reducible to language B, written $A \leq_T B$ if
a decider for B $\Longrightarrow$ a decider for A

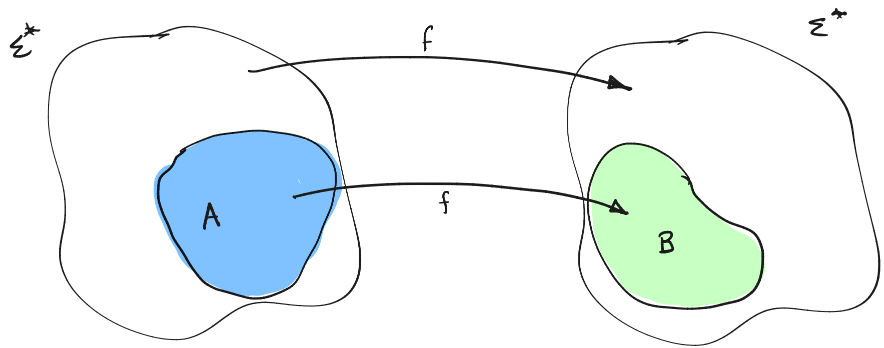
Important If $A \leq_T B$ and B is decidable then A is decidable

If $A \leq_T B$ and A is not decidable, then B is not decidable
(contrapositive)

More on Reducibilities

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

$$\forall x \in \Sigma^* \left(x \in A \iff f(x) \in B \right)$$

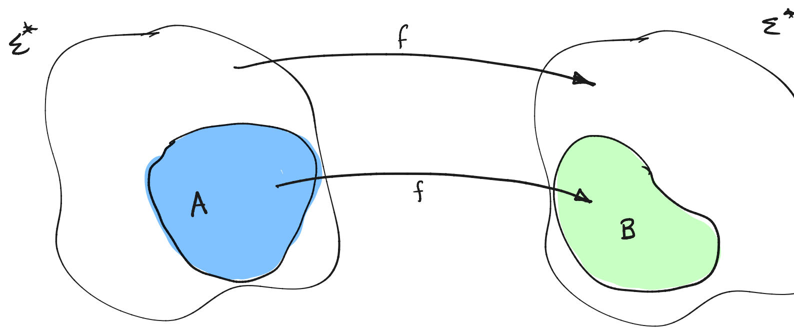


- f maps strings in A to strings in B, and strings not in A to strings not in B

More on Reducibilities

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

$$\forall x \in \Sigma^* \; \left(x \in A \iff f(x) \in B \right)$$



- f maps strings in A to strings in B, and strings not in A to strings not in B

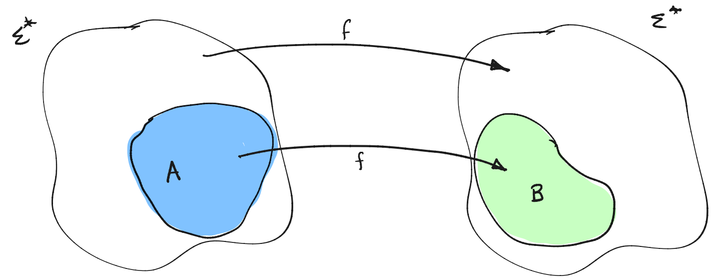
Lemma If $A \leq_M B$ then $A \leq_T B$

(mapping reductions are special case of Turing Reductions)

More on Reducibilities

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

$$\forall x \in \Sigma^* \left(x \in A \iff f(x) \in B \right)$$



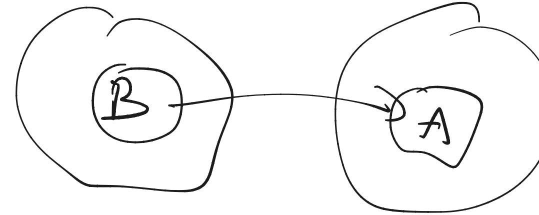
Lemma Let $A \leq_m B$. Then:

① B decidable $\Rightarrow$ A decidable (or equivalently, A undecidable $\to$ B undecidable)

② B recognizable/re $\Rightarrow$ A recognizable/r.e.

Lemma If we have $A \leq_m B$, then we also have $\bar{A} \leq_m \bar{B}$

$B \leq_m A$:



To show A is not decidable

Do I want to show

(1) $A \leq_m B$ for some B that is undecidable

or (2) $B \leq_m A$ for some B that is undecidable ?

this direction

$B \leq_m A$ means

(2): If A is decidable then B is decidable

so B not dec $\rightarrow$ A not dec. ✓

Example : HALT = $\{ \langle M, x\rangle \mid M$ halts on input $x \}$

Lemma 1 HALT is r.e. (exercise)

Lemma 2 Halt is not recursive/decidable

Proof of Lemma 2 We will show $A_{TM} \leq_T$ HALT

Then since A_{TM} not decidable, this implies HALT not decidable.
Let N be an (alleged) decider for HALT. We will use N to create a decider, N' for A_{TM}

N' :

on input $\langle M, x\rangle$: Check if input is legal encoding of a TM M, followed by x (halt if not) Run N on input $\langle M, x\rangle$ If N accepts, simulate M on x. Accept $\langle M, x\rangle$ if simulation accepts; otherwise reject $\langle M, x\rangle$ If N rejects, halt and reject

Example : HALT = $\{\langle M, x\rangle \mid M$ halts on input $x\}$

Lemma 1 HALT is r.e. (exercise)

Lemma 2 Halt is not recursive/decidable

Proof of Lemma 2 We will show $A_{TM} \leq_T$ HALT. Then since A_{TM} not decidable, this implies HALT not decidable.
Let N be an (alleged) decider for HALT. We will use N to create a decider, N' for A_{TM}

Note:
This is a Turing reduction but not a mapping reduction

N':

on input $\langle M, x\rangle$:
check if input is legal encoding of a TM M, followed by x (halt if not)
Run N on input $\langle M, x\rangle$
    If N accepts, simulate M on x. Accept $\langle M, x\rangle$ if simulation accepts; otherwise reject $\langle M, x\rangle$
    If N rejects, halt and reject

Proof of correctness: First, if N is a decider for HALT then N' will halt on all inputs.

Now for correctness: First if M halts on x, then N' just simulates M on x and does the same thing, so N' will also halt + accept $\langle M, x\rangle$
otherwise if M does not halt on x, then N will not accept so N' will also halt and reject.

Example : HALT = $\{ \langle M, x \rangle \mid M$ halts on input $x \}$

Lemma 1 HALT is r.e. (exercise)

Lemma 2 Halt is not recursive

Lemma 3 $\overline{\text{HALT}}$ is not r.e.

If Halt, $\overline{\text{HALT}}$ both re, then HALT would be decidable (By closure property). $\therefore$ By Lemma 2, $\overline{\text{HALT}}$ Not r.e.

Closure Prop:

(If L and $\overline{L}$ are both r.e., then they are both recursive.)

$\overline{\text{HALT}} = \{ \langle M, x \rangle \mid M$ does not halt on input $x \}$

Example: Nonempty = $\{ \langle M \rangle \mid M$ accepts at least one string $\}$
i.e. $\mathcal{L}(M)$ is not empty

(1) Nonempty is r.e. (Pf: use dovetailing)

Example Nonempty = { <M> | M accepts at least one string }

① Nonempty is r.e. (Pf : use dovetailing)

② Nonempty is Not recursive/decidable.

Assume for sake of contradiction N is a decider for Nonempty.

We will use N to construct a decider, N' for HALT

N' :

on input <M, x> :
    Let M' be a TM that on input w, M' ignores its input and simulates M on x.
    If M halts on x then M' halts and accepts
  Run N on <M'>
  If N accepts <M'> → halt and accept
  otherwise → halt and reject

* M' depend on M & x.

Example Nonempty = { <M> | M accepts at least one string }

① Nonempty is r.e. (Pf: use dovetailing)

② Nonempty is Not recursive/decidable.

Assume for sake of contradiction N is a decider for Nonempty.
We will use N to construct a decider, N' for A_{TM}

N' :

on input <M,x>: Let M' be a TM that on input w, M' ignores its input and simulates M on x. If M accepts x then M' halts and accepts Run N on <M'> If N accepts <M'> $\longrightarrow$ halt and accept otherwise $\longrightarrow$ halt and reject

M' accepts all strings if M accepts x
M' accepts NO strings if M does not accept x

$\mathcal{L}(M')$ Nonempty iff M accepts x

Note: This is a mapping reduction showing $A_{TM} \leq_m$ Nonempty:

$$f : \langle M,x \rangle \longrightarrow \langle M'_{M,x} \rangle$$