CS 6041
Theory of Computation

Reducibility

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Reducibility

- If A reduces to B, we can use a solution to B to solve A

- Example:
  - Look for a place - - > Get a map
  - Go to a place - -> Take a car

- If A is reduced to B:
  - If we can do B, then we can also do A
  - If we cannot do A, then we cannot do B

Counter-proposition

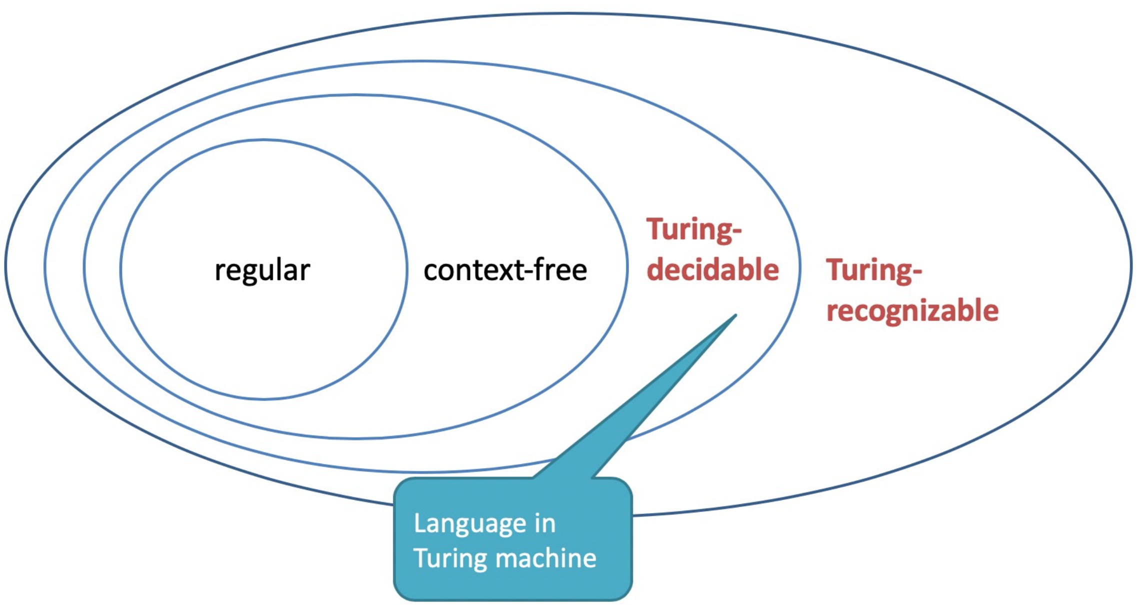
Revisit: The output of Turing Machine

- Accept
- Reject
- Loop

Halt -> Decidable

= Never Halt

Recognizable



Revisit: Decidability

- Decidable?

	DFA/NFA/RE	CFG	TM
Acceptance (A)	√	√	×
Emptiness (E)	√	√	
Equivalence (EQ)	√	×	

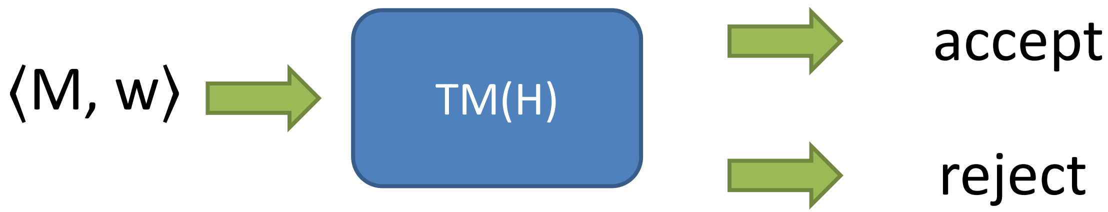
Revisit: Decidable problems for Turing Machine

- Acceptance problem for Turing Machine
  - Whether a Turing machine accepts a given input string

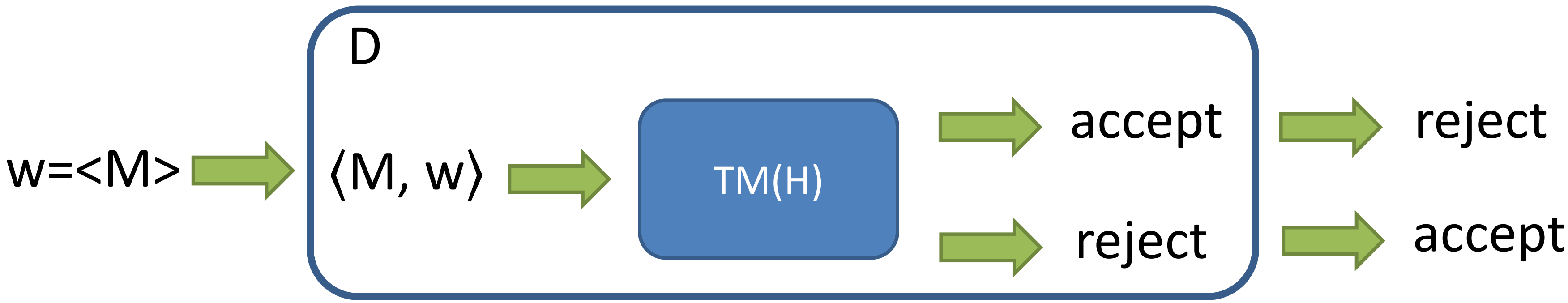
We do not know whether M accepts w or not

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

Revisit: Decidable problems for Turing Machine



Revisit: Decidable problems for Turing Machine



$$D(<M>) = \begin{cases} \text{accept,} & \text{if M does not accept } <M> \\ \text{reject,} & \text{if M accepts } <M> \end{cases}$$

$$D(<D>) = \begin{cases} \text{accept,} & \text{if D does not accept } <D> \\ \text{reject,} & \text{if D accepts } <D> \end{cases}$$

Contradiction!

Revisit: Decidability

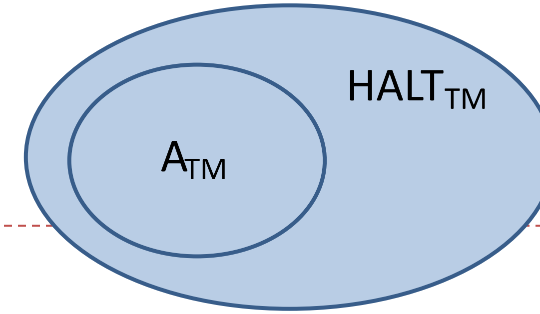
- Decidable?

	DFA/NFA/RE	CFG	TM
Acceptance (A)	√	√	×
Emptiness (E)	√	√	
Equivalence (EQ)	√	×	
Halt			?

1. Halting problem

- TM halting problem:
  - whether a Turing machine M halts (by accepting or rejecting) on a given input w

1. Halting problem



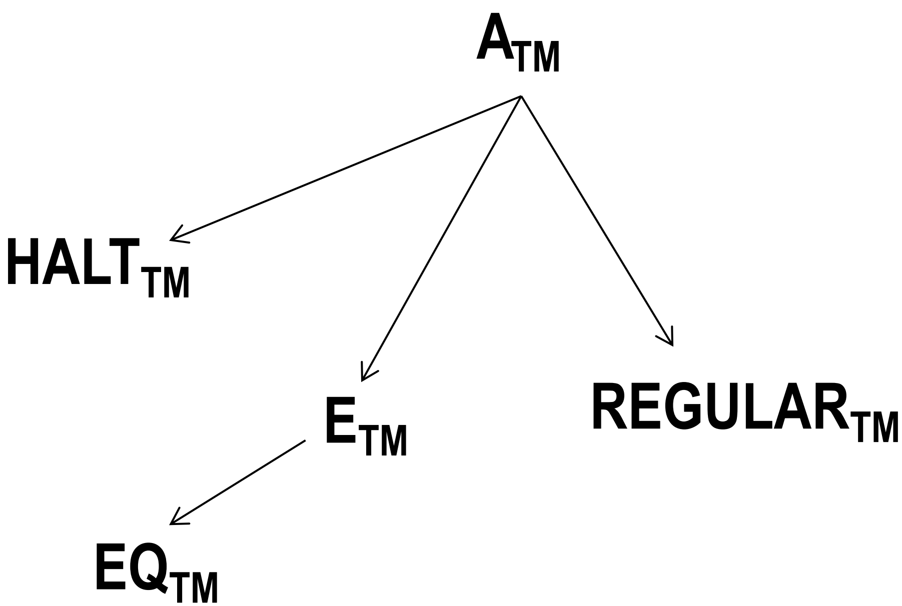
- Language
  - $HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}$.

    vs.

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

1. Halting problem

- Relationship of languages on reducibility



Theorem 5.1

- $HALT_{TM}$ is undecidable

- Proof (prove by contradiction):

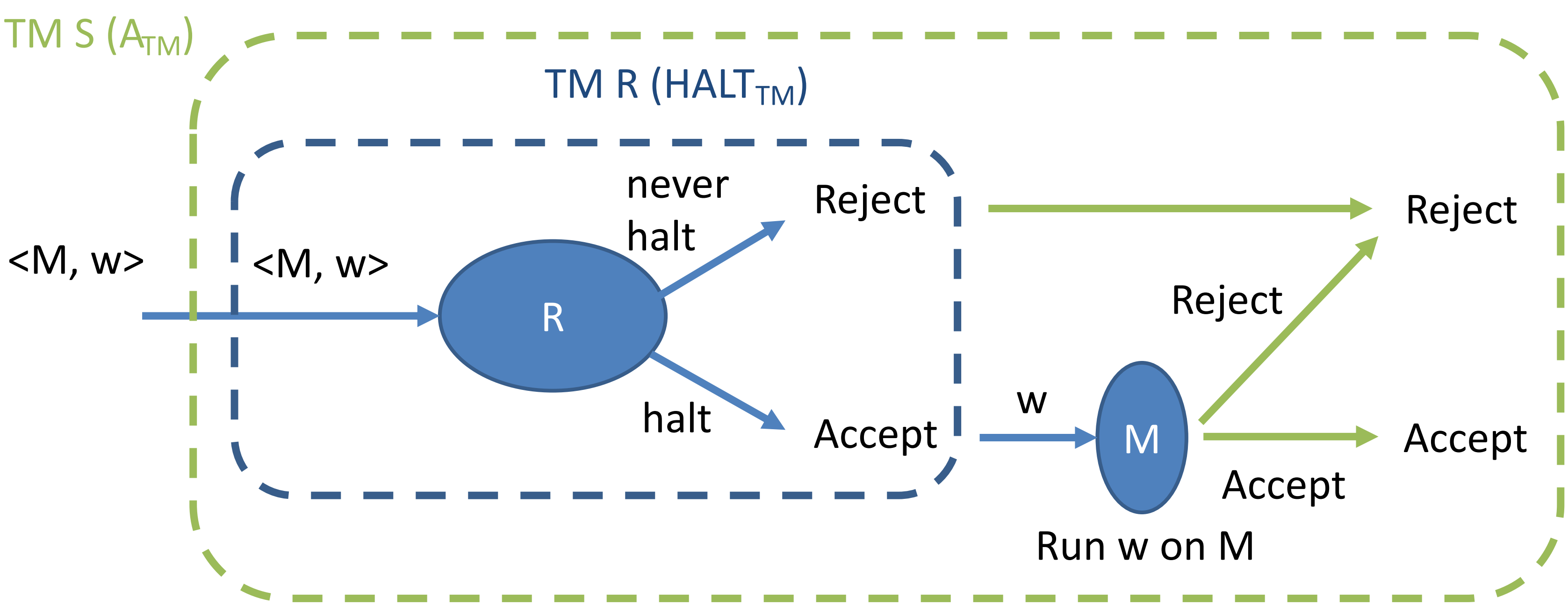
  Suppose TM R decides $HALT_{TM}$

  Then we create a TM S to decide A_{TM}

S = "On input ⟨M, w⟩, M is a TM and w is a string:

1. Run TM R on input ⟨M, w⟩.
2. If R rejects, which means never halt. Then S rejects.
3. If R accepts, which means R will halt (accept or reject) we simulate M on w until it halts.
4. If M has accepted, accept; if M has rejected, reject."

It means A_{TM} is decidable. Contradiction!



Theorem 5.1

- $HALT_{TM}$ is undecidable
- If the $HALT_{TM}$ is decidable, then we can get A_{TM} is also decidable. However, we already proved A_{TM} is undecidable.
- A_{TM} is reduced to $HALT_{TM}$

Rethink A_{TM}

- Acceptance problem for Turing Machine
  - Whether a Turing machine accepts a given input string

We do not know whether M accepts w or not is because TM could never halt

The output of Turing Machine

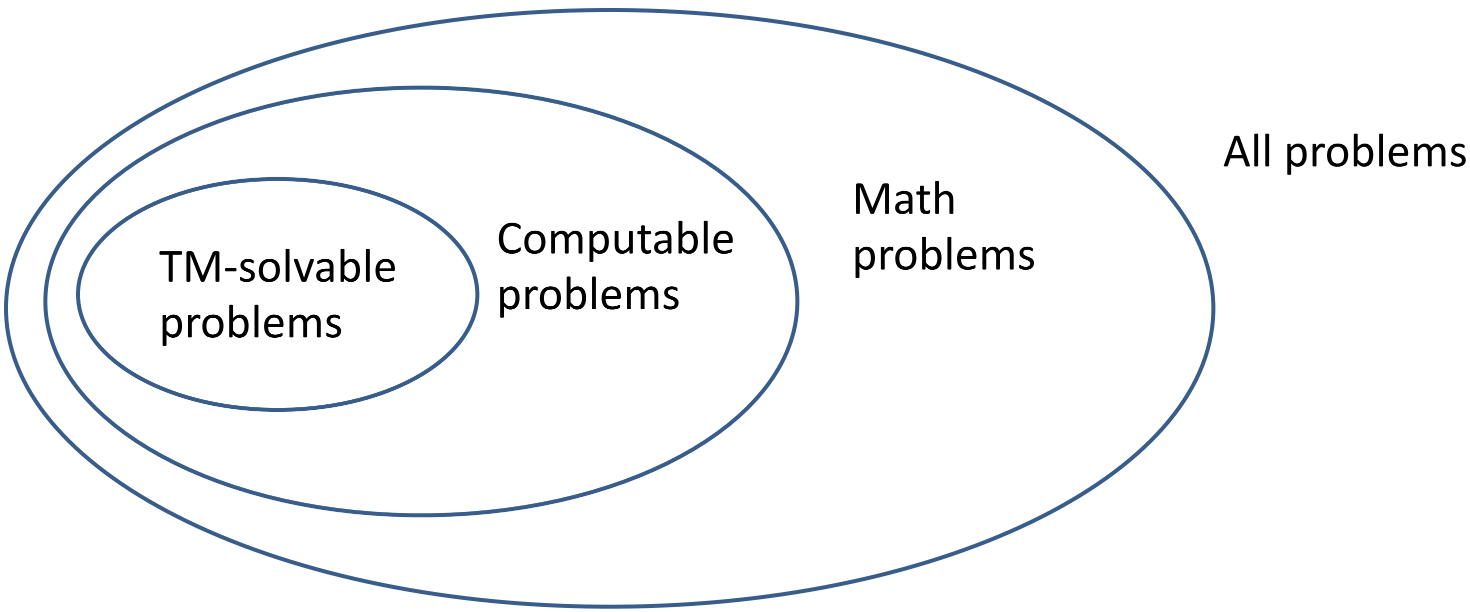
- Accept
- Reject

$\left.\begin{array}{l}\text{Accept}\\\text{Reject}\end{array}\right\}$ Halt

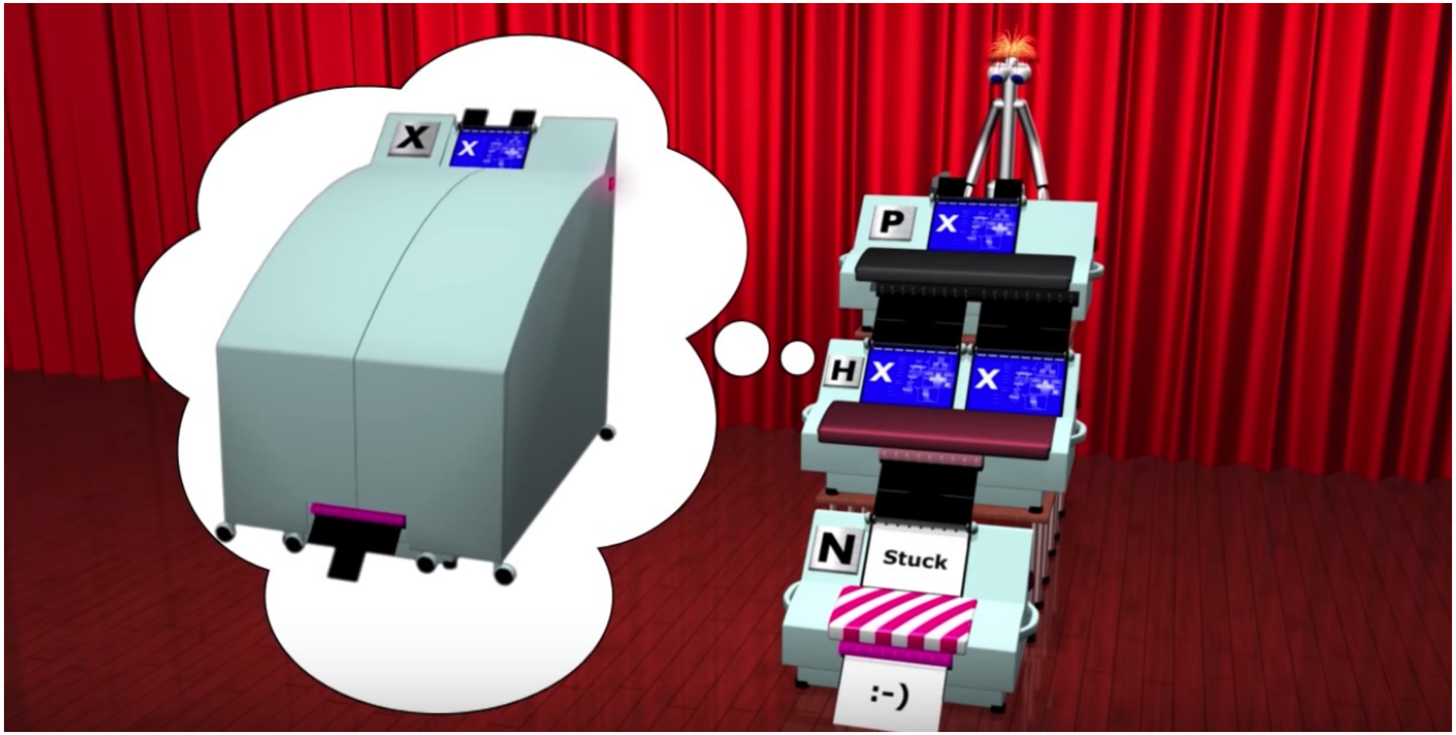
- Loop = Never Halt

HALT$_{TM}$

- The HALT$_{TM}$ problem just proves that the Turing machine (or computers) is not omnipotent



$HALT_{TM}$



https://youtu.be/92WHN-pAFCs

2. Emptiness of Turing machine

- Decidable?

	DFA/NFA/RE	CFG	TM
Acceptance (A)	√	√	×
Emptiness (E)	√	√	?
Equivalence (EQ)	√	×	

2. Emptiness of Turing machine

- Emptiness of Turing machine
  - Whether or not a TM never accept any string w

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

Theorem 5.2

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

- Proof:

Input is M
Nothing related to w

We need create contradiction between

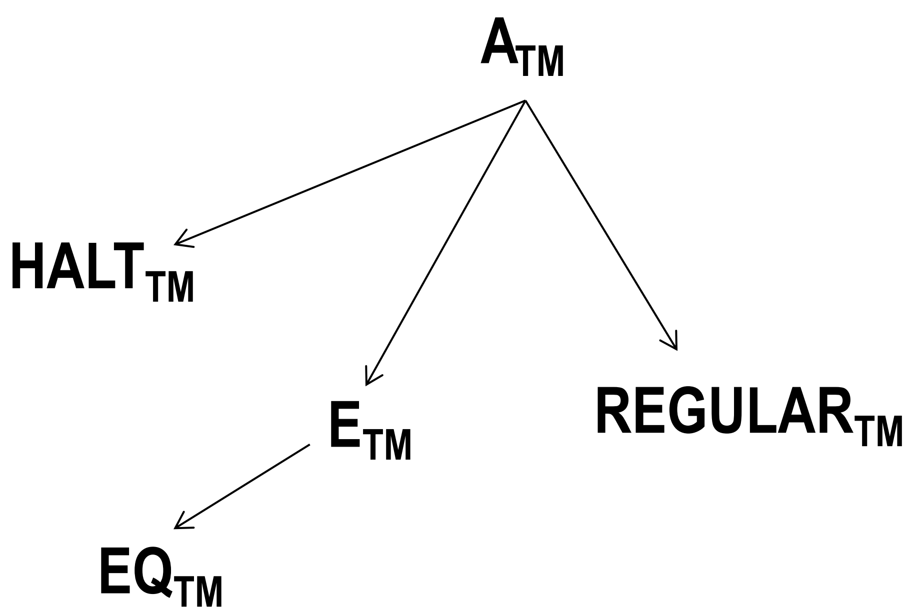
$$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$$

and

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

Theorem 5.2

- Relationship of languages on reducibility



Theorem 5.2 proof

- Proof:

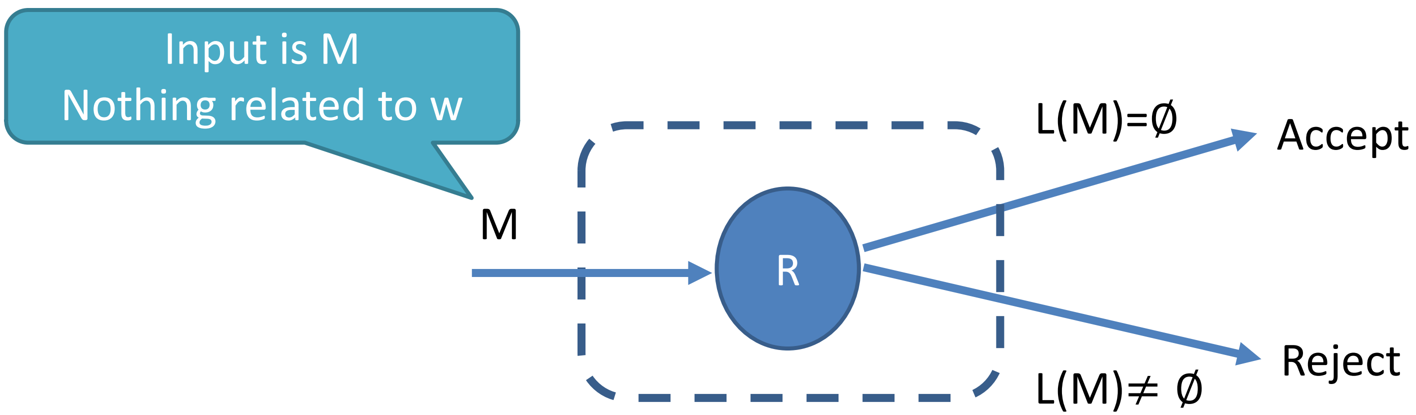
  Suppose E_{TM} is decidable, then TM R decides E_{TM}

  R = "On input M,

  if M does not accept anything, then L(M)=∅, R accept;

  if M accept something, then L(M)≠ ∅, R reject;

  "



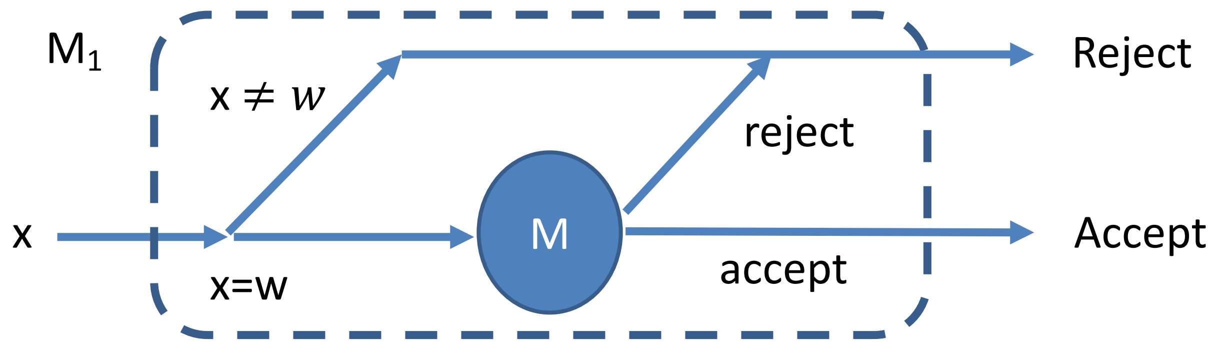
Theorem 5.2 proof

- Proof

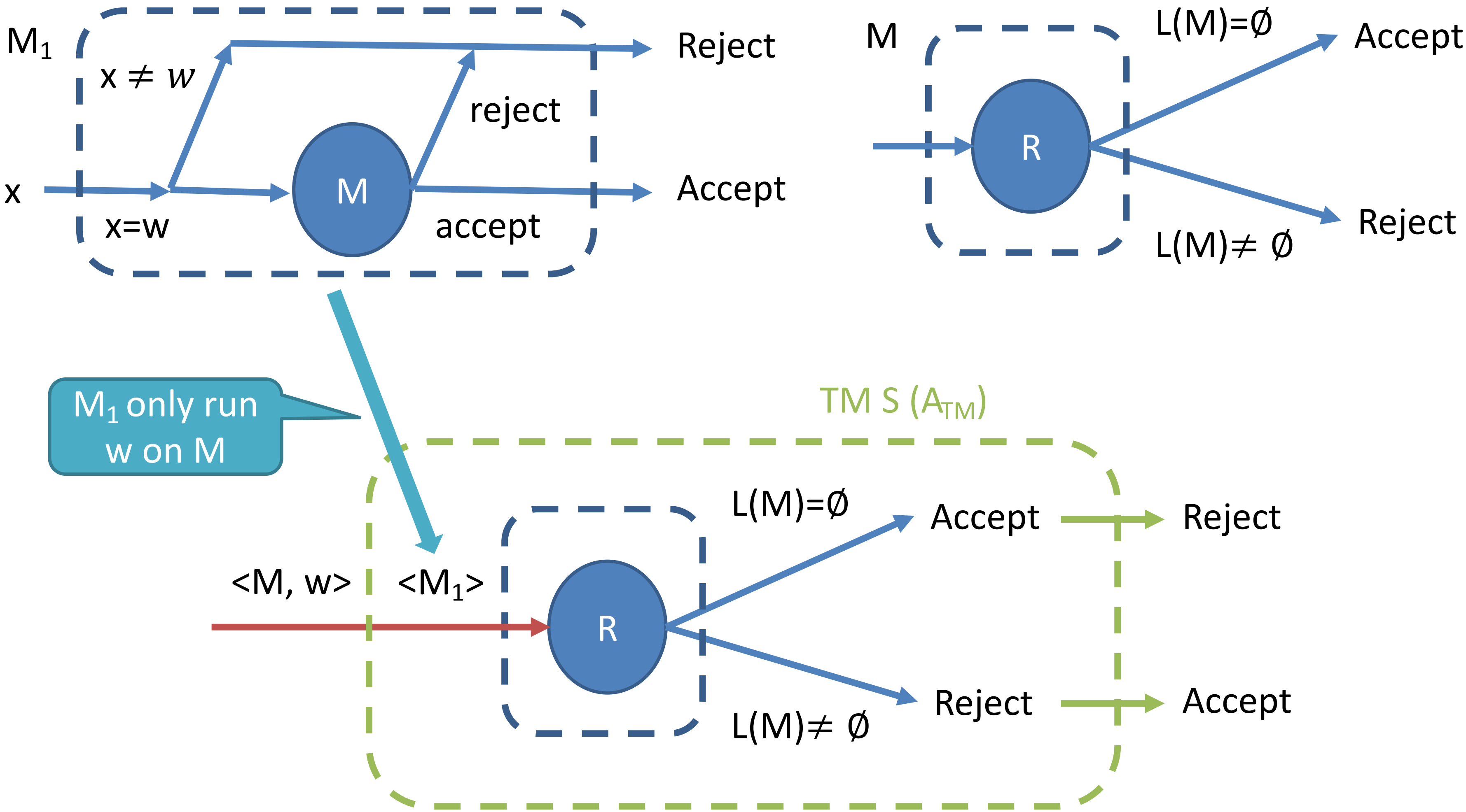
  Create a TM M_1, that

  - If input $\neq w$, M_1 reject (Only string M_1 can accept is w);
  - If input is w, test w on M
    - If M accepts w, M_1 accepts
    - If M rejects w, M_1 rejects

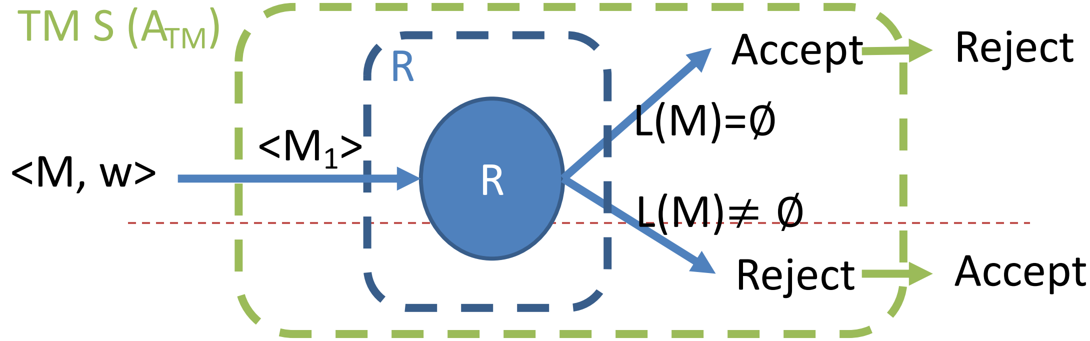
Create a TM M_1 involving both M and w



Theorem 5.2 proof



- # Proof



Integrate the R and M_1

R = "On input M_1,

    if M does not accept w, then L(M)=∅, R accept;

    if M accept w, then L(M)≠ ∅, R reject;

    "

Based on R, we can create TM S to decide A_{TM}

S = "On input <M, w>

    Run R on input M_1 (M_1 = <M, w>)

    If R accepts, S rejects;

    If R rejects, S accepts.

    ",

M rejects w ⇒ R accepts ⇒ S rejects

M accepts w ⇒ R rejects ⇒ S accepts

Contradiction! A_{TM} is not decidable

3. Regular issue of TM

- Decidable?

	DFA	CFG	TM
Acceptance (A)	√	√	×
Emptiness (E)	√	√	×
Equivalence (EQ)	√	×	
Halt			×
Regular			?

3. Regular issue of TM

- Regular issue of TM
  - Whether a given Turing machine has an equivalent finite automaton or recognizes a regular language

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

Theorem 5.3

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

- Proof:

Presentation in next lecture by students

4. Equivalence of TM

- Decidable?

	DFA	CFG	TM
Acceptance (A)	√	√	×
Emptiness (E)	√	√	×
Equivalence (EQ)	√	×	?
Halt			×
Regular			×

4. Equivalence of TM

- Definition
  - Whether two TMs can recognize the same language

- Language
  - $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid M_1$ and M_2 are TMs
    and
    $L(M_1) = L(M_2)\}$

Theorem 5.4

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

- Proof:

  Presentation in next lecture by students

4. Equivalence of TM

- Decidable?

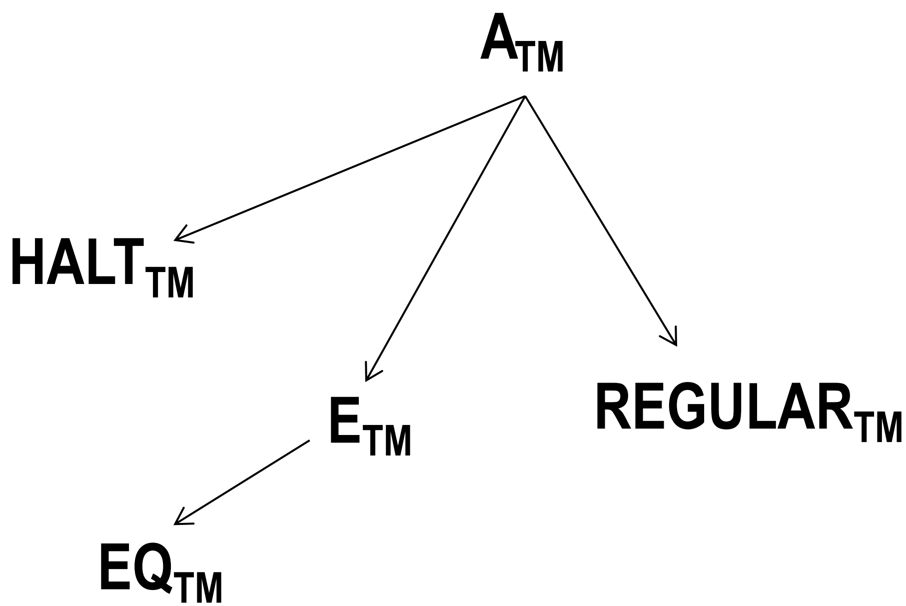
	DFA	CFG	TM
Acceptance (A)	√	√	×
Emptiness (E)	√	√	×
Equivalence (EQ)	√	×	×
Halt			×
Regular			×

Conclusion

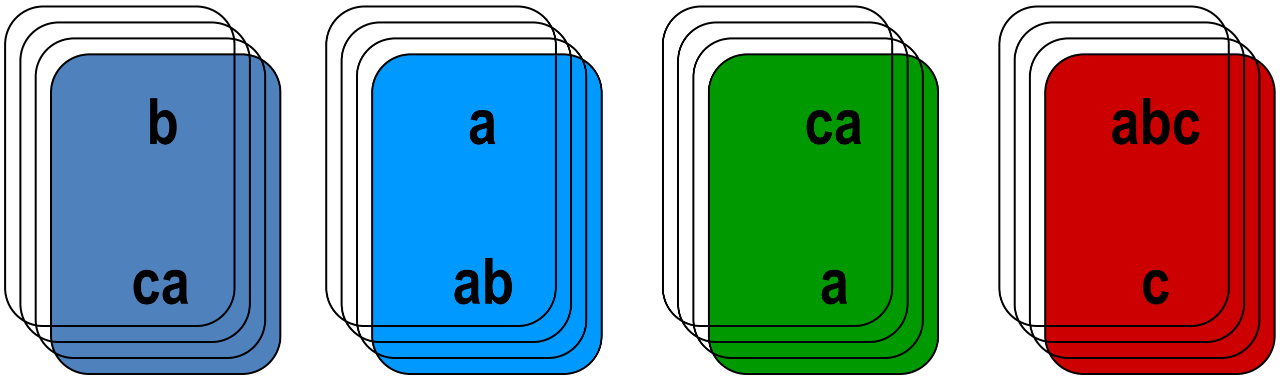
- $HALT_{TM}$ is undecidable
  - We do not know whether a TM will halt on a given input
- E_{TM} is undecidable
  - We do not know whether a TM never accept any strings
- $REGULAR_{TM}$ is undecidable
  - We do not know whether a TM has an equivalent DFA/NFA/RE
- EQ_{TM} is undecidable
  - We do not know whether two VMs recognize the same language

Conclusion

- Relationship of languages on reducibility

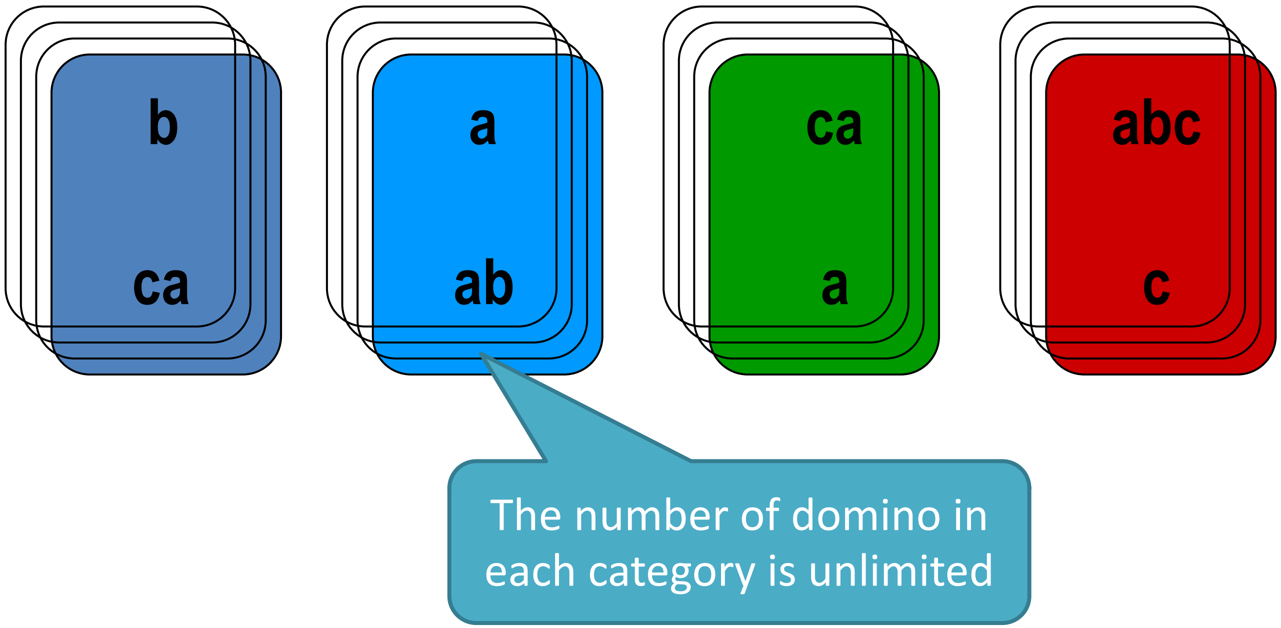


Post Correspondence Problem (PCP)

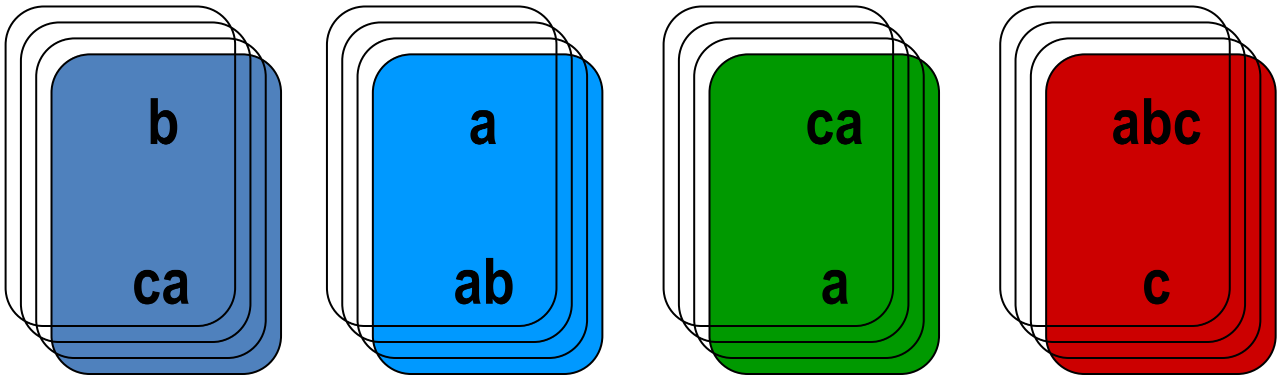


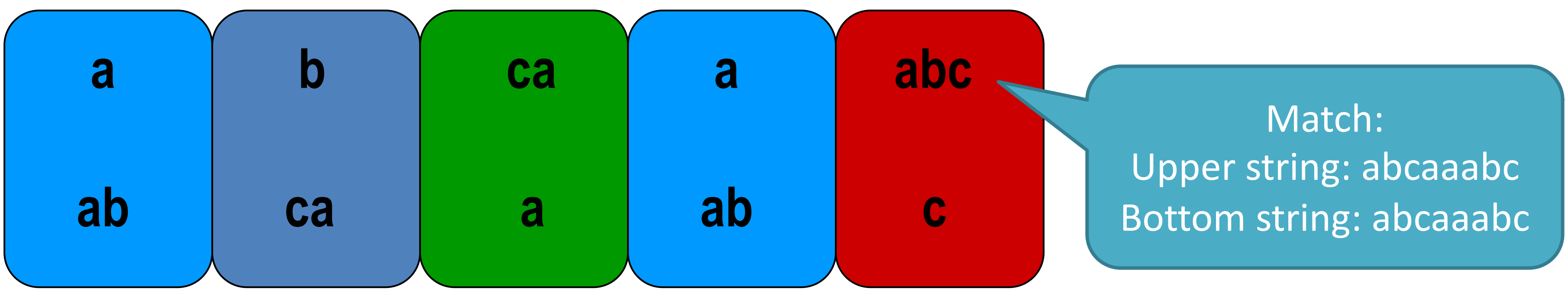
We have limited categories of dominos

Post Correspondence Problem (PCP)



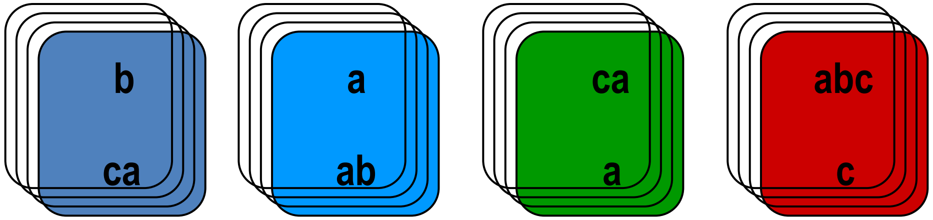
Post Correspondence Problem (PCP)

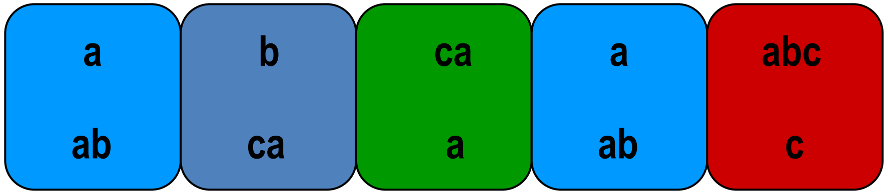




Post Correspondence Problem (PCP)

- Whether a collection of dominos has a match



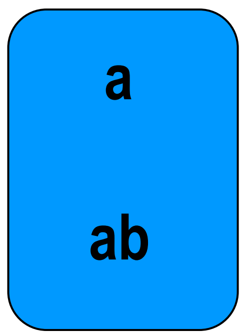


- PCP = {⟨P⟩ | P is an instance of the Post Correspondence Problem with a match}.

Description of PCP

An individual domino

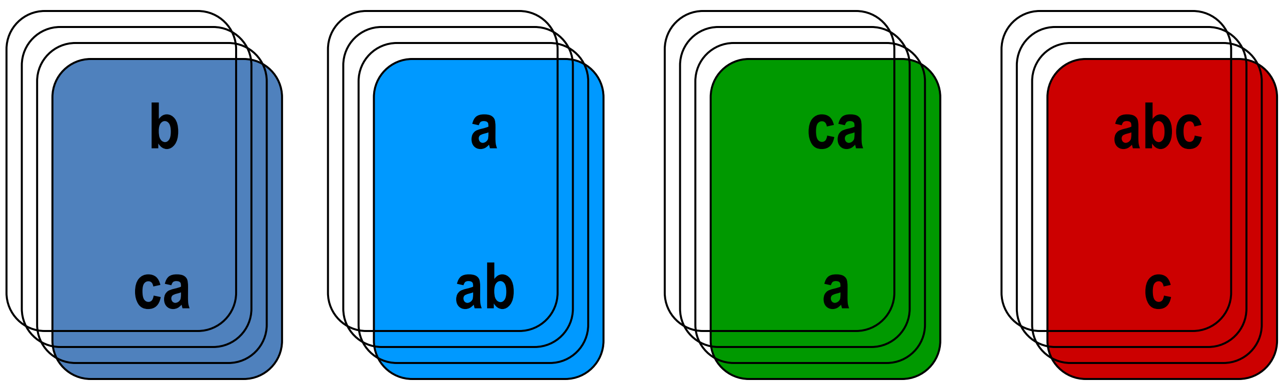
$$\left[\frac{a}{ab}\right]$$



Description of PCP

A collection of dominos

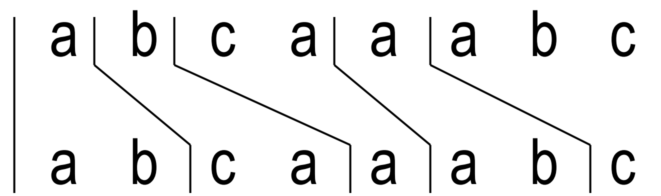
$$\left\{\left[\frac{b}{ca}\right],\left[\frac{a}{ab}\right],\left[\frac{ca}{a}\right],\left[\frac{abc}{c}\right]\right\}$$

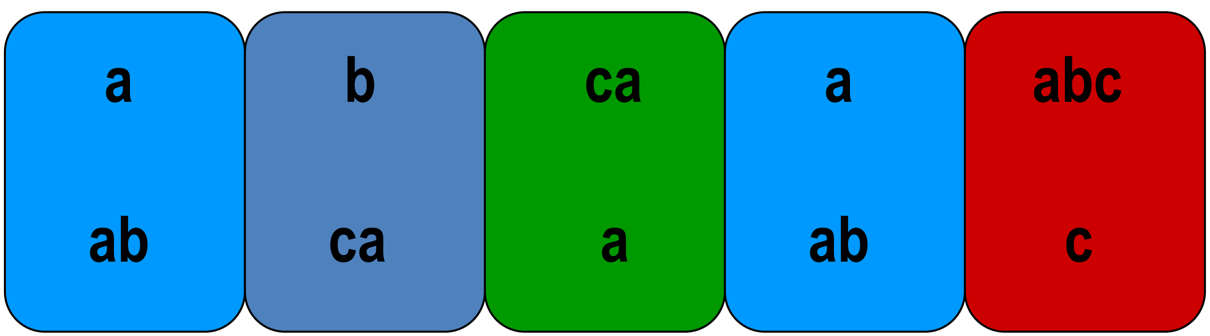


Description of PCP

A match

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





A collection without a match

- For a given collection

$$\left\{\left[\frac{abc}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{acc}{ba}\right]\right\}$$

- It cannot contain a match because every top string is longer than the corresponding bottom string

Theorem 5.15

- PCP is undecidable

- Proof idea:

  Suppose PCP is decidable

  We construct TM S to decide A_{TM} (Theorem 4.11: A_{TM} is undecidable)

Conclusion

- Relationship of languages on reducibility