Fundamental Computer Science

Giorgio Lucarelli

giorgio.lucarelli@imag.fr

January 31, 2018

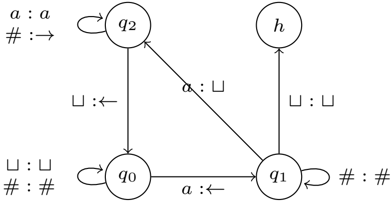
Slides & exercises

`http://moais.imag.fr/membres/giorgio.lucarelli/FCS`

Exercise

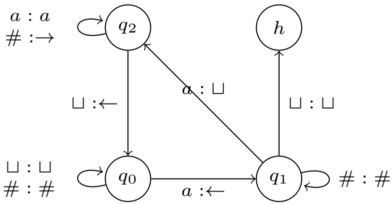
$$\Sigma = \{a\}, \quad \Gamma = \{a, \#, \sqcup\}, \quad s = q_0, \quad H = \{h\}$$

$$(q_0, \# \sqcup a^n \underline{a})$$



Exercise

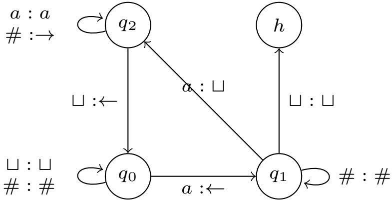
$$\Sigma = \{a\}, \quad \Gamma = \{a, \#, \sqcup\}, \quad s = q_0, \quad H = \{h\}$$



$$\begin{aligned}
(q_0, \# \sqcup a^n \underline{a}) &\vdash_M (q_1, \# \sqcup a^{n-1}\underline{a}a) \\
&\vdash_M (q_2, \# \sqcup a^{n-1}\underline{\sqcup}a) \\
&\vdash_M (q_0, \# \sqcup a^{n-2}\underline{a} \sqcup a) \\
&\vdash_M (q_1, \# \sqcup a^{n-3}\underline{a}a \sqcup a) \\
&\vdash_M (q_2, \# \sqcup a^{n-3}\underline{\sqcup}a \sqcup a) \\
&\vdash_M (q_0, \# \sqcup a^{n-4}\underline{a} \sqcup a \sqcup a) \\
&\vdash_M (q_1, \# \sqcup a^{n-5}\underline{a}a \sqcup a \sqcup a) \\
&\vdash_M (q_2, \# \sqcup a^{n-5}\underline{\sqcup}a \sqcup a \sqcup a) \\
&\ldots
\end{aligned}$$

Exercise

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

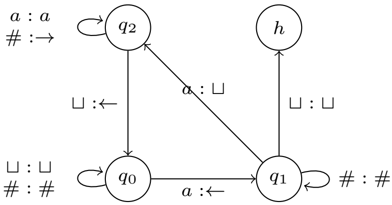


$$\begin{aligned}
(q_0, \# \sqcup a^n \underline{a}) &\vdash_M (q_1, \# \sqcup a^{n-1} \underline{a} a) \\
&\vdash_M (q_2, \# \sqcup a^{n-1} \underline{\sqcup} a) \\
&\vdash_M (q_0, \# \sqcup a^{n-2} \underline{a} \sqcup a) \\
&\vdash_M (q_1, \# \sqcup a^{n-3} \underline{a} a \sqcup a) \\
&\vdash_M (q_2, \# \sqcup a^{n-3} \underline{\sqcup} a \sqcup a) \\
&\vdash_M (q_0, \# \sqcup a^{n-4} \underline{a} \sqcup a \sqcup a) \\
&\vdash_M (q_1, \# \sqcup a^{n-5} \underline{a} a \sqcup a \sqcup a) \\
&\vdash_M (q_2, \# \sqcup a^{n-5} \underline{\sqcup} a \sqcup a \sqcup a) \\
&\ldots
\end{aligned}$$

$$\begin{aligned}
(q_0, \triangleright \sqcup aa\underline{a}) &\vdash_M (q_1, \triangleright \sqcup a\underline{a}a) \\
&\vdash_M (q_2, \triangleright \sqcup a\underline{\sqcup}a) \\
&\vdash_M (q_0, \triangleright \sqcup \underline{a} \sqcup a) \\
&\vdash_M (q_1, \triangleright \underline{\sqcup} a \sqcup a) \\
&\vdash_M (h, \triangleright \underline{\sqcup} a \sqcup a)
\end{aligned}$$

Exercise

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



$$\begin{aligned}
(q_0, \# \sqcup a^n \underline{a}) &\vdash_M (q_1, \# \sqcup a^{n-1} \underline{a} a) \\
&\vdash_M (q_2, \# \sqcup a^{n-1} \underline{\sqcup} a) \\
&\vdash_M (q_0, \# \sqcup a^{n-2} \underline{a} \sqcup a) \\
&\vdash_M (q_1, \# \sqcup a^{n-3} \underline{a} a \sqcup a) \\
&\vdash_M (q_2, \# \sqcup a^{n-3} \underline{\sqcup} a \sqcup a) \\
&\vdash_M (q_0, \# \sqcup a^{n-4} \underline{a} \sqcup a \sqcup a) \\
&\vdash_M (q_1, \# \sqcup a^{n-5} \underline{a} a \sqcup a \sqcup a) \\
&\vdash_M (q_2, \# \sqcup a^{n-5} \underline{\sqcup} a \sqcup a \sqcup a) \\
&\ldots
\end{aligned}$$

$$\begin{aligned}
(q_0, \triangleright \sqcup aa\underline{a}) &\vdash_M (q_1, \triangleright \sqcup a\underline{a}a) \\
&\vdash_M (q_2, \triangleright \sqcup a\underline{\sqcup}a) \\
&\vdash_M (q_0, \triangleright \sqcup \underline{a} \sqcup a) \\
&\vdash_M (q_1, \triangleright\underline{\sqcup} a \sqcup a) \\
&\vdash_M (h, \triangleright\underline{\sqcup} a \sqcup a)
\end{aligned}$$

$$\begin{aligned}
(q_0, \triangleright \sqcup a\underline{a}) &\vdash_M (q_1, \triangleright \sqcup \underline{a}a) \\
&\vdash_M (q_2, \triangleright \sqcup \underline{\sqcup}a) \\
&\vdash_M (q_0, \triangleright\underline{\sqcup} \sqcup a) \\
&\vdash_M (q_0, \triangleright\underline{\sqcup} \sqcup a) \\
&\ldots
\end{aligned}$$

Definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ such that $H = \{y, n\}$.

Any halting configuration whose state component is y (for "*yes*") is called an **accepting configuration**, while a halting configuration whose state component is n (for "*no*") is called a **rejecting configuration**.

Definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ such that $H = \{y, n\}$.

Any halting configuration whose state component is y (for "*yes*") is called an **accepting configuration**, while a halting configuration whose state component is n (for "*no*") is called a **rejecting configuration**.

We say that M **accepts** a string $w \in \Sigma^*$ if starting from an initial configuration yields an accepting configuration.
We say that M **rejects** a string $w \in \Sigma^*$ if starting from an initial configuration yields an rejecting configuration.

Definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ such that $H = \{y, n\}$.

Any halting configuration whose state component is y (for "*yes*") is called an **accepting configuration**, while a halting configuration whose state component is n (for "*no*") is called a **rejecting configuration**.

We say that M **accepts** a string $w \in \Sigma^*$ if starting from an initial configuration yields an accepting configuration.
We say that M **rejects** a string $w \in \Sigma^*$ if starting from an initial configuration yields an rejecting configuration.

We say that M **decides** a language $L \subseteq \Sigma^*$ if for any string $w \in \Sigma^*$: if $w \in L$ then M accepts w; and if $w \notin L$ then M rejects w.

Definitions

Consider a Turing Machine $M = (K, \Sigma, \Gamma, \delta, s, H)$ such that $H = \{y, n\}$.

Any halting configuration whose state component is y (for "*yes*") is called an **accepting configuration**, while a halting configuration whose state component is n (for "*no*") is called a **rejecting configuration**.

We say that M **accepts** a string $w \in \Sigma^*$ if starting from an initial configuration yields an accepting configuration.
We say that M **rejects** a string $w \in \Sigma^*$ if starting from an initial configuration yields an rejecting configuration.

We say that M **decides** a language $L \subseteq \Sigma^*$ if for any string $w \in \Sigma^*$: if $w \in L$ then M accepts w; and if $w \notin L$ then M rejects w.

We say that M **recognizes** (or **semidecides**) a language $L \subseteq \Sigma^*$ if for any string $w \in \Sigma^*$: $w \in L$ if and only if M accepts w.

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.

Proof.

Basic theorems

Theorem

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

Proof.

Transform the Turing Machine M that decides L such that M does not halt on input w if $w \notin L$. $\square$

Basic theorems

Theorem

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

Proof.

Transform the Turing Machine M that decides L such that M does not halt on input w if $w \notin L$. $\square$

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.

Proof.

Transform the Turing Machine M that decides L such that M does not halt on input w if $w \notin L$. $\square$

Theorem

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

Proof.

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

$\square$

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

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

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

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

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

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

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

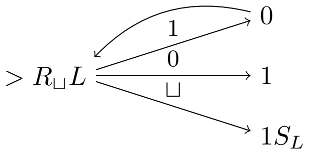
$$(s, \underline{\sqcup} w) \vdash_M^* (h, \underline{\sqcup} y)$$

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 ...

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

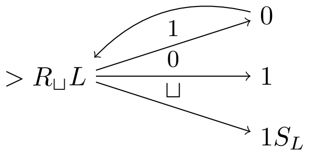
$$(s, \underline{\sqcup} w) \vdash_M^* (h, \underline{\sqcup} y)$$

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

Exercise

Prove that the language $L = \{a^n b^n c^n : n \geq 0\}$ is decidable.

Exercise

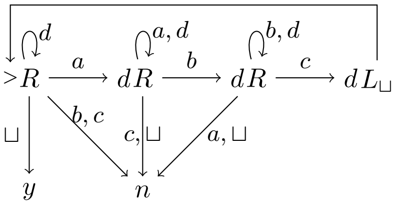
Prove that the language $L = \{a^n b^n c^n : n \geq 0\}$ is decidable.

Solution: We just need to give a Turing Machine that decides it. (give a Turing Machine composed by simple Turing Machines as described in the previous lecture)

Exercise

Prove that the language $L = \{a^n b^n c^n : n \geq 0\}$ is decidable.

Solution: We just need to give a Turing Machine that decides it. (give a Turing Machine composed by simple Turing Machines as described in the previous lecture)



More exercises

- Present Turing Machines that decide the following languages over $\{a, b\}$:
  (a) $\emptyset$
  (b) $\{\epsilon\}$
  (c) $\{a\}$
  (d) $\{a\}^*$

- Give a Turing Machine that *recognizes* the language a^*ba^*b.

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 $K \times (\Gamma \cup \{\leftarrow, \rightarrow\})$

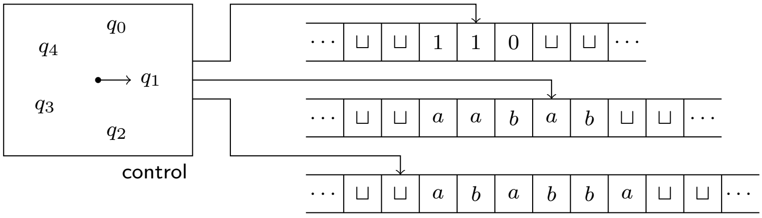
  extended: from $(K \setminus H) \times \Gamma$ to $K \times \Gamma \times \{\leftarrow, \rightarrow\}$

- 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

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

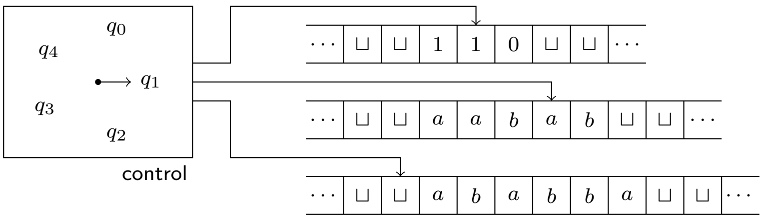


Multiple tapes

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 $M' = (K', \Sigma', \Gamma', \delta', s', H')$.

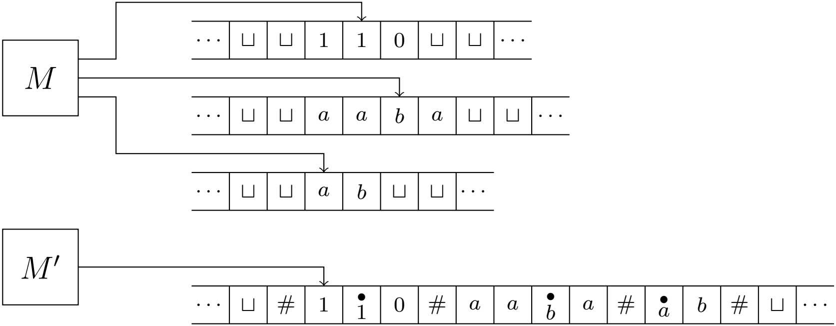
If M halts on input $w \in \Sigma^*$ after t steps, then M' halts on input w after $O(t(|w| + t))$ steps.

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 $w = w_1 w_2 \ldots w_n$:

1. Format the tape to represent the k tapes:
$$\# \dot{w}_1 w_2 \ldots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#$$
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 $w = w_1 w_2 \ldots w_n$:

1. Format the tape to represent the k tapes:
$$\# \dot{w}_1 w_2 \ldots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#$$
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 <u>**not**</u> 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 **<u>not</u>** 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

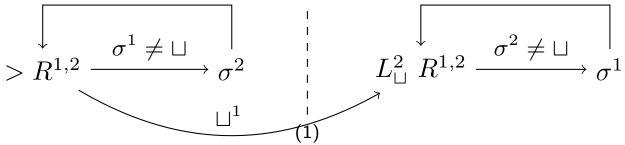
$$\begin{tikzcd}
> R^{1,2} \arrow[r, "\sigma^1 \neq \sqcup"] \arrow[rr, bend right, "\sqcup^1"'] & \sigma^2 \arrow[l, bend right] & L^2_{\sqcup}\ R^{1,2} \arrow[r, "\sigma^2 \neq \sqcup"] & \sigma^1 \arrow[l, bend right]
\end{tikzcd}$$

Multiple tapes: example with $k = 2$ tapes

$$
\begin{tikzcd}
{>}\, R^{1,2} \arrow[r, "\sigma^1 \neq \sqcup"] \arrow[rr, bend right, "\sqcup^1"'] & \sigma^2 \arrow[l, bend right] & L^2_{\sqcup} & R^{1,2} \arrow[r, "\sigma^2 \neq \sqcup"] & \sigma^1 \arrow[l, bend right]
\end{tikzcd}
$$

- 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

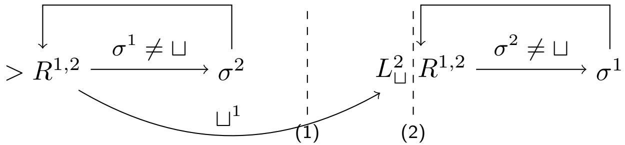
Multiple tapes: example with $k = 2$ tapes



- 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)		

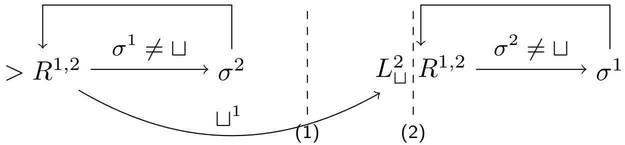
Multiple tapes: example with $k = 2$ tapes



- 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)		

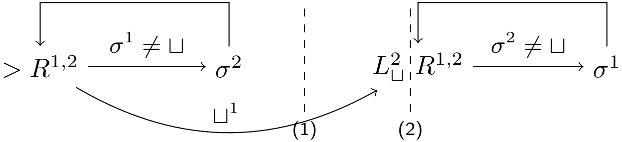
Multiple tapes: example with $k = 2$ tapes



- 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		

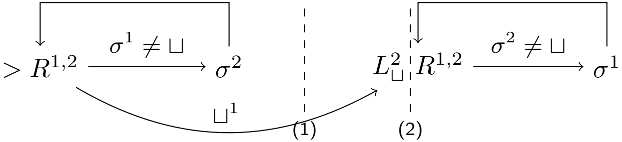
Multiple tapes: example with $k = 2$ tapes



- 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 tapes: exercise

- Construct a Turing Machine that **adds** two binary numbers.
  Tip: use 2 tapes.

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 $\underline{\sqcup} w$ 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

Multiple heads: example

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

- extend notation:
  - $\underline{\sigma}$, $\overline{\sigma}$, $\overline{\underline{\sigma}}$: 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

$$
\begin{tikzcd}
> R^2_{\sqcup} R^{1,2} \arrow[r, "\sigma^1 \neq \sqcup"] \arrow[d, "\sqcup^1"] & \sigma^2 \arrow[l, bend right] \\
L^{1,2}_{\sqcup} L^2_{\sqcup}
\end{tikzcd}
$$

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

Exercises

- Give an example of a Turing machine with one halting state that does not *compute* a function from strings to strings.
- Give an example of a Turing machine with two halting states, y and n, that does not *decide* a language.
- Can you give an example of a Turing machine with one halting state that does not *recognize* a language?
- Give a Turing Machine which takes as input an integer written in unary and *computes* the binary representation of the same integer (for example $< 11111111111 >_1$ becomes $< 1011 >_2$).

A small exam...

- Construct a Turing Machine that **multiplies** two binary numbers. Tip: use 3 tapes and the machine that performs the addition of two binary numbers as a subroutine.

Instructions

- groups of at most 4 (and at least 3)
- 1 answer per group
- 1 author clearly defined per group
- do not forget to give the names of all members of the group
- you have 30 minutes