CS 6041
Theory of Computation

Reducibility

Kun Suo

Computer Science, Kennesaw State University

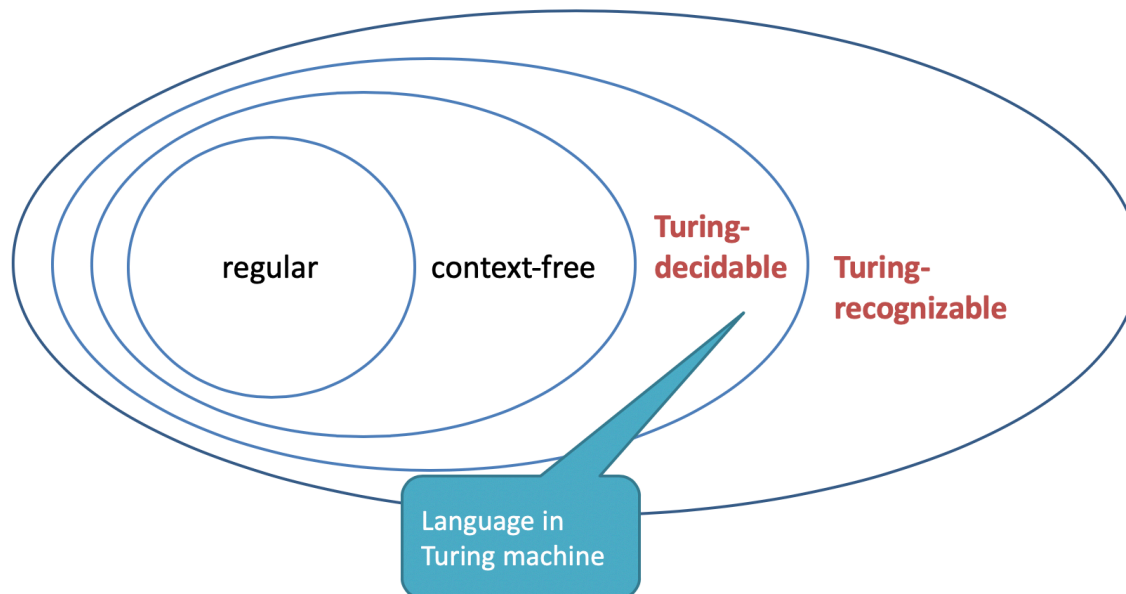https://kevinsuo.github.io/

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   Halt -> Decidable   Recognizable

- Loop   =   Never Halt

regular   context-free   **Turing-decidable**   **Turing-recognizable**

Language in Turing machine

Revisit: Decidability

- Decidable?

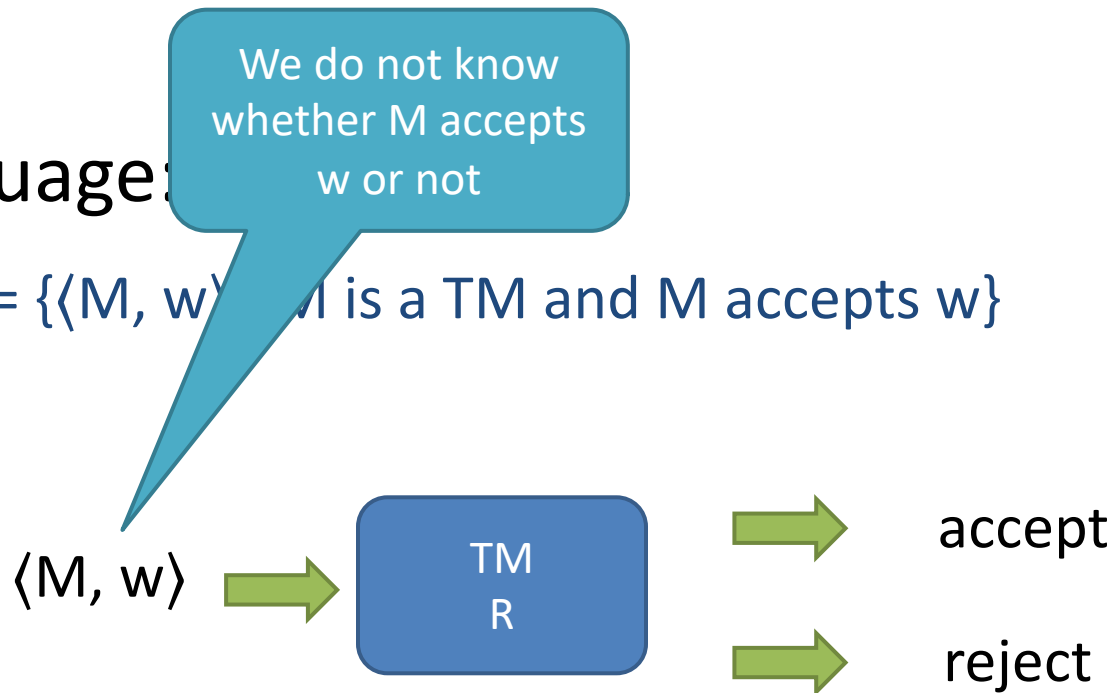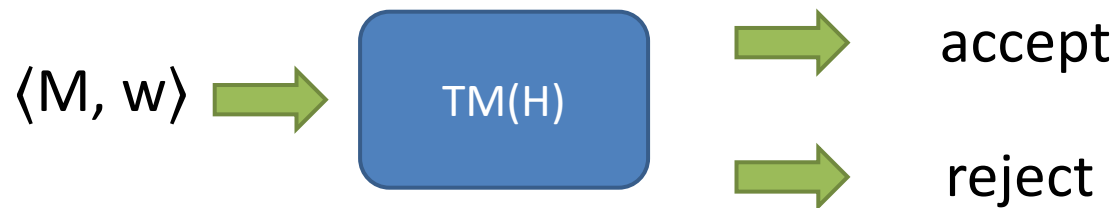|	DFA/NFA/RE	CFG	TM
Acceptance (A)	√	√	×
Emptiness (E)	√	√	
Equivalence (EQ)	√	×	

Revisit: Decidable problems for Turing Machine

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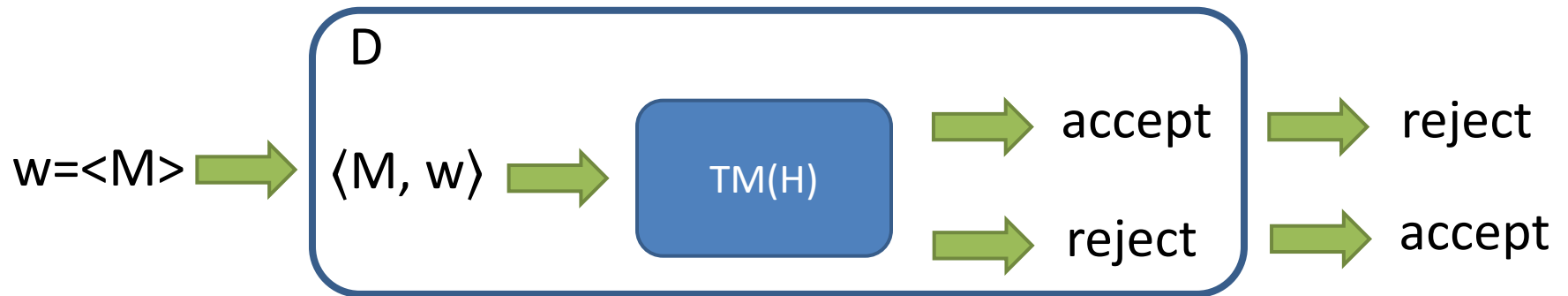 - Whether a Turing machine accepts a given input string

- Language:

  We do not know whether M accepts w or not

  - A_{TM} = {⟨M, w⟩ | M is a TM and M accepts w}

⟨M, w⟩ → TM R → accept

→ reject

Revisit: Decidable problems for Turing Machine

⟨M, w⟩ → TM(H) → accept

reject

w=<M> → D [⟨M, w⟩ → TM(H) → accept → reject

reject → accept]

Revisit: Decidable problems for Turing Machine

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

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

Revisit: Decidability

- Decidable?

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

1. Halting problem

- TM halting problem:

    o whether a Turing machine M halts (by accepting or rejecting) on a given input w

$\langle M, w \rangle$ ➡ **TM R** ➡ accept

➡ reject

1. Halting problem

- Language

    ○ $HALT_{TM}$ = {⟨M, w⟩ | M is a TM and M halts on input w}.

    vs.

    A_{TM} = {⟨M, w⟩ | M is a TM and M accepts input w}.

⟨M, w⟩ → TM R → accept
              → reject

Theorem 5.1

- HALT$_{TM}$ is undecidable

- Proof (prove by contradiction):

Suppose TM R decides HALT$_{TM}$

⟨M, w⟩ ➡ [TM R] ➡ accept
[TM R] ➡ reject

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!
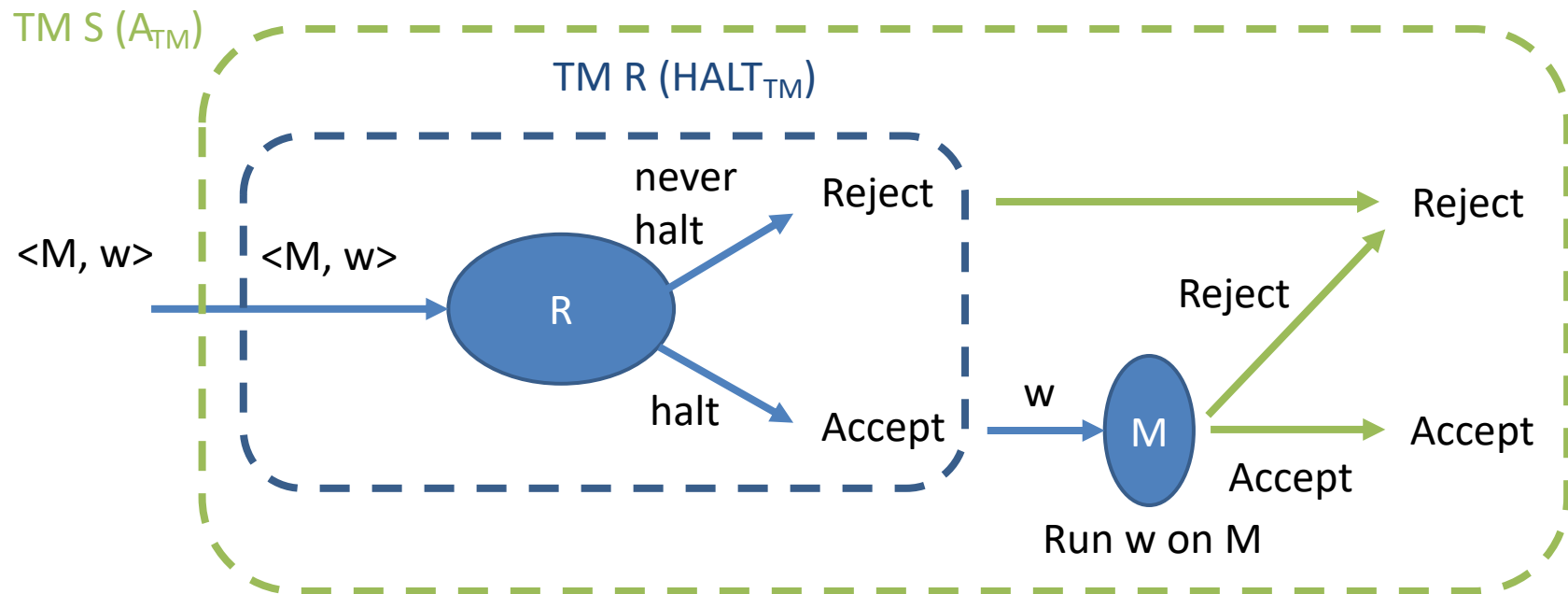
Theorem 5.1

- $HALT_{TM}$ 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.

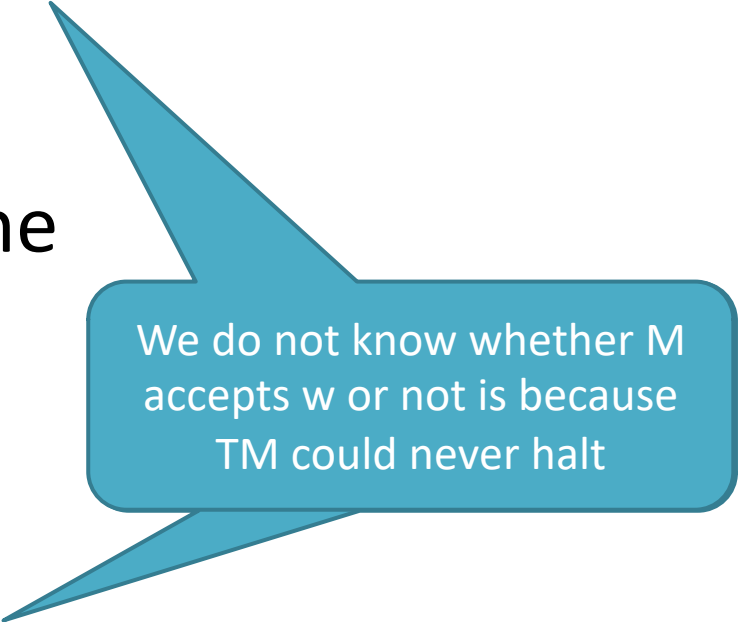- A_{TM} is reduced to $HALT_{TM}$

Rethink A$_{TM}$

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

The output of Turing Machine

- Accept ⎫
- Reject ⎬ Halt

- Loop = Never Halt

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

HALT_{TM}

```
God(source code, string) {
    if source code halt:
        return 1;
    else
        return 0;
}


Bug(source code) {
    if (God (source code, source code)) {
        while(1){}
    } else {
        halt;
    }
}
```
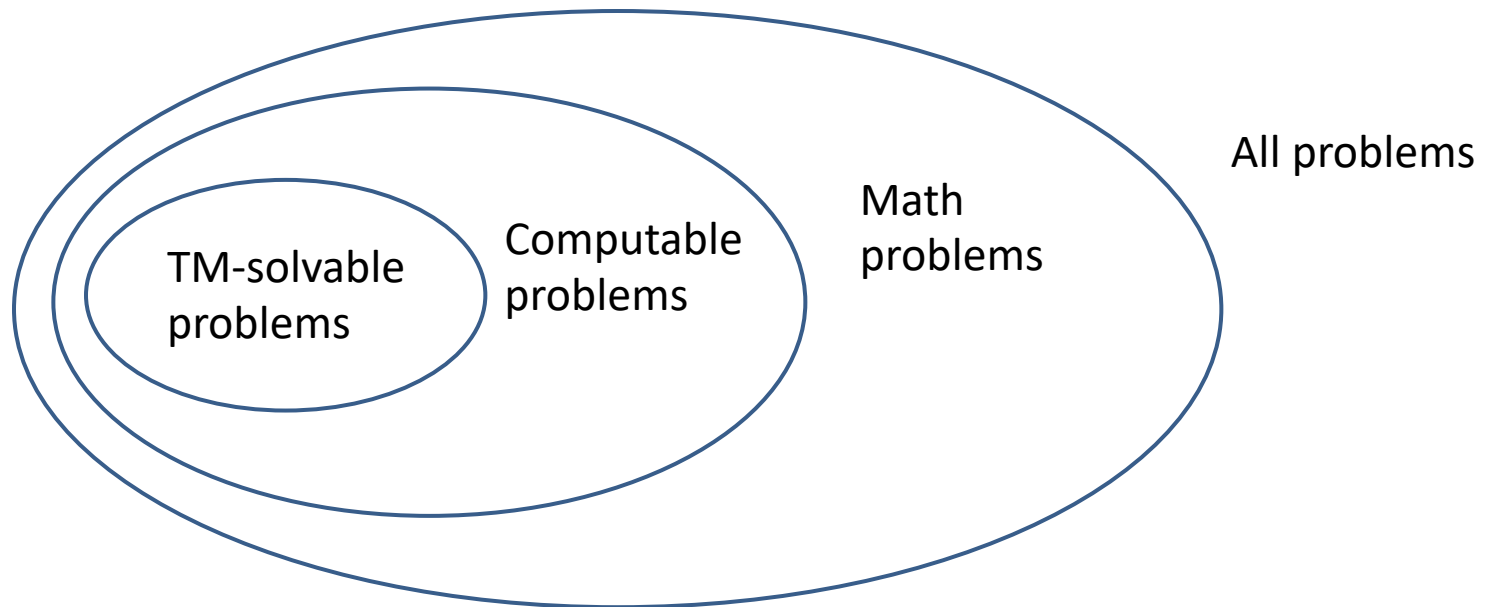
If God thinks bug code halts, return 1

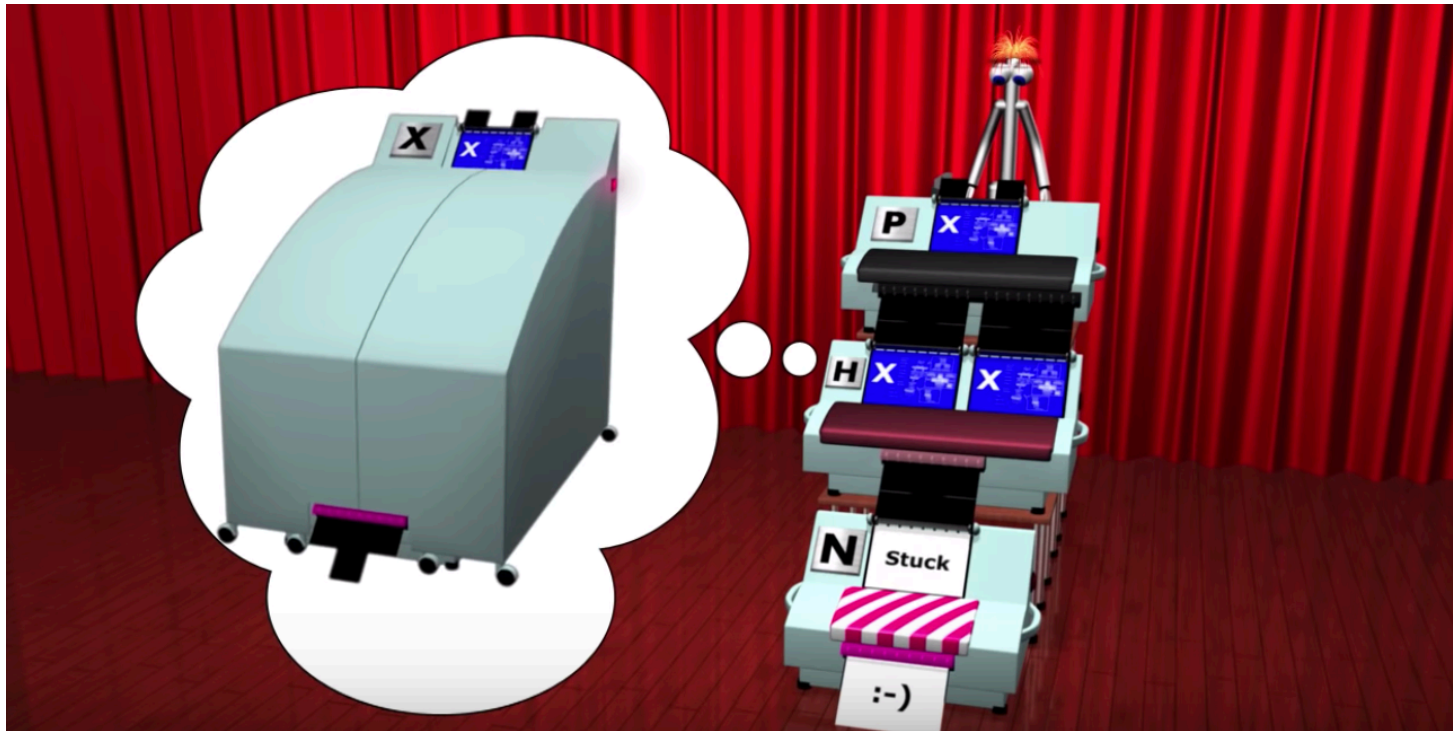If return 1, bug code never halts

What will happen to Bug(Bug)?

HALT$_{TM}$

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

HALT_{TM}

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

$\langle M \rangle$ → TM R → accept / reject

Theorem 5.2

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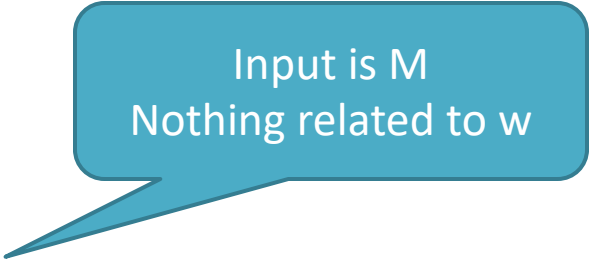
- Proof:

  We need create contradiction between

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

  Input is M
  Nothing related to w

  and

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

Theorem 5.2 proof

- Proof:
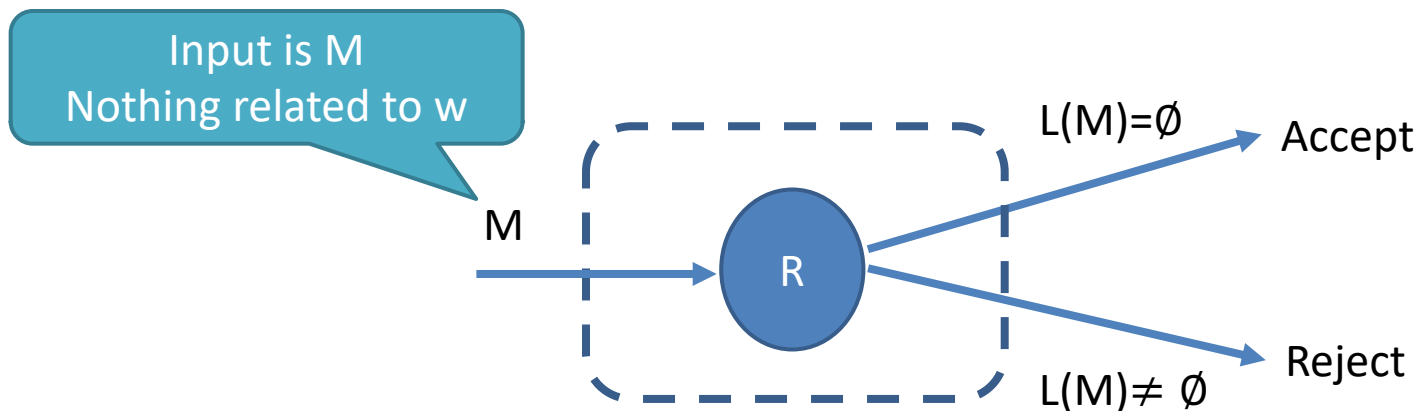
  Suppose E_{TM} is decidable, then TM R decides E_{TM}

  R = "On input M,

       if M does accept anything, then L(M)=∅, R accept;

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

       "

Input is M
Nothing related to w

M

R

L(M)=∅ → Accept

L(M)≠ ∅ → Reject

Theorem 5.2 proof

- Proof
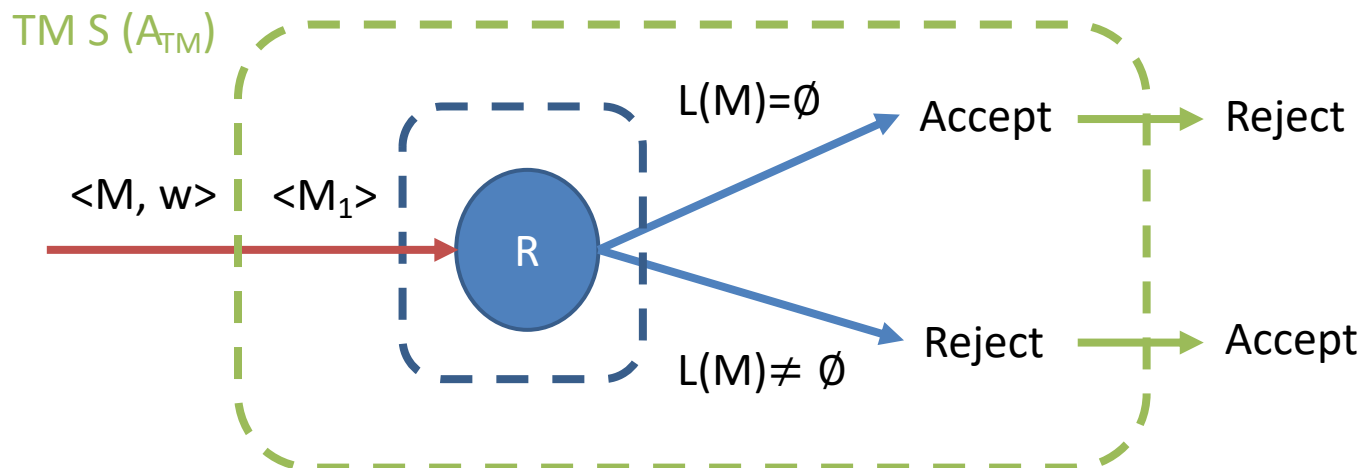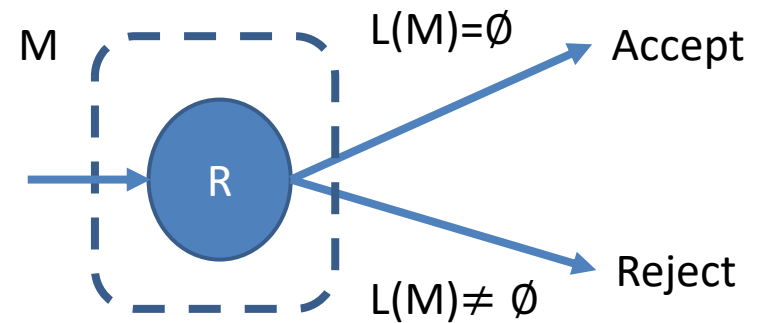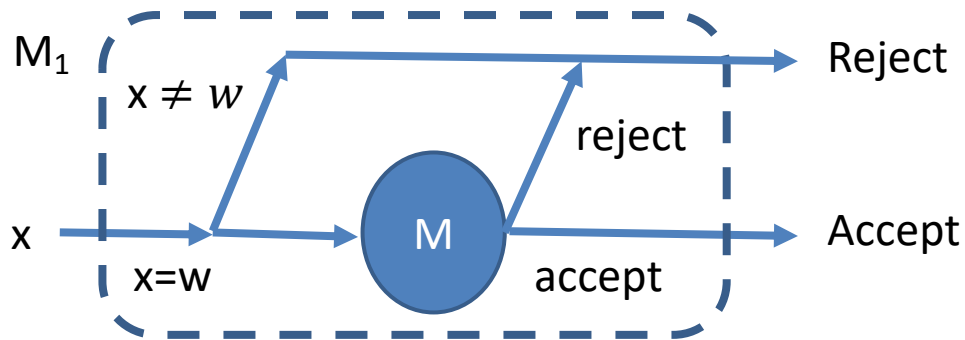
Create a TM M_1, that

> Create a TM M_1 involving both M and w

- ▸ 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

M_1

$x \neq w$ Reject

reject
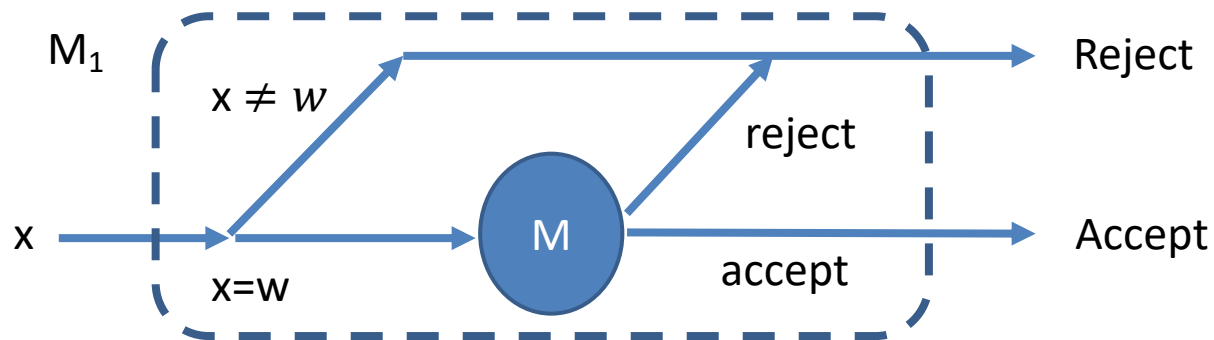
x   M   Accept

x=w   accept

Theorem 5.2 proof

- # Proof

Integrate the R and M_1

R = "On input M_1,

    if M does accept w, then L(M)=∅, R accept;

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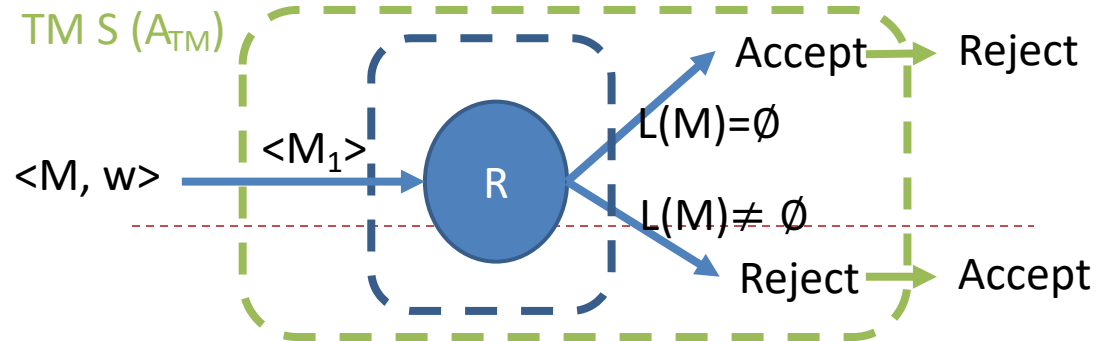
    "

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; ── M rejects w ⇒ R accepts ⇒ S rejects

    If R rejects, S accepts. ── M accepts w ⇒ R rejects ⇒ S accepts

    ",

Contradiction! A_{TM} is not decidable

TM S (A_{TM})

<M, w> → <M_1> → R
    Accept → Reject   L(M)=∅
    L(M)≠ ∅   Reject → Accept

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

  o Whether a given Turing machine has an equivalent finite automaton or recognizes a regular language

- Language

  o $REGULAR_{TM} = \{\langle M \rangle \mid M$ is a TM

  $$and$$

  $L(M)$ is a regular language$\}$.

Theorem 5.3

- REGULAR$_{TM}$ is undecidable.

  ○ REGULAR$_{TM}$ = {⟨M⟩| M is a TM and L(M) is a regular language}.
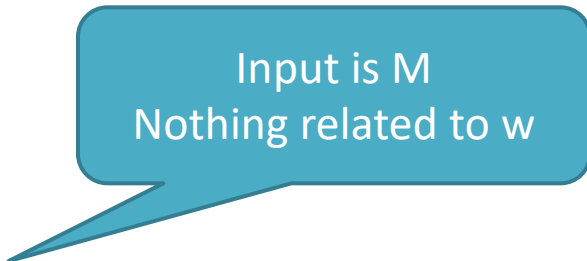
- Proof:

  We need create contradiction between

  > Input is M
  > Nothing related to w

       REGULAR$_{TM}$ = {⟨M⟩| M is a TM and L(M) is a regular language}.

  and

       A$_{TM}$ = {⟨M, w⟩| M is a TM and M accepts w}

Theorem 5.3 proof

Input is M
Nothing related to w

- Proof:
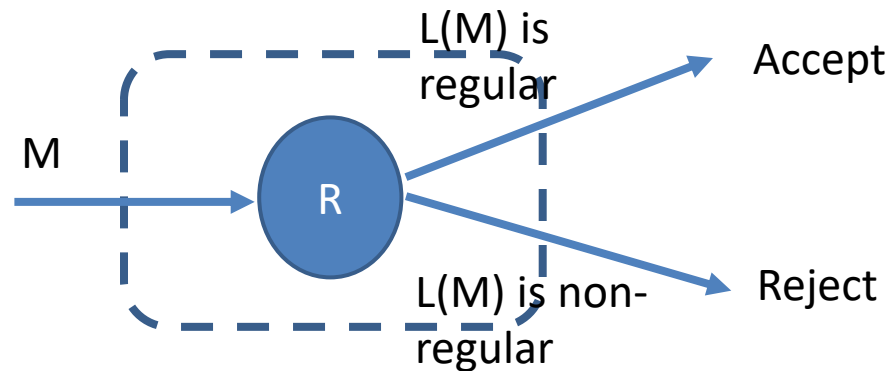
Suppose REGULAR$_{TM}$ is decidable, then TM R decides REGULAR$_{TM}$

R = "On input M,

if L(M) is not a regular language, R rejects;

if L(M) is a regular language, R accepts;

"

M → R
L(M) is regular → Accept
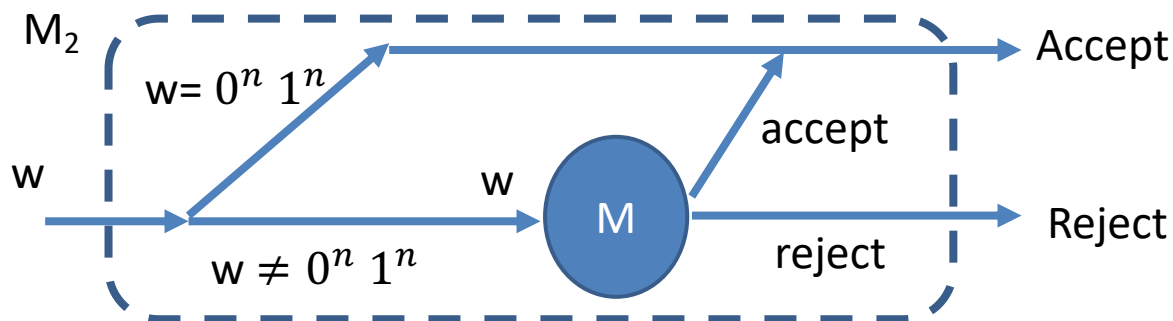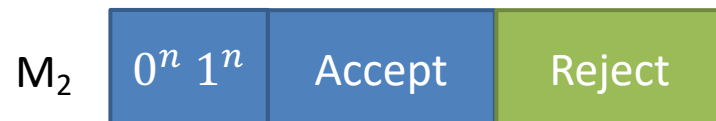L(M) is non-regular → Reject

Theorem 5.3 proof

- Proof

Create a TM M_2, for input w:

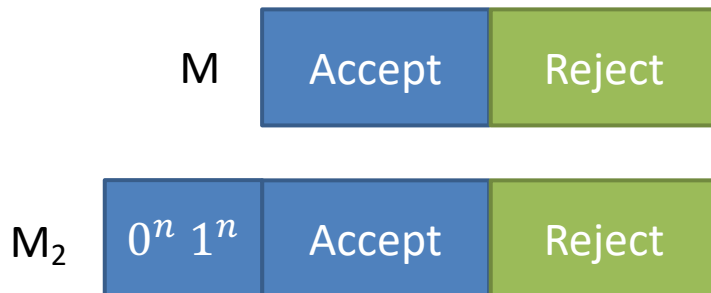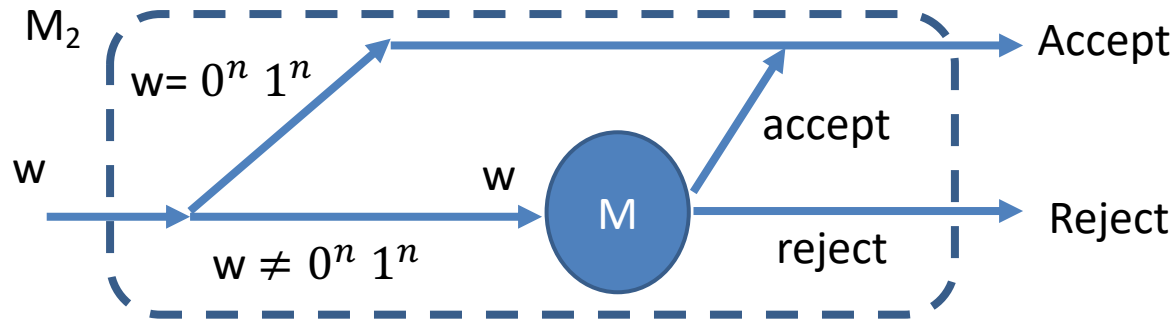If w is in format of $0^n 1^n$ (not regular language), accept;

If w is not in that format, run w on M:

If M accepts w, M_2 accepts;

otherwise, rejects.

Theorem 5.3 proof

$$L(M_2) = \begin{cases} regular, & if\ M\ accept\ w \\ 0^n\ 1^n, & if\ M\ does\ not\ accept\ w \end{cases}$$

Theorem 5.3 proof

$$L(M_2) = \begin{cases} regular, & if\ M\ accept\ w \\ 0^n\ 1^n, & if\ M\ does\ not\ accept\ w \end{cases}$$
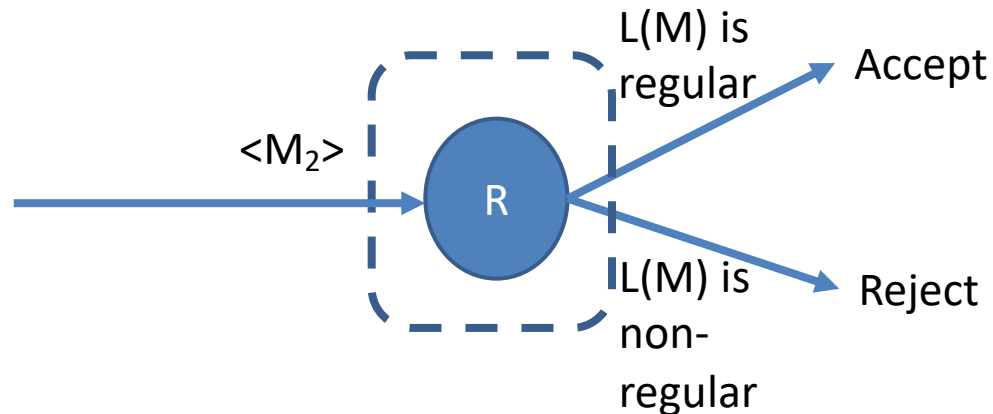
- Proof

Integrate the R and M_2

R = "On input M_2,

    if M does not accept w, then $L(M_2)$ is non-regular language ($0^n\ 1^n$), R rejects;

    if M accepts w, then $L(M_2)$ is a regular language, R accepts;

    "

Theorem 5.3 proof
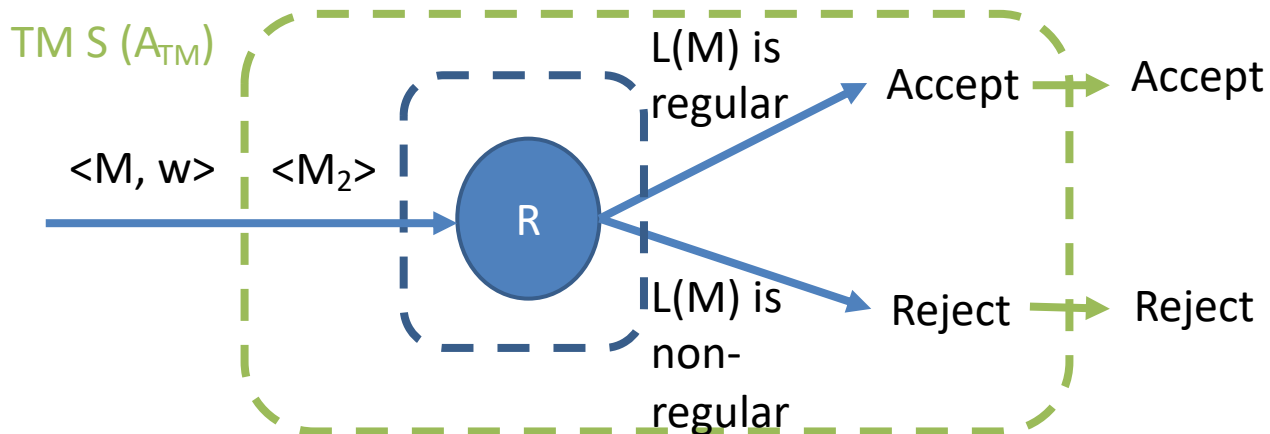
Based on R, we can create TM S to decide A_{TM}

S = "On input <M, w>, convert <M, w> into M_2

    Run R on input M_2

    If R accepts, S accepts;

       M accepts w $\Rightarrow$ R accepts $\Rightarrow$ S accepts

    If R rejects, S rejects. ",

Contradiction!
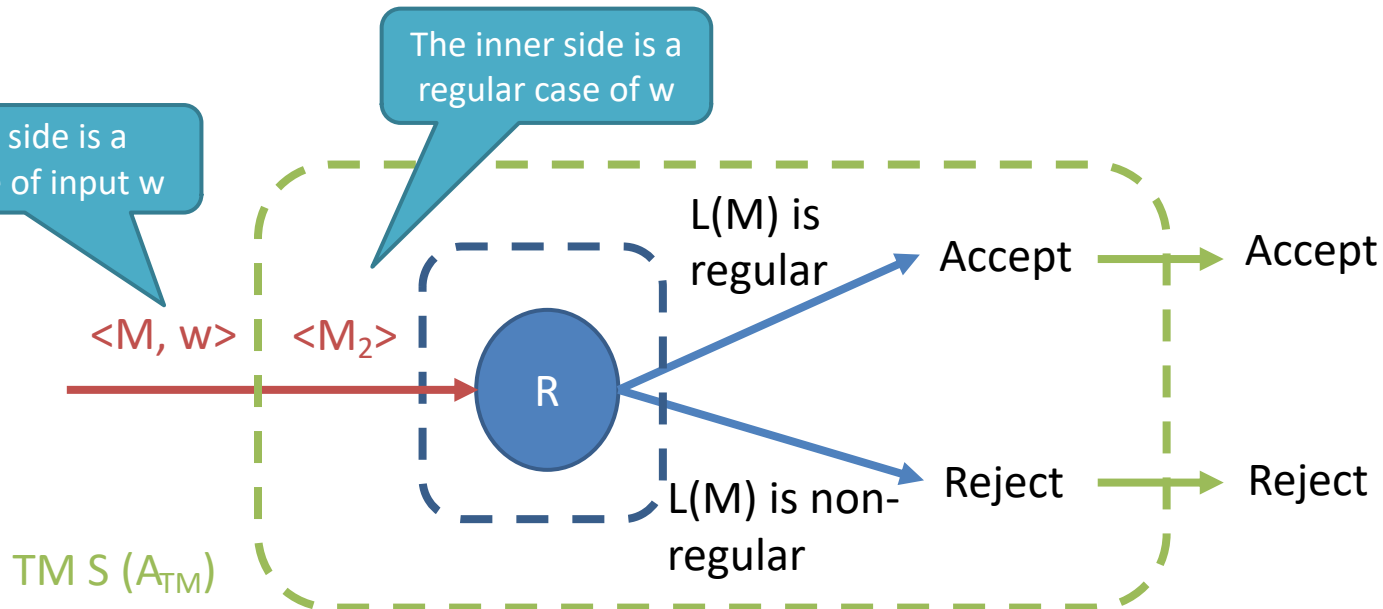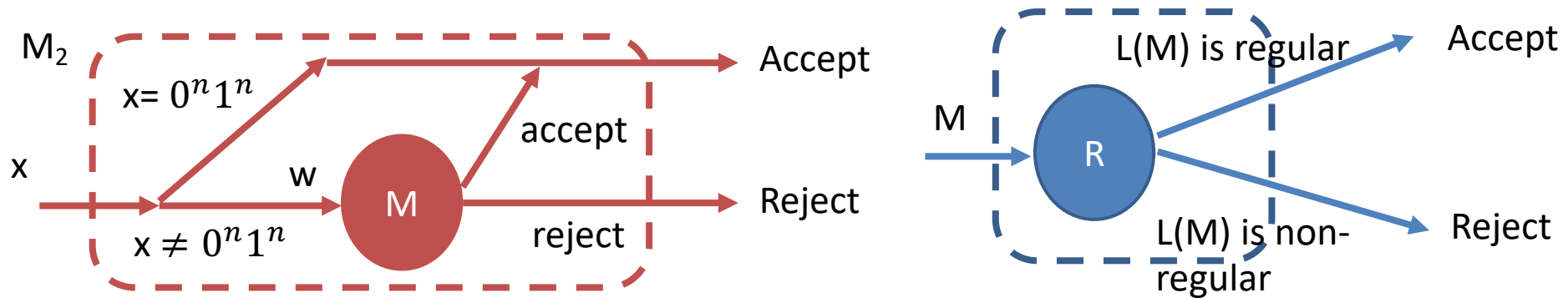
       M rejects w $\Rightarrow$ R rejects $\Rightarrow$ S rejects

Theorem 5.3 proof

M_2

$x= 0^n 1^n$

$x \neq 0^n 1^n$

x

w

accept

reject

M

Accept

Reject

L(M) is regular

L(M) is non-regular

M

R

Accept

Reject

The inner side is a regular case of w

The outer side is a general case of input w

<M, w>

<M_2>

R

L(M) is regular

L(M) is non-regular

Accept

Reject

Accept

Reject

TM S (A_{TM})

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

    $L(M_1) = L(M_2)\}$

$\langle M_1, M_2 \rangle$ → [TM R] → accept

[TM R] → reject

Theorem 5.4

- EQ$_{TM}$ is undecidable.

  ○ EQ$_{TM}$ = {⟨M$_1$,M$_2$⟩| M$_1$ and M$_2$ are TMs and L(M$_1$) = L(M$_2$)}

- Proof:

  We need create contradiction between

  $$EQ_{TM} = \{⟨M_1,M_2⟩| M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$$

  and

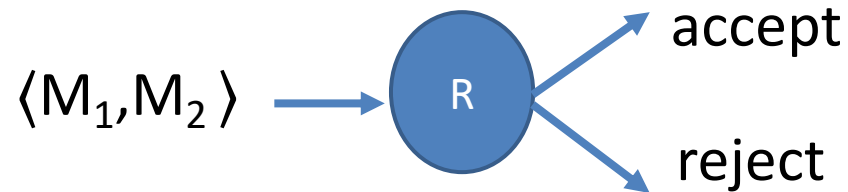 $$E_{TM} = \{⟨M⟩ | M \text{ is a TM and } L(M) = \emptyset \}$$

Theorem 5.4 proof

- Proof:

  Suppose EQ_{TM} is decidable, TM R decide EQ_{TM}

  R = "On input $<M_1, M_2>$,

    if $L(M_1) = L(M_2)$, accept;

    if $L(M_1) \neq L(M_2)$, reject."

  $\langle M_1, M_2 \rangle \longrightarrow$ R $\nearrow$ accept $\searrow$ reject

  Create another TM S for E_{TM}

  S = "On input $<M>$,

  $L(M_a) = \emptyset$

    Execute R on input $<M, M_a>$, M_a is a TM rejects all inputs,

    if R accepts, S accepts;

    if R rejects, S rejects."

Theorem 5.4 proof

- Proof:

$\langle M_1, M_2 \rangle \longrightarrow$ R $\longrightarrow$ accept

$\longrightarrow$ reject

$L(M_2) = \emptyset$

$\langle M \rangle \longrightarrow \langle M, M_2 \rangle \longrightarrow$ R $\longrightarrow$ accept $\longrightarrow$ accept

$\longrightarrow$ reject $\longrightarrow$ reject

S

S is decidable for E_{TM}
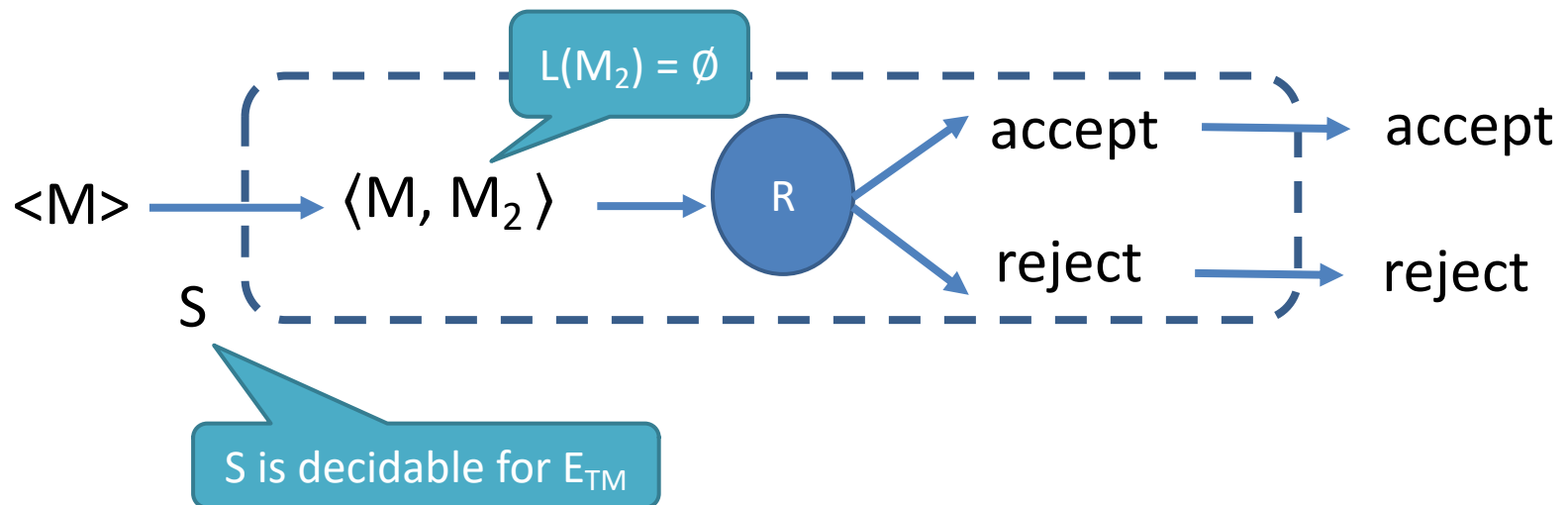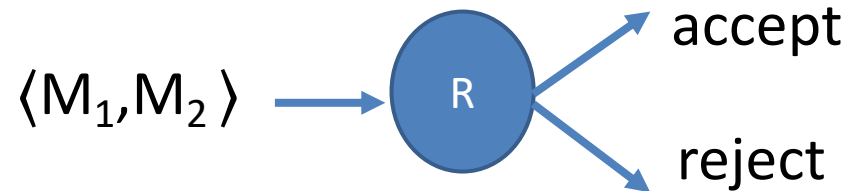
Theorem 5.4 proof

- Proof:

S = "On input <M>,

Execute R on input <M, M_a>, M_a is a TM rejects all inputs,

if L(M) = ∅, S accepts;

if L(M) ≠ ∅, S rejects."

Because R id decidable, then S is also decidable.

However, for E_{TM} is not decidable (based on theorem 5.2).

Contradiction!

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

$$A_{TM}$$

$$HALT_{TM}$$

$$E_{TM}$$

$$REGULAR_{TM}$$

$$EQ_{TM}$$