Problème de décisions autour des pavages

(École CIMPA Bobo-Dioulasso)

M. Sablik

LATP, Université Aix-Marseille

Novembre 2012

Problème de décision autour des pavages

Problème du domino:
Soit $\mathcal{A}$ un alphabet fini et $\mathcal{F}$ ensemble fini de motifs 2-dimensionnel.

$$\textit{Est-ce que } \mathbf{T}(\mathcal{A}, 2, \mathcal{F}) = \left\{ x \in \mathcal{A}^{\mathbb{Z}^2} : \forall p \in \mathcal{F},\ p \not\sqsubset x \right\} \neq \varnothing ?$$

Problèmes de décisions

Notion de (in)-décidabilité

- *Problème de décision*: problème qui prend en entrée une *instance* codée par une suite fini de bits, c'est à dire un élément de $\{0,1\}^*$, et répond en sortie "oui" ou "non" suivant si l'instance répond au problème ou pas.
- *Programme*: suite finie et non-ambiguë d'opérations ou d'instructions. Il peut être codé informatiquement par une suite fini de bits, c'est à dire aussi un élément de $\{0,1\}^*$.
- Un problème de décision est *décidable* si il existe un programme permettant de le résoudre sur n'importe quelle entrée et *indécidable* sinon.

Notion de (in)-décidabilité

- *Problème de décision*: problème qui prend en entrée une *instance* codée par une suite fini de bits, c'est à dire un élément de $\{0,1\}^*$, et répond en sortie "oui" ou "non" suivant si l'instance répond au problème ou pas.
- *Programme*: suite finie et non-ambiguë d'opérations ou d'instructions. Il peut être codé informatiquement par une suite fini de bits, c'est à dire aussi un élément de $\{0,1\}^*$.
- Un problème de décision est *décidable* si il existe un programme permettant de le résoudre sur n'importe quelle entrée et *indécidable* sinon.

Example:

"Si $a, b, c \in \mathbb{N}$, est-ce que $ax^2 + bx + c$ admet des racines réelles?"

→ On calcule $\Delta = b^2 - 4ac$,

→ Si $\Delta > 0$ on répond "oui", sinon on répond "non".

Notion de (in)-décidabilité

- *Problème de décision*: problème qui prend en entrée une *instance* codée par une suite fini de bits, c'est à dire un élément de $\{0,1\}^*$, et répond en sortie "oui" ou "non" suivant si l'instance répond au problème ou pas.
- *Programme*: suite finie et non-ambiguë d'opérations ou d'instructions. Il peut être codé informatiquement par une suite fini de bits, c'est à dire aussi un élément de $\{0,1\}^*$.
- Un problème de décision est *décidable* si il existe un programme permettant de le résoudre sur n'importe quelle entrée et *indécidable* sinon.
- Un problème de décision est *semi-décidable* si il existe un programme qui répond "oui" pour une instance positive du problème et tourne indéfinitivement sinon.

Proposition

Un problème de décision est décidable si et seulement si le problème et le problème complémentaire sont semi-décidable.

Problème de l'arrêt

Problème de l'arrêt

Est-ce qu'un programme exécuté sur une entrée donnée s'arrête?

Théorème *Turing 1937*

Le problème de l'arrêt est indécidable (bien qu'il est semi-décidable).

Problème de l'arrêt

Problème de l'arrêt

Est-ce qu'un programme exécuté sur une entrée donnée s'arrête?

Théorème *Turing 1937*

Le problème de l'arrêt est indécidable (bien qu'il est semi-décidable).

Idée de preuve:

Dans un petit village, il y a un barbier. Il rase les gens qui ne se rasent pas eux-mêmes, et seulement ces gens là.

Par qui est rasé le barbier?

Problème de l'arrêt

Problème de l'arrêt

Est-ce qu'un programme exécuté sur une entrée donnée s'arrête?

Théorème *Turing 1937*

Le problème de l'arrêt est indécidable (bien qu'il est semi-décidable).

Idée de preuve:

Supposons qu'il existe `SuperProg` vérifiant pour un programme `P` et une entrée w :

$$\texttt{SuperProg}(\texttt{P}, w) = \begin{cases} \text{"oui"} & \text{si P(w) s'arrête} \\ \text{"non"} & \text{sinon} \end{cases}$$

On construit le programme `Bug` tel que:

$$\texttt{Bug}(\texttt{P}) = \begin{cases} \text{"oui"} & \text{si SuperProg(P,P)="non"} \\ \text{boucle} & \text{sinon} \end{cases}$$

Problème de l'arrêt

Problème de l'arrêt

Est-ce qu'un programme exécuté sur une entrée donnée s'arrête?

Théorème *Turing 1937*

Le problème de l'arrêt est indécidable (bien qu'il est semi-décidable).

Idée de preuve:

Supposons qu'il existe `SuperProg` vérifiant pour un programme `P` et une entrée w :

$$\texttt{SuperProg}(\texttt{P}, w) = \begin{cases} \text{"oui"} & \text{si P(w) s'arrête} \\ \text{"non"} & \text{sinon} \end{cases}$$

On construit le programme Bug tel que:

$$\texttt{Bug}(\texttt{P}) = \begin{cases} \text{"oui"} & \text{si SuperProg(P,P)="non"} \\ \text{boucle} & \text{sinon} \end{cases}$$

Que fait le programme Bug sur l'entrée Bug?

Problème de l'arrêt (bis)

Problème de l'arrêt sur l'entrée vide

Est-ce qu'un programme exécuté sur une entrée donnée s'arrête sur l'entrée vide?

Corollaire *Turing 1937*

Le problème de l'arrêt sur l'entrée vide est indécidable (bien qu'il soit semi-décidable).

Preuve par réduction au problème de l'arrêt:

- supposons qu'un programme $\mathtt{P}_\varnothing$ décide si un programme s'arrête sur l'entrée vide;
- étant donné un programme $\mathtt{P}$ et une entrée w, on note $\mathtt{P}_w$ le programme qui calcule $\mathtt{P}(w)$ partant de l'entrée vide;
- on considère le programme `SuperProg` qui étant donnée l'entrée $(\mathtt{P}, w)$ place en mémoire w puis simule $\mathtt{P}$ sur l'entrée w;

Alors `SuperProg` décide le problème de l'arrêt.

Premiers problèmes de décisions pour les pavages

Cas de la dimension 1

Proposition

Le problème du domino est décidable en dimension 1.

$\mathbf{T}(\{0,1\},1,\{11\}) \neq \varnothing$?:

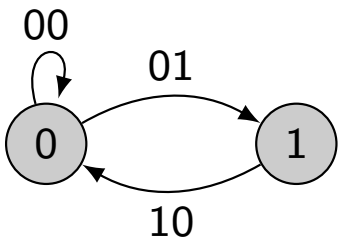
$\mathbf{T}(\{0,1\},1,\{111,000,101\}) \neq \varnothing$?:

Cas de la dimension 1

Proposition

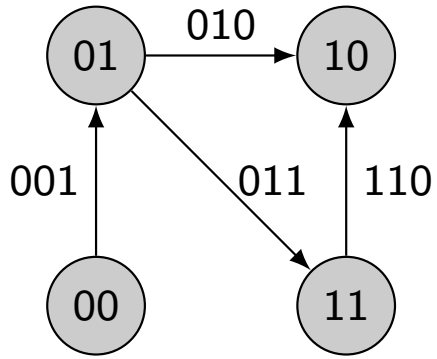
Le problème du domino est décidable en dimension 1.

$\mathbf{T}(\{0,1\},1,\{11\}) \neq \varnothing$?:



Oui

$\mathbf{T}(\{0,1\},1,\{111,000,101\}) \neq \varnothing$?:



Non

Le complémentaire du problème du domino

Proposition

Le complémentaire du problème du domino est semi-décidable.

Input: On donne les motifs interdits $\mathcal{F}$.
Programme:

- $n := 1$ et bool="true"
- Tant que bool faire:
  - On énumère tous les motifs de taille $[0, n]^d$.
  - S'il existe un motif qui ne contient aucun motif de $\mathcal{F}$:
    - ★ Alors $n := n + 1$
    - ★ Sinon bool="false"
- Le programme renvoie que le pavage ne pave pas le plan.

Que fait ce programme:

- Si $\mathbf{T}(\mathcal{A}, d, \mathcal{F}) = \varnothing$ il existe un n tel que tous les motifs de support $[0, n]^d$ contiennent un motif interdit, donc le programme s'arrête.
- Si $\mathbf{T}(\mathcal{A}, d, \mathcal{F}) \neq \varnothing$ le programme boucle indéfiniment.

Problème du pavage périodique

Problème du pavage périodique

Est-ce qu'un SFT admet une orbite périodique?

Proposition

Le problème du pavage périodique est semi-décidable.

Programme: On énumère tous les motifs de support $[0, n]^d$ jusqu'à en trouver un tel que l'orbite périodique engendrée ne contienne pas de motifs interdits.
Que fait ce programme: Ce programme s'arrête si et seulement si il trouve une orbite périodique.

Remarque de Wang

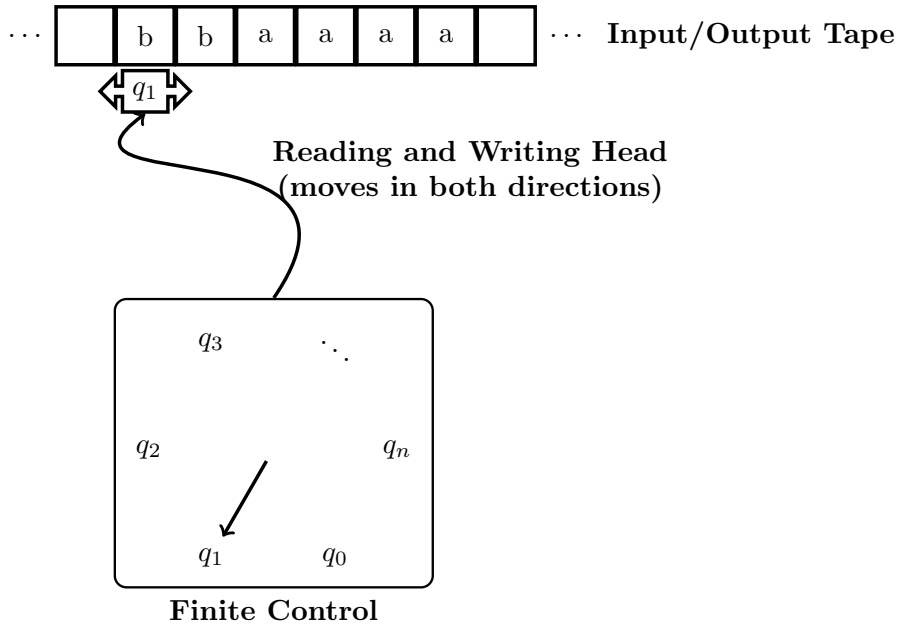
Si tout les sous-shifts de type fini admettent une configuration périodique alors le problème du domino est décidable.

Machines de Turing et Pavages

Machine de Turing

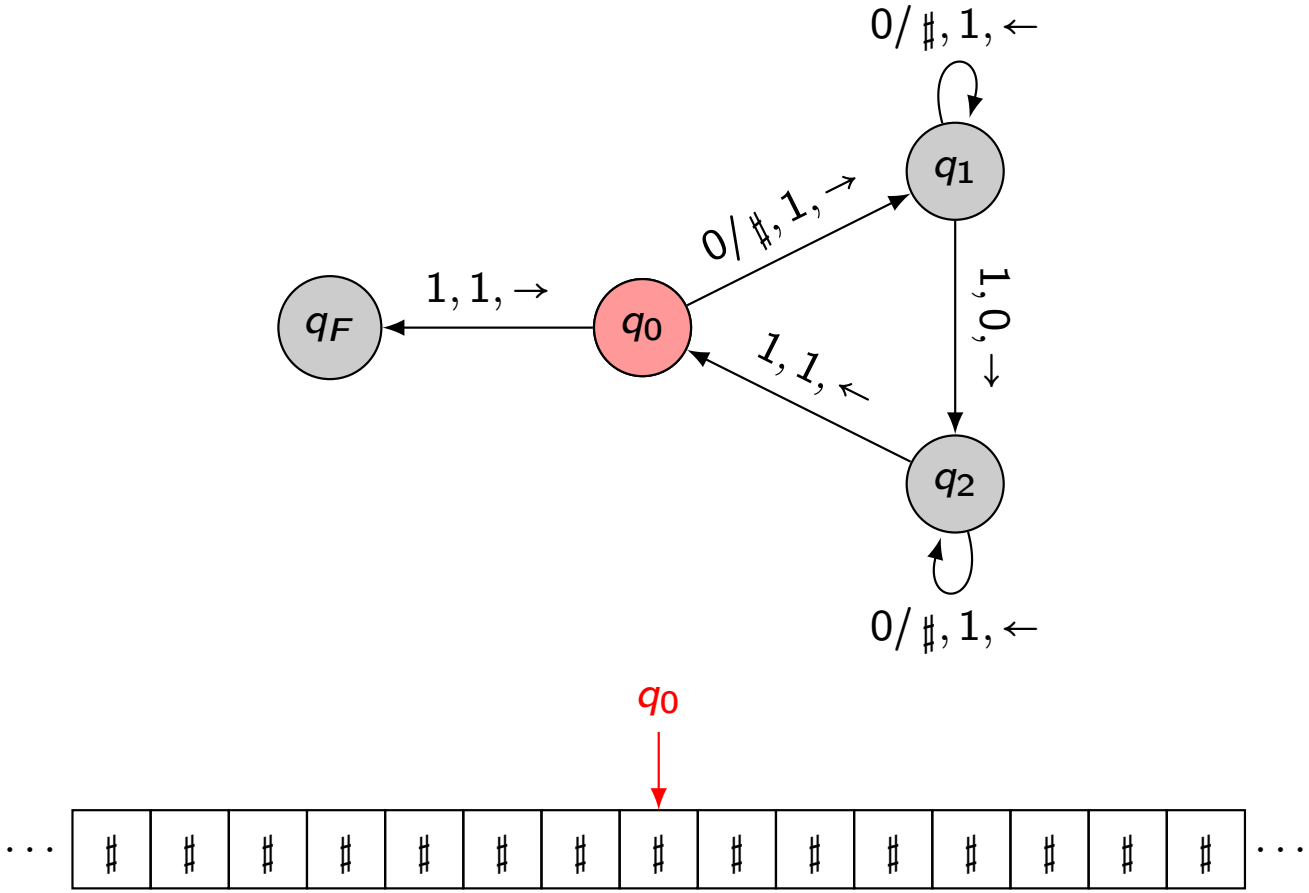
Une *machine de Turing* est la donnée $\mathcal{M} = (Q, \Gamma, \sharp, q_0, \delta, Q_F)$:

- Q: nombre fini d'états de la machine;
- q_0: état initial;
- q_F: état final;
- Γ alphabet fini;
- $\sharp \in \Gamma$ symbole blanc
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ fonction de transition.

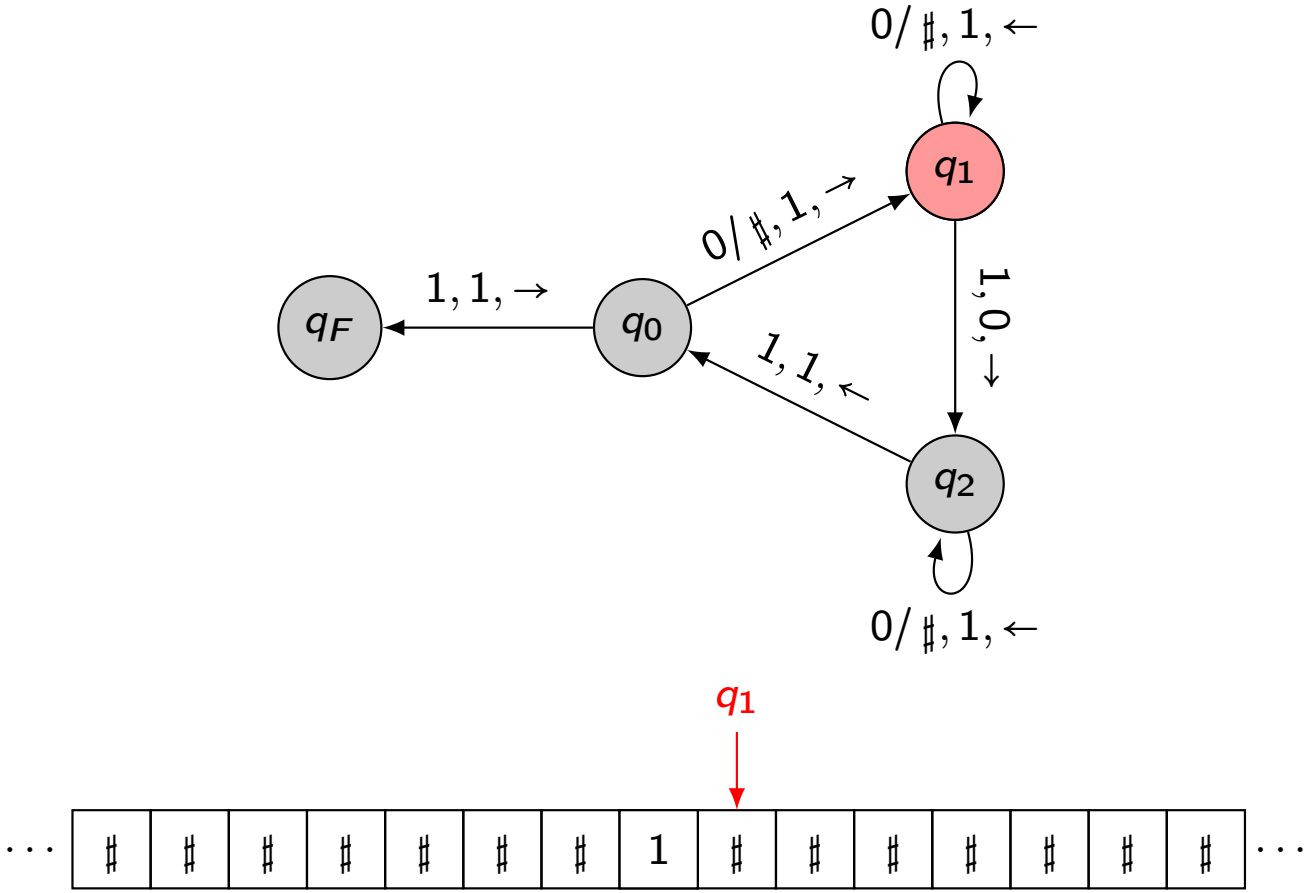


Étant donné un ruban $R \in \Gamma^{\mathbb{Z}}$ et une machine à la position $i \in \mathbb{Z}$ et l'état $q \in Q$, la machine calcule $\delta(q, R_i) = (q', a, \epsilon)$. Puis elle écrit a à la position i, se positionne en $i + \epsilon$ et se place à l'état q'.

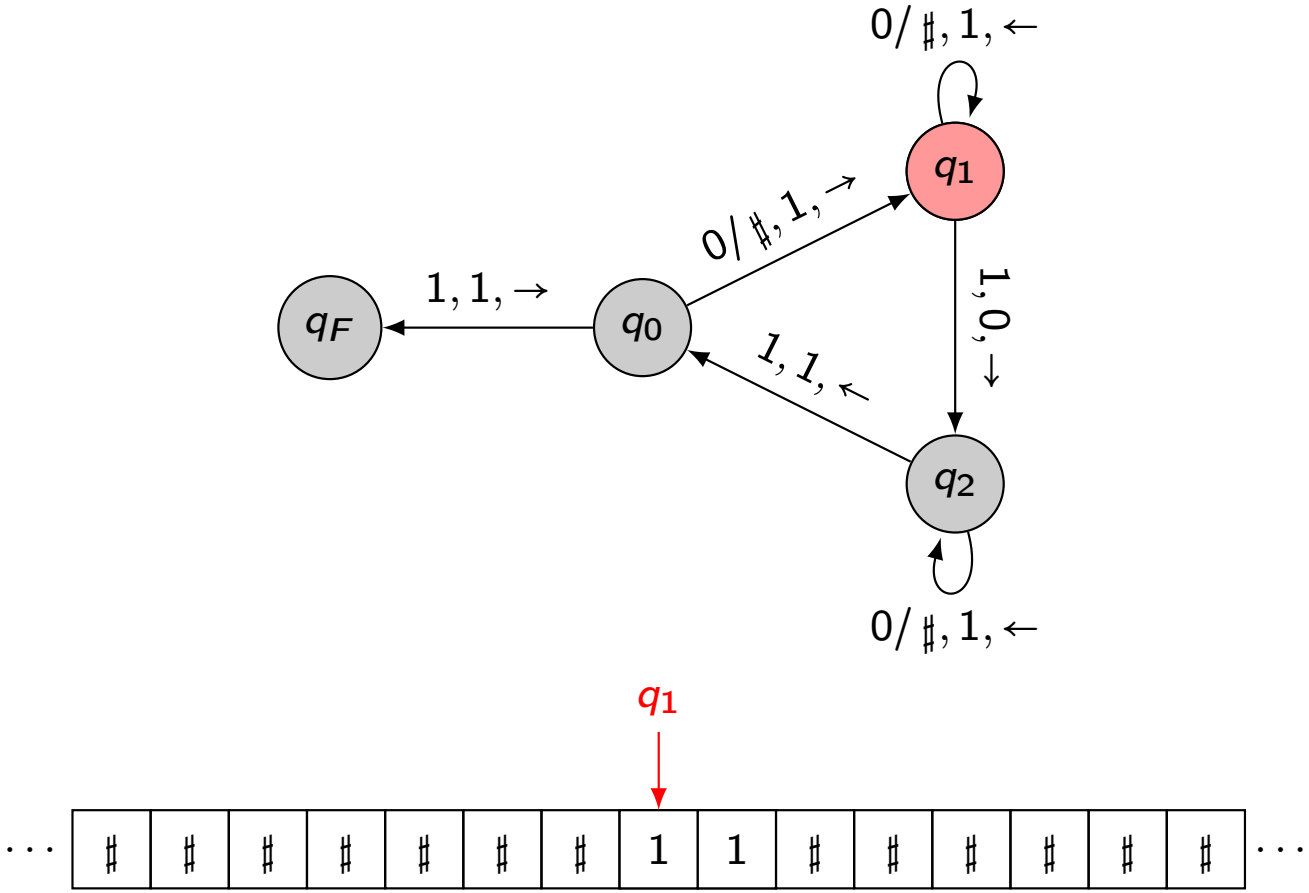
Fonctionnement d'une machine de Turing



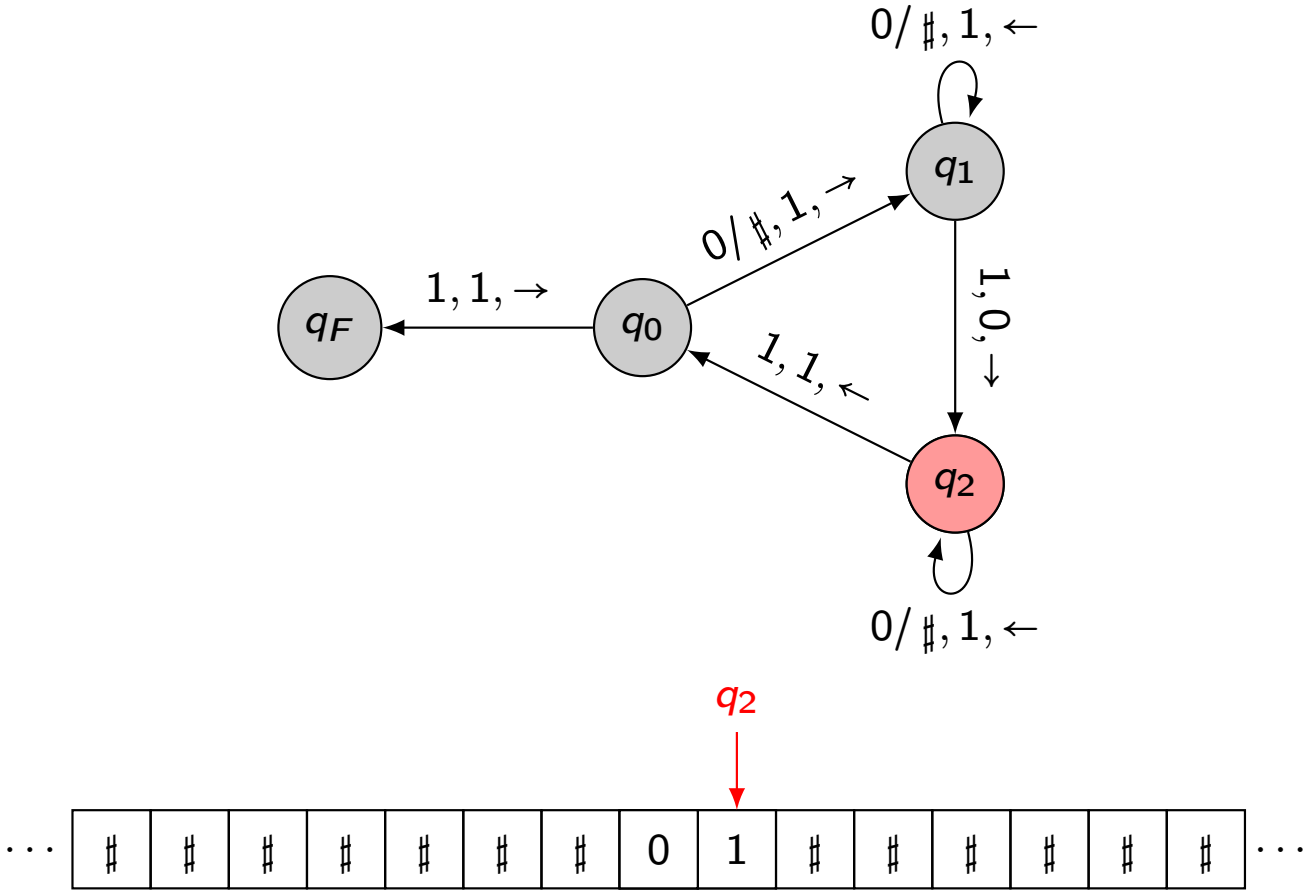
Fonctionnement d'une machine de Turing



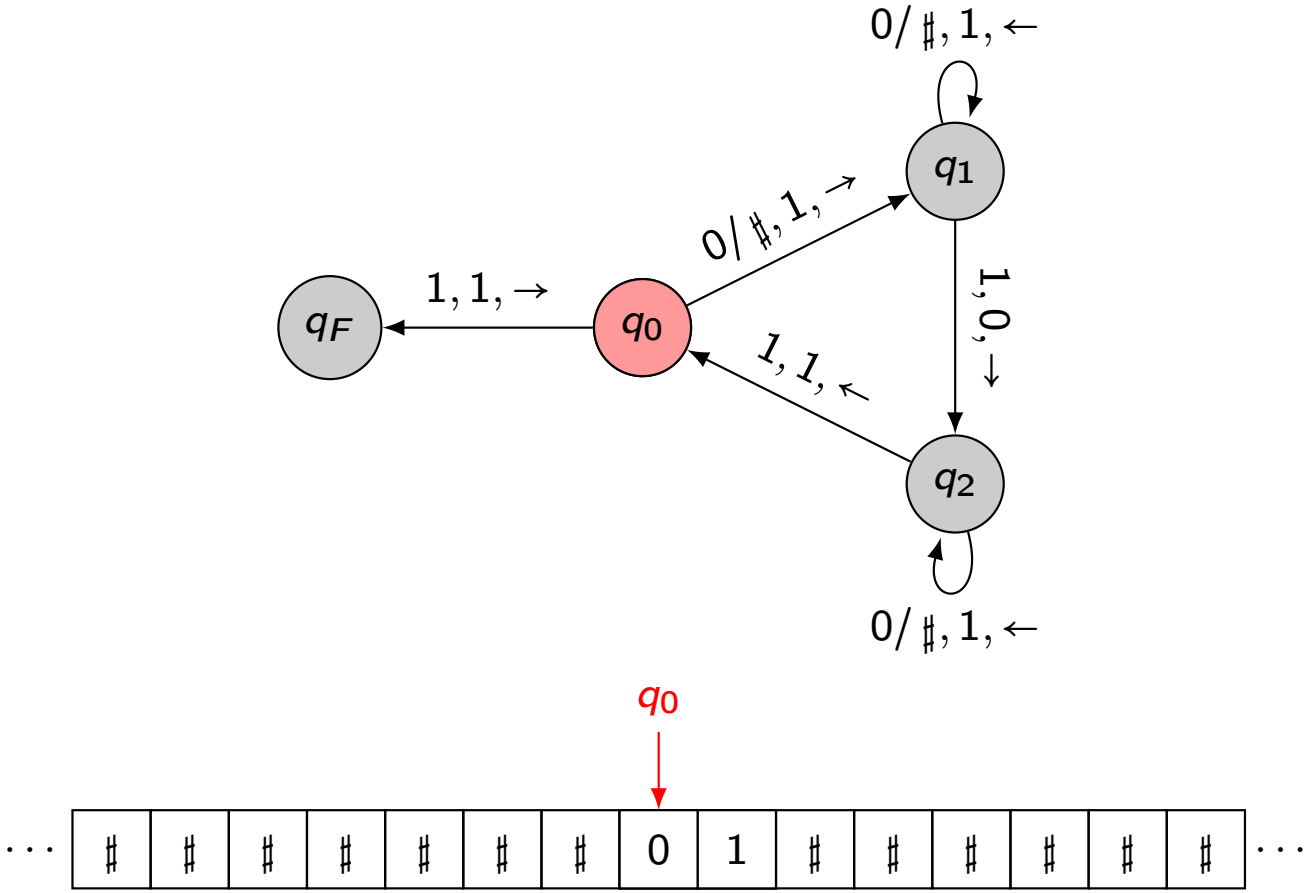
Fonctionnement d'une machine de Turing



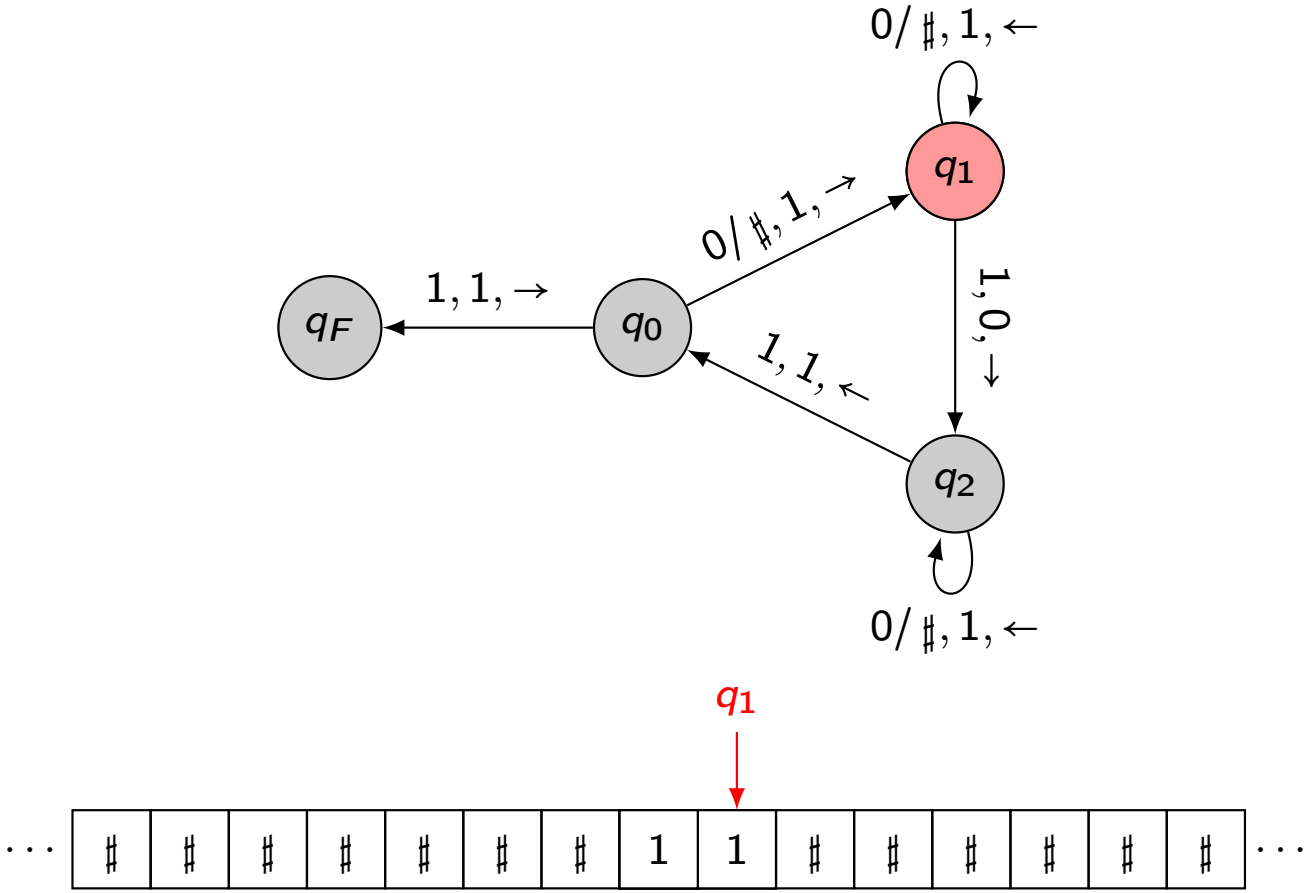
Fonctionnement d'une machine de Turing



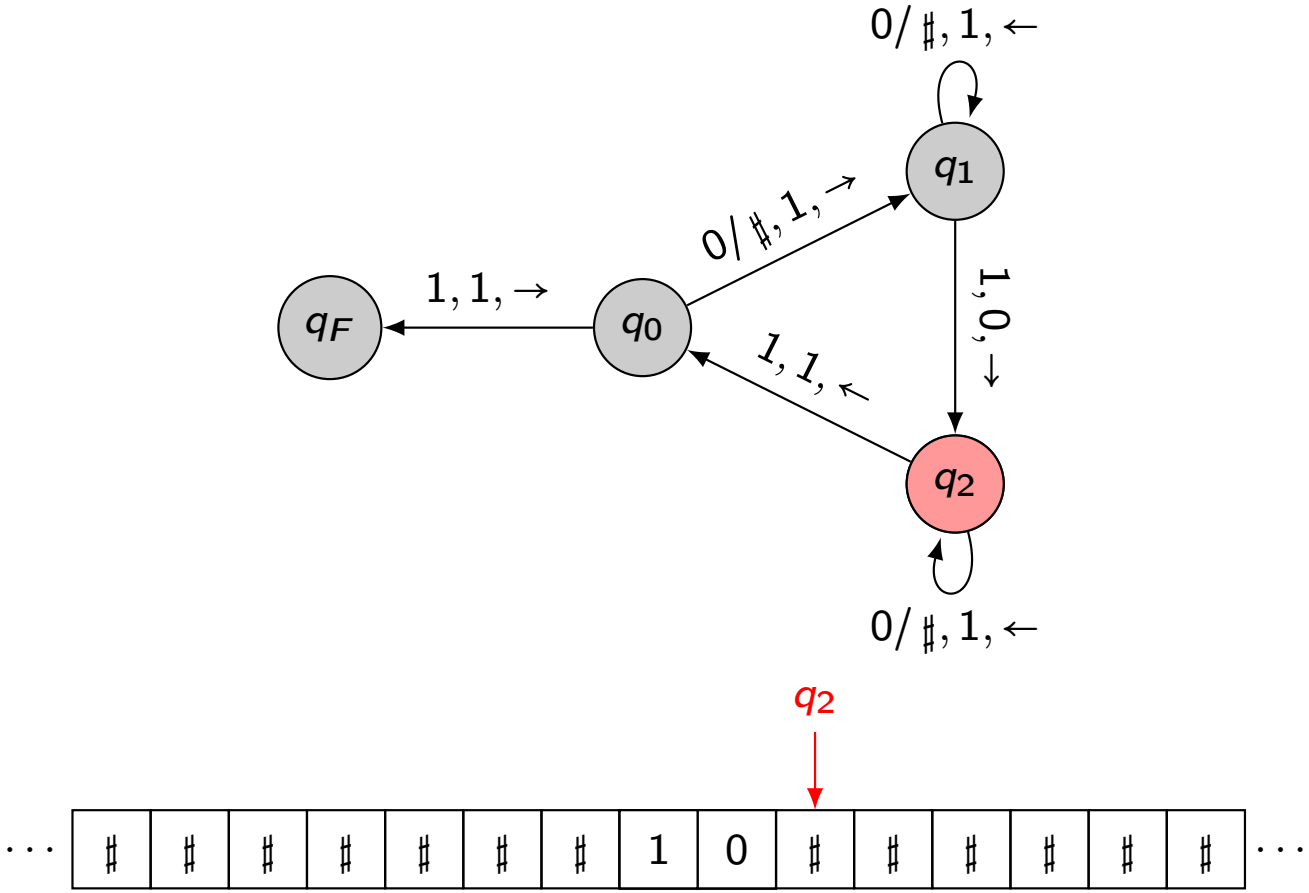
Fonctionnement d'une machine de Turing



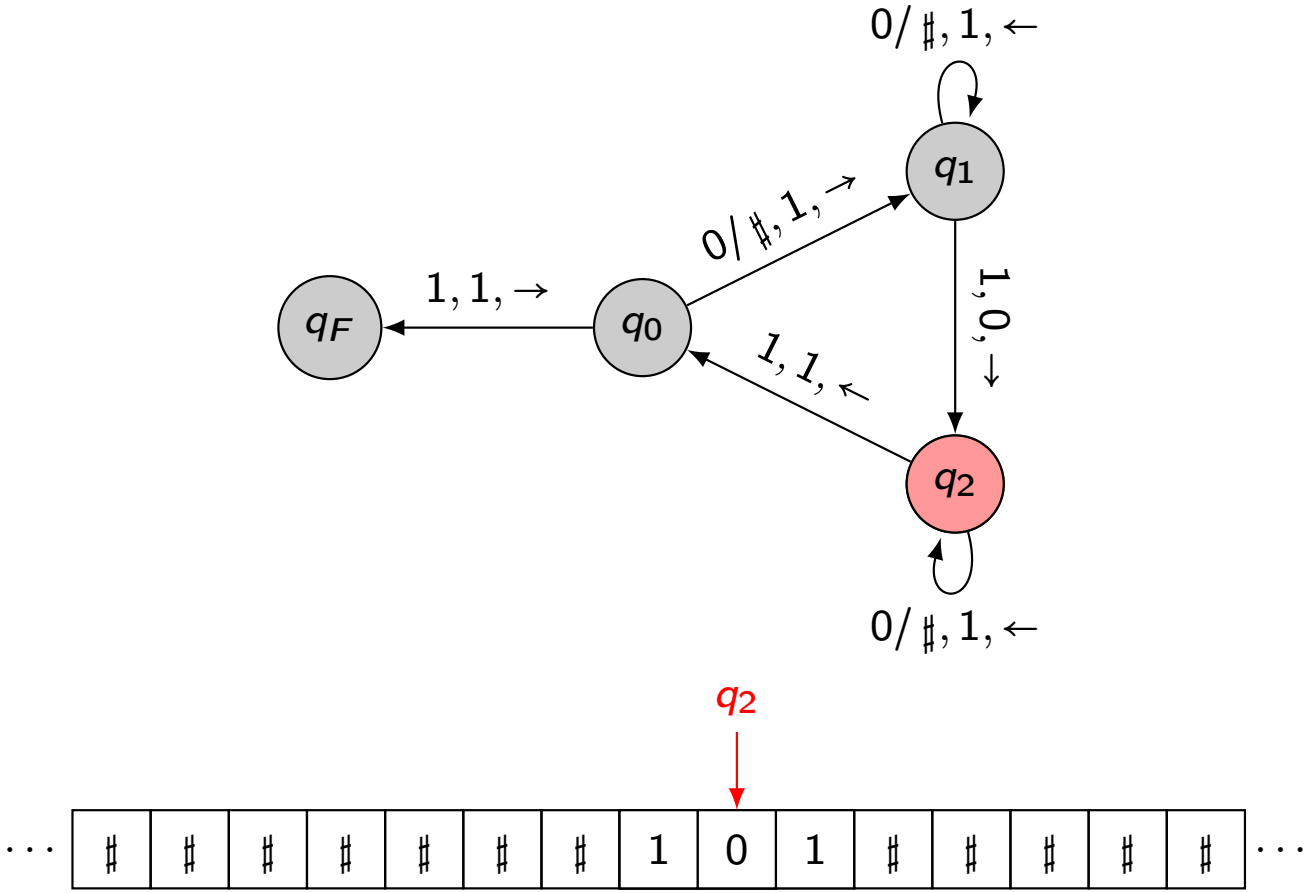
Fonctionnement d'une machine de Turing



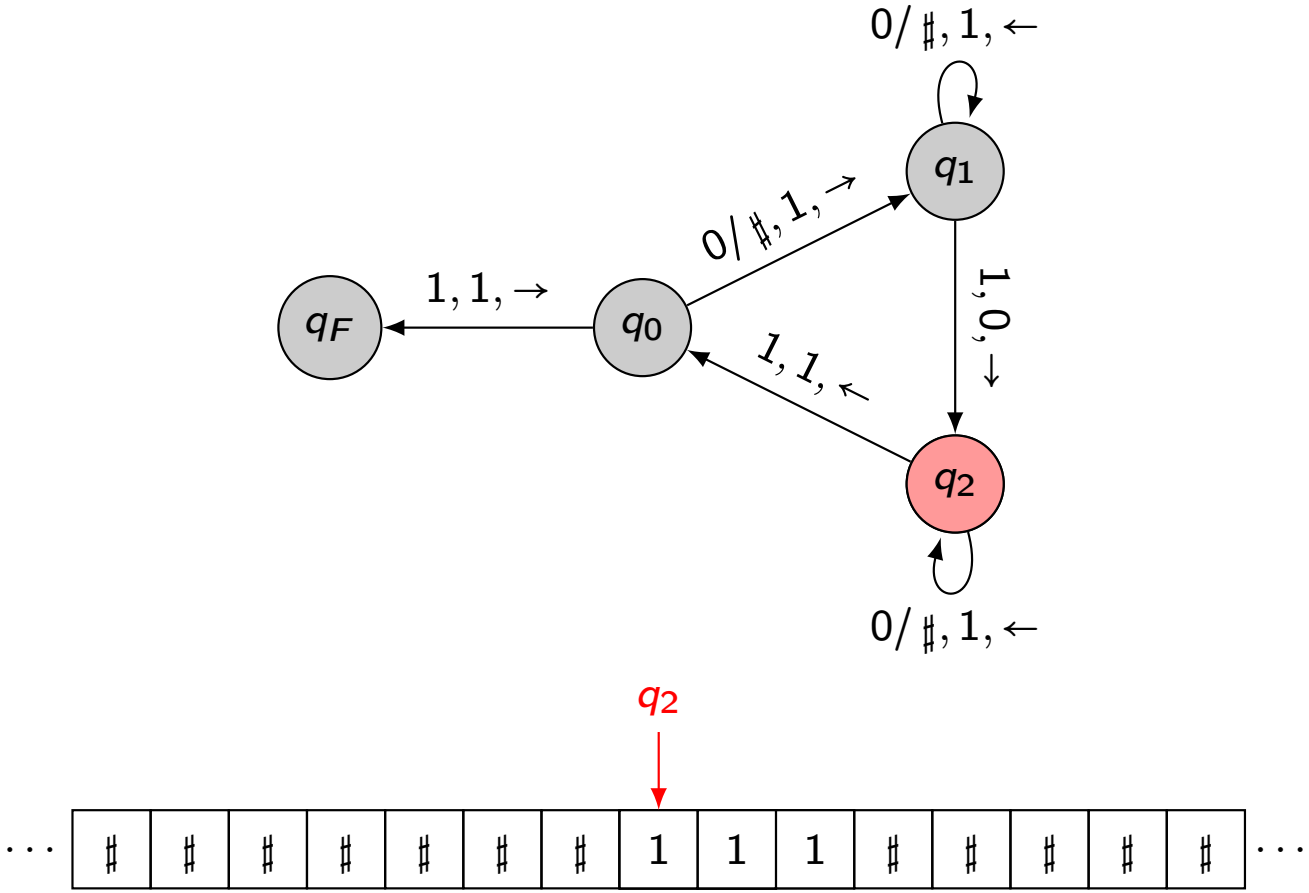
Fonctionnement d'une machine de Turing



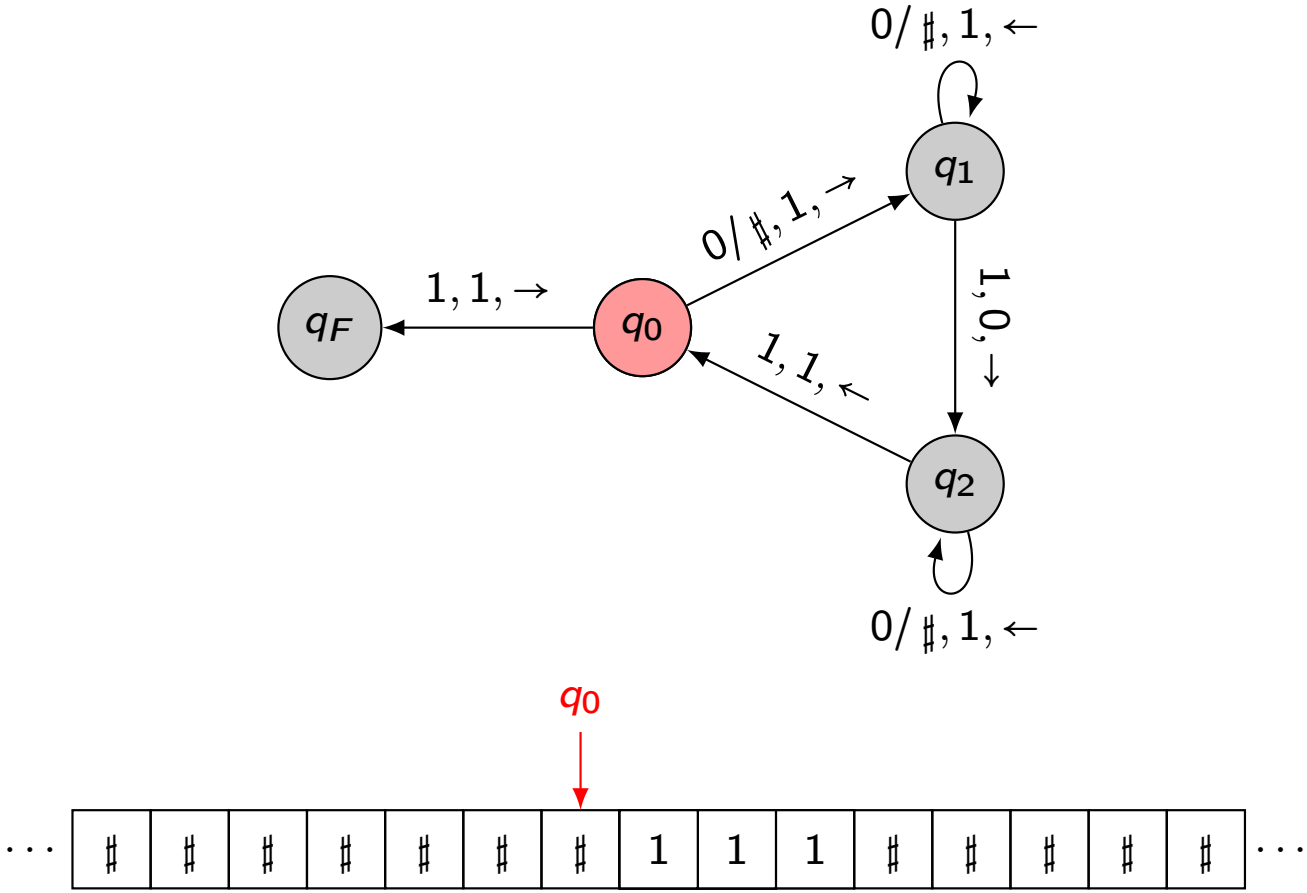
Fonctionnement d'une machine de Turing



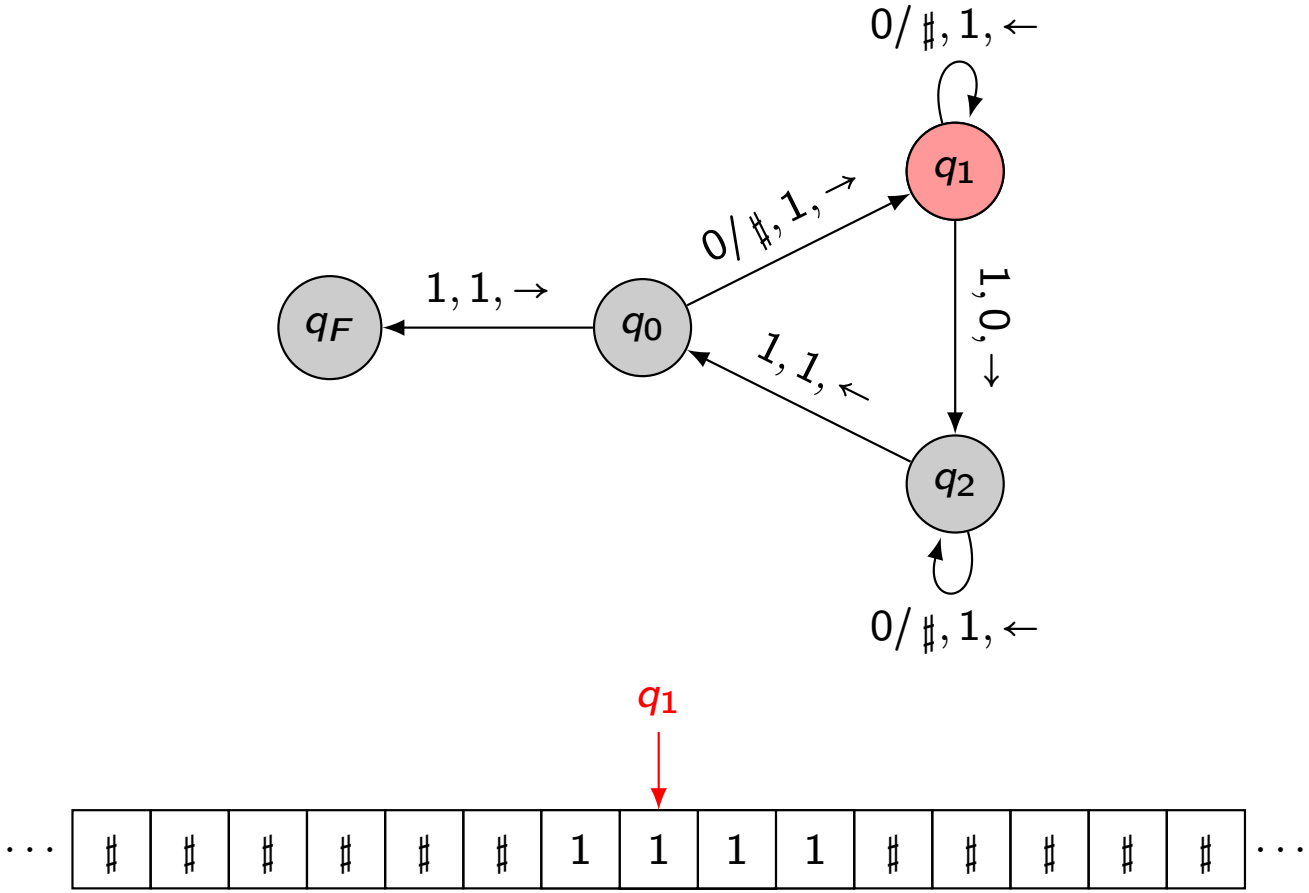
Fonctionnement d'une machine de Turing



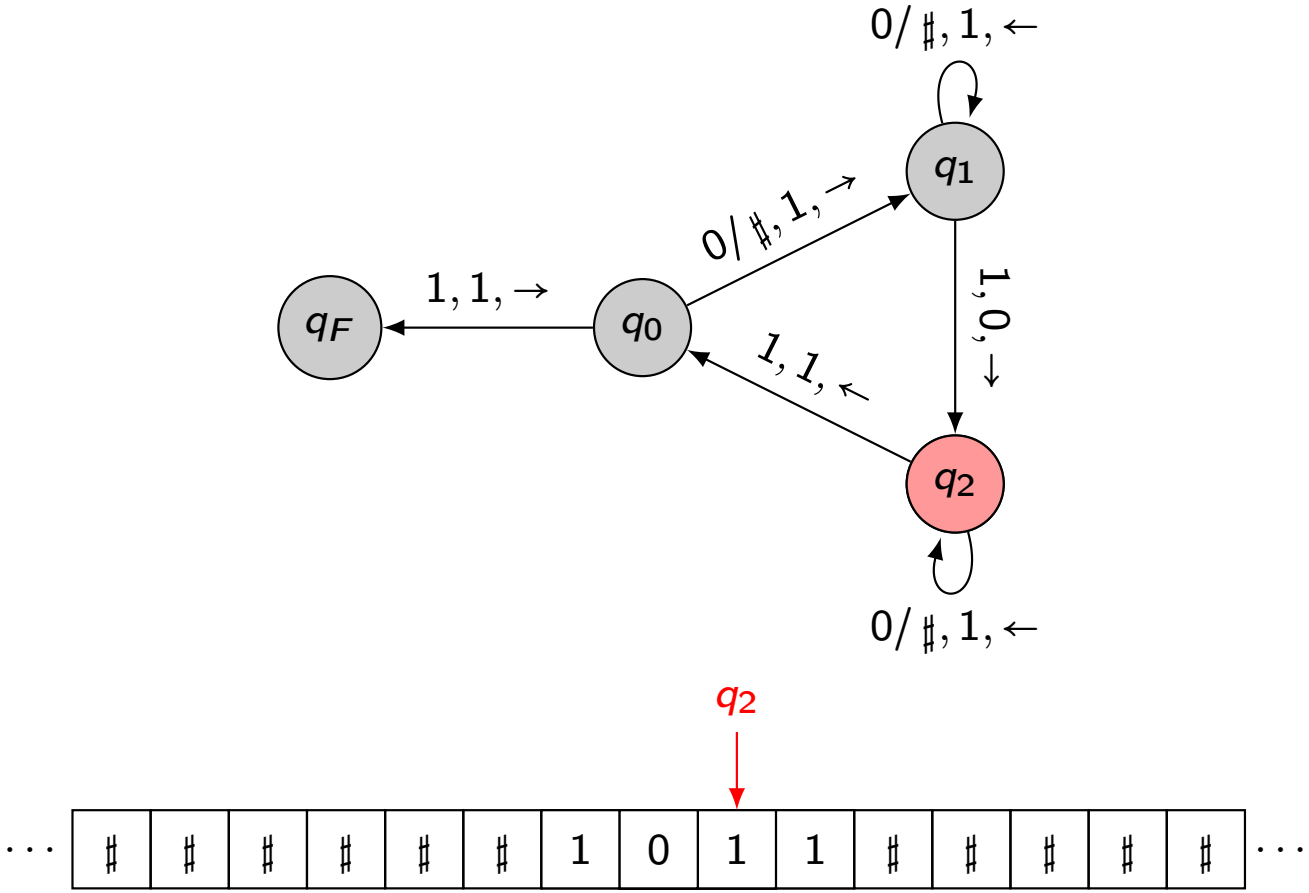
Fonctionnement d'une machine de Turing



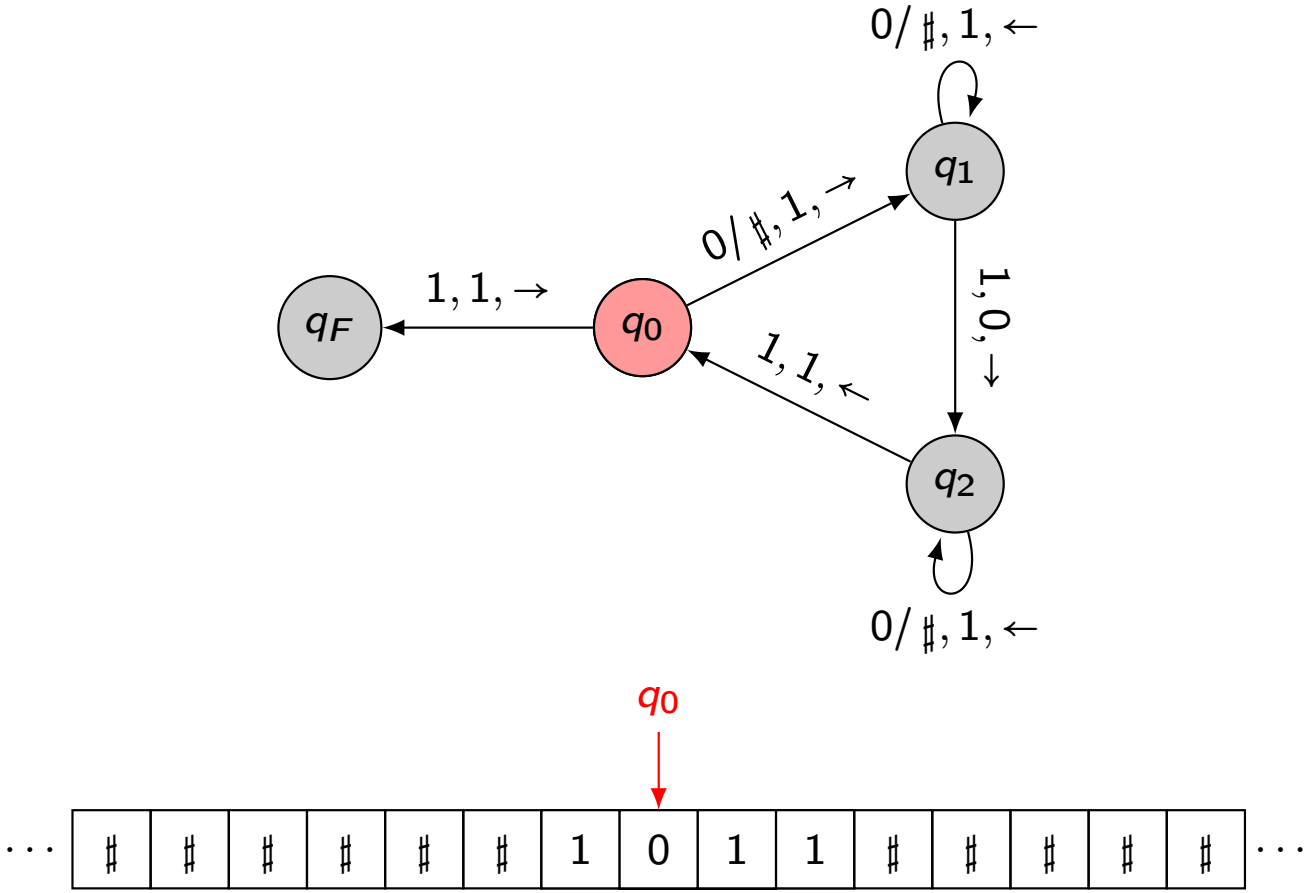
Fonctionnement d'une machine de Turing



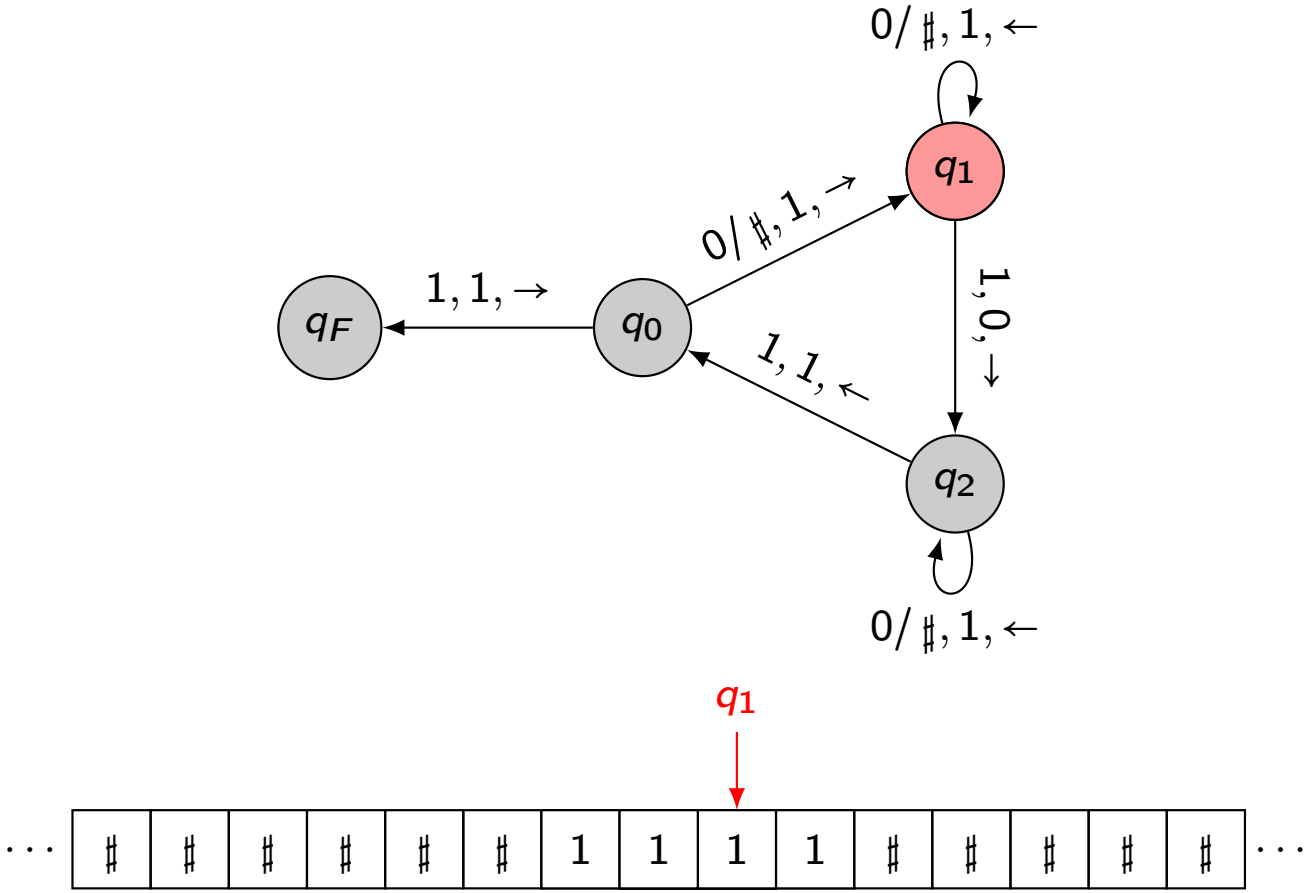
Fonctionnement d'une machine de Turing



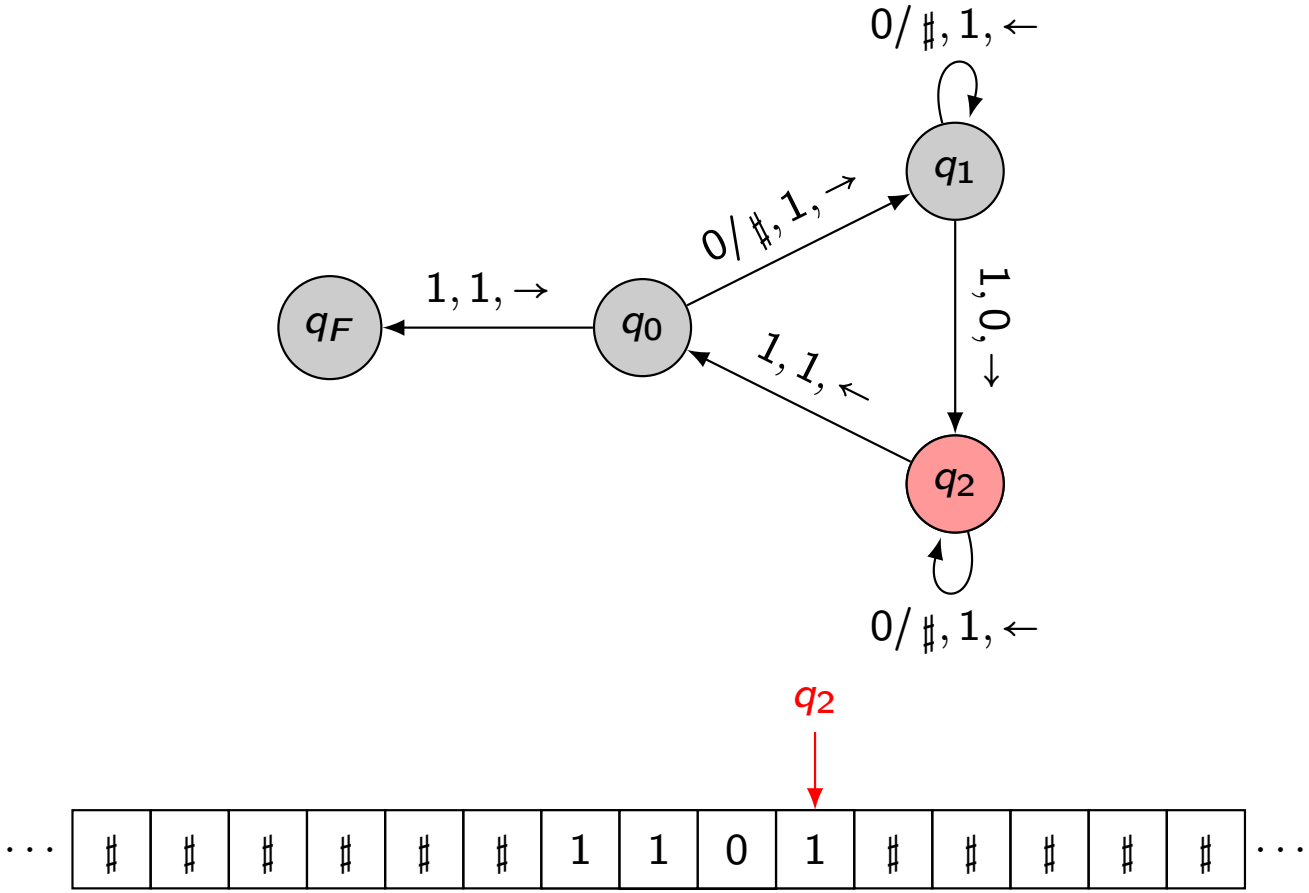
Fonctionnement d'une machine de Turing



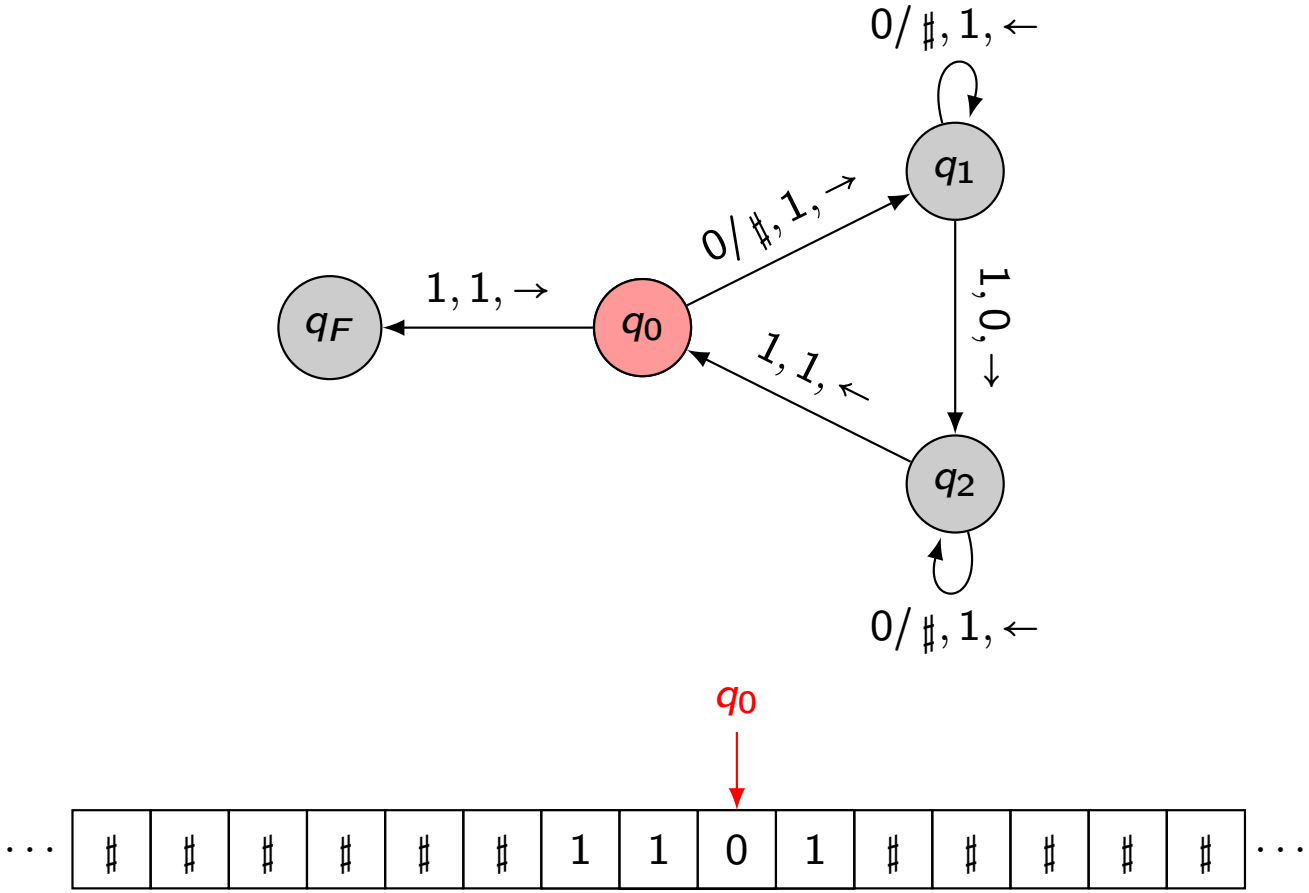
Fonctionnement d'une machine de Turing



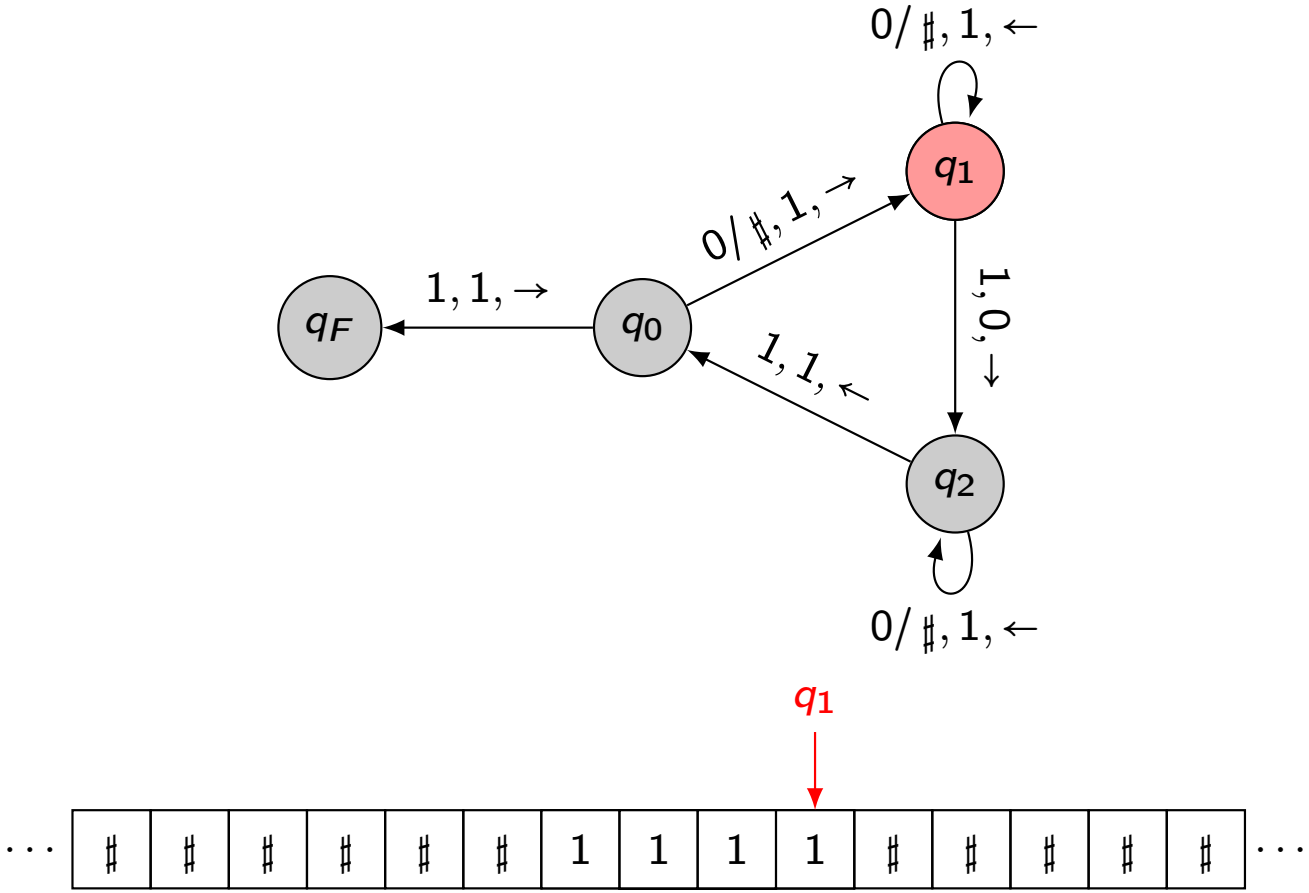
Fonctionnement d'une machine de Turing



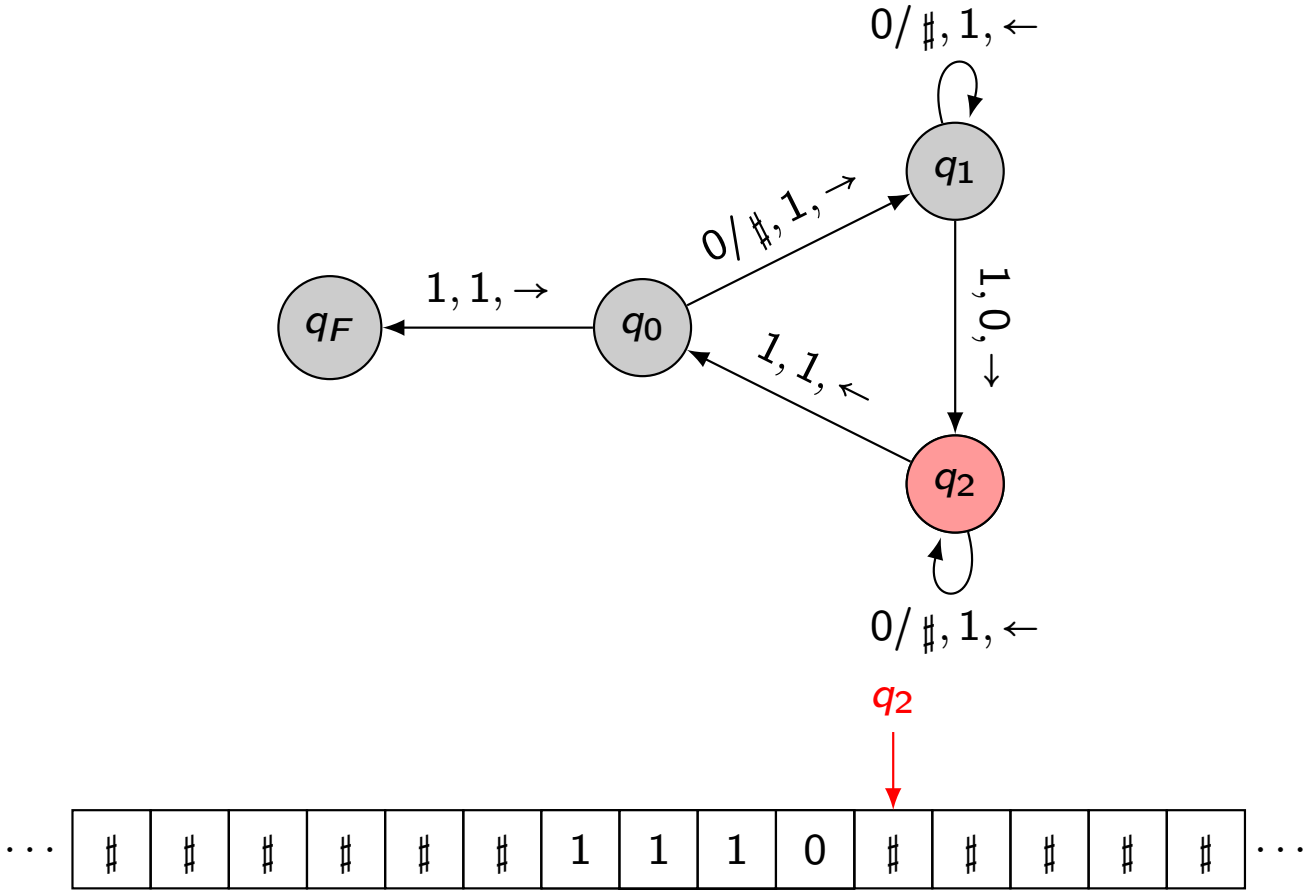
Fonctionnement d'une machine de Turing



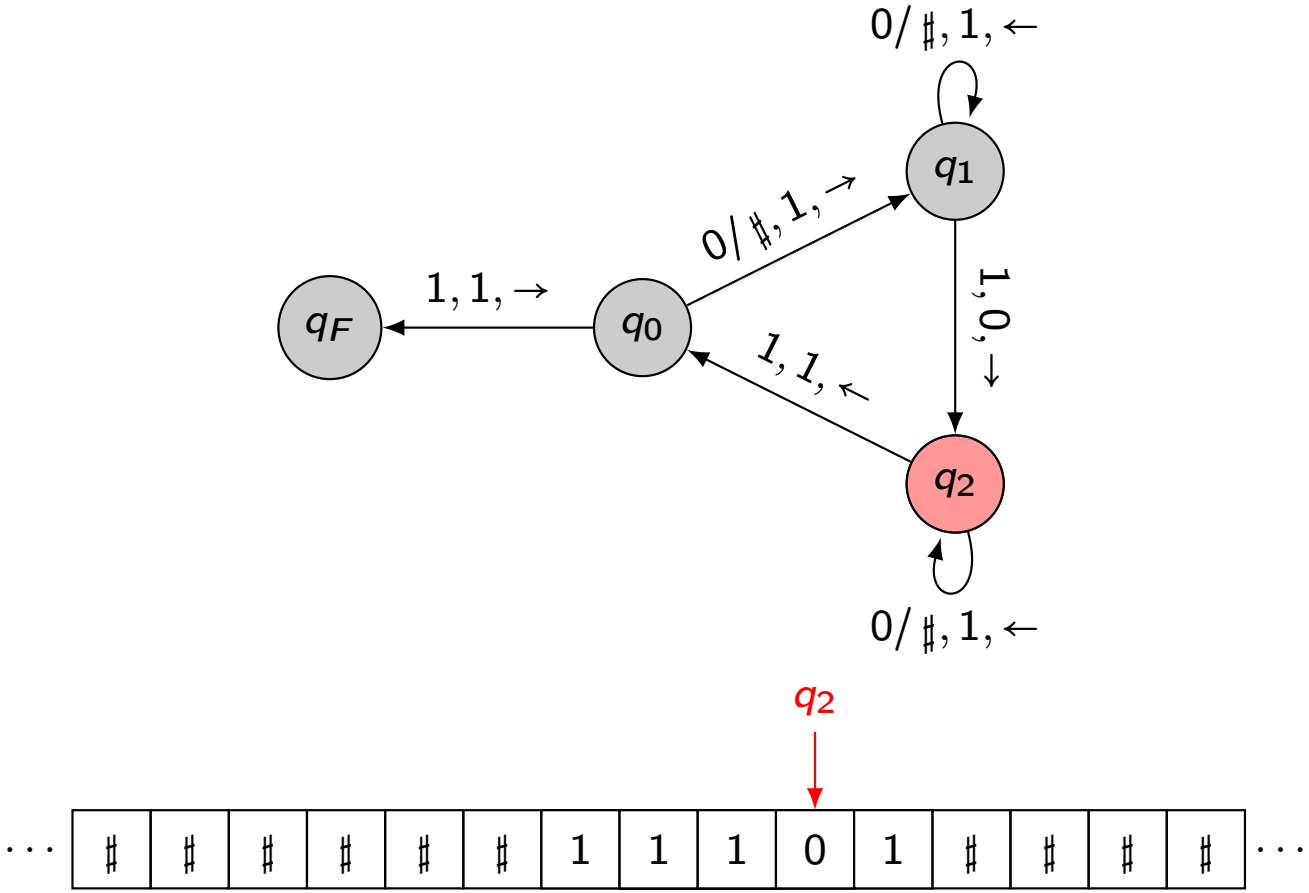
Fonctionnement d'une machine de Turing



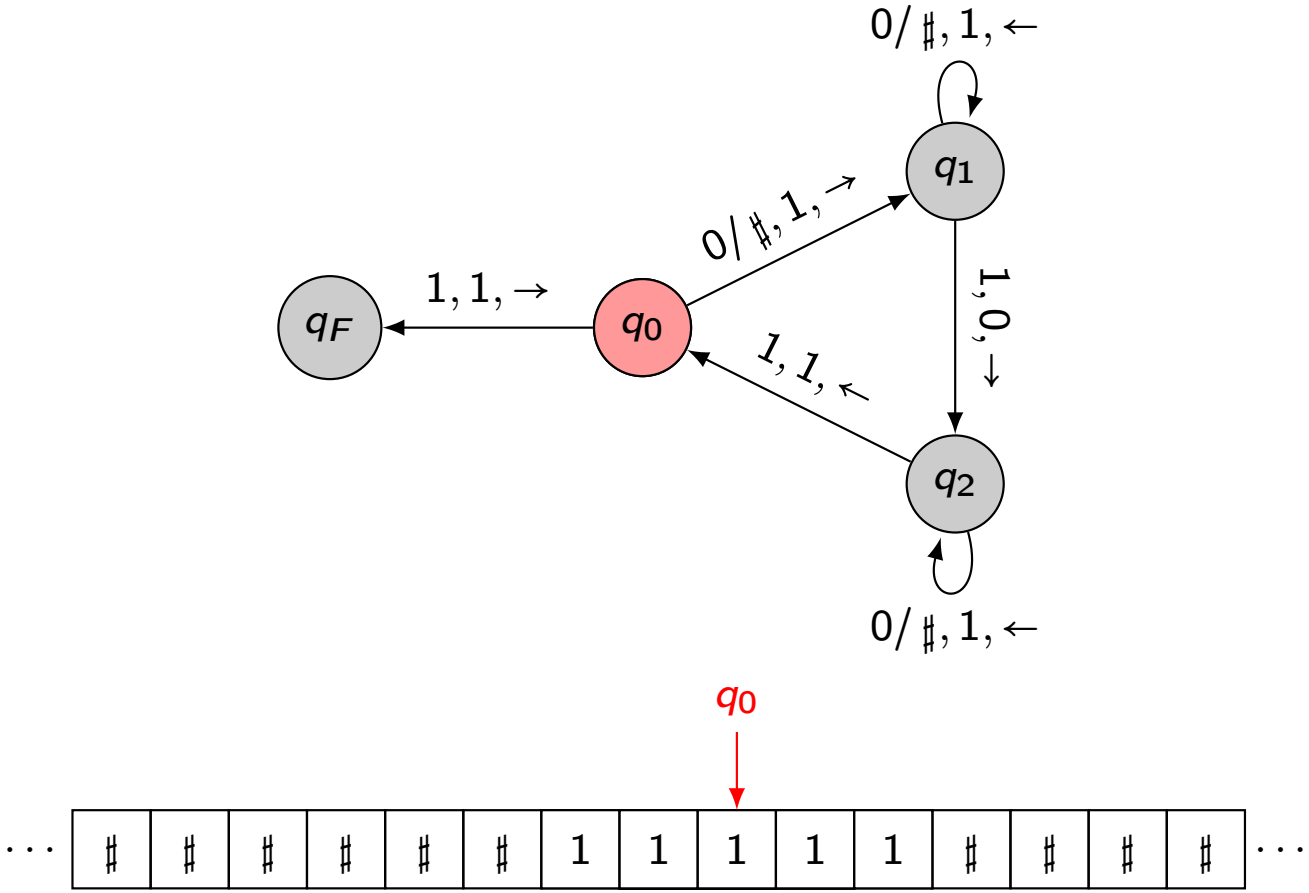
Fonctionnement d'une machine de Turing



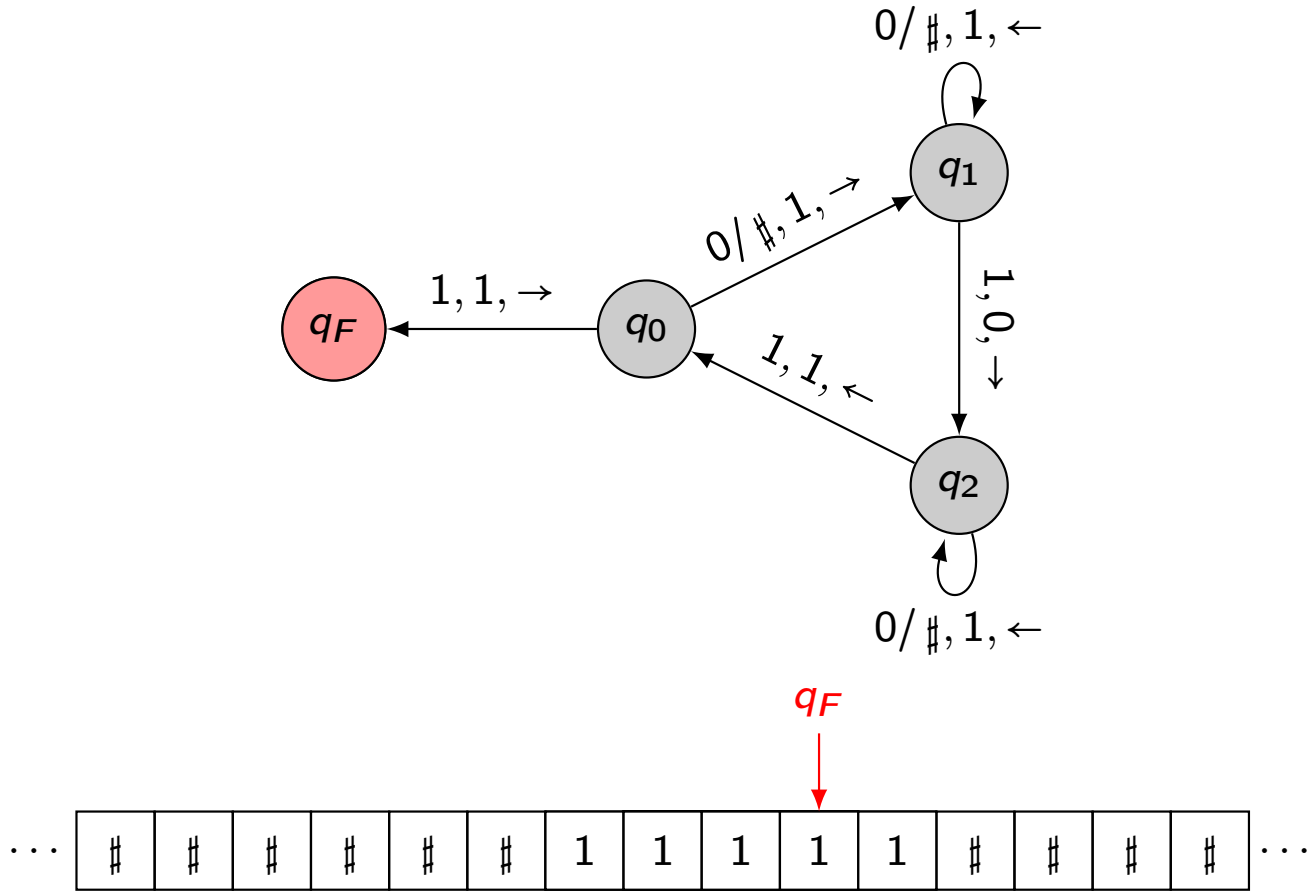
Fonctionnement d'une machine de Turing



Fonctionnement d'une machine de Turing

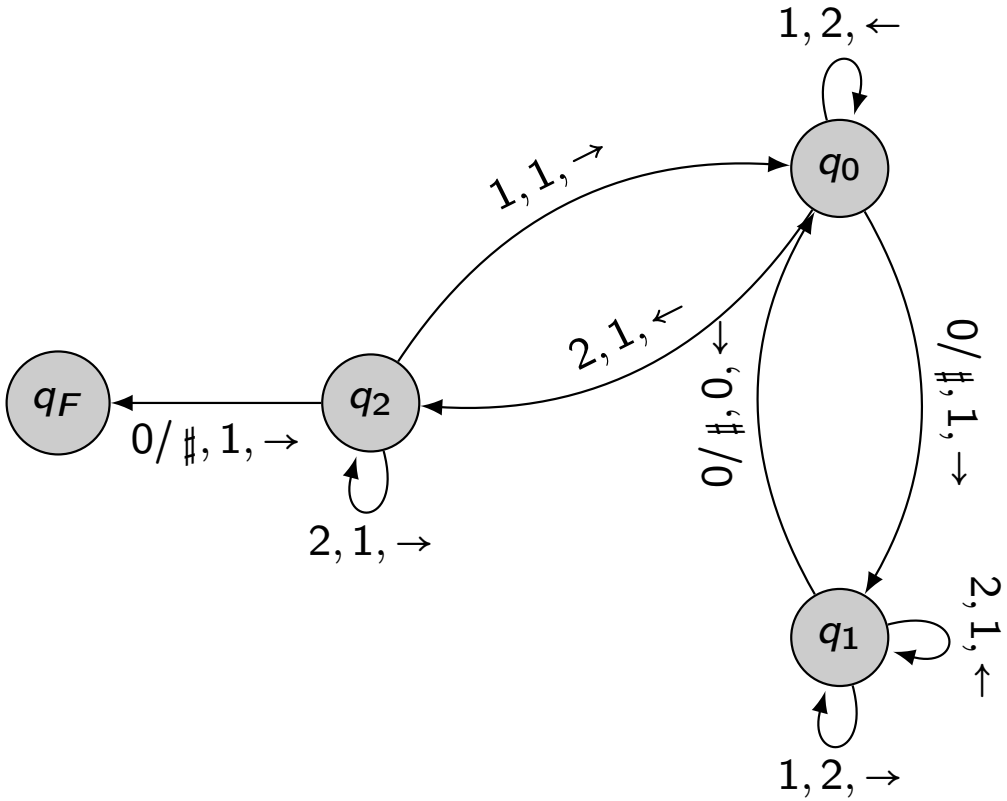


Fonctionnement d'une machine de Turing



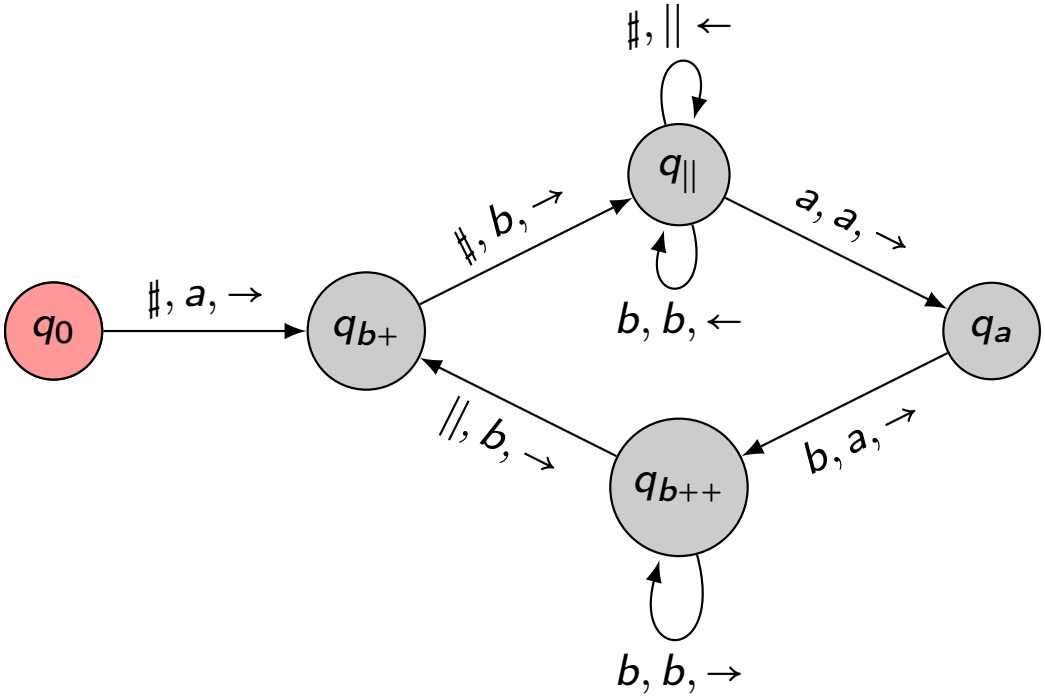
Cette machine de Turing s'arrête après 21 étapes de calculs.

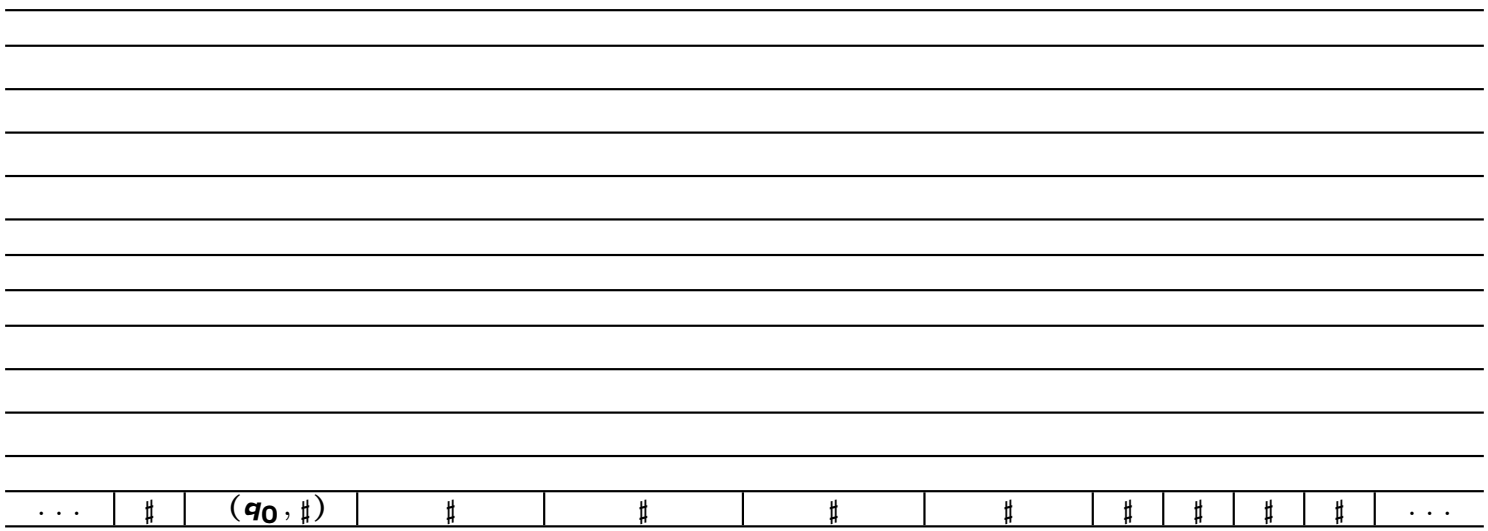
Monstre combinatoire



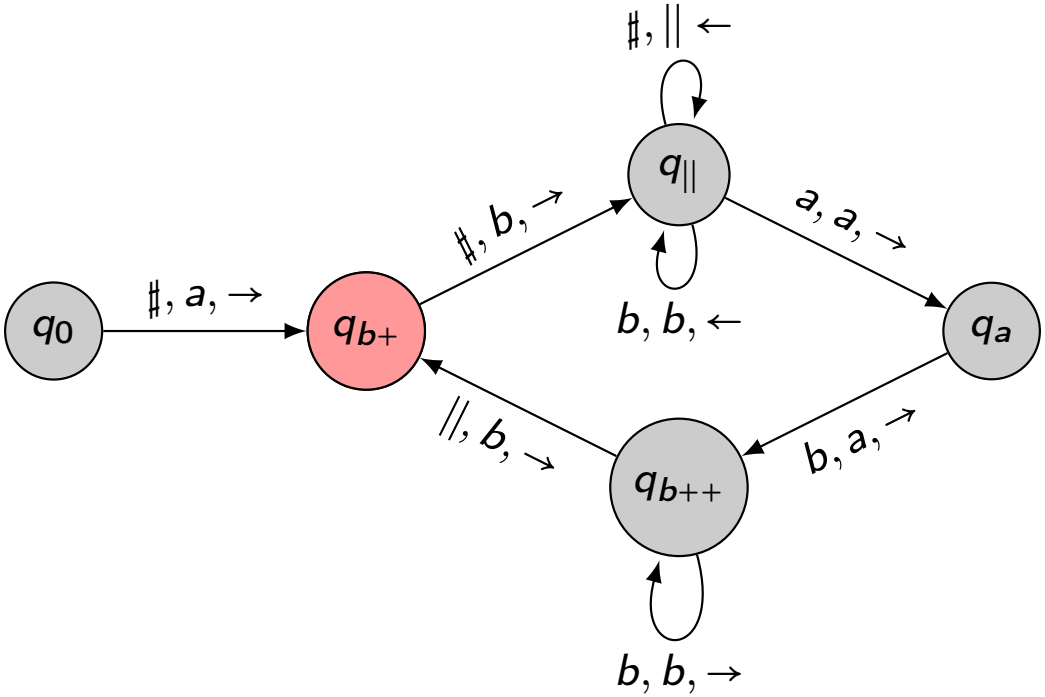
Partant d'un ruban vide, cette machine écrit 374×10^6 lettres en 119×10^{15} pas de calcul.

Fonctionnement d'une machine de Turing



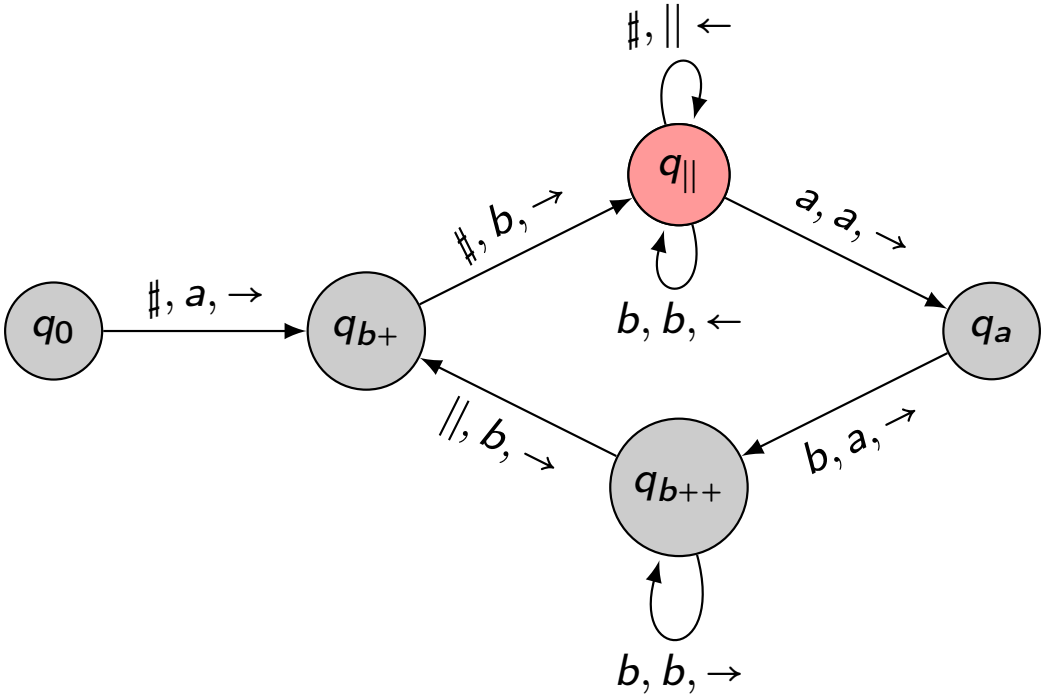


Fonctionnement d'une machine de Turing



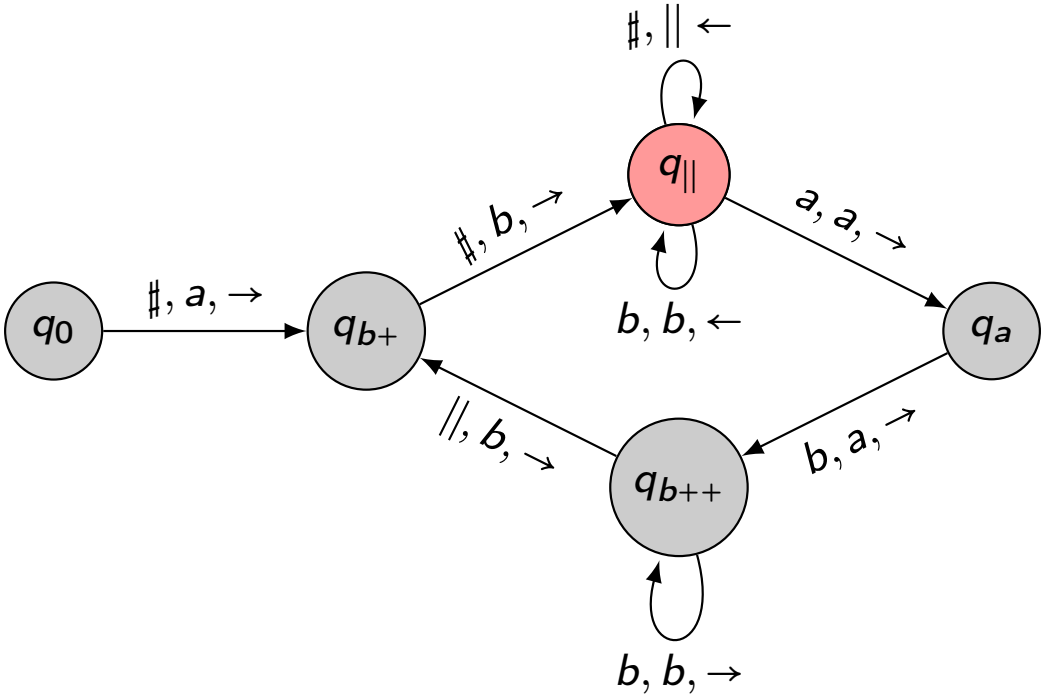
...	♯	a	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing



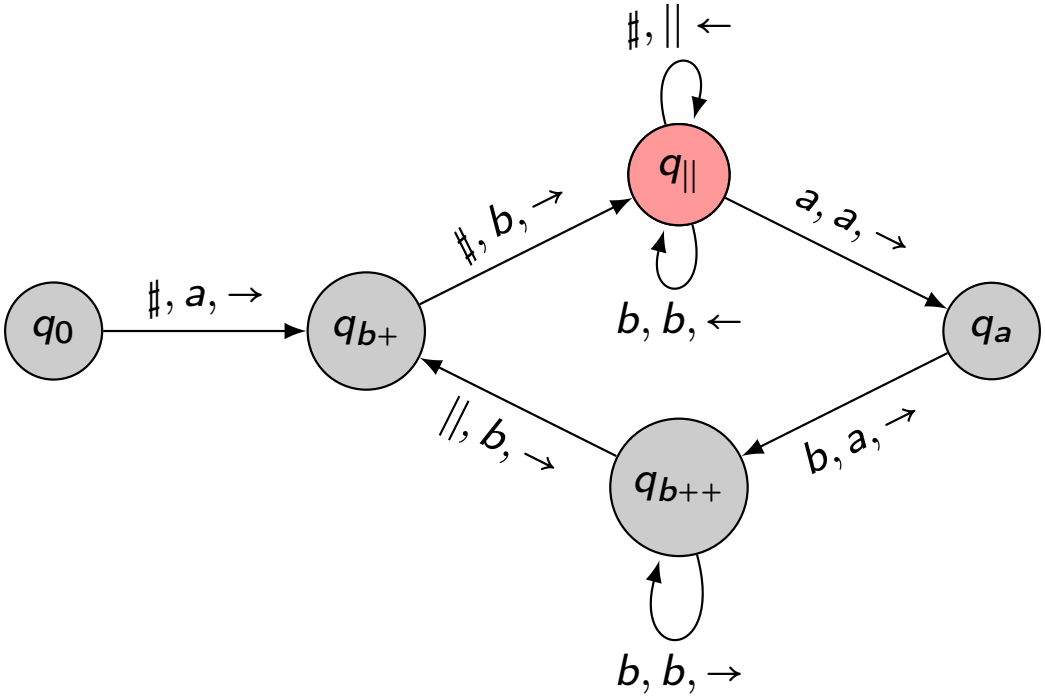
...	♯	a	b	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	...
...	♯	a	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing



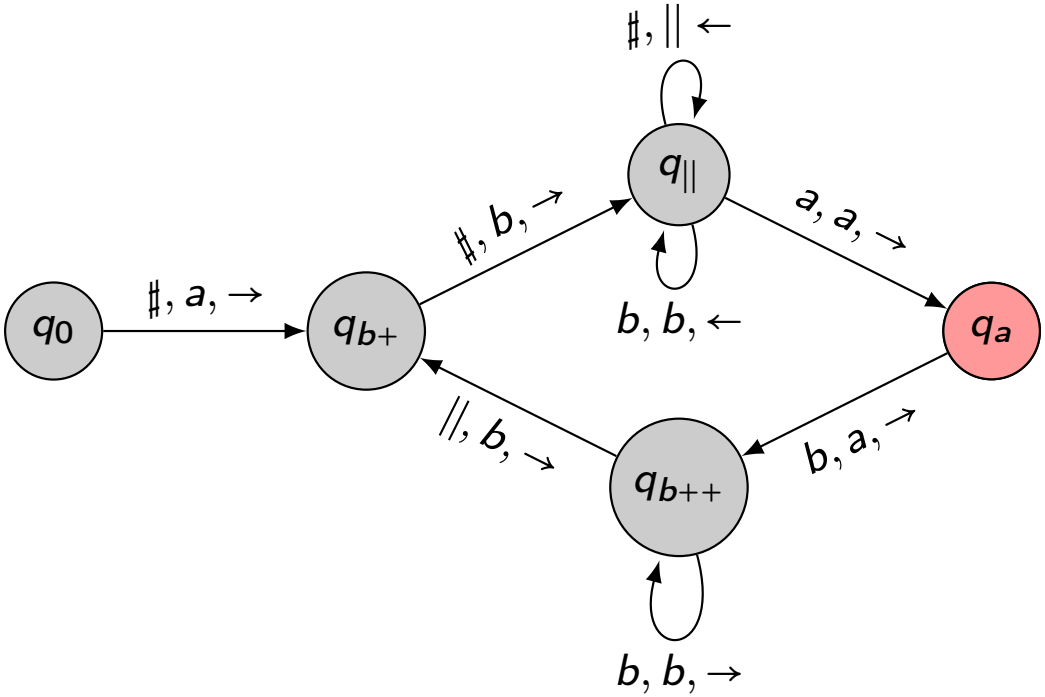
...	♯	a	$(q_{\|\|}, b)$	|	♯	♯	♯	♯	♯	♯	...
...	♯	a	b	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	...
...	♯	a	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing



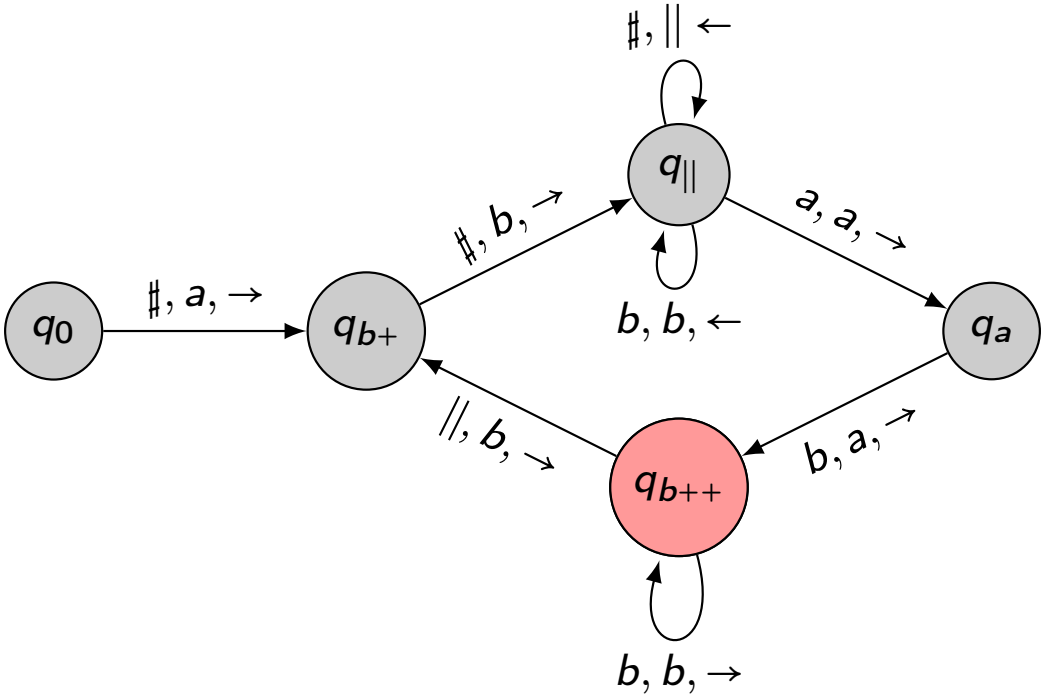
...	♯	$(q_{\|\|}, a)$	b	$\|\|$	♯	♯	♯	♯	♯	♯	...
...	♯	a	$(q_{\|\|}, b)$	$\|\|$	♯	♯	♯	♯	♯	♯	...
...	♯	a	b	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	...
...	♯	a	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing



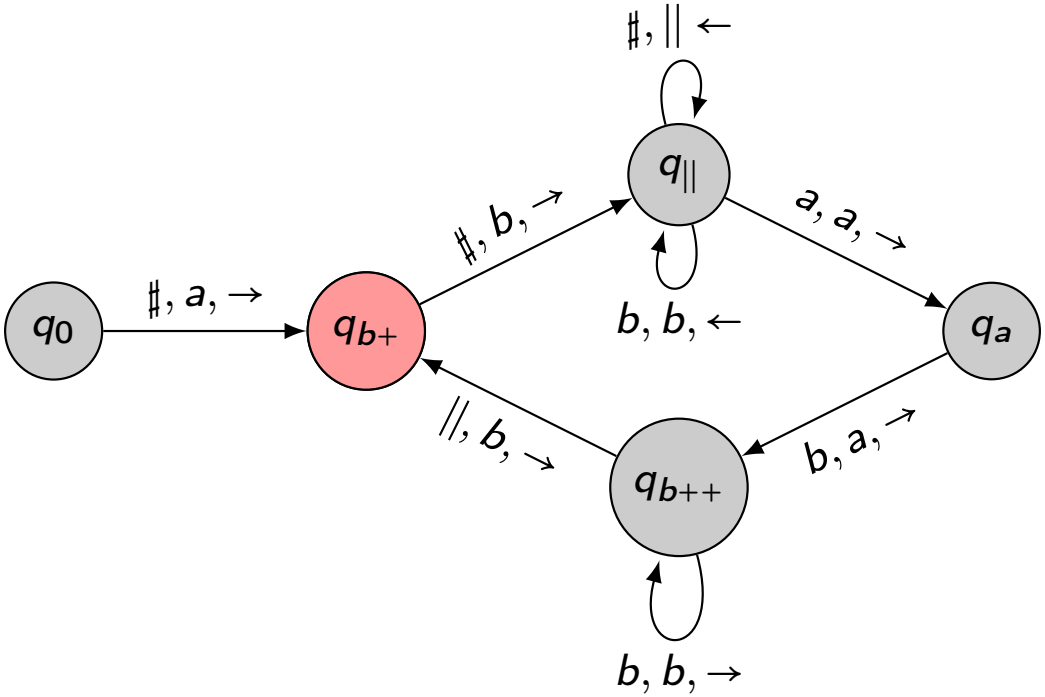
...	♯	a	(q_{a+}, b)	||	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_{\|\|}, a)$	b	||	♯	♯	♯	♯	♯	♯	...
...	♯	a	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	♯	♯	...
...	♯	a	b	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	...
...	♯	a	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing



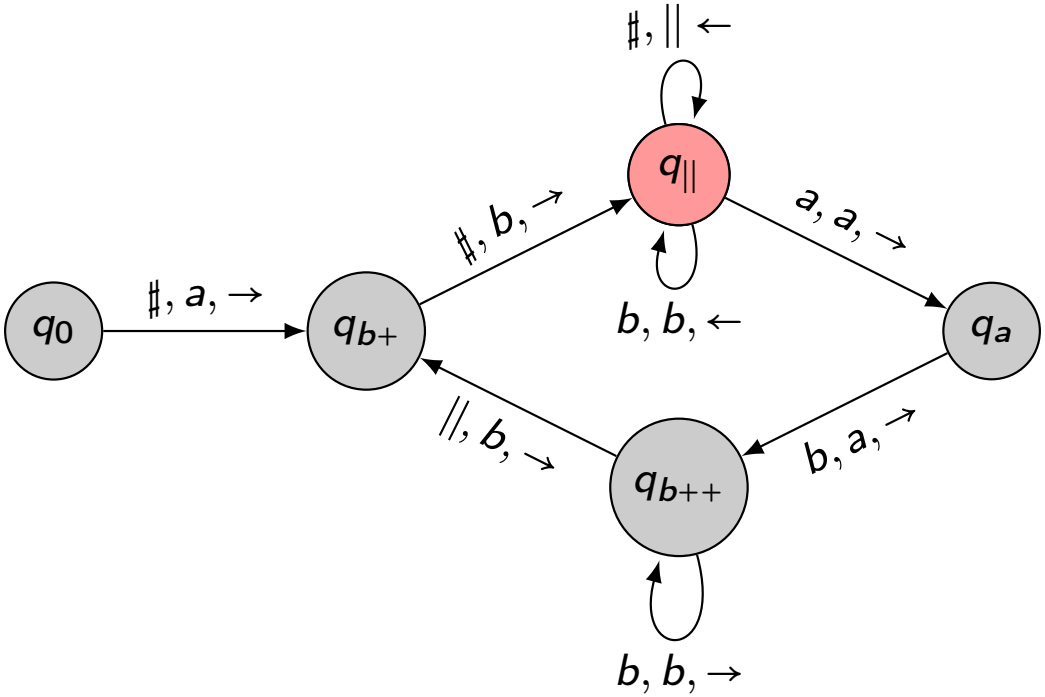
...	♯	**a**	**a**	$(q_{b++}, \|\|)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	(q_{a+}, b)	||	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_{\|\|}, a)$	**b**	||	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing



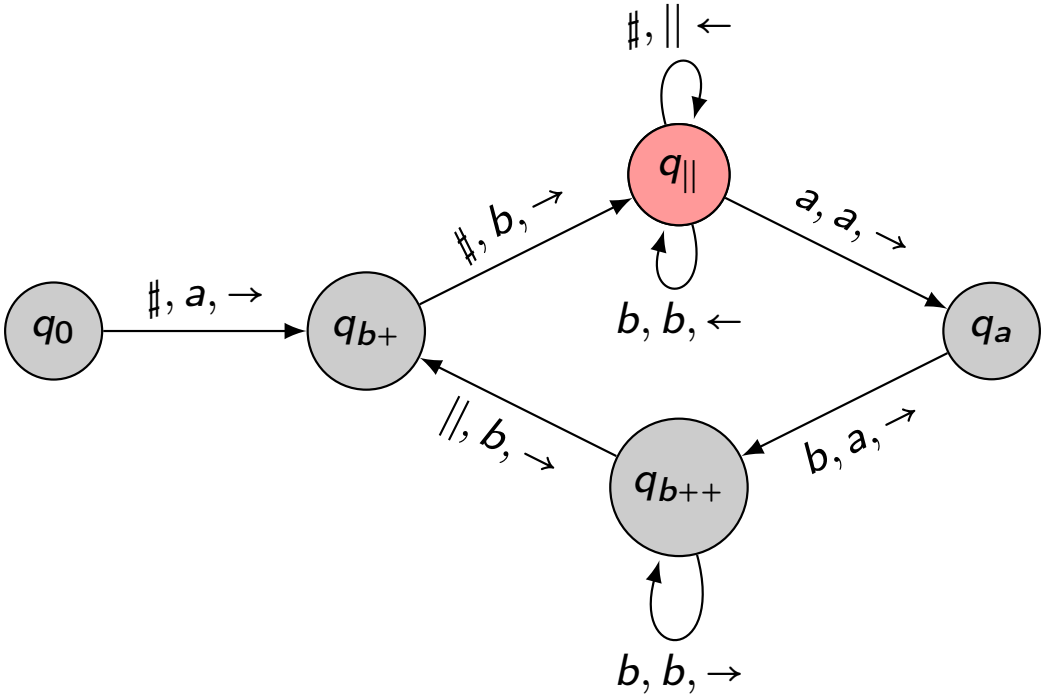
...	♯	**a**	**a**	**b**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	...
...	♯	**a**	**a**	$(q_{b++}, \|\|)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	(q_{a+}, b)	||	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_{\|\|}, a)$	**b**	||	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing



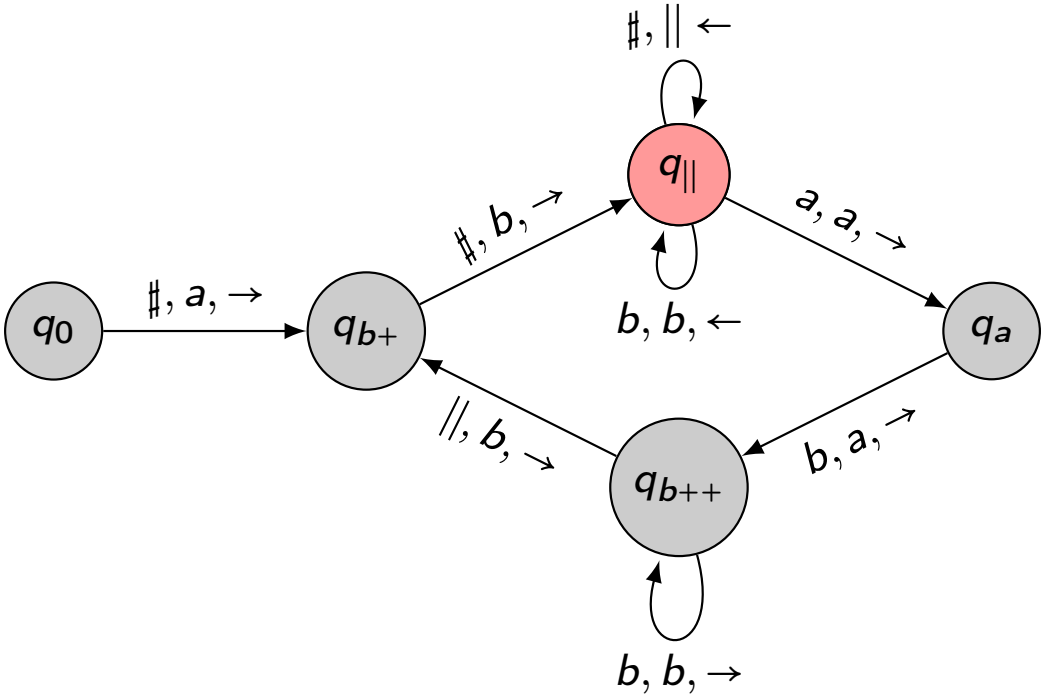
...	♯	**a**	**a**	**b**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	...
...	♯	**a**	**a**	**b**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	...
...	♯	**a**	**a**	$(q_{b++}, \|\|)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	(q_{a+}, b)	||	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_{\|\|}, a)$	**b**	||	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing



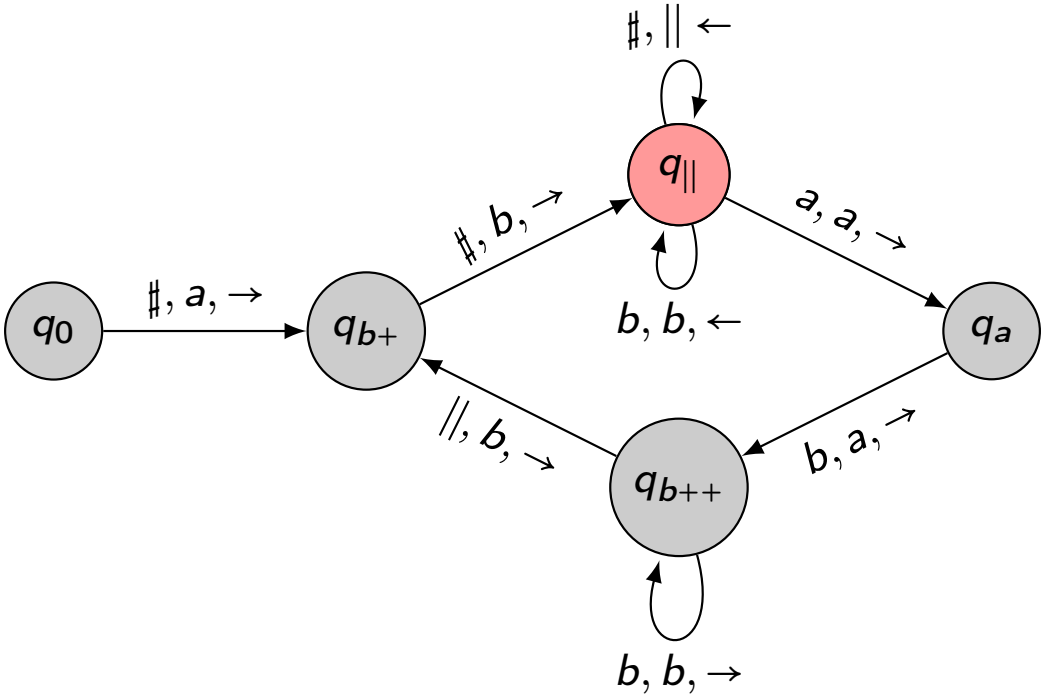
...	♯	**a**	**a**	**b**	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	...
...	♯	**a**	**a**	**b**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	...
...	♯	**a**	**a**	**b**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	...
...	♯	**a**	**a**	$(q_{b++}, \|\|)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	(q_{a+}, b)	||	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_{\|\|}, a)$	**b**	||	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing



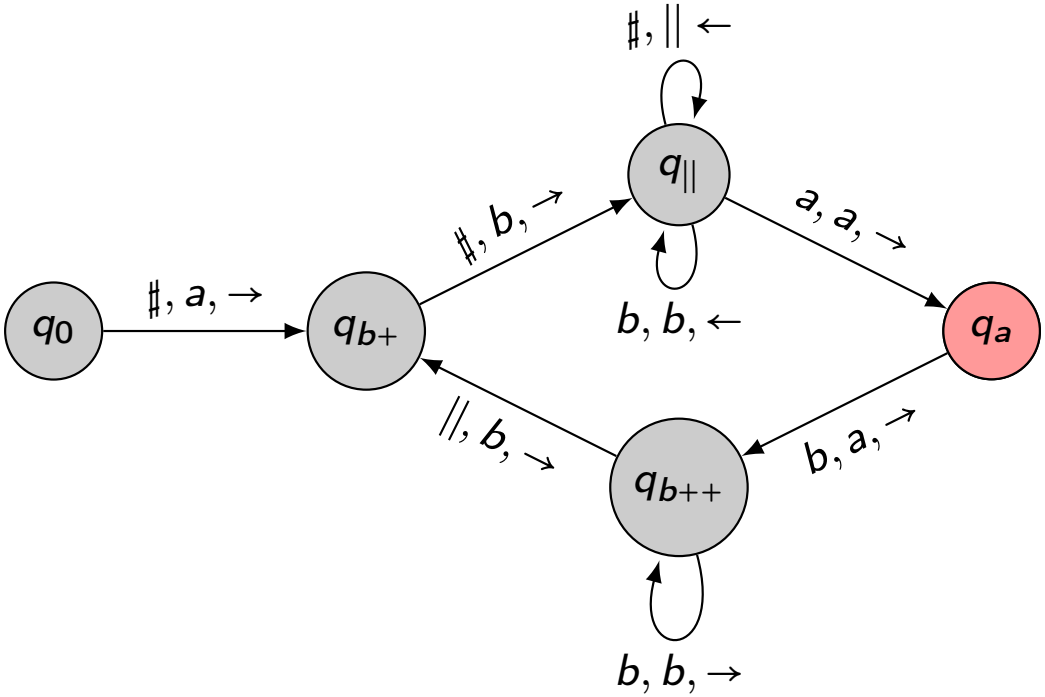
...	♯	**a**	**a**	$(q_{\|\|}, b)$	**b**	||	♯	♯	♯	♯	...
...	♯	**a**	**a**	**b**	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	...
...	♯	**a**	**a**	**b**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	...
...	♯	**a**	**a**	**b**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	...
...	♯	**a**	**a**	$(q_{b++}, \|\|)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	(q_{a+}, b)	||	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_{\|\|}, a)$	**b**	||	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing



. . .	♯	**a**	$(q_{\|\|}, a)$	**b**	**b**	||	♯	♯	♯	♯	. . .
. . .	♯	**a**	**a**	$(q_{\|\|}, b)$	**b**	||	♯	♯	♯	♯	. . .
. . .	♯	**a**	**a**	**b**	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	. . .
. . .	♯	**a**	**a**	**b**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	. . .
. . .	♯	**a**	**a**	**b**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	. . .
. . .	♯	**a**	**a**	$(q_{b++}, \|\|)$	♯	♯	♯	♯	♯	♯	. . .
. . .	♯	**a**	(q_{a+}, b)	||	♯	♯	♯	♯	♯	♯	. . .
. . .	♯	$(q_{\|\|}, a)$	**b**	||	♯	♯	♯	♯	♯	♯	. . .
. . .	♯	**a**	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	♯	♯	. . .
. . .	♯	**a**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	. . .
. . .	♯	**a**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	. . .
. . .	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	. . .

Fonctionnement d'une machine de Turing



...	♯	**a**	**a**	(q_{a+}, b)	**b**	‖	♯	♯	♯	♯	...
...	♯	**a**	$(q_{\|\|}, a)$	**b**	**b**	‖	♯	♯	♯	♯	...
...	♯	**a**	**a**	$(q_{\|\|}, b)$	**b**	‖	♯	♯	♯	♯	...
...	♯	**a**	**a**	**b**	$(q_{\|\|}, b)$	‖	♯	♯	♯	♯	...
...	♯	**a**	**a**	**b**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	...
...	♯	**a**	**a**	**b**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	...
...	♯	**a**	**a**	$(q_{b++}, \|\|)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	(q_{a+}, b)	‖	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_{\|\|}, a)$	**b**	‖	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{\|\|}, b)$	‖	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	**b**	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	...
...	♯	**a**	$(q_{b+}, ♯)$	♯	♯	♯	♯	♯	♯	♯	...
...	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	...

Fonctionnement d'une machine de Turing

On considère la machine de Turing $\mathcal{M}_{\text{ex}}$ qui énumère le langage $\{a^n b^n : n \in \mathbb{N}\}$.

	...	...	...	...	...	...	...	...	...	...	
	♯	a	a	a	a	b	(q_{bb}, b)	||	♯	♯	
	♯	a	a	a	a	(q_{bb}, b)	b	||	♯	♯	
	♯	a	a	a	(q_a, b)	b	b	||	♯	♯	
	♯	a	a	$(q_\|, a)$	b	b	b	||	♯	♯	
	♯	a	a	a	$(q_\|, b)$	b	b	||	♯	♯	
	♯	a	a	a	b	$(q_\|, b)$	b	||	♯	♯	
	♯	a	a	a	b	b	$(q_\|, b)$	||	♯	♯	
	♯	a	a	a	b	b	b	$(q_\|, ♯)$	♯	♯	
	♯	a	a	a	b	b	$(q_b, ♯)$	♯	♯	♯	
	♯	a	a	a	b	$(q_{bb}, \|\|)$	♯	♯	♯	♯	
	♯	a	a	a	(q_{bb}, b)	||	♯	♯	♯	♯	
	♯	a	a	(q_a, b)	b	||	♯	♯	♯	♯	
	♯	a	$(q_\|, a)$	b	b	||	♯	♯	♯	♯	
	♯	a	a	$(q_\|, b)$	b	||	♯	♯	♯	♯	
	♯	a	a	b	$(q_\|, b)$	||	♯	♯	♯	♯	
	♯	a	a	b	b	$(q_\|, ♯)$	♯	♯	♯	♯	
	♯	a	a	b	$(q_b, ♯)$	♯	♯	♯	♯	♯	
	♯	a	a	$(q_{bb}, \|\|)$	♯	♯	♯	♯	♯	♯	
	♯	a	(q_a, b)	||	♯	♯	♯	♯	♯	♯	
	♯	$(q_\|, a)$	b	||	♯	♯	♯	♯	♯	♯	
	♯	a	$(q_\|, b)$	||	♯	♯	♯	♯	♯	♯	
	♯	a	b	$(q_\|, ♯)$	♯	♯	♯	♯	♯	♯	
	♯	a	$(q_b, ♯)$	♯	♯	♯	♯	♯	♯	♯	
	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	

Comment coder ce diagramme espace temps à l'aide de règles locales?

Fonctionnement d'une machine de Turing

On considère la machine de Turing $\mathcal{M}_{\text{ex}}$ qui énumère le langage $\{a^n b^n : n \in \mathbb{N}\}$.

	...	...	...	...	...	...	...	...	...	...	
	♯	a	a	a	a	b	(q_{bb}, b)	||	♯	♯	
	♯	a	a	a	a	(q_{bb}, b)	b	||	♯	♯	
	♯	a	a	a	(q_a, b)	b	b	||	♯	♯	
	♯	a	a	$(q_{\|\|}, a)$	b	b	b	||	♯	♯	
	♯	a	a	a	$(q_{\|\|}, b)$	b	b	||	♯	♯	
	♯	a	a	a	b	$(q_{\|\|}, b)$	b	||	♯	♯	
	♯	a	a	a	b	b	$(q_{\|\|}, b)$	||	♯	♯	
	♯	a	a	a	b	b	b	$(q_{\|\|}, ♯)$	♯	♯	
	♯	a	a	a	b	b	$(q_b, ♯)$	♯	♯	♯	
	♯	a	a	a	b	$(q_{bb}, \|\|)$	♯	♯	♯	♯	
	♯	a	a	a	(q_{bb}, b)	||	♯	♯	♯	♯	
	♯	a	a	(q_a, b)	b	||	♯	♯	♯	♯	
	♯	a	$(q_{\|\|}, a)$	b	b	||	♯	♯	♯	♯	
	♯	a	a	$(q_{\|\|}, b)$	b	||	♯	♯	♯	♯	
	♯	a	a	b	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	
	♯	a	a	b	b	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	
	♯	a	a	b	$(q_b, ♯)$	♯	♯	♯	♯	♯	
	♯	a	a	$(q_{bb}, \|\|)$	♯	♯	♯	♯	♯	♯	
	♯	a	(q_a, b)	||	♯	♯	♯	♯	♯	♯	
	♯	$(q_{\|\|}, a)$	b	||	♯	♯	♯	♯	♯	♯	
	♯	a	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	♯	♯	
	♯	a	b	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	
	♯	a	$(q_b, ♯)$	♯	♯	♯	♯	♯	♯	♯	
	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	

Comment coder ce diagramme espace temps à l'aide de règles locales?

Fonctionnement d'une machine de Turing

On considère la machine de Turing $\mathcal{M}_{\text{ex}}$ qui énumère le langage $\{a^n b^n : n \in \mathbb{N}\}$.

	...	...	...	...	...	...	...	...	...	...	
	♯	a	a	a	a	b	(q_{bb}, b)	||	♯	♯	
	♯	a	a	a	a	(q_{bb}, b)	b	||	♯	♯	
	♯	a	a	a	(q_a, b)	b	b	||	♯	♯	
	♯	a	a	$(q_{\|\|}, a)$	b	b	b	||	♯	♯	
	♯	a	a	a	$(q_{\|\|}, b)$	b	b	||	♯	♯	
	♯	a	a	a	b	$(q_{\|\|}, b)$	b	||	♯	♯	
	♯	a	a	a	b	b	$(q_{\|\|}, b)$	||	♯	♯	
	♯	a	a	a	b	b	b	$(q_{\|\|}, ♯)$	♯	♯	
	♯	a	a	a	b	b	$(q_b, ♯)$	♯	♯	♯	
	♯	a	a	a	b	$(q_{bb}, \|\|)$	♯	♯	♯	♯	
	♯	a	a	a	(q_{bb}, b)	||	♯	♯	♯	♯	
	♯	a	a	(q_a, b)	b	||	♯	♯	♯	♯	
	♯	a	$(q_{\|\|}, a)$	b	b	||	♯	♯	♯	♯	
	♯	a	a	$(q_{\|\|}, b)$	b	||	♯	♯	♯	♯	
	♯	a	a	b	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	
	♯	a	a	b	b	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	
	♯	a	a	b	$(q_b, ♯)$	♯	♯	♯	♯	♯	
	♯	a	a	$(q_{bb}, \|\|)$	♯	♯	♯	♯	♯	♯	
	♯	a	(q_a, b)	||	♯	♯	♯	♯	♯	♯	
	♯	$(q_{\|\|}, a)$	b	||	♯	♯	♯	♯	♯	♯	
	♯	a	$(q_{\|\|}, b)$	||	♯	♯	♯	♯	♯	♯	
	♯	a	b	$(q_{\|\|}, ♯)$	♯	♯	♯	♯	♯	♯	
	♯	a	$(q_b, ♯)$	♯	♯	♯	♯	♯	♯	♯	
	♯	$(q_0, ♯)$	♯	♯	♯	♯	♯	♯	♯	♯	

Comment coder ce diagramme espace temps à l'aide de règles locales?

SFT associé à une machine de Turing

Soit $\mathcal{A}_{\mathcal{M}} = \Gamma \cup (Q \times \Gamma)$.
À une machine de Turing $\mathcal{M}$ on associe l'ensemble $P_{\mathcal{M}}$ des motifs 3×2 autorisés:

- si il n'y a pas de tête de lecture dans le voisinage, pour $x, y, z \in \Gamma$ on autorise le motif :

x	y	z
x	y	z

- si la tête de calcul est présente dans le voisinage, on code les règles de transition de la machine, par exemple $\delta(q_1, x) = (q_2, y, \leftarrow)$ sera codée par:

v	w	(q_2, z)
v	w	z

w	(q_2, z)	y
w	z	(q_1, x)

(q_2, z)	y	z'
z	(q_1, x)	z'

y	z'	z''
(q_1, x)	z'	z''

- enfin si q_F est un état final, alors la machine arrête son calcul ce que l'on traduit par le motif :

z	(q_F, x)	z'
z	(q_F, x)	z'

On considère le SFT $\mathbf{T}_{\mathcal{M}} = \mathbf{T}\left(\mathcal{A}_{\mathcal{M}}, 2, \mathcal{A}_{\mathcal{M}}^{[0,2]\times[0,1]} \setminus P_{\mathcal{M}}\right)$.
Le SFT $\mathbf{T}_{\mathcal{M}}$ contient tous les diagrammes espace temps produit par $\mathcal{M}$ mais aussi des configurations non bien initialisées.

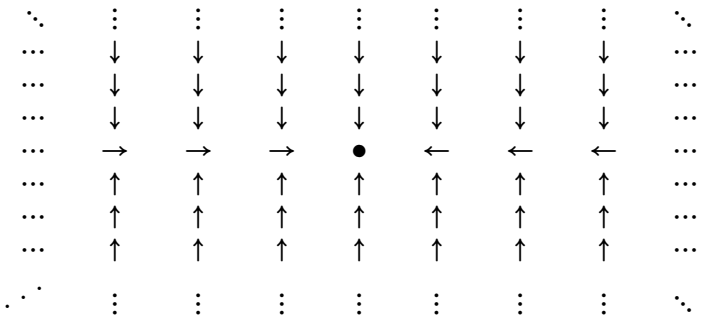
Problème du pavage avec une tuile fixée

Problème de compétion

Soit $\mathbf{T}$ un SFT, est-il possible de compléter un motif en une configuration $x \in \mathbf{T}$?

Soit $\mathcal{A}_\bullet = \{\bullet, \leftarrow, \rightarrow, \uparrow, \downarrow\}$ et on considère le SFT suivant:

$$\mathbf{T}(\mathcal{A}_\bullet, 2, \mathcal{F}_\bullet) = \overline{\mathcal{O}(x)} \text{ où } x =$$



Soit $\mathbf{T} = \mathbf{T}\left(\mathcal{A}_\bullet \times \mathcal{A}_\mathcal{M}, 2, \mathcal{F}_\bullet \times \mathcal{F}_\mathcal{M} \cup \mathcal{F}_{\texttt{Syncro}} \cup \{q_F\}\right)$ où $\mathcal{F}_{\texttt{Syncro}}$ synchronise:

- $\bullet$ avec $(q_0, \sharp)$,
- $\uparrow$, $\rightarrow$ et $\leftarrow$ avec $\sharp$

$(\bullet, (q_0, \sharp))$ se complète en une configuration de $\mathbf{T} \Longleftrightarrow \mathcal{M}$ ne s'arête pas

Théorème *(Wang 1961)*

Le problème de complétion est indécidable.

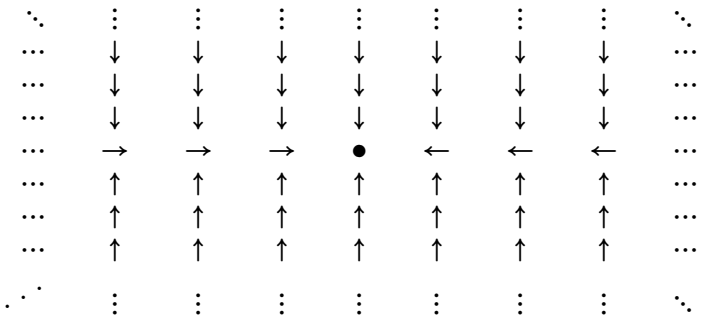
Problème du pavage avec une tuile fixée

Problème de compétion

Soit $\mathbf{T}$ un SFT, est-il possible de compléter un motif en une configuration $x \in \mathbf{T}$?

Soit $\mathcal{A}_\bullet = \{\bullet, \leftarrow, \rightarrow, \uparrow, \downarrow\}$ et on considère le SFT suivant:

$$\mathbf{T}(\mathcal{A}_\bullet, 2, \mathcal{F}_\bullet) = \overline{\mathcal{O}(x)} \text{ où } x =$$



Soit $\mathbf{T} = \mathbf{T}\left(\mathcal{A}_\bullet \times \mathcal{A}_\mathcal{M}, 2, \mathcal{F}_\bullet \times \mathcal{F}_\mathcal{M} \cup \mathcal{F}_{\texttt{Syncro}} \cup \{q_F\}\right)$ où $\mathcal{F}_{\texttt{Syncro}}$ synchronise:

- $\bullet$ avec $(q_0, \sharp)$,
- $\uparrow$, $\rightarrow$ et $\leftarrow$ avec $\sharp$

$(\bullet, (q_0, \sharp))$ se complète en une configuration de $\mathbf{T} \iff \mathcal{M}$ ne s'arête pas

Théorème *(Wang 1961)*

Le problème de complétion est indécidable.

Aucun lien avec le problème du domino: Par compacité il n'existe pas de sous-shift tel qu'une tuile apparaisse une unique fois dans chaque configuration.

Problème du pavage périodique

Problème du pavage périodique

Est-ce qu'un SFT admet une orbite périodique?

Théorème *(Gurevich et Koryakov (1972)*

Le problème du pavage périodique est indécidable.

Problème du pavage périodique

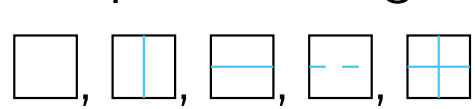
Problème du pavage périodique

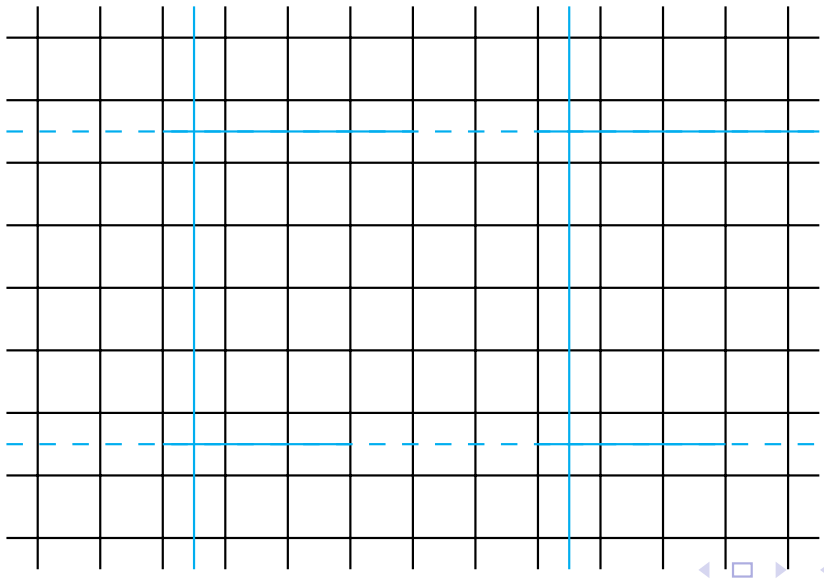
Est-ce qu'un SFT admet une orbite périodique?

Théorème *(Gurevich et Koryakov (1972)*

Le problème du pavage périodique est indécidable.

Soit $\mathbf{T}_{\texttt{Apé}} = \mathbf{T}(\mathcal{A}_{\texttt{Apé}}, 2, \mathcal{F}_{\texttt{Apé}}) \neq \varnothing$ un SFT qui n'a pas de configuration périodique.

Soit $\mathbf{T}_{\texttt{Line}} = \mathbf{T}(\mathcal{A}_{\texttt{Line}}, 2, \mathcal{F}_{\texttt{Line}})$ où $\mathcal{A}_{\texttt{Line}} =$  le SFT qui produit des configurations du type:



Problème du pavage périodique

Problème du pavage périodique

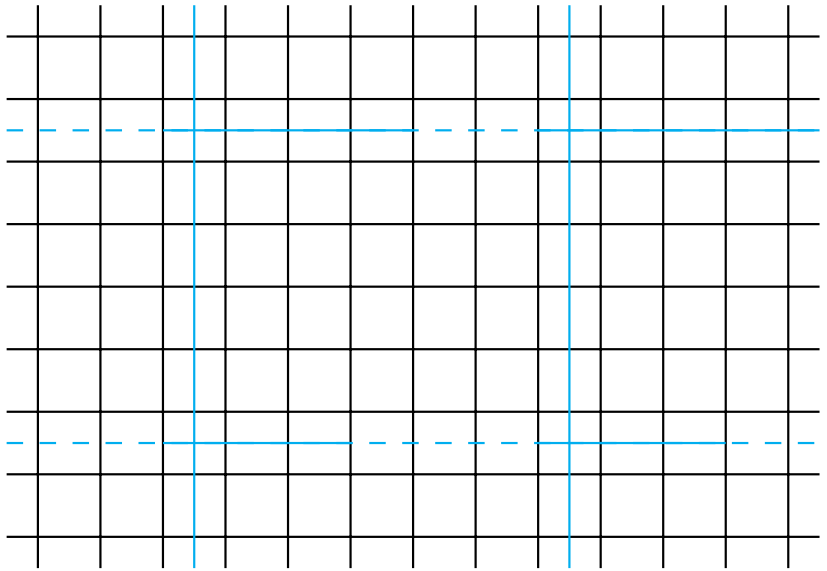
Est-ce qu'un SFT admet une orbite périodique?

Théorème *(Gurevich et Koryakov (1972)*

Le problème du pavage périodique est indécidable.

Soit $\mathcal{M}$ une machine de Turing et on considère le SFT
$\mathbf{T}_{\texttt{Final}} = \mathbf{T}(\mathcal{A}_{\texttt{Apé}} \times \mathcal{A}_{\texttt{Line}} \times \mathcal{A}_{\mathcal{M}}, 2, \mathcal{F}_{\texttt{Line}} \cup \widetilde{\mathcal{F}_{\texttt{Apé}}} \cup \widetilde{\mathcal{F}_{\mathcal{M}}} \cup \mathcal{F}_{\texttt{Syncro}})$ où

- $\widetilde{\mathcal{F}_{\texttt{Apé}}}$ vérifie que $\mathcal{F}_{\texttt{Apé}}$ est vérifié s'il n'y a pas de ligne sur la couche $\mathcal{A}_{\texttt{Line}}$,
- $\widetilde{\mathcal{F}_{\mathcal{M}}}$ vérifie que $\mathcal{F}_{\mathcal{M}}$ est vérifiée s'il n'y a pas de ligne sur la couche $\mathcal{A}_{\texttt{Line}}$,
- $\mathcal{F}_{\texttt{Syncro}}$ vérifie qu'il y a l'état $(q_0, \sharp)$ sur le coin en bas à gauche, $\sharp$ sur toute la ligne du bas et que q_F apparait lors de la transition  .



$\mathbf{T}_{\texttt{Final}}$ admet une configuration périodique $\iff$ $\mathcal{M}$ s'arrête

Retour au problème du domino

Historique du problème du domino

Théorème

Le problème du domino, c'est à dire de se demander "si étant d onné un ensemble de motif $\mathcal{F}$ on a $\mathbf{T}(\mathcal{A}, 2, \mathcal{F}) \neq \varnothing$", est indécidable.

- *Wang 1961:* Wang conjecture que tout SFT admet une configuration périodique et donc que le problème du domino est décidable
- *Berger 1966:* preuve du théorème à partir d'un SFT apériodique réalisé à partir de 20 426 tuiles de Wang
- *Robinson 1971* preuve du théorème à partir d'un SFT apériodique réalisé à partir de 56 tuiles de Wang

Historique du problème du domino

Théorème

Le problème du domino, c'est à dire de se demander "si étant d onné un ensemble de motif $\mathcal{F}$ on a $\mathbf{T}(\mathcal{A}, 2, \mathcal{F}) \neq \varnothing$", est indécidable.

- *Wang 1961:* Wang conjecture que tout SFT admet une configuration périodique et donc que le problème du domino est décidable
- *Berger 1966:* preuve du théorème à partir d'un SFT apériodique réalisé à partir de 20 426 tuiles de Wang
- *Robinson 1971* preuve du théorème à partir d'un SFT apériodique réalisé à partir de 56 tuiles de Wang

Idée de la preuve:

On cherche à construire un sous-shift $\mathbf{T}_{\texttt{Calcul}(\mathcal{M})}$ avec les propriétés suivantes :

- $\mathbf{T}_{\texttt{Calcul}(\mathcal{M})}$ est sofique;
- toute configuration de $\mathbf{T}_{\texttt{Calcul}(\mathcal{M})}$ contient des espaces de calcul de taille arbitrairement grande, en espace et en temps.

Un tel sous-shift assure de voir apparaître les diagrammes espace-temps de la machine $\mathcal{M}$. Dans le relèvement de $\mathbf{T}_{\texttt{Calcul}(\mathcal{M})}$, il suffit d'interdire les configuration contenant l'état q_F de $\mathcal{M}$.

Historique du problème du domino

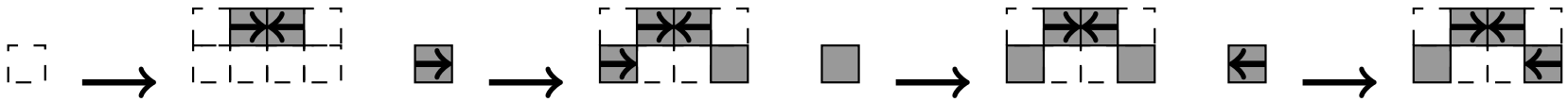
Théorème

Le problème du domino, c'est à dire de se demander "si étant d onné un ensemble de motif $\mathcal{F}$ on a $\mathbf{T}(\mathcal{A}, 2, \mathcal{F}) \neq \varnothing$", est indécidable.

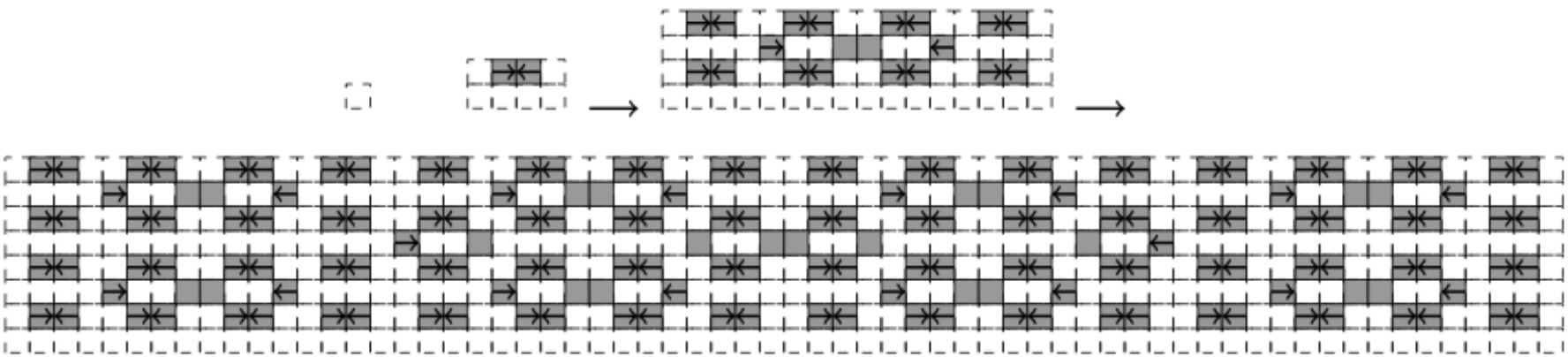
- *Wang 1961:* Wang conjecture que tout SFT admet une configuration périodique et donc que le problème du domino est décidable
- *Berger 1966:* preuve du théorème à partir d'un SFT apériodique réalisé à partir de 20 426 tuiles de Wang
- *Robinson 1971* preuve du théorème à partir d'un SFT apériodique réalisé à partir de 56 tuiles de Wang
- *Kari 2007:* Construction basée sur le jeu à 13 tuiles mais il réduit au problème de l'immortalité des fonction affine par morceaux due à *Hooper (1966)*.

La grille permettant de coder le calcul

On considère l'alphabet $\mathcal{G}_1 = \{\square, \blacksquare, \boxplus, \boxminus\}$ et la substitution $s_{\texttt{Grid}}$:

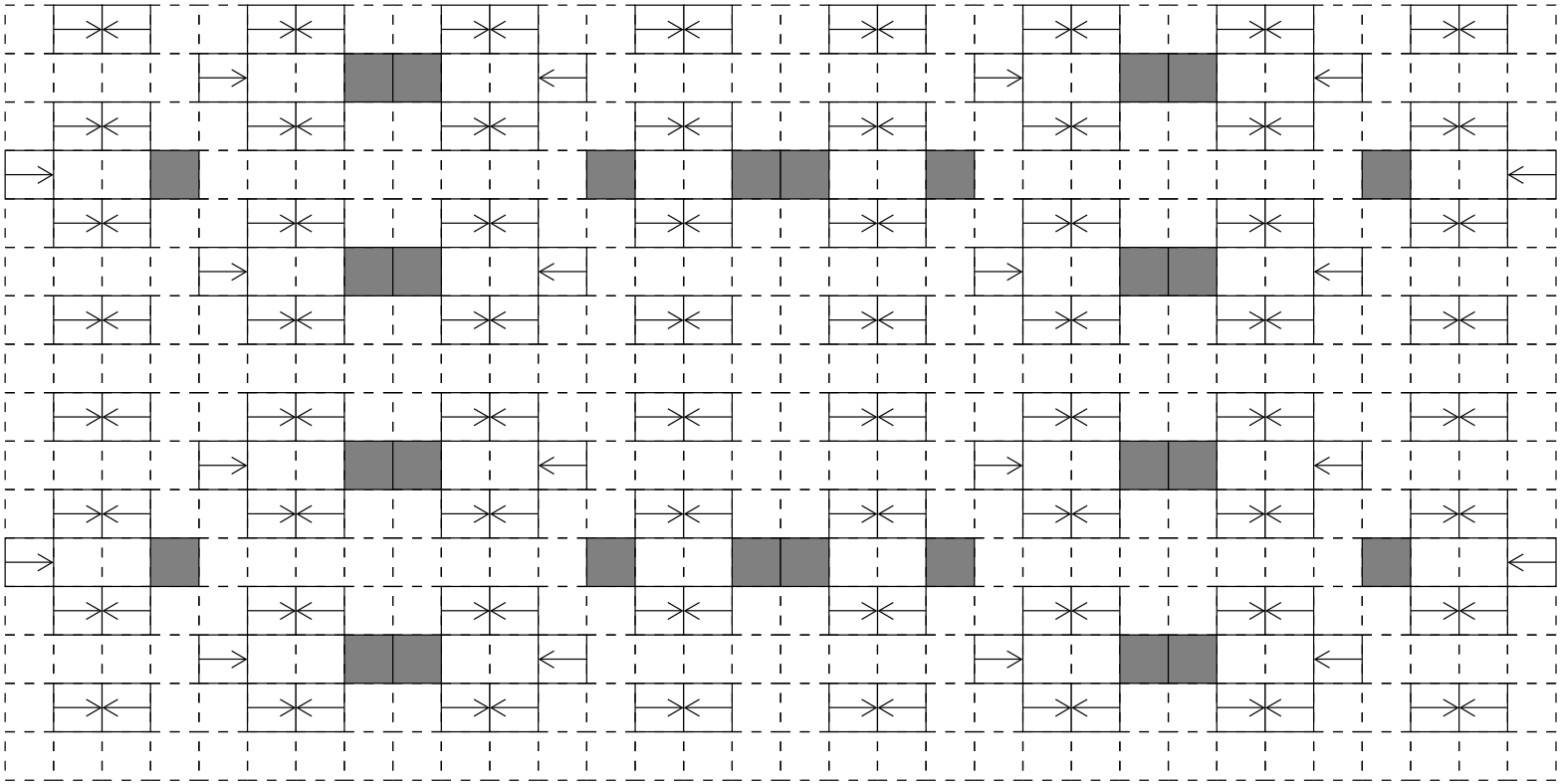


Ce qui donne après quatre itérations de $s_{\texttt{Grid}}$:



Par le théorème de *Mozes*, $\mathbf{T}_{\texttt{Grid}}$ est sofique.

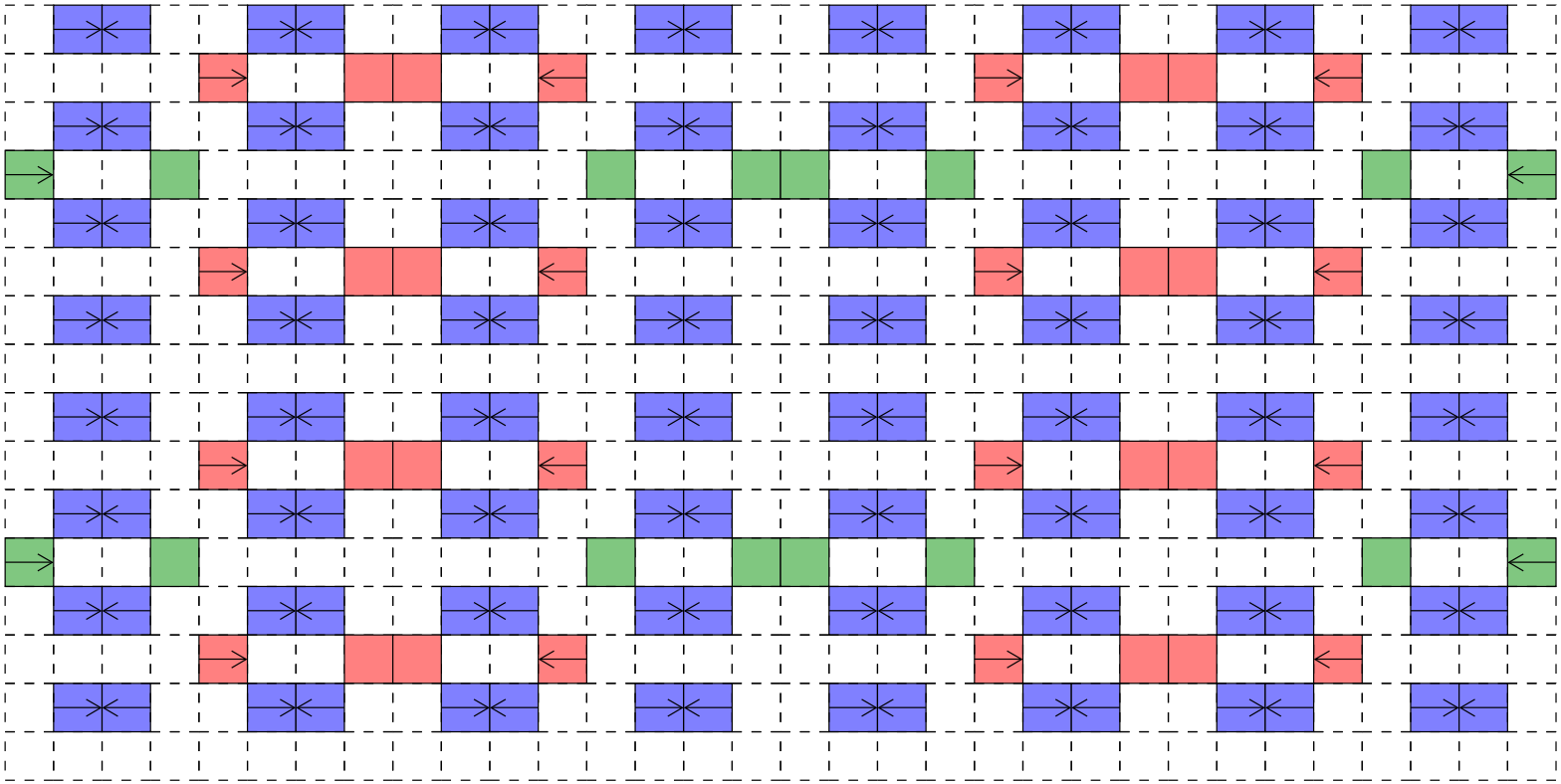
Description des zones de calcul



□ : tuile de communication

 ,  , ■ : tuiles de calcul

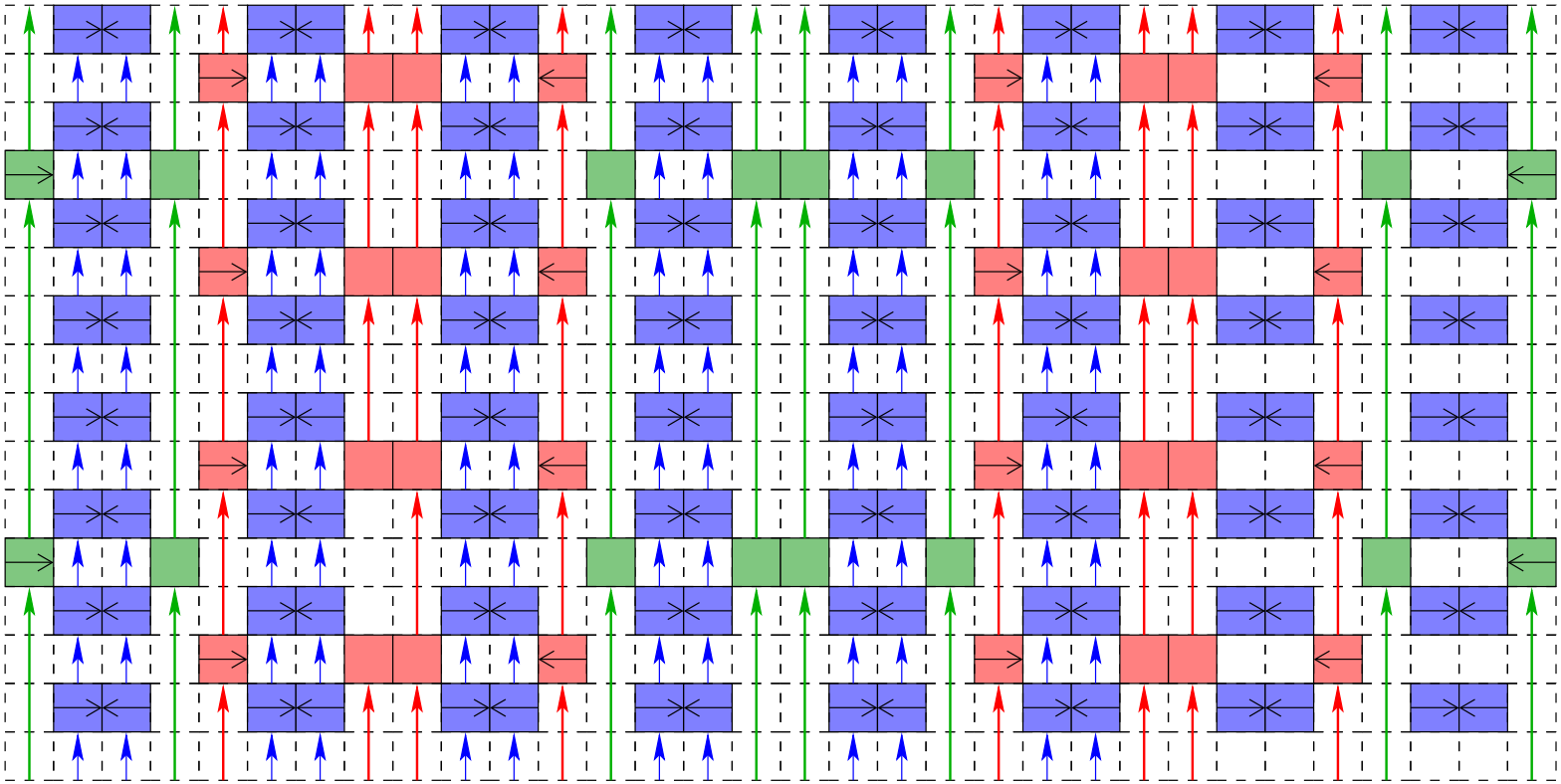
Description des zones de calcul



□ : tuile de communication

 ,  , ■ : tuiles de calcul

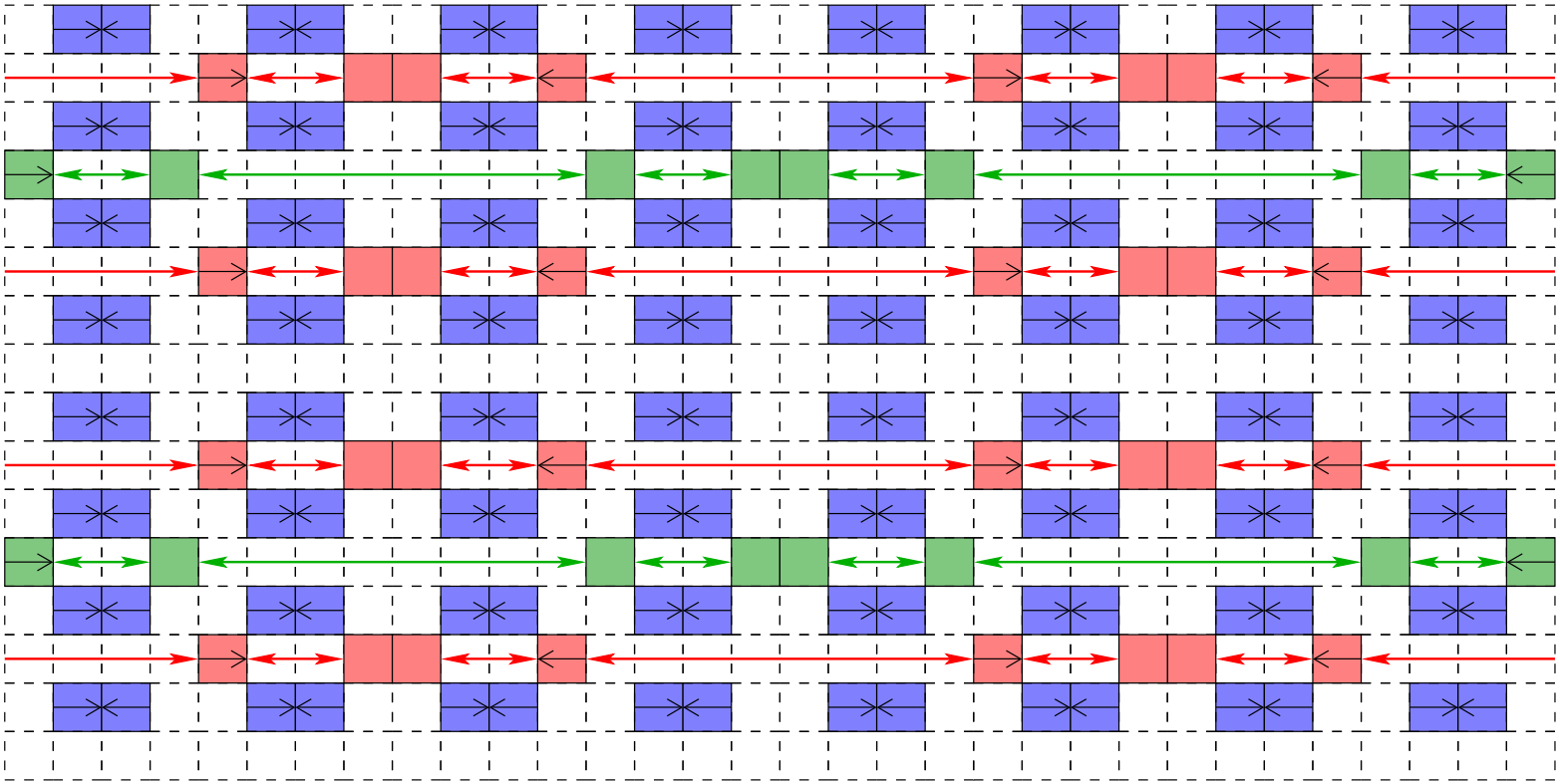
Description des zones de calcul



□ : tuile de communication

⊟, ⊟, ■ : tuiles de calcul

Description des zones de calcul

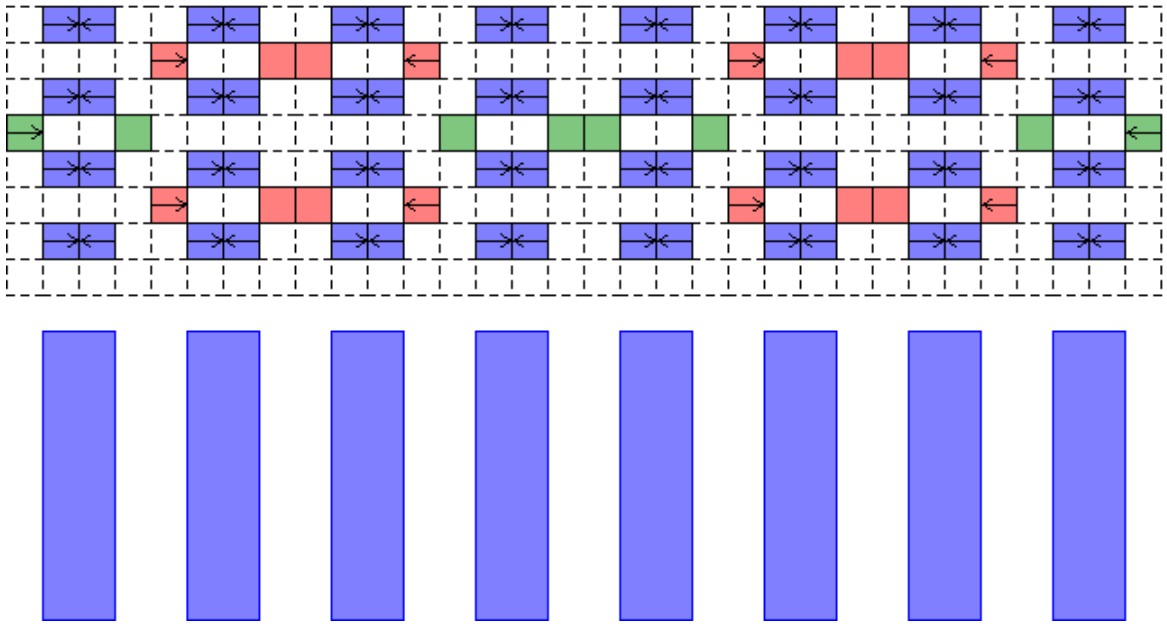


□ : tuile de communication

 ,  , ■ : tuiles de calcul

Description des zones de calcul

On a fractionné la zone de calcul en bandes de différents niveaux (niveau 1, niveau 2, niveau 3):

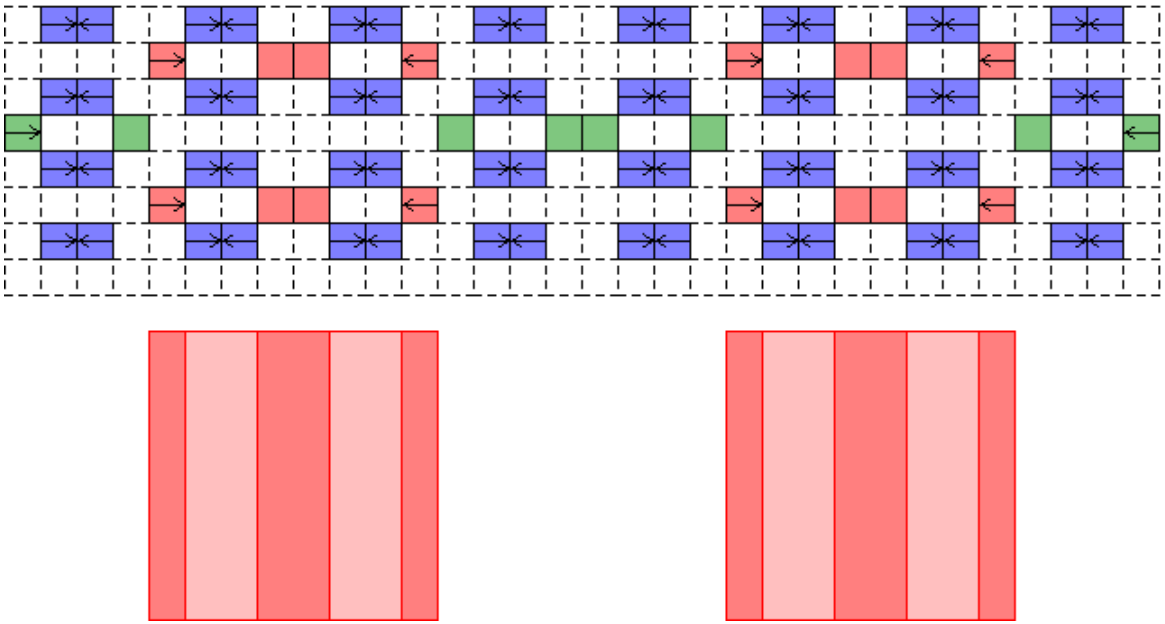


Une bande de calcul de niveau n a les propriétés suivantes:

- son épaisseur est 2^n,
- chaque ligne contient des tuiles de calcul de même niveau,
- il y a des zones de calcul de niveau n toutes les 2^n lignes.

Description des zones de calcul

On a fractionné la zone de calcul en bandes de différents niveaux (niveau 1, niveau 2, niveau 3):

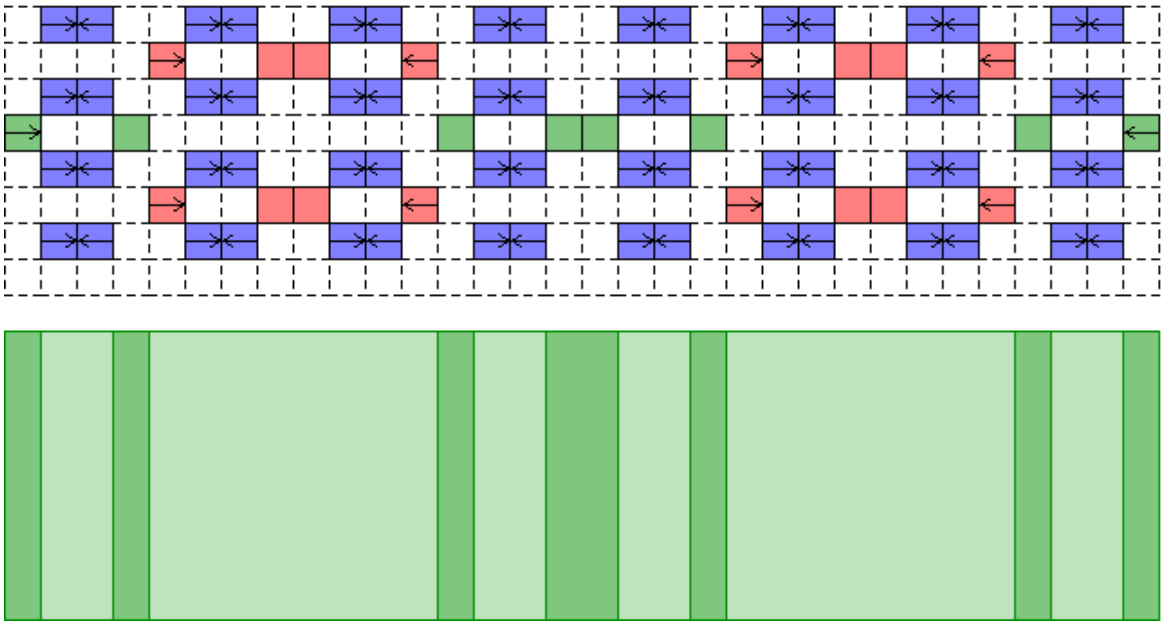


Une bande de calcul de niveau n a les propriétés suivantes:

- son épaisseur est 2^n,
- chaque ligne contient des tuiles de calcul de même niveau,
- il y a des zones de calcul de niveau n toutes les 2^n lignes.

Description des zones de calcul

On a fractionné la zone de calcul en bandes de différents niveaux (niveau 1, niveau 2, niveau 3):

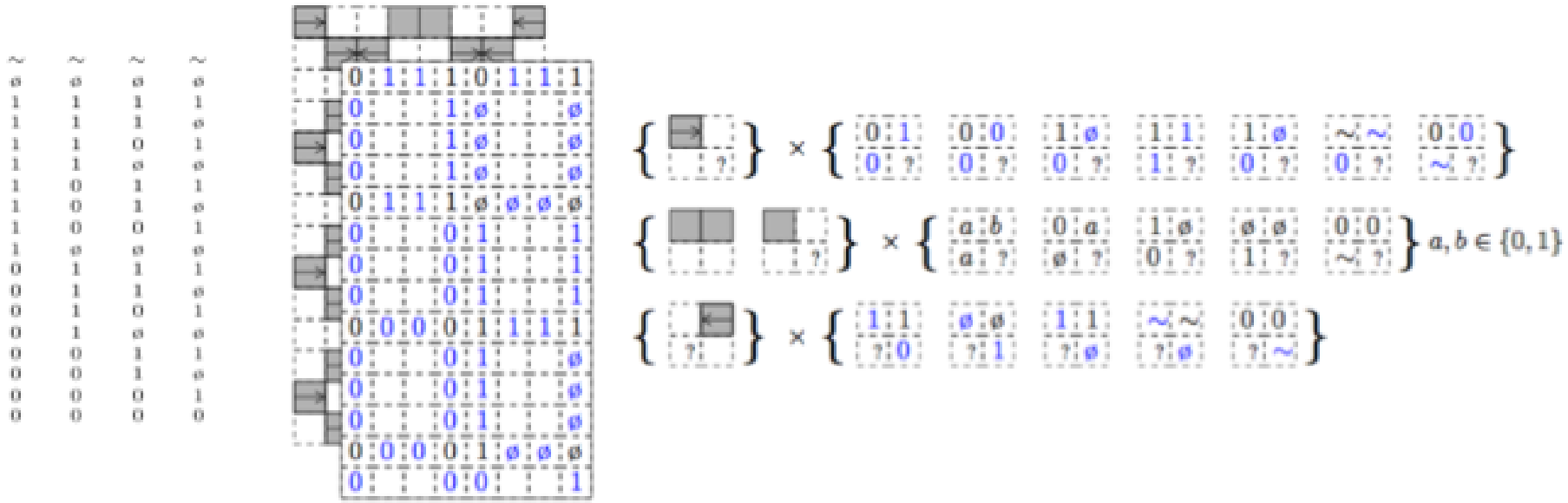


Une bande de calcul de niveau n a les propriétés suivantes:

- son épaisseur est 2^n,
- chaque ligne contient des tuiles de calcul de même niveau,
- il y a des zones de calcul de niveau n toutes les 2^n lignes.

Initialisation des calculs: coder une horloge

Soit $\mathcal{C} = \{0, 1, \varnothing, \sim\}$, on considère l'alphabet $\widetilde{\mathcal{C}} = \mathcal{C} \cup$
Pour initialiser le calcul, on code une horloge par des règles finies:



The layer 2 can be defined as:

$$\mathbf{T}_{\texttt{Clock}} = \mathbf{FT}_{\mathbf{Count} \cup \mathbf{Consist} \cup \mathbf{Synchro}} \left(\mathbf{Prod} \left(\mathbf{T}_{\texttt{Grid}}, \mathcal{C}^{\mathbb{Z}^2} \right) \right)$$

Pour une bande de niveau n, ceci permet d'initialiser le calcul tout les 2^{2^n}.

Comment simuler une machine de Turing avec $\mathbf{T}_{\mathtt{Calcul}(\mathcal{M})}$

On rajoute des règles de type fini à $\mathbf{T}_{\mathtt{Clock}} \times \mathcal{A}^3_{\mathcal{M}}$ pour simuler les calcule de $\mathcal{M}$ sur les différentes bandes:

- conditions **Init** : Quand l'horloge est à 0, on note le mot vide sur tout la zone de calcul excepté au début noté par le symbole $\boxed{\rightarrow}$ qui est dans l'état q_0;

Comment simuler une machine de Turing avec $\mathbf{T}_{\mathtt{Calcul}(\mathcal{M})}$

On rajoute des règles de type fini à $\mathbf{T}_{\mathtt{Clock}} \times \mathcal{A}_{\mathcal{M}}^{3}$ pour simuler les calcule de $\mathcal{M}$ sur les différentes bandes:

- conditions **Init** : Quand l'horloge est à 0, on note le mot vide sur tout la zone de calcul excepté au début noté par le symbole $\boxplus$ qui est dans l'état q_0;
- conditions **Comp** :Quand l'horloge n'est plus à 0, on utilise les règles de $\mathbf{T}_{\mathcal{M}}$;

Comment simuler une machine de Turing avec $\mathbf{T}_{\mathtt{Calcul}(\mathcal{M})}$

On rajoute des règles de type fini à $\mathbf{T}_{\mathtt{Clock}} \times \mathcal{A}^3_{\mathcal{M}}$ pour simuler les calcule de $\mathcal{M}$ sur les différentes bandes:

- conditions **Init** : Quand l'horloge est à 0, on note le mot vide sur tout la zone de calcul excepté au début noté par le symbole $\boxplus$ qui est dans l'état q_0;
- conditions **Comp** :Quand l'horloge n'est plus à 0, on utilise les règles de $\mathbf{T}_{\mathcal{M}}$;
- conditions **Transfer** : dans les zone de communication, les états de la machine de Turing sont transférés horizontalement et verticalement;

Comment simuler une machine de Turing avec $\mathbf{T}_{\mathtt{Calcul}(\mathcal{M})}$

On rajoute des règles de type fini à $\mathbf{T}_{\mathtt{Clock}} \times \mathcal{A}^3_{\mathcal{M}}$ pour simuler les calcule de $\mathcal{M}$ sur les différentes bandes:

- conditions **Init** : Quand l'horloge est à 0, on note le mot vide sur tout la zone de calcul excepté au début noté par le symbole $\boxplus$ qui est dans l'état q_0;
- conditions **Comp** :Quand l'horloge n'est plus à 0, on utilise les règles de $\mathbf{T}_{\mathcal{M}}$;
- conditions **Transfer** : dans les zone de communication, les états de la machine de Turing sont transférés horizontalement et verticalement;
- conditions **Bound** : Si le calcul sort en dehors des zones de calcul, on supprime la tête de lecture.

Comment simuler une machine de Turing avec $\mathbf{T}_{\mathtt{Calcul}(\mathcal{M})}$

On rajoute des règles de type fini à $\mathbf{T}_{\mathtt{Clock}} \times \mathcal{A}_{\mathcal{M}}^{3}$ pour simuler les calcule de $\mathcal{M}$ sur les différentes bandes:

- conditions **Init** : Quand l'horloge est à 0, on note le mot vide sur tout la zone de calcul excepté au début noté par le symbole $\boxplus$ qui est dans l'état q_0;
- conditions **Comp** :Quand l'horloge n'est plus à 0, on utilise les règles de $\mathbf{T}_{\mathcal{M}}$;
- conditions **Transfer** : dans les zone de communication, les états de la machine de Turing sont transférés horizontalement et verticalement;
- conditions **Bound** : Si le calcul sort en dehors des zones de calcul, on supprime la tête de lecture.

Pour one machine de Turing $\mathcal{M}$, On considère le SFT:

$$\mathbf{T}_{\mathcal{M}} = \mathbf{FT}_{\mathbf{Work}_{\mathcal{M}}}\left(\mathbf{Prod}\left(\mathbf{T}_{\mathtt{Grid}}, \mathcal{A}_{\mathcal{M}}^{\mathbb{Z}^2}\right)\right)$$

where $\mathbf{Work}_{\mathcal{M}} = \mathbf{Transfer} \cup \mathbf{Init} \cup \mathbf{Comp} \cup \mathbf{Bound}$.

Comment simuler une machine de Turing avec $\mathbf{T}_{\mathtt{Calcul}(\mathcal{M})}$

Motif de $\mathbf{T}_{\mathtt{Calcul}(\mathcal{M})}$: