5. Reducibility

- Reductions
  - Example, formal definition
  - Computable functions
  - Mapping reducibility
- Undecidable problems
  - computation histories
  - Rice's Theorem
  - Post Correspondence Problem
  - a non-RE and non-co-RE problem

Reductions

- Given a new problem NEW, want to determine if it is undecidable
  - prove from scratch that the problem is undecidable (dream up a diag. argument)
  - show how to transform a known undecidable problem OLD into NEW so that solution to NEW can be used to solve OLD
- A **reduction** is a way of converting one problem into another such that a solution to the second problem can be used to solve the first problem.

Reductions

Reductions are one of the most important and widely used techniques in theoretical Computer Science.

- especially for proving problems “hard”
  - often difficult to do “from scratch”
  - sometimes not known how to do from scratch
  - reductions allow proof by giving an algorithm to perform the transformation

Example Reduction

- Try to prove undecidable:

  $A_{TM} = \{<M, w> \mid M \text{ accepts input } w\}$

- We know this language is undecidable:

  $HALT = \{<M, w> \mid M \text{ halts on input } w\}$

- Idea:

  reduction

  - suppose A_{TM} is decidable
  - show that we can use A_{TM} to decide HALT
  - conclude HALT is decidable. Contradiction.

Example Reduction

- Deciding HALT using a procedure that decides A_{TM} ("reducing HALT to A_{TM}").
  - on input <M, w>
  - check if <M, w> $\in A_{TM}$
    - if yes, then M halts on w; **ACCEPT**
    - if no, then M either rejects w or it loops in w
  - construct M' by swapping q_{accept}/q_{reject} in M
  - check if <M', w> $\in A_{TM}$
    - if yes, then M' accepts w, so M rejects w; **ACCEPT**
    - if no, then M neither accepts nor rejects w; **REJECT**

Another Example

- Try to prove undecidable:

  $E_{TM} = \{\langle M\rangle \mid L(M) = \emptyset\}$

- which problem should we **reduce from**?
  - $HALT = \{\langle M, w\rangle \mid M \text{ halts on input } w\}$
  - $A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts input } w\}$

- Proof:
  - suppose E_{TM} is decidable
  - we showed how to use E_{TM} to decide A_{TM}
  - conclude A_{TM} is decidable. Contradiction.

Another Example

- On input <M, w>:
  - construct TM M' from description of M
    - on input x, if $x \neq w$, then reject
    - else simulate M on x, and accept if M does.
  - check if <M'> $\in E_{TM}$
    - if no, M must accept w; **ACCEPT**
    - if yes, M cannot accept w; **REJECT**

Definition of Reduction

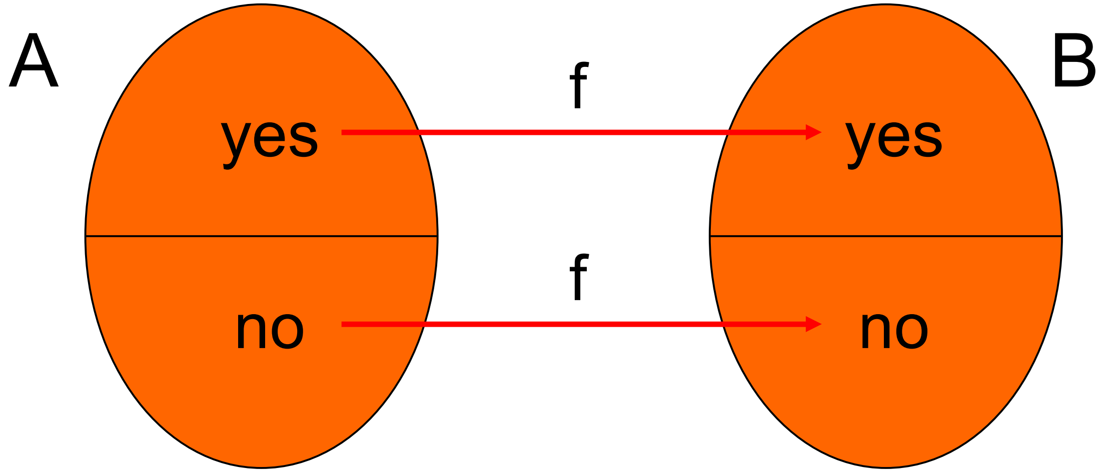
- Can you reduce co-HALT to HALT?

- We know that HALT is RE
- Does this show that co-HALT is RE?
  - recall, we showed co-HALT is not RE

- our notion of reduction cannot distinguish complements

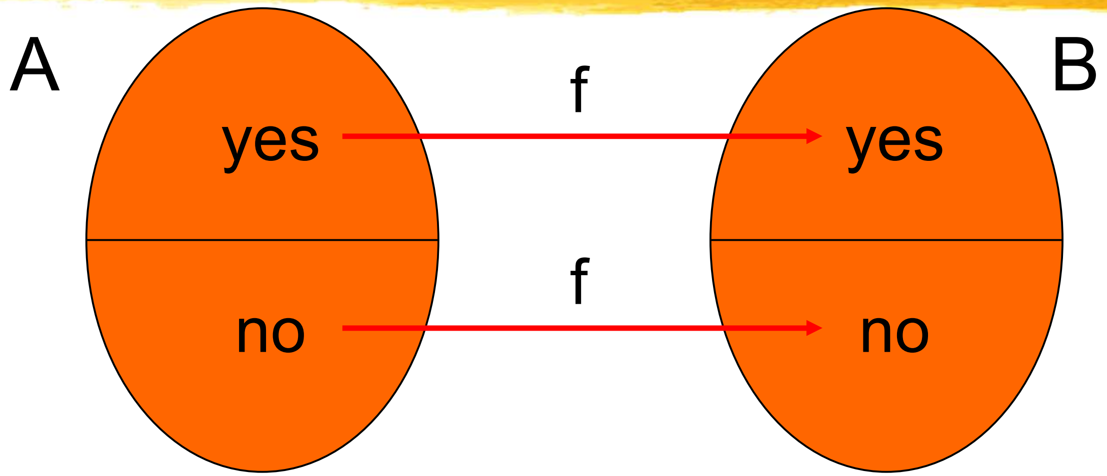
Definition of Reduction

- More refined notion of reduction:
  - “many-one” reduction (commonly)
  - “mapping” reduction (book)



reduction from language A to language B

Definition of Reduction



- function f should be computable

Definition: $f : \Sigma^* \rightarrow \Sigma^*$ is computable if there exists a TM M_f such that on every $w \in \Sigma^*$ M_f halts on w with $f(w)$ written on its tape.

Definition of Reduction

Definition: A is mapping reducible to B, written $A \leq_m B$, if there is a computable function f such that for all w

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

- "yes maps to yes and no maps to no" means:

  $w \in A$ maps to $f(w) \in B$

  & $w \notin A$ maps to $f(w) \notin B$

- f is called the reduction of A to B

Using Reductions

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

Proof:

- decider for A: on input w, compute f(w), run decider for B, do whatever it does.

- Main use: given language NEW, prove it is undecidable by showing OLD $\leq_m$ NEW, where OLD is known to be undecidable
  - common to reduce in wrong direction!

Using Reductions

Theorem: if A $\leq_m$ B and B is RE, then A is RE

Proof:

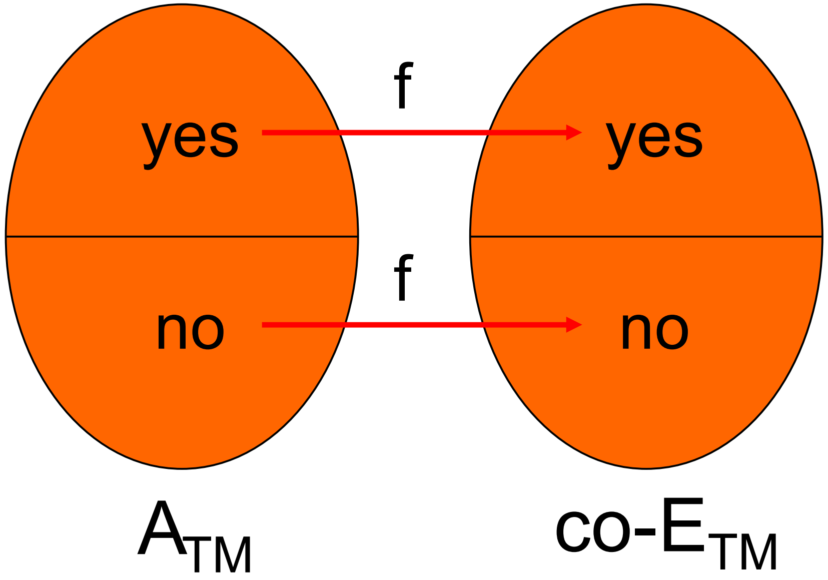
- TM for recognizing A: on input w, compute f(w), run TM that recognizes B, do whatever it does.

- Main use: given language NEW, prove it is not RE by showing OLD $\leq_m$ NEW, where OLD is known to be not RE.

Mapping Reduction Example

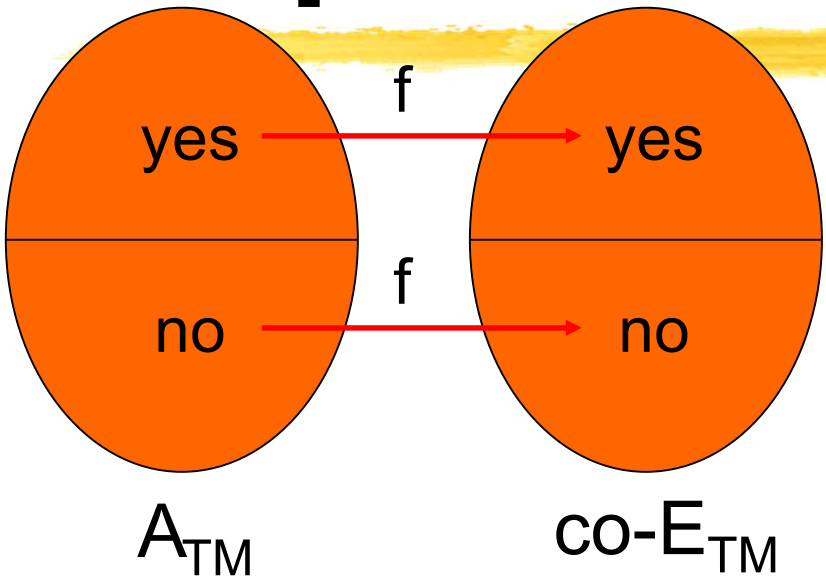
- E_{TM} is undecidable. Consider:

$$\text{co-}E_{TM} = \{\langle M\rangle \mid L(M) \neq \emptyset\}$$



- f(<M, w>) = <M'> where M' is TM that
  - on input x, if $x \neq w$, then reject
  - else simulate M on x, and accept if M does
- f clearly computable

Mapping Reduction Example



- f(<M, w>) = <M'> where M' is TM that
  - on input x, if $x \neq w$, then reject
  - else simulate M on x, and accept if M does
- f clearly computable

- yes maps to yes?
  - if <M, w> $\in A_{TM}$, then f(M, w) $\in$ co-E_{TM}
- no maps to no?
  - if <M, w> $\notin A_{TM}$, then f(M, w) $\notin$ co-E_{TM}

Undecidable Problems

Theorem: The language

REGULAR = {<M> | M is a TM and L(M) is regular}

is undecidable.

Proof:

- reduce from A_{TM} (i.e. show $A_{TM} \leq_m$ REGULAR)
- what should f(<M, w>) produce?

Undecidable Problems

Proof:

- f(<M, w>) = <M'> described below

M' on input x:

• if x has form 0^n1^n, accept

• else simulate M on w and accept if M accepts w

• is f computable?

• YES maps to YES?

$<M, w> \in A_{TM} \Rightarrow f(M, w) \in REGULAR$

• NO maps to NO?

$<M, w> \notin A_{TM} \Rightarrow f(M, w) \notin REGULAR$

Computation Histories

- Recall configuration of a TM: string uqv with u,v $\in \Gamma^*$, q $\in$ Q
- The sequence of configurations M goes through on input w is a computation history of M on input w
  - may be accepting, or rejecting
  - reserve the term for halting computations
  - nondeterministic machines may have several computation histories for a given input.

Linear Bounded Automata

LBA definition: TM that is prohibited from moving head off **right** side of input.

- machine prevents such a move, just like a TM prevents a move off left of tape

- How many possible configurations for a LBA M on input w with $|w| = n$, m states, and $p = |\Gamma|$?
  - counting gives: mnp^n

Dec. and Undec. Problems

- two problems we have seen with respect to TMs, now regarding LBAs:
  - LBA acceptance:

    $A_{LBA} = \{\langle M, w\rangle \mid \text{LBA } M \text{ accepts input } w\}$
  - LBA emptiness:

    $E_{LBA} = \{\langle M\rangle \mid \text{LBA } M \text{ has } L(M) = \emptyset\}$
- Both decidable? both undecidable? one decidable?

Dec. and Undec. Problems

Theorem: A_{LBA} is decidable.

Proof:

- input <M, w> where M is a LBA
- key: only mnp^n configurations
- if M hasn't halted after this many steps, it must be looping forever.
- simulate M for mnp^n steps
- if it halts, accept or reject accordingly,
- else reject since it must be looping

Dec. and Undec. Problems

Theorem: E_{LBA} is undecidable.

Proof:

- reduce from co-A_{TM} (i.e. show co-$A_{TM} \leq_m E_{LBA}$)
- what should f(<M, w>) produce?
- Idea:
  - produce LBA B that accepts exactly the accepting computation histories of M on input w

Dec. and Undec. Problems

Proof:

- $f(<M, w>) = <B>$ described below

on input x, check if x has form

$$\#C_1\#C_2\#C_3\#\ldots\#C_k\#$$

- check that C_1 is the start configuration for M on input w
- check that $C_i \Rightarrow^1 C_{i+1}$
- check that C_k is an accepting configuration for M

- is B an LBA?
- is f computable?
- YES maps to YES?

  $<M, w> \in \text{co-}A_{TM} \Rightarrow f(M, w) \in E_{LBA}$

- NO maps to NO?

  $<M, w> \notin \text{co-}A_{TM} \Rightarrow f(M, w) \notin E_{LBA}$

Dec. and Undec. Problems

- two problems regarding Context-Free Grammars:
  - does a CFG generate all strings:

    $ALL_{CFG} = \{<G> \mid G \text{ is a CFG and } L(G) = \Sigma^*\}$
  - CFG emptiness:

    $E_{CFG} = \{<G> \mid G \text{ is a CFG and } L(G) = \emptyset\}$

- Both decidable? both undecidable? one decidable?

Dec. and Undec. Problems

Theorem: ALL_{CFG} is undecidable.

Proof:

- reduce from co-A_{TM} (i.e. show co-$A_{TM} \leq_m ALL_{CFG}$)
- what should f(<M, w>) produce?
- Idea:
  - produce CFG G that generates all strings that are ***not*** accepting computation histories of M on w

Dec. and Undec. Problems

Proof:

- build a NPDA, then convert to CFG
- want to accept strings not of this form,

$$\#C_1\#C_2\#C_3\#\ldots\#C_k\#$$

plus strings of this form but where

- C_1 is not the start config. of M on input w, or
- C_k is not an accept config. of M on input w, or
- C_i does not yield in one step C_{i+1} for some i

Dec. and Undec. Problems

Proof:

- our NPDA nondeterministically checks one of:
  - C_1 is not the start config. of M on input w, or
  - C_k is not an accept config. of M on input w, or
  - C_i does not yield in one step C_{i+1} for some i
  - input has fewer than two #'s
- to check third condition:
  - nondeterministically guess C_i starting position
  - how to check that C_i doesn't yield in 1 step C_{i+1}?

Dec. and Undec. Problems

Proof:

- checking:
  - C_i does not yield in one step C_{i+1} for some i
- push C_i onto stack
- at #, start popping C_i and compare to C_{i+1}
  - accept if mismatch away from head location, or
  - symbols around head changed in a way inconsistent with M's transition function.
- is everything described possible with NPDA?

Dec. and Undec. Problems

Proof:

- Problem: cannot compare C_i to C_{i+1}
- could prove in same way that proved

  $\{ww: w \in \Sigma^*\}$ not context-free
- recall that

  $\{ww^R: w \in \Sigma^*\}$ is context-free
- free to tweak construction of G in the reduction
- solution: write computation history:

$$\#C_1\#C_2^R\#C_3\#C_4^R\ldots\#C_k\#$$

Dec. and Undec. Problems

Proof:

- $f(\langle M, w\rangle) = \langle G\rangle$ equiv. to NPDA below:

on input x, accept if not of form:

$$\#C_1\#C_2^R\#C_3\#C_4^R\ldots\#C_k\#$$

- accept if C_1 is the not the start configuration for M on input w
- accept if check that C_i does not yield in one step C_{i+1}
- accept if C_k is not an accepting configuration for M

- is f computable?
- YES maps to YES?

  $\langle M, w\rangle \in \text{co-}A_{TM} \Rightarrow f(M, w) \in ALL_{CFG}$

- NO maps to NO?

  $\langle M, w\rangle \notin \text{co-}A_{TM} \Rightarrow f(M, w) \notin ALL_{CFG}$

Rice's Theorem

- We have seen that the following properties of TM's are undecidable:
  - TM accepts string w
  - TM halts on string w
  - TM accepts the empty language
  - TM accepts a regular language
- Can we describe a single generic reduction for all these proofs?
- Yes. *Every* property of TMs is undecidable!

Rice's Theorem

- A TM property is a language P for which
  - if $L(M_1) = L(M_2)$ then $<M_1> \in P$ iff $<M_2> \in P$
- TM property P is nontrivial if
  - there exists a TM M_1 for which $<M_1> \in P$, and
  - there exists a TM M_2 for which $<M_2> \notin P$.

Rice's Theorem: Every nontrivial TM property is undecidable.

Rice's Theorem

- The setup:
  - let $T_\varnothing$ be a TM for which $L(T_\varnothing) = \varnothing$
    - technicality: if $<T_\varnothing> \in P$ then work with property co-P instead of P.
    - conclude co-P undecidable; therefore P undec. due to closure under complement
  - so, WLOG, assume $<T_\varnothing> \notin P$
  - non-triviality ensures existence of TM M_1 such that $<M_1> \in P$

Rice's Theorem

Proof:

- reduce from A_{TM} (i.e. show $A_{TM} \leq_m P$)
- what should f(<M, w>) produce?
- f(<M, w>) = <M'> described below:

M' on input x,

- accept iff M accepts w and M_1 accepts x

(intersection of two RE languages)

- f computable?
- YES maps to YES?

$<M, w> \in A_{TM} \Rightarrow$
$L(f(M, w)) = L(M_1) \Rightarrow$
$f(M, w) \in P$

Rice's Theorem

Proof:

- reduce from A_{TM} (i.e. show $A_{TM} \leq_m P$)
- what should f(<M, w>) produce?
- f(<M, w>) = <M'> described below:

M' on input x,

• accept iff M accepts w and M_1 accepts x

(intersection of two RE languages)

• NO maps to NO?

$<M, w> \notin A_{TM} \Rightarrow$
$L(f(M, w)) = L(T_{\emptyset}) \Rightarrow$
$f(M, w) \notin P$

Post Correspondence Problem

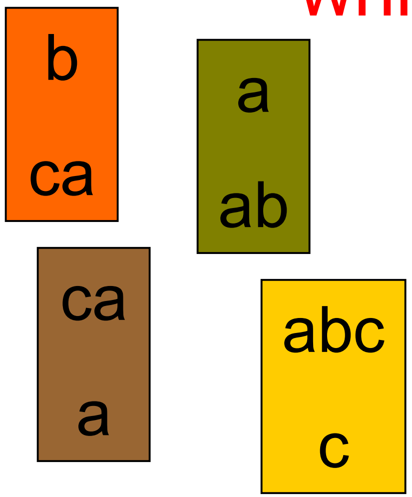
- There are many undecidable problems unrelated to TMs and automata

- classic example: Post Correspondence Problem

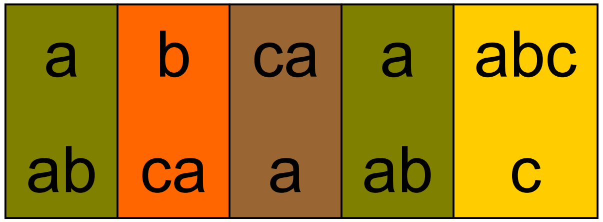
$$PCP = \{<(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)> \mid x_i, y_i \in \Sigma^* \text{ and there exists } (a_1, a_2, \ldots, a_n) \text{ for which } x_{a_1}x_{a_2}\ldots x_{a_n} = y_{a_1}y_{a_2}\ldots y_{a_n}\}$$

Post Correspondence Problem

$$PCP = \{\langle (x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k) \rangle \mid x_i, y_i \in \Sigma^* \text{ and there exists } (a_1, a_2, \ldots, a_n) \text{ for which } x_{a_1}x_{a_2}\ldots x_{a_n} = y_{a_1}y_{a_2}\ldots y_{a_n}\}$$



"tiles"



abcaaabc = abcaaabc

"match"

Post Correspondence Problem

<u>Theorem</u>: PCP is undecidable.

Proof:

- reduce from A_{TM} (i.e. show $A_{TM} \leq_m$ PCP)
- two-step reduction makes it easier
- first, show $A_{TM} \leq_m$ MPCP
  (MPCP = "modified PCP")
- next, show MPCP $\leq_m$ PCP

Post Correspondence Problem

$$\text{MPCP} = \{\langle (x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)\rangle \mid x_i, y_i \in \Sigma^* \text{ and there exists } (a_1, a_2, \ldots, a_n) \text{ for which } x_1 x_{a_1} x_{a_2} \ldots x_{a_n} = y_1 y_{a_1} y_{a_2} \ldots y_{a_n}\}$$

Proof of MPCP $\leq_m$ PCP:

- notation: for a string $u = u_1 u_2 u_3 \ldots u_m$
  - $*u$ means the string $*u_1*u_2*u_3*u_4\ldots*u_m$
  - $u*$ means the string $u_1*u_2*u_3*u_4\ldots*u_m*$
  - $*u*$ means the string $*u_1*u_2*u_3*u_4\ldots*u_m*$

Post Correspondence Problem

Proof of MPCP $\leq_m$ PCP:

- given an instance $(x_1, y_1), \ldots, (x_k, y_k)$ of MPCP
- produce an instance of PCP:

  $(*x_1, *y_1*)$, $(*x_1, y_1*)$, $(*x_2, y_2*)$, $\ldots$, $(*x_k, y_k*)$, $(*\blacklozenge, \blacklozenge)$

- YES maps to YES?
  - given a match in original MPCP instance, can produce a match in the new PCP instance
- NO maps to NO?
  - given a match in the new PCP instance, can produce a match in the original MPCP instance

Post Correspondence Problem

- YES maps to YES?
  - given a match in original MPCP instance, can produce a match in the new PCP instance

x_1	x_4	x_5	x_2	x_1	x_3	x_4	x_4
y_1	y_4	y_5	y_2	y_1	y_3	y_4	y_4

$*x_1$	$*x_4$	$*x_5$	$*x_2$	$*x_1$	$*x_3$	$*x_4$	$*x_4$	$*\blacklozenge$
$*y_1*$	y_4*	y_5*	y_2*	y_1*	y_3*	y_4*	y_4*	$\blacklozenge$

Post Correspondence Problem

can't match unless start with this tile

- NO maps to NO?
  - given a match in the new PCP instance, can produce a match in the original MPCP instance

$*x_1$	$*x_4$	$*x_5$	$*x_2$	$*x_1$	$*x_3$	$*x_4$	$*x_4$	$*\blacklozenge$
$*y_1*$	y_4*	y_5*	y_2*	y_1*	y_3*	y_4*	y_4*	$\blacklozenge$

x_1	x_4	x_5	x_2	x_1	x_3	x_4	x_4
y_1	y_4	y_5	y_2	y_1	y_3	y_4	y_4

"*" symbols must align

can only appear at the end

Post Correspondence Problem

Proof of $A_{TM} \leq_m$ MPCP:

- given instance of A_{TM}: <M, w>
- idea: a match will record an accepting computation history for M on input w
- start tile records starting configuration:
  - add tile $(\#, \#q_0w_1w_2w_3\ldots w_n\#)$

$$\left[\frac{\#}{\#q_0w_1w_2\ldots w_n\#}\right] = \left[\frac{\#}{\#C_1\#}\right]$$

Post Correspondence Problem

$\#$ / $\# q_0 w_1 w_2 \ldots w_n \#$	? / ?	? / ?	…	? / ?	=	$\#C_1\#$ / $\#C_1\#C_2\#$

- tiles for head motions to the right:
  - for all a, b $\in \Gamma$ and all q, r $\in$ Q with q $\neq q_{reject}$, if $\delta(q, a) = (r, b, R)$, add tile (qa, br)

qa
br

- tiles for head motions to the left:
  - for all a,b,c $\in \Gamma$ and all q, r $\in$ Q with q $\neq q_{reject}$, if $\delta(q, a) = (r, b, L)$, add tile (cqa, rcb)

cqa
rcb

Post Correspondence Problem

#	?	?	…	?	=	$\#C_1\#$
$\# \, q_0 w_1 w_2 \ldots w_n \#$	?	?		?		$\#C_1\#C_2\#$

- tiles for copying (not near head)
  - for all a $\in \Gamma$, add tile (a, a)
- tiles for copying # marker
  - add tile (#, #)
- tiles for copying # marker and adding _ to end of tape
  - add tile (#, _#)

a
a

#
#

#
_#

Post Correspondence Problem

#	?	…	?	=	$\#uaq_{accept}v\#$
$\#uaq_{accept}v\#$	?		?		$\#uaq_{accept}v\#uq_{accept}v\#$

- tiles for deleting symbols to left of q_{accept}
  - for all $a \in \Gamma$, add tile $(aq_{accept}, q_{accept})$

aq_{accept}
q_{accept}

Post Correspondence Problem

$$\left[\frac{\#}{\#uq_{accept}\#}\right]\left[\frac{?}{?}\right]\ \cdots\ \left[\frac{?}{?}\right] = \left[\frac{\#uq_{accept}av\#}{\#uq_{accept}av\#uq_{accept}v\#}\right]$$

- tiles for deleting symbols to right of q_{accept}
  - for all $a \in \Gamma$, add tile $(q_{accept}a,\ q_{accept})$

$$\left[\frac{q_{accept}a}{q_{accept}}\right]$$

Post Correspondence Problem

$$\left[\frac{\#}{\#q_{accept}\#}\right]\left[\frac{?}{?}\right]\cdots\left[\frac{?}{?}\right] = \left[\frac{\#q_{accept}\#\#}{\#q_{accept}\#\#}\right]$$

- tiles for completing the match
  - add tile $(q_{accept}\#\#, \#)$

$$\left[\frac{q_{accept}\#\#}{\#}\right]$$

Post Correspondence Problem

- YES maps to YES?
  - by construction, if M accepts w, there is a way to assemble the tiles to achieve this match:

$\#C_1\#C_2\#C_3\#\ldots\#C_m\#$
$\#C_1\#C_2\#C_3\#\ldots\#C_m\#$

where $\#C_1\#C_2\#C_3\#\ldots\#C_m\#$ is an accepting computation history

- NO maps to NO?
  - sketch: at any step if the "intended" next tile is not used, then it is impossible to recover and produce a match in the end (case analysis)

Post Correspondence Problem

We have proved:

Theorem: PCP is undecidable.

by showing:

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

Beyond RE and co-RE

- We saw (by a counting argument) that there is *some* language that is neither RE nor co-RE.
- We will prove this for a natural language:

$$EQ_{TM} = \{<M_1, M_2> \mid L(M_1) = L(M_2)\}$$

- Recall:
  - A_{TM} is undecidable, but RE, therefore not in co-RE
  - co-A_{TM} is undecidable, but co-RE, therefore not in RE

Beyond RE and co-RE

<u>Theorem</u>: EQ_{TM} is neither RE nor co-RE.

Proof:

- not RE:
  - reduce from co-A_{TM} (i.e. show co-$A_{TM} \leq_m EQ_{TM}$)
  - what should f(<M, w>) produce?
- not co-RE:
  - reduce from A_{TM} (i.e. show $A_{TM} \leq_m EQ_{TM}$)
  - what should f(<M, w>) produce?

Beyond RE and co-RE

Proof ($A_{TM} \leq_m EQ_{TM}$)

- $f(<M, w>) = <M_1, M_2>$ described below:

TM M_1: on input x,

- accept

TM M_2: on input x,

- simulate M on input w
- accept if M accepts w

- YES maps to YES?

  $<M, w> \in A_{TM} \Rightarrow$
  $L(M_1) = \Sigma^*, L(M_2) = \Sigma^*, \Rightarrow$
  $f(<M, w>) \in EQ_{TM}$

- NO maps to NO?

  $<M, w> \notin A_{TM} \Rightarrow$
  $L(M_1) = \Sigma^*, L(M_2) = \emptyset \Rightarrow$
  $f(<M, w>) \notin EQ_{TM}$

Beyond RE and co-RE

Proof (co-$A_{TM} \leq_m EQ_{TM}$)

- f(<M, w>) = <M_1, M_2> described below:

TM M_1: on input x,

- reject

TM M_2: on input x,

- simulate M on input w
- accept if M accepts w

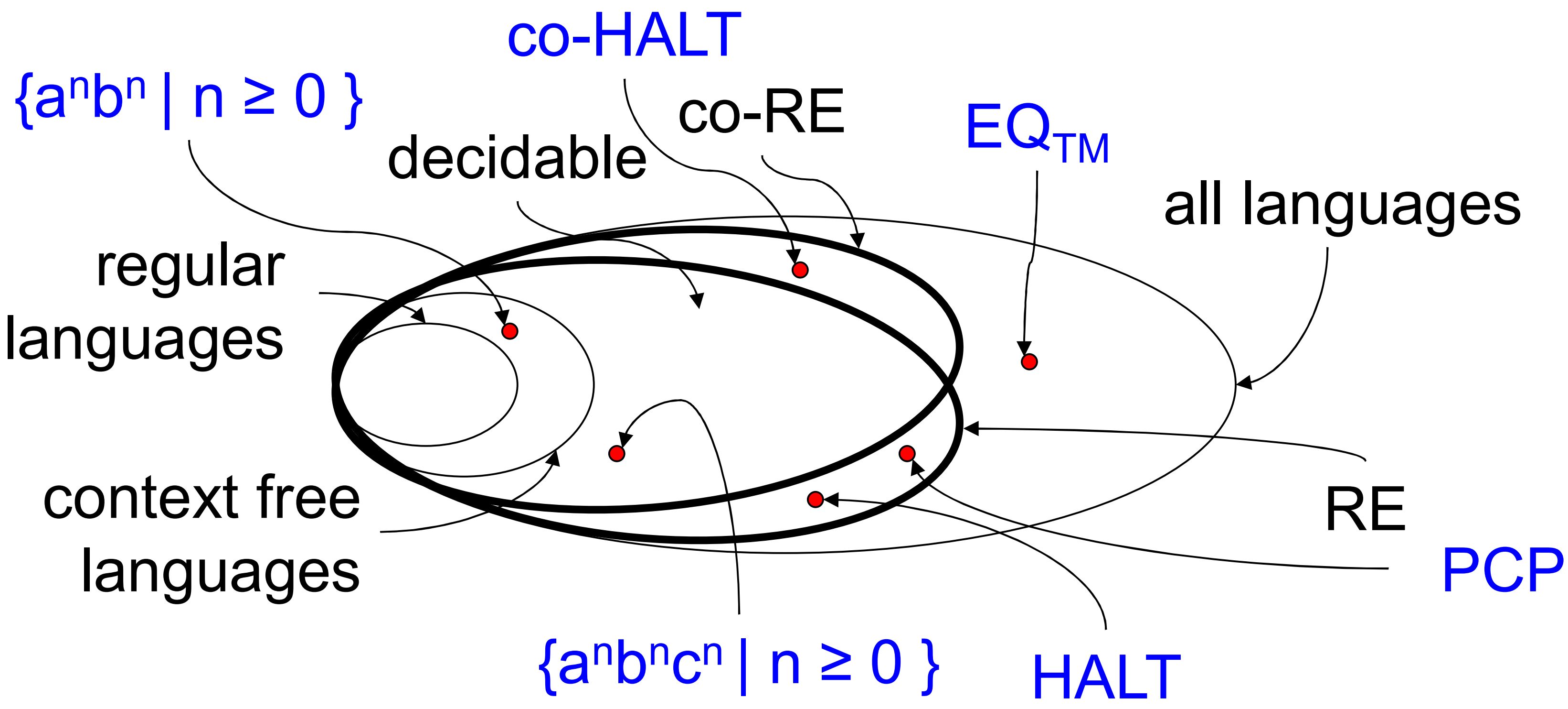
- YES maps to YES?

  <M, w> $\in$ co-$A_{TM} \Rightarrow$
  $L(M_1) = \emptyset$, $L(M_2) = \emptyset \Rightarrow$
  f(<M, w>) $\in EQ_{TM}$

- NO maps to NO?

  <M, w> $\notin$ co-$A_{TM} \Rightarrow$
  $L(M_1) = \emptyset$, $L(M_2) = \Sigma^*$, $\Rightarrow$
  f(<M, w>) $\notin EQ_{TM}$

Summary



co-HALT
$\{a^n b^n \mid n \geq 0\}$
co-RE
EQ_{TM}
decidable
all languages
regular
languages
context free
languages
RE
PCP
$\{a^n b^n c^n \mid n \geq 0\}$
HALT