Fundamental Computer Science
Sequence 1: Turing Machines

Denis Trystram (inspired by Giorgio Lucarelli)

MoSIG1, 2020

Organization

Classes

- 30 hours in total
  (half Theory, half Exercises/Practice).
- 5 topics
  1. Universal Computing Model: the Turing Machine
  2. Alternative model: λ-Calculus
  3. NP-completeness
  4. Approximation Theory
  5. Introduction to Quantum Computing

Evaluation

- Exam: 70%
- Reading session: 30%

References

Books

- M. Garey and D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman
- Harry Lewis and Christos Papadimitriou, *Elements of the Theory of Computation*, Prentice-Hall
- Christos Papadimitriou, *Computational Complexity*, Pearson
- S. Arora and B. Barak, *Computational complexity – a modern approach*, Cambridge
- Vijay Vazirani, *Approximation Algorithms*, Springer
- Arnold Rosenberg, The pillars of Computation Theory, Springer

Agenda

Objective of the session

Present (and discuss) the universal computational model of **Turing machine**.

Preliminary

What is an *Algorithm*?

Preliminary

What is an *Algorithm*?
The first question is to discuss what can be calculated by a *Computer.*

Informally: this is a step-by-step procedure composed that solves a *problem*.

Preliminary

What is an *Algorithm*?
The first question is to discuss what can be calculated by a *Computer.*

Informally: this is a step-by-step procedure composed that solves a *problem*.

Desired properties

- clearly defined steps (formalization)
- efficiency (complexity – how many steps?)
- termination

Short History

- Etymology:
  - Al-Khwārizmī – a Persian mathematician of the 9th century
  - $\alpha\rho\iota\theta\mu\acute{o}\varsigma$ – the Greek word that means "number"

- Euclid's algorithm for computing the *greatest common divisor* (3rd century BC)
- End of 19th century - beginning of 20th century: mathematical formalizations (proof systems, axioms, etc). Is there an algorithm for any problem?
- Entscheidungsproblem (a challenge proposed by David Hilbert 1928): create an algorithm which is able to decide if a mathematical statement is true in a finite number of operations
- Church-Turing thesis (1930's): provides a formal definition of an algorithm (λ-calculus, Turing machine) and show that a solution to Entscheidungsproblem does not exist

Preliminaries

alphabet: a finite set of symbols

- examples: Roman alphabet $\{a, b, \ldots, z\}$, binary alphabet $\{0, 1\}$

Preliminaries

alphabet: a finite set of symbols

- examples: Roman alphabet $\{a, b, \ldots, z\}$, binary alphabet $\{0, 1\}$

string: a finite sequence of symbols over an alphabet

- examples: $science$, 0011101
- ϵ: the empty string
- Σ^*: the set of all strings over an alphabet Σ (including ϵ)

Preliminaries

alphabet: a finite set of symbols

- examples: Roman alphabet $\{a, b, \ldots, z\}$, binary alphabet $\{0, 1\}$

string: a finite sequence of symbols over an alphabet

- examples: $science$, 0011101
- ϵ: the empty string
- Σ^*: the set of all strings over an alphabet Σ (including ϵ)

language: a set strings over an alphabet Σ (i.e., a subset of Σ^*)

- examples: $\emptyset$, Σ, Σ^*
- more examples:

  $L = \{w \in \Sigma^* : w \text{ has some property } P\}$

  $L = \{w \in \Sigma^* : w = w^R\}$ $\quad (w^R = \text{reverse of } w)$

  $L = \{w \in \{0, 1\}^* : w \text{ has an equal number of 0's and 1's}\}$

Preliminaries

alphabet: a finite set of symbols

- examples: Roman alphabet $\{a, b, \ldots, z\}$, binary alphabet $\{0, 1\}$

string: a finite sequence of symbols over an alphabet

- examples: $science$, 0011101
- ϵ: the empty string
- Σ^*: the set of all strings over an alphabet Σ (including ϵ)

language: a set strings over an alphabet Σ (i.e., a subset of Σ^*)

- examples: $\emptyset$, Σ, Σ^*
- more examples:

  $L = \{w \in \Sigma^* : w \text{ has some property } P\}$
  $L = \{w \in \Sigma^* : w = w^R\}$ $\quad (w^R = \text{reverse of } w)$
  $L = \{w \in \{0, 1\}^* : w \text{ has an equal number of 0's and 1's}\}$
  $L = \{w \in \{1, 2, \ldots, n\} : w \text{ is a permutation of } \{1, 2, \ldots, n\}$
  corresponding to a Hamiltonian Path$\}$

Preliminaries

Define first what is a *problem*.
An id, the list of input (with their coding) and a question.

Preliminaries

Define first what is a *problem*.
An id, the list of input (with their coding) and a question.

Decision problem: a problem that can be posed as an yes/no question.

Preliminaries

Define first what is a *problem*.
An id, the list of input (with their coding) and a question.

Decision problem: a problem that can be posed as an yes/no question.

- example:
  Prime
  Given a integer n
  Is n a prime?

Preliminaries

Define first what is a *problem*.
An id, the list of input (with their coding) and a question.

Decision problem: a problem that can be posed as an yes/no question.

- example:
  Prime
  Given a integer n
  Is n a prime?

- another example:
  - Given a graph $G = (V, E)$, is there a permutation π of the vertex set such that $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$ for all i, $1 \leq i \leq |V - 1|$?

Preliminaries

Define first what is a *problem*.
An id, the list of input (with their coding) and a question.

Decision problem: a problem that can be posed as an yes/no question.

- example:
  Prime
  Given a integer n
  Is n a prime?

- another example:
  - Given a graph $G = (V, E)$, is there a permutation π of the vertex set such that $(v_{\pi(i)}, v_{\pi(i+1)}) \in E$ for all i, $1 \leq i \leq |V - 1|$?
    (Hamiltonian Path)

Beyond decision problems

Optimization: a problem of searching for the best answer

Beyond decision problems

Optimization: a problem of searching for the best answer

- example: Given a graph $G = (V, E)$, two vertices $s, t \in V$ and an integer distance $d(e)$ for each $e \in E$, find the path p between s and t such that the sum of distances of the edges in p is minimized.

Beyond decision problems

Optimization: a problem of searching for the best answer

- example: Given a graph $G = (V, E)$, two vertices $s, t \in V$ and an integer distance $d(e)$ for each $e \in E$, find the path p between s and t such that the sum of distances of the edges in p is minimized.
- decision version: Given a graph $G = (V, E)$, two vertices $s, t \in V$, an integer distance $d(e)$ for each $e \in E$ **and an integer** D, is there a path p between s and t such that the sum of distances of the edges in p is at most D?

Preliminaries

Observation 1:

In most of these lectures we will deal with decision problems

Preliminaries

Observation 1:
In most of these lectures we will deal with decision problems

Observation 2:
A decision problem is defined by the input and the yes/no question

- examples of input:
  - Given a set of numbers $A = \{a_1, a_2, \ldots, a_n\}$
  - Given a graph $G = (V, E)$
  - Given a graph $G = (V, E)$ and a positive weight $w(e)$ for each $e \in E$

Preliminaries

Observation 1:
In most of these lectures we will deal with decision problems

Observation 2:
A decision problem is defined by the input and the yes/no question

- examples of input:
  - Given a set of numbers $A = \{a_1, a_2, \ldots, a_n\}$
  - Given a graph $G = (V, E)$
  - Given a graph $G = (V, E)$ and a positive weight $w(e)$ for each $e \in E$
- $< I >$: string encoding of the input
  - $< a_1, a_2, \ldots, a_n >$
  - $<$ adjacency matrix of $G >$
  - $<$ adjacency matrix of $G, w(e)\ \forall e \in E >$

Preliminaries

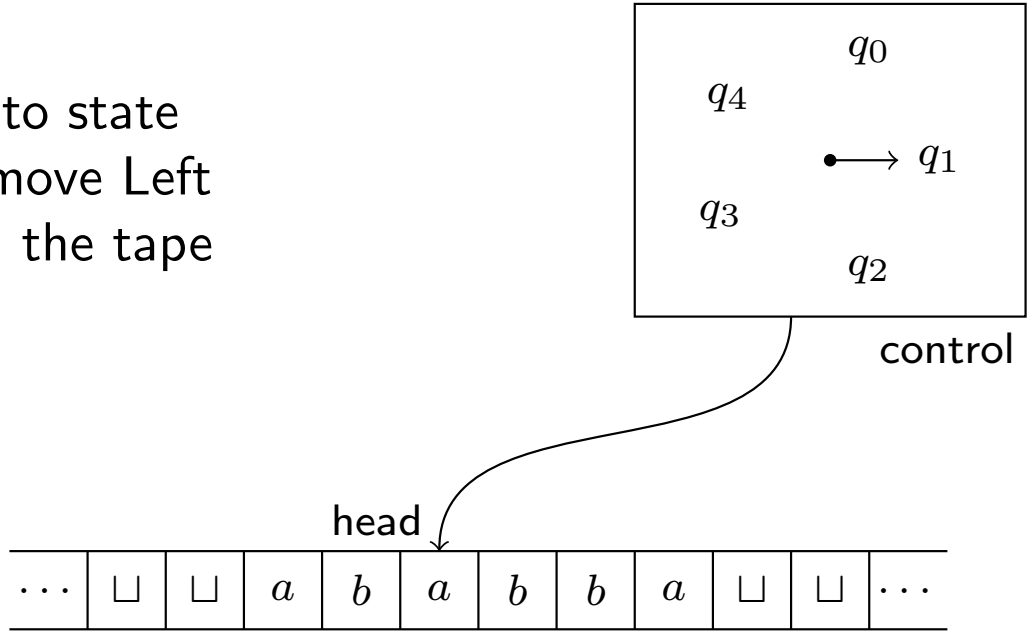
Observation 1:
In most of these lectures we will deal with decision problems

Observation 2:
A decision problem is defined by the input and the yes/no question

- examples of input:
  - Given a set of numbers $A = \{a_1, a_2, \ldots, a_n\}$
  - Given a graph $G = (V, E)$
  - Given a graph $G = (V, E)$ and a positive weight $w(e)$ for each $e \in E$
- $< I >$: string encoding of the input
  - $< a_1, a_2, \ldots, a_n >$
  - $<$ adjacency matrix of $G >$
  - $<$ adjacency matrix of $G, w(e)\ \forall e \in E >$
- $|I|$: size of the input (in binary)
  - $\log_2 a_1 + \log_2 a_2 + \ldots \log_2 a_n$
  - $|V|^2$
  - $|V|^2 + \sum_{e \in E} \log_2 w(e)$

Turing machine

- memory: an infinite tape
  - initially, it contains the input string
  - move the head left or right
  - read and/or write to current cell
- control states
  - finite number of them
  - one current state
- At each step:
  – move from state to state
  – read or write or move Left or move Right in the tape



Turing machine: formal definition

A Turing Machine (M) is a six-tuple $(K, \Sigma, \Gamma, \delta, s, H)$, where

- K is a finite set of states
- Σ is the input alphabet not containing the *blank* symbol $\sqcup$
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $s \in K$: the initial state
- $H \subseteq K$: the set of halting states
- δ: the transition function from $(K \setminus H) \times \Gamma$ to $K \times (\Gamma \cup \{\leftarrow, \rightarrow\})$

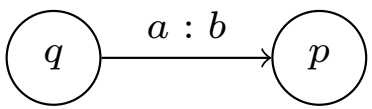
Turing machine: formal definition

A Turing Machine (M) is a six-tuple $(K, \Sigma, \Gamma, \delta, s, H)$, where

- K is a finite set of states
- Σ is the input alphabet not containing the *blank* symbol $\sqcup$
- Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $s \in K$: the initial state
- $H \subseteq K$: the set of halting states
- δ: the transition function from $(K \setminus H) \times \Gamma$ to $K \times (\Gamma \cup \{\leftarrow, \rightarrow\})$

In general, $\delta(q, a) = (p, b)$ means that when M is in the state q and reads a in the tape, it goes to the state p and

– if $b \in \Sigma$, writes b in the place of a
– if $b \in \{\leftarrow, \rightarrow\}$, moves the head either Left or Right



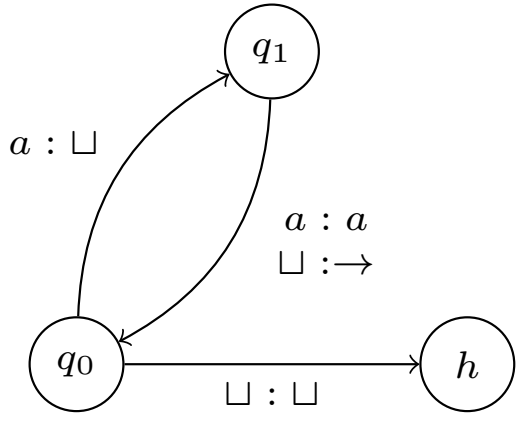
A first example

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



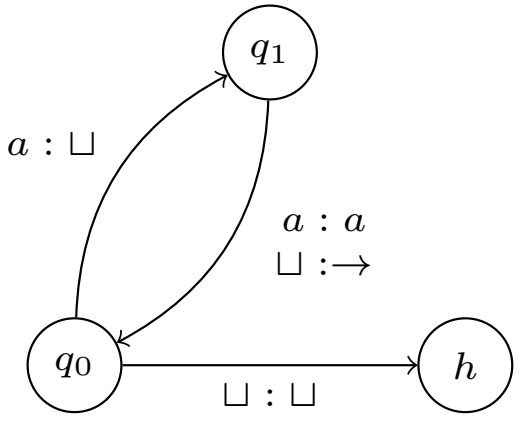
A first example

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$(q_0, \underline{a}aa)$$

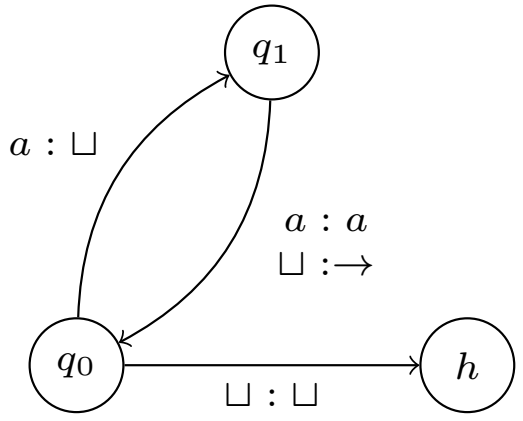
A first example

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$(q_0, \underline{a}aa) \quad \vdash_M \quad (q_1, \underline{\sqcup}aa)$$

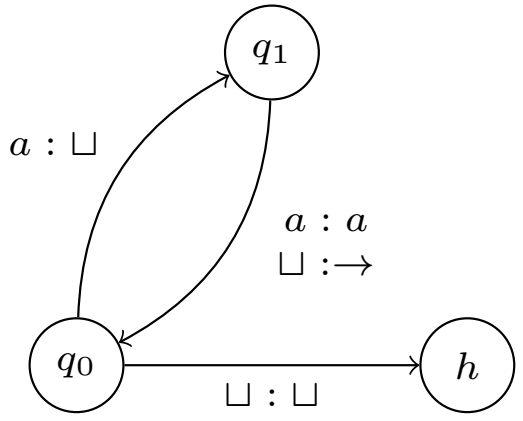
A first example

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$(q_0, \underline{a}aa) \vdash_M (q_1, \underline{\sqcup}aa) \vdash_M (q_0, \sqcup\underline{a}a)$$

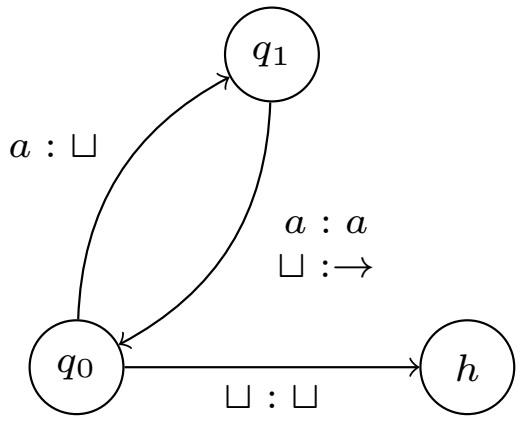
A first example

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$(q_0, \underline{a}aa) \vdash_M (q_1, \underline{\sqcup}aa) \vdash_M (q_0, \sqcup\underline{a}a)$$
$$\vdash_M (q_1, \sqcup\underline{\sqcup}a)$$

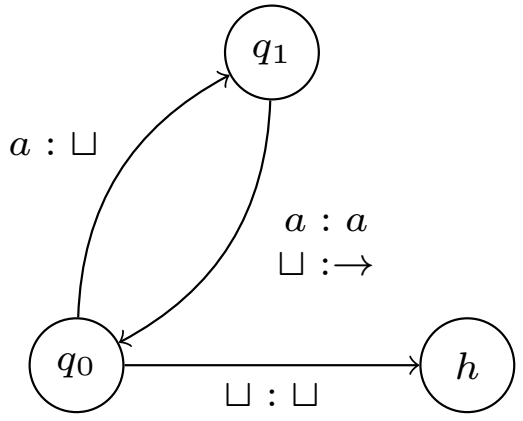
A first example

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$\begin{aligned}(q_0, \underline{a}aa) &\vdash_M (q_1, \underline{\sqcup}aa) \vdash_M (q_0, \sqcup\underline{a}a) \\ &\vdash_M (q_1, \sqcup\underline{\sqcup}a) \vdash_M (q_0, \sqcup \sqcup \underline{a})\end{aligned}$$

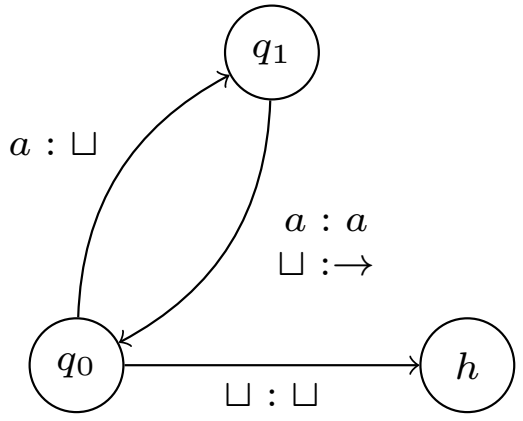
A first example

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$\begin{aligned}
(q_0, \underline{a}aa) &\vdash_M (q_1, \underline{\sqcup}aa) \vdash_M (q_0, \sqcup\underline{a}a) \\
&\vdash_M (q_1, \sqcup\underline{\sqcup}a) \vdash_M (q_0, \sqcup\sqcup\underline{a}) \\
&\vdash_M (q_1, \sqcup\sqcup\underline{\sqcup})
\end{aligned}$$

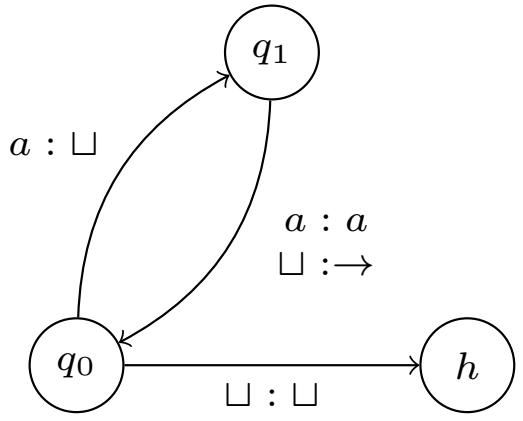
A first example

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$\begin{aligned}
(q_0, \underline{a}aa) &\vdash_M (q_1, \underline{\sqcup}aa) \vdash_M (q_0, \sqcup\underline{a}a) \\
&\vdash_M (q_1, \sqcup\underline{\sqcup}a) \vdash_M (q_0, \sqcup\sqcup\underline{a}) \\
&\vdash_M (q_1, \sqcup\sqcup\underline{\sqcup}) \vdash_M (q_0, \sqcup\sqcup\sqcup\underline{\sqcup})
\end{aligned}$$

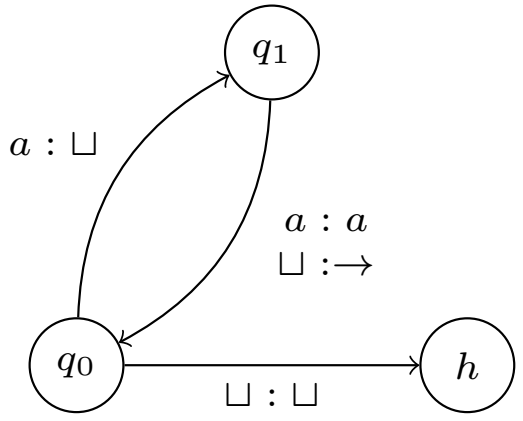
A first example

Consider the Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ where

$$K = \{q_0, q_1, h\}, \quad \Sigma = \{a\}, \quad \Gamma = \{a, \sqcup\}, \quad s = q_0, \quad H = \{h\},$$

and δ is given by the table. How does M proceed?

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_1	a	(q_0, a)
q_1	$\sqcup$	$(q_0, \rightarrow)$



$$\begin{aligned}
(q_0, \underline{a}aa) &\vdash_M (q_1, \underline{\sqcup}aa) \vdash_M (q_0, \sqcup\underline{a}a) \\
&\vdash_M (q_1, \sqcup\underline{\sqcup}a) \vdash_M (q_0, \sqcup\sqcup\underline{a}) \\
&\vdash_M (q_1, \sqcup\sqcup\underline{\sqcup}) \vdash_M (q_0, \sqcup\sqcup\sqcup\underline{\sqcup}) \\
&\vdash_M (h, \sqcup\sqcup\sqcup\underline{\sqcup})
\end{aligned}$$

Formalize the notation

Definition

A **configuration** of a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a member of $K \times \Gamma^* \times \Gamma^*((\Gamma \setminus \{\sqcup\}) \cup \{\epsilon\})$.

- informally: a triplet describing
  - the current state
  - the contents of the tape on the left of the head (including head's position)
  - the contents of the tape on the right of the head

Formalize the notation

Definition

A **configuration** of a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a member of $K \times \Gamma^* \times \Gamma^*((\Gamma \setminus \{\sqcup\}) \cup \{\epsilon\})$.

- informally: a triplet describing
  - the current state
  - the contents of the tape on the left of the head (including head's position)
  - the contents of the tape on the right of the head
- example: $(q_1, \sqcup a, a)$ or simply $(q_1, \sqcup \underline{a}a)$ or simply $(q_1, \underline{a}a)$

Formalize the notation

Definition

A **configuration** of a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a member of $K \times \Gamma^* \times \Gamma^*((\Gamma \setminus \{\sqcup\}) \cup \{\epsilon\})$.

- informally: a triplet describing
  - the current state
  - the contents of the tape on the left of the head (including head's position)
  - the contents of the tape on the right of the head
- example: $(q_1, \sqcup a, a)$ or simply $(q_1, \sqcup \underline{a}a)$ or simply $(q_1, \underline{a}a)$

Initial configuration: $(s, \underline{a}w)$ where $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a Turing Machine, $a \in \Sigma$, $w \in \Sigma^*$ and aw is the *input string*

Formalize the notation

Definition

A **configuration** of a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a member of $K \times \Gamma^* \times \Gamma^*((\Gamma \setminus \{\sqcup\}) \cup \{\epsilon\})$.

- informally: a triplet describing
  - the current state
  - the contents of the tape on the left of the head (including head's position)
  - the contents of the tape on the right of the head
- example: $(q_1, \sqcup a, a)$ or simply $(q_1, \sqcup \underline{a}a)$ or simply $(q_1, \underline{a}a)$

Initial configuration: $(s, \underline{a}w)$ where $M = (K, \Sigma, \Gamma, \delta, s, H)$ is a Turing Machine, $a \in \Sigma$, $w \in \Sigma^*$ and aw is the *input string*

Halted configuration: a configuration whose state belongs to H

- example: $(h, \sqcup \sqcup \sqcup\sqcup, \epsilon)$ or simply $(h, \sqcup \sqcup \sqcup\underline{\sqcup})$ or simply $(h, \underline{\sqcup})$

Formalize the notation

Definition

Consider a Turing Machine M and two configurations C_1 and C_2 of M. If M can go from C_1 to C_2 in a *single step*, then we write

$$C_1 \vdash_M C_2$$

Formalize the notation

Definition

Consider a Turing Machine M and two configurations C_1 and C_2 of M. If M can go from C_1 to C_2 in a *single step*, then we write

$$C_1 \vdash_M C_2$$

Definition

Consider a Turing Machine M and two configurations C_1 and C_2 of M. If M can go from C_1 to C_2 using a *sequence* of configurations, then we say that C_1 **yields** C_2 and we write

$$C_1 \vdash_M^* C_2$$

Formalize the notation

Definition

Consider a Turing Machine M and two configurations C_1 and C_2 of M. If M can go from C_1 to C_2 in a *single step*, then we write

$$C_1 \vdash_M C_2$$

Definition

Consider a Turing Machine M and two configurations C_1 and C_2 of M. If M can go from C_1 to C_2 using a *sequence* of configurations, then we say that C_1 **yields** C_2 and we write

$$C_1 \vdash_M^* C_2$$

Definition

A **computation** of a Turing Machine M is a sequence of configurations $C_0, C_1, \ldots, C_n$, for some $n \geq 0$, such that

$$C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_n$$

The **length** of the computation is n (or it performs n steps).

Determinism or not?

Definition

Implicitly, the transition δ is deterministic.

Determinism or not?

Definition

Implicitly, the transition δ is deterministic.

Non-deterministic Turing Machine

What happens is several outputs are allowed at each step?

The choice is among k fixed possibilities, random, round-robin, etc.

Determinism or not?

Definition

Implicitly, the transition δ is deterministic.

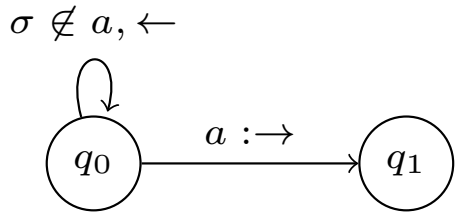
Non-deterministic Turing Machine

What happens is several outputs are allowed at each step?

The choice is among k fixed possibilities, random, round-robin, etc.

This point will be detailed in the next lecture.

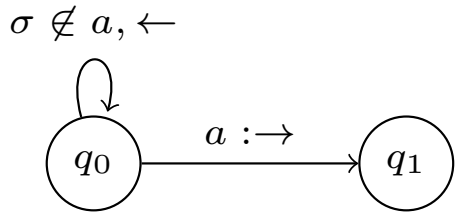
A more general notation for Turing Machines



Turing Machine $L_a = (K, \Sigma, \Gamma, \delta, s, H)$ where:

- $K = \{q_0, q_1\}$
- $a \in \Sigma$
- $s = q_0$
- $H = \{q_1\}$

A more general notation for Turing Machines

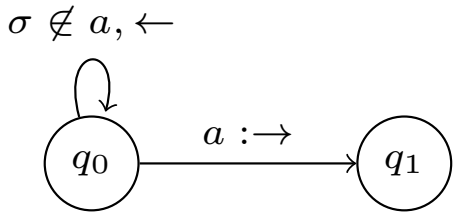


Turing Machine $L_a = (K, \Sigma, \Gamma, \delta, s, H)$ where:

- $K = \{q_0, q_1\}$
- $a \in \Sigma$
- $s = q_0$
- $H = \{q_1\}$

- Define similar simple Turing Machines
  - examples: L, R, L_a, R_a, L^2, R^2, a, $\sqcup$, etc

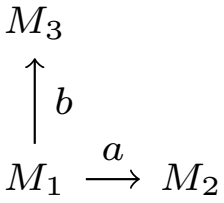
A more general notation for Turing Machines



Turing Machine $L_a = (K, \Sigma, \Gamma, \delta, s, H)$ where:

- $K = \{q_0, q_1\}$
- $a \in \Sigma$
- $s = q_0$
- $H = \{q_1\}$

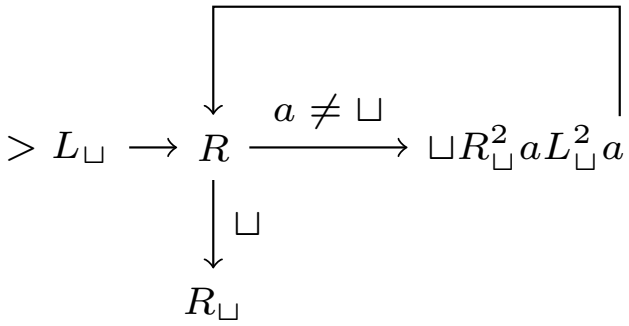
- Define similar simple Turing Machines
  - examples: L, R, L_a, R_a, L^2, R^2, a, $\sqcup$, etc
- Combine simple machines to construct more complicated ones



1. Run M_1
2. If M_1 finishes and the head reads a then run M_2 starting from this a
3. Else run M_3 starting from this b

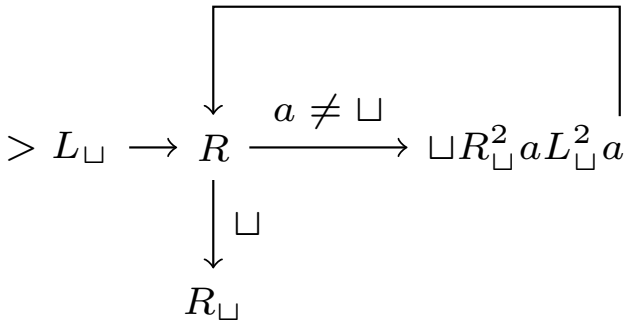
Example

What is the goal of the following Turing Machine?



Example

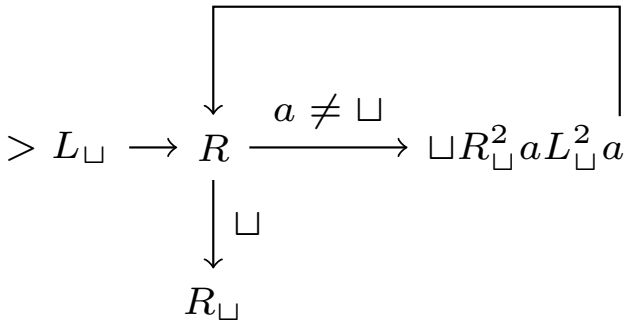
What is the goal of the following Turing Machine?



$(\sqcup abc\underline{\sqcup}) \quad \vdash^*_M \quad (\underline{\sqcup} abc\sqcup) \qquad (L_\sqcup)$

Example

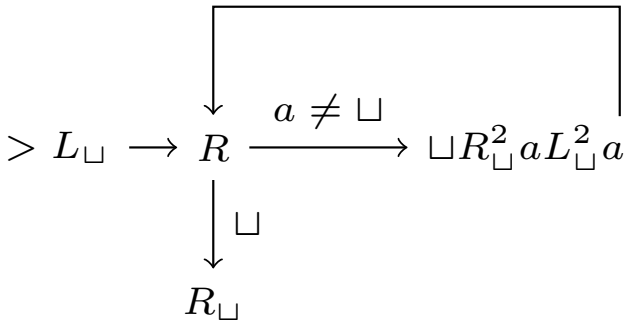
What is the goal of the following Turing Machine?



$$
\begin{array}{llll}
(\sqcup abc\underline{\sqcup}) & \vdash_M^* & (\underline{\sqcup} abc\sqcup) & (L_\sqcup) \\
 & \vdash_M & (\sqcup \underline{a} bc\sqcup) & (R)
\end{array}
$$

Example

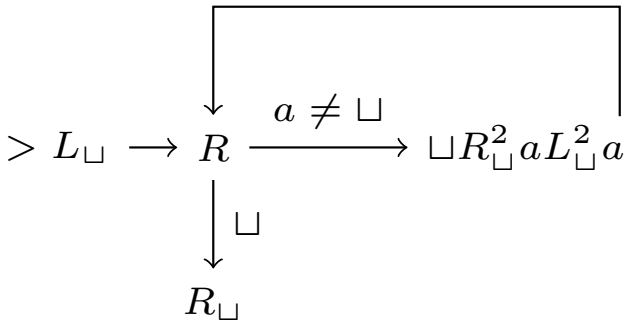
What is the goal of the following Turing Machine?



$$
\begin{array}{lll}
(\sqcup abc\underline{\sqcup}) & \vdash_M^* & (\underline{\sqcup} abc\sqcup) \qquad (L_\sqcup) \\
 & \vdash_M & (\sqcup \underline{a} bc\sqcup) \qquad (R) \\
 & \vdash_M & (\sqcup\underline{\sqcup} bc\sqcup) \qquad (\sqcup)
\end{array}
$$

Example

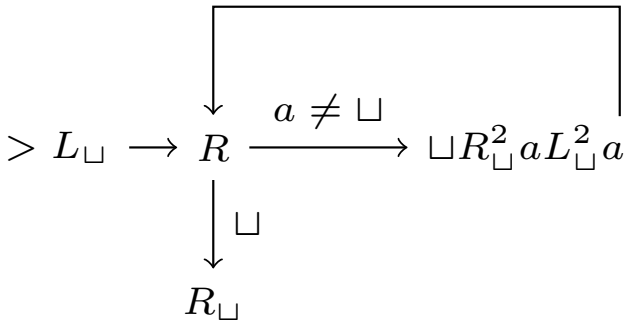
What is the goal of the following Turing Machine?



$$
\begin{array}{lll}
(\sqcup abc\underline{\sqcup}) & \vdash_M^* & (\underline{\sqcup} abc\sqcup) \qquad (L_\sqcup) \\
& \vdash_M & (\sqcup \underline{a} bc\sqcup) \qquad (R) \\
& \vdash_M & (\sqcup\underline{\sqcup} bc\sqcup) \qquad (\sqcup) \\
& \vdash_M^* & (\sqcup \sqcup bc \sqcup \underline{\sqcup}) \qquad (R^2_\sqcup)
\end{array}
$$

Example

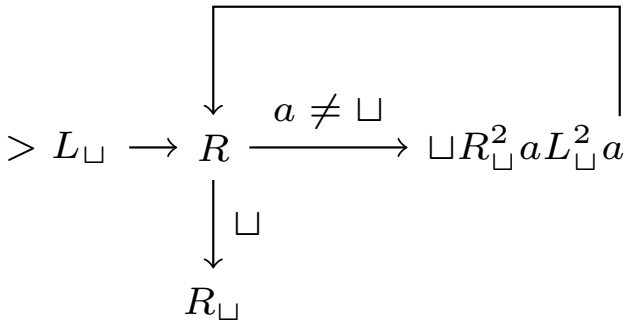
What is the goal of the following Turing Machine?



$$
\begin{array}{lll}
(\sqcup abc\underline{\sqcup}) & \vdash^*_M & (\underline{\sqcup} abc\sqcup) \quad (L_\sqcup) \\
& \vdash_M & (\sqcup \underline{a} bc\sqcup) \quad (R) \\
& \vdash_M & (\sqcup\underline{\sqcup} bc\sqcup) \quad (\sqcup) \\
& \vdash^*_M & (\sqcup \sqcup bc \sqcup \underline{\sqcup}) \quad (R^2_\sqcup) \\
& \vdash_M & (\sqcup \sqcup bc \sqcup \underline{a}) \quad (a)
\end{array}
$$

Example

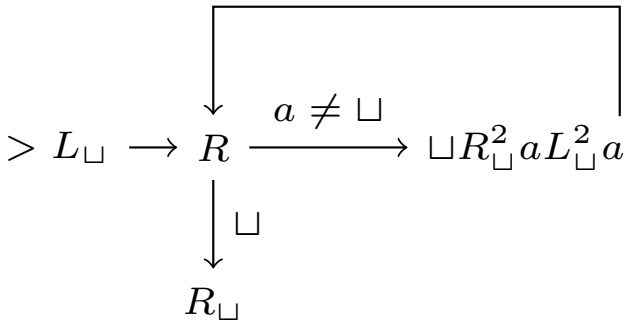
What is the goal of the following Turing Machine?



$$
\begin{array}{lll}
(\sqcup abc\underline{\sqcup}) & \vdash^*_M & (\underline{\sqcup} abc\sqcup) \quad (L_\sqcup) \\
& \vdash_M & (\sqcup \underline{a} bc\sqcup) \quad (R) \\
& \vdash_M & (\sqcup\underline{\sqcup} bc\sqcup) \quad (\sqcup) \\
& \vdash^*_M & (\sqcup \sqcup bc \sqcup \underline{\sqcup}) \quad (R^2_\sqcup) \\
& \vdash_M & (\sqcup \sqcup bc \sqcup \underline{a}) \quad (a) \\
& \vdash^*_M & (\sqcup\underline{\sqcup} bc \sqcup a) \quad (L^2_\sqcup)
\end{array}
$$

Example

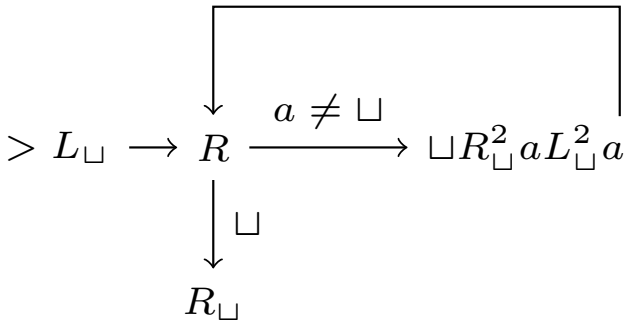
What is the goal of the following Turing Machine?



$$
\begin{array}{lll}
(\sqcup abc\underline{\sqcup}) & \vdash^*_M & (\underline{\sqcup} abc\sqcup) \qquad (L_\sqcup) \\
& \vdash_M & (\sqcup \underline{a} bc\sqcup) \qquad (R) \\
& \vdash_M & (\sqcup\underline{\sqcup} bc\sqcup) \qquad (\sqcup) \\
& \vdash^*_M & (\sqcup \sqcup bc \sqcup \underline{\sqcup}) \qquad (R^2_\sqcup) \\
& \vdash_M & (\sqcup \sqcup bc \sqcup \underline{a}) \qquad (a) \\
& \vdash^*_M & (\sqcup\underline{\sqcup} bc \sqcup a) \qquad (L^2_\sqcup) \\
& \vdash_M & (\sqcup \underline{a} bc \sqcup a) \qquad (a)
\end{array}
$$

Example

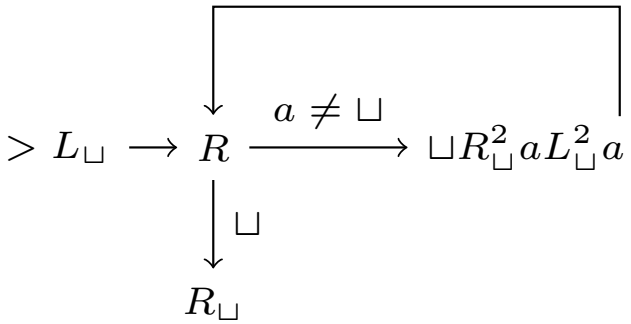
What is the goal of the following Turing Machine?



$$
\begin{array}{lll}
(\sqcup abc\underline{\sqcup}) & \vdash^*_M & (\underline{\sqcup} abc\sqcup) \quad (L_\sqcup) \\
& \vdash_M & (\sqcup \underline{a} bc\sqcup) \quad (R) \\
& \vdash_M & (\sqcup\underline{\sqcup} bc\sqcup) \quad (\sqcup) \\
& \vdash^*_M & (\sqcup \sqcup bc \sqcup \underline{\sqcup}) \quad (R^2_\sqcup) \\
& \vdash_M & (\sqcup \sqcup bc \sqcup \underline{a}) \quad (a) \\
& \vdash^*_M & (\sqcup\underline{\sqcup} bc \sqcup a) \quad (L^2_\sqcup) \\
& \vdash_M & (\sqcup \underline{a} bc \sqcup a) \quad (a) \\
& \vdash_M & (\sqcup a \underline{b} c \sqcup a) \quad (R)
\end{array}
$$

Example

What is the goal of the following Turing Machine?



$$
\begin{array}{lll}
(\sqcup abc\underline{\sqcup}) & \vdash^*_M & (\underline{\sqcup} abc\sqcup) \quad (L_\sqcup) \\
& \vdash_M & (\sqcup \underline{a} bc\sqcup) \quad (R) \\
& \vdash_M & (\sqcup\underline{\sqcup} bc\sqcup) \quad (\sqcup) \\
& \vdash^*_M & (\sqcup \sqcup bc \sqcup \underline{\sqcup}) \quad (R^2_\sqcup) \\
& \vdash_M & (\sqcup \sqcup bc \sqcup \underline{a}) \quad (a) \\
& \vdash^*_M & (\sqcup\underline{\sqcup} bc \sqcup a) \quad (L^2_\sqcup) \\
& \vdash_M & (\sqcup \underline{a} bc \sqcup a) \quad (a) \\
& \vdash_M & (\sqcup a \underline{b} c \sqcup a) \quad (R)
\end{array}
$$

Solution:
transforms $\sqcup w \underline{\sqcup}$ to $\sqcup w \sqcup w \underline{\sqcup}$

Generalize more the notation ...

High-level description

- give an algorithmic description of how the Turing Machine works in finite and discrete steps
- what is allowed?

Generalize more the notation ...

High-level description

- give an algorithmic description of how the Turing Machine works in finite and discrete steps
- what is allowed? $\rightarrow$ almost everything!!

Generalize more the notation ...

High-level description

- give an algorithmic description of how the Turing Machine works in finite and discrete steps
- what is allowed? $\rightarrow$ almost everything!!

Example

$M =$ "On input w:

1. scan the input from left to right to be sure that is member of $a^*b^*c^*$ and *reject* if not
2. find the leftmost a in the tape and if such an a does not exist, then
   - scan the input for a c and if such a c exists then *reject* else *accept*
3. replace a by $\hat{a}$
4. scan the input for the leftmost b and if such a b does not exist, then restore all b's (replace all $\hat{b}$ by b) and goto 2
5. replace b by $\hat{b}$
6. scan to the right for the first c and if such a c does not exist, then *reject*
7. replace c by $\hat{c}$ and goto 4"

Generalize more the notation ...

High-level description

- give an algorithmic description of how the Turing Machine works in finite and discrete steps
- what is allowed? $\rightarrow$ almost everything!!

Example

$L = \{a^i b^j c^k : i \times j = k\}$

$M =$ "On input w:

1. scan the input from left to right to be sure that is member of $a^*b^*c^*$ and *reject* if not
2. find the leftmost a in the tape and if such an a does not exist, then
   - scan the input for a c and if such a c exists then *reject* else *accept*
3. replace a by $\hat{a}$
4. scan the input for the leftmost b and if such a b does not exist, then restore all b's (replace all $\hat{b}$ by b) and goto 2
5. replace b by $\hat{b}$
6. scan to the right for the first c and if such a c does not exist, then *reject*
7. replace c by $\hat{c}$ and goto 4"

Definitions

A language L is called **decidable** (or **Turing-decidable** or **recursive**) if there is a Turing Machine that decides it.

Definitions

A language L is called **decidable** (or **Turing-decidable** or **recursive**) if there is a Turing Machine that decides it.

A language L is called **Turing-recognizable** (or **recursively enumerable**) if there is a Turing Machine that recognizes it.

Basic theorems

Theorem

If a language L is decidable, then it is Turing-recognizable.

Basic theorems

Theorem

If a language L is decidable, then it is Turing-recognizable.

Theorem

If a language L is decidable, then its complement $\bar{L}$ is also.

Proof.

Basic theorems

Theorem

If a language L is decidable, then it is Turing-recognizable.

Theorem

If a language L is decidable, then its complement $\bar{L}$ is also.

Proof.

$$\delta'(q,a) = \begin{cases} n & \text{if } \delta(q,a) = y \\ y & \text{if } \delta(q,a) = n \\ \delta(q,a) & \text{otherwise} \end{cases}$$

□

More definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, \{h\})$ and a string $w \in \Sigma^*$. Suppose that M halts on input w and for some $y \in \Sigma^*$ we have

$$(s, \sqcup w) \vdash_M^* (h, \sqcup y)$$

Then, y is the **output** of M on input w and is denoted by $M(w)$.

More definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, \{h\})$ and a string $w \in \Sigma^*$. Suppose that M halts on input w and for some $y \in \Sigma^*$ we have

$$(s, \sqcup w) \vdash_M^* (h, \sqcup y)$$

Then, y is the **output** of M on input w and is denoted by $M(w)$.

Consider a function $f : \Sigma^* \to \Sigma^*$. We say that M **computes** the function f if $M(w) = f(w)$ for all $w \in \Sigma^*$.

More definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, \{h\})$ and a string $w \in \Sigma^*$. Suppose that M halts on input w and for some $y \in \Sigma^*$ we have

$$(s, \sqcup w) \vdash_M^* (h, \sqcup y)$$

Then, y is the **output** of M on input w and is denoted by $M(w)$.

Consider a function $f : \Sigma^* \to \Sigma^*$. We say that M **computes** the function f if $M(w) = f(w)$ for all $w \in \Sigma^*$.

A function f is called **decidable** (or **recursive**) if there is a Turing Machine that computes it.

More definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, \{h\})$ and a string $w \in \Sigma^*$. Suppose that M halts on input w and for some $y \in \Sigma^*$ we have

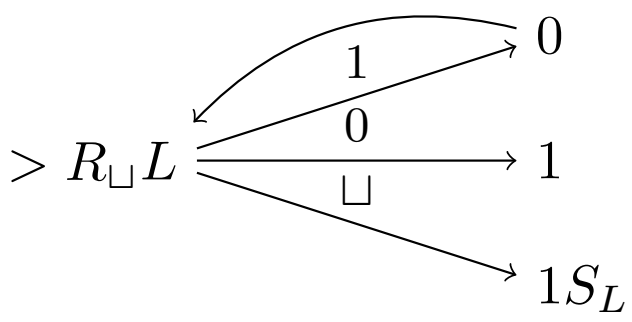
$$(s, \sqcup w) \vdash_M^* (h, \sqcup y)$$

Then, y is the **output** of M on input w and is denoted by $M(w)$.

Consider a function $f : \Sigma^* \to \Sigma^*$. We say that M **computes** the function f if $M(w) = f(w)$ for all $w \in \Sigma^*$.

A function f is called **decidable** (or **recursive**) if there is a Turing Machine that computes it.

Example



The output with input ␣100010111 is ...

More definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, \{h\})$ and a string $w \in \Sigma^*$. Suppose that M halts on input w and for some $y \in \Sigma^*$ we have

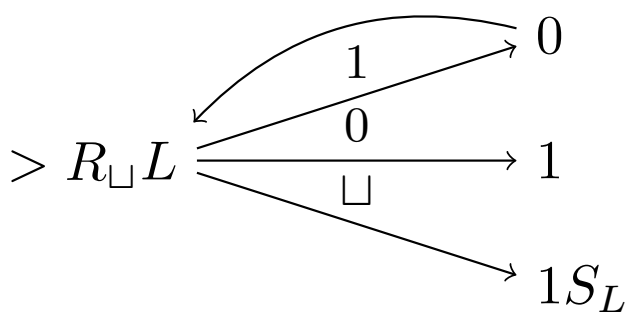
$$(s, \sqcup w) \vdash_M^* (h, \sqcup y)$$

Then, y is the **output** of M on input w and is denoted by $M(w)$.

Consider a function $f : \Sigma^* \to \Sigma^*$. We say that M **computes** the function f if $M(w) = f(w)$ for all $w \in \Sigma^*$.

A function f is called **decidable** (or **recursive**) if there is a Turing Machine that computes it.

Example



The output with input $\sqcup 100010111$ is ... $\sqcup 100011000$

Computes the function $succ(n) = n + 1$ in binary

Extensions of the Turing Machine

We have already seen an extension:

- write in the tape and move left or right at the same time
- modify the definition of the transition function

  initial: from $(K \setminus H) \times \Gamma$ to $K \times (\Gamma \cup \{\leftarrow, \rightarrow\})$

  extended: from $(K \setminus H) \times \Gamma$ to $K \times \Gamma \times \{\leftarrow, \rightarrow\}$

Extensions of the Turing Machine

We have already seen an extension:

- write in the tape and move left or right at the same time
- modify the definition of the transition function

  initial: from $(K \setminus H) \times \Gamma$ to $K \times (\Gamma \cup \{\leftarrow, \rightarrow\})$

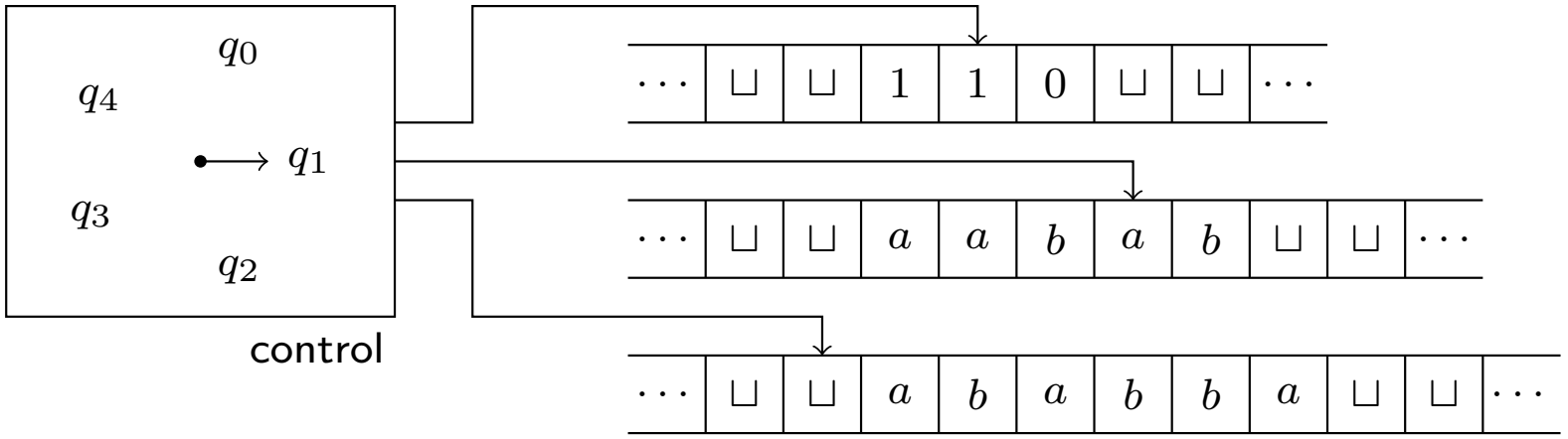
  extended: from $(K \setminus H) \times \Gamma$ to $K \times \Gamma \times \{\leftarrow, \rightarrow\}$

- if the extended Turing Machine halts on input w after t steps, then the initial Turing Machine halts on input w after at most $2t$ steps

Multiple tapes

A k-tape Turing Machine (M) is a sextuple $(K, \Sigma, \Gamma, \delta, s, H)$, where K, Σ, Γ, s and H are as in the definition of the ordinary Turing Machine, and δ is a transition function

$$\text{from} \quad (K \setminus H) \times \Gamma^k \quad \text{to} \quad K \times (\Gamma \cup \{\leftarrow, \rightarrow\})^k$$

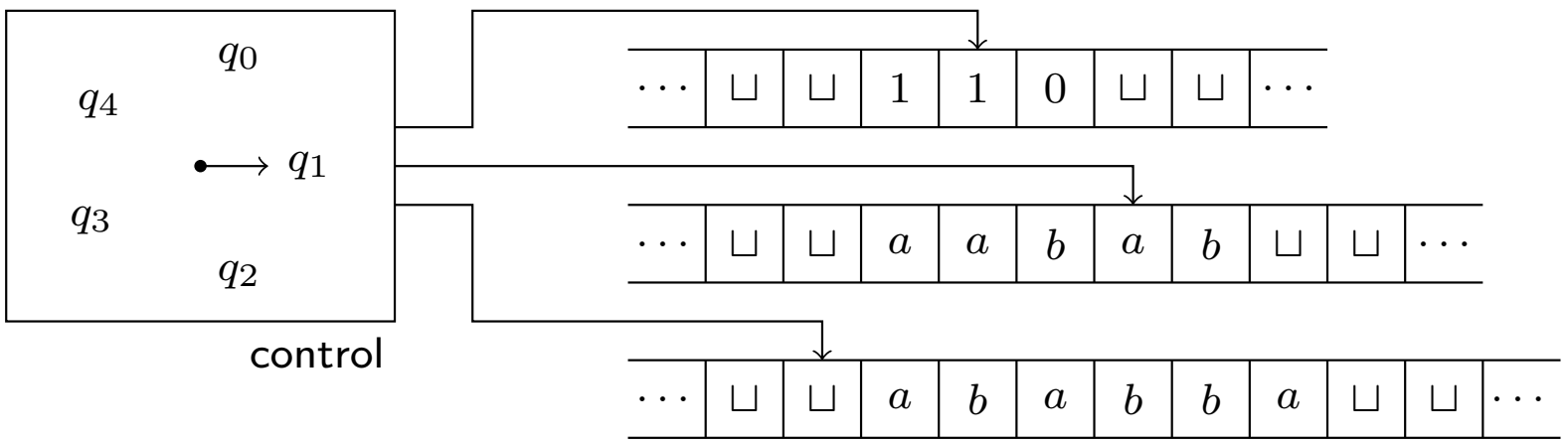


Multiple tapes

A k-tape Turing Machine (M) is a sextuple $(K, \Sigma, \Gamma, \delta, s, H)$, where K, Σ, Γ, s and H are as in the definition of the ordinary Turing Machine, and δ is a transition function

$$\text{from} \quad (K \setminus H) \times \Gamma^k \quad \text{to} \quad K \times (\Gamma \cup \{\leftarrow, \rightarrow\})^k$$

$$(\text{from} \quad (K \setminus H) \times \Gamma^k \quad \text{to} \quad K \times \Gamma^k \times \{\leftarrow, \rightarrow\}^k)$$



Multiple tapes

Theorem

Every k-tape, $k > 1$, Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ has an equivalent single tape Turing Machine $M' = (K', \Sigma', \Gamma', \delta', s', H')$.

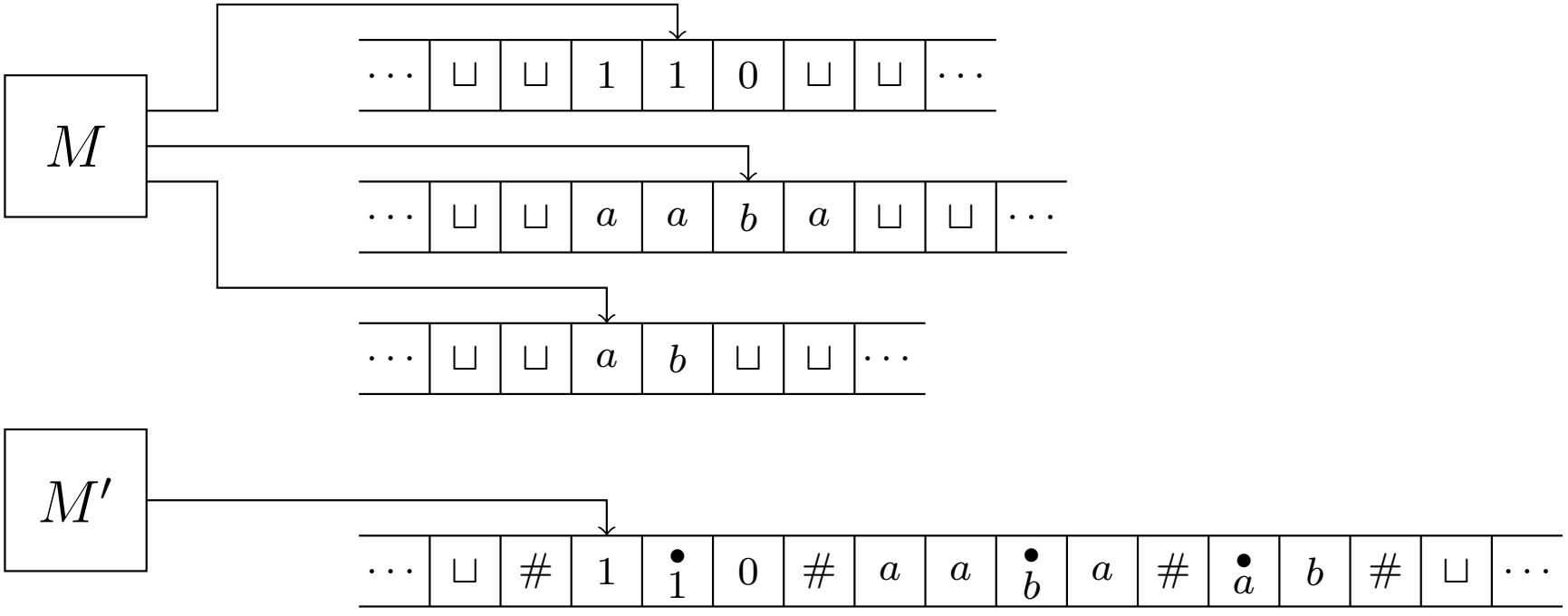
If M halts on input $w \in \Sigma^*$ after t steps, then M' halts on input w after $O(t(|w| + t))$ steps.

Sketch of the proof:

- M' simulates M in a single tape
- $\#$ is used as delimiter to separate the contents of different tapes
- dotted symbols are used to indicate the actual position of the head of each tape
  - for each symbol $\sigma \in \Gamma$, add both σ and $\dot{\sigma}$ in Γ'
- use the same set of halting states

Multiple tapes

Sketch of the proof:



Multiple tapes

Sketch of the proof:

$M' =$ "On input $w = w_1 w_2 \dots w_n$:

1. Format the tape to represent the k tapes:
$$\#\dot{w}_1 w_2 \dots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \dots \#$$
2. For each step that M performs, scan the tape from left to right to determine the symbols under the virtual heads. Then, do a second scan to update the tapes according to the transition function of M.

Multiple tapes

Sketch of the proof:

$M' =$ "On input $w = w_1 w_2 \dots w_n$:

1. Format the tape to represent the k tapes:
$$\#\dot{w}_1 w_2 \dots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \dots \#$$
2. For each step that M performs, scan the tape from left to right to determine the symbols under the virtual heads. Then, do a second scan to update the tapes according to the transition function of M.
3. If at any point there is a need to move a virtual head outside the area marked for the corresponding tape, then shift right the contents of all tapes succeeding."

Multiple tapes

Sketch of the proof:

$M' =$ "On input $w = w_1 w_2 \dots w_n$:

1. Format the tape to represent the k tapes:
$$\#\dot{w}_1 w_2 \dots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \dots \#$$
2. For each step that M performs, scan the tape from left to right to determine the symbols under the virtual heads. Then, do a second scan to update the tapes according to the transition function of M.
3. If at any point there is a need to move a virtual head outside the area marked for the corresponding tape, then shift right the contents of all tapes succeeding."

Number of steps for M':

1. $O(|w|)$

2. & 3. $O(|w| + t)$ per step $\Rightarrow O(t(|w| + t))$ in total
   - size of the tape no more than $O(|w| + t)$

Multiple tapes: conclusion

The multiple tape Turing Machine is **not** more powerful !!

Multiple tapes: conclusion

The multiple tape Turing Machine is **not** more powerful !!

... but it is more easy to construct and to understand !

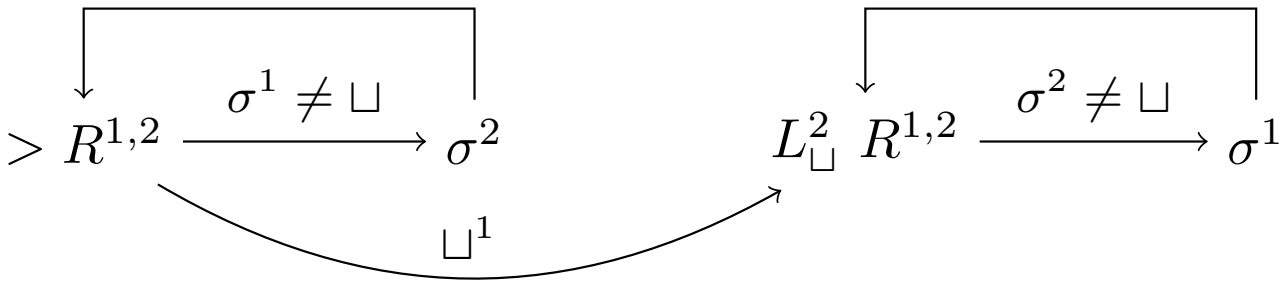
Multiple tapes: conclusion

The multiple tape Turing Machine is **not** more powerful !!

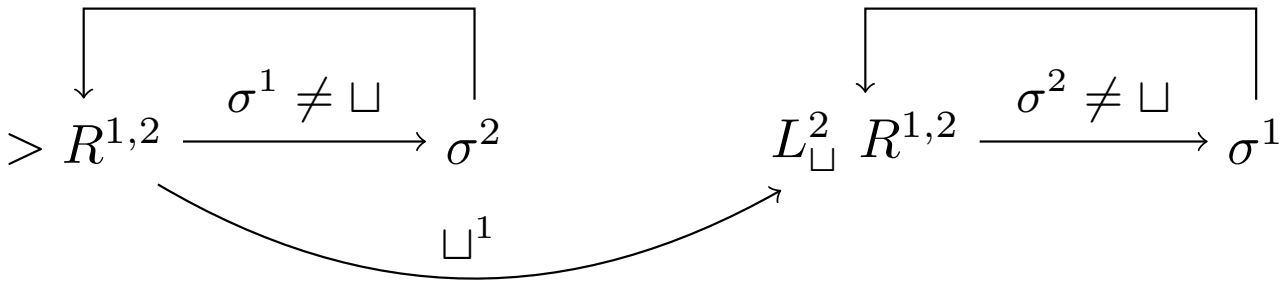
... but it is more easy to construct and to understand !

... and it can be used to simulate functions in an easier way
(a function can use one or more not used tapes)

Multiple tapes: example with $k = 2$ tapes

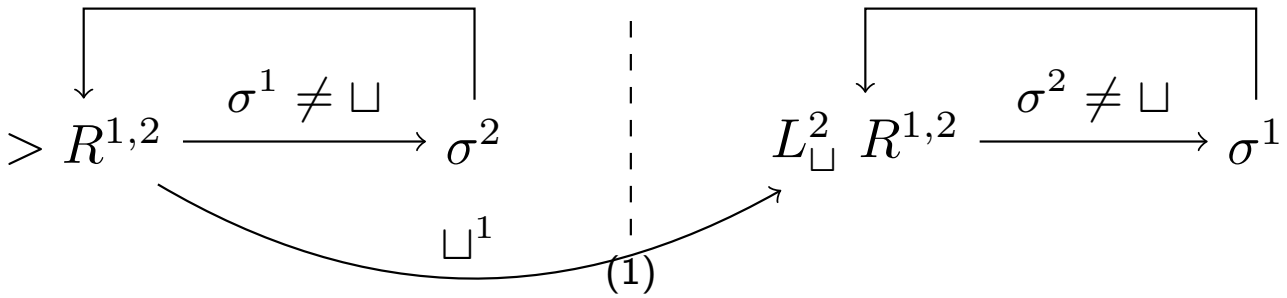


Multiple tapes: example with $k = 2$ tapes



- extend notation:
  - $R^{1,2}$: move the head of both tapes on the right
  - σ^2 (as a state): write in the tape 2 the symbol σ
  - σ^2 (as a label): if the head of tape 2 reads the symbol σ

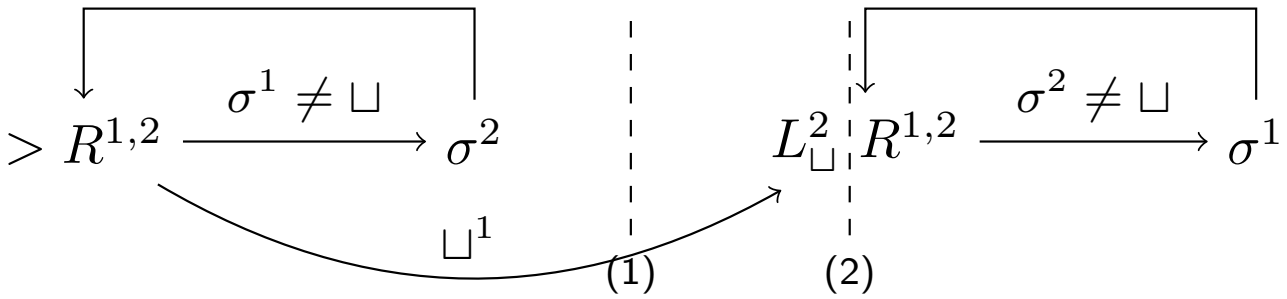
Multiple tapes: example with $k = 2$ tapes



- extend notation:
  - $R^{1,2}$: move the head of both tapes on the right
  - σ^2 (as a state): write in the tape 2 the symbol σ
  - σ^2 (as a label): if the head of tape 2 reads the symbol σ

	tape 1	tape 2
initially	$\underline{\sqcup} w$	$\underline{\sqcup}$
after (1)		

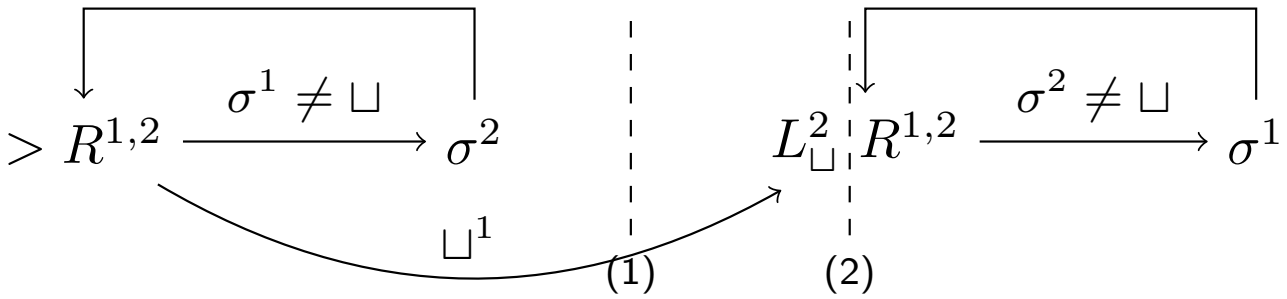
Multiple tapes: example with $k = 2$ tapes



- extend notation:
  - $R^{1,2}$: move the head of both tapes on the right
  - σ^2 (as a state): write in the tape 2 the symbol σ
  - σ^2 (as a label): if the head of tape 2 reads the symbol σ

	tape 1	tape 2
initially	$\underline{\sqcup}w$	$\underline{\sqcup}$
after (1)	$\sqcup w\underline{\sqcup}$	$\sqcup w\underline{\sqcup}$
after (2)		

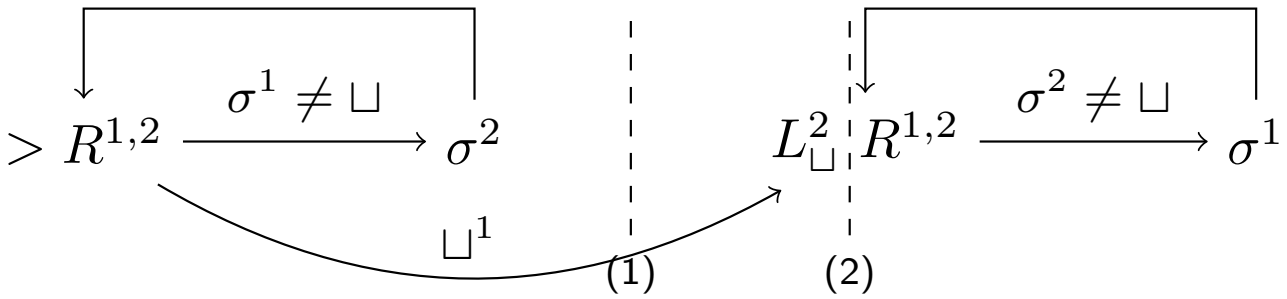
Multiple tapes: example with $k = 2$ tapes



- extend notation:
  - $R^{1,2}$: move the head of both tapes on the right
  - σ^2 (as a state): write in the tape 2 the symbol σ
  - σ^2 (as a label): if the head of tape 2 reads the symbol σ

	tape 1	tape 2
initially	$\underline{\sqcup}w$	$\underline{\sqcup}$
after (1)	$\sqcup w\underline{\sqcup}$	$\sqcup w\underline{\sqcup}$
after (2)	$\sqcup w\underline{\sqcup}$	$\underline{\sqcup}w\sqcup$
at the end		

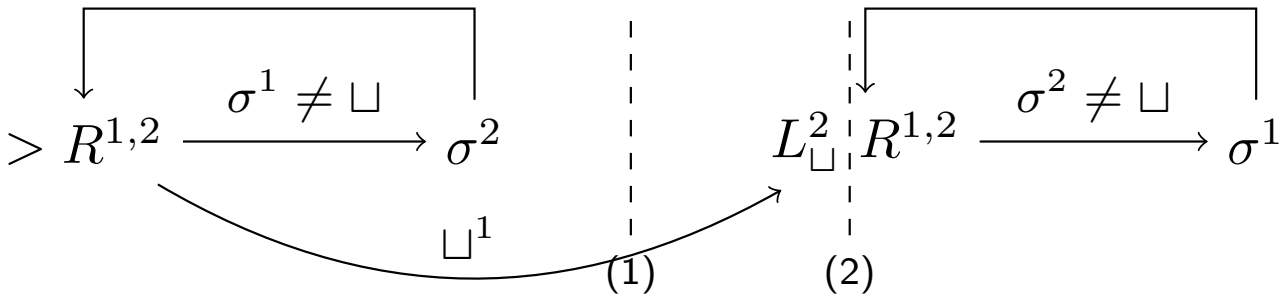
Multiple tapes: example with $k = 2$ tapes



- extend notation:
  - $R^{1,2}$: move the head of both tapes on the right
  - σ^2 (as a state): write in the tape 2 the symbol σ
  - σ^2 (as a label): if the head of tape 2 reads the symbol σ

	tape 1	tape 2
initially	$\underline{\sqcup}w$	$\underline{\sqcup}$
after (1)	$\sqcup w\underline{\sqcup}$	$\sqcup w\underline{\sqcup}$
after (2)	$\sqcup w\underline{\sqcup}$	$\underline{\sqcup}w\sqcup$
at the end	$\sqcup w \sqcup w\underline{\sqcup}$	$\sqcup w\underline{\sqcup}$

Multiple tapes: example with $k = 2$ tapes



- extend notation:
  - $R^{1,2}$: move the head of both tapes on the right
  - σ^2 (as a state): write in the tape 2 the symbol σ
  - σ^2 (as a label): if the head of tape 2 reads the symbol σ

	tape 1	tape 2
initially	$\underline{\sqcup}w$	$\underline{\sqcup}$
after (1)	$\sqcup w\underline{\sqcup}$	$\sqcup w\underline{\sqcup}$
after (2)	$\sqcup w\underline{\sqcup}$	$\underline{\sqcup}w\sqcup$
at the end	$\sqcup w \sqcup w\underline{\sqcup}$	$\sqcup w\underline{\sqcup}$

transforms w to $w \sqcup w$

Multiple heads

Definition (informal)

- at each step all heads can read/write/move
- we need a convention if two heads try writing in the same place

Multiple heads

Definition (informal)

- at each step all heads can read/write/move
- we need a convention if two heads try writing in the same place

Theorem

Every multiple head Turing Machine M has an equivalent single head Turing Machine M'.

The simulation by M' of M on an input w which leads to a halting state takes time quadratic to the size of the input $|w|$ and the number of steps t that M performs.

Proof (sketch):

Multiple heads

Definition (informal)

- at each step all heads can read/write/move
- we need a convention if two heads try writing in the same place

Theorem

Every multiple head Turing Machine M has an equivalent single head Turing Machine M'.

The simulation by M' of M on an input w which leads to a halting state takes time quadratic to the size of the input $|w|$ and the number of steps t that M performs.

Proof (sketch):

- scan the tape twice
  1. find the symbols at the head positions (which transition to follow?)
  2. write/move the heads according to the transition
- same arguments as before for the number of steps

Multiple heads

Definition (informal)

- at each step all heads can read/write/move
- we need a convention if two heads try writing in the same place

Theorem

Every multiple head Turing Machine M has an equivalent single head Turing Machine M'.

The simulation by M' of M on an input w which leads to a halting state takes time quadratic to the size of the input $|w|$ and the number of steps t that M performs.

Proof (sketch):

- scan the tape twice
  1. find the symbols at the head positions (which transition to follow?)
  2. write/move the heads according to the transition
- same arguments as before for the number of steps

Multiple heads

Definition (informal)

- at each step all heads can read/write/move
- we need a convention if two heads try writing in the same place

Theorem

Every multiple head Turing Machine M has an equivalent single head Turing Machine M'.

The simulation by M' of M on an input w which leads to a halting state takes time quadratic to the size of the input $|w|$ and the number of steps t that M performs.

Proof (another one):

Multiple heads

Definition (informal)

- at each step all heads can read/write/move
- we need a convention if two heads try writing in the same place

Theorem

Every multiple head Turing Machine M has an equivalent single head Turing Machine M'.

The simulation by M' of M on an input w which leads to a halting state takes time quadratic to the size of the input $|w|$ and the number of steps t that M performs.

Proof (another one):

$\cdots$	$\sqcup$	m	y	$\sqcup$	i	n	p	u	t	$\sqcup$	$\cdots$
			$\wedge$								
							$\wedge$				
				$\wedge$							

Multiple heads: example

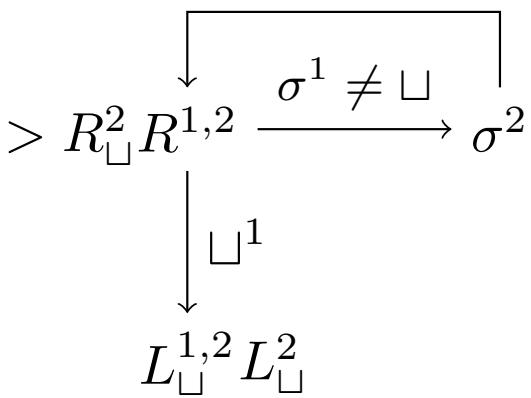
Give a Machine Turing with two heads that transforms the input $\underline{\overline{\sqcup}}w$ to $\underline{\overline{\sqcup}}w \sqcup w$.

- extend notation:
  - $\underline{\sigma}$, $\overline{\sigma}$, $\underline{\overline{\sigma}}$: the position of the 1st, 2nd and both heads, respectively
  - $R^{1,2}$: move both heads on the right
  - σ^2 (as a state): write in the position of head 2 the symbol σ
  - σ^2 (as a label): if the head 2 reads the symbol σ

Multiple heads: example

Give a Machine Turing with two heads that transforms the input $\underline{\sqcup} w$ to $\underline{\sqcup} w \sqcup w$.

- extend notation:
  - $\underline{\sigma}$, $\overline{\sigma}$, $\underline{\overline{\sigma}}$: the position of the 1st, 2nd and both heads, respectively
  - $R^{1,2}$: move both heads on the right
  - σ^2 (as a state): write in the position of head 2 the symbol σ
  - σ^2 (as a label): if the head 2 reads the symbol σ



Unbounded tapes

What happens if the tape is bounded in one direction?

Unbounded tapes

What happens if the tape is bounded in one direction?

Theorem

Every two-direction unbounded tape Turing Machine M has an equivalent single-direction unbounded tape Turing Machine.

Two-dimensional tape

Definition (informal)

- move the head left/right/up/down

Two-dimensional tape

Definition (informal)

- move the head left/right/up/down

Why?

Two-dimensional tape

Definition (informal)

- move the head left/right/up/down

Why?

- for example, to represent more easily two-dimensional matrices

Two-dimensional tape

Definition (informal)

- move the head left/right/up/down

Why?

- for example, to represent more easily two-dimensional matrices

Theorem

Every two-dimensional tape Turing Machine M has an equivalent single-dimensional tape Turing Machine M'.

The simulation by M' of M on an input w which leads to a halting state takes time polynomial to the size of the input $|w|$ and the number of steps t that M performs.

Proof (sketch):

Two-dimensional tape

Definition (informal)

- move the head left/right/up/down

Why?

- for example, to represent more easily two-dimensional matrices

Theorem

Every two-dimensional tape Turing Machine M has an equivalent single-dimensional tape Turing Machine M'.

The simulation by M' of M on an input w which leads to a halting state takes time polynomial to the size of the input $|w|$ and the number of steps t that M performs.

Proof (sketch):

- use a multiple tape Turing Machine
- each tape corresponds to one line of the two-dimensional memory

Discussion

- We can even combine the presented extensions and still not get a stronger model

Discussion

- We can even combine the presented extensions and still not get a stronger model
- Observation: a computation in the prototype Turing Machine needs a number of steps which is bounded by a polynomial of the size of the input and of the number steps in any of the extended model