CS310 Automata Theory – 2016-2017

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Lecture 23: Turing machines, computability
March 14, 2017

Last three classes

Introduction to Turing machines

What are Turing machines? Informal and formal definitions.

Last three classes

Introduction to Turing machines

What are Turing machines? Informal and formal definitions.

Examples. $\{a^n b^n \mid n \geq 0\}$, $\{w\#w \mid w \in \{a,b\}^*\}$, $\{a^{2^n} \mid n \geq 0\}$.

Last three classes

Introduction to Turing machines

What are Turing machines? Informal and formal definitions.

Examples. $\{a^n b^n \mid n \geq 0\}$, $\{w\#w \mid w \in \{a,b\}^*\}$, $\{a^{2^n} \mid n \geq 0\}$.

Configurations of a Turing machine.

Last three classes

Introduction to Turing machines

What are Turing machines? Informal and formal definitions.

Examples. $\{a^n b^n \mid n \geq 0\}$, $\{w\#w \mid w \in \{a,b\}^*\}$, $\{a^{2^n} \mid n \geq 0\}$.

Configurations of a Turing machine.

Turing recognizable and Turing decidable languages.

Last three classes

Introduction to Turing machines

What are Turing machines? Informal and formal definitions.

Examples. $\{a^n b^n \mid n \geq 0\}$, $\{w\#w \mid w \in \{a,b\}^*\}$, $\{a^{2^n} \mid n \geq 0\}$.

Configurations of a Turing machine.

Turing recognizable and Turing decidable languages.

k-tape TMs equivalent to 1-tape TMs.

Variants of Turing machines

k-tape Turing machines

Variants of Turing machines

k-tape Turing machines

Usual TM + Multiples tapes + independent tape-head for each tape.

Variants of Turing machines

k-tape Turing machines

Usual TM + Multiples tapes + independent tape-head for each tape.

$\delta \subseteq Q \times \Gamma^k \times Q \times \Gamma^k \times \{L, R, S\}^k$.

Variants of Turing machines

k-tape Turing machines

Usual TM + Multiples tapes + independent tape-head for each tape.

$\delta \subseteq Q \times \Gamma^k \times Q \times \Gamma^k \times \{L, R, S\}^k$.

Example

Variants of Turing machines

k-tape Turing machines

Usual TM + Multiples tapes + independent tape-head for each tape.

$\delta \subseteq Q \times \Gamma^k \times Q \times \Gamma^k \times \{L, R, S\}^k$.

Example

Given: 1^n on the input tape

Variants of Turing machines

k-tape Turing machines

Usual TM $+$ Multiples tapes $+$ independent tape-head for each tape.

$\delta \subseteq Q \times \Gamma^k \times Q \times \Gamma^k \times \{L, R, S\}^k$.

Example

Given: 1^n on the input tape

Output: 1^{n^2} on the same tape.

Variants of Turing machines

k-tape Turing machines

Usual TM + Multiples tapes + independent tape-head for each tape.

$\delta \subseteq Q \times \Gamma^k \times Q \times \Gamma^k \times \{L, R, S\}^k$.

Example

Given:	1^n on the input tape
Output:	1^{n^2} on the same tape.

Are k-tape TMs more powerful than 1-tape TMs?

Variants of Turing machines

k-tape Turing machines

Usual TM + Multiples tapes + independent tape-head for each tape.

$\delta \subseteq Q \times \Gamma^k \times Q \times \Gamma^k \times \{L, R, S\}^k$.

Example

Given:	1^n on the input tape
Output:	1^{n^2} on the same tape.

Are k-tape TMs more powerful than 1-tape TMs?

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

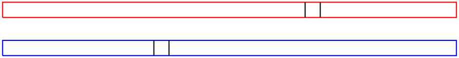
Proof sketch:

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

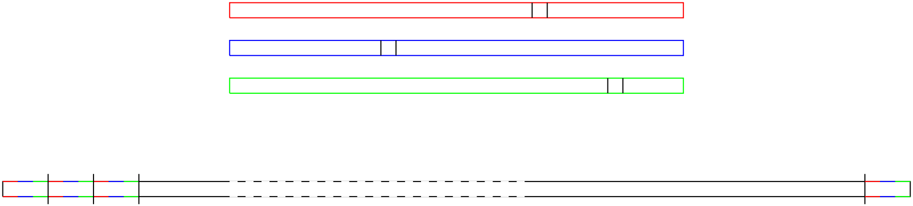
Proof sketch:

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:

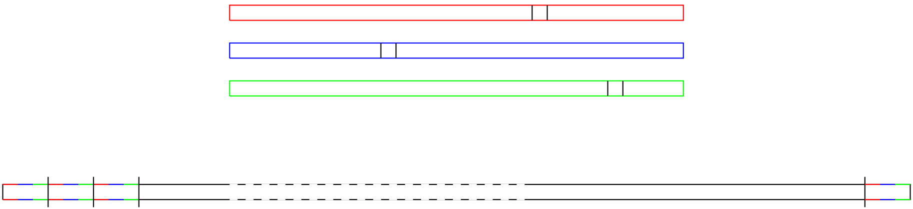
Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej},)$ be the k-tape Turing machine.

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej},)$ be the k-tape Turing machine.

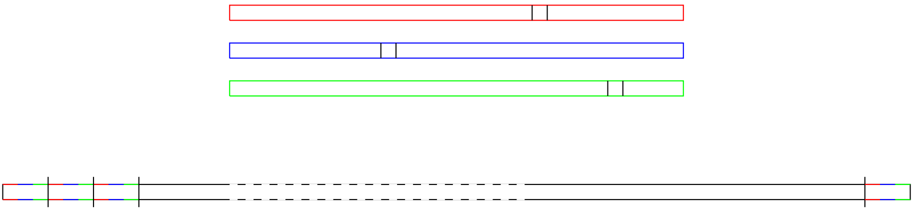
Let $M' = (Q', \Sigma, \Gamma', \delta', q_0, q_{acc}, q_{rej})$ be such that

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej},)$ be the k-tape Turing machine.

Let $M' = (Q', \Sigma, \Gamma', \delta', q_0, q_{acc}, q_{rej})$ be such that,

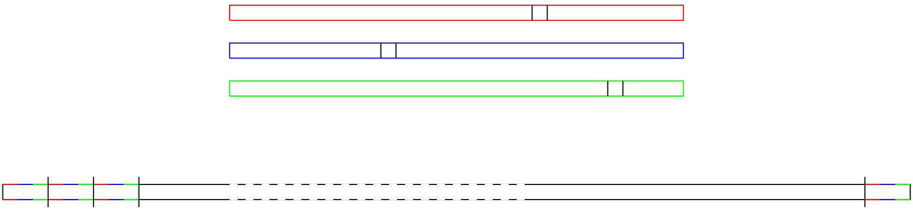
$\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej},)$ be the k-tape Turing machine.

Let $M' = (Q', \Sigma, \Gamma', \delta', q_0, q_{acc}, q_{rej})$ be such that,

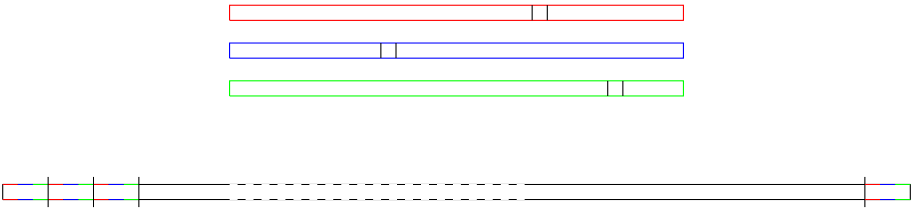
$\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$, $\Gamma = \Gamma \cup \overline{\Gamma} \cup \{\#\}$.

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej},)$ be the k-tape Turing machine.

Let $M' = (Q', \Sigma, \Gamma', \delta', q_0, q_{acc}, q_{rej})$ be such that,

$\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$, $\Gamma = \Gamma \cup \overline{\Gamma} \cup \{\#\}$.

$\overline{\Gamma}$ symbols used to denote tape head positions.

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

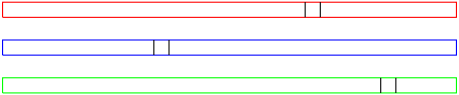
Proof sketch:

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:

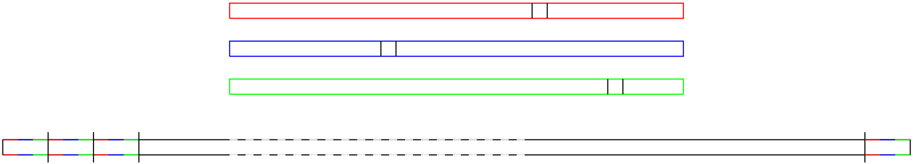


k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:

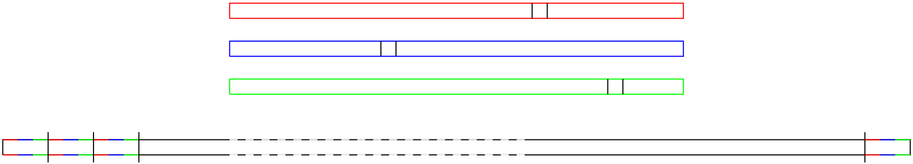


k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



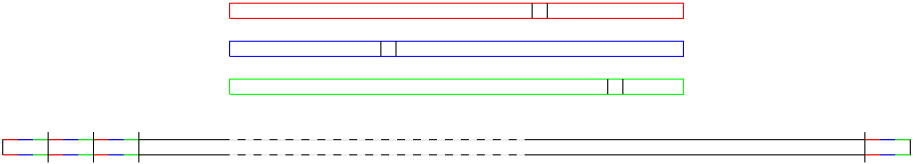
To simulate 1 step of M

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



To simulate 1 step of M, M' works follows:

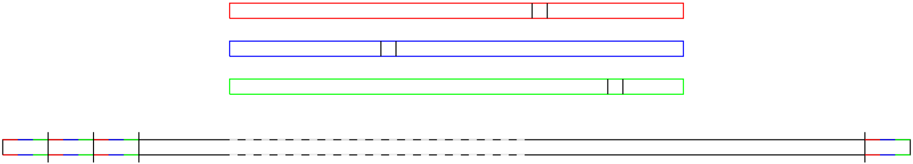
reads the tape left to right once, remembeing the marked symbols in its states

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



To simulate 1 step of M, M' works follows:

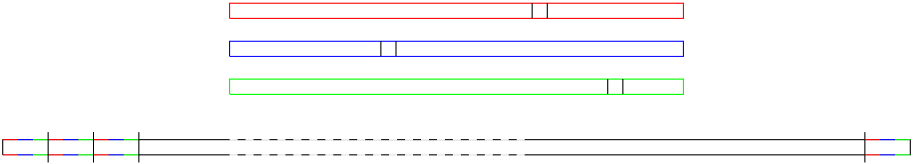
- reads the tape left to right once, remembeing the marked symbols in its states,
- uses δ to determine the next state

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



To simulate 1 step of M, M' works follows:

reads the tape left to right once, remembeing the marked symbols in its states,

uses δ to determine the next state,

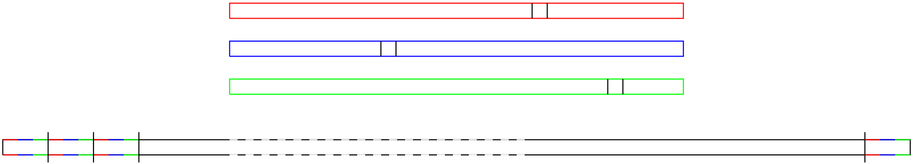
sweeps the input left to right again

k-tape Turing machines

Theorem

Every k-tape Turing machine has an equivalent 1-tape Turing machine.

Proof sketch:



To simulate 1 step of M, M' works follows:

reads the tape left to right once, remembeing the marked symbols in its states,

uses δ to determine the next state,

sweeps the input left to right again to update marked symbols.

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.

Proof.

($\Rightarrow$)

If L is Turing decidable then L is also Turing recognizable

If L is Turing decidable, then $\overline{L}$ is also Turing decidable.

Therefore, $\overline{L}$ is also Turing recognizable.

($\Leftarrow$)

Let M_1, M_2 be two TMs recognizing $L, \overline{L}$, respectively.

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.

Proof.

($\Rightarrow$)

If L is Turing decidable then L is also Turing recognizable

If L is Turing decidable, then $\overline{L}$ is also Turing decidable.

Therefore, $\overline{L}$ is also Turing recognizable.

($\Leftarrow$)

Let M_1, M_2 be two TMs recognizing $L, \overline{L}$, respectively.

We wish to come up with a TM M that will decide L.

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.

Proof.

($\Rightarrow$)

If L is Turing decidable then L is also Turing recognizable

If L is Turing decidable, then $\overline{L}$ is also Turing decidable.

Therefore, $\overline{L}$ is also Turing recognizable.

($\Leftarrow$)

Let M_1, M_2 be two TMs recognizing $L, \overline{L}$, respectively.

We wish to come up with a TM M that will decide L.

Idea: on input w run both M_1, M_2

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.

Proof.

$(\Rightarrow)$

If L is Turing decidable then L is also Turing recognizable

If L is Turing decidable, then $\overline{L}$ is also Turing decidable.

Therefore, $\overline{L}$ is also Turing recognizable.

$(\Leftarrow)$

Let M_1, M_2 be two TMs recognizing $L, \overline{L}$, respectively.

We wish to come up with a TM M that will decide L.

Idea: on input w run both M_1, M_2, if M_1 reaches accepting configuration then accept

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.

Proof.

$(\Rightarrow)$

If L is Turing decidable then L is also Turing recognizable

If L is Turing decidable, then $\overline{L}$ is also Turing decidable.

Therefore, $\overline{L}$ is also Turing recognizable.

$(\Leftarrow)$

Let M_1, M_2 be two TMs recognizing $L, \overline{L}$, respectively.

We wish to come up with a TM M that will decide L.

Idea: on input w run both M_1, M_2, if M_1 reaches accepting configuration then accept. Else M_2 will reach the accepting configuraion. In that case, reject.

Turing recognizability for $L, \overline{L} \Rightarrow$ Turing decidibility for L

We design 2-tape TM M, using TMs M_1, M_2 as follows:

Turing recognizability for $L, \overline{L} \Rightarrow$ Turing decidibility for L

We design 2-tape TM M, using TMs M_1, M_2 as follows:

M copies input from tape 1 to tape 2.

Turing recognizability for $L, \overline{L} \Rightarrow$ Turing decidibility for L

We design 2-tape TM M, using TMs M_1, M_2 as follows:

M copies input from tape 1 to tape 2.

It acts as M_1 on tape 1 and as M_2 on tape 2.

Turing recognizability for $L, \overline{L} \Rightarrow$ Turing decidibility for L

We design 2-tape TM M, using TMs M_1, M_2 as follows:

M copies input from tape 1 to tape 2.

It acts as M_1 on tape 1 and as M_2 on tape 2.

M keeps track of the state control of M_1, M_2 in $Q_1 \times Q_2$.

Turing recognizability for $L, \overline{L} \Rightarrow$ Turing decidibility for L

We design 2-tape TM M, using TMs M_1, M_2 as follows:

M copies input from tape 1 to tape 2.

It acts as M_1 on tape 1 and as M_2 on tape 2.

M keeps track of the state control of M_1, M_2 in $Q_1 \times Q_2$.

Can you give a full decsription of M?

Turing recognizability for $L, \overline{L} \Rightarrow$ Turing decidibility for L

We design 2-tape TM M, using TMs M_1, M_2 as follows:

M copies input from tape 1 to tape 2.

It acts as M_1 on tape 1 and as M_2 on tape 2.

M keeps track of the state control of M_1, M_2 in $Q_1 \times Q_2$.

Can you give a full decsription of M? DIY!

Turing machines as strings

Every TM represented as a string in $\{0, 1\}^*$ with the following properties:

Turing machines as strings

Every TM represented as a string in $\{0,1\}^*$ with the following properties:

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

Turing machines as strings

Every TM represented as a string in $\{0,1\}^*$ with the following properties:

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

Every TM is represented by infinitely many strings.

Turing machines as strings

Every TM represented as a string in $\{0,1\}^*$ with the following properties:

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

Every TM is represented by infinitely many strings.

Notation

Turing machines as strings

Every TM represented as a string in $\{0,1\}^*$ with the following properties:

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

Every TM is represented by infinitely many strings.

Notation

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

Turing machines as strings

Every TM represented as a string in $\{0,1\}^*$ with the following properties:

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

Every TM is represented by infinitely many strings.

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 represented as a string in $\{0,1\}^*$ with the following properties:

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

Every TM is represented by infinitely many strings.

Notation

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

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

Turing machines as strings

Lemma

There exists a language which is not Turing recognizable.

Turing machines as strings

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

Turing machines as strings

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

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

Turing machines as strings

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

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

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

Turing machines as strings

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

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

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

Turing machines as strings

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

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

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 machines as strings

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

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

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

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

Therefore, set of all languages is uncountable.

Turing machines as strings

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

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

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

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

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

Turing machines as strings

Lemma

There exists a language which is not Turing recognizable.

Proof.

Fix an alphabet Σ.

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

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

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

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

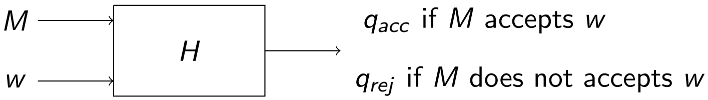
A_{TM} is not Turing decidable.

A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.

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

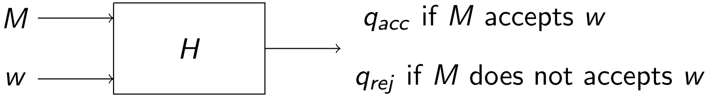


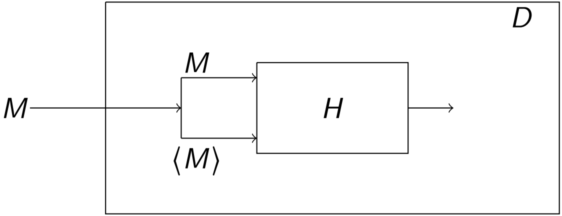
A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.

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



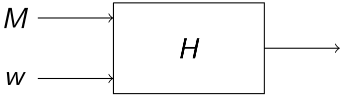


A decision problem about TMs

Lemma

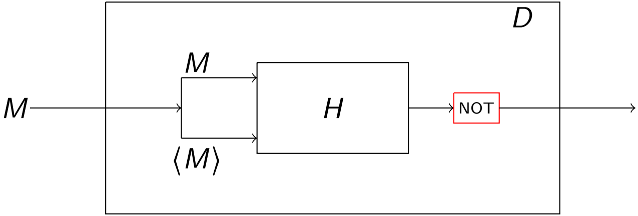
A_{TM} is not Turing decidable.

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



q_{acc} if M accepts w

q_{rej} if M does not accepts w

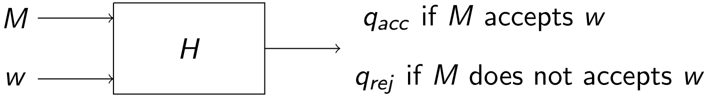


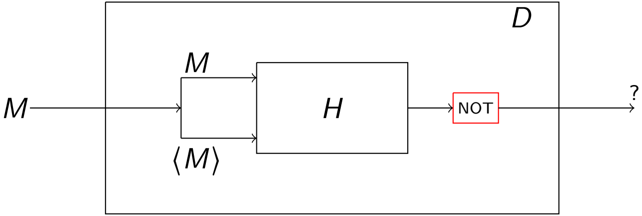
A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.

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



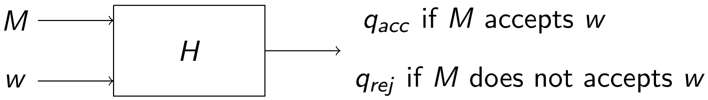


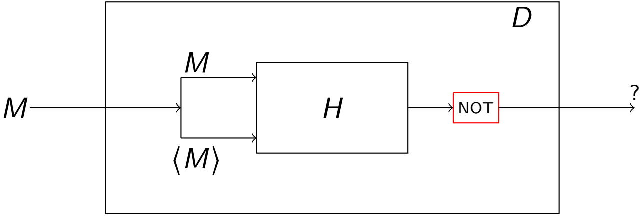
A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.

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



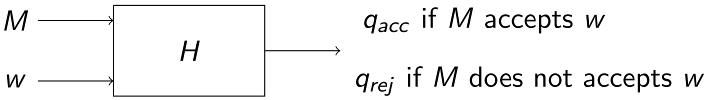


A decision problem about TMs

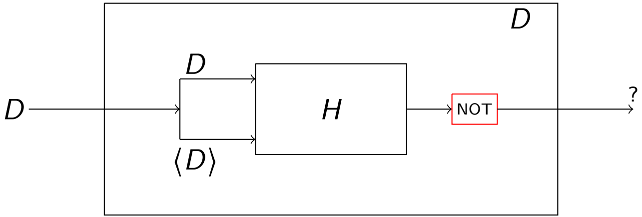
Lemma

A_{TM} is not Turing decidable.

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



What happens if we give D as input to itself?

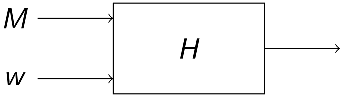


A decision problem about TMs

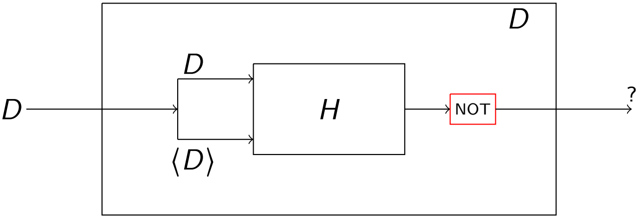
Lemma

A_{TM} is not Turing decidable.

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



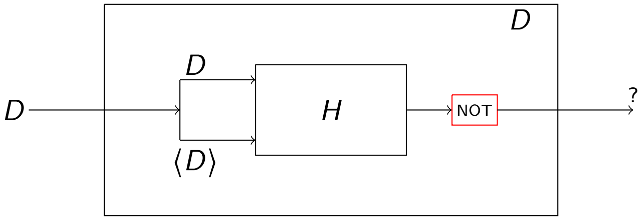
What happens if we give D as input to itself?



A decision problem about TMs

Lemma

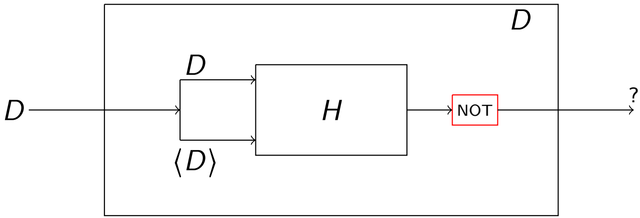
A_{TM} is not Turing decidable.



A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.

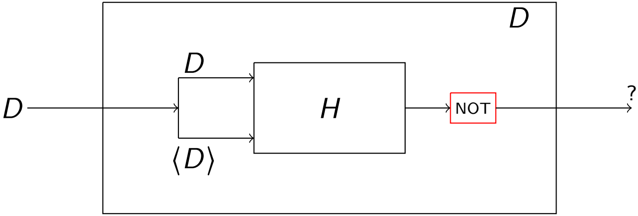


If D accepts $\langle D \rangle$

A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.

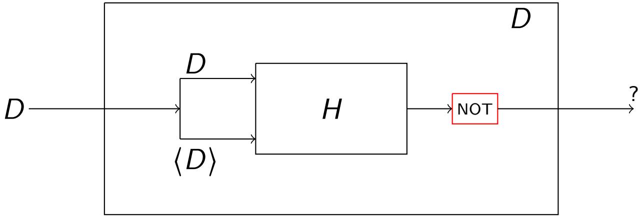


If D accepts $\langle D \rangle$ then D rejects $\langle D \rangle$.

A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.



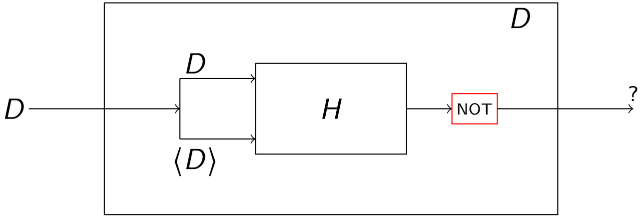
If D accepts $\langle D\rangle$ then D rejects $\langle D\rangle$.

If D rejects $\langle D\rangle$

A decision problem about TMs

Lemma

A_{TM} is not Turing decidable.



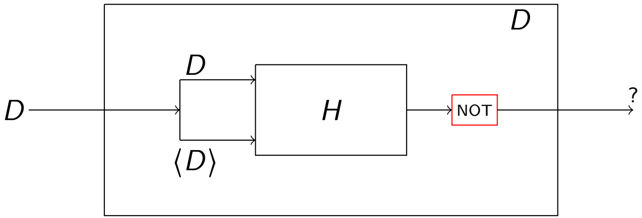
If D accepts $\langle D \rangle$ then D rejects $\langle D \rangle$.

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 $\langle D \rangle$. ☹