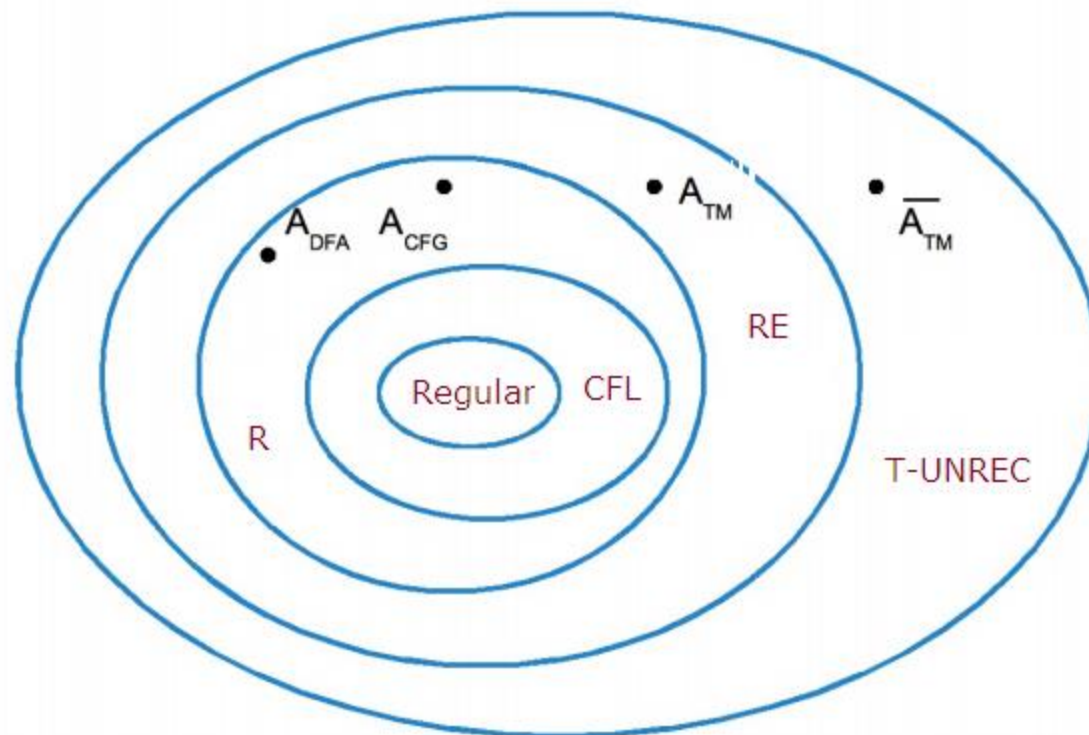


Reducibility

A way to show some languages are
undecidable !

<https://www.andrew.cmu.edu/user/ko/pdfs/lecture-16.pdf>

THE LANDSCAPE OF THE CHOMSKY HIERARCHY



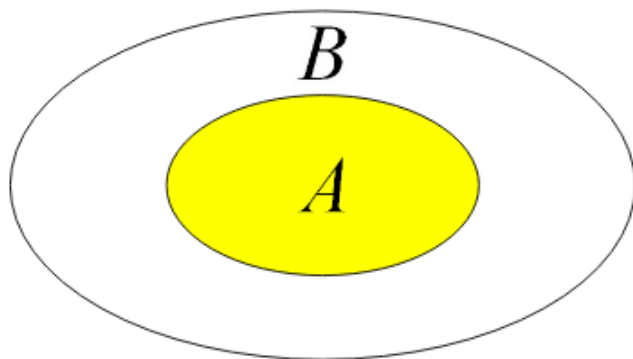
REDUCIBILITY

- A **reduction** is a way of converting one problem to another problem, so that the solution to the second problem can be used to solve the first problem.
 - Finding the area of a rectangle, reduces to measuring its width and height
 - Solving a set of linear equations, reduces to inverting a matrix.

Problem A is reduced to problem B



If we can solve problem B then
we can solve problem A .



A reduces to B

- $A \leq B$
- Find area of a rectangle \leq find length and find width of rectangle.
- Solving B means you know how to solve the fundamental ingredients (which are needed to solve A).
- A solution to B can be used to solve A.
- Note, a solution to A may not be enough to solve B.
 - Knowing area of a rectangular is not enough to find its length and width !!

A reduces to B

- $A \leq B$
- Solving B means you know how to solve the fundamental ingredients (which are needed to solve A).
- A solution to B can be used to solve A.
- An algorithm that solves B can be converted to an algorithm that solves A

A reduces to B

- $A \leq B$
- If B is decidable, then so is A.
- Contrapositive, if A is undecidable then so is B.

Problem A is reduced to problem B



If B is decidable then A is decidable.



If A is undecidable then B is undecidable.

PROVING UNDECIDABILITY VIA REDUCTIONS

THEOREM 5.1

$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$ is undecidable.

PROVING UNDECIDABILITY VIA REDUCTIONS

THEOREM 5.1

$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$ is undecidable.

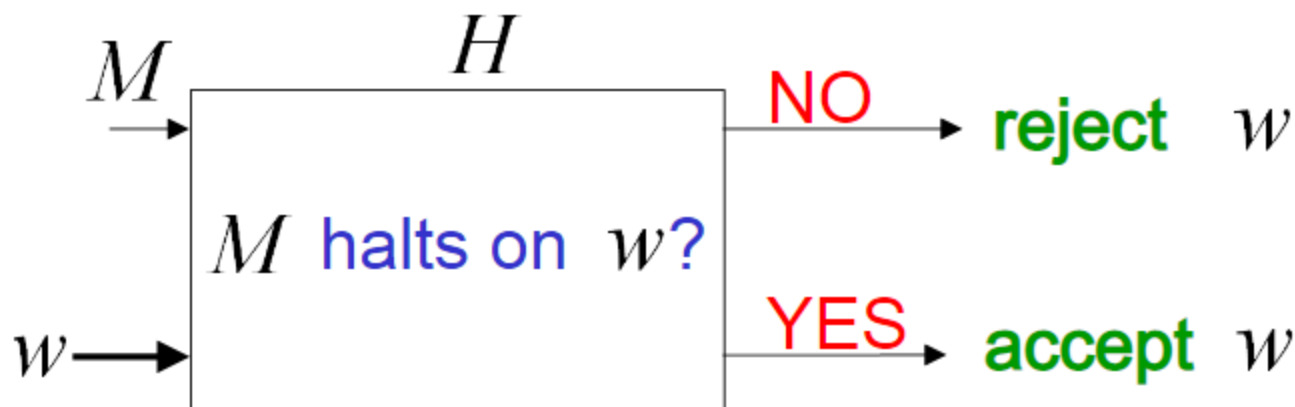
- We show that A_{TM} is reducible to $HALT_{TM}$
- Since A_{TM} is undecidable, so is $HALT_{TM}$

PROVING UNDECIDABILITY VIA REDUCTIONS

THEOREM 5.1

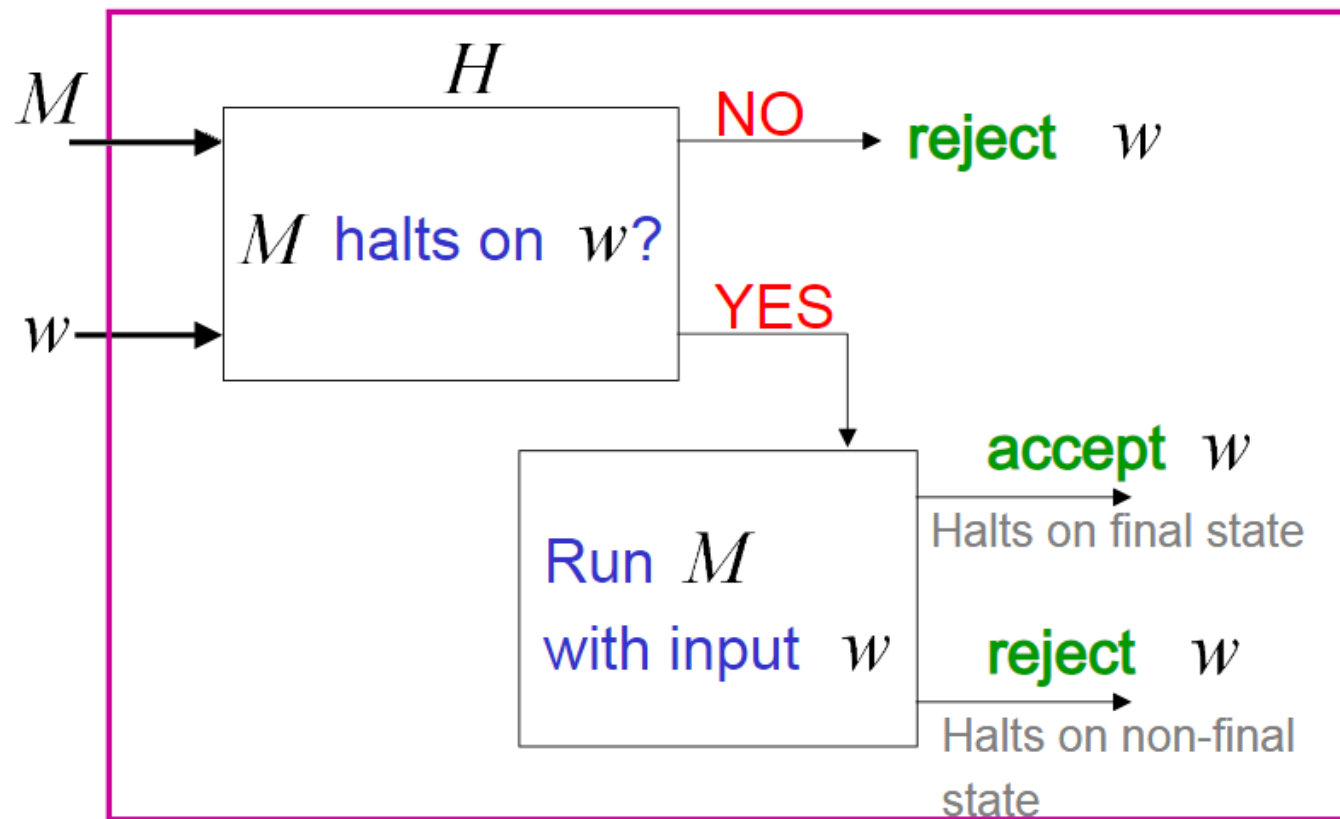
$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$ is undecidable.

- Suppose $HALT_{TM}$ is decidable, this means

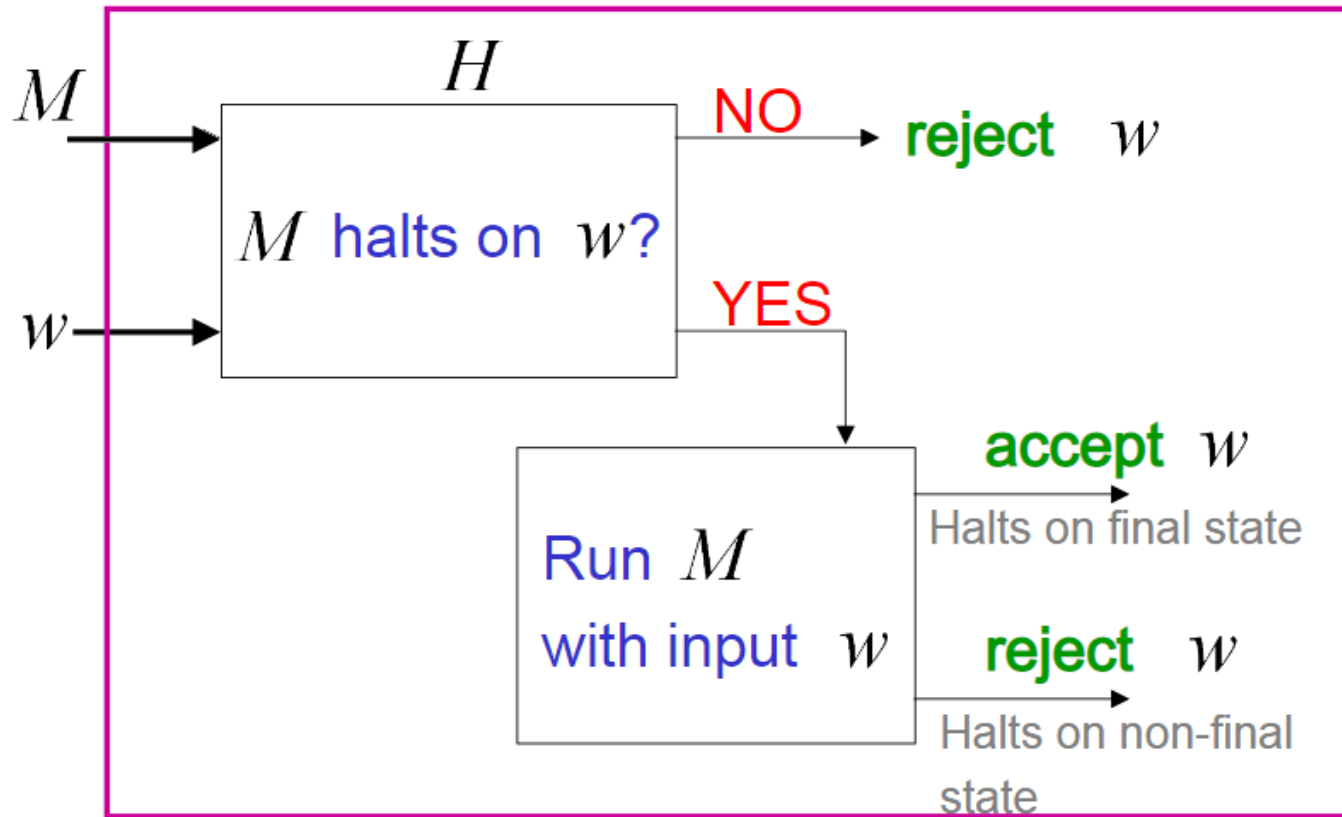


- Then A_{TM} is decidable.

- Then A_{TM} is decidable.



- Then A_{TM} is decidable.



- **Contradiction**

This diagram shows how A_{TM} can be reduced to $HALT_{TM}$

PROVING UNDECIDABILITY VIA REDUCTIONS

THEOREM 5.2

$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Phi\}$ is undecidable.

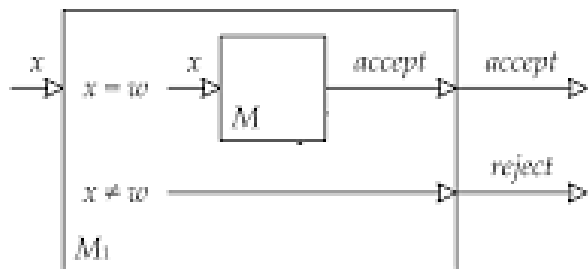
- A decider for A_{TM} via E_{TM} is possible.
- We are given $\langle M, w \rangle$, and asked to find whether $\langle M, w \rangle \in A_{TM}$?
- For this, First create M_1

PROVING UNDECIDABILITY VIA REDUCTIONS

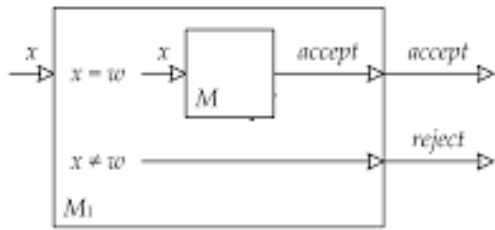
THEOREM 5.2

$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Phi\}$ is undecidable.

- A decider for A_{TM} via E_{TM} is possible.
- We are given $\langle M, w \rangle$, and asked to find whether $\langle M, w \rangle \in A_{TM}$?
- For this, First create M_1



Now, $L(M_1)$ is what?



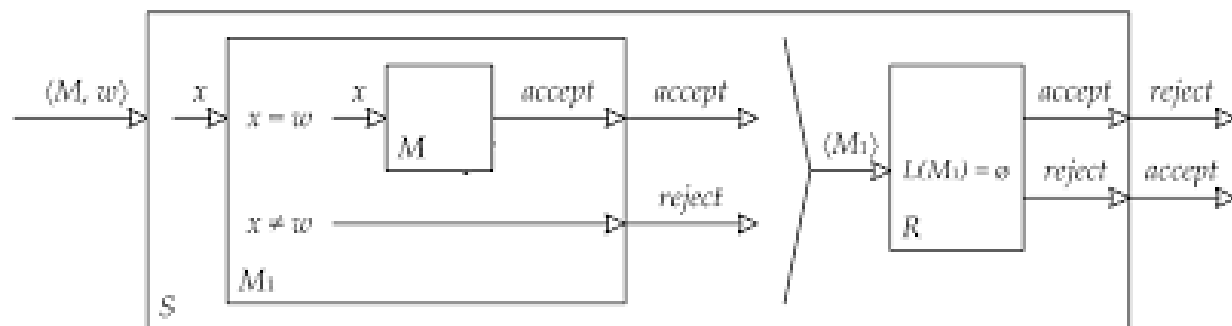
- Now, $L(M_1)$ is what?
- $L(M_1)$ is either $\{w\}$ or is ϕ
- Now, if M_1 is in E_{TM}
 - This means, $L(M_1) = \phi$
 - This means, $\langle M, w \rangle \notin A_{TM}$
- Now, if M_1 is not in E_{TM}
 - This means, $L(M_1) \neq \phi$
 - This means, $\langle M, w \rangle \in A_{TM}$

PROVING UNDECIDABILITY VIA REDUCTIONS

THEOREM 5.2

$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Phi\}$ is undecidable.

- A decider for A_{TM} via E_{TM} is possible.



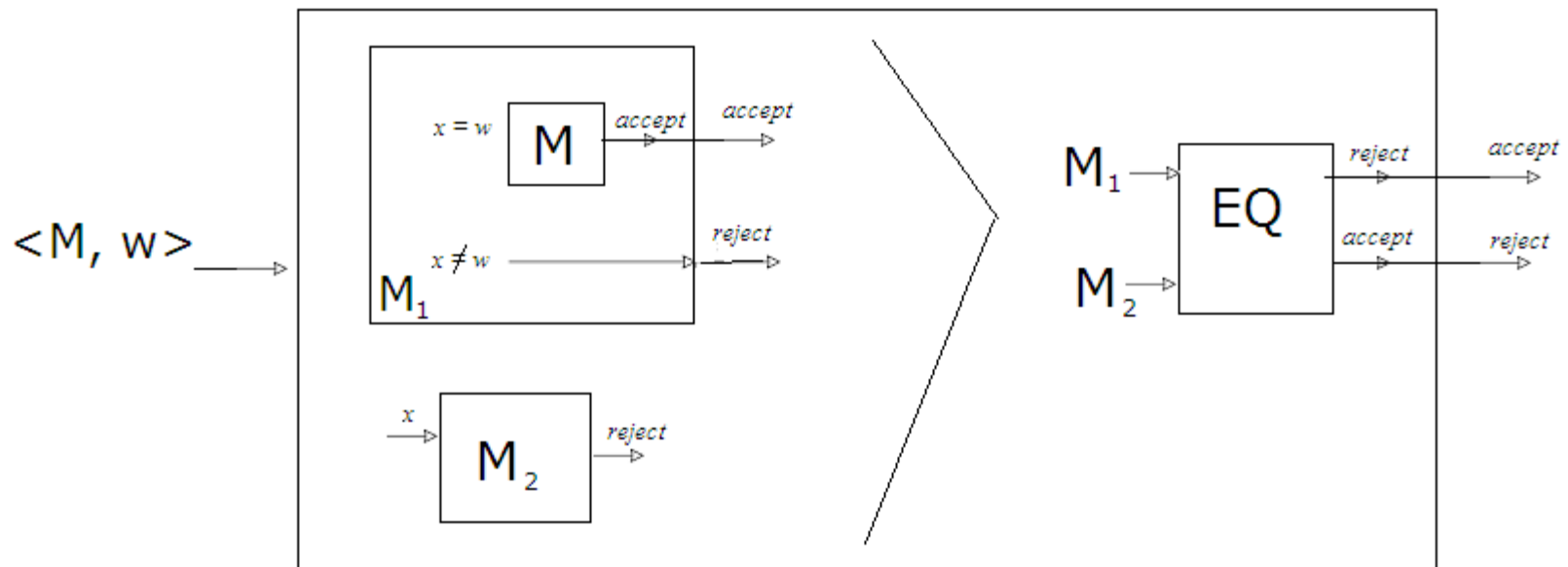
- That is, A_{TM} can be reduced to E_{TM} .
- Contradiction

TESTING FOR LANGUAGE EQUALITY

THEOREM 5.4

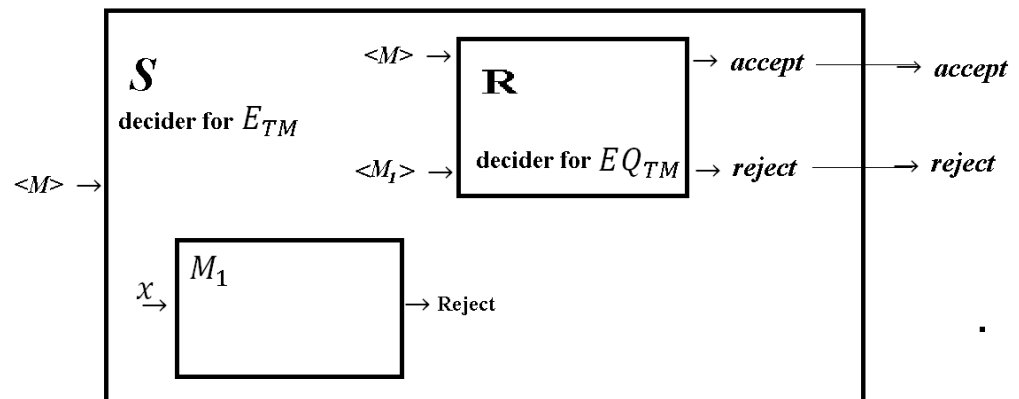
$EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$ is undecidable.

- Decider for A_{TM} via EQ_{TM}
- Following is the reduction of A_{TM} to EQ_{TM}



Alternate way to show EQ_{TM} is undecidable.

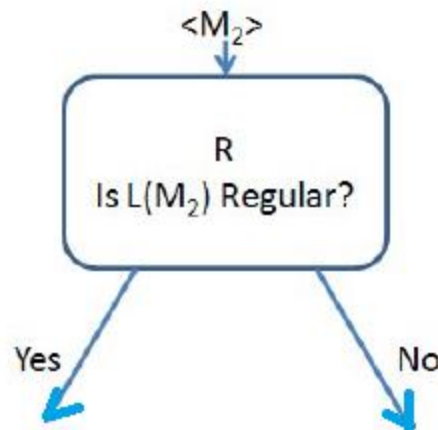
- Since, we know E_{TM} is undecidable,
- We can try to reduce E_{TM} to EQ_{TM}
- Let R be a decider for EQ_{TM}
- We can build a decider (call this S) for E_{TM} by using R



TESTING FOR REGULARITY

$REGULAR_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language} \}$ is undecidable.

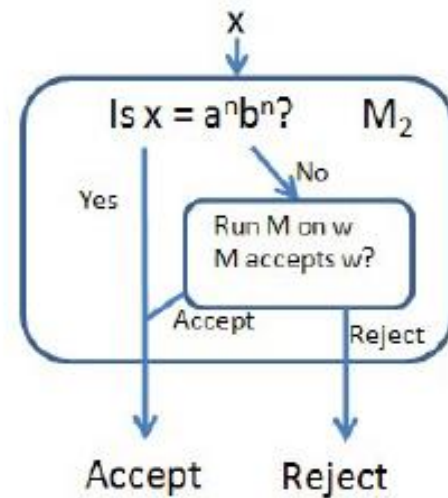
- If $REGULAR_{TM}$ is decidable, then this can be used to decide A_{TM} .
- Let R be a decider for $REGULAR_{TM}$.



How R can be used to decide

$$\langle M, w \rangle \in A_{TM} \quad ?$$

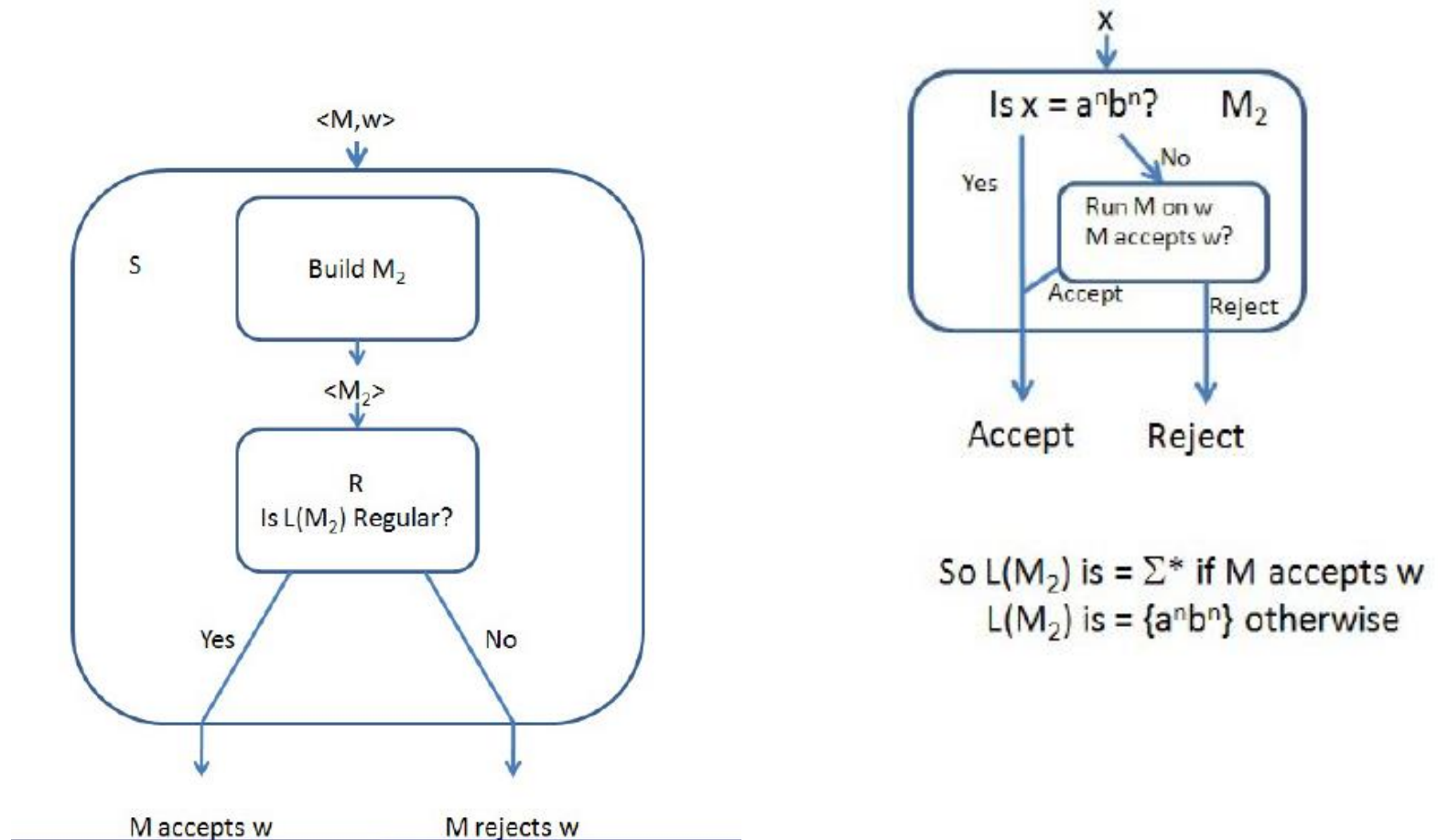
- Build M_2 as shown



So $L(M_2)$ is $= \Sigma^*$ if M accepts w
 $L(M_2)$ is $= \{a^n b^n\}$ otherwise

How R can be used to decide $\langle M, w \rangle \in A_{TM}$?

- Build M_2 as shown



- Similar to $REGULAR_{TM}$, we can show CFL_{TM} is undecidable.
 - That is, finding whether a TM's language is CFL or not is undecidable.
 - In fact, we can extend this. TM's language is finite or not is undecidable.
 - General theorem in this regard is called ***The Rice's Theorem.***

More formal way of reductions

MAPPING REDUCTION

Computable function

- **DEFINITION 5.17**

A function $f: \Sigma^* \rightarrow \Sigma^*$ is a *computable function* if some Turing machine M , on every input w , halts with just $f(w)$ on its tape.

- For example,

we can make a machine that takes input $\langle m, n \rangle$ and returns $m + n$, the sum of m and n .

Mapping Reductions

Definition: Let A and B be two languages. We say that there is a **mapping reduction** from A to B , and denote

$$A \leq_m B$$

if there is a **computable function**

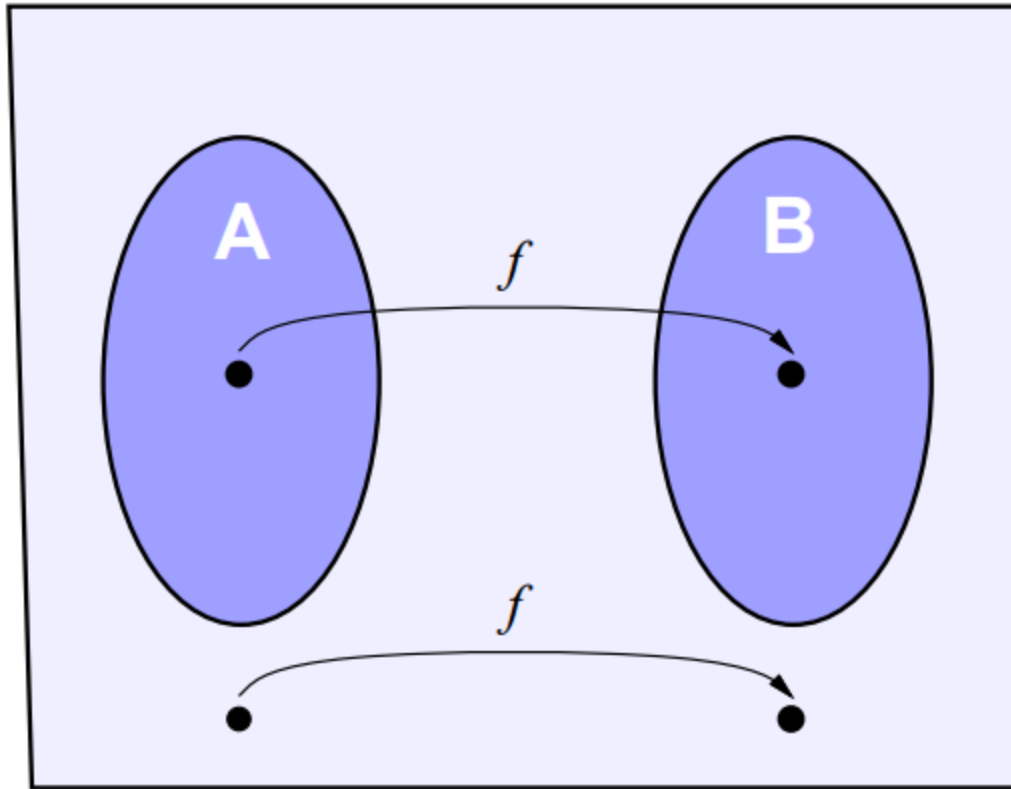
$$f : \Sigma^* \longrightarrow \Sigma^*$$

such that, for every w ,

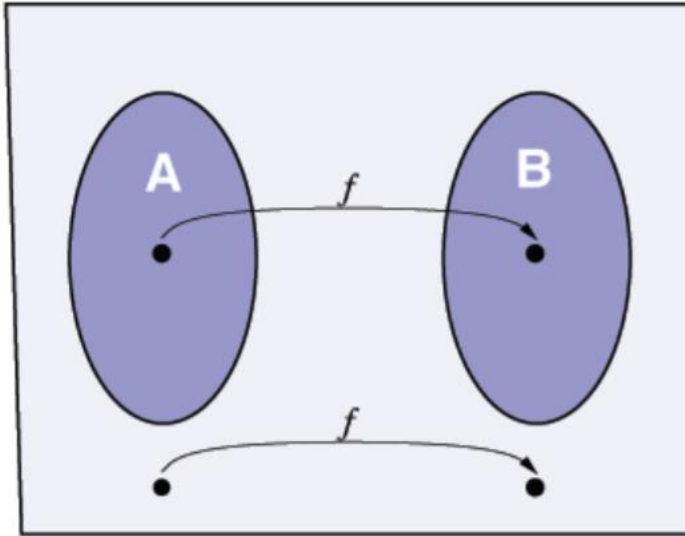
$$w \in A \iff f(w) \in B.$$

The function f is called the **reduction** from A to B .

Mapping Reductions



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



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

Notice that $A \leq_m B$ implies $\overline{A} \leq_m \overline{B}$.

Mapping Reductions

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.

Mapping Reductions

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

In fact, this has been our principal tool for proving undecidability of languages other than A_{TM} .

Example: Halting

Recall that

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

$$H_{\text{TM}} = \{ \langle M, w \rangle \mid \text{TM } M \text{ halts on input } w \}$$

Earlier we proved that

- H_{TM} undecidable
- by (de facto) reduction from A_{TM} .

Let's reformulate this.

Example: Halting

Define a **computable function**, f :

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

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

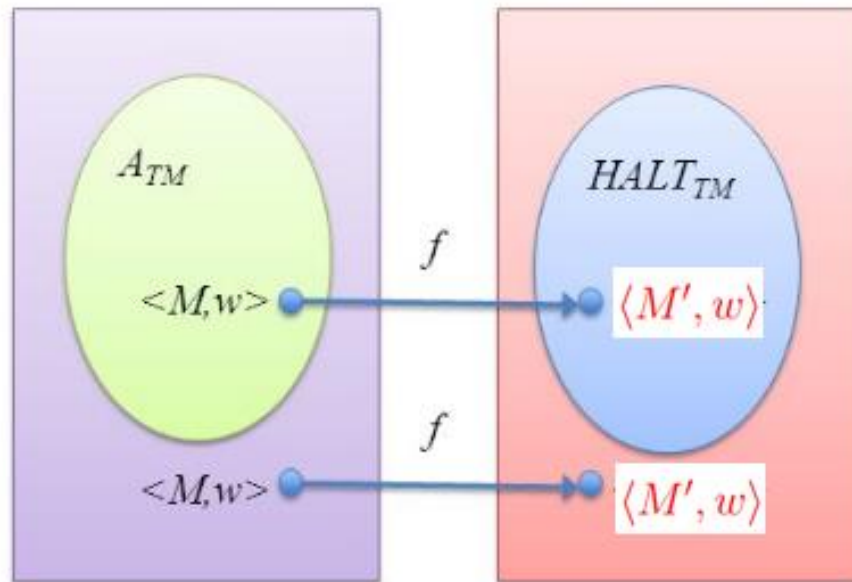
- run M on x
- If M accepts, *accept*.
- if M rejects, **enter a loop**.

- output $\langle M', w \rangle$

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

$$\leq_m$$

$$HALT_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM \& } M \text{ halts on input } w \}$$



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

- $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts } w \}$
- $\overline{E_{TM}} = \{ \langle M \rangle \mid L(M) \neq \phi \}$
- $f: \Sigma^* \rightarrow \Sigma^*$ can be defined as

Create M' : On input x ,

if $x \neq w$, output “Reject”;

if $x = w$, output the result of running w on M .



Mapping Reductions: Reminders

Theorem 1:

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

Theorem2 :

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

Mapping Reductions: Corollaries

Corollary 1: If $A \leq_m B$ and A is undecidable, then B is undecidable.

Corollary 2: If $A \leq_m B$ and A is not in \mathcal{RE} , then B is not in \mathcal{RE} .

Corollary 3: If $A \leq_m B$ and A is not in $co\mathcal{RE}$, then B is not in $co\mathcal{RE}$.

TM Equality

Theorem: Both EQ_{TM} and its complement, $\overline{EQ_{TM}}$, are not enumerable. Stated differently, EQ_{TM} is neither enumerable nor co-enumerable.

- We show that A_{TM} is reducible to EQ_{TM} . The **same function** is also a mapping reduction from $\overline{A_{TM}}$ to $\overline{EQ_{TM}}$, and thus $\overline{EQ_{TM}}$ is **not enumerable**.
- We then show that A_{TM} is reducible to $\overline{EQ_{TM}}$. The **new function** is also a mapping reduction from $\overline{A_{TM}}$ to EQ_{TM} , and thus EQ_{TM} is **not enumerable**.

$$A_{TM} \leq_m EQ_{TM}$$

Proof: The following TM computes the reduction:

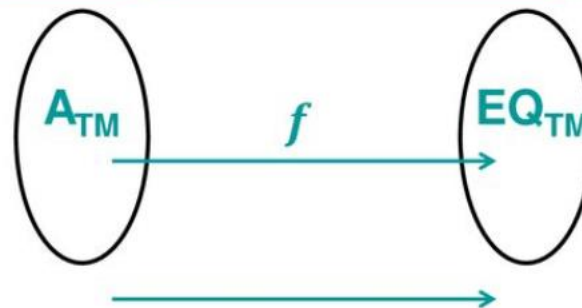
F = `` On input $\langle M, w \rangle$, where M is a TM and w is a string:

1. Construct TMs M', M'' .

$M' =$ `` On input x ,
 1. Ignore the input.
 2. Run TM M on input w .
 3. If it accepts, **accept**.’’

$M'' =$ `` **Accept**.’’

2. **Output** $\langle M', M'' \rangle$.’’



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

Proof: We give a mapping reduction $A_{TM} \leq_m \overline{EQ_{TM}}$

The following TM computes the reduction:

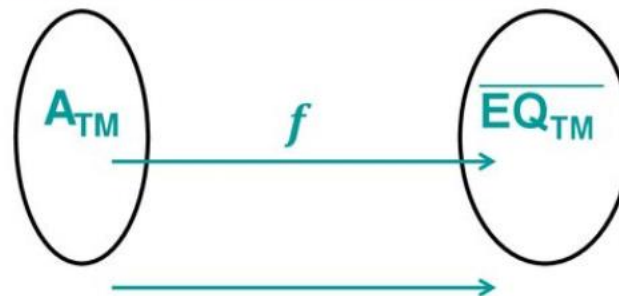
F = `` On input $\langle M, w \rangle$, where M is a TM and w is a string:

1. Construct TMs M', M'' .

$M' =$ `` On input x ,
 1. Ignore the input.
 2. Run TM M on input w .
 3. If it accepts, **accept.**''

$M'' =$ `` **Reject.**''

2. **Output** $\langle M', M'' \rangle$."



Language Hierarchy (revisited)

Set of Languages (= set of "set of strings")

Set of Decidable Language

Set of Recognizable Language

$\{0^n 1^n 2^n\}$

$\{ww\}$

EQ_{TM}

$\overline{EQ_{TM}}$

A_{TM}

$\overline{A_{TM}}$

$\{0^n 1^n\}$

$\{w \mid w = w^R\}$

Set of Context-Free Language

Set of Regular Language

$\{0^x 1^y\}$

$\{w \text{ with even } |w|\}$

Non Trivial Properties of \mathcal{RE} Languages

A few examples

- L is finite.
- L is infinite.
- L contains the empty string.
- L contains no prime number.
- L is co-finite.
- . . .

All these are **non-trivial** properties of enumerable languages, since for each of them there is $L_1, L_2 \in \mathcal{RE}$ such that L_1 satisfies the property but L_2 does not.

Are there any **trivial** properties of \mathcal{RE} languages?

Rice's Theorem

Theorem Let \mathcal{C} be a proper non-empty subset of the set of enumerable languages. Denote by $L_{\mathcal{C}}$ the set of all TMs encodings, $\langle M \rangle$, such that $L(M)$ is in \mathcal{C} . Then $L_{\mathcal{C}}$ is undecidable.

(See problem 5.22 in Sipser's book)

Proof by reduction from A_{TM} .

Given M and w , we will construct M_0 such that:

- If M accepts w , then $\langle M_0 \rangle \in L_{\mathcal{C}}$.
- If M does not accept w , then $\langle M_0 \rangle \notin L_{\mathcal{C}}$.

<http://www.cs.tau.ac.il/~bchor/CM09/Compute9.pdf>

Has (towards end) some good slides on Rice's theorem and its consequences.

POST CORRESPONDENCE PROBLEM

- Undecidability is not just confined to problems concerning automata and languages.
- There are other “natural” problems which can be proved undecidable.
- The **Post correspondence problem** (PCP) is a tiling problem over strings.
- A tile or a domino contains two strings, t and b ; e.g., $\begin{bmatrix} ca \\ a \end{bmatrix}$.
- Suppose we have dominos

$$\left\{ \begin{bmatrix} b \\ ca \end{bmatrix}, \begin{bmatrix} a \\ ab \end{bmatrix}, \begin{bmatrix} ca \\ a \end{bmatrix}, \begin{bmatrix} abc \\ c \end{bmatrix} \right\}$$

- A **match** is a list of these dominos so that when concatenated the top and the bottom strings are identical. For example,

$$\begin{bmatrix} a \\ ab \end{bmatrix} \begin{bmatrix} b \\ ca \end{bmatrix} \begin{bmatrix} ca \\ a \end{bmatrix} \begin{bmatrix} a \\ ab \end{bmatrix} \begin{bmatrix} abc \\ c \end{bmatrix} = \frac{abcaaabc}{abcaaabc}$$

- The set of dominos $\left\{ \left[\frac{abc}{ab} \right], \left[\frac{ca}{a} \right], \left[\frac{acc}{ba} \right], \right\}$ does not have a solution.

POST CORRESPONDENCE PROBLEM

AN INSTANCE OF THE PCP

A PCP instance over Σ is a finite collection P of dominos

$$P = \left\{ \left[\frac{t_1}{b_1} \right], \left[\frac{t_2}{b_2} \right], \dots, \left[\frac{t_k}{b_k} \right] \right\}$$

where for all i , $1 \leq i \leq k$, $t_i, b_i \in \Sigma^*$.

MATCH

Given a PCP instance P , a **match** is a nonempty sequence

$$i_1, i_2, \dots, i_\ell$$

of numbers from $\{1, 2, \dots, k\}$ (with repetition) such that

$$t_{i_1} t_{i_2} \cdots t_{i_\ell} = b_{i_1} b_{i_2} \cdots b_{i_\ell}$$

POST CORRESPONDENCE PROBLEM

AN INSTANCE OF THE PCP

A PCP instance over Σ is a finite collection P of dominos

$$P = \left\{ \left[\frac{t_1}{b_1} \right], \left[\frac{t_2}{b_2} \right], \dots, \left[\frac{t_k}{b_k} \right] \right\}$$

where for all i , $1 \leq i \leq k$, $t_i, b_i \in \Sigma^*$.

MATCH

Given a PCP instance P , a **match** is a nonempty sequence

$$i_1, i_2, \dots, i_\ell$$

of numbers from $\{1, 2, \dots, k\}$ (with repetition) such that
 $t_{i_1} t_{i_2} \dots t_{i_\ell} = b_{i_1} b_{i_2} \dots b_{i_\ell}$

QUESTION:

Does a given PCP instance P have a match?

POST CORRESPONDENCE PROBLEM

QUESTION:

Does a given PCP instance P have a match?

LANGUAGE FORMULATION:

$PCP = \{\langle P \rangle \mid P \text{ is a PCP instance and it has a match}\}$

THEOREM 5.15

PCP is undecidable.

POST CORRESPONDENCE PROBLEM

QUESTION:

Does a given PCP instance P have a match?

LANGUAGE FORMULATION:

$PCP = \{\langle P \rangle \mid P \text{ is a PCP instance and it has a match}\}$

THEOREM 5.15

PCP is undecidable.

Proof: By reduction using computation histories. If PCP is decidable then so is A_{TM} . That is, if PCP has a match, then M accepts w .

- That is, A_{TM} can be reduced to PCP .
- This reduction is via a simplified PCP called $MPCP$ (modified PCP).
- Several undecidable properties of CFGs are obtained by reducing them (i.e., the corresponding languages) from PCP.

