COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

March 30, 2023

Lecture 25: Undecidability

Recap

- Turing machines - definition (single tape, deterministic), examples.
- Languages - Turing recognizable vs Turing decidable
- Robustness: TMs are extemely robust
- Variants: k-tapes, doubly infinite tapes, Enumerators, NTMs, Queue machines, 2 stacks, counter machines,...

Recap

- Turing machines - definition (single tape, deterministic), examples.
- Languages - Turing recognizable vs Turing decidable
- Robustness: TMs are extemely robust
- Variants: k-tapes, doubly infinite tapes, Enumerators, NTMs, Queue machines, 2 stacks, counter machines,...
- Today: undecidability.

Turing recognizability vs Decidability

Definition

A language L is said to be Turing recognizable if there is a Turing machine M such that $\forall w \in L$, M has at least one accepting run on w. For words not in L

- the machine may run forever
- or may reach q_{rej}

Turing recognizability vs Decidability

Definition

A language L is said to be Turing recognizable if there is a Turing machine M such that $\forall w \in L$, M has at least one accepting run on w. For words not in L

- the machine may run forever
- or may reach q_{rej}

A language L is said to be Turing decidable if there is a Turing machine M such that for all $w \in \Sigma^*$, M halts on w and

- if $w \in L$, M has an accepting run on w.
- if $w \notin L$, all runs of M on w are rejecting runs.

Hierarchy of languages and devices

Regular $\subsetneq$ Context-free $\subsetneq$ Decidable $\subsetneq$ Turing Recongizable

Hierarchy of languages and devices

Regular $\subsetneq$ Context-free $\subsetneq$ Decidable $\subsetneq$ Turing Recongizable

DFA/NFA/2-DFA/2-NFA < NPDA < Algorithms < Semi-algorithms

- Regular and context-free languages are decidable.

Hierarchy of languages and devices

Regular $\subsetneq$ Context-free $\subsetneq$ Decidable $\subsetneq$ Turing Recongizable

DFA/NFA/2-DFA/2-NFA < NPDA < Algorithms < Semi-algorithms

- Regular and context-free languages are decidable.

$$A_M = \{\langle M, w\rangle \mid w \in L(M)\}$$

Hierarchy of languages and devices

Regular $\subsetneq$ Context-free $\subsetneq$ Decidable $\subsetneq$ Turing Recongizable

DFA/NFA/2-DFA/2-NFA < NPDA < Algorithms < Semi-algorithms

- Regular and context-free languages are decidable.

$$A_M = \{\langle M, w\rangle \mid w \in L(M)\}$$

$$E_M = \{\langle M\rangle \mid L(M) = \varnothing\}$$

Hierarchy of languages and devices

Regular $\subsetneq$ Context-free $\subsetneq$ Decidable $\subsetneq$ Turing Recongizable

DFA/NFA/2-DFA/2-NFA < NPDA < Algorithms < Semi-algorithms

- Regular and context-free languages are decidable.

$$A_M = \{\langle M, w\rangle \mid w \in L(M)\}$$

$$E_M = \{\langle M\rangle \mid L(M) = \varnothing\}$$

$$EQ_M = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$$

Hierarchy of languages and devices

Regular $\subsetneq$ Context-free $\subsetneq$ Decidable $\subsetneq$ Turing Recongizable

DFA/NFA/2-DFA/2-NFA < NPDA < Algorithms < Semi-algorithms

- Regular and context-free languages are decidable.

$$A_M = \{\langle M, w\rangle \mid w \in L(M)\}$$

$$E_M = \{\langle M\rangle \mid L(M) = \emptyset\}$$

$$EQ_M = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$$

- Reading exercise: Theorems 4.2-4.9 in Sipser's book.

Hierarchy of languages and devices

Regular $\subsetneq$ Context-free $\subsetneq$ Decidable $\subsetneq$ Turing Recongizable

DFA/NFA/2-DFA/2-NFA < NPDA < Algorithms < Semi-algorithms

- Regular and context-free languages are decidable.

$$A_M = \{\langle M, w \rangle \mid w \in L(M)\}$$

$$E_M = \{\langle M \rangle \mid L(M) = \varnothing\}$$

$$EQ_M = \{\langle M_1, M_2 \rangle \mid L(M_1) = L(M_2)\}$$

- Reading exercise: Theorems 4.2-4.9 in Sipser's book.
- Important: Encoding programs as data.

Turing machines as strings

Every TM can be represented as a string in $\{0, 1\}^*$.

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Every string over $\{0,1\}^*$ represents some TM.

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Every string over $\{0,1\}^*$ represents some TM.

If a string does not represent any TM, as per our encoding, let us assume that it represents a TM that does nothing.

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Every string over $\{0,1\}^*$ represents some TM.

If a string does not represent any TM, as per our encoding, let us assume that it represents a TM that does nothing.

Every TM is represented by infinitely many strings.

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Every string over $\{0,1\}^*$ represents some TM.

If a string does not represent any TM, as per our encoding, let us assume that it represents a TM that does nothing.

Every TM is represented by infinitely many strings.

Any encoding of TMs will have a null character, say 010101. Then for any string $\alpha \in \{0,1\}^*$, suppose it represents machine M then all strings of the form $(010101)^*\alpha$ also represent the same machine M.

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Every string over $\{0,1\}^*$ represents some TM.

If a string does not represent any TM, as per our encoding, let us assume that it represents a TM that does nothing.

Every TM is represented by infinitely many strings.

Any encoding of TMs will have a null character, say 010101. Then for any string $\alpha \in \{0,1\}^*$, suppose it represents machine M then all strings of the form $(010101)^*\alpha$ also represent the same machine M.

This has a similar effect as adding comments in the C program.

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Every string over $\{0,1\}^*$ represents some TM.

If a string does not represent any TM, as per our encoding, let us assume that it represents a TM that does nothing.

Every TM is represented by infinitely many strings.

Any encoding of TMs will have a null character, say 010101. Then for any string $\alpha \in \{0,1\}^*$, suppose it represents machine M then all strings of the form $(010101)^*\alpha$ also represent the same machine M.
This has a similar effect as adding comments in the C program.

Notation

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Every string over $\{0,1\}^*$ represents some TM.

If a string does not represent any TM, as per our encoding, let us assume that it represents a TM that does nothing.

Every TM is represented by infinitely many strings.

Any encoding of TMs will have a null character, say 010101. Then for any string $\alpha \in \{0,1\}^*$, suppose it represents machine M then all strings of the form $(010101)^*\alpha$ also represent the same machine M.

This has a similar effect as adding comments in the C program.

Notation

$M \longrightarrow \langle M \rangle$, a string representation of M.

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Every string over $\{0,1\}^*$ represents some TM.

If a string does not represent any TM, as per our encoding, let us assume that it represents a TM that does nothing.

Every TM is represented by infinitely many strings.

Any encoding of TMs will have a null character, say 010101. Then for any string $\alpha \in \{0,1\}^*$, suppose it represents machine M then all strings of the form $(010101)^*\alpha$ also represent the same machine M.
This has a similar effect as adding comments in the C program.

Notation

$M \longrightarrow \langle M \rangle$, a string representation of M.

$\alpha \longrightarrow M_\alpha$, a machine corresponding to α.

Turing machines as strings

Every TM can be represented as a string in $\{0,1\}^*$.

Just encode the description of the machine.

Every string over $\{0,1\}^*$ represents some TM.

If a string does not represent any TM, as per our encoding, let us assume that it represents a TM that does nothing.

Every TM is represented by infinitely many strings.

Any encoding of TMs will have a null character, say 010101. Then for any string $\alpha \in \{0,1\}^*$, suppose it represents machine M then all strings of the form $(010101)^*\alpha$ also represent the same machine M.

This has a similar effect as adding comments in the C program.

Notation

$M \longrightarrow \langle M \rangle$, a string representation of M.

$\alpha \longrightarrow M_\alpha$, a machine corresponding to α.

$\{0,1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

$\{0,1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

Lemma

The set $\{0,1\}^*$ *is countable.*

$\{0, 1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

Lemma

The set $\{0, 1\}^*$ is countable.

Proof.

Let $x \in \{0, 1\}^*$.

$\{0,1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

Lemma

The set $\{0,1\}^*$ is countable.

Proof.

Let $x \in \{0,1\}^*$.

Let $\phi(x)$ be defined as the number in $y \in \mathbb{N}$ such that

$\{0,1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

Lemma

The set $\{0,1\}^*$ is countable.

Proof.

Let $x \in \{0,1\}^*$.

Let $\phi(x)$ be defined as the number in $y \in \mathbb{N}$ such that y is a binary encoding of the number $1x$.

$\{0,1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

Lemma

The set $\{0,1\}^*$ is countable.

Proof.

Let $x \in \{0,1\}^*$.

Let $\phi(x)$ be defined as the number in $y \in \mathbb{N}$ such that y is a binary encoding of the number $1x$.

ϕ is a map from $\{0,1\}^*$ to $\mathbb{N}$.

$\{0,1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

Lemma

The set $\{0,1\}^*$ is countable.

Proof.

Let $x \in \{0,1\}^*$.

Let $\phi(x)$ be defined as the number in $y \in \mathbb{N}$ such that y is a binary encoding of the number $1x$.

ϕ is a map from $\{0,1\}^*$ to $\mathbb{N}$.

If $|x| \neq |x'|$ then $\phi(x) \neq \phi(x')$.

$\{0,1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

Lemma

The set $\{0,1\}^*$ is countable.

Proof.

Let $x \in \{0,1\}^*$.

Let $\phi(x)$ be defined as the number in $y \in \mathbb{N}$ such that y is a binary encoding of the number $1x$.

ϕ is a map from $\{0,1\}^*$ to $\mathbb{N}$.

If $|x| \neq |x'|$ then $\phi(x) \neq \phi(x')$. If $|x| = |x'|$, then bin$(1x) \neq$ bin$(1x')$ as long as $x \neq x'$.

$\{0,1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

Lemma

The set $\{0,1\}^*$ is countable.

Proof.

Let $x \in \{0,1\}^*$.

Let $\phi(x)$ be defined as the number in $y \in \mathbb{N}$ such that y is a binary encoding of the number $1x$.

ϕ is a map from $\{0,1\}^*$ to $\mathbb{N}$.

If $|x| \neq |x'|$ then $\phi(x) \neq \phi(x')$. If $|x| = |x'|$, then bin$(1x) \neq$ bin$(1x')$ as long as $x \neq x'$.

Hence the map ϕ is injective.

$\{0,1\}^*$ is countable

Definition (Countable set)

A set S is said to be countable if there is an injective map from S to $\mathbb{N}$.

Lemma

The set $\{0,1\}^*$ is countable.

Proof.

Let $x \in \{0,1\}^*$.

Let $\phi(x)$ be defined as the number in $y \in \mathbb{N}$ such that y is a binary encoding of the number $1x$.

ϕ is a map from $\{0,1\}^*$ to $\mathbb{N}$.

If $|x| \neq |x'|$ then $\phi(x) \neq \phi(x')$. If $|x| = |x'|$, then bin$(1x) \neq$ bin$(1x')$ as long as $x \neq x'$.

Hence the map ϕ is injective.

Cantor's diagonalisation

Theorem (Cantor, 1891)

There is no bijection between $\mathbb{N}$ and $2^{\mathbb{N}}$ (set of all subsets of $\mathbb{N}$).

Proof.

Suppose for the sake of contradiction that there is a bijection, say f, between set of all subsets of $\mathbb{N}$.

	0	1	2	3	...
$\emptyset$					
$\{1\}$					
$\{2\}$					
$\{1,2\}$					
$\vdots$					
$\vdots$					

Cantor's diagonalisation

Theorem (Cantor, 1891)

There is no bijection between $\mathbb{N}$ and $2^{\mathbb{N}}$ (set of all subsets of $\mathbb{N}$).

Proof.

Suppose for the sake of contradiction that there is a bijection, say f, between set of all subsets of $\mathbb{N}$.

	0	1	2	3	…
$\emptyset$	✗	✗	✗	✗	…
$\{1\}$	✗	✓	✗	✗	…
$\{2\}$	✗	✗	✓	✗	…
$\{1,2\}$	✗	✓	✓	✗	…
:	…	…	…	…	…
:	…	…	…	…	…

Cantor's diagonalisation

Theorem (Cantor, 1891)

There is no bijection between $\mathbb{N}$ and $2^{\mathbb{N}}$ (set of all subsets of $\mathbb{N}$).

Proof.

Suppose for the sake of contradiction that there is a bijection, say f, between set of all subsets of $\mathbb{N}$.

	0	1	2	3	…
$\emptyset$	✓	✗	✗	✗	…
$\{1\}$	✗	✗	✗	✗	…
$\{2\}$	✗	✗	✗	✗	…
$\{1,2\}$	✗	✓	✓	✓	…
⋮	…	…	…	…	…
⋮	…	…	…	…	…

Cantor's diagonalisation

Theorem (Cantor, 1891)

There is no bijection between $\mathbb{N}$ and $2^{\mathbb{N}}$ (set of all subsets of $\mathbb{N}$).

Proof.

Suppose for the sake of contradiction that there is a bijection, say f, between set of all subsets of $\mathbb{N}$.

	0	1	2	3	...
$\emptyset$	✓	✗	✗	✗	...
$\{1\}$	✗	✗	✗	✗	...
$\{2\}$	✗	✗	✗	✗	...
$\{1,2\}$	✗	✓	✓	✓	...
⋮	...	...	...	...	...
⋮	...	...	...	...	...

The inverted diagonal set does not belong to any of the existing sets! □

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Let L be a language, i.e. $L \subseteq \Sigma^*$

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Let L be a language, i.e. $L \subseteq \Sigma^*$, $w \in \Sigma^*$.

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Let L be a language, i.e. $L \subseteq \Sigma^*$, $w \in \Sigma^*$.

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Let L be a language, i.e. $L \subseteq \Sigma^*$, $w \in \Sigma^*$.

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

$$\text{languages over } \Sigma^* \xrightarrow{\text{bijection}} 2^{\mathbb{N}}$$

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Let L be a language, i.e. $L \subseteq \Sigma^*$, $w \in \Sigma^*$.

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

$$\text{languages over } \Sigma^* \xrightarrow{\text{bijection}} 2^{\mathbb{N}}$$

- Therefore, set of all languages is uncountable.

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Let L be a language, i.e. $L \subseteq \Sigma^*$, $w \in \Sigma^*$.

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

$$\text{languages over } \Sigma^* \xrightarrow{\text{bijection}} 2^{\mathbb{N}}$$

- Therefore, set of all languages is uncountable.
- However, the set of all TMs is countable.

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Let L be a language, i.e. $L \subseteq \Sigma^*$, $w \in \Sigma^*$.

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

$$\text{languages over } \Sigma^* \xrightarrow{\text{bijection}} 2^{\mathbb{N}}$$

- Therefore, set of all languages is uncountable.
- However, the set of all TMs is countable. ($\{0,1\}^*$ is countable.)

Turing recognizable languages

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Let L be a language, i.e. $L \subseteq \Sigma^*$, $w \in \Sigma^*$.

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

$$\text{languages over } \Sigma^* \xrightarrow{\text{bijection}} 2^{\mathbb{N}}$$

- Therefore, set of all languages is uncountable.
- However, the set of all TMs is countable. ($\{0,1\}^*$ is countable.)
- There must be a language which is not Turing recognizable.

A decision problem about TMs

$$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$$

A decision problem about TMs

$$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$$

Lemma

A_{TM} is Turing recognizable.

A decision problem about TMs

$$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$$

Lemma

A_{TM} *is Turing recognizable.*

Proof sketch

A decision problem about TMs

$$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$$

Lemma

A_{TM} *is Turing recognizable.*

Proof sketch

Design a TM, say N such that

A decision problem about TMs

$$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$$

Lemma

A_{TM} *is Turing recognizable.*

Proof sketch

Design a TM, say N such that,

N behaves like M on w at each step

A decision problem about TMs

$$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$$

Lemma

A_{TM} *is Turing recognizable.*

Proof sketch

Design a TM, say N such that,

N behaves like M on w at each step,

if M reaches q_{acc} then N also accepts.

A decision problem about TMs

$$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$$

Lemma

A_{TM} is Turing recognizable.

Proof sketch

Design a TM, say N such that,

N behaves like M on w at each step,

if M reaches q_{acc} then N also accepts.

Is A_{TM} decidable?

A decision problem about TMs

Lemma

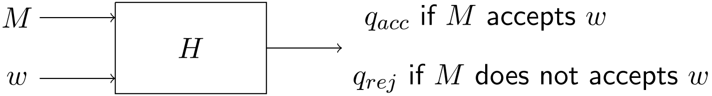
$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$ *is not Turing decidable.*

A decision problem about TMs

Lemma

$A_{TM} = \{(M, w) \mid M$ *accepts* $w\}$ *is not Turing decidable.*

Assume that there exists M such that M decides A_{TM}.

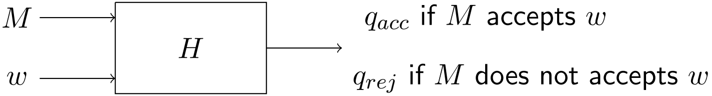


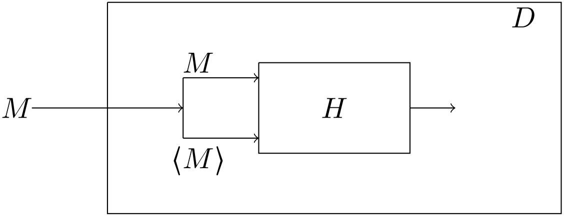
A decision problem about TMs

Lemma

$A_{TM} = \{(M, w) \mid M$ *accepts* $w\}$ *is not Turing decidable.*

Assume that there exists M such that M decides A_{TM}.



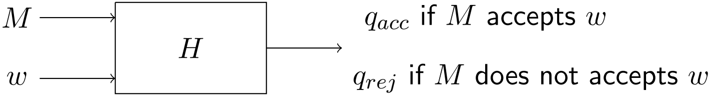


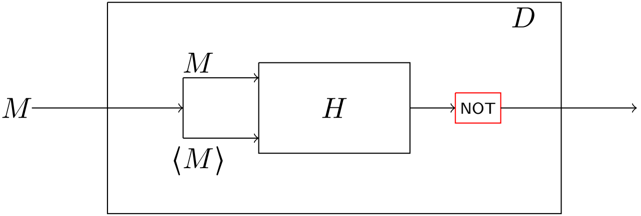
A decision problem about TMs

Lemma

$A_{TM} = \{(M, w) \mid M$ *accepts* $w\}$ *is not Turing decidable.*

Assume that there exists M such that M decides A_{TM}.



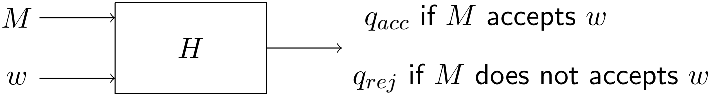


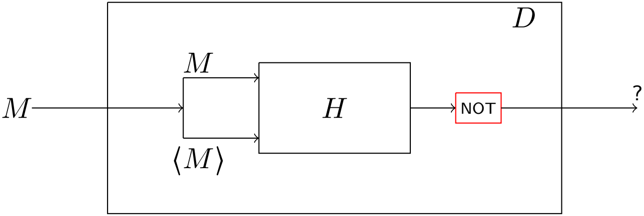
A decision problem about TMs

Lemma

$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$ *is not Turing decidable.*

Assume that there exists M such that M decides A_{TM}.



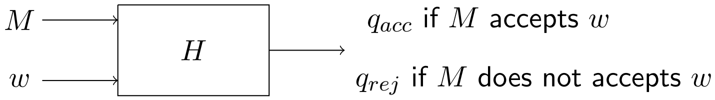


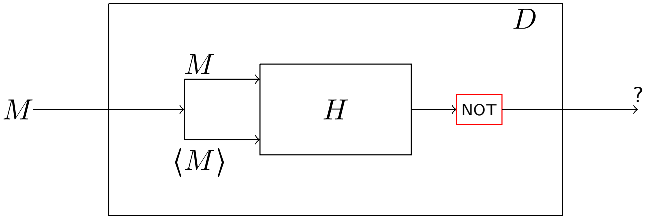
A decision problem about TMs

Lemma

$A_{TM} = \{(M, w) \mid M$ *accepts* $w\}$ *is not Turing decidable.*

Assume that there exists M such that M decides A_{TM}.



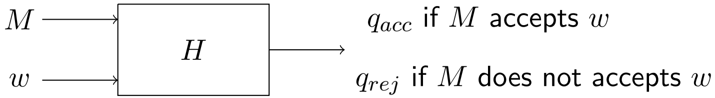


A decision problem about TMs

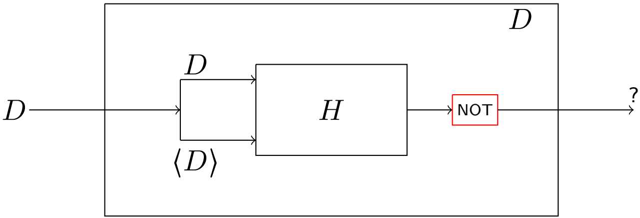
Lemma

A_{TM} *is not Turing decidable.*

Assume that there exists H such that H decides A_{TM}.



What happens if we give D as input to itself?

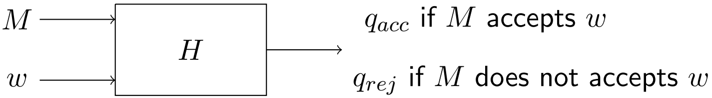


A decision problem about TMs

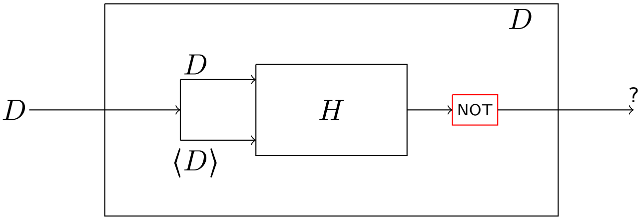
Lemma

A_{TM} *is not Turing decidable.*

Assume that there exists H such that H decides A_{TM}.



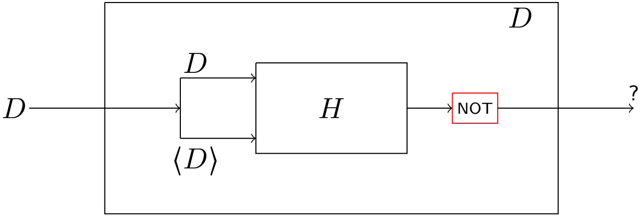
What happens if we give D as input to itself?



A decision problem about TMs

Lemma

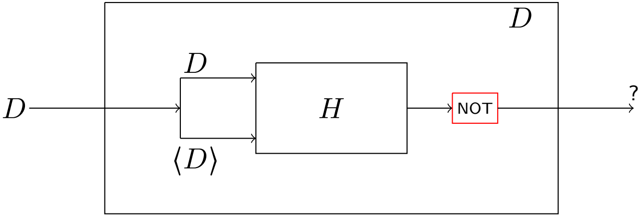
A_{TM} *is not Turing decidable.*



A decision problem about TMs

Lemma

A_{TM} *is not Turing decidable.*

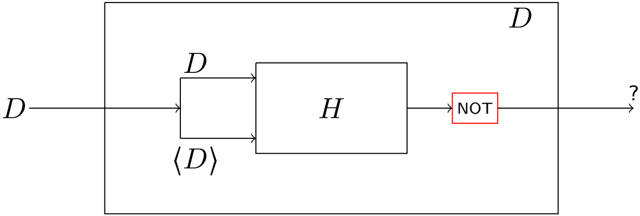


If D accepts $\langle D \rangle$

A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.

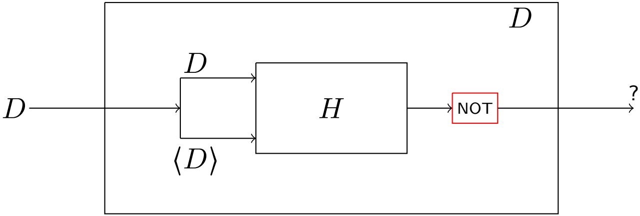


If D accepts $\langle D \rangle$ then D rejects $\langle D \rangle$.

A decision problem about TMs

Lemma

A_{TM} *is not Turing decidable.*



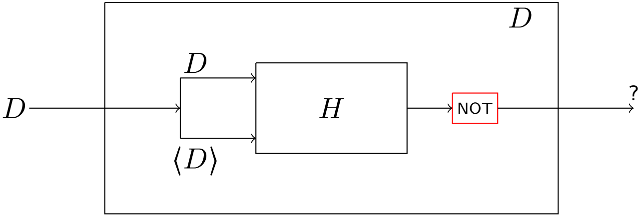
If D accepts $\langle D \rangle$ then D rejects $\langle D \rangle$.

If D rejects $\langle D \rangle$

A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.



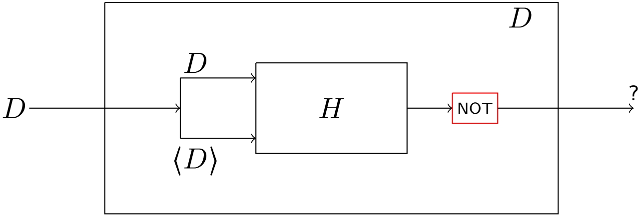
If D accepts $\langle D \rangle$ then D rejects $\langle D \rangle$.

If D rejects $\langle D \rangle$ then D accepts $\langle D \rangle$.

A decision problem about TMs

Lemma

A_{TM} *is not Turing decidable.*



If D accepts $\langle D \rangle$ then D rejects $\langle D \rangle$.

If D rejects $\langle D \rangle$ then D accepts $\langle D \rangle$. ☹

A few notable things

Note the following about the proof.

A few notable things

Note the following about the proof.

H accepts $\langle M, w\rangle$ when M accepts w.

A few notable things

Note the following about the proof.

H accepts $\langle M, w \rangle$ when M accepts w.

D rejects $\langle M \rangle$ when M accepts $\langle M \rangle$.

A few notable things

Note the following about the proof.

H accepts $\langle M, w \rangle$ when M accepts w.

D rejects $\langle M \rangle$ when M accepts $\langle M \rangle$.

D rejects $\langle D \rangle$ when D accepts $\langle D \rangle$.

Diagonalization inside the proof

Behaviour of the machines.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots\ldots$	$\ldots\ldots$
M_1	$\checkmark$		$\checkmark$	$\checkmark\ldots$	$\ldots\ldots$

Diagonalization inside the proof

Behaviour of the machines.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots\ldots$	$\ldots\ldots$
M_1	$\checkmark$		$\checkmark$	$\checkmark\ldots$	$\ldots\ldots$
M_2	$\checkmark$	$\times$		$\times\ \ldots$	$\checkmark\ldots\times\checkmark\ \ldots$

Diagonalization inside the proof

Behaviour of the machines.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots\ldots$	$\ldots\ldots$
M_1	✓		✓	✓ $\ldots$	$\ldots\ldots$
M_2	✓	$\times$		$\times$ $\ldots$	✓ $\ldots \times$ ✓ $\ldots$
M_3	$\times$	$\times$	✓	$\ldots \times$	✓ $\ldots\ldots$
$\vdots$					
$\vdots$					

Diagonalization inside the proof

Behaviour of H.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots\ldots$	$\ldots\ldots$
M_1	$\checkmark$	$\times$	$\checkmark$	$\checkmark\ldots$	$\ldots\ldots$
M_2	$\checkmark$	$\times$	$\times$	$\times\ \ldots$	$\checkmark\ldots\times\checkmark\ \ldots$
M_3	$\times$	$\times$	$\checkmark$	$\ldots\times$	$\checkmark\ldots\ldots$
$\vdots$					
$\vdots$					

Diagonalization inside the proof

Behaviour of H.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots\ldots$	$\ldots\ldots$
M_1	$\checkmark$	$\times$	$\checkmark$	$\checkmark \ldots$	$\ldots\ldots$
M_2	$\checkmark$	$\times$	$\times$	$\times$ $\ldots$	$\checkmark \ldots \times \checkmark$ $\ldots$
M_3	$\times$	$\times$	$\checkmark$	$\ldots \times$	$\checkmark \ldots\ldots$
$\vdots$					
$\vdots$					

Diagonalization inside the proof

Behaviour of D.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots\ldots$	$\ldots\ldots$
M_1	$\cancel{\checkmark}$ $\times$	$\times$	$\checkmark$	$\checkmark \ldots$	$\ldots\ldots$
M_2	$\checkmark$	$\cancel{\times}$ $\checkmark$	$\times$	$\times \ldots$	$\checkmark \ldots \times \checkmark \ldots$
M_3	$\times$	$\times$	$\cancel{\checkmark}$ $\times$	$\ldots \times$	$\checkmark \ldots\ldots$
$\vdots$					
$\vdots$					

Diagonalization inside the proof

Behaviour of D on itself.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\dots \langle D \rangle \dots$	$\dots\dots$
M_1	$\cancel{\checkmark}$ $\times$	$\times$	$\checkmark$	$\checkmark \dots$	$\dots\dots$
M_2	$\checkmark$	$\cancel{\times}$ $\checkmark$	$\times$	$\times \dots$	$\checkmark \dots \times \checkmark \dots$
M_3	$\times$	$\times$	$\cancel{\checkmark}$ $\times$	$\dots \times$	$\checkmark \dots\dots$
$\vdots$					
$\vdots$					
D				$\dots$ $?$ $\dots$	$\dots\dots$

Back to Comparing decidability and recognizability

Theorem

A language L is Turing decidable if and only if L and $\overline{L}$ are both Turing recognizable.

Back to Comparing decidability and recognizability

Theorem

A language L is Turing decidable if and only if L and $\overline{L}$ are both Turing recognizable.

- A_{TM} is Turing recongizable.

Back to Comparing decidability and recognizability

Theorem

A language L is Turing decidable if and only if L and $\overline{L}$ are both Turing recognizable.

- A_{TM} is Turing recongizable.
- If $\overline{A_{TM}}$ is also Turing recognizable, then A_{TM} will be decidable!

Back to Comparing decidability and recognizability

Theorem

A language L is Turing decidable if and only if L and $\overline{L}$ are both Turing recognizable.

- A_{TM} is Turing recongizable.
- If $\overline{A_{TM}}$ is also Turing recognizable, then A_{TM} will be decidable!

Corollary

$\overline{A_{TM}}$ is not Turing recognizable.

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the description of any Turing machine M and an input w, simulate the machine M on w.

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the description of any Turing machine M and an input w, simulate the machine M on w.

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the description of any Turing machine M and an input w, simulate the machine M on w.

Lemma

Universal Turing machine (UTM) exists.

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the description of any Turing machine M and an input w, simulate the machine M on w.

Lemma

Universal Turing machine (UTM) exists. [Turing, 1937]

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the description of any Turing machine M and an input w, simulate the machine M on w.

Lemma

Universal Turing machine (UTM) exists. [Turing, 1937]

Proof.

We will prove the lemma by explicitly constructing such a machine.

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the description of any Turing machine M and an input w, simulate the machine M on w.

Lemma

Universal Turing machine (UTM) exists. [Turing, 1937]

Proof.

We will prove the lemma by explicitly constructing such a machine.

Proof idea:

1. Find a good encoding for Turing machines.

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the description of any Turing machine M and an input w, simulate the machine M on w.

Lemma

Universal Turing machine (UTM) exists. [Turing, 1937]

Proof.

We will prove the lemma by explicitly constructing such a machine.

Proof idea:

1. Find a good encoding for Turing machines.
2. Tape 1: Hold the input, namely M and w.

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the description of any Turing machine M and an input w, simulate the machine M on w.

Lemma

Universal Turing machine (UTM) exists. [Turing, 1937]

Proof.

We will prove the lemma by explicitly constructing such a machine.

Proof idea:

1. Find a good encoding for Turing machines.
2. Tape 1: Hold the input, namely M and w.
3. Tape 2: Copy the decription of M and use it for referencing moves.

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the description of any Turing machine M and an input w, simulate the machine M on w.

Lemma

Universal Turing machine (UTM) exists. [Turing, 1937]

Proof.

We will prove the lemma by explicitly constructing such a machine.

Proof idea:

1. Find a good encoding for Turing machines.
2. Tape 1: Hold the input, namely M and w.
3. Tape 2: Copy the decription of M and use it for referencing moves.
4. Tape 3: Store the current state M and letter of w being read.

Universal Turing machines

Definition

A Turing machine is called a Universal Turing machine if it can given the description of any Turing machine M and an input w, simulate the machine M on w.

Lemma

Universal Turing machine (UTM) exists. [Turing, 1937]

Proof.

We will prove the lemma by explicitly constructing such a machine.

Proof idea:

1. Find a good encoding for Turing machines.
2. Tape 1: Hold the input, namely M and w.
3. Tape 2: Copy the decription of M and use it for referencing moves.
4. Tape 3: Store the current state M and letter of w being read.

□