COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

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

March 20, 2023

Lecture 23:Turing Machines: Variants, CT Thesis (Part 2)

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.

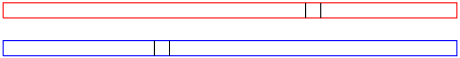
Proof sketch:

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.

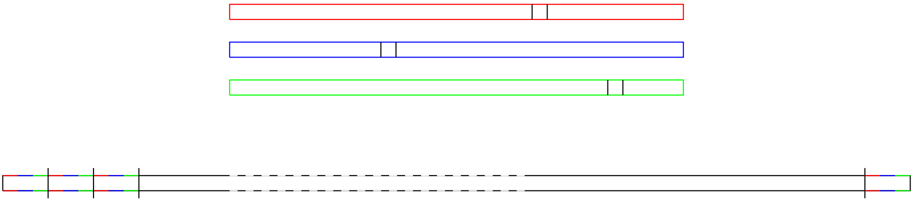
Proof sketch:

k-tape Turing machines

Theorem

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

Proof sketch:

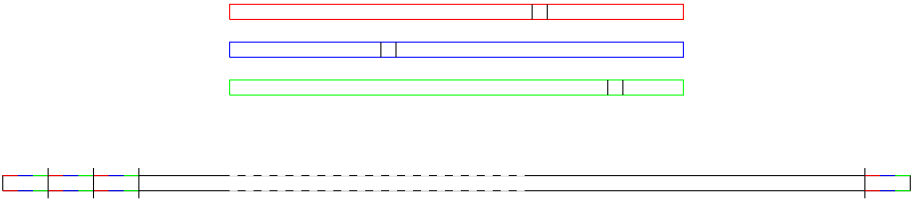


k-tape Turing machines

Theorem

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

Proof sketch:



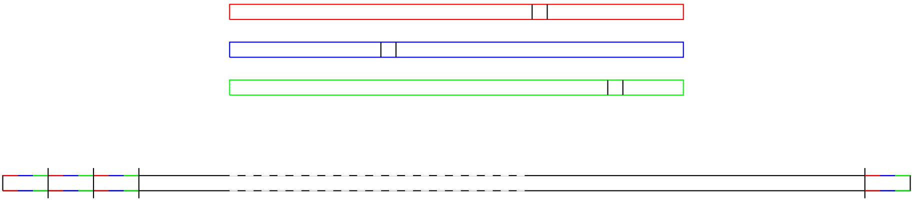
Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej},)$ be the k-tape Turing machine.

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.

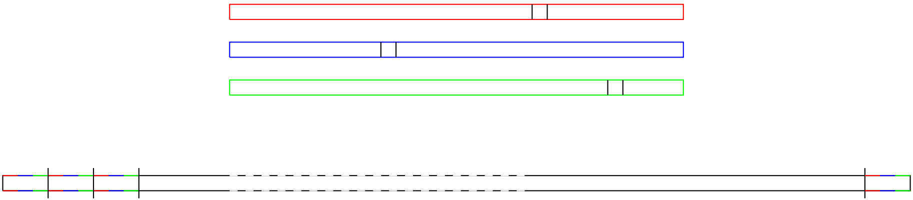
Let $M' = (Q', \Sigma, \Gamma', \delta', q_0, q_{acc}, q_{rej})$ be such that

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,

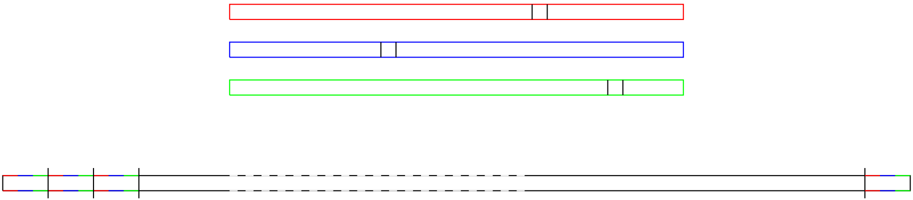
$\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$

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,

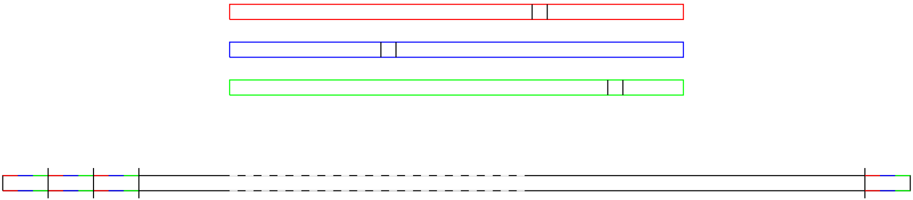
$\overline{\Gamma} = \{\overline{a} \mid a \in \Gamma\}$, $\Gamma' = \Gamma \cup \overline{\Gamma} \cup \{\#\}$.

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.

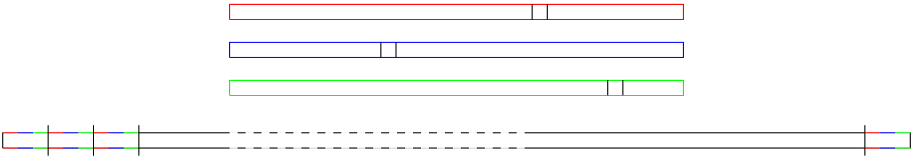
Proof sketch:

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.

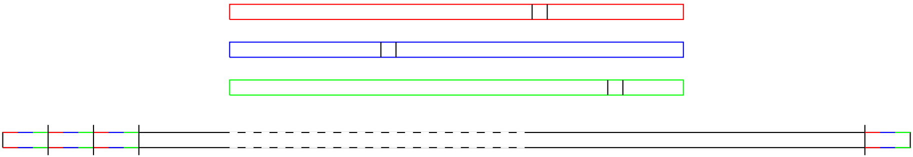
Proof sketch:

k-tape Turing machines

Theorem

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

Proof sketch:

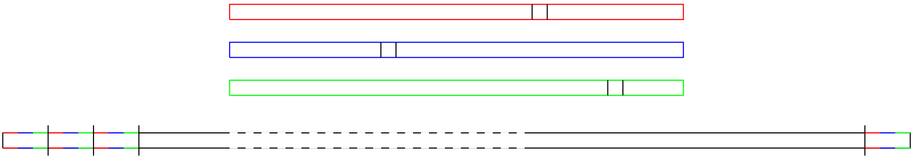


k-tape Turing machines

Theorem

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

Proof sketch:



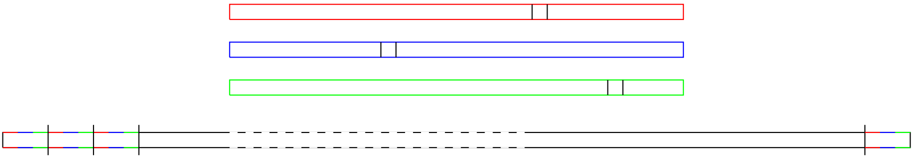
To simulate 1 step of M

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:

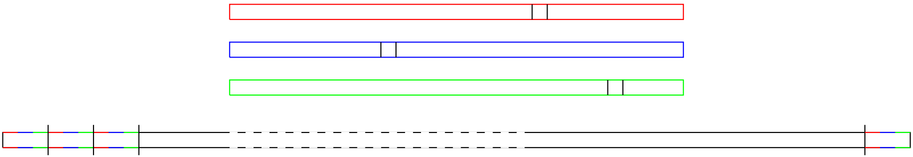
reads the tape left to right once, remembering the marked symbols in its states

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,

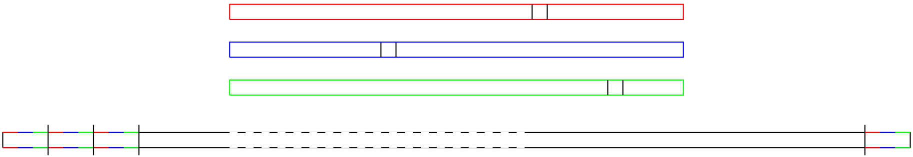
uses δ to determine the next state

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:

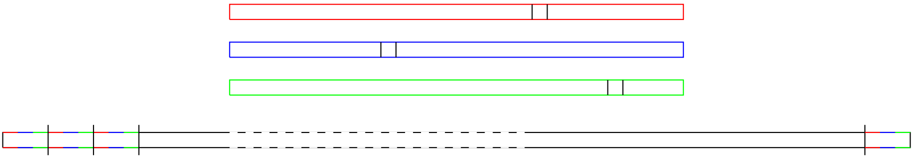
- reads the tape left to right once, remembering the marked symbols in its states,
- uses δ to determine the next state,
- sweeps the input left to right again

k-tape Turing machines

Theorem

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

Proof sketch:



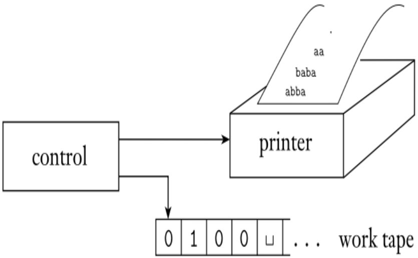
To simulate 1 step of M, M' works follows:

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

uses δ to determine the next state,

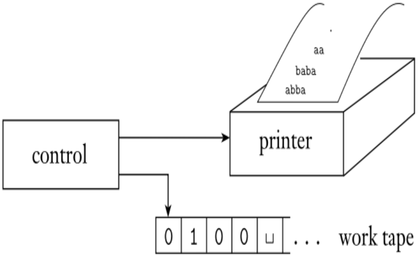
sweeps the input left to right again to update marked symbols.

Enumerators



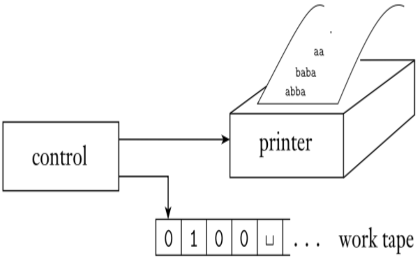
- Turing machine with an attached printer.
- **Exercise:** Formally define it!

Enumerators



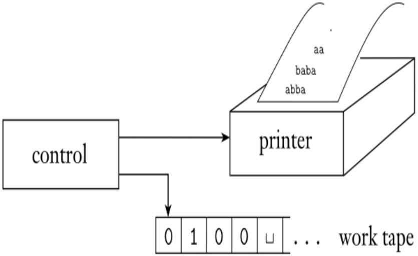
- Turing machine with an attached printer.
- **Exercise:** Formally define it!
- An enumerator E starts with a blank input on its work tape.
- If the enumerator doesn't halt, it may print an infinite list of strings.

Enumerators



- Turing machine with an attached printer.
- **Exercise:** Formally define it!
- An enumerator E starts with a blank input on its work tape.
- If the enumerator doesn't halt, it may print an infinite list of strings.
- The language enumerated by E is the collection of all the strings that it eventually prints out.

Enumerators



- Turing machine with an attached printer.
- **Exercise:** Formally define it!
- An enumerator E starts with a blank input on its work tape.
- If the enumerator doesn't halt, it may print an infinite list of strings.
- The language enumerated by E is the collection of all the strings that it eventually prints out.
- E may generate the strings of the language in any order, possibly with repetitions.

Enumerators vs Recognizers

Enumerators vs Recognizers

Theorem

A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof.

Enumerators vs Recognizers

Theorem

A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof.

($\Rightarrow$) On input w:

Enumerators vs Recognizers

Theorem

A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof.

($\Rightarrow$) On input w:

1. Run E. Every time that E outputs a string, compare it with w.

Enumerators vs Recognizers

Theorem

A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof.

($\Rightarrow$) On input w:

1. Run E. Every time that E outputs a string, compare it with w.
2. If w ever appears in the output of E, accept.

Enumerators vs Recognizers

Theorem

A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof.

($\Rightarrow$) On input w:

1. Run E. Every time that E outputs a string, compare it with w.
2. If w ever appears in the output of E, accept.

($\Leftarrow$)

Enumerators vs Recognizers

Theorem

A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof.

($\Rightarrow$) On input w:

1. Run E. Every time that E outputs a string, compare it with w.
2. If w ever appears in the output of E, accept.

($\Leftarrow$) Ignore the input. Repeat the following for $i = 1, 2, 3, \ldots$.

Enumerators vs Recognizers

Theorem

A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof.

($\Rightarrow$) On input w:

1. Run E. Every time that E outputs a string, compare it with w.
2. If w ever appears in the output of E, accept.

($\Leftarrow$) Ignore the input. Repeat the following for $i = 1, 2, 3, \ldots$.

1. Run M for i steps on each input, $s_1, s_2, \ldots, s_i$.

Enumerators vs Recognizers

Theorem

A language is Turing-recognizable if and only if some enumerator enumerates it.

Proof.

($\Rightarrow$) On input w:

1. Run E. Every time that E outputs a string, compare it with w.
2. If w ever appears in the output of E, accept.

($\Leftarrow$) Ignore the input. Repeat the following for $i = 1, 2, 3, \ldots$.

1. Run M for i steps on each input, $s_1, s_2, \ldots, s_i$.
2. If any computations accepts, print out the corresponding s_j.

$\square$

Remark: Turing Recognizable = Recursively Enumerable languages.