CS/Math 361

Announcements

Homework 9: due Friday

Delayed office hours today - 2:30?

Volunteer opportunity this afternoon

Twisted Histories

$L_{Computation\text{-}History}$

Definition: Given a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ and a string $w \in \Sigma^*$, we define the language of computation histories for M on w as:

$L_{Computation\text{-}History}(M, w) =$

$\{w_0 w_1 \ldots w_n$ | each w_i is a configuration for M,
w_0 is the initial configuration for w,
w_n is a final/accept configuration, &
each w_i yields w_{i+1} according to δ $\}$

$L^R_{Computation\text{-}History}$

Definition: Given a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ and a string $w \in \Sigma^*$, we define the language of computation histories for M on w as:

$L^R_{Computation\text{-}History}(M, w) =$
$\{w_0 w^R_1 \ldots w_n \mid$ each w_i is a configuration for M,
w_0 is the initial configuration for w,
w_n is a final/accept configuration,
each wi yields w_{i+1} according to δ, &
every other w_i is reversed $\}$

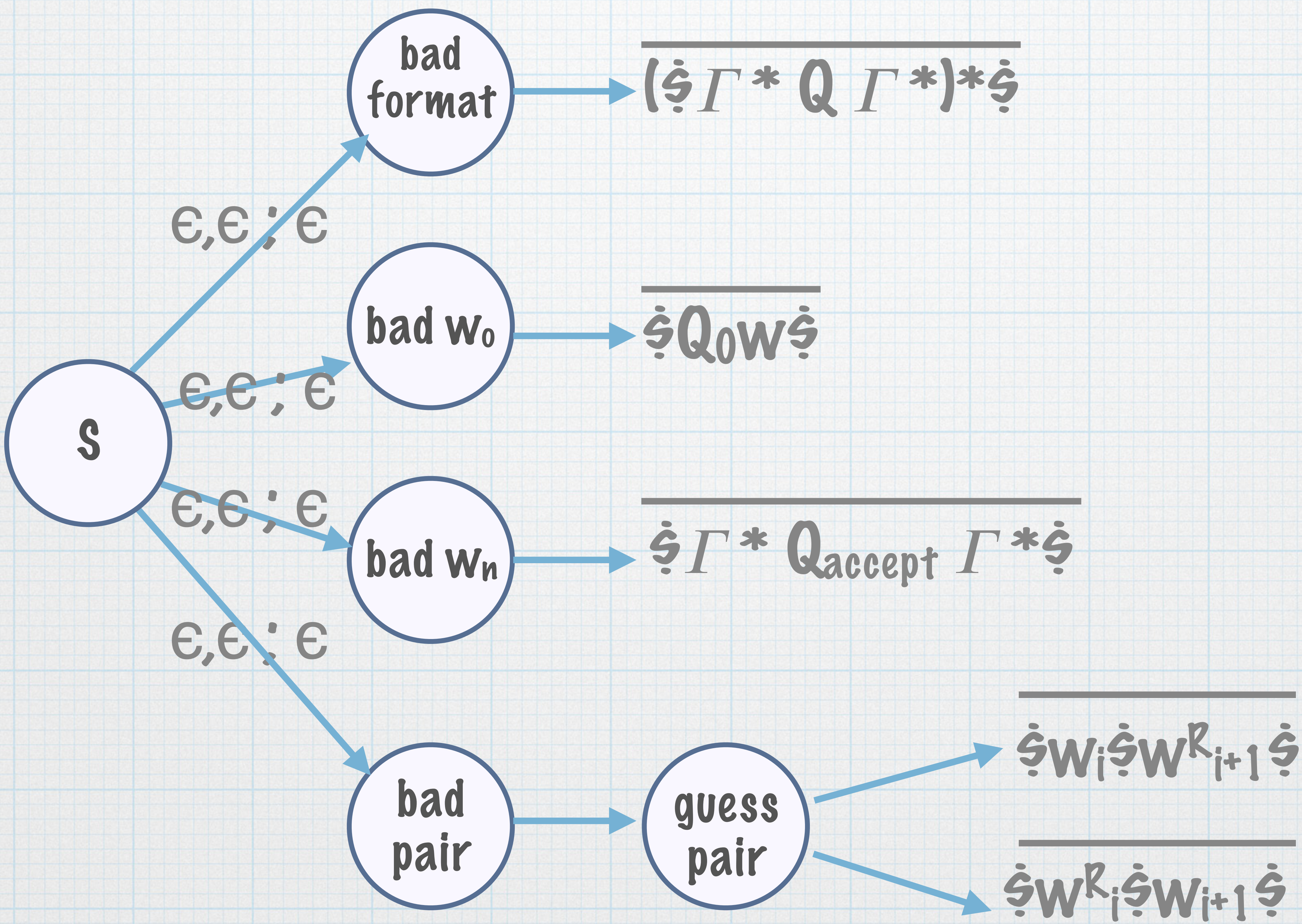
$L^R_{\text{Computation-History}}$

Definition: Given a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ and a string $w \in \Sigma^*$, we define the language of computation histories for M on w as:

$L^R_{\text{Computation-History}}(M, w) =$

$\{w_0 w^R_1 \ldots w_n$ | some w_i is not a configuration for M,
w_0 is not the initial configuration for w,
w_n is not final/accept configuration, or
some wi does not yields w_{i+1}, but
every other w_i is reversed $\}$

L^R Computation-History



Proof that ALL_{CFG} not R.E.

Assume that ALL_{CFG} is recognized by M_{ALL} and construct a machine $M_{\overline{A_{TM}}}$ that behaves as follows:

1. On input ⟨M,w⟩:
   1.1. Construct an encoding $\langle G_{(M,w)} \rangle$ for a CFG for the language $\overline{L^R_{Computation\text{-}History}(M, w)}$
   1.2. Run M_{ALL} on $\langle G_{(M,w)} \rangle$ (and accept if it does).
2. If the input is not of the form ⟨M,w⟩, accept

This machine would recognize $M_{\overline{A_{TM}}}$ which we have shown is impossible, so M_{ALL} must not exist and ALL_{CFG} must not be recognizable.

	DECIDABLE	R.E.	not R.E.	not co-R.E.
E_{TM}			X	
$\overline{E_{TM}}$		X		
EQ_{CFG}			X	
$\overline{EQ_{CFG}}$		X		
EQ_{TM}				X
$\overline{EQ_{TM}}$				X
ALL_{CFG}			X	
$\overline{ALL_{CFG}}$		X		
ALL_{TM}				X
$\overline{ALL_{TM}}$				X

Proof that E_{TM} not R.E.

Assume that E_{TM} is recognized by M_E and construct a machine $M_{\overline{A_{TM}}}$ that behaves as follows:

1. On input ⟨M,w⟩:
   1.1. Construct a description, ⟨M'⟩, of a TM M' that behaves as follows:

      On input w', simulate M on input w and:

      i. accept w' if M accepts w.
      ii. otherwise reject w' or loop.

   1.2. Run M_E on ⟨M'⟩ (and accept if it does).
2. If the input is not of the form ⟨M,w⟩, accept

Computable Functions

Definition: A function $f: \Sigma^* \rightarrow \Sigma^*$ is computable iff there is a Turing machine M such that on every input w, M halts with f(w) on its tape.

Computable Functions

f(<M,w>) = <M'>
where M' runs M on w
and
accepts its input w' only if M accepts w.

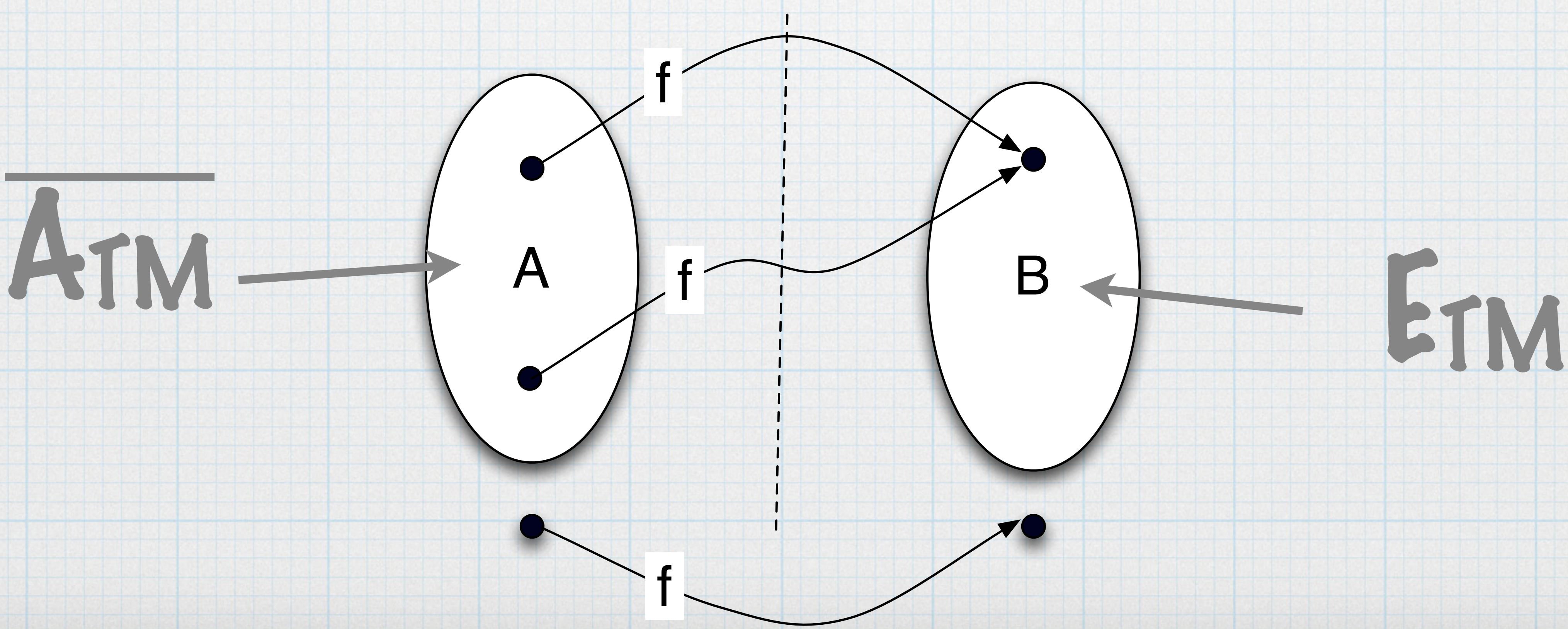
Computable Functions

$f(\langle M,w \rangle) = \langle M' \rangle$

where M′ runs M on w

and

accepts its input w′ only if M accepts w.

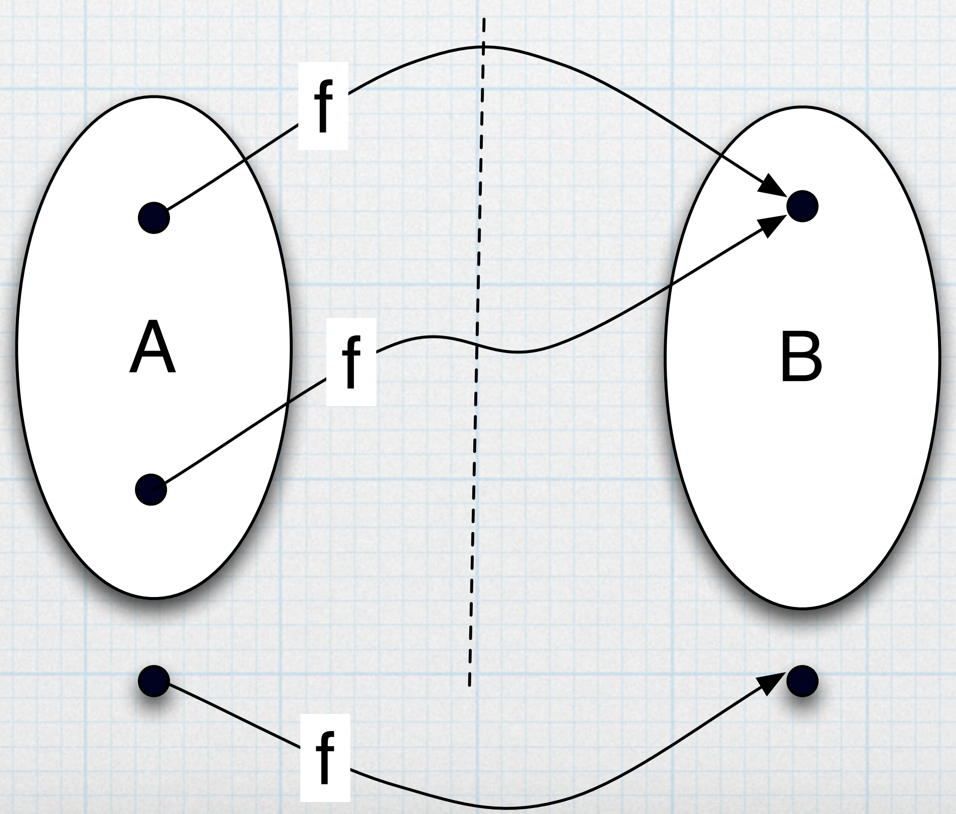


Mapping Reducible

Definition: Language A is many-to-one reducible to language B if there exists a computable function $f: \Sigma^* \to \Sigma^*$ such that

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

for every $w \in \Sigma^*$. We call f a reduction, write $A \leq_m B$ and say that A is easier than (or just as hard as) B.

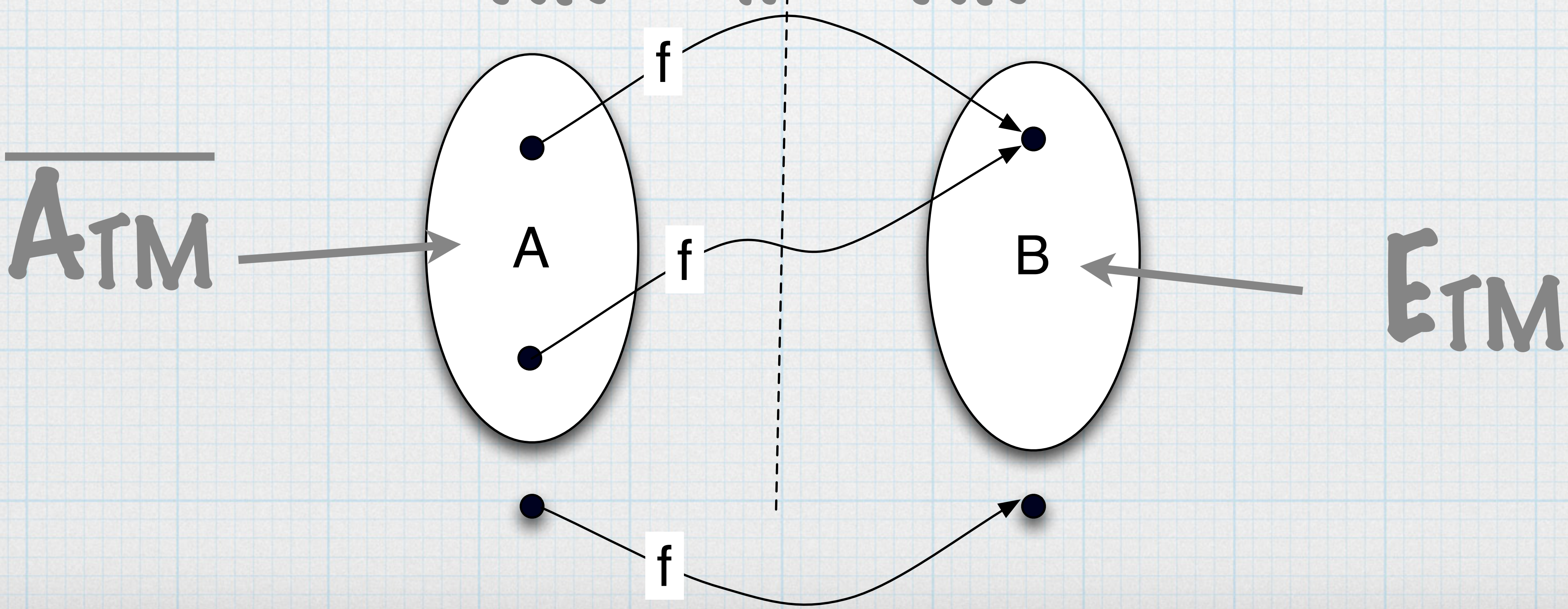


Computable Functions

$A \leq_m B$ iff there exists computable $f: \Sigma^* \rightarrow \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$

$f(\langle M,w \rangle) = \langle M' \rangle$
where M′ runs M on w
and
accepts its input w′ only if M accepts w.

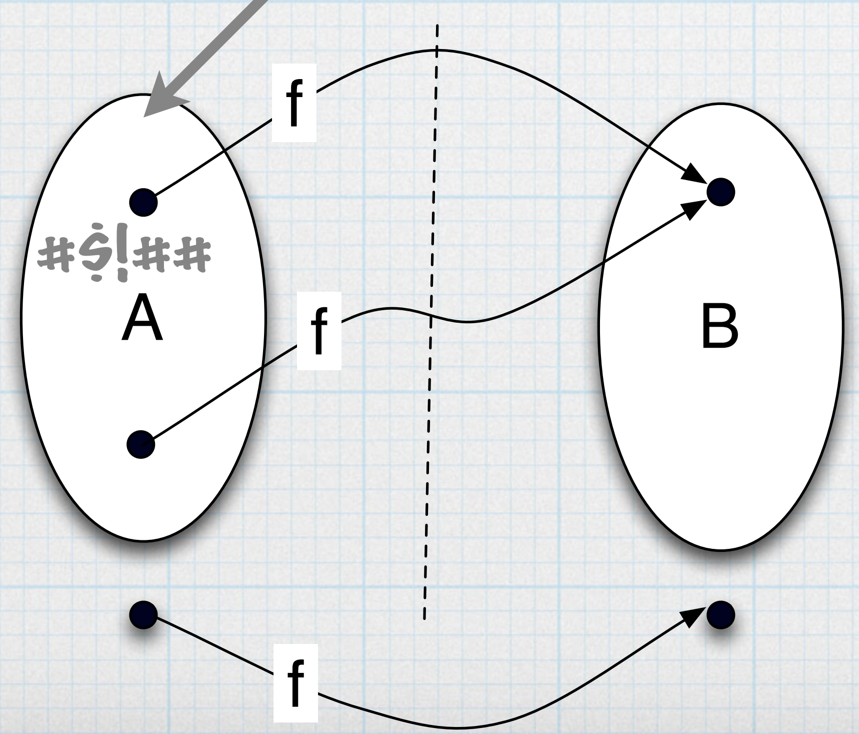
$A_{TM} \leq_m E_{TM}$



Computable Functions

f(⟨M,w⟩) = ⟨M′⟩
where M′ runs M on w
and
accepts its input w′ only if M accepts w.

#$!## ∈ $\overline{A_{TM}}$



A $\leq_m$ B iff there exists computable f: $\Sigma^* \rightarrow \Sigma^*$ such that

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

Computable Functions

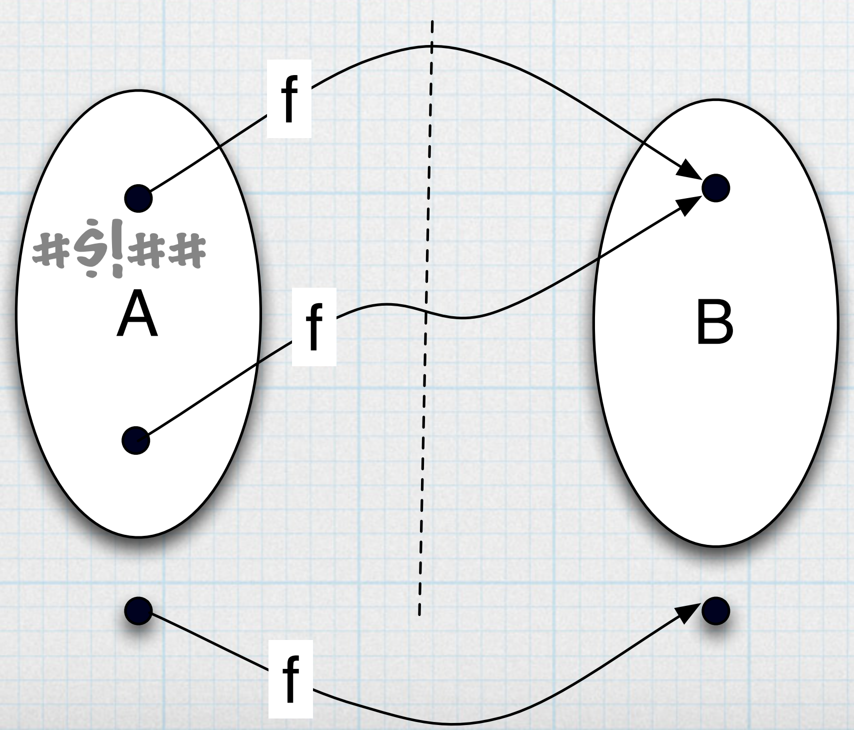
$f(x) = \langle M' \rangle$, if $x = \langle M, w \rangle$

$f(x) = \langle EMPTY \rangle$, otherwise

where

$L(EMPTY) = \phi$

M′ runs M on w and accepts its input w′ only if M accepts w.



$A \leq_m B$ iff there exists computable $f: \Sigma^* \rightarrow \Sigma^*$ such that

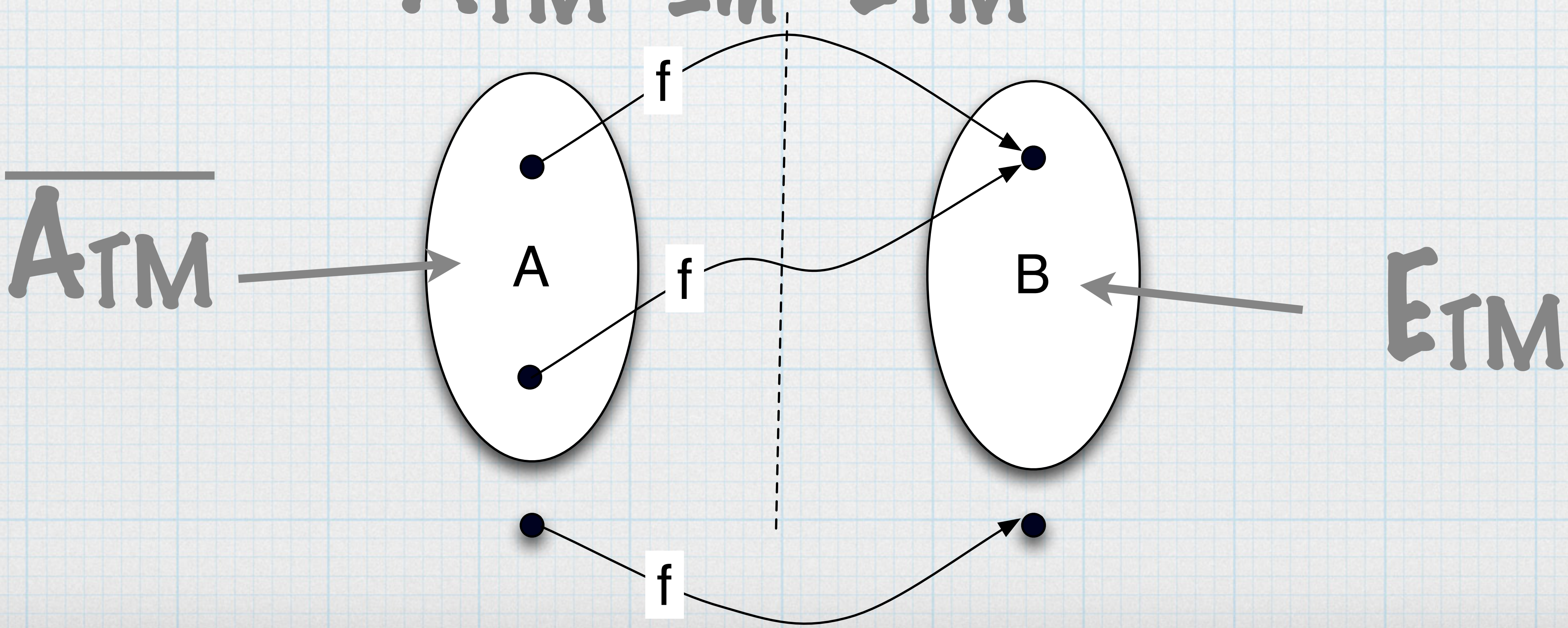
$w \in A \Leftrightarrow f(w) \in B$

Computable Functions

$A \leq_m B$ iff there exists computable $f: \Sigma^* \rightarrow \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$

$f(\langle M,w \rangle) = \langle M' \rangle$
where M' runs M on w
and
accepts its input w' only if M accepts w.

$\overline{A_{TM}} \leq_m E_{TM}$

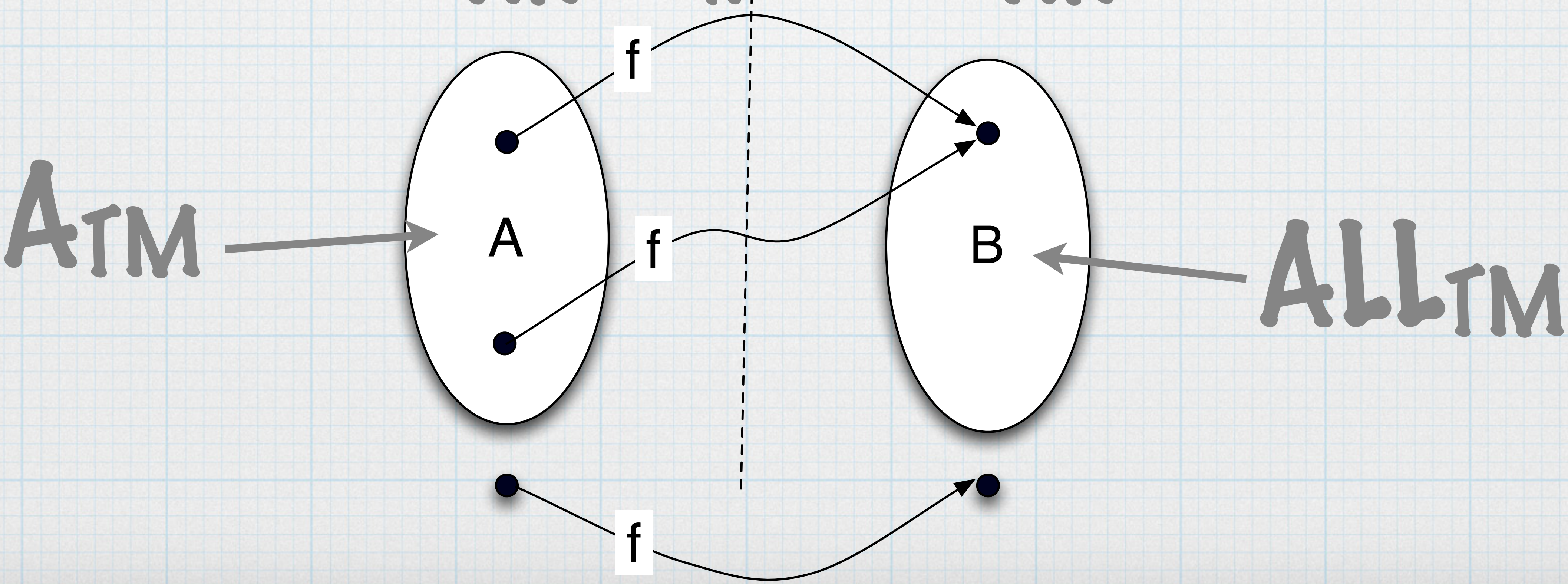


Computable Functions

$A \leq_m B$ iff there exists computable $f: \Sigma^* \to \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$

$f(\langle M,w \rangle) = \langle M' \rangle$
where M' runs M on w
and
accepts its input w' only if M accepts w.

$A_{TM} \leq_m ALL_{TM}$

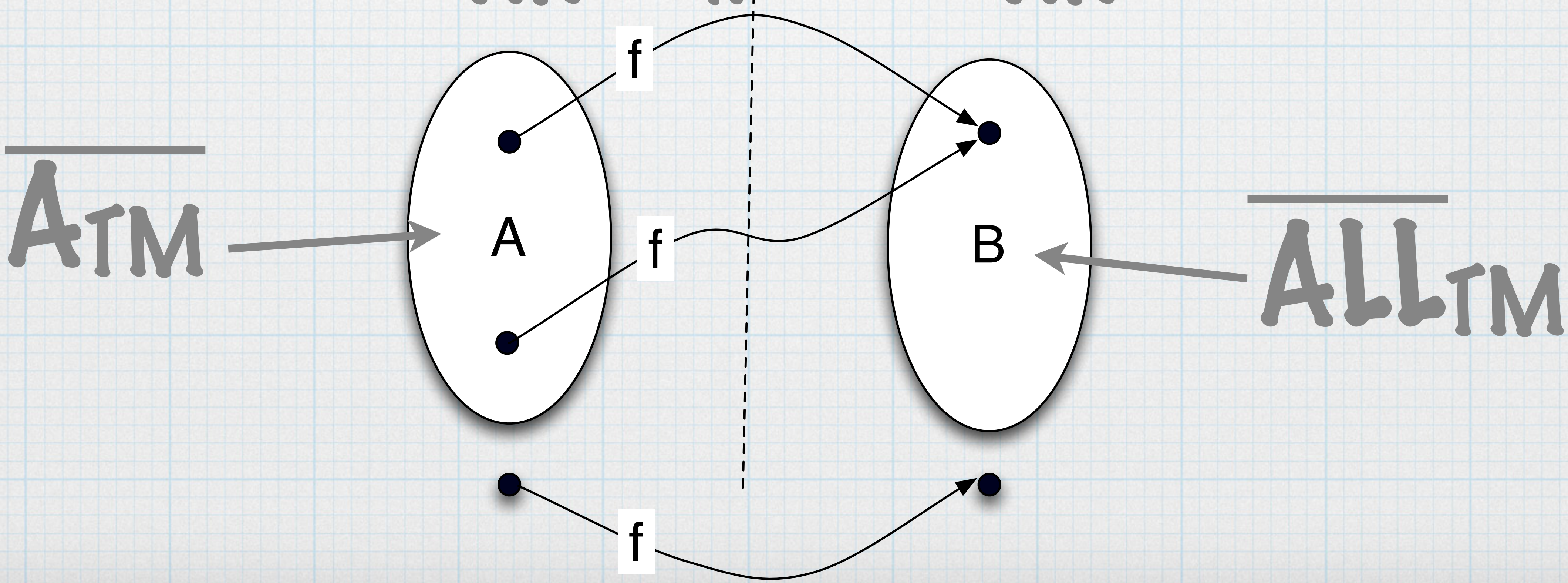


Computable Functions

$A \le_m B$ iff there exists computable $f: \Sigma^* \rightarrow \Sigma^*$ such that $w \in A \Leftrightarrow f(w) \in B$

f(<M,w>) = <M′>
where M′ runs M on w
and
accepts its input w′ only if M accepts w.

$\overline{A_{TM}} \le_m \overline{ALL_{TM}}$



Reductio

* If by assuming M decides B we can build M′ that decides A then …

➡ if B is decidable, A is decidable

➡ if A is undecidable, B is undecidable

Reductio

* If by assuming M decides B we can build M′ that decides A then ...
  ➡ if B is decidable, A is decidable
  ➡ if A is undecidable, B is undecidable

* If $A \leq_m B$ then ...
  ➡ if B is decidable, A is decidable
  ➡ if A is undecidable, B is undecidable

Reductio

* If by assuming M recognizes B we can build M′ that recognizes A then ...
  - ➡ if B is R.E., A is R.E.
  - ➡ if A is not R.E., B is not R.E.
* If A $\leq_m$ B then ...
  - ➡ if B is R.E., A is R.E.
  - ➡ if A is not R.E., B is not R.E.

Mapping Reducible

$DISJOINT_{TM} = \{ \langle M,N \rangle \mid L(M) \cap L(N) \text{ is empty} \}$

f(w) = ???

$\Leftrightarrow$

$E_{TM} \leq_m DISJOINT_{TM}$

Mapping Reducible

$DISJOINT_{TM}$ = { ⟨M,N⟩ | L(M) ∩ L(N) is empty }

f(w) = ???, if w = <M>,
f(w) = ???, otherwise.

⇔

$E_{TM} \leq_m DISJOINT_{TM}$

Mapping Reducible

$DISJOINT_{TM} = \{ \langle M,N \rangle \mid L(M) \cap L(N) \text{ is empty} \}$

f(w) = <M, EVERY >, if w = <M>
f(w) = < EVERY, EVERY>, otherwise
where
EVERY is a TM that accepts all strings
$\Leftrightarrow$

$E_{TM} \leq_m DISJOINT_{TM}$

Mapping Reducible

$EQ_{CFG} = \{ \langle G,H \rangle \mid G \;\&\; H \text{ are CFGs, } L(G) = L(H) \}$

f(w) = ???

$\Leftrightarrow$

$ALL_{CFG} \leq_m EQ_{CFG}$

Mapping Reducible

EQ_{CFG} = { ⟨G,H⟩ | G & H are CFGs, L(G) = L(H) }

f(w) = ???, if w = <G>
f(w) = ???, otherwise

⟺

$ALL_{CFG} \leq_m EQ_{CFG}$

Mapping Reducible

EQ_{CFG} = { ⟨G,H⟩ | G & H are CFGs, L(G) = L(H) }

f(<G>) = <G, EVERY >, if w = <G>,
f(w) = <EVERY, NONE> , otherwise
where
EVERY is a CFG that includes all strings
and NONE is a CFG that includes no strings.

$\Leftrightarrow$

$ALL_{CFG} \leq_m EQ_{CFG}$

Comparing Hardness

$\overline{E_{TM}} \leq_M A_{TM}$

$A_{TM} \leq_M \overline{E_{TM}}$

Comparing Hardness

$\overline{E_{TM}} \leq_M A_{TM}$

$A_{TM} \leq_M \overline{E_{TM}}$

* Given <M,w> generate <M'> where on input w', M' runs M on w.

Comparing Hardness

$\overline{E_{TM}} \leq_m A_{TM}$

$A_{TM} \leq_m \overline{E_{TM}}$

* Given <M,w> generate <M'> where on input w', M' runs M on w.
* Given w ≠ <M,w>, generate <EMPTY>, where L(EMPTY) = ϕ

Comparing Hardness

$\overline{E_{TM}} \leq_M A_{TM}$

* Given <M>, generate <M′,ϵ> where on w′, M′ dovetails running M on all w & accepts w′ if any w ∈ L(M).

$A_{TM} \leq_M \overline{E_{TM}}$

* Given <M,w> generate <M′> where on input w′, M′ runs M on w.
* Given w ≠ <M,w>, generate <EMPTY>, where L(EMPTY) = ϕ

Comparing Hardness

$\overline{E_{TM}} \leq_M A_{TM}$

- Given <M>, generate <M′,ϵ> where on w′, M′ dovetails running M on all w & accepts w′ if any w ϵ L(M),
- Given w generate <ALL,ϵ>

$A_{TM} \leq_M \overline{E_{TM}}$

- Given <M,w> generate <M′> where on input w′, M′ runs M on w.
- Given w ≠ <M,w>, generate <EMPTY>, where L(EMPTY) = ϕ

Turing Equivalence

$\overline{E_{TM}} \leq_M A_{TM}$

* Given <M>, generate <M′,ϵ> where on w′, M′ dovetails running M on all w & accepts w′ if any w ϵ L(M).
* Given w generate <ALL,ϵ>

$A_{TM} \leq_M \overline{E_{TM}}$

* Given <M,w> generate <M′> where on input w′, M′ runs M on w.
* Given w ≠ <M,w>, generate <EMPTY>, where L(EMPTY) = ϕ

$\overline{E_{TM}} \equiv_M A_{TM}$

Turing Equivalence

$E_{TM} \leq_M \overline{A_{TM}}$

* Given <M>, generate <M′,ϵ> where on w′, M′ dovetails running M on all w & accepts w′ if any w ϵ L(M).
* Given w generate <ALL,ϵ>

$\overline{A_{TM}} \leq_M E_{TM}$

* Given <M,w> generate <M′> where on input w′, M′ runs M on w.
* Given w ≠ <M,w>, generate <EMPTY>, where L(EMPTY) = ϕ

$$E_{TM} \equiv_M \overline{A_{TM}}$$

Rice's Theorem

Any non-trivial property of a Turing machine's language is undecidable.

Trivial Pursuit

* $ISALanguage_{TM}$ =
  $\{\langle M \rangle |$ M is a TM and $L(M) \subseteq \Sigma^*\}$
* $UNRecognizable_{TM}$ =
  $\{\langle M \rangle |$ M is a TM and L(M) is not recognizable $\}$
* $LITTLE_{TM}$ =
  $\{\langle M \rangle | M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ is a TM and $|Q| < 99\}$

Rice's Theorem

Any non-trivial property of a Turing machine's language is undecidable.

Rice's Theorem

Suppose that L is a language with

$\varnothing \subset L \subset \{\langle M \rangle \mid \langle M \rangle$ is a valid Turing machine $\}$

such that

if $L(M) = L(N)$ then $\langle M \rangle \in L$ iff $\langle N \rangle \in L$

then L is undecidable.

Rice's Theorem

Suppose that L is a language with

$\emptyset \subset L \subset \{\langle M \rangle \mid \langle M \rangle$ is a valid Turing machine $\}$

such that

if $L(M) = L(N)$ then $\langle M \rangle \in L$ iff $\langle N \rangle \in L$

then L and $\overline{L}$ are undecidable.

Rice's Theorem

Suppose that L is a language with

$\varnothing \subset L \subset \{\langle M \rangle \mid \langle M \rangle$ is a valid Turing machine $\}$

such that

if $L(M) = L(N)$ then $\langle M \rangle \in L$ iff $\langle N \rangle \in L$

and for all $\langle M \rangle \in L$, $L(M) \neq \phi$,

then L and $\overline{L}$ are undecidable.

Suppose that L is a language with

$\varnothing \subset L \subset \{\langle M \rangle \mid \langle M \rangle$ is a valid Turing machine $\}$

such that

if $L(M) = L(N)$ then $\langle M \rangle \in L$ iff $\langle N \rangle \in L$

and for all $\langle M \rangle \in L$, $L(M) \neq \phi$,

then L and $\overline{L}$ are undecidable.

Suppose that L is a language with

$\emptyset \subset L \subset \{\langle M \rangle \mid \langle M \rangle$ is a valid Turing machine $\}$

such that

if $L(M) = L(N)$ then $\langle M \rangle \in L$ iff $\langle N \rangle \in L$

and for all $\langle M \rangle \in L$, $L(M) \neq \phi$,

then L and $\overline{L}$ are undecidable.

PROOF: Show that $A_{TM} \leq_m L$.

Suppose that L is a language with

$\varnothing \subset L \subset \{\langle M\rangle \mid \langle M\rangle$ is a valid Turing machine $\}$

such that

if $L(M) = L(N)$ then $\langle M\rangle \in L$ iff $\langle N\rangle \in L$

and for all $\langle M\rangle \in L$, $L(M) \neq \phi$,

then L and $\overline{L}$ are undecidable.

PROOF: Show that $A_{TM} \leq_m L$.

Find a computable function $f(\langle M,w\rangle) = \langle M'\rangle$ such that

if $w \in L(M)$ then $\langle M'\rangle \in L$

if $w \notin L(M)$ then $\langle M'\rangle \notin L$

Find a computable function f (<M,w>) = <M′> such that

if $w \in L(M)$ then $\langle M' \rangle \in L$

if $w \notin L(M)$ then $\langle M' \rangle \notin L$

Choose any $\langle M_{inL} \rangle \in L$.

Find a computable function f (<M,w>) = <M′> such that

if $w \in L(M)$ then $L(M') = L(M_{inL})$

if $w \notin L(M)$ then $\langle M' \rangle = \phi$

Find a computable function $f(\langle M,w \rangle) = \langle M' \rangle$ such that

if $w \in L(M)$ then $\langle M' \rangle \in L$

if $w \notin L(M)$ then $\langle M' \rangle \notin L$

Choose any $\langle M_{inL} \rangle \in L$.

Let f = On input $\langle M, w \rangle$, construct a TM M′ which:

on input w′, simulates M on w and

if M accepts w, runs M_{inL} on w′

else rejects.

Decidable Questions?

* $REVERSIBLE_{TM} = \{\langle M \rangle \mid w \in L(M) \text{ iff } w^R \in L(M)\}$
* $REGULAR_{TM} = \{\langle M \rangle \mid L(M) \text{ is regular}\}$
* $DISJOINT_{TM} = \{\langle M,N \rangle \mid L(M) \cap L(N) \text{ is empty}\}$
* $PRIME_{TM} = \{\langle M \rangle \mid w \in L(M) \Rightarrow |w| \text{ is prime}\}$
* $QUAD_{TM} = \{\langle M \rangle \mid M \text{ runs} < |w|^2 \text{ steps on all inputs}\}$