CS 6041
Theory of Computation

Decidability

Kun Suo

Computer Science, Kennesaw State University

https://kevinsuo.github.io/

Decidable problems concerning regular languages

- Acceptance problem for DFAs
  - whether a DFA accepts a string
- Acceptance problem for NFAs
  - whether a NFA accepts a string
- Regular expression decidability
  - Whether a regular expression generates a string
- Emptiness testing for DFAs
  - Whether a DFA is empty
- Equivalence of DFAs
  - Whether two DFAs recognize the same language

A_{DFA}

E_{DFA}

EQ_{DFA}

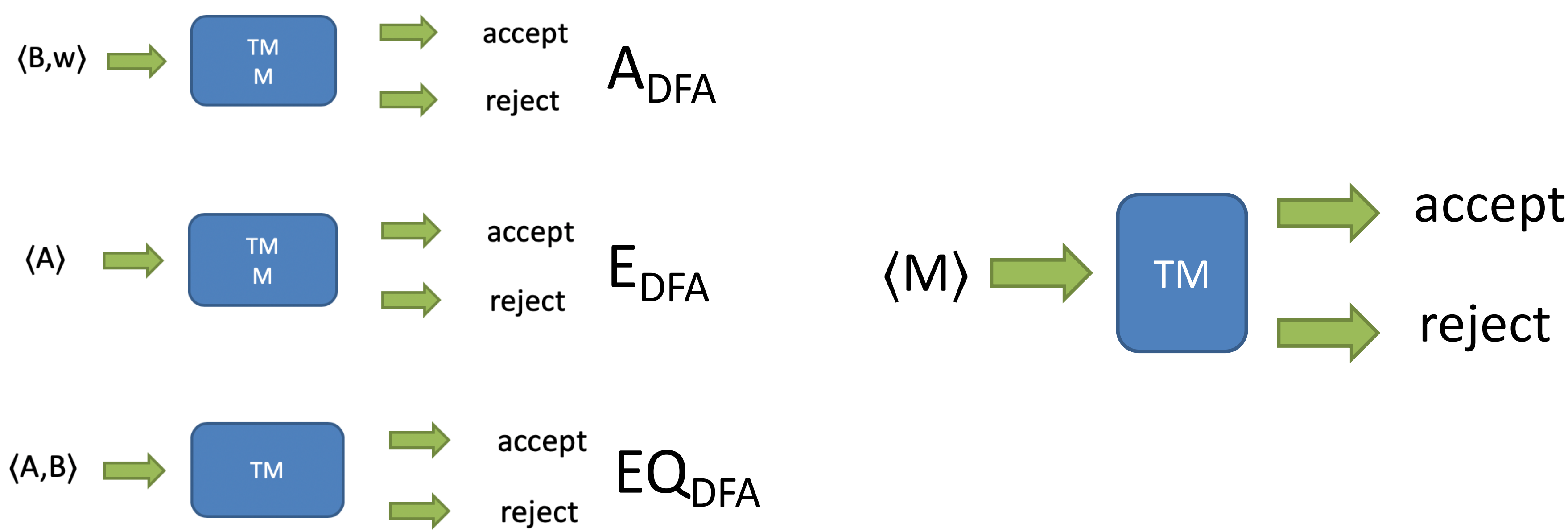
Decidability

- Decidable?

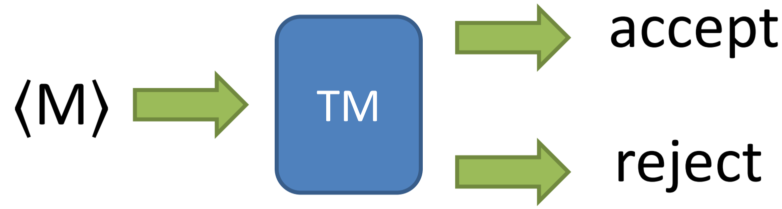
	DFA/NFA/RE	CFG	TM
Acceptance (A)	√		
Emptiness (E)	√		
Equivalence (EQ)	√		

Question

- Prove: A = {<M> | M is a DFA that doesn't accept any string containing an odd number of 1s}. Show that A is decidable.



Question



- Prove: A = {<M> | M is a DFA that doesn't accept any string containing an odd number of 1s}. Show that A is decidable.

X = "On input <M> where M is a DFA:

1, construct a DFA O that accepts any string contains an odd number of 1s

2, construct a DFA B such that L(B)=L(M) ∩ L(O)

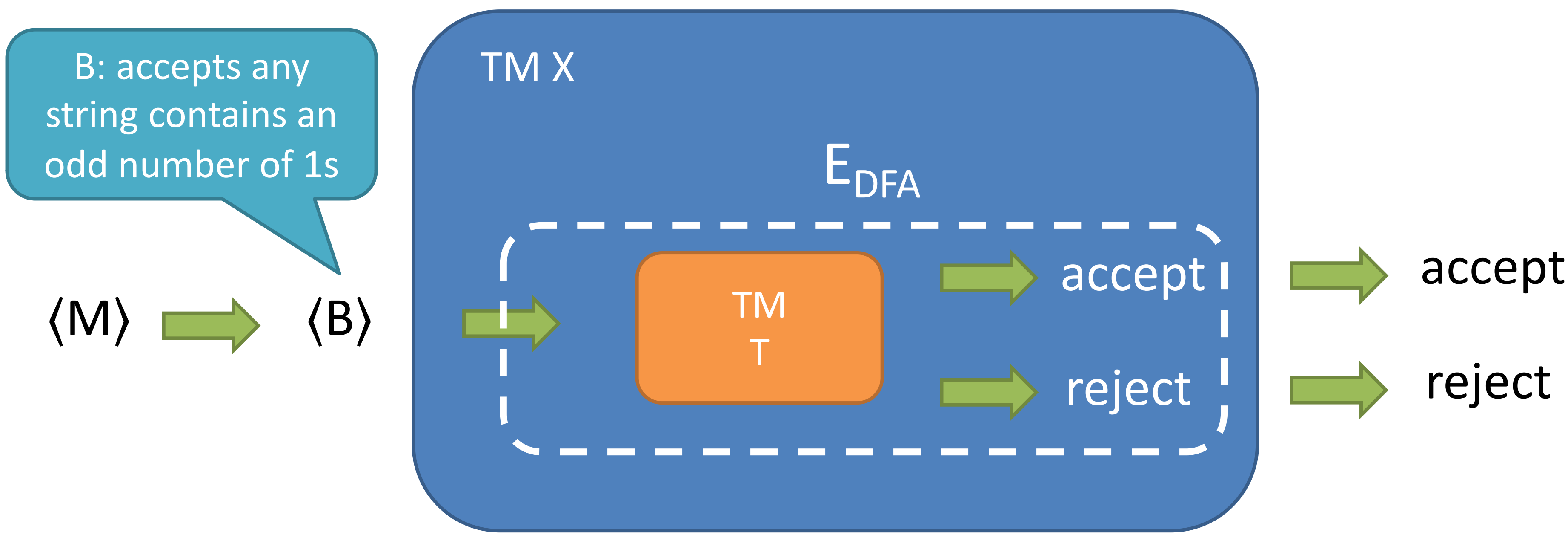
3, run TM T from E_{DFA} on input <B>

4, if T accepts, X accepts; if T rejects, X accepts.

"

Question

- Prove: A = {<M> | M is a DFA that doesn't accept any string containing an odd number of 1s}. Show that A is decidable.



Decidable problems concerning CFL/CFGs

- CFG generation decidability
  - Whether a CFG generates a particular string

A_{CFG}

- Emptiness testing for CFGs
  - Whether a CFG is empty

E_{CFG}

- Equivalence of CFGs
  - Whether two CFGs recognize the same language

EQ_{CFG}

- CFL decidability
  - Whether a CFL is decidable

Decidability

- Decidable?

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

Decidability

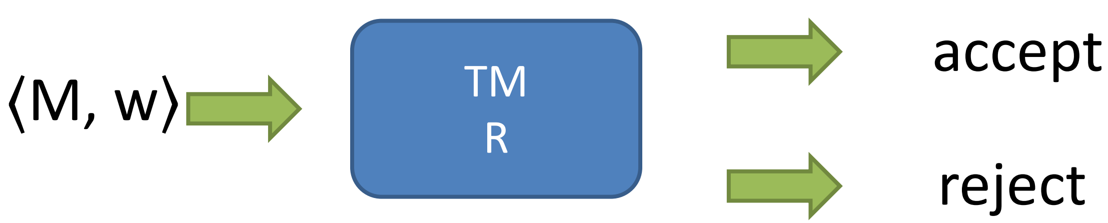
- Decidable?

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

Decidable problems for Turing Machine

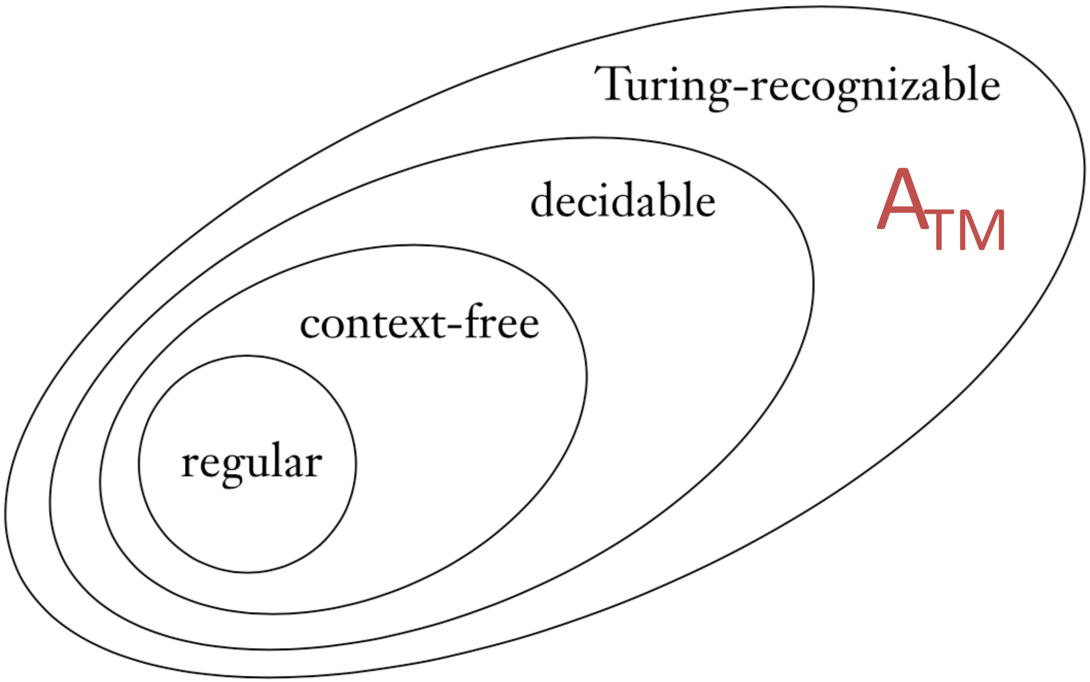
- Acceptance problem for Turing Machine
  - Whether a Turing machine accepts a given input string

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



Theorem 4.11

- A_{TM} is undecidable



Theorem 4.11 proof

- A_{TM} is undecidable

- Proof idea:

  Use M as the input string

  A_{TM}={ <M,w> | TM M accept string w }

  D_{TM}={ <M, <M>> | TM M accept string <M> }

  D_{TM} is a special case of A_{TM}

  If D_{TM} is undecidable, then A_{TM} must be undecidable

Theorem 4.11 proof details

- Proof by contradiction:

Suppose language A_{TM} is decidable, then

There exists a TM H can decide A_{TM}

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

Theorem 4.11 proof details

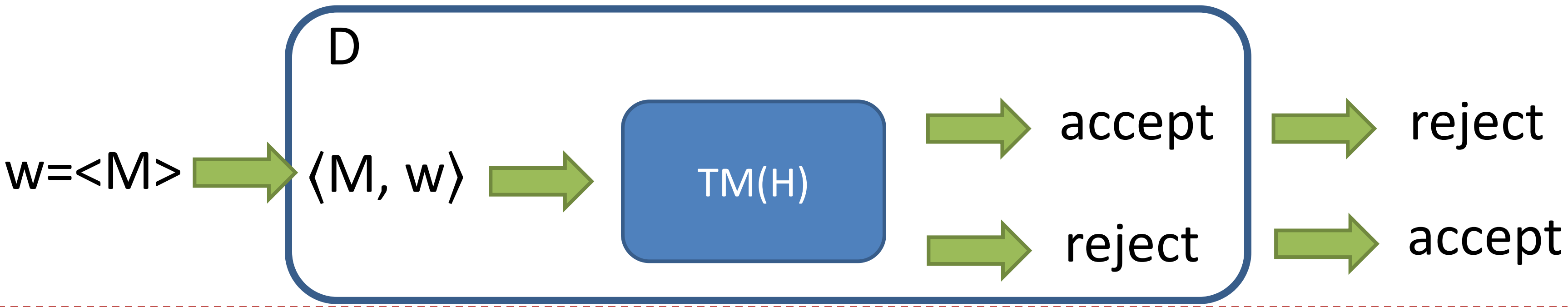
Create TM D, D="On input ⟨M⟩, where M is a TM:

(1) Run H on input <M, <M>>

(2) If H accepts, D reject;

if H rejects, D accept."

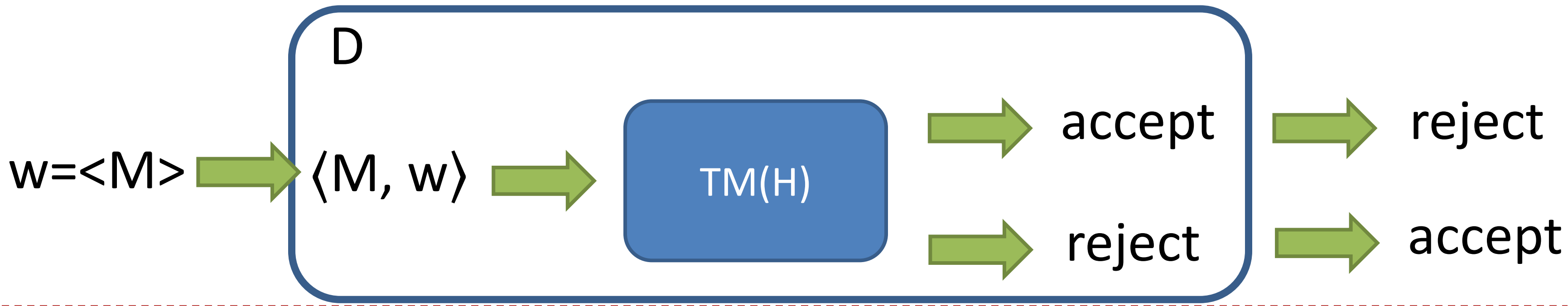
$$D(<M>) = \begin{cases} \text{accept,} & \text{if M does not accept} < M > \\ \text{reject,} & \text{if M accepts} < M > \end{cases}$$



Theorem 4.11 proof details

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

For TM D, what will happen when input is <D>?



Theorem 4.11 proof details

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

For TM D, what will happen when input is <D>?

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

Then we have *D(<D>)=accept* and *D(<D>)=reject* at the same time. Contradiction!

Theorem 4.11 proof details

- Proof by contradiction:

  Suppose language A_{TM} is decidable, then

  There exists a TM H can decide A_{TM}

- Suppose is wrong, thus A_{TM} is undecidable

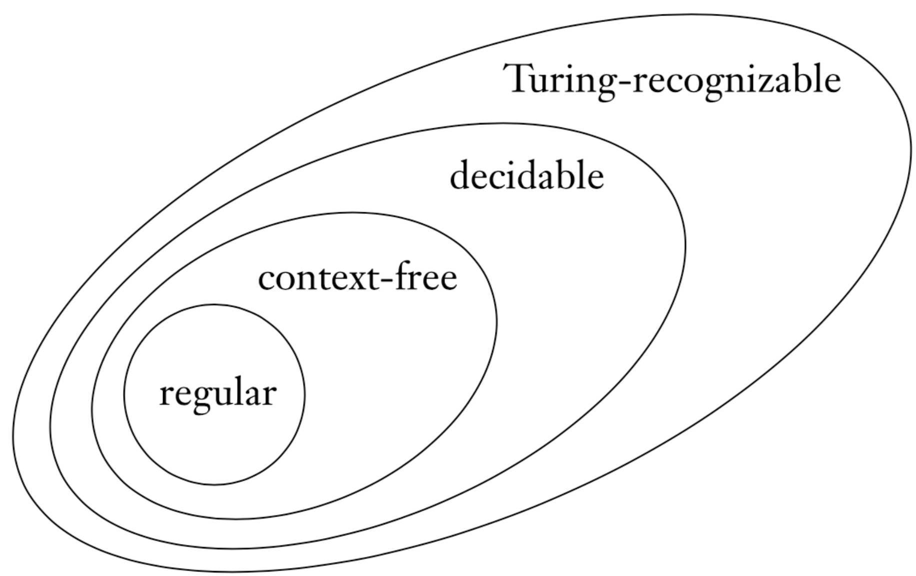
Theorem 4.11

- A_{TM} is undecidable

- In other words, we do not know whether a Turing machine accepts a given input string

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

Loop = Never Halt



Explanation

- A_{TM} is undecidable

- Explanation by using ***diagonalization method***

Suppose language A_{TM} is decidable, then

There exists a TM H can decide A_{TM}

Results of H(<M, w>)

Because TM H can **decide** A_{TM} , so the result of H(M, w) is either accept or reject

M_1							
M_2							
M_3							
M_4							
M_5							
M_6							
⋮							

Results of H(<M, w>)

Because TM H can **decide** A_{TM} , so the result of H(M, w) is either accept or reject

	$\langle w_1 \rangle$	$\langle w_2 \rangle$	$\langle w_3 \rangle$	$\langle w_4 \rangle$	$\langle w_5 \rangle$	$\langle w_6 \rangle$	…
M_1							
M_2							
M_3							
M_4							
M_5							
M_6							
⋮							

Results of H(<M, w>)

Because TM H can **decide** A_{TM} , so the result of H(M, w) is either accept or reject

	$\langle w_1 \rangle$	$\langle w_2 \rangle$	$\langle w_3 \rangle$	$\langle w_4 \rangle$	$\langle w_5 \rangle$	$\langle w_6 \rangle$	$\cdots$
M_1	accept	reject	accept	reject	accept	accept	$\cdots$
M_2	reject	accept	reject	reject	accept	reject	$\cdots$
M_3	reject	reject	reject	reject	reject	reject	$\cdots$
M_4	accept	reject	accept	reject	accept	reject	$\cdots$
M_5	accept	accept	accept	accept	accept	accept	$\cdots$
M_6	reject	accept	reject	reject	reject	accept	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Results of H(<M, <M>>)

Just set w as <M>

Because TM H can **decide** A_{TM} , so the result of H(M, w) is either accept or reject

This is H(<M, <M>>)

The result does not change

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	$\langle M_6 \rangle$	$\cdots$
M_1	accept	reject	accept	reject	accept	accept	$\cdots$
M_2	reject	accept	reject	reject	accept	reject	$\cdots$
M_3	reject	reject	reject	reject	reject	reject	$\cdots$
M_4	accept	reject	accept	reject	accept	reject	$\cdots$
M_5	accept	accept	accept	accept	accept	accept	$\cdots$
M_6	reject	accept	reject	reject	reject	accept	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Results of D(<M>) = opposite of H(<M, <M>>)

Because TM H can **decide** A_{TM} , so the result of H(M,w) is either accept or reject

This is the opposite of H(<M, <M>>)

	<M_1>			M_4>	<M_5>	<M_6>	⋯
M_1	reject	reject	accept	reject	accept	accept	⋯
M_2	reject	reject	reject	reject	accept	reject	⋯
M_3	reject	reject	accept	reject	reject	reject	⋯
M_4	accept	reject	accept	accept	accept	reject	⋯
M_5	accept	accept	accept	accept	reject	accept	⋯
M_6	reject	accept	reject	reject	reject	reject	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Results of D(<M>) = opposite of H(<M, <M>>)

H(<M, <M>>)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	$\langle M_6 \rangle$	⋯
M_1	accept	reject	accept	reject	accept	accept	⋯
M_2	reject	accept	reject	reject	accept	reject	⋯
M_3	reject	reject	reject	reject	reject	reject	⋯
M_4	accept	reject	accept	reject	accept	reject	⋯
M_5	accept	accept	accept	accept	accept	accept	⋯
M_6	reject	accept	reject	reject	reject	accept	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

D(<M>) = opposite of H(<M, <M>>)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\langle M_5 \rangle$	$\langle M_6 \rangle$	⋯
M_1	reject	reject	accept	reject	accept	accept	⋯
M_2	reject	reject	reject	reject	accept	reject	⋯
M_3	reject	reject	accept	reject	reject	reject	⋯
M_4	accept	reject	accept	accept	accept	reject	⋯
M_5	accept	accept	accept	accept	reject	accept	⋯
M_6	reject	accept	reject	reject	reject	reject	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Results of D(<D>) ?

Because TM H can **decide** A_{TM} , so the result of H(M,w) is either accept or reject

	<M$_1$>	<M$_2$>	<M$_3$>	<M$_4$>	…	<D>	…
M_1	reject	reject	accept	reject	…	accept	…
M_2	reject	reject	reject	reject	…	reject	…
M_3	reject	reject	accept	reject	…	reject	…
M_4	accept	reject	accept	accept	…	reject	…
⋮	⋮	⋮	⋮	⋮	⋱	⋮	
D	reject	reject	accept	accept	…	?	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Results of D(<D>) ?

Diagonalization method

Then we have *D(<D>)=accept* and *D(<D>)=reject* at the same time. Contradiction!

	<M_1>	<M_2>	<M_3>	<M_4>	⋯	<D>	⋯
M_1	reject	reject	accept	reject	⋯	accept	⋯
M_2	reject	reject	reject	reject	⋯	reject	⋯
M_3	reject	reject	accept	reject	⋯	reject	⋯
M_4	accept	reject	accept	accept	⋯	reject	⋯
⋮	⋮	⋮	⋮	⋮	⋱	⋮	
D	reject	reject	accept	accept	⋯	?	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Decidability

- Decidable?

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

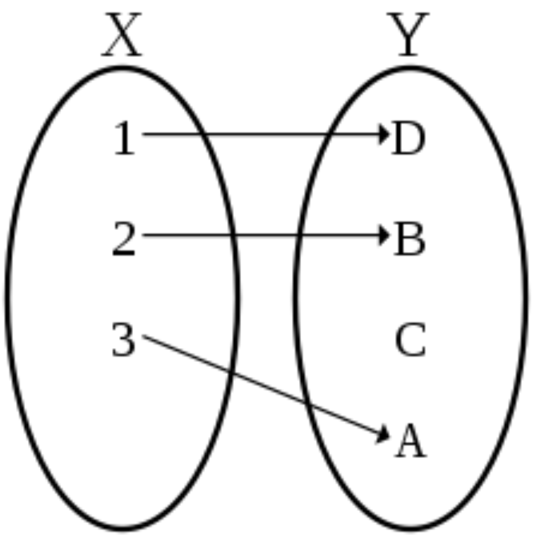
Countable

- A set is ***countable*** if either it is finite, or it has the same size as N.

- A = {1, 2, 3}

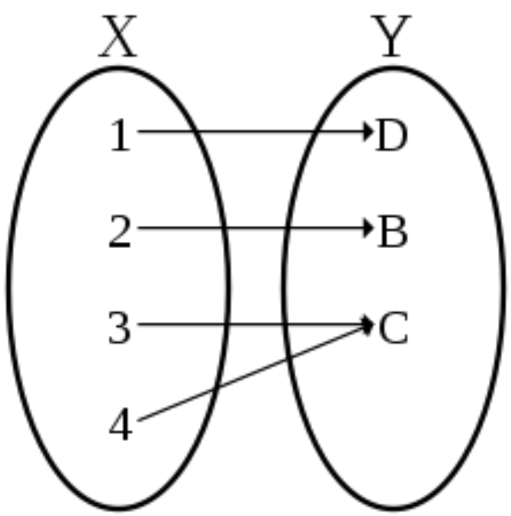
Set Element Relationship

- **One-to-one**: if different elements of source set is mapped to different elements of destination set.



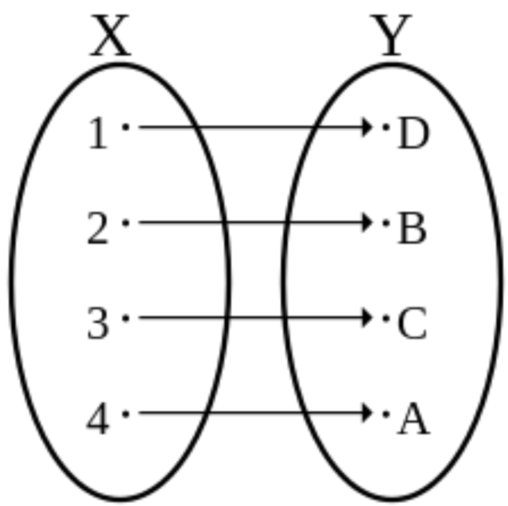
Set Element Relationship

- **Onto**: if different elements of destination set has at least one element mapped to it from the source set.



Set Element Relationship

- **correspondence**: Every element in the source set is mapped to a single element in the destination set; and vice verse.
- **Correspondence = one-to-one & onto**



Question: True or False

- Let X be the set {1,2,3,4,5} and Y be the set {6,7,8,9,10}. We describe the functions f: X→Y and g: X→Y in the following tables.
- f() is one-to-one

n	$f(n)$
1	6
2	7
3	6
4	7
5	6

n	$g(n)$
1	10
2	9
3	8
4	7
5	6

False. Because f(1) = f(3)

Question: True or False

- Let X be the set {1,2,3,4,5} and Y be the set {6,7,8,9,10}. We describe the functions f: X→Y and g: X→Y in the following tables.
- f() is onto

n	$f(n)$
1	6
2	7
3	6
4	7
5	6

n	$g(n)$
1	10
2	9
3	8
4	7
5	6

False. Not exist x in X letting f(x) = 10

Question: True or False

- Let X be the set {1,2,3,4,5} and Y be the set {6,7,8,9,10}. We describe the functions f: X→Y and g: X→Y in the following tables.
- g() is one-to-one

n	$f(n)$
1	6
2	7
3	6
4	7
5	6

n	$g(n)$
1	10
2	9
3	8
4	7
5	6

True.

Question: True or False

- Let X be the set {1,2,3,4,5} and Y be the set {6,7,8,9,10}. We describe the functions f: X→Y and g: X→Y in the following tables.
- g() is onto

n	$f(n)$
1	6
2	7
3	6
4	7
5	6

n	$g(n)$
1	10
2	9
3	8
4	7
5	6

True.

Question: True or False

- Let X be the set {1,2,3,4,5} and Y be the set {6,7,8,9,10}. We describe the functions f: X→Y and g: X→Y in the following tables.
- g() is correspondence

n	$f(n)$
1	6
2	7
3	6
4	7
5	6

n	$g(n)$
1	10
2	9
3	8
4	7
5	6

True. Because g is both one-to-one and onto.

Countable

- A set is ***countable*** if either it is finite, or it has the same size as N or subset of N (correspondence relationship).

- Mapping - -> Size of infinite set
  - f(n) = n

- A = {1,2,3, …}

n	f(n)
1	1
2	2
3	3
…	…
n	n

Countable

- A set is ***countable*** if either it is finite, or it has the same size as N or subset of N (correspondence relationship).

- Mapping - -> Size of infinite set
  - f(n) = 2n

- B = {2,4,6, …}

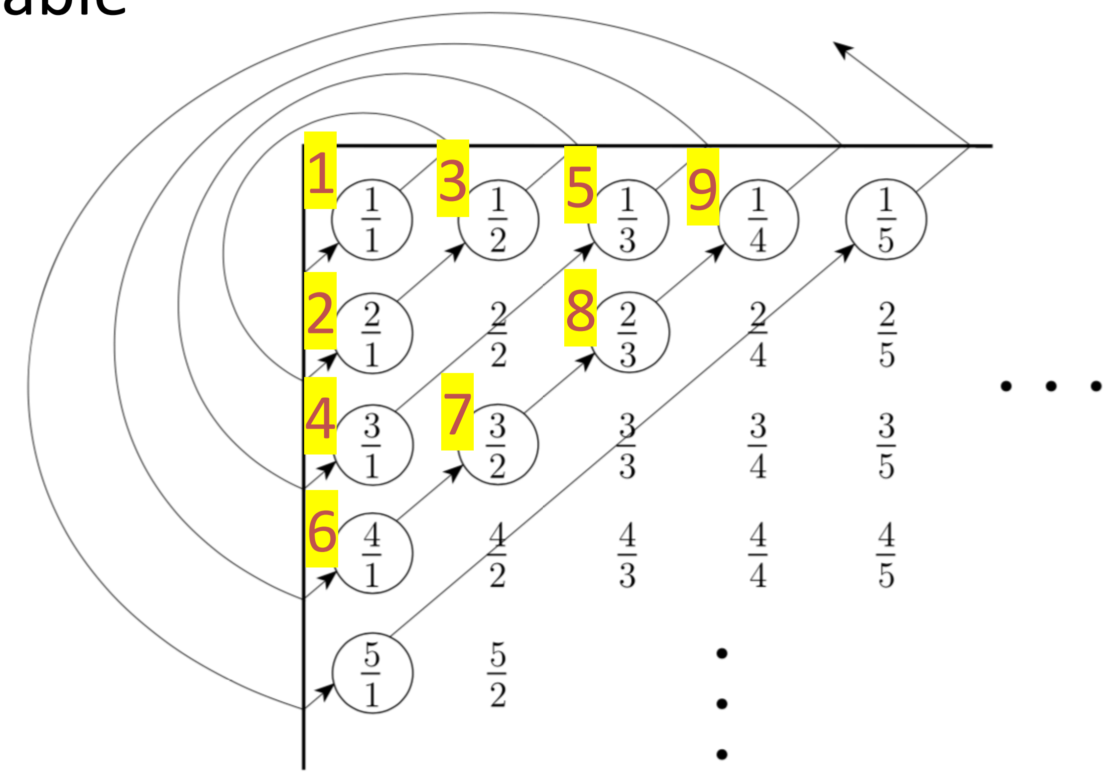
n	f(n)
1	2
2	4
3	6
…	…
n	2n

Countable and Diagonalization method

- Q = { $\frac{m}{n}$ | m,n ∈ N} be the set of positive rational numbers, Q is countable

- A mapping between of N and Q (prove by construction)

k --> $\frac{m}{n}$



Uncountable

- Theorem: R is uncountable

- Proof by construction:

  Suppose R is countable, then there exist a mapping f between N and R

  Let f(1) = 3.14159..., f(2) = 55.55555..., f(3) = ...,

n	$f(n)$
1	3.14159...
2	55.55555...
3	0.12345...
4	0.50000...
$\vdots$	$\vdots$

Uncountable

- Proof:

  Then we construct a value x:

  the ith digit of x is different than that in $f(n)$

n	$f(n)$
1	3.14159...
2	55.55555...
3	0.12345...
4	0.50000...
⋮	⋮

  x =0.4641...

n	$f(n)$
1	$3.\underline{1}4159\ldots$
2	$55.5\underline{5}555\ldots$
3	$0.12\underline{3}45\ldots$
4	$0.500\underline{0}0\ldots$
⋮	⋮

  for each n, and x,

  $$x \notin f(n)$$

  So there is no mapping between N and R

Countable vs. Uncountable

- To prove a set is countable
  - Finite or find a f(n)

- To prove a set is uncountable
  - Prove by construction that no f(n) exists

Question: True or False?

- Odd number set (e.g., {1,3,5, …}) is countable.

True.

Mapping - -> Size of infinite set

$f(n) = 2n-1$

n	f(n)
1	1
2	3
3	5
…	…
n	2n-1

Question: True or False?

- Integer number set Z (e.g., {…, -2,-1,0,1,2 …}) is countable.

True.

Mapping Z → N

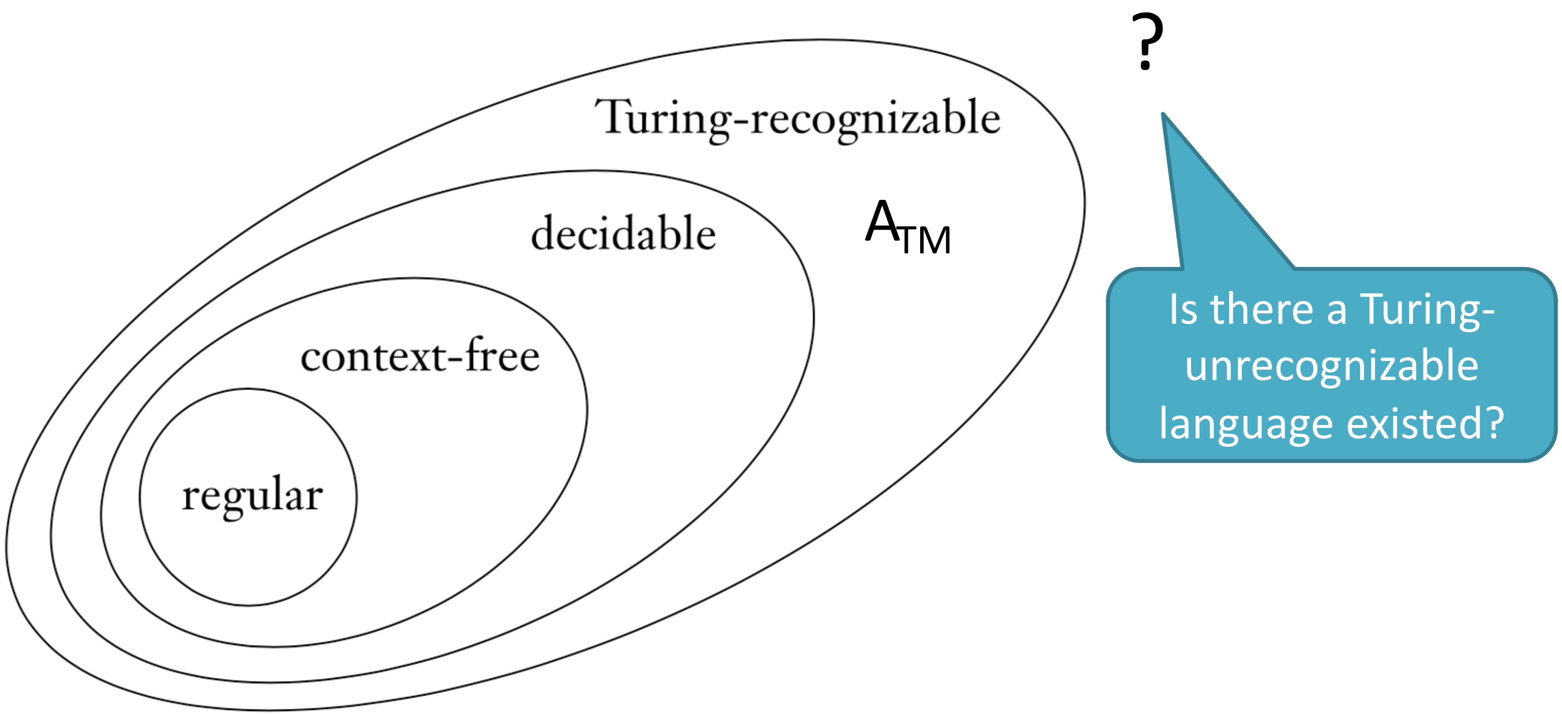
$f(n) = 2n$, if $n>=0$

$f(n) = -1-2n$, if $n < 0$

Z	N
-k	2k-1
…	…
-2	3
-1	1
0	0
1	2
…	…
k	2k

Review of Theorem 4.11

- A_{TM} is undecidable



Theorem 4.22

- Complement of A: $\overline{A}$
  - $\overline{A} = \Sigma^* - A$

- Theorem 4.22
  - A is decidable $\Leftrightarrow$ A and $\overline{A}$ are Turing-recognizable

Operation on languages

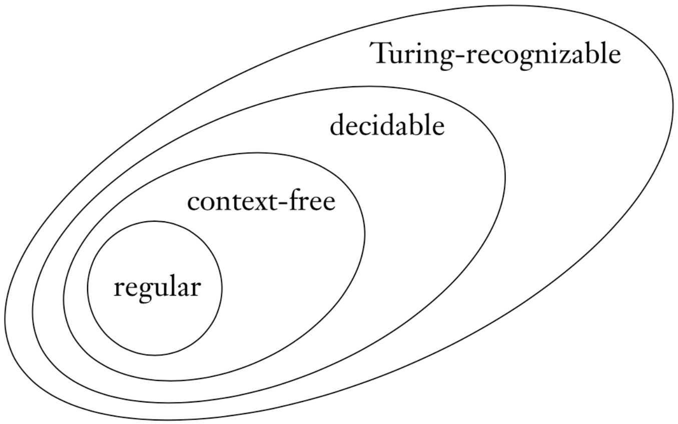
	RL: DFA/NFA/RE	CFL: CFG/PDA	TM-decidable
Union	close	close	close
Concatenation	close	close	close
Intersection	close	not close	close
Star	close	close	close
Complement	close	not close	close
Boolean operation	close	/	close

A is decidable $\Longrightarrow$ A and $\overline{\mathrm{A}}$ are Turing-recognizable

Proof:

If A is decidable, as the operation on decidable language is close, thus $\overline{\mathrm{A}}$ is also decidable

Because all Turing-decidable languages are Turing-recognizable, therefore, A and $\overline{\mathrm{A}}$ are Turing-recognizable



A is decidable ⇐ A and $\overline{A}$ are Turing-recognizable

Proof:

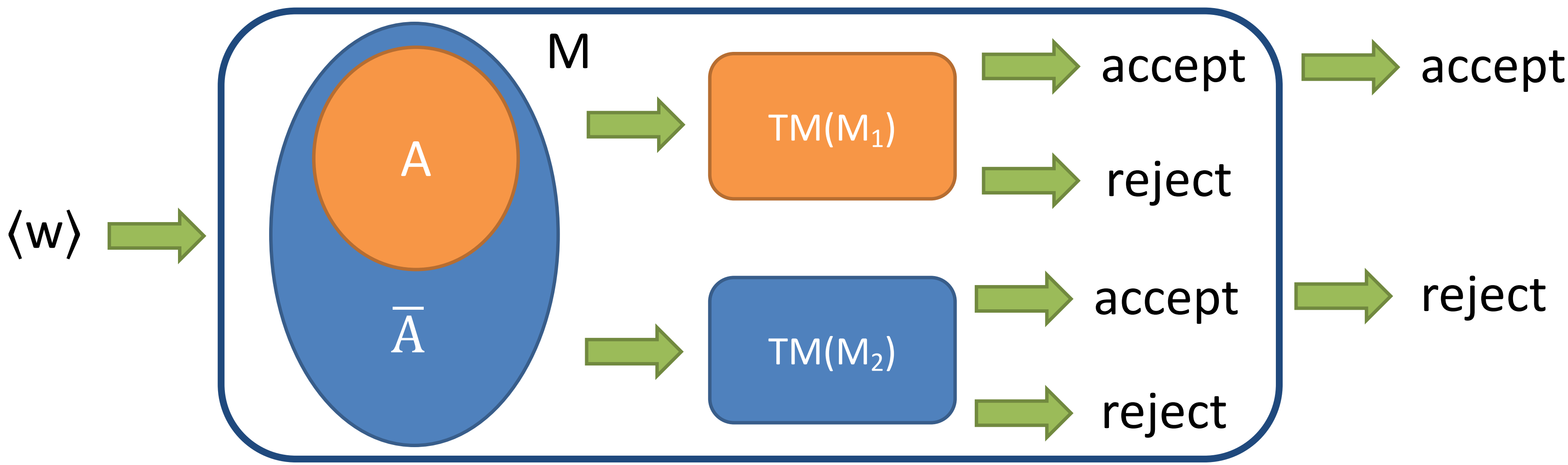
If A and $\overline{A}$ are Turing-recognizable. Let M_1 is recognizer TM of A and M_2 is recognizer TM of $\overline{A}$. Create a TM M as a decider for A,

M = "On input w:

Run both M_1 and M_2 on input w in parallel.

If M_1 accepts, accept;

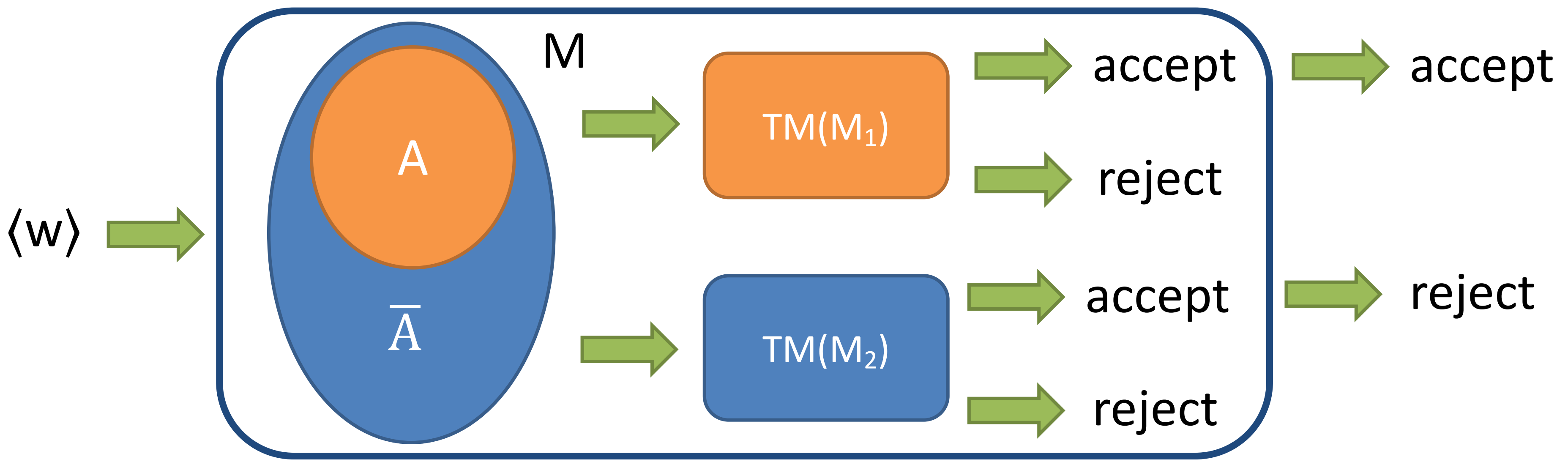
if M_2 accepts, reject."



Theorem 4.22 proof

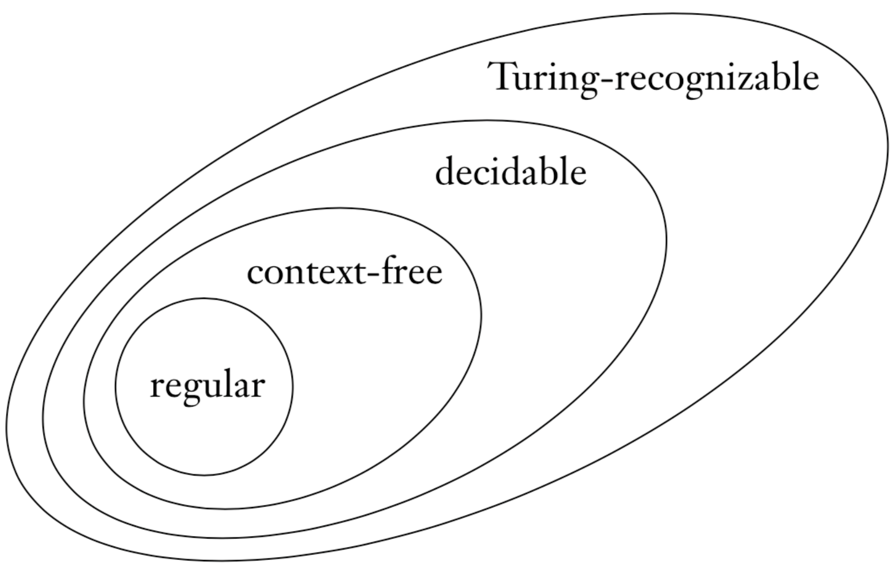
Because for each string w, it is either in A or $\overline{A}$. Thus for M_1 and M_2, one TM must accept w. When M_1 or M_2 accepts w, M will halt

Also, because M accepts all strings in A (for M_1) and reject all strings not in A ($\overline{A}$ for M_2). Thus, A is decidable



Corollary 4.23

- Corollary 4.23
  - $\overline{A_{TM}}$ is not Turing-recognizable
  - In other words, is there a language that TM cannot recognize?



$\overline{A_{TM}}$

Corollary 4.23 proof

- Corollary 4.23: $\overline{A_{TM}}$ is not Turing-recognizable
  - $A_{TM} = \{\langle M,w \rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$ is not decidable

Proof:

Suppose $\overline{A_{TM}}$ is Turing-recognizable

because A_{TM} is Turing-recognizable (based on definition)

So A_{TM} is Turing-decidable (theorem 4.22)

However, A_{TM} is undecidable (theorem 4.11)

Contradiction.

Conclusion on decidability

- Decidable?

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

- Diagonalization method to prove a language is undecidable

- Non Turing-recognizable language $\overline{A_{TM}}$ exists