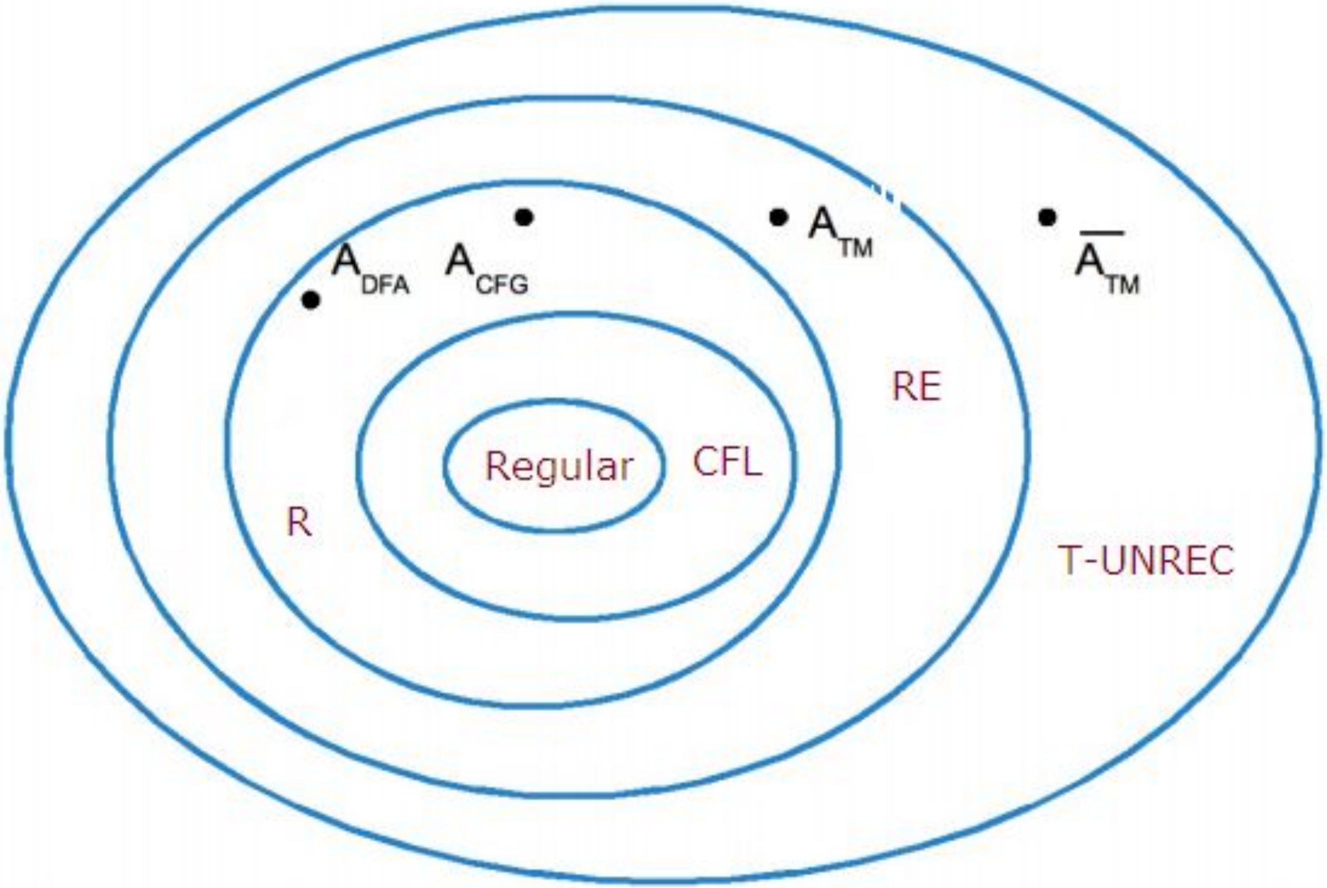
Reducibility

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

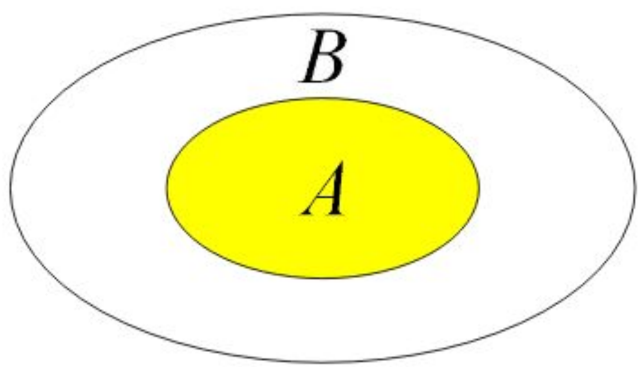


Reducibility

- 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 : One more example

- $A \leq B$
- Doing injection to a patient $\leq$ doing surgery to a patient
- 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

$$\Downarrow$$

If B is decidable then A is decidable.

$$\Downarrow$$

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

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

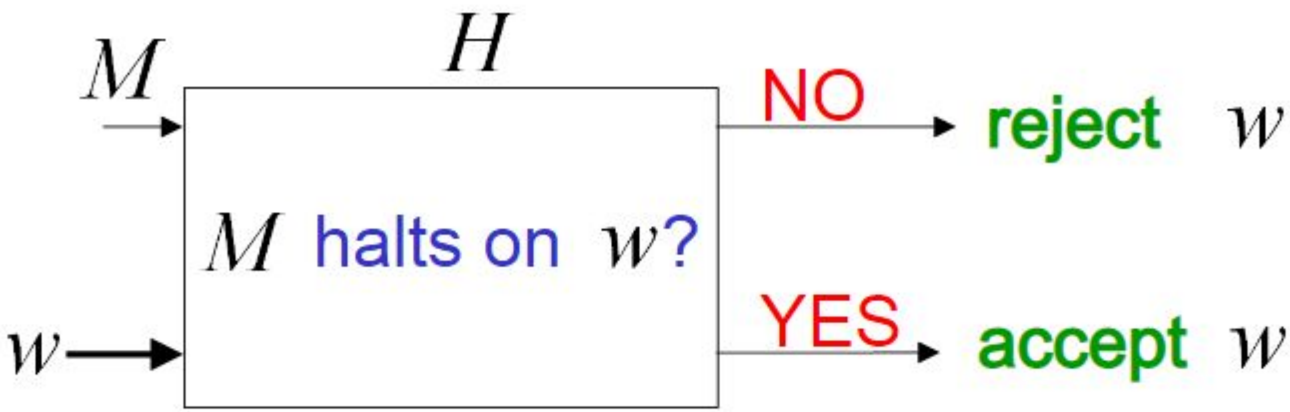
- We show that A_{TM} is reducible to $HALT_{TM}$
- Since A_{TM} is undecidable, so is $HALT_{TM}$

Proving Undecidability via Reductions

Theorem 5.1

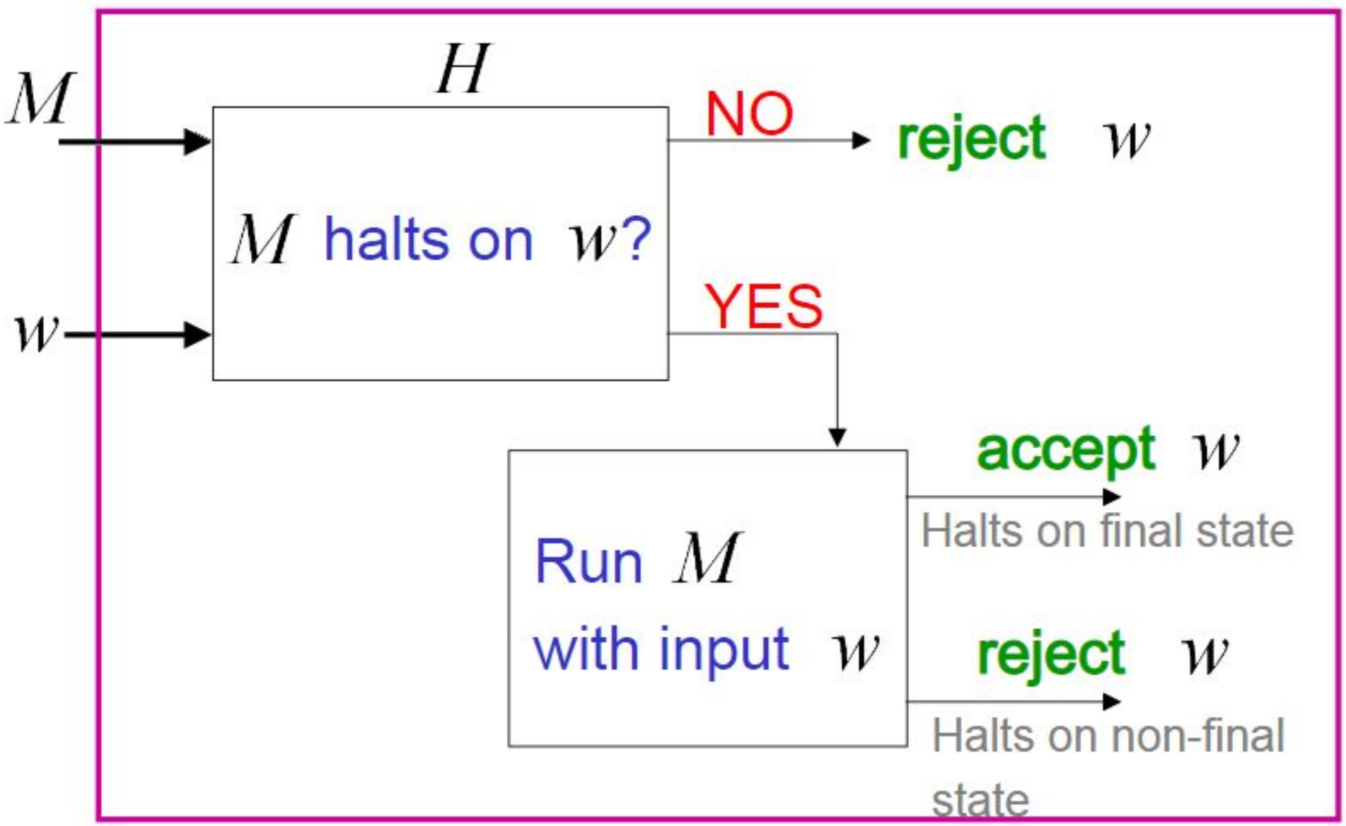
$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$ is undecidable.

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

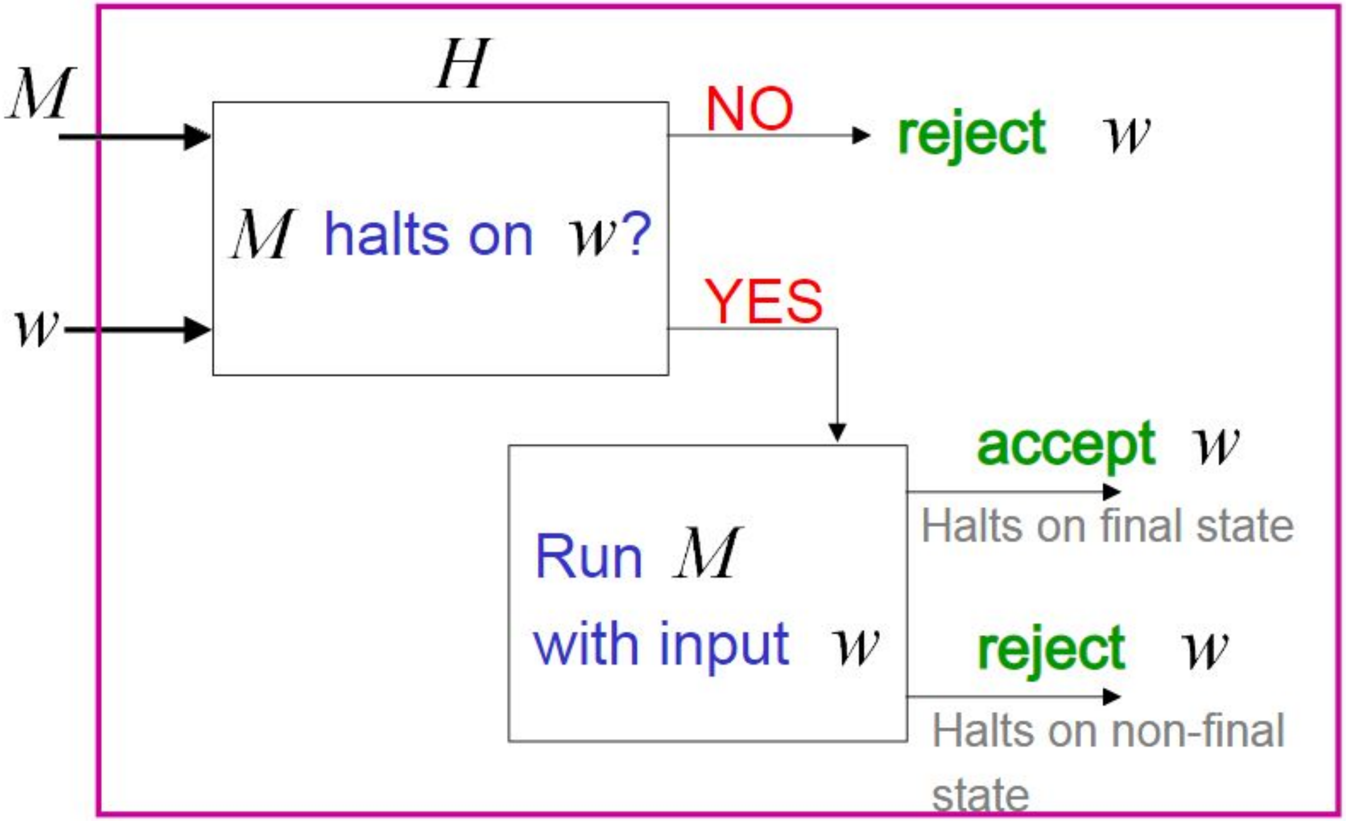


- Then A_{TM} is decidable.

- 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 $< M, w >$, and asked to find whether $< M, w > \in A_{TM}$?
- For this, First create M_1 (from the given $< M, w >$)

-

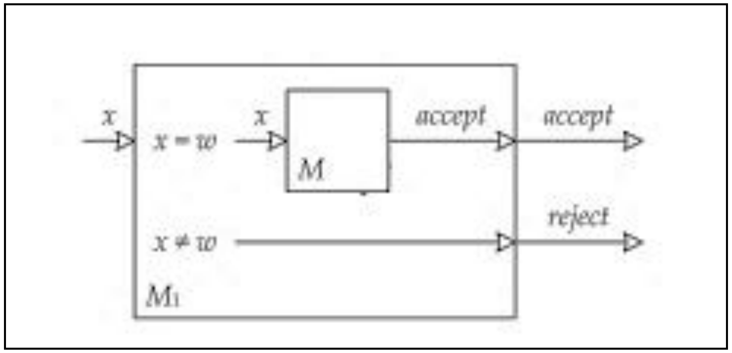
- A decider for A_{TM} via E_{TM} is possible.
- We are given $< M, w >$, and asked to find whether $< M, w > \in A_{TM}$?
- For this, First create M_1 (from the given $< M, w >$)

Proving Undecidability via Reductions

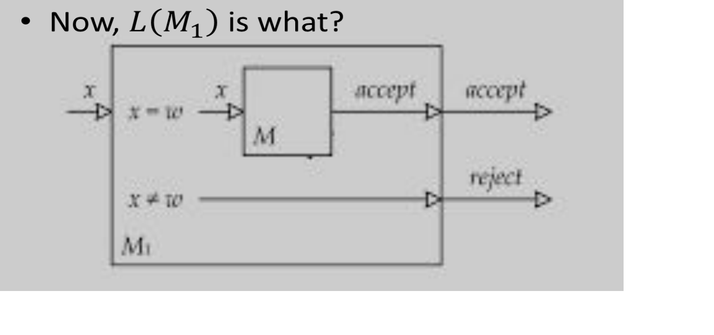
Theorem 5.2

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

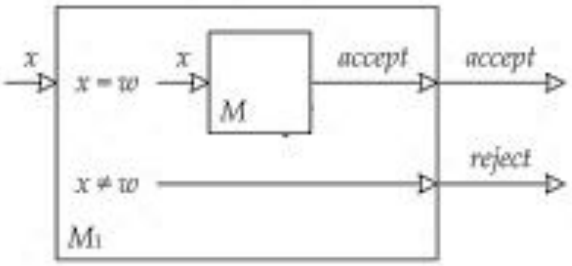
- A decider for A_{TM} via E_{TM} is possible.
- We are given $< M, w >$, and asked to find whether $< M, w > \in A_{TM}$?
- For this, First create M_1



Now, $L(M_1)$ is what?



- Now, $L(M_1)$ is what?



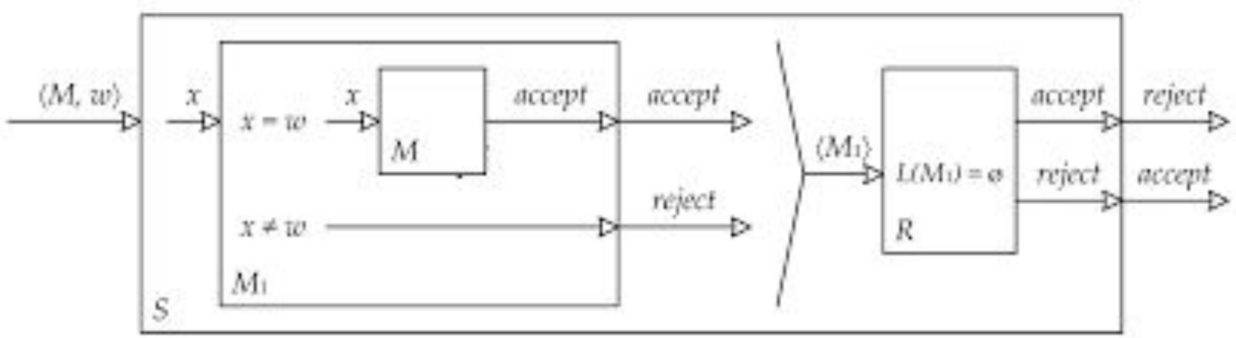
- Now, $L(M_1)$ is what?
- $L(M_1)$ is either $\{w\}$ or is ϕ
- Now, if M_1 is in E_{TM}
  - This means, $L(M_1) = \phi$
  - This means, $< M, w > \notin A_{TM}$
- Now, if M_1 is not in E_{TM}
  - This means, $L(M_1) \neq \ \phi$
  - This means, $< M, w > \in A_{TM}$

Proving Undecidability via Reductions

Theorem 5.2

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

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



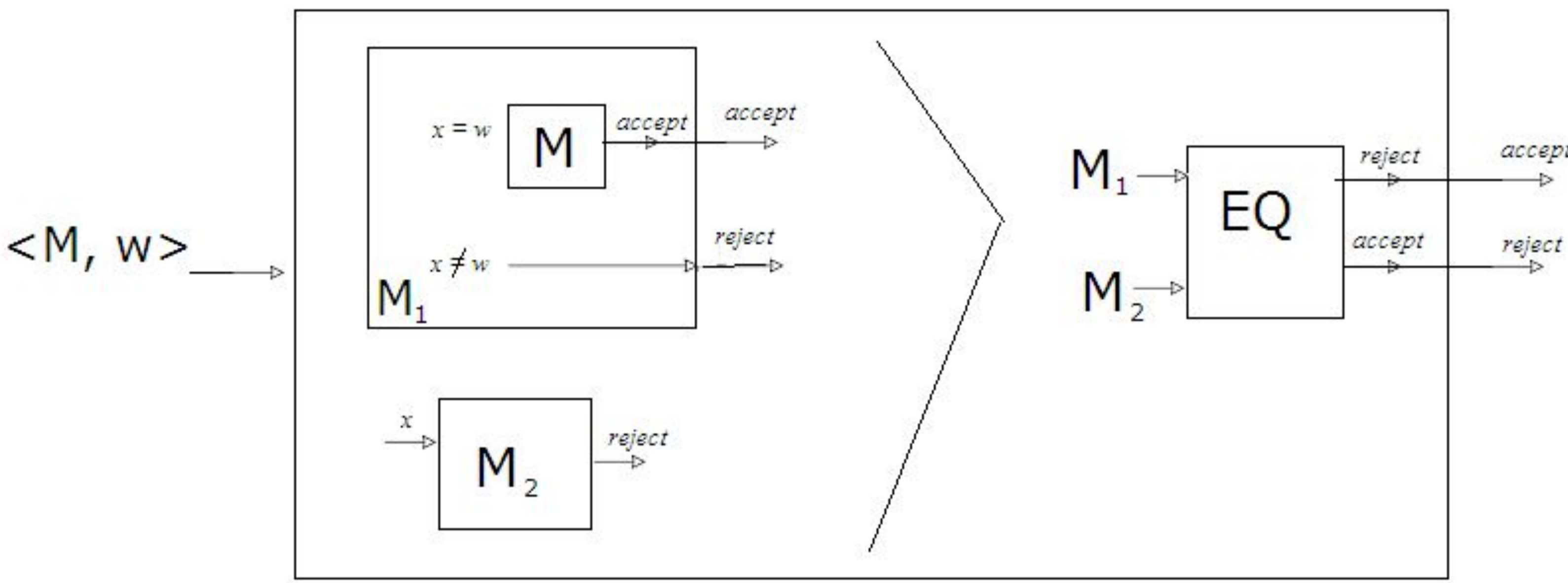
- That is, A_{TM} can be reduced to E_{TM}.
- Contradiction

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.

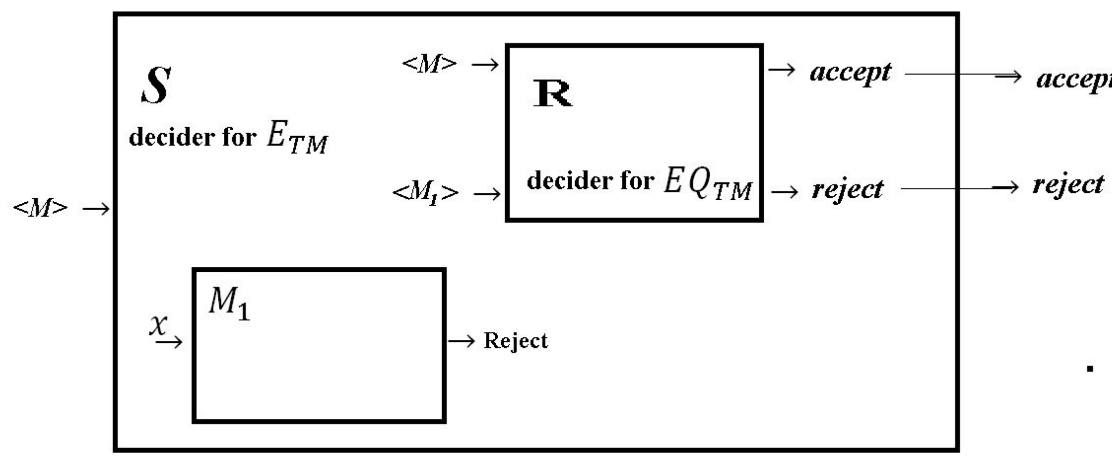
- Decider for A_{TM} via EQ_{TM}
- Following is the reduction of A_{TM} to EQ_{TM}



This is a decider for A_{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

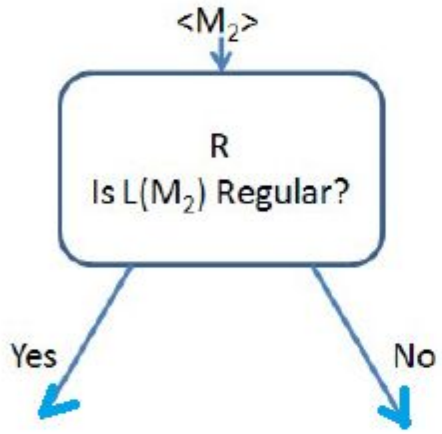


***S*: Decider for E_{TM}**

Testing for Regularity

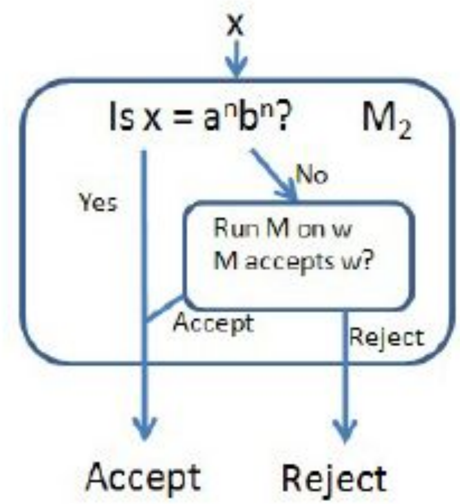
$REGULAR_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$ is undecidable.

- 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 $<M,w> \in A_{TM}$?

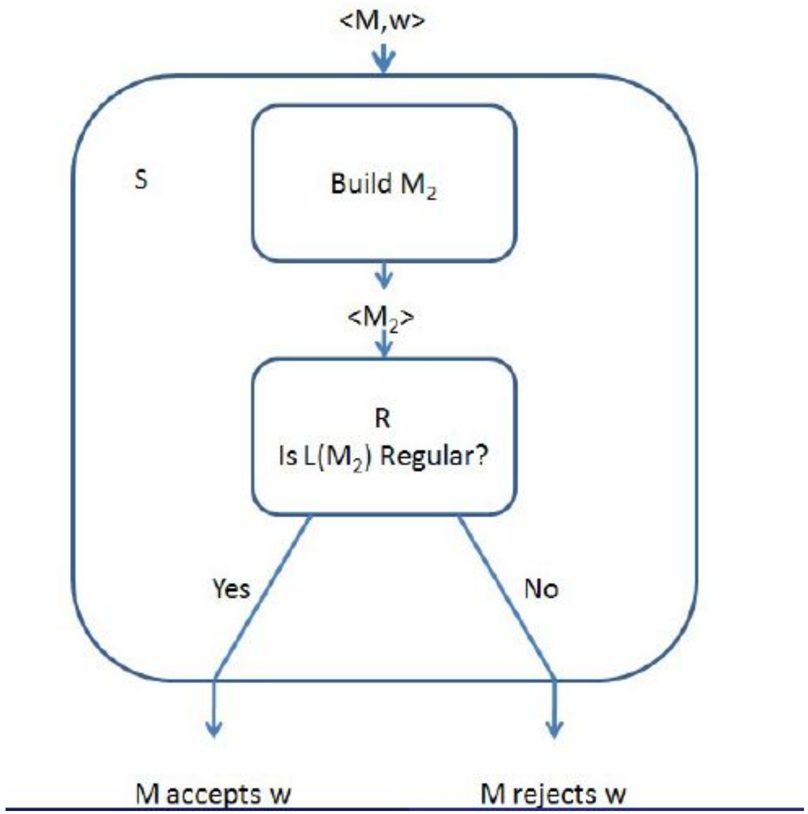
- Build M_2 as shown

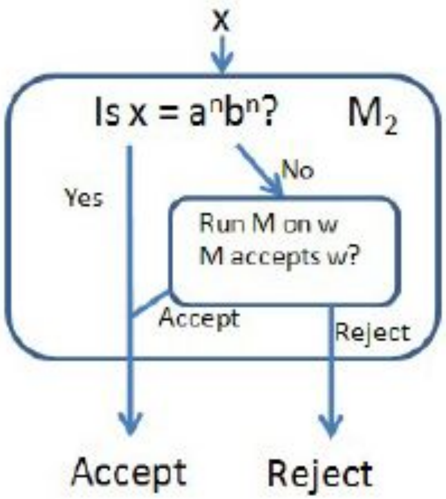


So $L(M_2)$ is = Σ^* if M accepts w
$L(M_2)$ is = $\{a^nb^n\}$ otherwise

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

- Build M_2 as shown





So $L(M_2)$ is $= \Sigma^*$ if M accepts w

$L(M_2)$ is $= \{a^n b^n\}$ otherwise

- 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

WE CAN GET MORE REFINED ANSWERS

Computable function

- **DEFINITION 5.17**

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

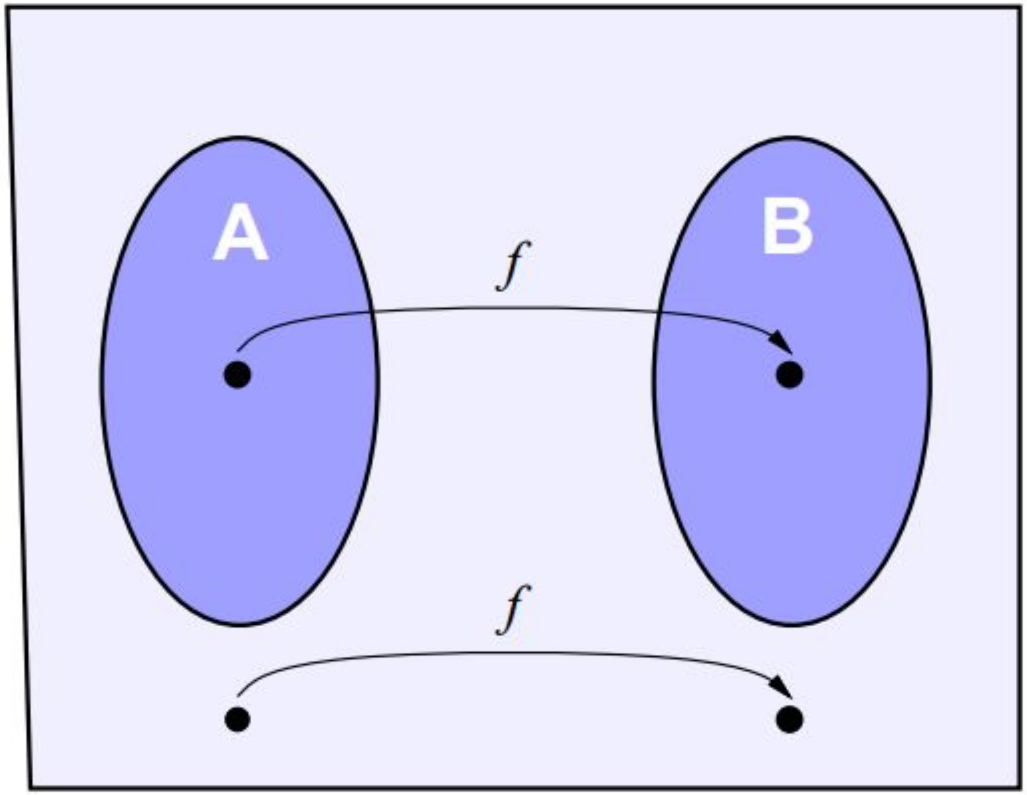
$$f : \Sigma^* \longrightarrow \Sigma^*$$

such that, for every w,

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

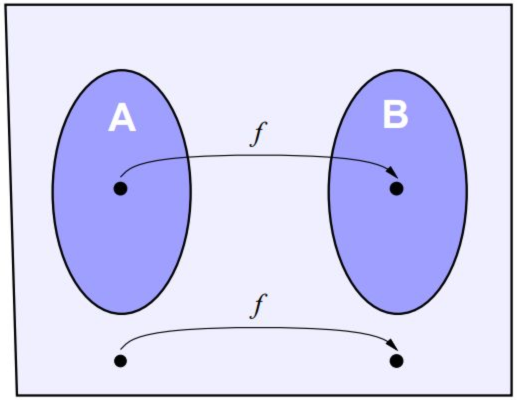
The function f is called the reduction from A to B.

Mapping Reductions



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

Mapping Reductions

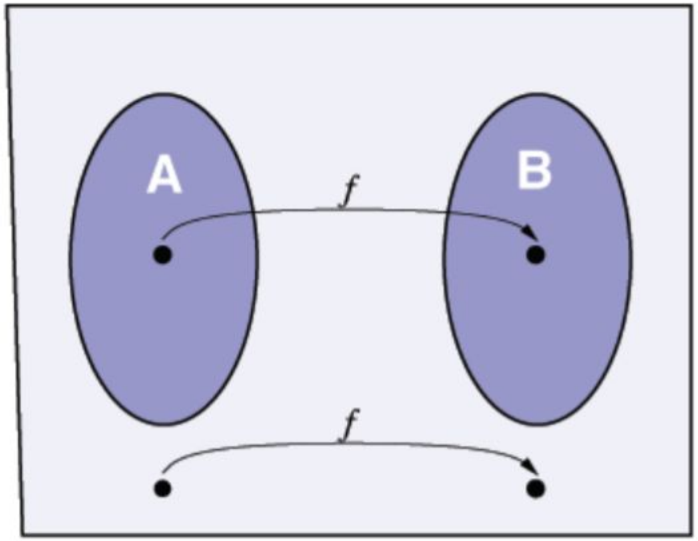


A mapping reduction converts questions about membership in A to membership in B

An Example:

Let A is 0* , and B is {0, 1}
Let the f is defined as below.

If $w \in 0^*$ and $|w|$ $is\ even\ then\ f(w) = 0,$
Else if $w \in 0^*$ and $|w|$ $is\ odd\ then\ f(w) = 1,$
Else if $w \notin 0^*$ $then\ f(w) = 11.$



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

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

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

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

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

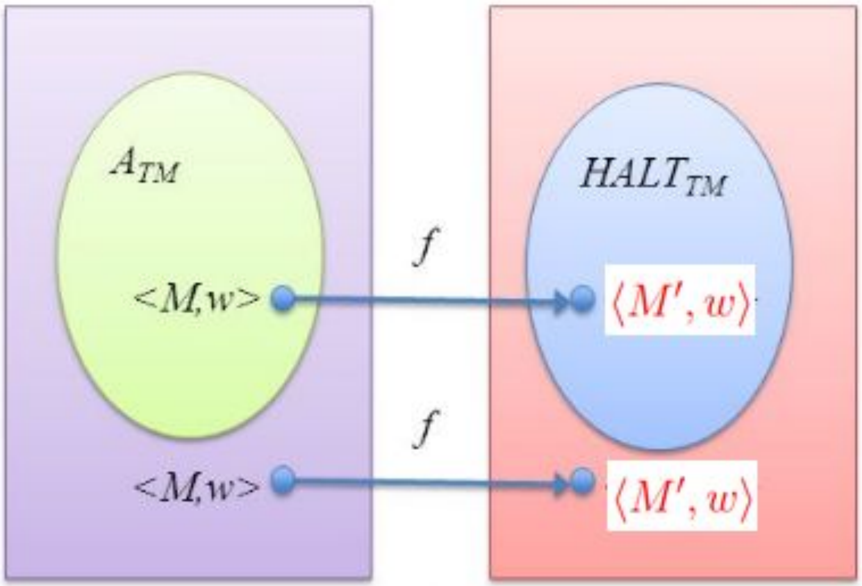
The following machine computes this function f.
$F =$ on input $\langle M, w \rangle$:

- 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} = {$<M,w>$ | M is a TM and M accepts w}

$\leq_m$

$HALT_{TM}$ = {$<M,w>$ | M is a TM & M halts on input w}



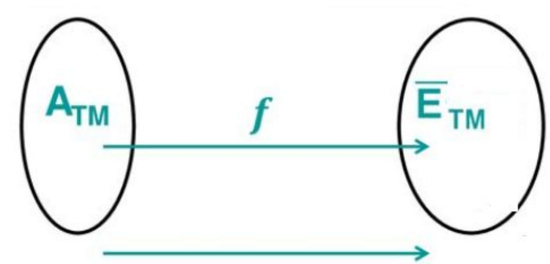
$A_{TM} \leq_m \overline{E_{TM}}$

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

Create M' : On input x,

if $x \neq w$, output "Reject";

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



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 $\mathcal{RE}$.

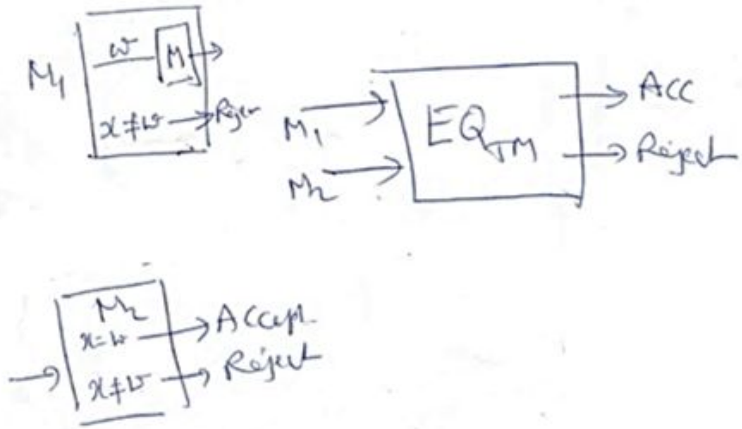
Corollary 3: If $A \leq_m B$ and A is not in $co\mathcal{RE}$, then B is not in $co\mathcal{RE}$.

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.

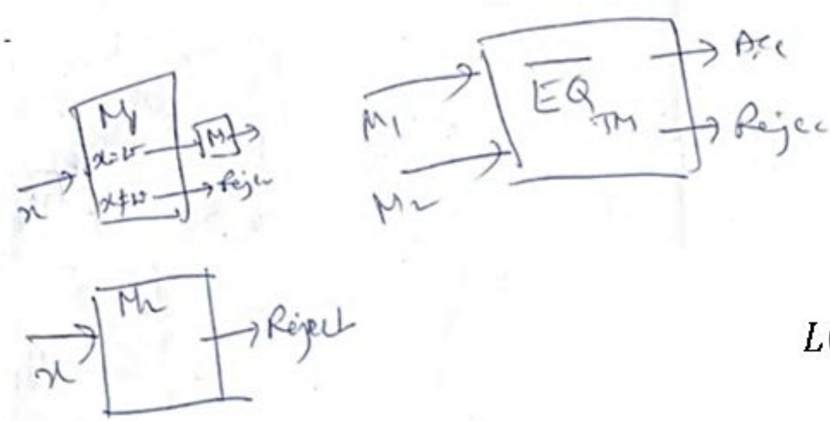
$A_{TM} \leq_m EQ_{TM}$



$$L(M_1) = \begin{cases} \{w\} & if\ M\ accepts\ w \\ \phi, & Otherwie \end{cases}$$

$$L(M_2) = \{w\}$$

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



$$L(M_1) = \begin{cases} \{w\} & if\ M\ accepts\ w \\ \phi, & Otherwie \end{cases}$$

$$L(M_2) = \phi$$

Alternate solutions found in the net.

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

Proof: The following TM computes the reduction:

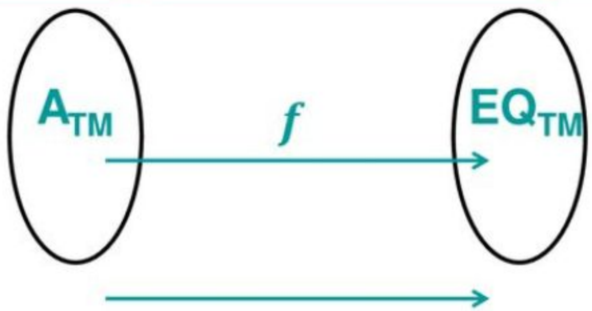
F = `` On input $\langle M, w\rangle$, where M is a TM and w is a string:

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 $<M', M''>$."



$$L(M') = \begin{cases} \Sigma^*, if\ M\ accepts\ w \\ \phi, \quad Otherwise \end{cases}$$

$$L(M'') = \Sigma^*$$

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

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

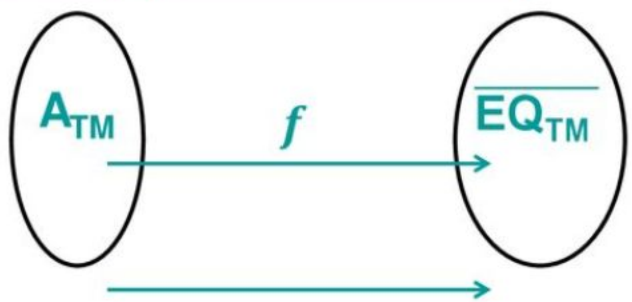
The following TM computes the reduction:

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

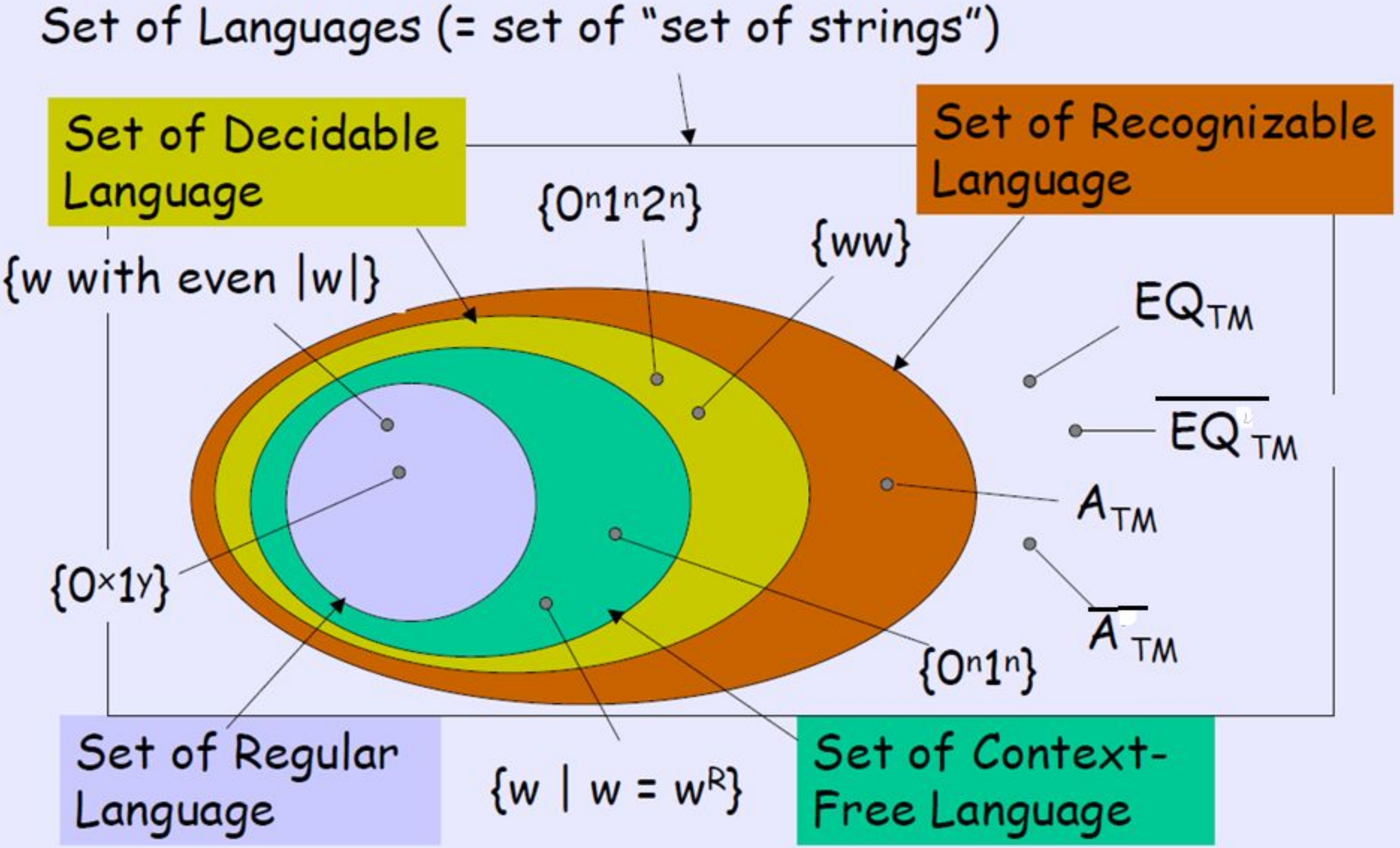
2. Output $<M', M''>$.”



$$L(M') = \begin{cases} \Sigma^*, if\ M\ accepts\ w \\ \phi, \quad Otherwise \end{cases}$$

$$L(M'') = \phi$$

Language Hierarchy (revisited)



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