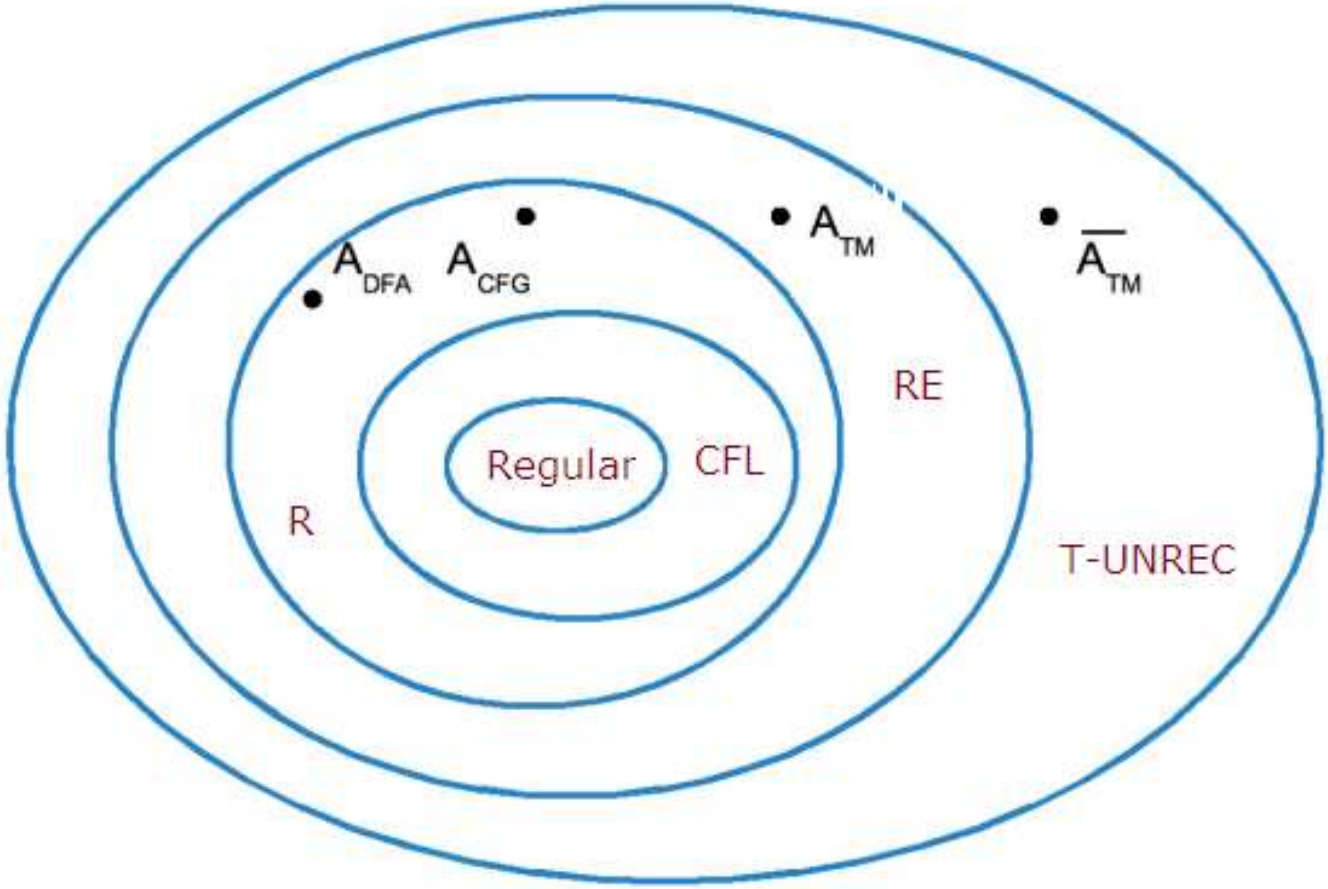
Reducibility

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

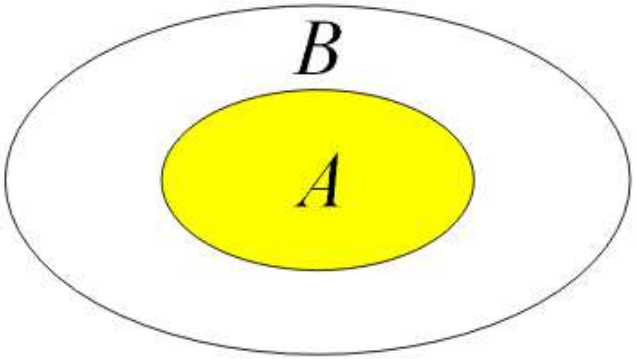


REDUCIBILITY

- 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.

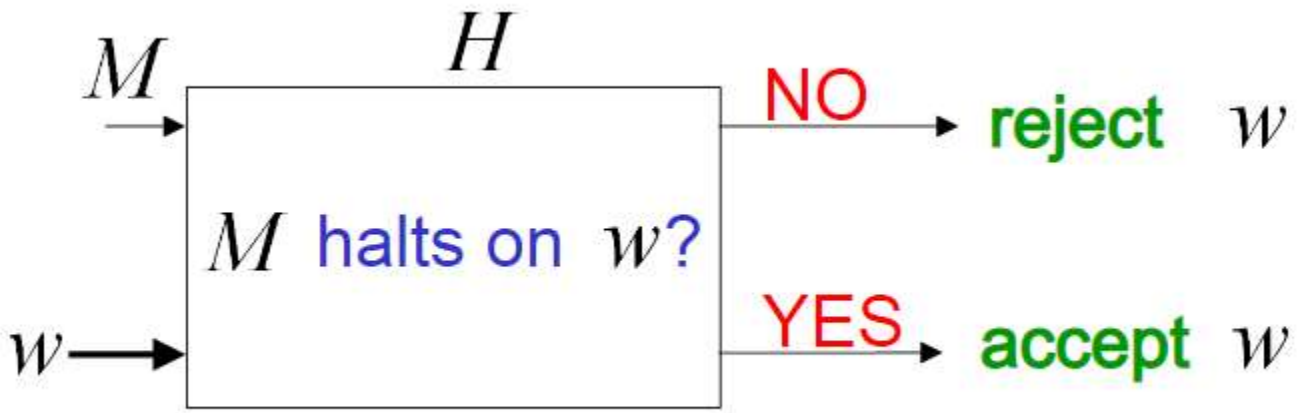
- We show that A_{TM} is reducible to $HALT_{TM}$
- Since A_{TM} is undecidable, so is $HALT_{TM}$

Proving Undecidability via Reductions

Theorem 5.1

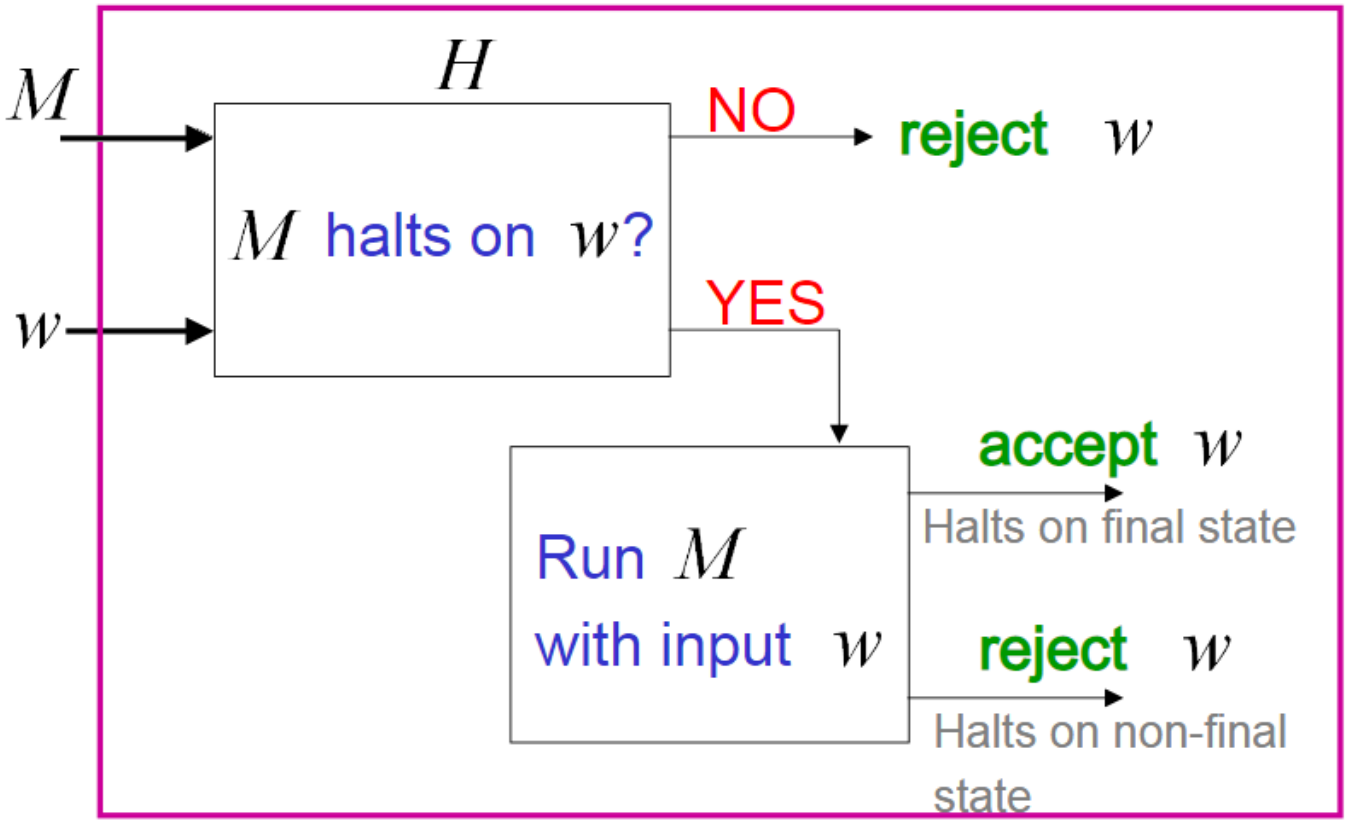
$HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$ is undecidable.

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

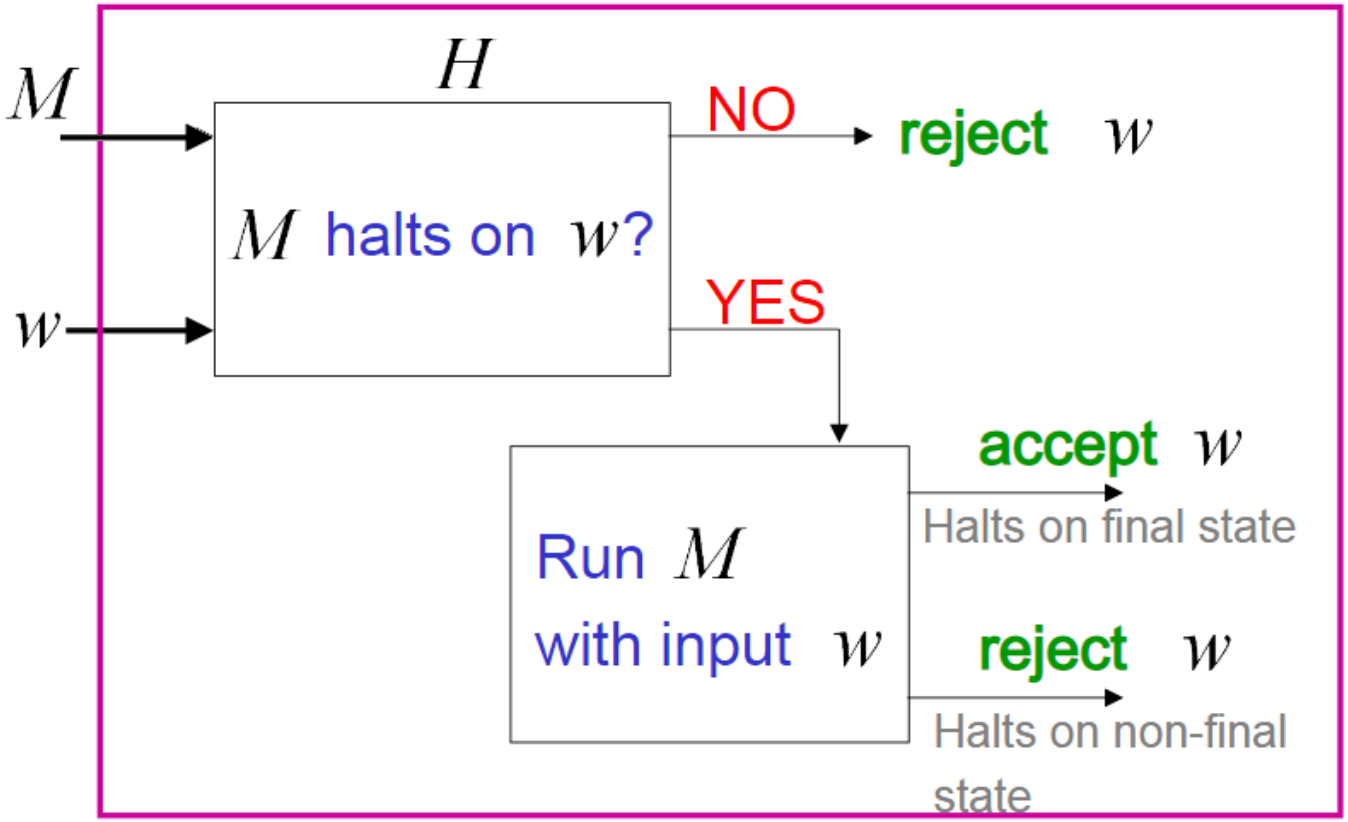


- Then A_{TM} is decidable.

- 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.

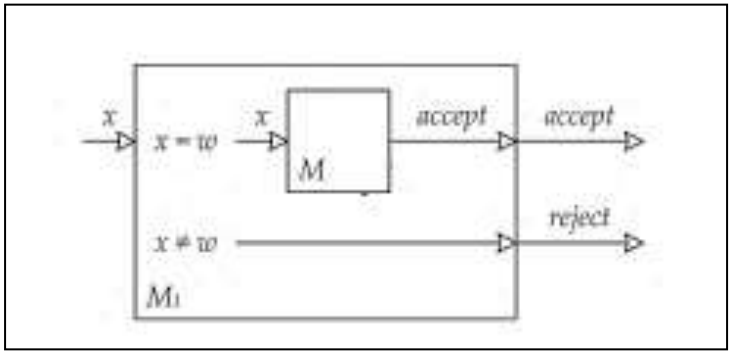
- A decider for A_{TM} via E_{TM} is possible.
- We are given $< M, w >$, and asked to find whether $< M, w > \in A_{TM}$?
- For this, First create M_1

Proving Undecidability via Reductions

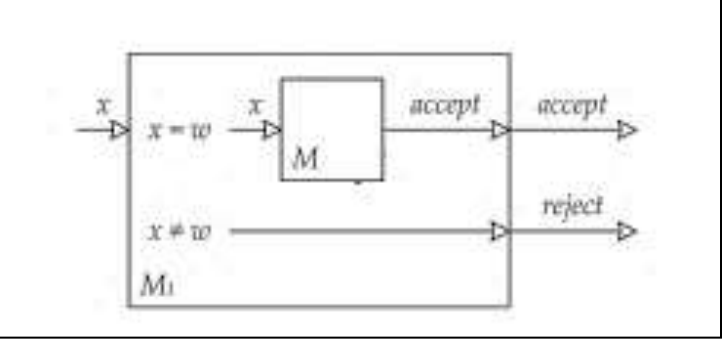
Theorem 5.2

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

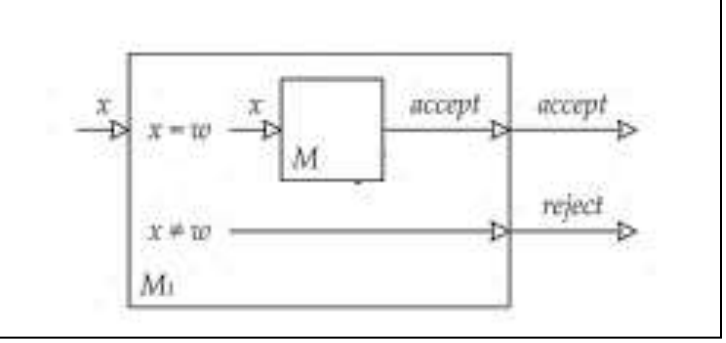
- A decider for A_{TM} via E_{TM} is possible.
- We are given $< M, w >$, and asked to find whether $< M, w > \in A_{TM}$?
- For this, First create M_1



Now, $L(M_1)$ is what?



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



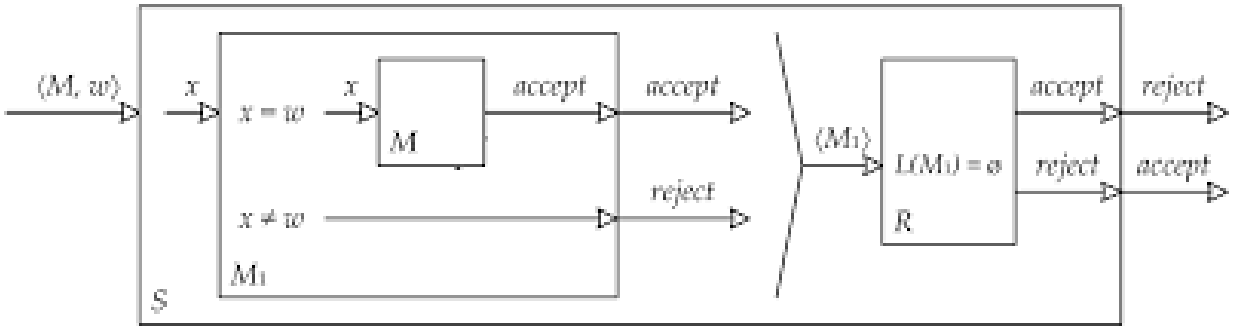
- Now, $L(M_1)$ is what?
- $L(M_1)$ is either $\{w\}$ or is ϕ
- Now, if M_1 is in E_{TM}
  - This means, $L(M_1) = \phi$
  - This means, $< M, w > \notin A_{TM}$
- Now, if M_1 is not in E_{TM}
  - This means, $L(M_1) \neq \phi$
  - This means, $< M, w > \in A_{TM}$

Proving Undecidability via Reductions

Theorem 5.2

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

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



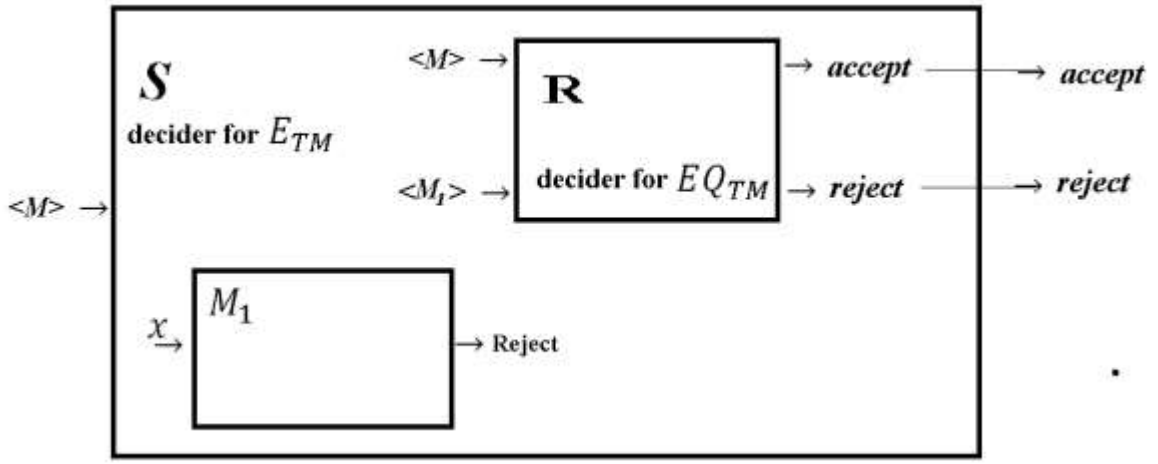
- That is, A_{TM} can be reduced to E_{TM}.
- Contradiction

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.

- 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



This is a decider for E_{TM}

PROOF We let TM R decide EQ_{TM} and construct TM S to decide E_{TM} as follows.

S = "On input $\langle M \rangle$, where M is a TM:

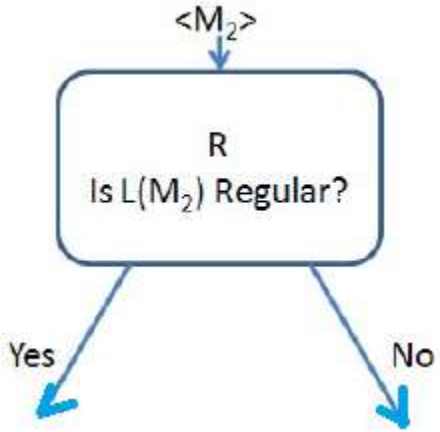
1. Run R on input $\langle M, M_1 \rangle$, where M_1 is a TM that rejects all inputs.
2. If R accepts, *accept*; if R rejects, *reject*."

If R decides EQ_{TM}, S decides E_{TM}. But E_{TM} is undecidable by Theorem 5.2, so EQ_{TM} also must be undecidable.

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 $REGULAR_{TM}$.

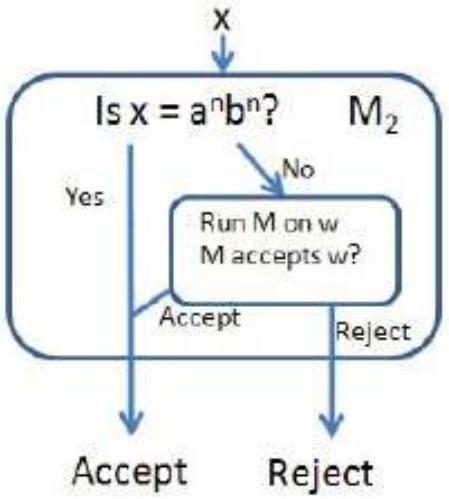


How R can be used to decide $\langle M,w \rangle \in A_{TM}$?

- Build M_2 as shown

We design M_2 to recognize the nonregular language $\{0^n1^n \mid n \geq 0\}$ if M does not accept w, and to recognize the regular language Σ^* if M accepts w.
We must specify how S can construct such an M_2 from M and w.

Here, M_2 works by automatically accepting all strings in $\{0^n1^n \mid n \geq 0\}$. In addition, if M accepts w, M_2 accepts all other strings.

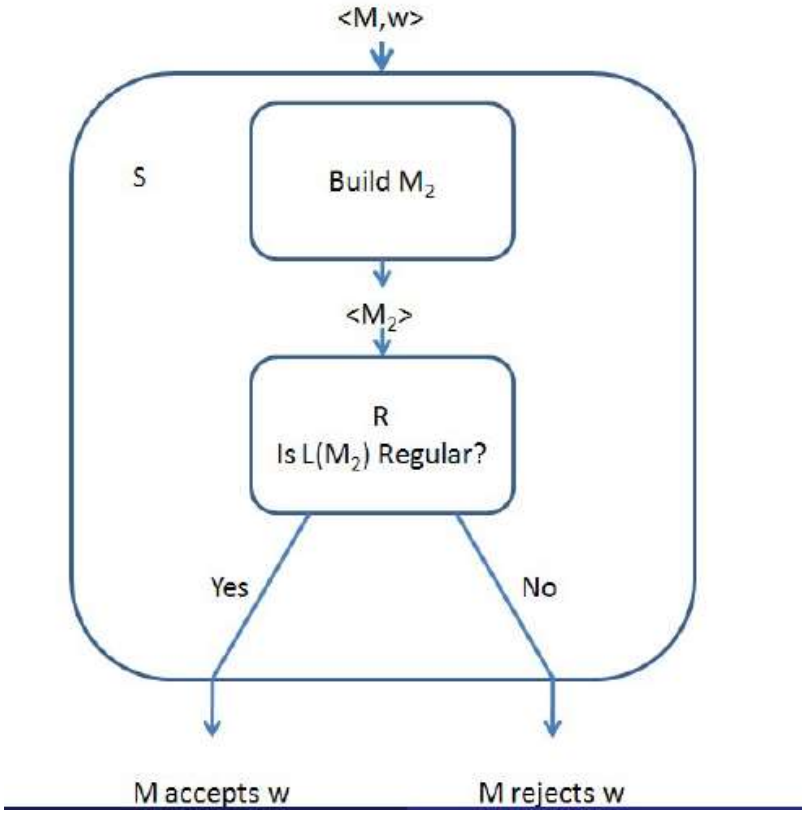


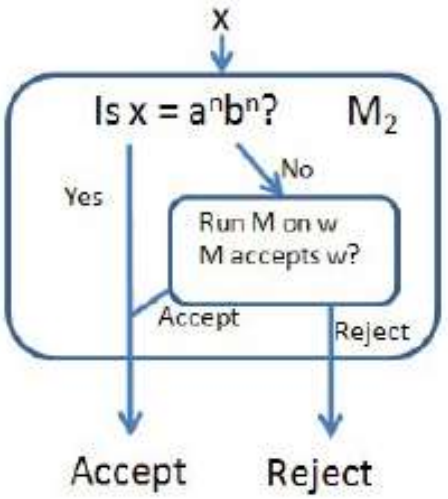
So $L(M_2)$ is = Σ^* if M accepts w
$L(M_2)$ is = $\{a^nb^n\}$ otherwise

We assume that $REGULAR_{TM}$ is decidable by a TM R and use this assumption to construct a TM S that decides A_{TM}.

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

- Build M_2 as shown





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

We assume that REGULAR_{TM} is decidable by a TM R and use this assumption to construct a TM S that decides A_{TM}.

- 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

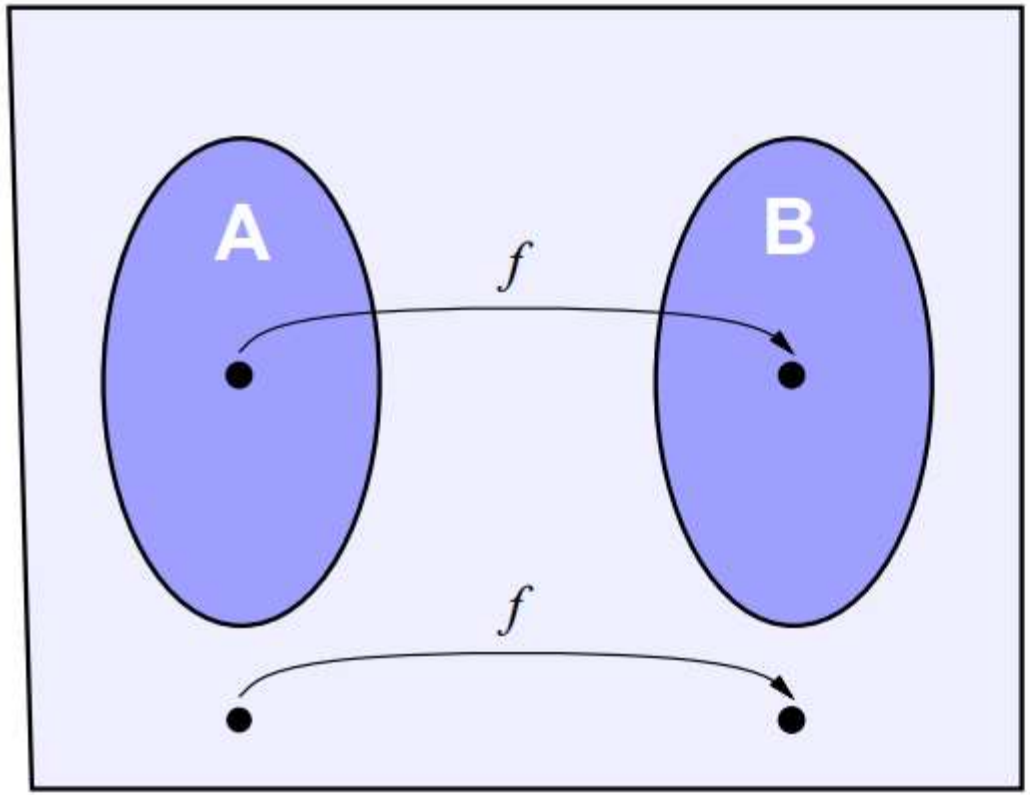
$$f : \Sigma^* \longrightarrow \Sigma^*$$

such that, for every w,

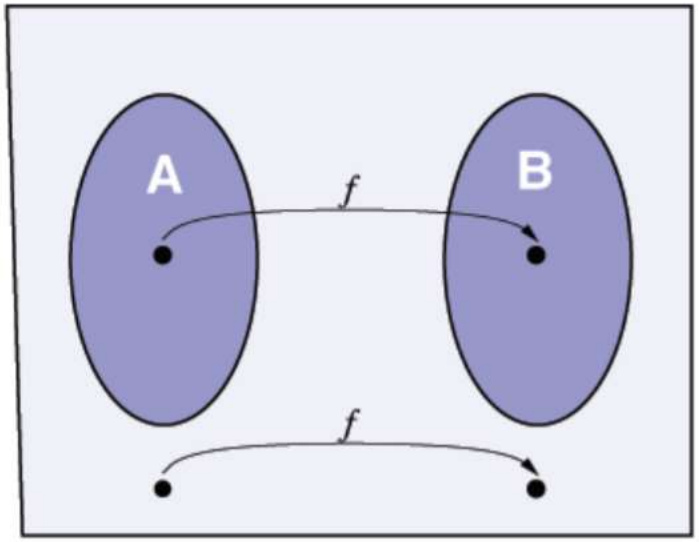
$$w \in A \Longleftrightarrow f(w) \in B.$$

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



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

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

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 $\langle M, w \rangle \in A_{\mathrm{TM}} \iff \langle M', w' \rangle \in H_{\mathrm{TM}}$.

Example: Halting

The following machine computes this function f.

$F =$ on input $\langle M, w \rangle$:

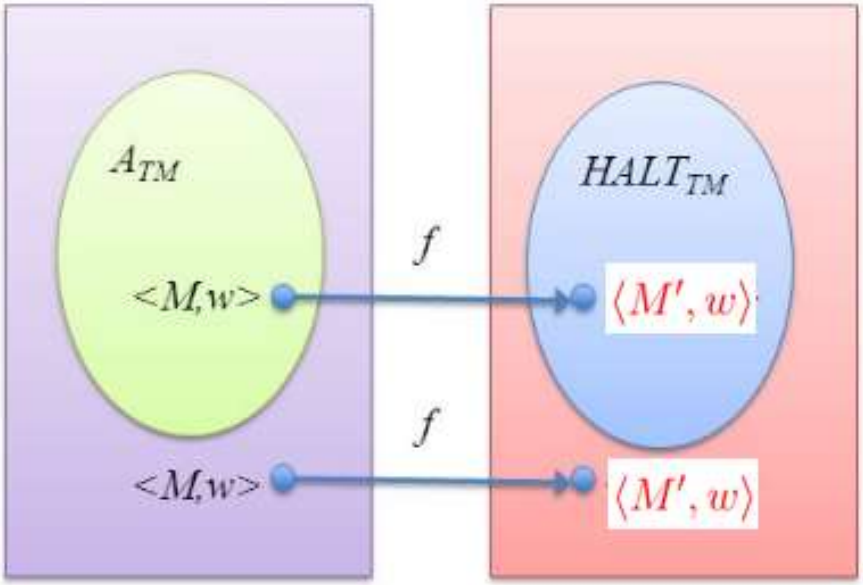
- Construct the following machine M'.

  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}



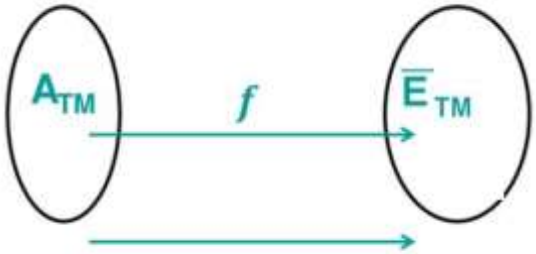
$A_{TM} \leq_m \overline{E_{TM}}$

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



More Theorems Re-examined: E_{TM}

Original Method:

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

M_1 = "On input x:

1. If $x \neq w$, reject.
2. If $x = w$, run M on input w and accept if M does."

Assume that TM R decides E_{TM} and construct TM S that decides A_{TM} as follows.

S = "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

1. Use the description of M and w to construct the TM M_1 just described.
2. Run R on input $\langle M_1 \rangle$.
3. If R accepts, reject; if R rejects, accept."

Mapping Reduction:

Problem: The mapping in the proof is actually A_{TM} to $\neg E_{TM}$ (pay attention to the negation).

Notice: Decidability is not affected by complementation. But can we create a pure mapping reduction?

Proof that a Mapping Reduction is impossible: Suppose for a contradiction that $A_{TM} \leq_m E_{TM}$ via reduction f. It follows from the definition of mapping reducibility that $\neg A_{TM} \leq_m \neg E_{TM}$ via the same reduction function f. However, $\neg E_{TM}$ (Exercise 4.5) is Turing-recognizable and $\neg A_{TM}$ is not Turing-recognizable.

The sensitivity of mapping reducibility to complementation is important in the use of reducibility to prove nonrecognizability of certain languages.

We can also use mapping reducibility to show that problems are not Turingrecognizable.

THEOREM 5.28

If $A \leq_{\mathrm{m}} B$ and B is Turing-recognizable, then A is Turing-recognizable.

COROLLARY 5.29

If $A \leq_{\mathrm{m}} B$ and A is not Turing-recognizable, then B is not Turing-recognizable.

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 $co\mathcal{RE}$.

THEOREM 5.30

EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

PROOF First we show that EQ_{TM} is not Turing-recognizable. We do so by showing that A_{TM} is reducible to $\overline{EQ_{\mathsf{TM}}}$. The reducing function f works as follows.

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

1. Construct the following two machines, M_1 and M_2.
   M_1 = "On any input:
      1. *Reject*."
   M_2 = "On any input:
      1. Run M on w. If it accepts, *accept*."
2. Output $\langle M_1, M_2\rangle$."

Here, M_1 accepts nothing. If M accepts w, M_2 accepts everything, and so the two machines are not equivalent. Conversely, if M doesn't accept w, M_2 accepts nothing, and they are equivalent. Thus f reduces A_{TM} to $\overline{EQ_{\mathsf{TM}}}$, as desired.

Key: We know that $\overline{A_{TM}}$ is not enumerable

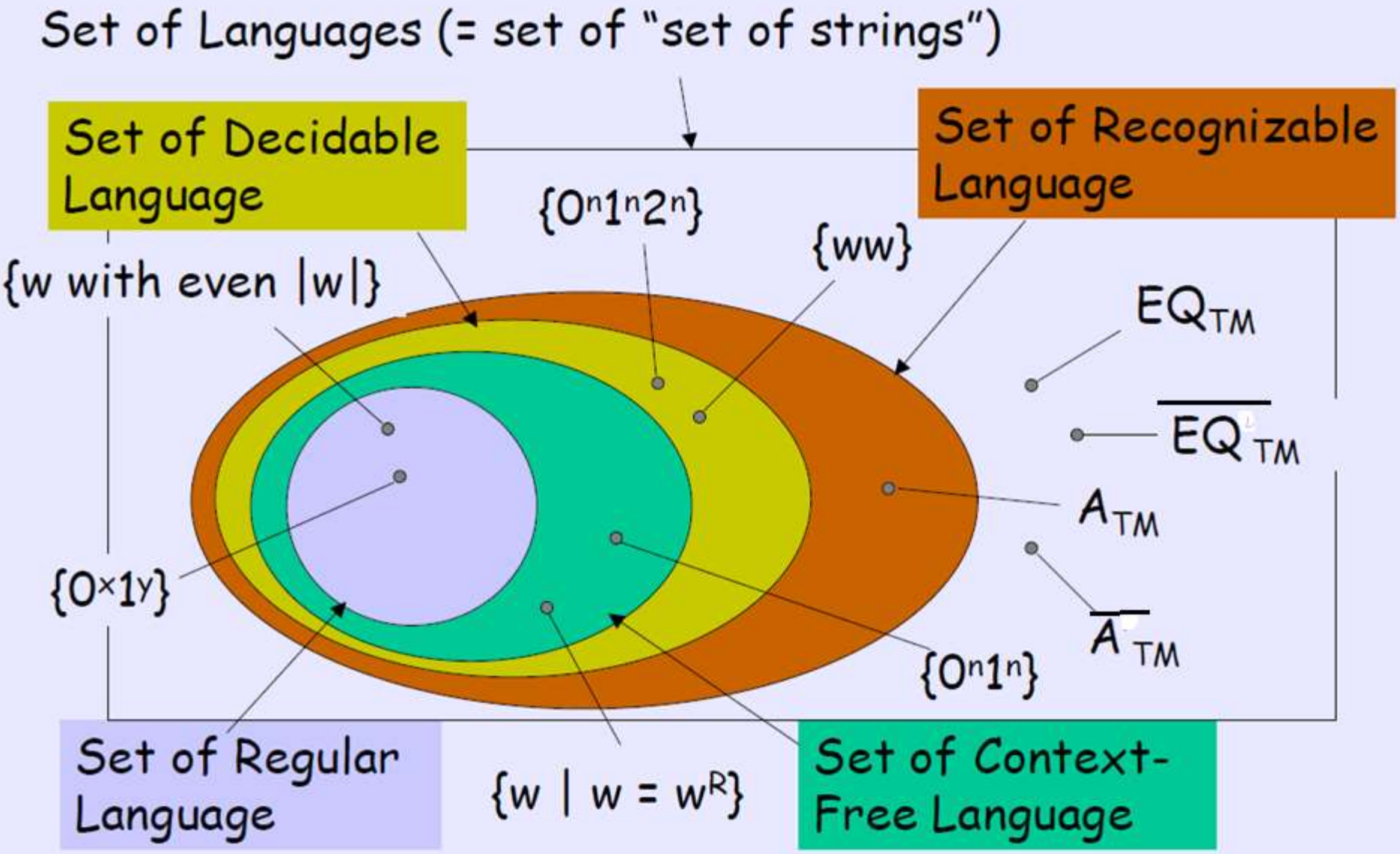
To show that $\overline{EQ_{\mathsf{TM}}}$ is not Turing-recognizable, we give a reduction from A_{TM} to the complement of $\overline{EQ_{\mathsf{TM}}}$—namely, EQ_{TM}. Hence we show that $A_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$. The following TM G computes the reducing function g.

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

1. Construct the following two machines, M_1 and M_2.
   M_1 = "On any input:
      1. *Accept*."
   M_2 = "On any input:
      1. Run M on w.
      2. If it accepts, *accept*."
2. Output $\langle M_1, M_2 \rangle$."

The only difference between f and g is in machine M_1. In f, machine M_1 always rejects, whereas in g it always accepts. In both f and g, M accepts w iff M_2 always accepts. In g, M accepts w iff M_1 and M_2 are equivalent. That is why g is a reduction from A_{TM} to EQ_{TM}.

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?

•A property is called to be trivial if either it is not satisfied by any recursively enumerable languages, or if it is satisfied by all recursively enumerable languages.

•A non-trivial property is satisfied by some recursively enumerable languages and are not satisfied by others.

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

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., $\left[\frac{ca}{a}\right]$.
- Suppose we have dominos

$$\left\{\left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right]\right\}$$

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

$$\left[\frac{a}{ab}\right]\left[\frac{b}{ca}\right]\left[\frac{ca}{a}\right]\left[\frac{a}{ab}\right]\left[\frac{abc}{c}\right] = \frac{abcaaabc}{abcaaabc}$$

- 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

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

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

$$i_1, i_2, \ldots, i_\ell$$

of numbers from $\{1, 2, \ldots, k\}$ (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

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

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

$$i_1, i_2, \ldots, i_\ell$$

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

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.

RE

- Closed under union
- Closed under intersection
- **Not** closed under complementation

R

- Closed under union
- Closed under intersection
- Closed under complementation.