CS310 Automata Theory – 2016-2017

Nutan Limaye

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

Lecture 22: Turing machines, computability
March 09, 2017

Last two classes

Introduction to Turing machines

What are Turing machines? Informal and formal definitions.

Last two classes

Introduction to Turing machines

What are Turing machines? Informal and formal definitions.

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

Last two classes

Introduction to Turing machines

What are Turing machines? Informal and formal definitions.

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

Homework: $\{a^{2^n} \mid n \geq 0\}$.

Last two classes

Introduction to Turing machines

What are Turing machines? Informal and formal definitions.

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

Homework: $\{a^{2^n} \mid n \geq 0\}$.

Configurations of a Turing machine.

Last two classes

Introduction to Turing machines

What are Turing machines? Informal and formal definitions.

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

Homework: $\{a^{2^n} \mid n \geq 0\}$.

Configurations of a Turing machine.

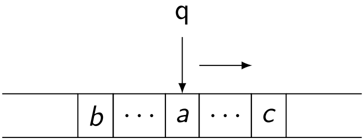
Turing recognizable and Turing decidable languages.

Turing machines

What is a Turing machine? (Informal description.)

Turing machines

What is a Turing machine? (Informal description.)



Read and write on the input tape. Head moves left/right.

The tape is infinite.

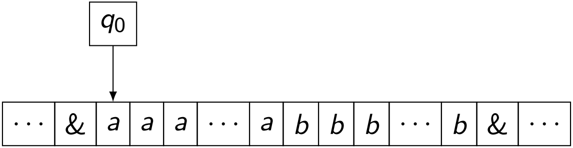
A special symbol & to indicate blank cells.

Initially all cells blank except the part where the input is written.

Special states for accepting and rejecting.

Example

$L_{a,b} = \{a^n b^n \mid n \geq 0\}.$



Formal definition

Definition

A Turing machine (TM) is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f, q_{rej})$

Q:	set of states	Σ:	input alphabet
q_0:	start state	Γ:	tape alphabet, $\Sigma \subseteq \Gamma$, & $\in \Gamma$
q_{acc}:	accept state	q_{rej}:	reject state

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

Formal definition

Definition

A Turing machine (TM) is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f, q_{rej})$

Q:	set of states	Σ:	input alphabet
q_0:	start state	Γ:	tape alphabet, $\Sigma \subseteq \Gamma$, $\& \in \Gamma$
q_{acc}:	accept state	q_{rej}:	reject state

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

Understanding δ

For a $q \in Q, a \in \Gamma$ if $\delta(q, a) = (p, b, L)$,

then p is the new state of the machine,

b is the letter with which a gets overwritten,

the head moves to the left of the current position.

Turing machine for a non-context free language

Example

Turing machine for a non-context free language

Example

$$\mathrm{EQ} = \{w \cdot \# \cdot w \mid w \in \Sigma^*\}.$$

Give a full description of a Turing machine for the above language.

Configuration

Definition

The configuration of a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f, q_{rej})$ is given by

$$\Gamma^* \times Q \times \Gamma^*$$

Let $u, v \in \Gamma^*$, $a, b, c \in \Gamma$ and $q, q' \in Q$.

Suppose $(q', c, L) \in \delta(q, b)$ is a transition in M,
then starting from $u \cdot a \cdot q \cdot b \cdot v$ in one step we get $u \cdot q' \cdot a \cdot c \cdot v$.

Configuration

Definition

The configuration of a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_f, q_{rej})$ is given by

$$\Gamma^* \times Q \times \Gamma^*$$

Let $u, v \in \Gamma^*$, $a, b, c \in \Gamma$ and $q, q' \in Q$.

Suppose $(q', c, L) \in \delta(q, b)$ is a transition in M,
then starting from $u \cdot a \cdot q \cdot b \cdot v$ in one step we get $u \cdot q' \cdot a \cdot c \cdot v$.

We say that $u \cdot a \cdot q \cdot b \cdot v$ **yields** $u \cdot q' \cdot a \cdot c \cdot v$.
We denote it by $u \cdot a \cdot q \cdot b \cdot v \mapsto u \cdot q' \cdot a \cdot c \cdot v$.

Special configurations

Start configuration

We assume that the head is on the left of the input in the beginning.

Therefore, $q_0 \cdot w$ is the start configuration.

Accepting configuration

Any configulation that contains q_{acc} is an accepting configuration.

Rejecting configuration

Any configulation that contains q_{rej} is a rejecting configuration.

Halting configurations: if a configuration is accepting or rejecting then it is called a halting configuration.

A TM may not halt!

Acceptance by a TM

A TM M is said to accept a word $w \in \Sigma^*$ if there exists a sequence of configurations $C_0, C_1, \ldots, C_k$ such that

C_0 is a start configuration,

$C_i \mapsto C_{i+1}$ for all $0 \leq i \leq k-1$,

C_k is an accepting configuration.

Acceptance by a TM

A TM M is said to accept a word $w \in \Sigma^*$ if there exists a sequence of configurations $C_0, C_1, \ldots, C_k$ such that

C_0 is a start configuration,

$C_i \mapsto C_{i+1}$ for all $0 \leq i \leq k-1$,

C_k is an accepting configuration.

The notion of rejection by TM is not as straightforward!

Turing recognizable languages

Definition

A language L is said to be Turing recognizable if there is a Turing machine M such that

Turing recognizable languages

Definition

A language L is said to be Turing recognizable if there is a Turing machine M such that $\forall w \in L$, M reaches an accepting configuration on w.

Turing recognizable languages

Definition

A language L is said to be Turing recognizable if there is a Turing machine M such that $\forall w \in L$, M reaches an accepting configuration on w.

We say that M recognizes L.

Turing recognizable languages

Definition

A language L is said to be Turing recognizable if there is a Turing machine M such that $\forall w \in L$, M reaches an accepting configuration on w.

We say that M recognizes L.

For words not in L

Turing recognizable languages

Definition

A language L is said to be Turing recognizable if there is a Turing machine M such that $\forall w \in L$, M reaches an accepting configuration on w.

We say that M recognizes L.

For words not in L

the machine may run forever,

or may reach q_{rej},

both are valid outcomes,

and the machine is allowed to do either of the two.

Turning decidable languages

Definition

A language L is said to be Turing decidable if there is a Turing machine M such that

Turning decidable languages

Definition

A language L is said to be Turing decidable if there is a Turing machine M such that

$$\forall w \in L,\ M \text{ reaches the accepting configuration on } w.$$

Turning decidable languages

Definition

A language L is said to be Turing decidable if there is a Turing machine M such that

$$\forall w \in L,\ M \text{ reaches the accepting configuration on } w.$$

$$\forall w \notin L,\ M \text{ reaches the rejecting configuration on } w.$$

Turning decidable languages

Definition

A language L is said to be Turing decidable if there is a Turing machine M such that

$$\forall w \in L,\ M \text{ reaches the accepting configuration on } w.$$

$$\forall w \notin L,\ M \text{ reaches the rejecting configuration on } w.$$

We say that M decides L.

Turning decidable languages

Definition

A language L is said to be Turing decidable if there is a Turing machine M such that

$$\forall w \in L,\ M \text{ reaches the accepting configuration on } w.$$

$$\forall w \notin L,\ M \text{ reaches the rejecting configuration on } w.$$

We say that M decides L.

If a language L is Turing decidable then

Turning decidable languages

Definition

A language L is said to be Turing decidable if there is a Turing machine M such that

$\forall w \in L$, M reaches the accepting configuration on w.

$\forall w \notin L$, M reaches the rejecting configuration on w.

We say that M decides L.

If a language L is Turing decidable then

the TM deciding L always halts.

Turning decidable languages

Definition

A language L is said to be Turing decidable if there is a Turing machine M such that

$\forall w \in L$, M reaches the accepting configuration on w.

$\forall w \notin L$, M reaches the rejecting configuration on w.

We say that M decides L.

If a language L is Turing decidable then

the TM deciding L always halts.

L is also Turing recognizable.

Turning decidable languages

Definition

A language L is said to be Turing decidable if there is a Turing machine M such that

$\forall w \in L$, M reaches the accepting configuration on w.

$\forall w \notin L$, M reaches the rejecting configuration on w.

We say that M decides L.

If a language L is Turing decidable then

the TM deciding L always halts.

L is also Turing recognizable.

Turing decidable languages form a subclass of Turing recognizable languages.

Comparing decidability and recognizability

Theorem

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

Comparing decidability and recognizability

Theorem

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

Proof.

$(\Rightarrow)$

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

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 (as we just saw).

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 (as we just saw).

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

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 (as we just saw).

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

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

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 (as we just saw).

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.

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 (as we just saw).

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.

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 (as we just saw).

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

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 (as we just saw).

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

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 (as we just saw).

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.

□

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.

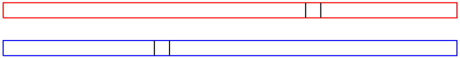
Proof sketch:

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.

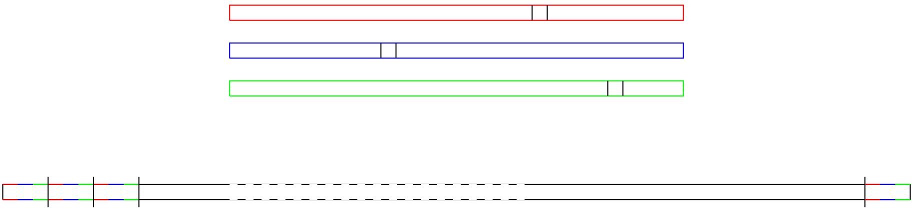
Proof sketch:

k-tape Turing machines

Theorem

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

Proof sketch:

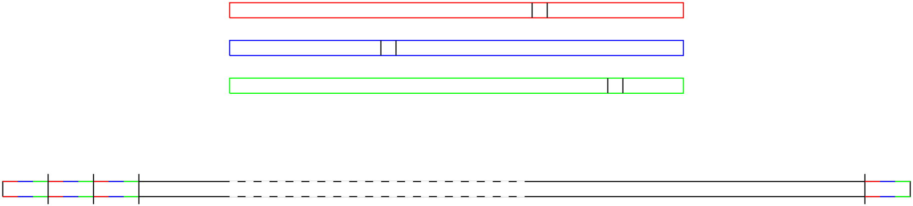


k-tape Turing machines

Theorem

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

Proof sketch:



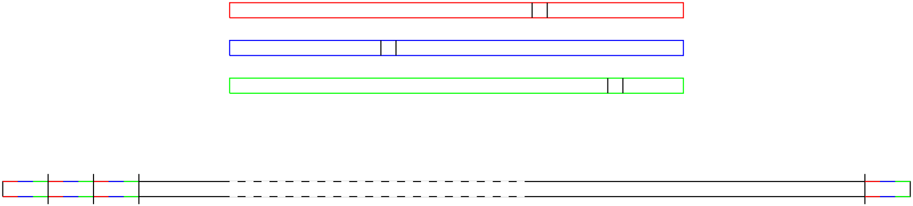
Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej},)$ be the k-tape Turing machine.

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.

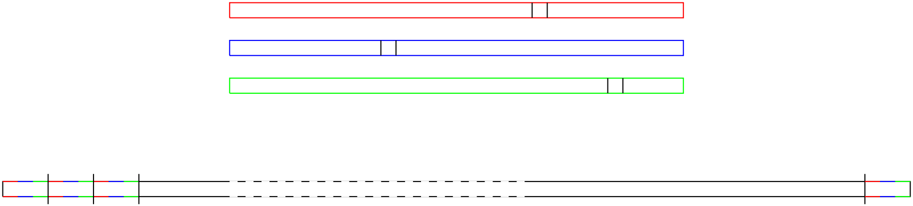
Let $M' = (Q', \Sigma, \Gamma', \delta', q_0, q_{acc}, q_{rej})$ be such that

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,

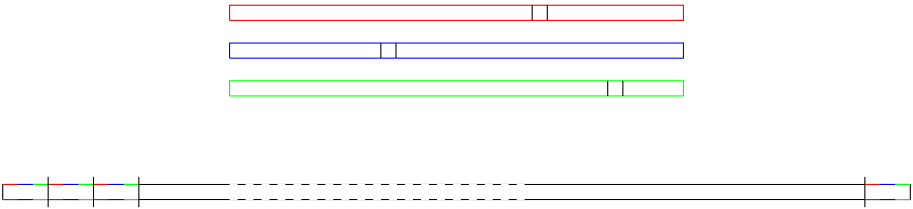
$\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$

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,

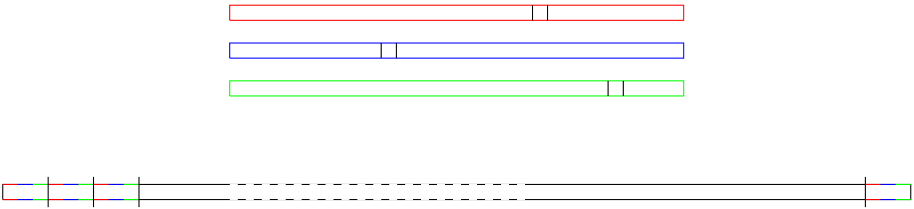
$\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$, $\Gamma = \Gamma \cup \overline{\Gamma} \cup \{\#\}$.

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.

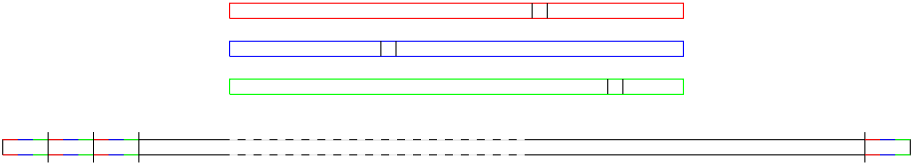
Proof sketch:

k-tape Turing machines

Theorem

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

Proof sketch:

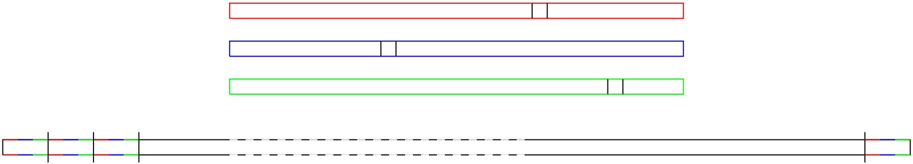


k-tape Turing machines

Theorem

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

Proof sketch:



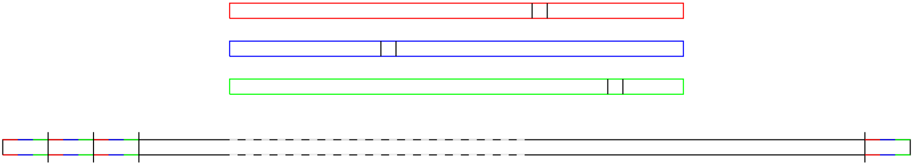
To simulate 1 step of M

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:

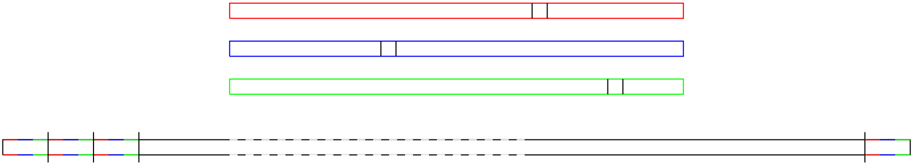
reads the tape left to right once, remembeing the marked symbols in its states

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,

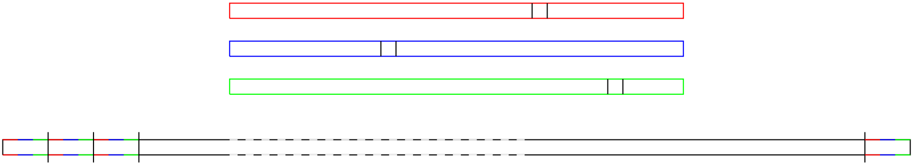
uses δ to determine the next state

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,

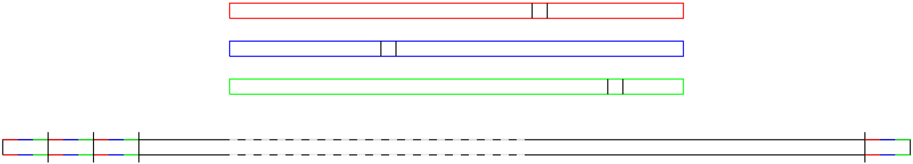
sweeps the input left to right again

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.