COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

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

March 19, 2023

Lecture 22:Turing Machines: Variants, CT Thesis

Recap

- Deterministic single-tape Turing Machines

Recap

- Deterministic single-tape Turing Machines

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

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

Recap

- Deterministic single-tape Turing Machines

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

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

- Configurations: start, accept, rejecting, halting.

Recap

- Deterministic single-tape Turing Machines

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

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

- Configurations: start, accept, rejecting, halting.
- Acceptance: Accepting vs rejecting configuration/run.
- L is Turing recognizable

Recap

- Deterministic single-tape Turing Machines

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

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

- Configurations: start, accept, rejecting, halting.
- Acceptance: Accepting vs rejecting configuration/run.
- L is Turing recognizable $\implies \exists M \forall w \in L$, ($M$ has an accepting run on w).

Recap

- Deterministic single-tape Turing Machines

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

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

- Configurations: start, accept, rejecting, halting.
- Acceptance: Accepting vs rejecting configuration/run.
- L is Turing recognizable $\implies \exists M \forall w \in L$, ($M$ has an accepting run on w).
- L is Turing decidable

Recap

- Deterministic single-tape Turing Machines

Definition

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

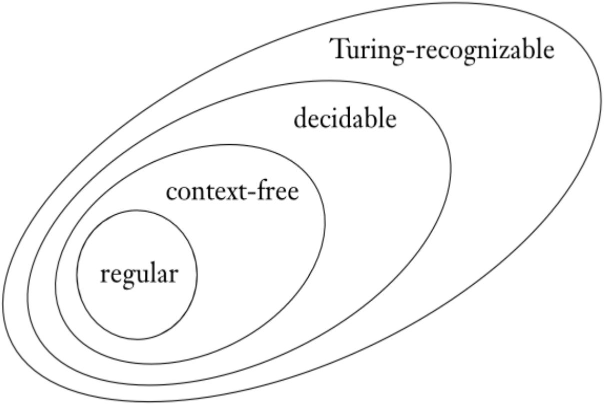
Q:	set of states	Σ:	input alphabet
q_0:	start state	Γ:	tape alphabet, $\Sigma \subseteq \Gamma$, $\& \in \Gamma$
q_{acc}:	accept state	q_{rej}:	reject state

$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$.

- Configurations: start, accept, rejecting, halting.
- Acceptance: Accepting vs rejecting configuration/run.
- L is Turing recognizable $\implies \exists M \forall w \in L$, ($M$ has an accepting run on w).
- L is Turing decidable $\implies \exists M(\forall w \in L,\ M$ has an accepting run on w) and ($\forall w \notin L, M$ has a rejecting run on w).

A hierarchy of languages

A hierarchy of 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

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

If L is Turing decidable, then $\overline{L}$ is also Turing decidable. (Needs proof.)

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

If L is Turing decidable, then $\overline{L}$ is also Turing decidable. (Needs proof.)

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

If L is Turing decidable, then $\overline{L}$ is also Turing decidable. (Needs proof.)

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

If L is Turing decidable, then $\overline{L}$ is also Turing decidable. (Needs proof.)

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

If L is Turing decidable, then $\overline{L}$ is also Turing decidable. (Needs proof.)

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

If L is Turing decidable, then $\overline{L}$ is also Turing decidable. (Needs proof.)

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.

Closure Properties

Theorem

Turing recognizable languages are closed under the following operations:

1. *Union*
2. *Intersection*
3. *Concatenation*
4. *Kleene closure*
5. *Homomorphism*

Closure Properties

Theorem

Turing recognizable languages are closed under the following operations:

1. *Union*
2. *Intersection*
3. *Concatenation*
4. *Kleene closure*
5. *Homomorphism*

Turing decidable languages are closed under the following operations:

1. *Union*
2. *Intersection*
3. *Complement*
4. *Concatenation*
5. *Kleene closure*

Robustness

- Robustness of a mathematical object (such as proof, definition, algorithm, method, etc.) is measured by its invariance to certain changes

Robustness

- Robustness of a mathematical object (such as proof, definition, algorithm, method, etc.) is measured by its invariance to certain changes
- To prove that a mathematical object is robust one needs to show that it is equivalent with its variants.

Robustness

- Robustness of a mathematical object (such as proof, definition, algorithm, method, etc.) is measured by its invariance to certain changes
- To prove that a mathematical object is robust one needs to show that it is equivalent with its variants.
- Are DFA/NFA/PDA robust?

Robustness

- Robustness of a mathematical object (such as proof, definition, algorithm, method, etc.) is measured by its invariance to certain changes
- To prove that a mathematical object is robust one needs to show that it is equivalent with its variants.
- Are DFA/NFA/PDA robust?
- Is the definition of a Turing machine robust?

Robustness

- Robustness of a mathematical object (such as proof, definition, algorithm, method, etc.) is measured by its invariance to certain changes
- To prove that a mathematical object is robust one needs to show that it is equivalent with its variants.
- Are DFA/NFA/PDA robust?
- Is the definition of a Turing machine robust?
- Variants of TM with multiple tapes or with nondeterminism abound.

Robustness

- Robustness of a mathematical object (such as proof, definition, algorithm, method, etc.) is measured by its invariance to certain changes
- To prove that a mathematical object is robust one needs to show that it is equivalent with its variants.
- Are DFA/NFA/PDA robust?
- Is the definition of a Turing machine robust?
- Variants of TM with multiple tapes or with nondeterminism abound.
- Original model of a TM and its variants all have the same computation power, i.e., they recognize the same class of languages.

Robustness

- Robustness of a mathematical object (such as proof, definition, algorithm, method, etc.) is measured by its invariance to certain changes
- To prove that a mathematical object is robust one needs to show that it is equivalent with its variants.
- Are DFA/NFA/PDA robust?
- Is the definition of a Turing machine robust?
- Variants of TM with multiple tapes or with nondeterminism abound.
- Original model of a TM and its variants all have the same computation power, i.e., they recognize the same class of languages.
- Hence, robustness of TM definition is measured by the invariance of its computation power to certain changes in design features of the machine.

A Toy Variant

Transition function of a TM in our definition forces the head to move to the left or right after each step.

A Toy Variant

Transition function of a TM in our definition forces the head to move to the left or right after each step. Let us vary the type of transition function permitted.

A Toy Variant

Transition function of a TM in our definition forces the head to move to the left or right after each step. Let us vary the type of transition function permitted.

- Suppose the head is allowed to stay put, i.e., $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$.
- Does this feature allow TM to recognize additional languages?

A Toy Variant

Transition function of a TM in our definition forces the head to move to the left or right after each step. Let us vary the type of transition function permitted.

- Suppose the head is allowed to stay put, i.e., $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$.
- Does this feature allow TM to recognize additional languages? **Answer:** NO.
- Sketch of proof:
  - An S transition can be represented by two transitions: one that move to the left followed by one that moves to the right.
  - Since we can convert a TM which stay put into one that does not have this facility the answer is No.

A Toy Variant

Transition function of a TM in our definition forces the head to move to the left or right after each step. Let us vary the type of transition function permitted.

- Suppose the head is allowed to stay put, i.e., $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$.
- Does this feature allow TM to recognize additional languages? **Answer:** NO.
- Sketch of proof:
  - An S transition can be represented by two transitions: one that move to the left followed by one that moves to the right.
  - Since we can convert a TM which stay put into one that does not have this facility the answer is No.
- To show that two models of TM are equivalent we need to show that we can simulate one by another.

A Toy Variant

Transition function of a TM in our definition forces the head to move to the left or right after each step. Let us vary the type of transition function permitted.

- Suppose the head is allowed to stay put, i.e.,
  $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$.
- Does this feature allow TM to recognize additional languages?
  Answer: NO.
- Sketch of proof:
  - An S transition can be represented by two transitions: one that move to the left followed by one that moves to the right.
  - Since we can convert a TM which stay put into one that does not have this facility the answer is No.
- To show that two models of TM are equivalent we need to show that we can simulate one by another.

Exercise: What about $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, S\}$

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.

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

Variants of Turing machines

k-tape Turing machines

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

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

Example

Variants of Turing machines

k-tape Turing machines

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

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

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.

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

Example

Given: 1^n on the input tape

k-tape Turing machines

Example

Given: 1^n on the input tape

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

0 While there is a 1 symbol on the first tape,

k-tape Turing machines

Example

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

0 While there is a 1 symbol on the first tape,
    0.1 Change the leftmost 1 symbol to X.

k-tape Turing machines

Example

Given: 1^n on the input tape

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

0 While there is a 1 symbol on the first tape,

- *0.1* Change the leftmost 1 symbol to X.
- *0.2* For each X or 1 symbol on the first tape

k-tape Turing machines

Example

Given: 1^n on the input tape

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

0 While there is a 1 symbol on the first tape,

  0.1 Change the leftmost 1 symbol to X.

  0.2 For each X or 1 symbol on the first tape
    write a 1 symbol on the second tape.

k-tape Turing machines

Example

Given: 1^n on the input tape

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

0 While there is a 1 symbol on the first tape,

- *0.1* Change the leftmost 1 symbol to X.
- *0.2* For each X or 1 symbol on the first tape
    write a 1 symbol on the second tape.
  end for

end while

k-tape Turing machines

Example

Given: 1^n on the input tape

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

0 While there is a 1 symbol on the first tape,

- *0.1* Change the leftmost 1 symbol to X.
- *0.2* For each X or 1 symbol on the first tape
  write a 1 symbol on the second tape.
  end for

end while

1 Copy the contents of the second tape on the first tape.

k-tape Turing machines

Example

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

0 While there is a 1 symbol on the first tape,
  0.1 Change the leftmost 1 symbol to X.
  0.2 For each X or 1 symbol on the first tape
      write a 1 symbol on the second tape.
    end for
  end while

1 Copy the contents of the second tape on the first tape.

2 Halt and accept.

k-tape Turing machines

Example

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

0 While there is a 1 symbol on the first tape,

0.1 Change the leftmost 1 symbol to X.

0.2 For each X or 1 symbol on the first tape
write a 1 symbol on the second tape.
end for

end while

1 Copy the contents of the second tape on the first tape.

2 Halt and accept.

k-tape Turing machines

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.

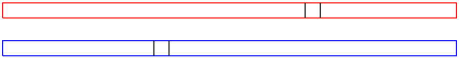
Proof sketch:

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.

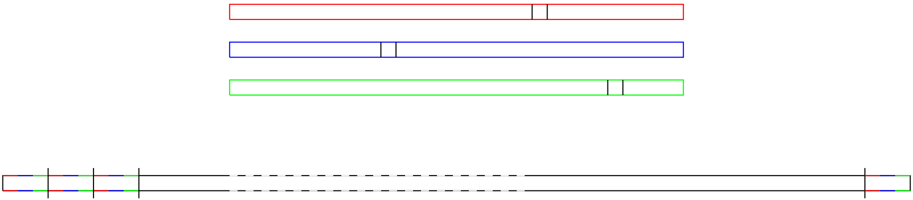
Proof sketch:

k-tape Turing machines

Theorem

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

Proof sketch:

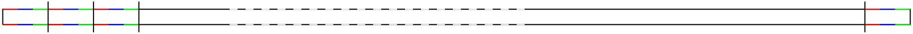


k-tape Turing machines

Theorem

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

Proof sketch:



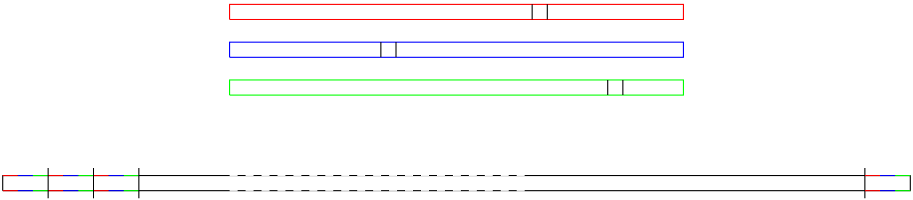
Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej},)$ be the k-tape Turing machine.

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.

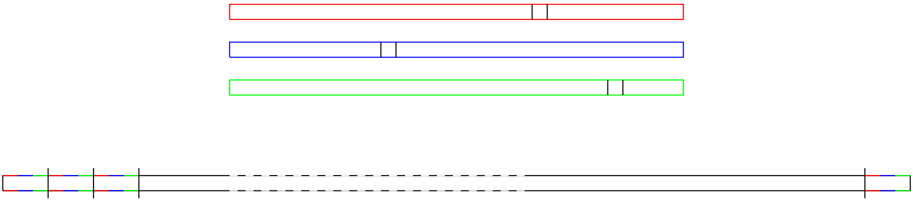
Let $M' = (Q', \Sigma, \Gamma', \delta', q_0, q_{acc}, q_{rej})$ be such that

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,

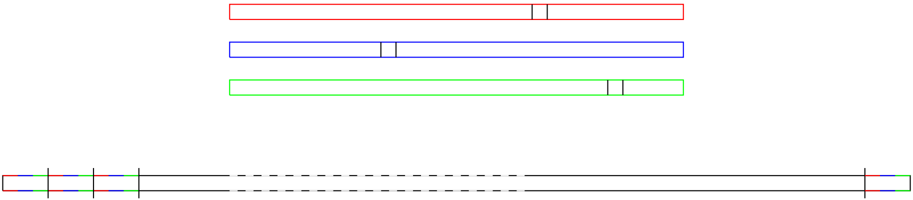
$\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$

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,

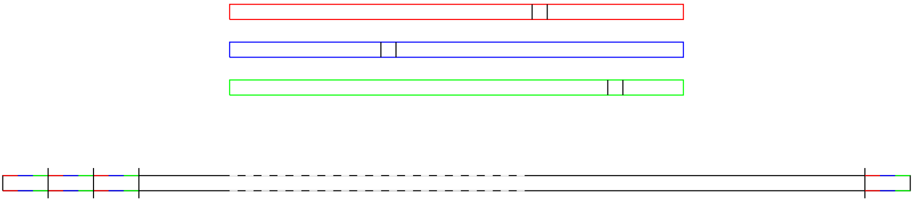
$\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$, $\Gamma' = \Gamma \cup \overline{\Gamma} \cup \{\#\}$.

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.

k-tape Turing machines

Theorem

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

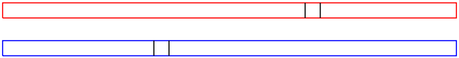
Proof sketch:

k-tape Turing machines

Theorem

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

Proof sketch:

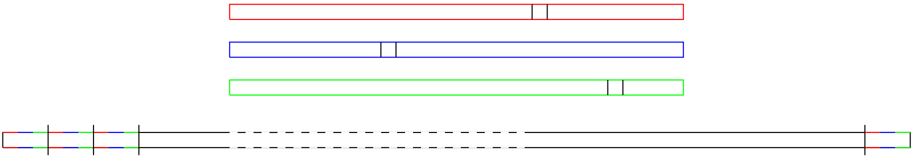


k-tape Turing machines

Theorem

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

Proof sketch:



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

- reads the tape left to right once, remembering the marked symbols in its states,
- uses δ to determine the next state,
- sweeps the input left to right again to update marked symbols.

k-tape Turing machines

Theorem

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

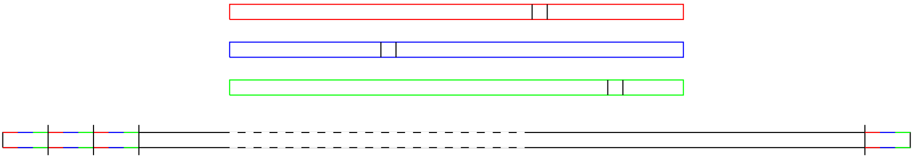
Proof sketch:

k-tape Turing machines

Theorem

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

Proof sketch:

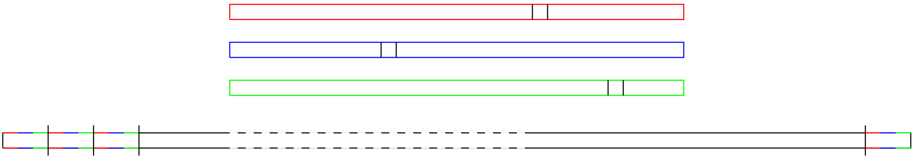


k-tape Turing machines

Theorem

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

Proof sketch:



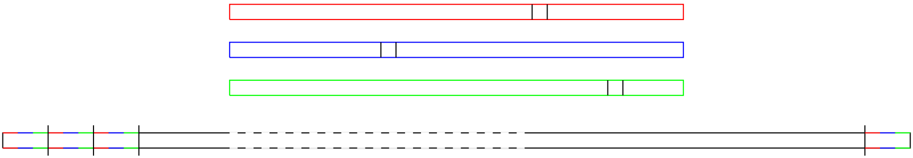
To simulate 1 step of M

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:

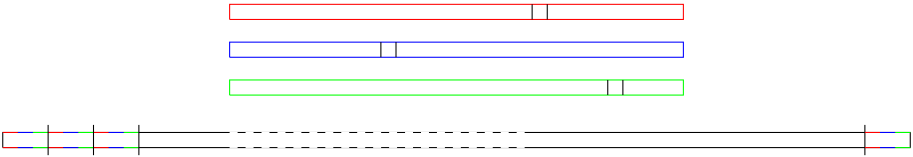
reads the tape left to right once, remembering the marked symbols in its states

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, remembering the marked symbols in its states,

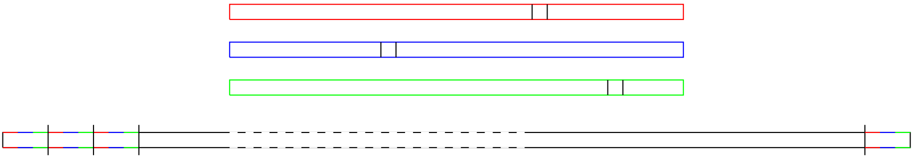
uses δ to determine the next state

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:

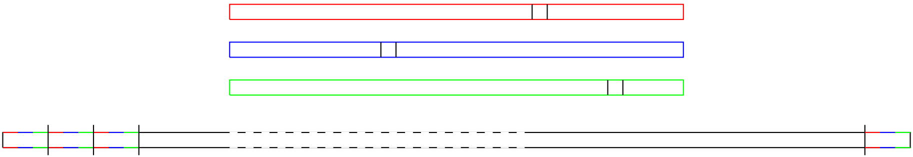
- reads the tape left to right once, remembering the marked symbols in its states,
- uses δ to determine the next state,
- sweeps the input left to right again

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, remembering the marked symbols in its states,

uses δ to determine the next state,

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