Peut-on tout calculer ?

Jonathan Laurent

École Normale Supérieure de Paris

18 Décembre , 2014

Quelques problèmes

Donnée : un mot w
Question : w est-il un palindrome ?

Exemple : `radar` est un palindrome mais `paris` n'en est pas un.

Quelques problèmes

Donnée : un mot w
Question : w est-il un palindrome ?

Exemple : `radar` est un palindrome mais `paris` n'en est pas un.

Donnée : un entier n
Question : n est-il premier ?

Quelques problèmes

Donnée : un mot w
Question : w est-il un palindrome ?

Exemple : `radar` est un palindrome mais `paris` n'en est pas un.

Donnée : un entier n
Question : n est-il premier ?

Donnée : un ensemble de villes et une longueur d
Question : y a-t-il un itinéraire de longueur $\leq d$ passant par toutes les villes ?

Quelques problèmes

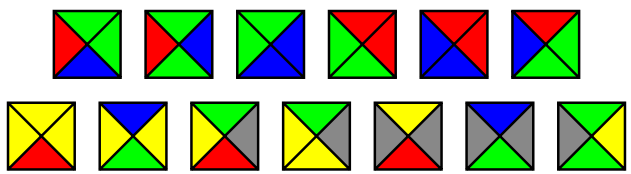
Donnée : un mot w
Question : w est-il un palindrome ?

Exemple : `radar` est un palindrome mais `paris` n'en est pas un.

Donnée : un entier n
Question : n est-il premier ?

Donnée : un ensemble de villes et une longueur d
Question : y a-t-il un itinéraire de longueur $\leq d$ passant par toutes les villes ?

Donnée : un ensemble de tuiles de Wang Γ
Question : peut-on paver le plan à l'aide des tuiles de Γ en utilisant chaque tuile au moins une fois ?



Quelques problèmes

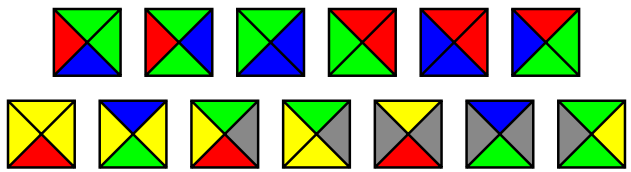
Donnée : un mot w
Question : w est-il un palindrome ?

Exemple : `radar` est un palindrome mais `paris` n'en est pas un.

Donnée : un entier n
Question : n est-il premier ?

Donnée : un ensemble de villes et une longueur d
Question : y a-t-il un itinéraire de longueur $\leq d$ passant par toutes les villes ?

Donnée : un ensemble de tuiles de Wang Γ
Question : peut-on paver le plan à l'aide des tuiles de Γ en utilisant chaque tuile au moins une fois ?



Donnée : un polynôme multivarié à n variables et à coefficients entiers P
Question : l'équation $P(x_1, \cdots, x_n) = 0$ admet-elle des solutions entières ?

Exemple : l'équation $x^2 - 2 \cdot (y^2 + 1)^2 = 0$ n'admet pas de solutions entières.

Formaliser la notion de problème

Alphabets, mots et langages

Définition

- Un **alphabet** est un ensemble fini de symboles.
- Un **mot**, ou **chaîne** sur un alphabet Σ est une suite finie sur Σ. L'ensemble des chaînes sur Σ est noté Σ^*.
- Un **langage** sur l'alphabet Σ est un ensemble de mots sur Σ, c'est à dire une partie de Σ^*.

Remarque : on utilisera souvent l'alphabet binaire $\mathbf{B} = \{0, 1\}$

Exemples :

- 011010 est un mot sur $\mathbf{B}$
- $\{ \epsilon, 01, 1101, 100, 1001010 \}$ est un langage sur $\mathbf{B}$

Formaliser la notion de problème

Problèmes et langages

Considérons le problème *généralisé* suivant :

Donnée : un objet $x \in D$ — D est un ensemble

Question : a-t-on $P(x)$? — P est un prédicat sur l'ensemble D

Formaliser la notion de problème

Problèmes et langages

Considérons le problème *généralisé* suivant :

Donnée : un objet $x \in D$ — D est un ensemble
Question : a-t-on $P(x)$? — P est un prédicat sur l'ensemble D

On se donne un alphabet Σ ainsi qu'une fonction d'encodage

$$enc : D \to \Sigma^*$$

enc doit être **injective**.

On associe donc à notre problème le langage :

$$L = \{\, enc(x) \mid x \in D \text{ et } P(x) \,\}$$

Formaliser la notion de problème

Problèmes et langages

Considérons le problème *généralisé* suivant :

Donnée : un objet $x \in D$ — D est un ensemble
Question : a-t-on $P(x)$? — P est un prédicat sur l'ensemble D

On se donne un alphabet Σ ainsi qu'une fonction d'encodage

$$enc : D \to \Sigma^*$$

enc doit être **injective**.

On associe donc à notre problème le langage :

$$L = \{\, enc(x) \mid x \in D \text{ et } P(x) \,\}$$

Exemples

- $L_{palindromes} = \{\, w \in \mathbf{B}^* \mid w = \overline{w} \,\}$
- $L_{premiers} = \{10\,,\ 11\,,\ 101\,,\ 111\,,\ 10111\,,\ \cdots\}$

Problèmes décidables et algorithmes

Un langage sur l'alphabet Σ *(ie. un problème)* est dit **décidable** s'il existe un algorithme A tel que :

- A prend en entrée des mots de Σ^* et répond `vrai` ou `faux` en **temps fini**.
- A répond `vrai` sur l'entrée x si et seulement si $x \in L$

Il est dit **indécidable** sinon.

Problèmes décidables et algorithmes

Un langage sur l'alphabet Σ *(ie. un problème)* est dit **décidable** s'il existe un algorithme A tel que :

- A prend en entrée des mots de Σ^* et répond `vrai` ou `faux` en **temps fini**.
- A répond `vrai` sur l'entrée x si et seulement si $x \in L$

Il est dit **indécidable** sinon.

Problème : qu'est-ce qu'un algorithme ?

Problèmes décidables et algorithmes

Un langage sur l'alphabet Σ *(ie. un problème)* est dit **décidable** s'il existe un algorithme A tel que :

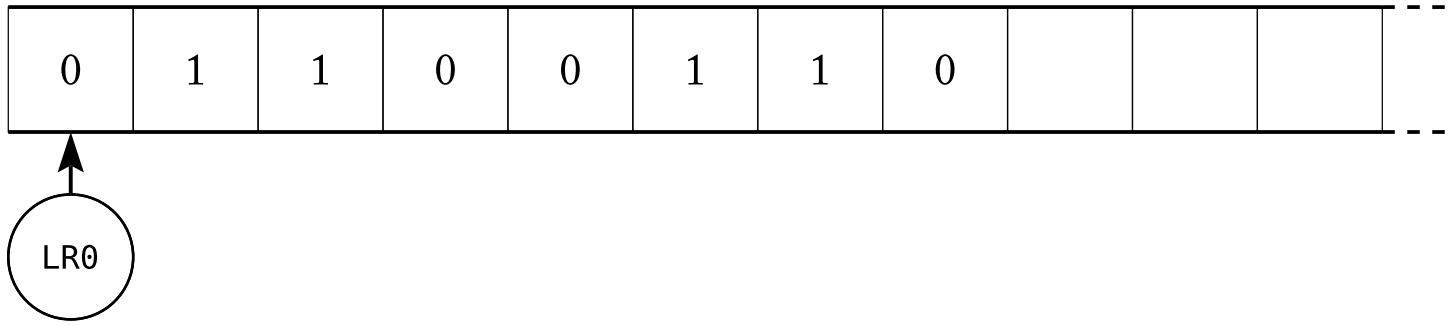
- A prend en entrée des mots de Σ^* et répond `vrai` ou `faux` en **temps fini**.
- A répond `vrai` sur l'entrée x si et seulement si $x \in L$

Il est dit **indécidable** sinon.

Problème : qu'est-ce qu'un algorithme ?

Un programme écrit en python décidant le langage des palindromes :

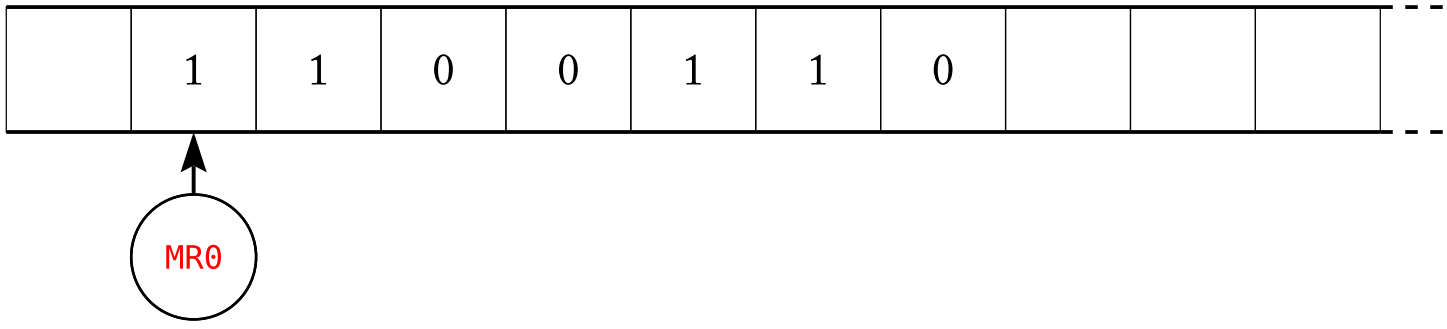
```
def palindrome(mot) :
    if len(mot) <= 1 :
        return True
    else :
        return mot[0] == mot[-1] and palindrome(mot[1:-1])
```



Définition

Une machine de Turing est un sextuplet $(Q, \Sigma, \Gamma, q_0, F, \delta)$ où

- Q est un ensemble fini d'états
- Σ est un alphabet fini dit *alphabet d'entrée*. On prendra $\Sigma = \{0, 1\}$
- $\Gamma = \Sigma \cup \{\sqcup\}$ est l'*alphabet de travail*
- q_0 est l'état initial
- $F \subset Q$ un ensemble d'états finals. On prendra $F = \{q_+, q_-\}$
- $\delta : \Gamma \times (Q \setminus F) \longrightarrow \Gamma \times Q \times \{\leftarrow, \rightarrow\}$ est dite fonction de transition



Définition

Une machine de Turing est un sextuplet $(Q, \Sigma, \Gamma, q_0, F, \delta)$ où

- Q est un ensemble fini d'états
- Σ est un alphabet fini dit *alphabet d'entrée*. On prendra $\Sigma = \{0, 1\}$
- $\Gamma = \Sigma \cup \{\sqcup\}$ est l'*alphabet de travail*
- q_0 est l'état initial
- $F \subset Q$ un ensemble d'états finals. On prendra $F = \{q_+, q_-\}$
- $\delta : \Gamma \times (Q \setminus F) \longrightarrow \Gamma \times Q \times \{\leftarrow, \rightarrow\}$ est dite fonction de transition

Une machine pour reconnaître les palindromes

Il est décidé par la machine de Turing suivante :

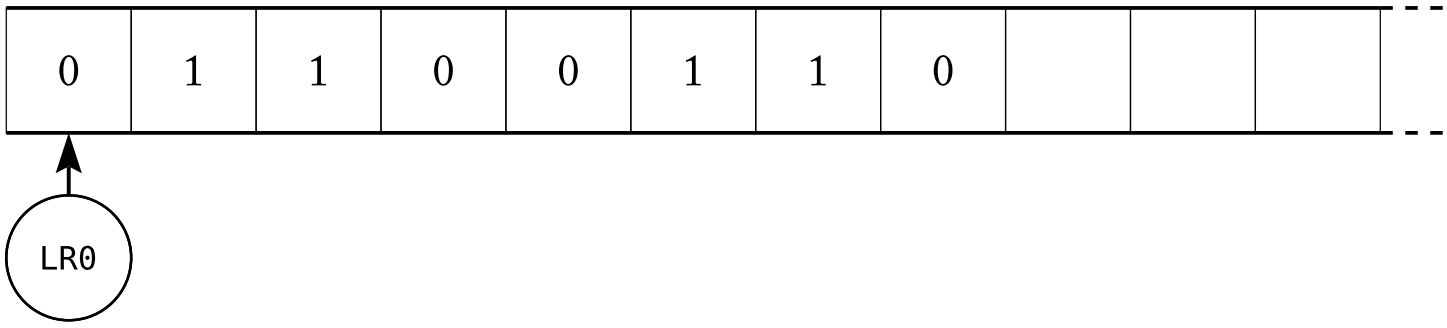
- $Q = \{\texttt{LR0}, \texttt{LL0}, \texttt{MR0}, \texttt{ML0}, \texttt{LR1}, \texttt{LL1}, \texttt{MR1}, \texttt{ML1}, q_+, q_-\}$
- $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \sqcup\}$
- $q_0 = \texttt{LR0}$
- $F = \{q_+, q_-\}$

Table de transitions :

	LR1	LR0	LL1	LL0
0	$(\sqcup, \texttt{MR0}, \rightarrow)$	$(\sqcup, \texttt{MR0}, \rightarrow)$	$(0, \texttt{-}, \leftarrow)$	$(\sqcup, \texttt{ML0}, \leftarrow)$
1	$(\sqcup, \texttt{MR1}, \rightarrow)$	$(\sqcup, \texttt{MR1}, \rightarrow)$	$(\sqcup, \texttt{ML1}, \leftarrow)$	$(1, \texttt{-}, \leftarrow)$
$\sqcup$	$(\sqcup, +, \rightarrow)$	$(\sqcup, +, \rightarrow)$	$(\sqcup, +, \leftarrow)$	$(\sqcup, +, \leftarrow)$

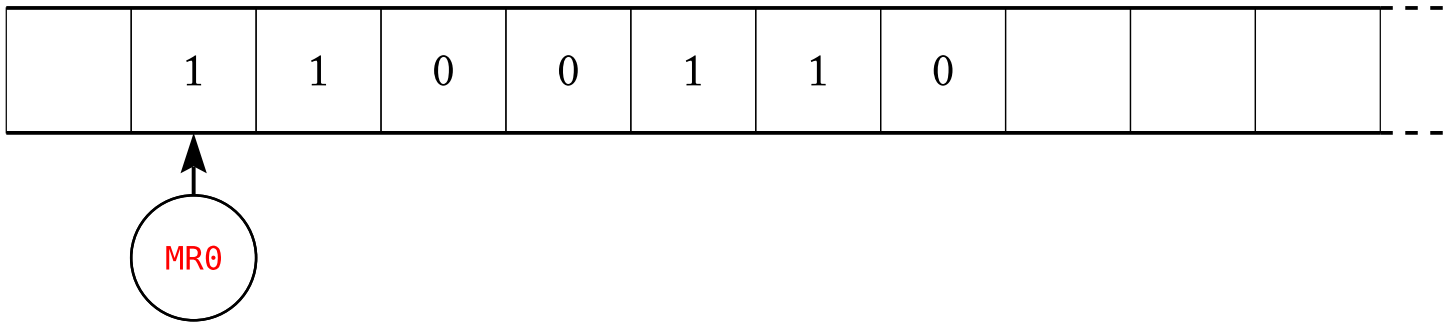
	MR1	MR0	ML1	ML0
0	$(0, \texttt{MR1}, \rightarrow)$	$(0, \texttt{MR0}, \rightarrow)$	$(0, \texttt{ML1}, \leftarrow)$	$(0, \texttt{ML0}, \leftarrow)$
1	$(1, \texttt{MR1}, \rightarrow)$	$(1, \texttt{MR0}, \rightarrow)$	$(1, \texttt{ML1}, \leftarrow)$	$(1, \texttt{ML0}, \leftarrow)$
$\sqcup$	$(\sqcup, \texttt{LL1}, \leftarrow)$	$(\sqcup, \texttt{LL0}, \leftarrow)$	$(\sqcup, \texttt{LR1}, \rightarrow)$	$(\sqcup, \texttt{LR0}, \rightarrow)$

Une machine pour reconnaître les palindromes



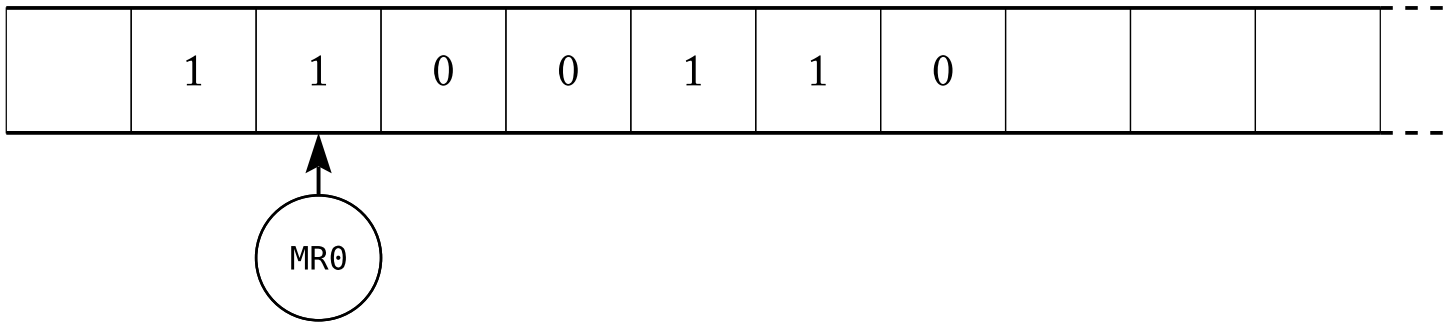
$$\delta(0\,,\,\texttt{LR0}) = (\sqcup\,,\,\texttt{MR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



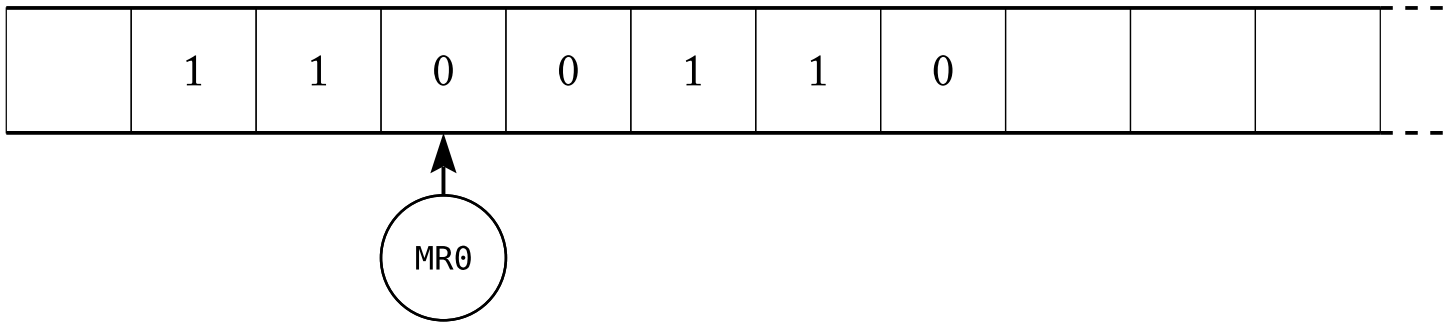
$$\delta(1\,,\,\texttt{MR0}) = (1\,,\,\texttt{MR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



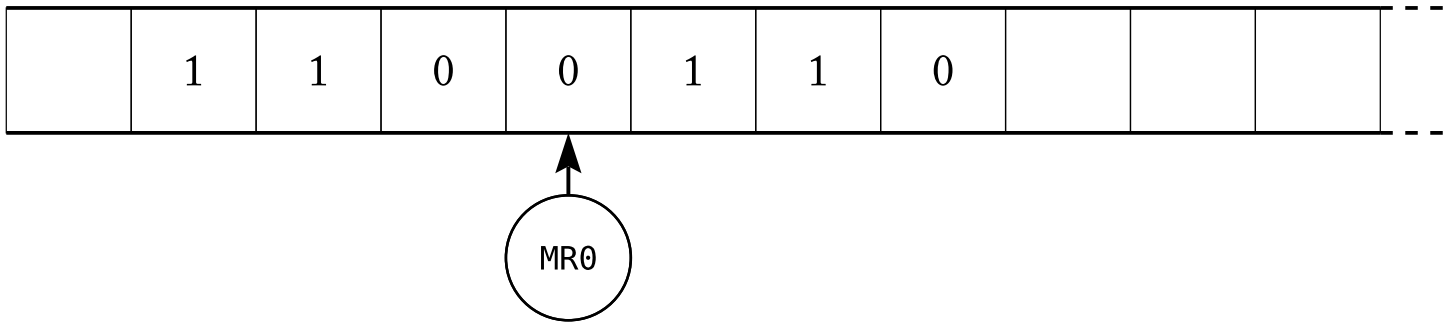
$$\delta(1\,,\,\texttt{MR0}) = (1\,,\,\texttt{MR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



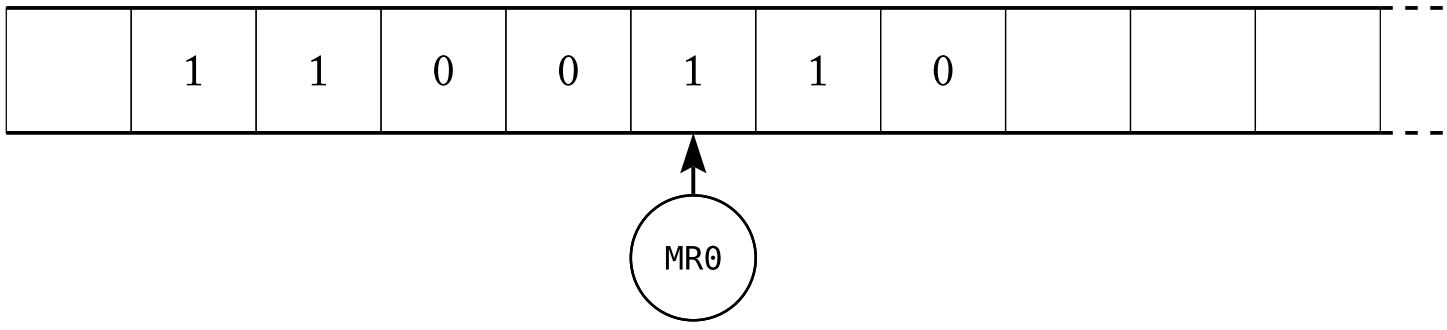
$$\delta(0\,,\,\texttt{MR0}) = (0\,,\,\texttt{MR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



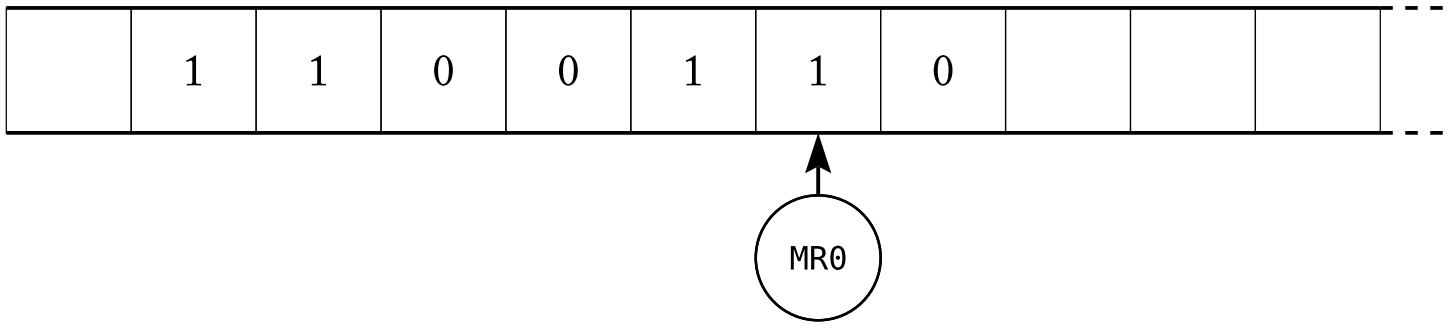
$$\delta(0\,,\,\texttt{MR0}) = (0\,,\,\texttt{MR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



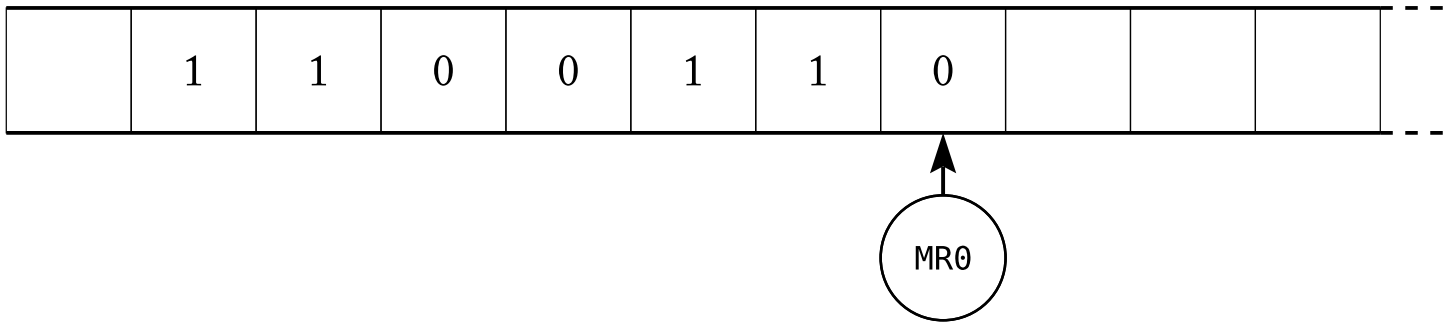
$$\delta(1\,,\,\texttt{MR0}) = (1\,,\,\texttt{MR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



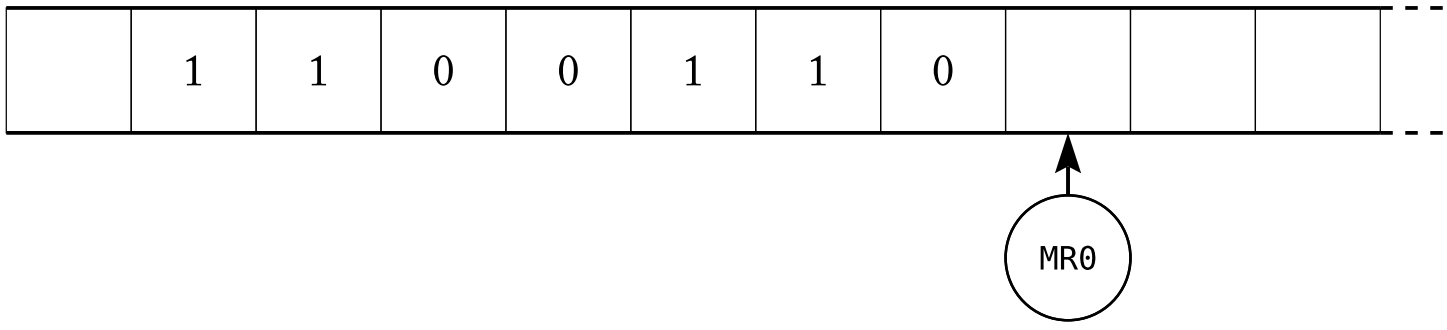
$$\delta(1\,,\,\texttt{MR0}) = (1\,,\,\texttt{MR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



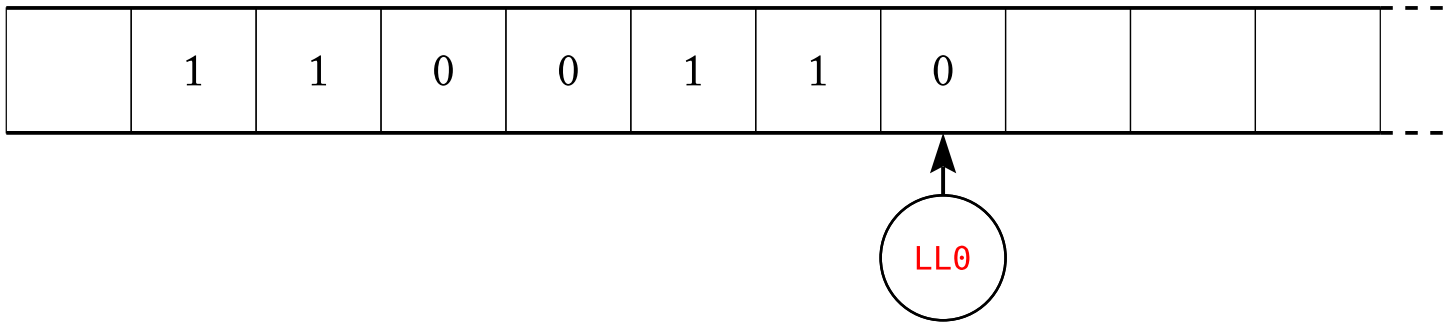
$$\delta(0\,,\,\texttt{MR0}) = (0\,,\,\texttt{MR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



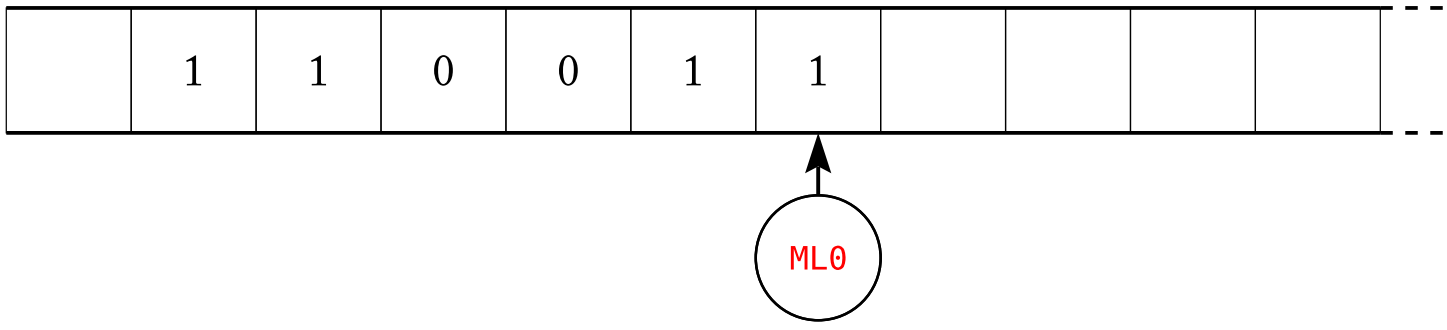
$$\delta(\sqcup\,,\,\texttt{MR0}) = (\sqcup\,,\,\texttt{LL0}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



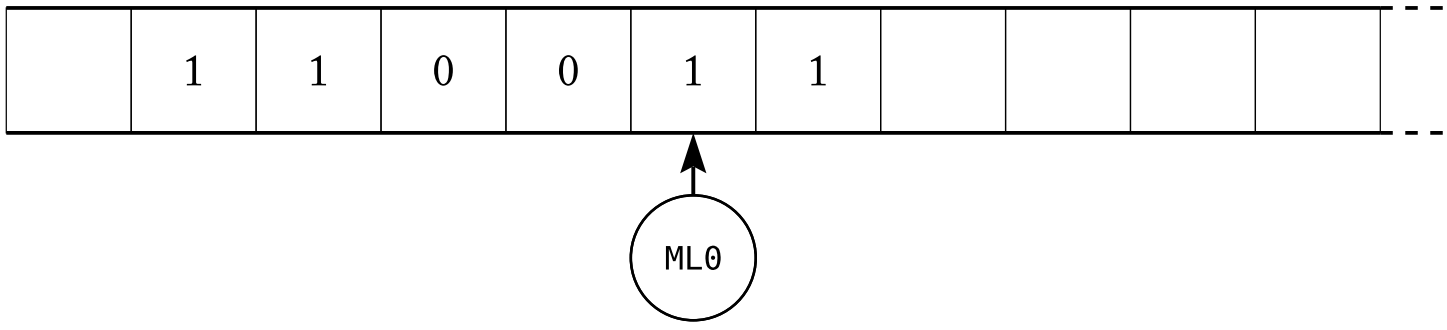
$$\delta(\mathtt{0}\,,\,\mathtt{LL0}) = (\sqcup\,,\,\mathtt{ML0}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



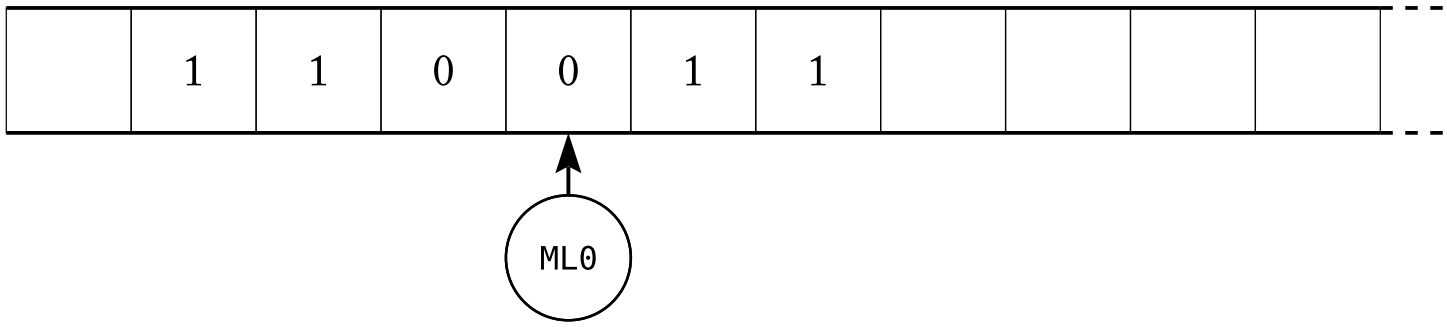
$$\delta(1\,,\,\texttt{ML0}) = (1\,,\,\texttt{ML0}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



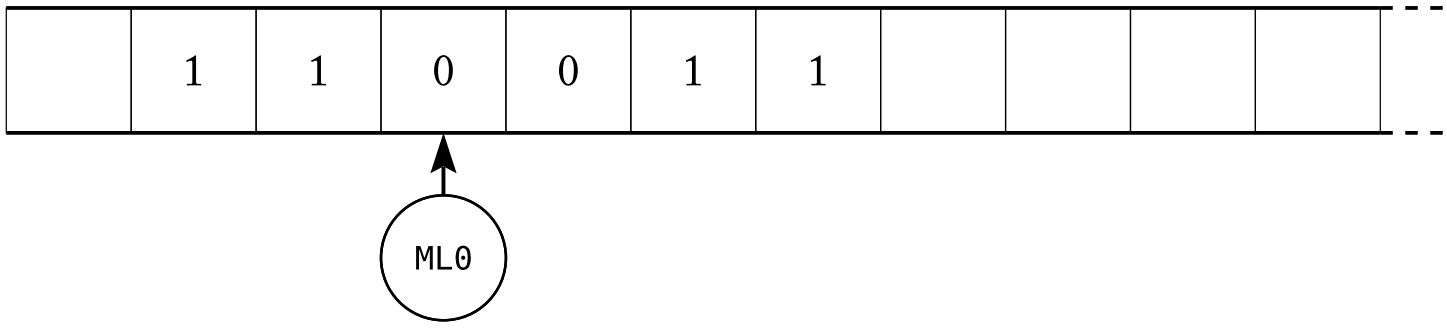
$$\delta(1\,,\,\texttt{ML0}) = (1\,,\,\texttt{ML0}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



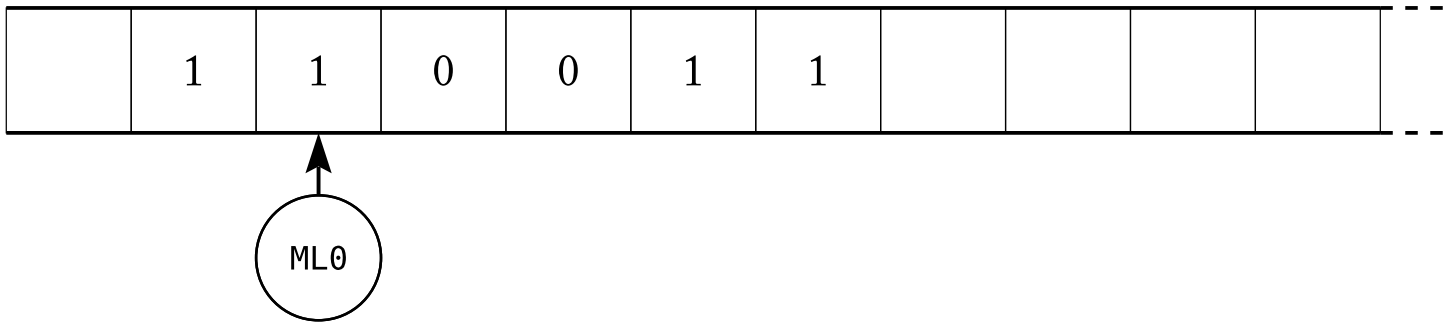
$$\delta(0\,,\,\texttt{ML0}) = (0\,,\,\texttt{ML0}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



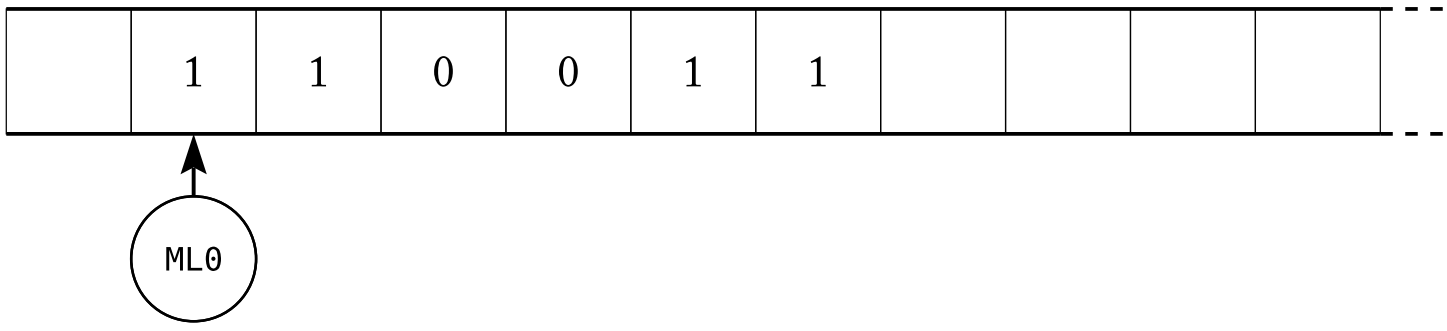
$$\delta(0\,,\,\texttt{ML0}) = (0\,,\,\texttt{ML0}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



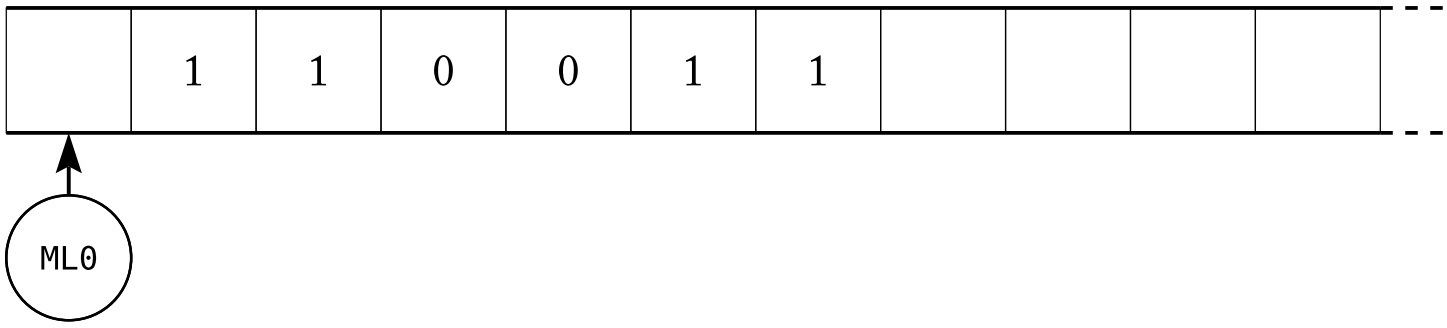
$$\delta(1\,,\,\texttt{ML0}) = (1\,,\,\texttt{ML0}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



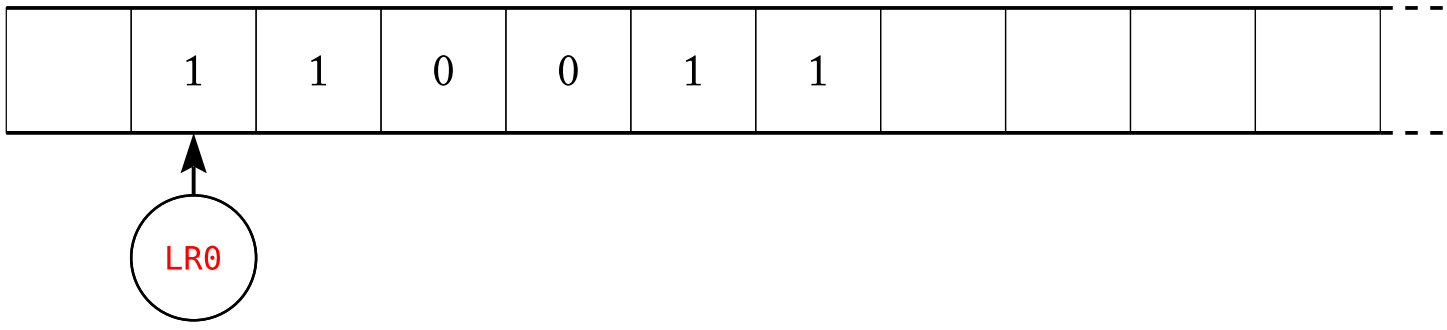
$$\delta(1\,,\,\texttt{ML0}) = (1\,,\,\texttt{ML0}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



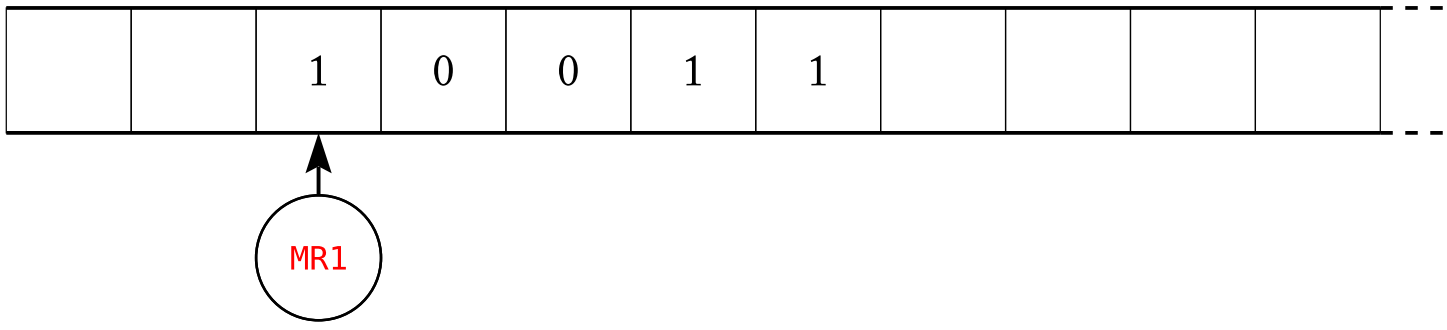
$$\delta(\sqcup\,,\,\texttt{ML0}) = (\sqcup\,,\,\texttt{LR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



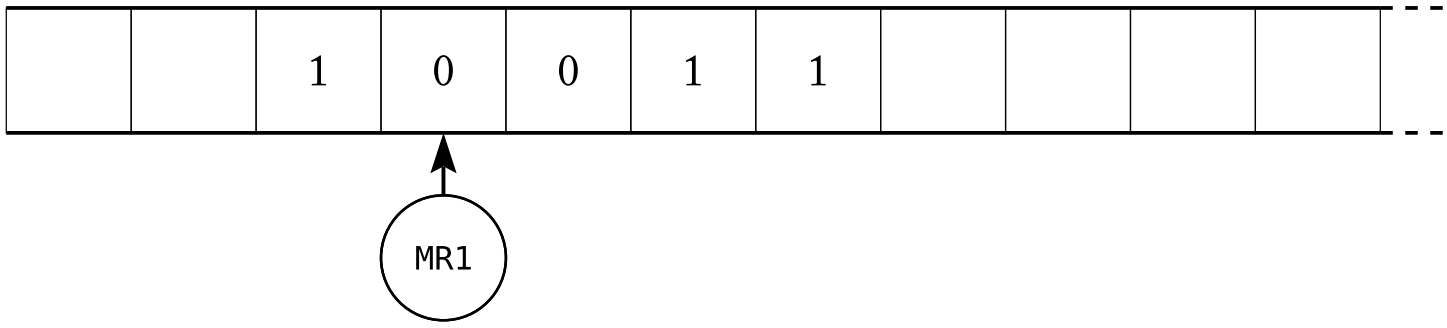
$$\delta(1\,,\,\texttt{LR0}) = (\sqcup\,,\,\texttt{MR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



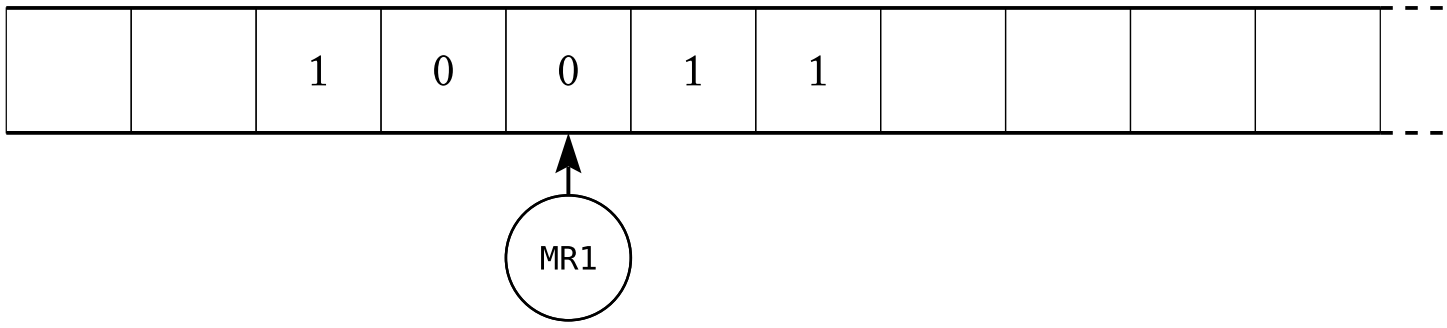
$$\delta(1\,,\,\texttt{MR1}) = (1\,,\,\texttt{MR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



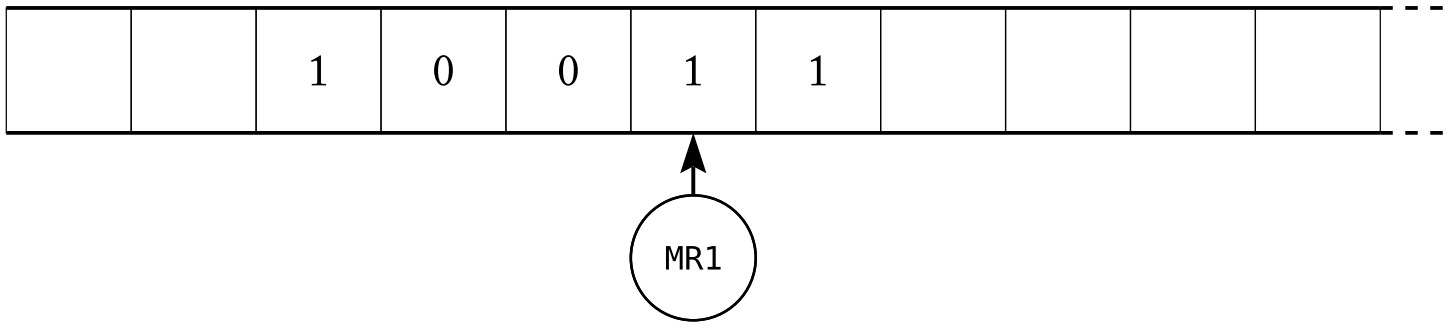
$$\delta(0\,,\,\texttt{MR1}) = (0\,,\,\texttt{MR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



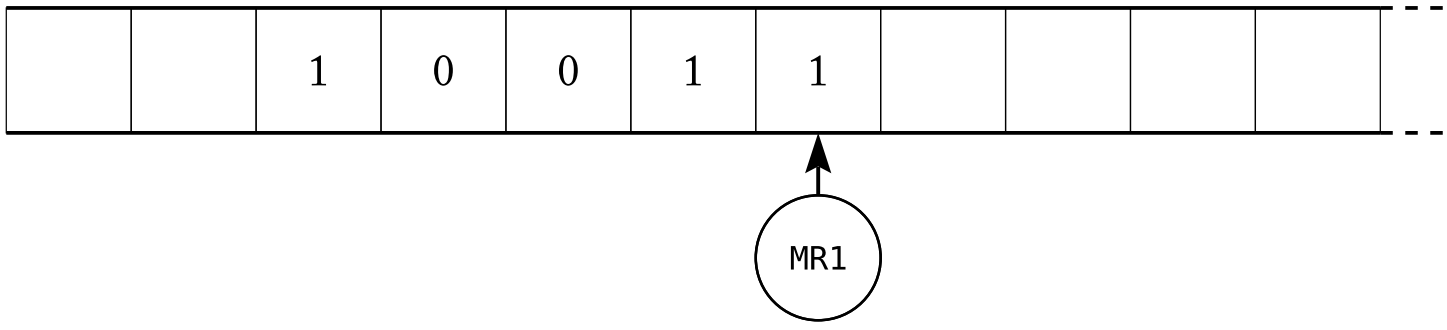
$$\delta(0\,,\,\texttt{MR1}) = (0\,,\,\texttt{MR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



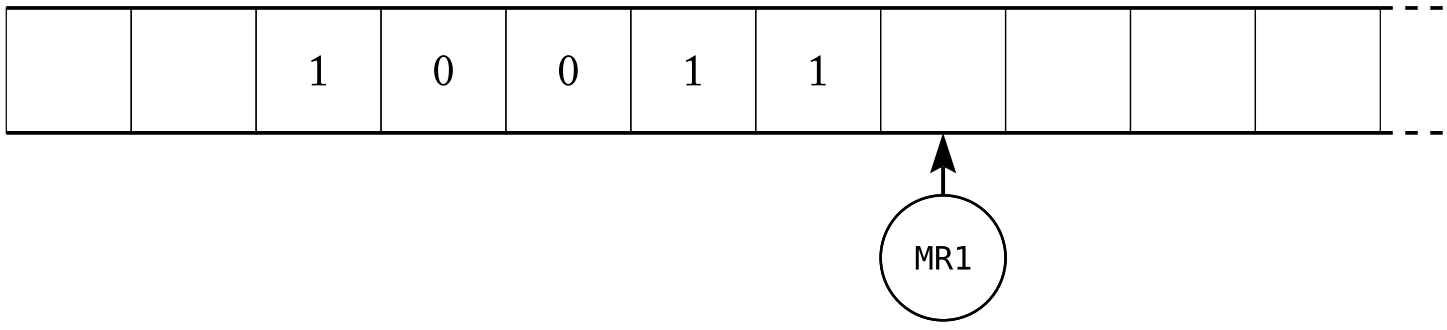
$$\delta(1\,,\,\texttt{MR1}) = (1\,,\,\texttt{MR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



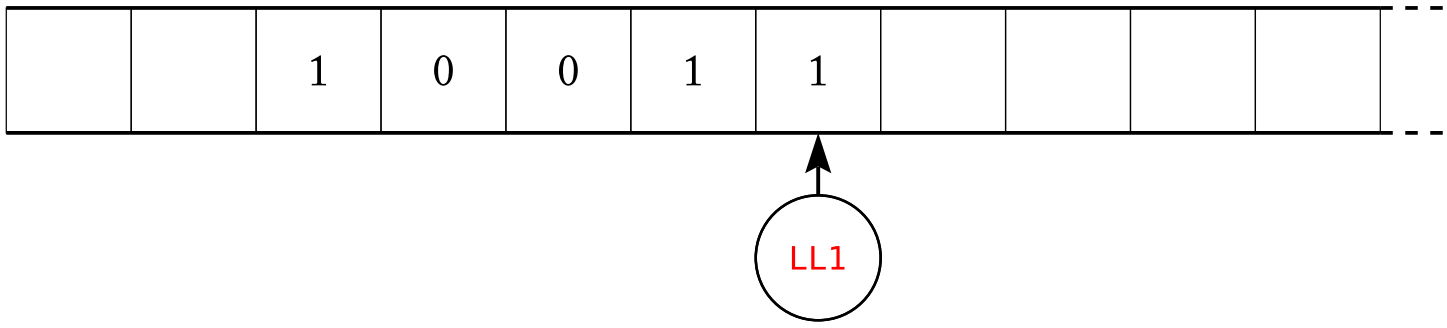
$$\delta(1\,,\,\texttt{MR1}) = (1\,,\,\texttt{MR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



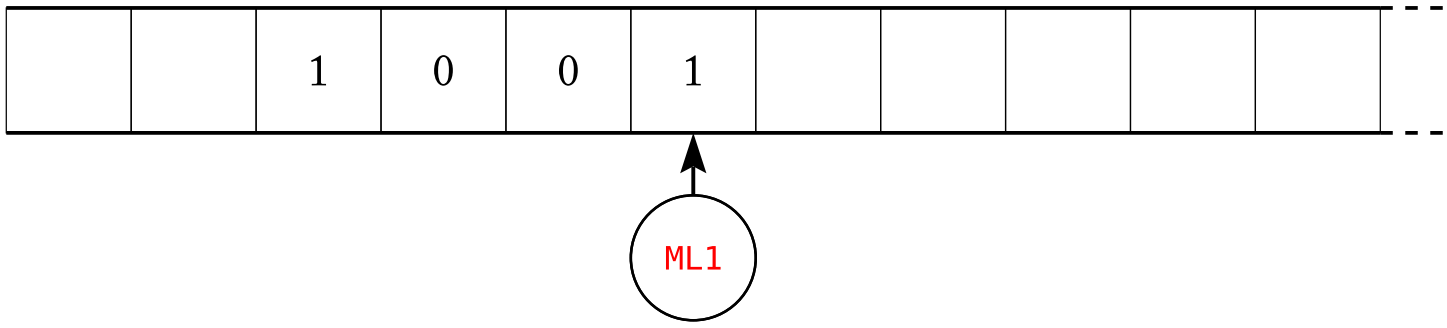
$$\delta(\sqcup\,,\,\texttt{MR1}) = (\sqcup\,,\,\texttt{LL1}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



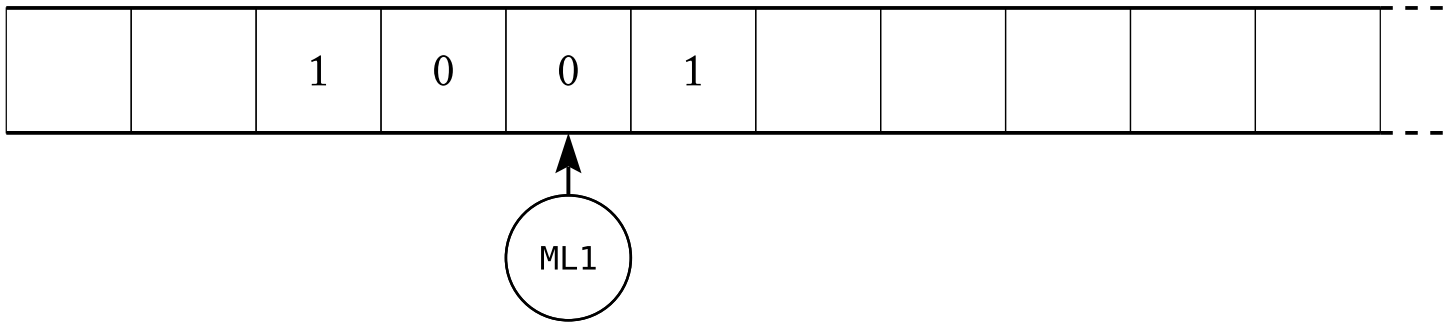
$$\delta(1\,,\,\texttt{LL1}) = (\sqcup\,,\,\texttt{ML1}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



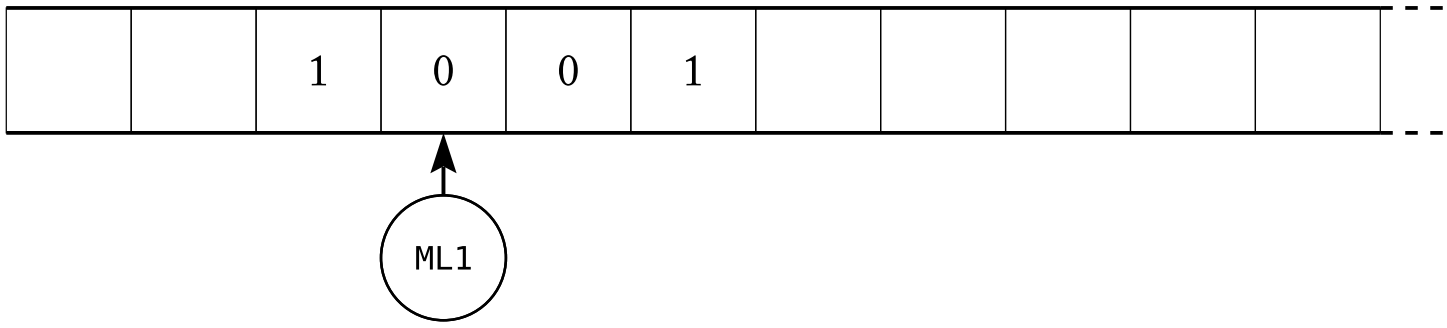
$$\delta(1\,,\,\texttt{ML1}) = (1\,,\,\texttt{ML1}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



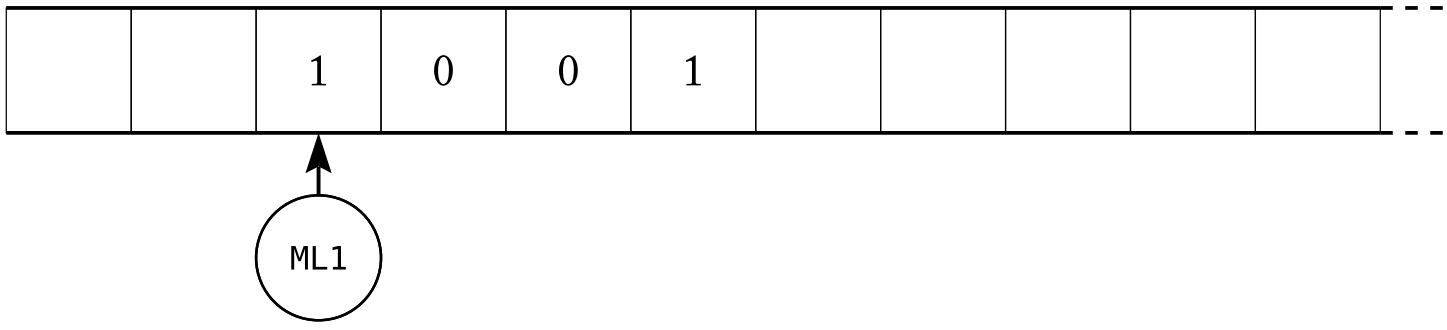
$$\delta(0\,,\,\texttt{ML1}) = (0\,,\,\texttt{ML1}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



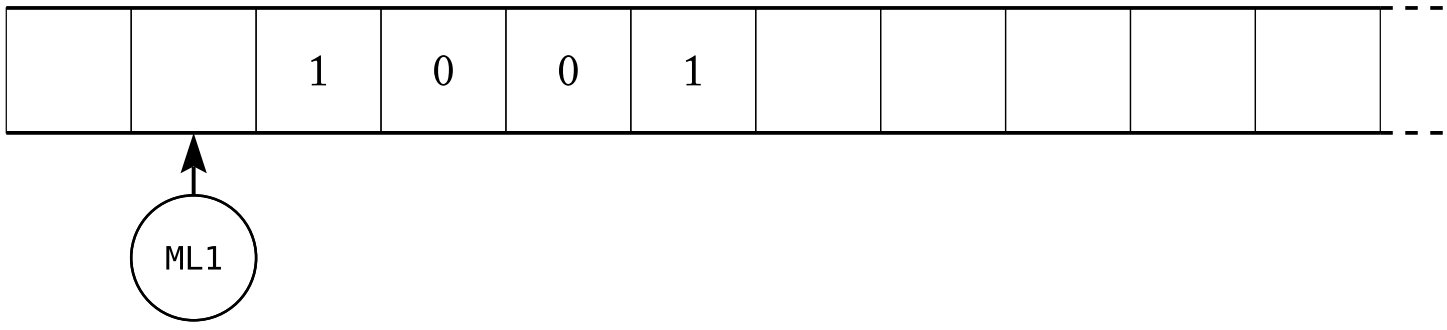
$$\delta(0\,,\,\texttt{ML1}) = (0\,,\,\texttt{ML1}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



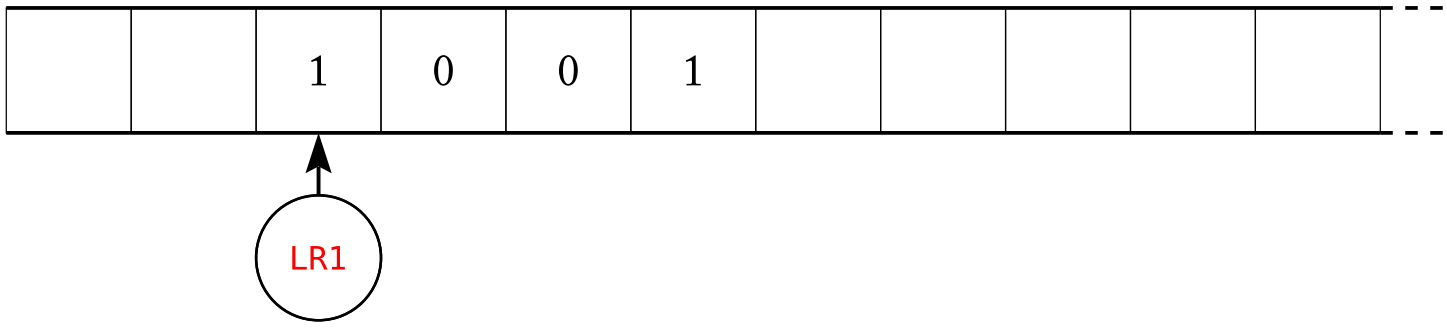
$$\delta(1\,,\,\texttt{ML1}) = (1\,,\,\texttt{ML1}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



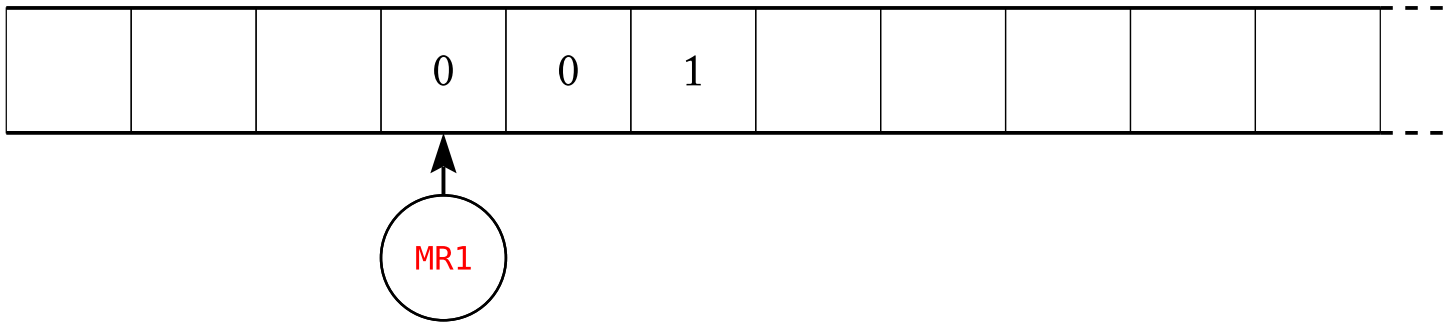
$$\delta(\sqcup\,,\,\texttt{ML1}) = (\sqcup\,,\,\texttt{LR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



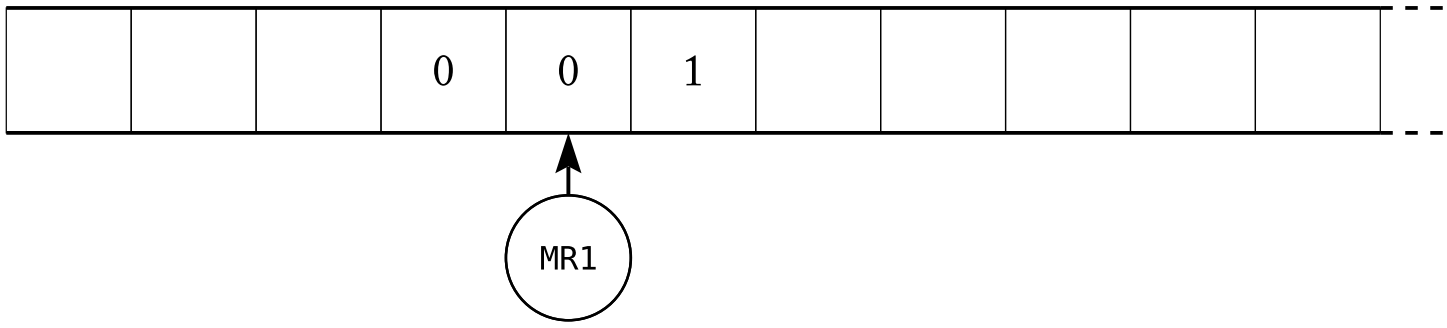
$$\delta(1\,,\,\texttt{LR1}) = (\sqcup\,,\,\texttt{MR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



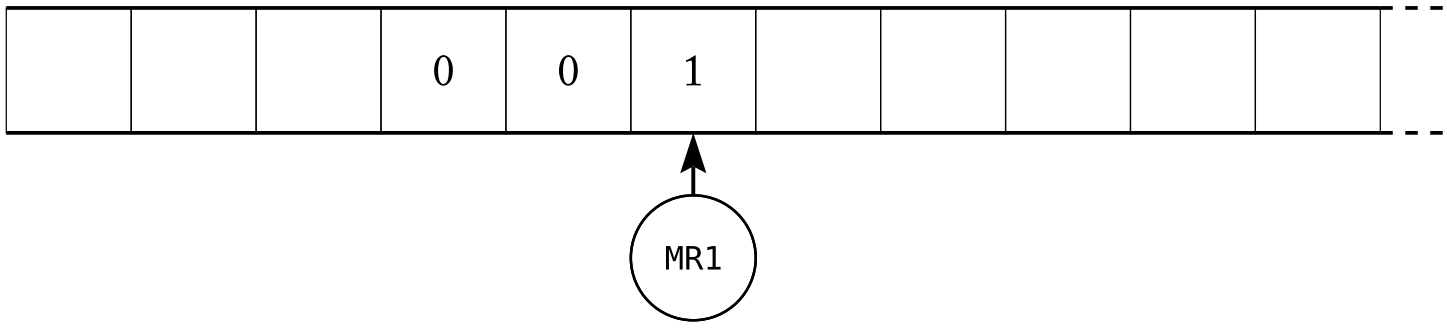
$$\delta(0\,,\,\texttt{MR1}) = (0\,,\,\texttt{MR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



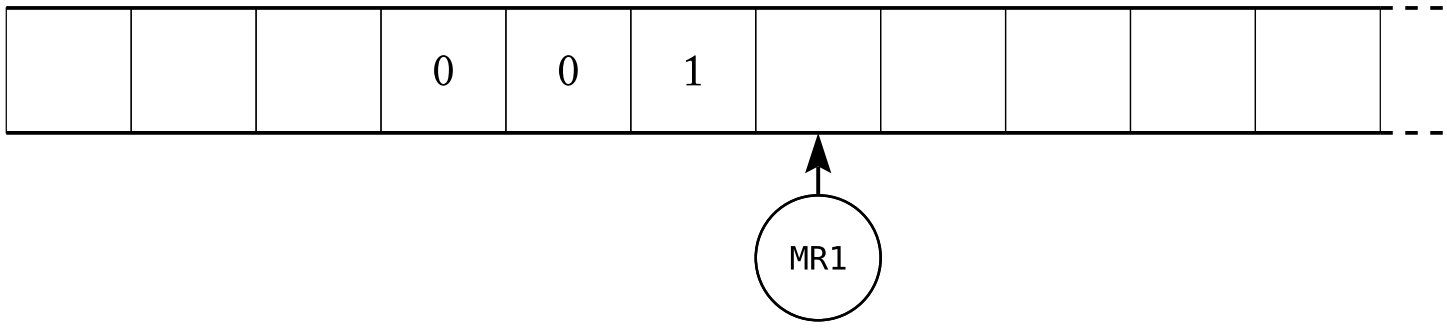
$$\delta(0\,,\,\texttt{MR1}) = (0\,,\,\texttt{MR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



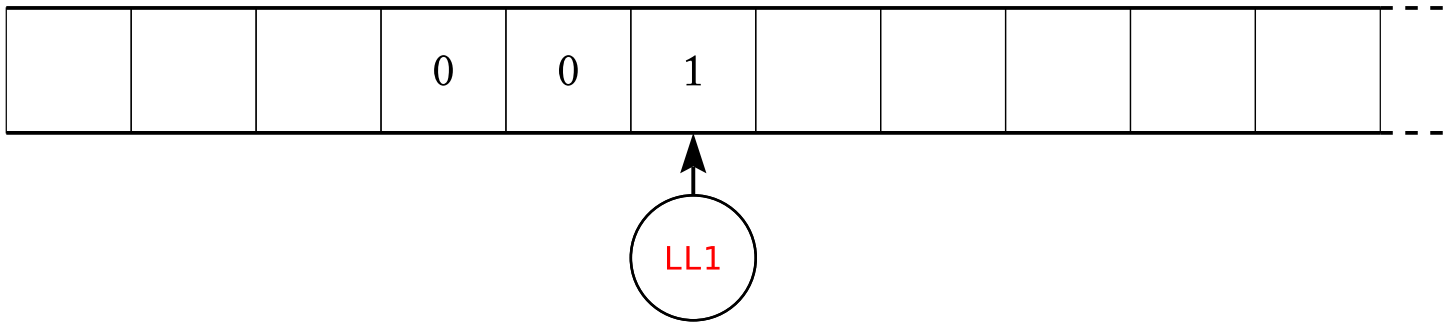
$$\delta(1\,,\,\texttt{MR1}) = (1\,,\,\texttt{MR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



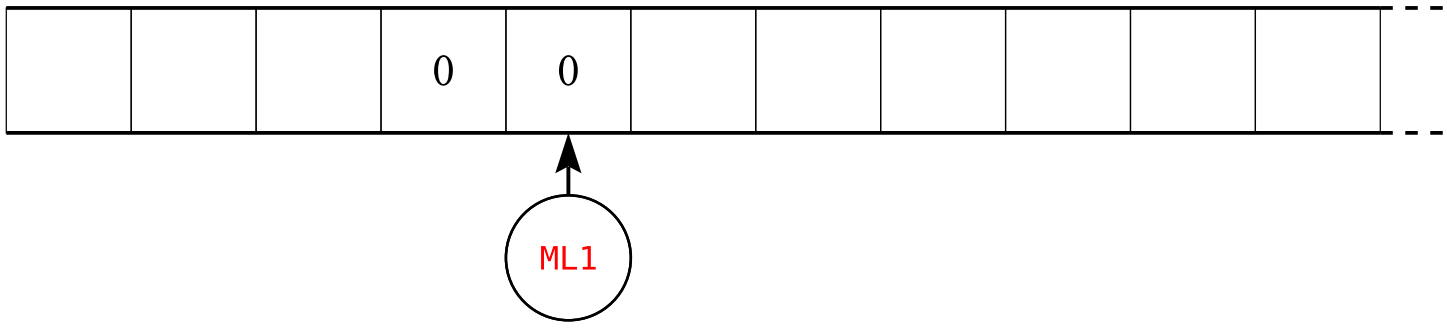
$$\delta(\sqcup\,,\,\texttt{MR1}) = (\sqcup\,,\,\texttt{LL1}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



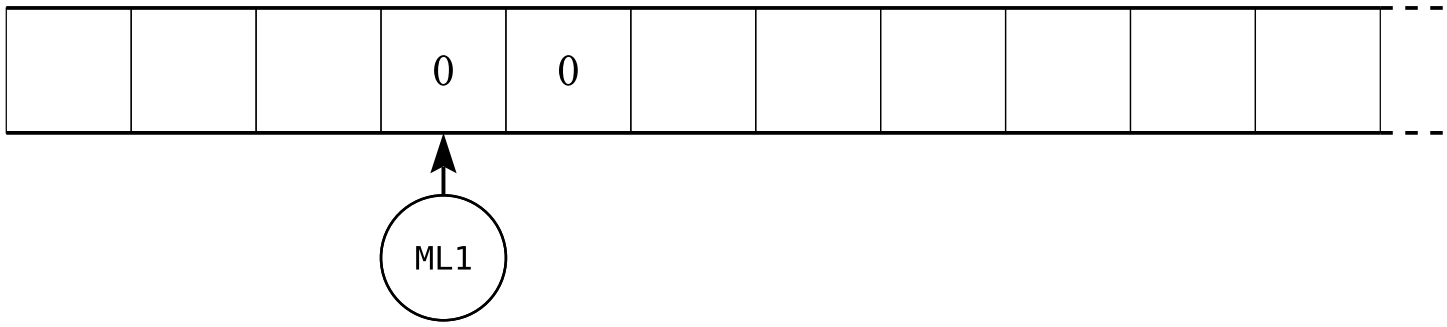
$$\delta(1\,,\,\texttt{LL1}) = (\sqcup\,,\,\texttt{ML1}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



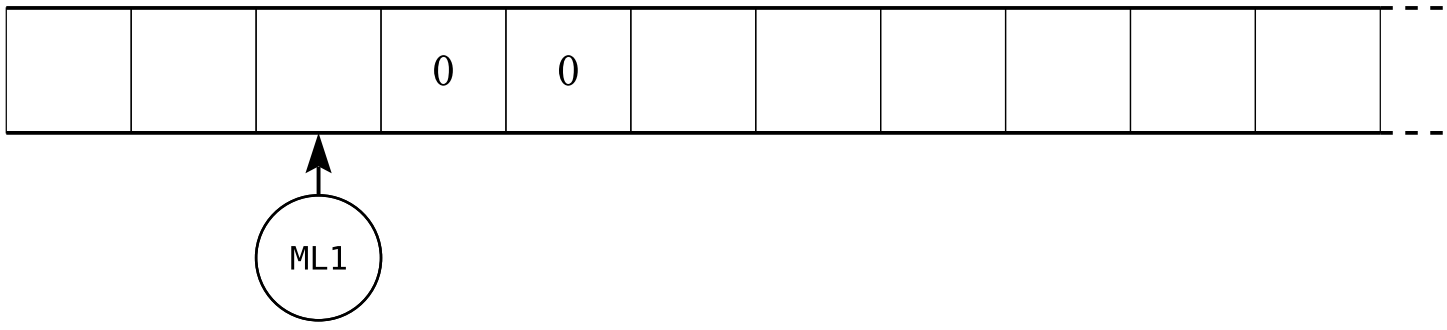
$$\delta(0\,,\,\texttt{ML1}) = (0\,,\,\texttt{ML1}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



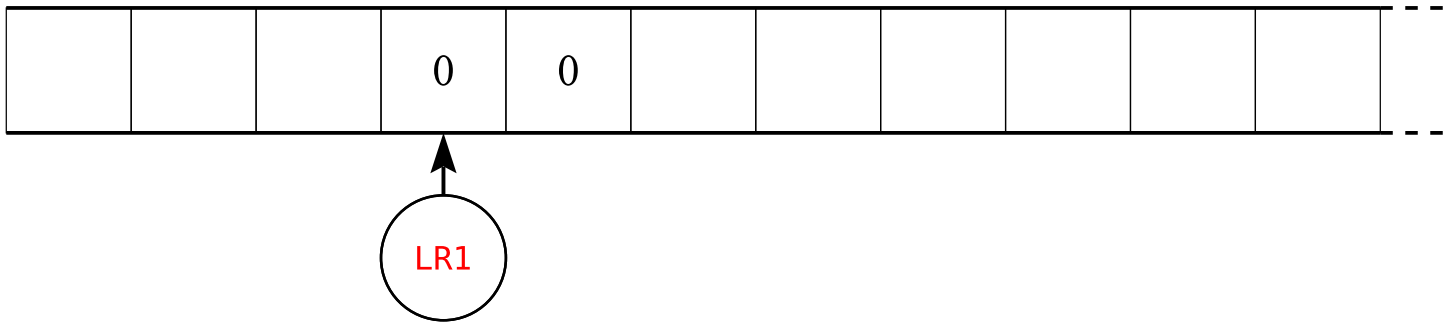
$$\delta(0\,,\,\texttt{ML1}) = (0\,,\,\texttt{ML1}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



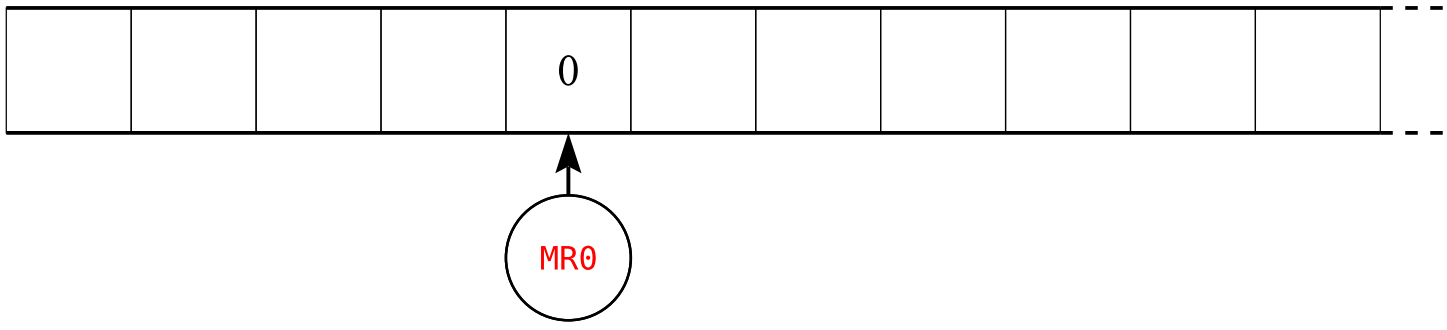
$$\delta(\sqcup\,,\,\texttt{ML1}) = (\sqcup\,,\,\texttt{LR1}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



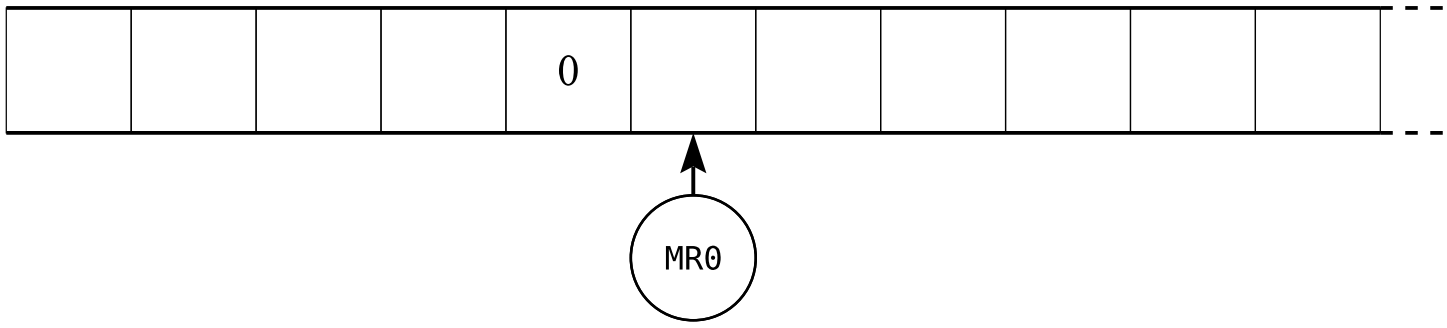
$$\delta(0\,,\,\texttt{LR1}) = (\sqcup\,,\,\texttt{MR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



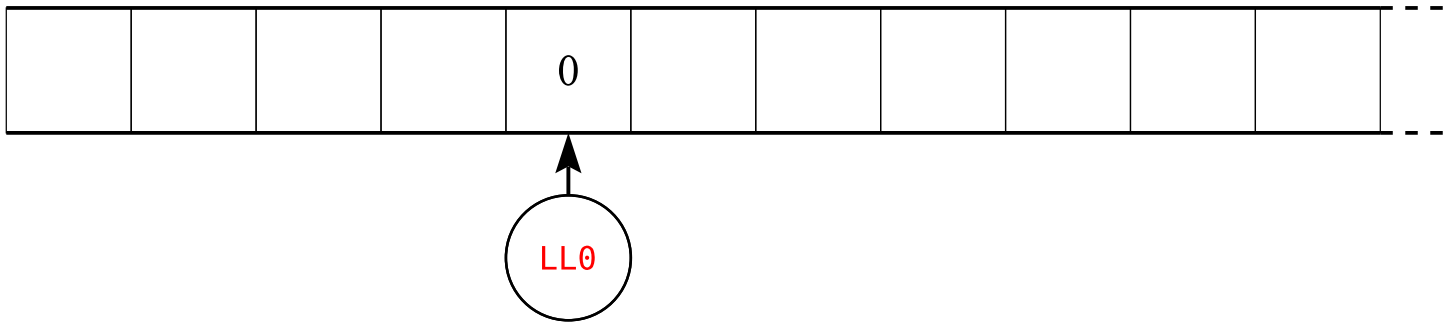
$$\delta(0\,,\,\texttt{MR0}) = (0\,,\,\texttt{MR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



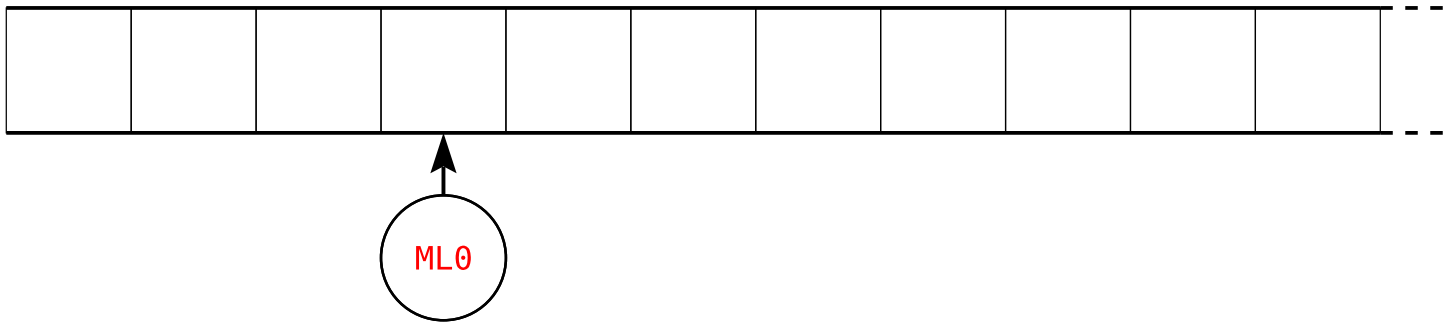
$$\delta(\sqcup, \texttt{MR0}) = (\sqcup, \texttt{LL0}, \leftarrow)$$

Une machine pour reconnaître les palindromes



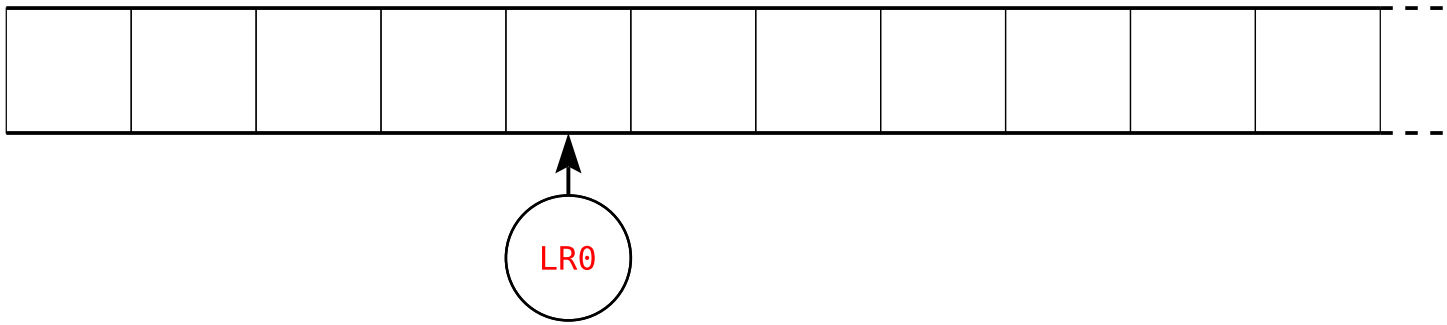
$$\delta(\mathtt{0}\,,\,\mathtt{LL0}) = (\sqcup\,,\,\mathtt{ML0}\,,\,\leftarrow)$$

Une machine pour reconnaître les palindromes



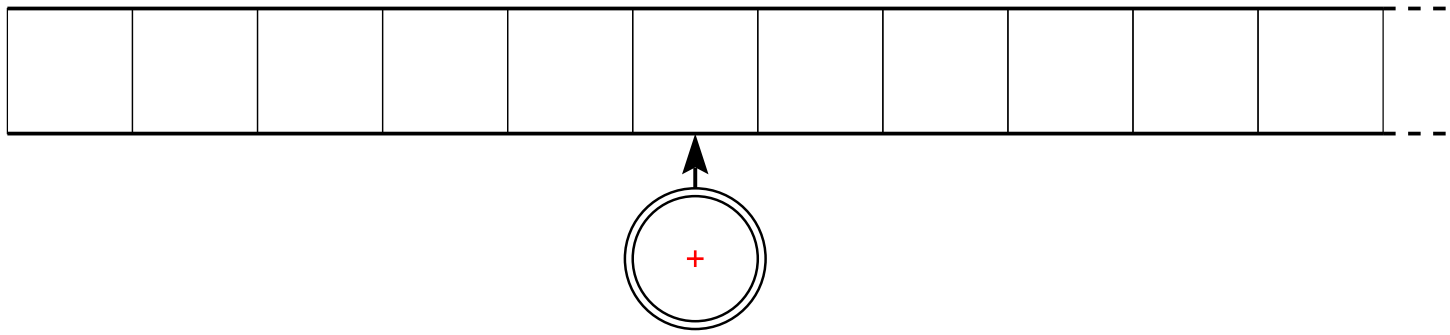
$$\delta(\sqcup\,,\,\texttt{ML0}) = (\sqcup\,,\,\texttt{LR0}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



$$\delta(\sqcup\,,\,\texttt{LR0}) = (\sqcup\,,\,\texttt{+}\,,\,\rightarrow)$$

Une machine pour reconnaître les palindromes



Thèse de Church-Turing et notion de machine universelle

Thèse de Church-Turing

Le modèle de la machine de Turing formalise correctement la notion intuitive d'algorithme fini, déterministe et vérifiable.

Arguments :

- Tous les modèles alternatifs qui ont été développés se sont avérés équivalents, dans le sens où ils décident les même langages : machines à registres, fonctions récursives, lambda-calcul...
- Il existe une machine de Turing universelle

Thèse de Church-Turing et notion de machine universelle

Thèse de Church-Turing

Le modèle de la machine de Turing formalise correctement la notion intuitive d'algorithme fini, déterministe et vérifiable.

Arguments :

- Tous les modèles alternatifs qui ont été développés se sont avérés équivalents, dans le sens où ils décident les même langages : machines à registres, fonctions récursives, lambda-calcul...
- Il existe une machine de Turing universelle

Définition (machine universelle)

Une machine de Turing $\mathcal{M}_\Omega$ est dite universelle lorsque :

- Elle prend en entrée l'encodage d'une machine $\mathcal{M}$ et une chaîne x
- Elle termine si et seulement si $\mathcal{M}$ termine sur x et retourne alors le même résultat

Thèse de Church-Turing et notion de machine universelle

Thèse de Church-Turing

Le modèle de la machine de Turing formalise correctement la notion intuitive d'algorithme fini, déterministe et vérifiable.

Arguments :

- Tous les modèles alternatifs qui ont été développés se sont avérés équivalents, dans le sens où ils décident les même langages : machines à registres, fonctions récursives, lambda-calcul...
- Il existe une machine de Turing universelle

Définition (machine universelle)

Une machine de Turing $\mathcal{M}_\Omega$ est dite universelle lorsque :

- Elle prend en entrée l'encodage d'une machine $\mathcal{M}$ et une chaîne x
- Elle termine si et seulement si $\mathcal{M}$ termine sur x et retourne alors le même résultat

On peut maintenant se poser en toute rigueur la question de la **décidabilité** d'un problème.

Le problème de l'arrêt

Énoncé et indécidabilité

Donnée : une machine de Turing $\mathcal{M}$ et un mot x
Question : $\mathcal{M}$ termine-t-elle sur l'entrée x ?

Le problème de l'arrêt

Énoncé et indécidabilité

Donnée : une machine de Turing $\mathcal{M}$ et un mot x
Question : $\mathcal{M}$ termine-t-elle sur l'entrée x ?

Le problème de l'arrêt est **indécidable**. En effet, supposons que nous disposions d'une fonction `arret` telle que

```
arret(prog, entree)
```

retourne `true` si et seulement si `prog` est une chaîne de caractère contenant le code source d'un programme valide qui s'arrête sur l'entrée `entree`.

Le problème de l'arrêt

Énoncé et indécidabilité

Donnée : une machine de Turing $\mathcal{M}$ et un mot x
Question : $\mathcal{M}$ termine-t-elle sur l'entrée x ?

Le problème de l'arrêt est **indécidable**. En effet, supposons que nous disposions d'une fonction `arret` telle que

`arret(prog, entree)`

retourne `true` si et seulement si `prog` est une chaîne de caractère contenant le code source d'un programme valide qui s'arrête sur l'entrée `entree`.

Considérons alors :

```
def diag(s) :
    if arret(s, s) :
        while true :
            pass
    else :
        return true
```

Question : le programme `diag` termine-t-il sur son propre code source ?

Le problème de l'arrêt

Une conséquence

Théorème

Il existe une machine de Turing $\mathcal{M}$ et une entrée x tels que :

1. *$\mathcal{M}$ ne termine pas sur x*
2. *L'énoncé 1. n'est pas démontrable*

Preuve :
On raisonne ensuite par l'absurde : supposons le théorème faux.

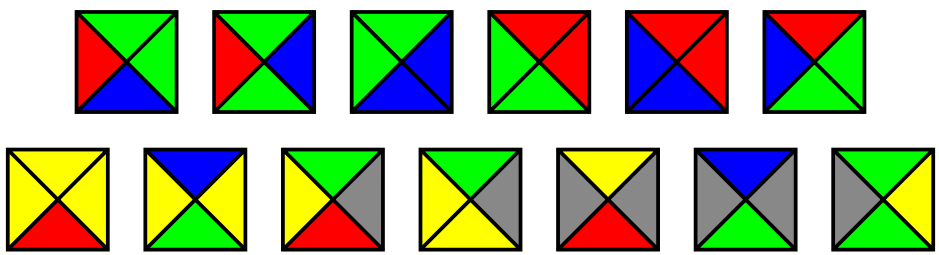
On se donne ensuite une machine $\mathcal{M}$ et une chaîne x. On veut savoir si $\mathcal{M}$ termine sur x. Pour cela, on lance deux procédures en parallèle :

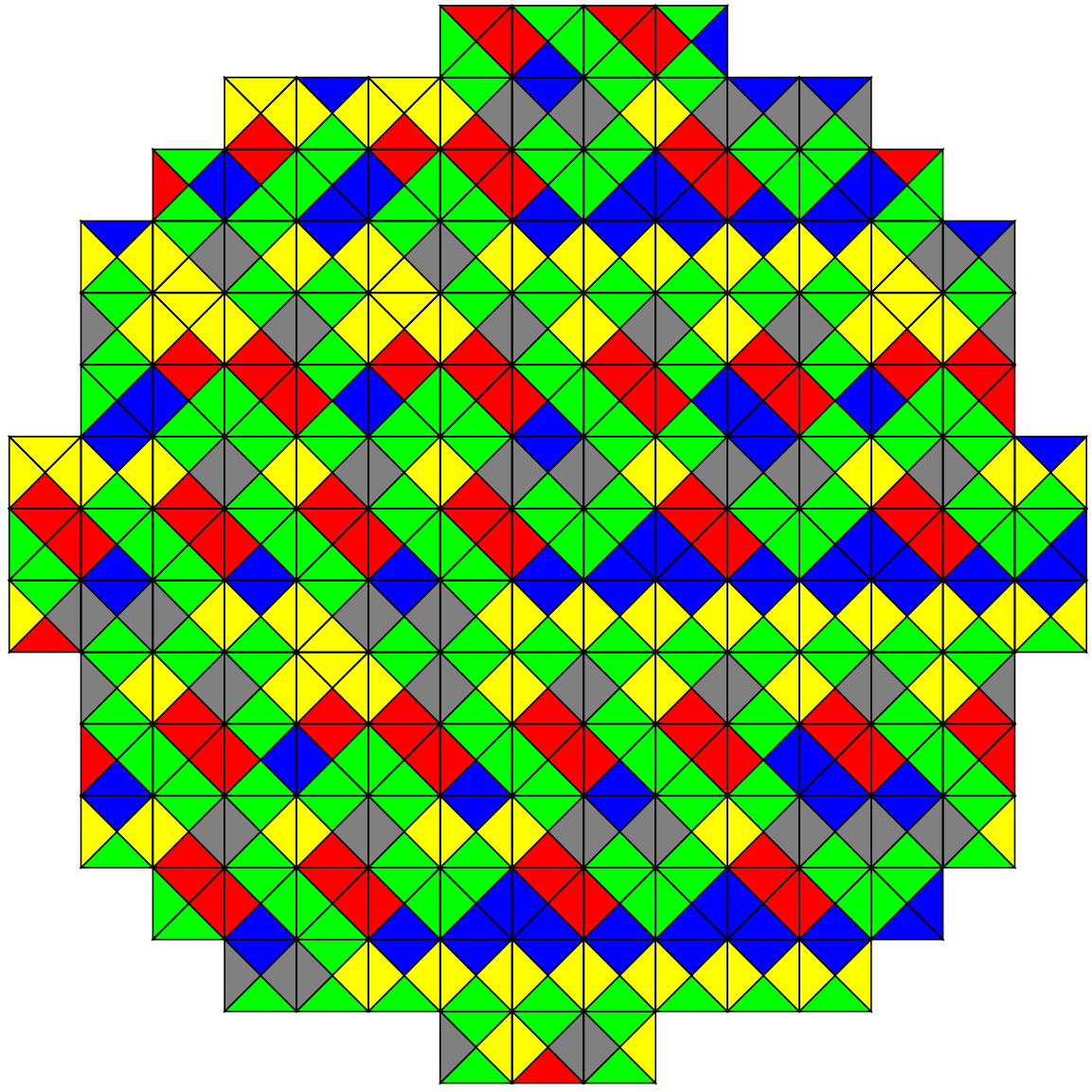
1. On simule l'exécution de $\mathcal{M}$ sur x
2. On énumère toutes les chaînes de caractère, par taille croissante, jusqu'à trouver une preuve de la non terminaison de $\mathcal{M}$ sur x.

Une des deux procédures se termine en temps fini et on décide ainsi du problème de l'arrêt, ce qui est absurde.

Définition

- Soit C un ensemble de couleurs. Une tuile de Wang est un élément de C^4 qui peut être vu comme un carré dont chaque arrête a été colorée.
- Soit Γ un ensemble de tuiles et $A \subset \mathbf{Z}^2$. Un **pavage** de A par Γ est une application de A dans Γ telle que deux tuiles ne peuvent être adjacentes que par des arrêtes de même couleur. Il est dit surjectif s'il utilise chaque tuile de Γ au moins une fois.
- On dit que Γ pave le plan si Γ pave $\mathbf{Z}^2$.





Définitions

- Un pavage f est dit simplement périodique s'il admet une période, c'est à dire un vecteur $u \in \mathbf{Z}^2_*$ tel que :

$$\forall t \in \mathbf{Z}^2,\ f(t+u) = f(t)$$

- Un pavage est dit périodique s'il admet deux périodes non colinéaires.

Propriété

Si un jeu fini de tuiles Γ permet de paver simplement le plan, alors il permet de paver le plan.

Conjecture (Wang, 1961)

Tout jeu fini de tuiles permettant de paver le plan permet de le paver de manière périodique.

Propriété

Un jeu de tuiles permet de paver le plan si et seulement si il permet de paver tout carré de taille $n \times n$.

Propriété

Un jeu de tuiles permet de paver le plan si et seulement si il permet de paver tout carré de taille $n \times n$.

Preuve :

Supposons qu'il est possible de paver tout carré de taille $n \times n$ et donnons-nous un tel pavage P_n pour tout n. L'algorithme suivant construit alors un pavage P_∞ du plan *couronne par couronne* :

- Initialement, P_∞ est vide et $\Omega = \{P_n \mid n \in \mathbf{Z}^2\}$
- Pour k de 1 à ∞ faire :
  - Choisir une couronne de taille k telle que, augmenté de celle-ci, P_∞ constitue un sous-pavage d'une infinité d'éléments de Ω.
  - Étendre P_∞ avec cette couronne.
  - Retirer de Ω les pavages qui n'étendent pas P_∞.

Propriété

Un jeu de tuiles permet de paver le plan si et seulement si il permet de paver tout carré de taille $n \times n$.

Preuve :

Supposons qu'il est possible de paver tout carré de taille $n \times n$ et donnons-nous un tel pavage P_n pour tout n. L'algorithme suivant construit alors un pavage P_∞ du plan *couronne par couronne* :

- Initialement, P_∞ est vide et $\Omega = \{P_n \mid n \in \mathbf{Z}^2\}$
- Pour k de 1 à ∞ faire :
  - Choisir une couronne de taille k telle que, augmenté de celle-ci, P_∞ constitue un sous-pavage d'une infinité d'éléments de Ω.
  - Étendre P_∞ avec cette couronne.
  - Retirer de Ω les pavages qui n'étendent pas P_∞.

Corollaire

Un jeu fini de tuiles permet de paver le plan ssi il permet de paver $\mathbf{N}^2$.

On considère le problème suivant, dit **problème de pavage de Wang** :

Donnée : un jeu fini de tuiles Γ
Question : peut-on paver le plan avec Γ en utilisant toutes les tuile disponibles ?

Théorème (Berger, 1966)

Le problème du pavage de Wang est indécidable.

On considère le problème suivant, dit **problème de pavage de Wang** :

Donnée : un jeu fini de tuiles Γ
Question : peut-on paver le plan avec Γ en utilisant toutes les tuile disponibles ?

Théorème (Berger, 1966)

Le problème du pavage de Wang est indécidable.

La preuve utilise le lemme suivant, variante de l'indécidabilité du théorème de l'arrêt :

Lemme

Le problème de l'arrêt sur le mot vide (ie. le problème qui consiste à déterminer si une machine de Turing $\mathcal{M}$ termine sur l'entrée ϵ) est indécidable.

Preuve du théorème principal :

Supposons que le problème du pavage de Wang est décidable. On va montrer que cela implique la décidabilité du problème de l'arrêt sur le mot vide, ce qui est absurde.

Donnons nous une machine $\mathcal{M}$. On va tenter de trouver un jeu de tuiles $\Gamma_{\mathcal{M}}$ tel que

$$\mathcal{M} \text{ ne termine pas sur } \epsilon \Leftrightarrow \text{ il existe un pavage surjectif de } \mathbf{N}^2 \text{ par } \Gamma_{\mathcal{M}}$$

Théorème

Il existe un jeu fini de tuiles qui permet de paver le plan mais qui n'admet aucun pavage périodique.

Théorème

Il existe un jeu fini de tuiles qui permet de paver le plan mais qui n'admet aucun pavage périodique.

Preuve :

On utilise les résultats suivants :

- L'indécidabilité du problème de pavage de Wang
- Un jeu fini de tuiles permet de paver le plan de manière surjective si et seulement si il permet de paver tout carré de taille $n \times n$ de manière surjective pour n assez grand.

Si Γ est un jeu fini de tuiles, on lance alors en parallèle les deux procédures suivantes :

1. Pour n croissant, on teste tous les agencements de tuiles possibles sur un carré de $n \times n$ à la recherche d'un motif surjectif et périodique.
2. Pour n croissant, on détermine s'il est possible de paver de manière surjective un carré de taille $n \times n$ par recherche exhaustive.

Si la conjecture de Wang est vraie, une des deux procédures termine forcemment, et on décide ainsi le problème de pavage de Wang.

Autour du dixième problème de Hilbert

Énoncé historique

10. De la possibilité de résoudre une équation diophantienne. *Soit la donnée d'une équation diophantienne à un nombre quelconque d'inconnues et à coefficients entiers : on doit trouver une méthode par laquelle, au moyen d'un nombre fini d'opérations, on pourra décider si l'équation est résoluble en nombres entiers.*

Le problème est résolu par la négative en 1970 par Yuri Matiyasevich.

Autour du dixième problème de Hilbert

Énoncé historique

10. De la possibilité de résoudre une équation diophantienne. *Soit la donnée d'une équation diophantienne à un nombre quelconque d'inconnues et à coefficients entiers : on doit trouver une méthode par laquelle, au moyen d'un nombre fini d'opérations, on pourra décider si l'équation est résoluble en nombres entiers.*

Le problème est résolu par la négative en 1970 par Yuri Matiyasevich.

Idée de la preuve

Matiyasevich montre l'existence de :

- Un polynôme $P \in \mathbf{Z}[X_1, \cdots, X_n]$
- Une fonction ϕ qui à une machine $\mathcal{M}$ et une chaîne x associe un entier $\phi(\mathcal{M}, x)$ tels que :

$$\mathcal{M} \text{ termine sur } x \Leftrightarrow \exists\, x_1 \cdots x_{n-1} \in \mathbf{Z},\ P(\phi(\mathcal{M}, x), x_1, \cdots x_{n-1}) = 0$$

Si un problème est décidable, on s'intéresse maintenant à la quantité de ressources de calcul nécessaire à sa résolution.

Définitions

- Si $\mathcal{M}$ est une machine de Turing et $n \in \mathbf{N}$, on note $T_{\mathcal{M}}(n)$ le nombre maximal d'étapes de calcul éffectuées par $\mathcal{M}$ sur une entrée de taille $\leq n$.
- Une machine de Turing est dite polynomiale lorsque sa fonction de complexité est un polynôme.
- On note **P** la classe des langages décidés par une machine de Turing polynomiale.

Définitions

Dans le cas de la machine que nous avons présentée pour reconnaître des palindromes, on a $T_{\mathcal{M}}(n) = O(n^2)$.
Ainsi, le langage des palindromes est dans **P**.

Des questions ?