Pavages, Apériodicité et Calculabilité

École d'été *Math-en-Folie*, Lyon

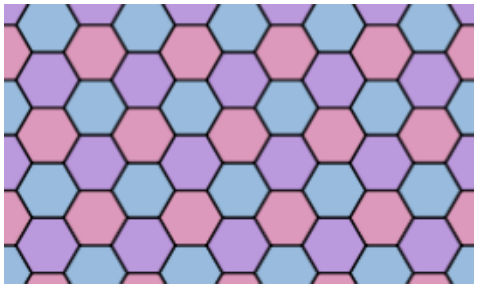
Nathalie AUBRUN

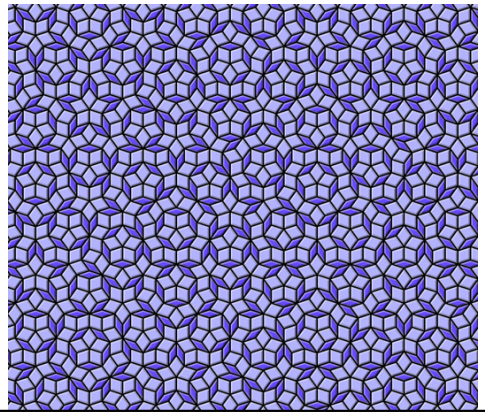
Chargée de recherche CNRS, LIP, ENS de Lyon

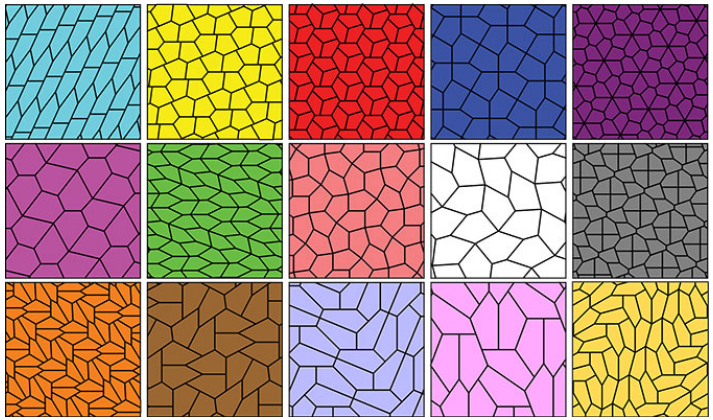
Lundi 22 août 2016

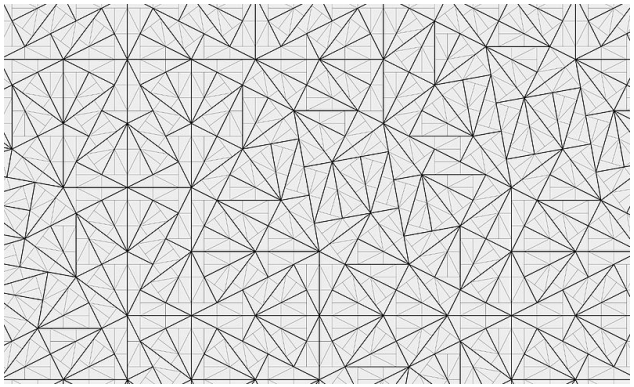
Les pavages, qu'est-ce que c'est ?

Recouvrir le plan (ou l'espace) avec des copies de certaines tuiles de base.









Pour quoi faire ?

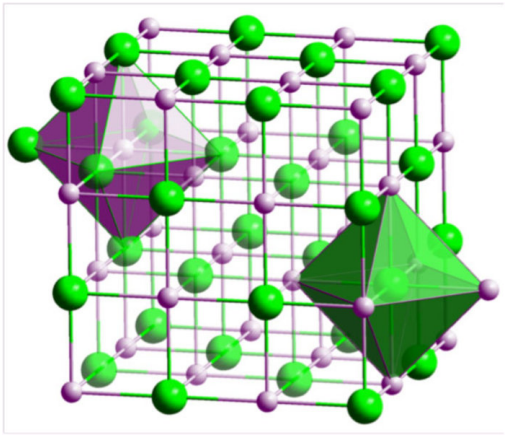
- Ornementer des bâtiments (alhambra de Grenade)
  (17 types de pavages périodiques du plan)

Pour quoi faire ?

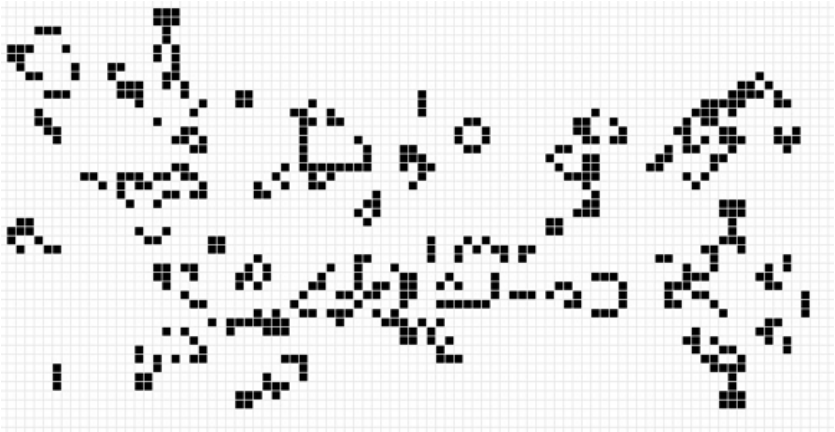
- Paver des rues (Helsinki)
  (pavage de Penrose)

Pour quoi faire ?

- Intérêt en cristallographie (comprendre la structure atomique de la matière)



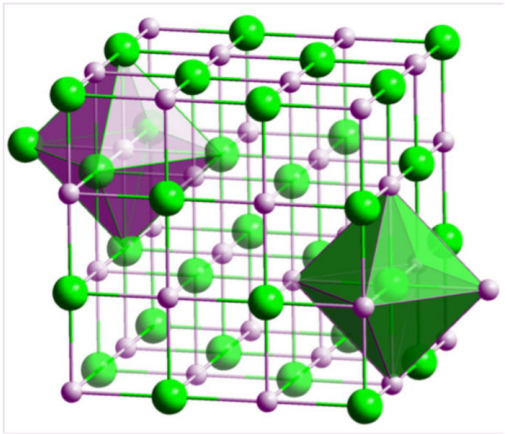
- Modélisation de phénomènes décrits localement (Automates Cellulaires)



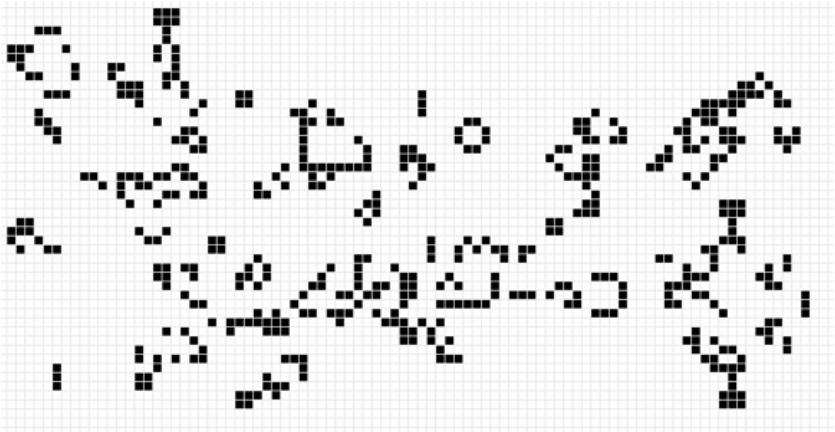
- Définir une manière de réaliser des calculs : tuiles de Wang

Pour quoi faire ?

- Intérêt en cristallographie (comprendre la structure atomique de la matière)



- Modélisation de phénomènes décrits localement (Automates Cellulaires)



- **Définir une manière de réaliser des calculs : tuiles de Wang**

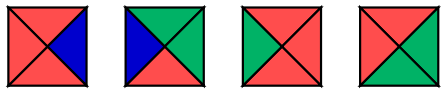
Plan de l'exposé

1. Des tuiles de Wang pour paver le plan
   - Définition
   - Quelques exemples
2. Deux problèmes classiques
   - Le problème du Domino
   - Jeux de tuiles apériodiques
3. Peut-on tout calculer ?
   - Algorithmes, programmes et calculs
   - Un modèle de calcul : les machines de Turing
   - Le problème de l'arrêt

Tuiles de Wang (1961)

Dans cet exposé, on se restreint à des pavages

- du plan,
- par des tuiles carrées portant une couleur par arête,
- avec nombre fini de couleurs,

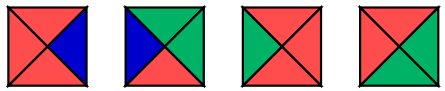


- où les tuiles sont placées sommet contre sommet,
- où deux tuiles voisines portent la même couleur sur leur arête commune.

Tuiles de Wang (1961)

Dans cet exposé, on se restreint à des pavages

- du plan,
- par des tuiles carrées portant une couleur par arête,
- avec nombre fini de couleurs,



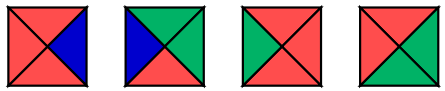
- où les tuiles sont placées sommet contre sommet,
- où deux tuiles voisines portent la même couleur sur leur arête commune.

Interdiction de tourner les tuiles ! !

Tuiles de Wang (1961)

Dans cet exposé, on se restreint à des pavages

- du plan,
- par des tuiles carrées portant une couleur par arête,
- avec nombre fini de couleurs,



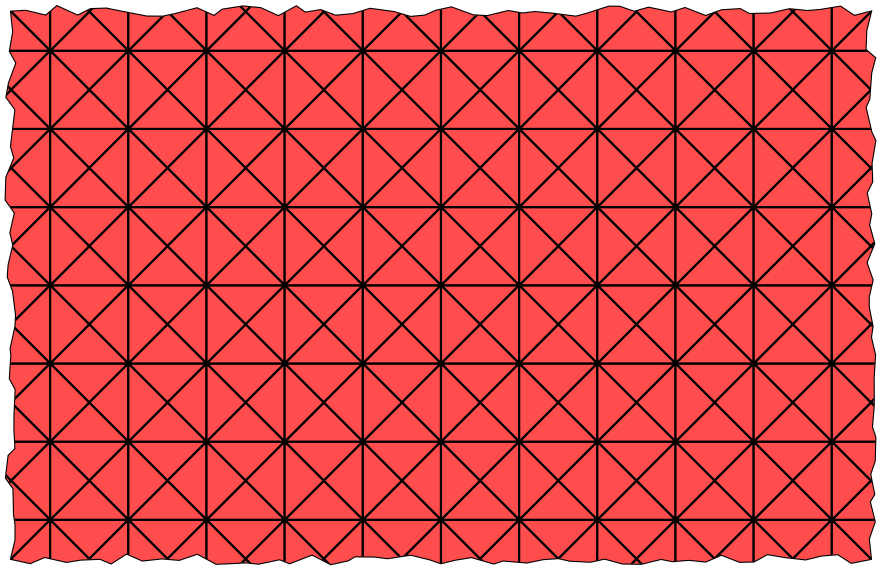
- où les tuiles sont placées sommet contre sommet,
- où deux tuiles voisines portent la même couleur sur leur arête commune.

Interdiction de tourner les tuiles ! !

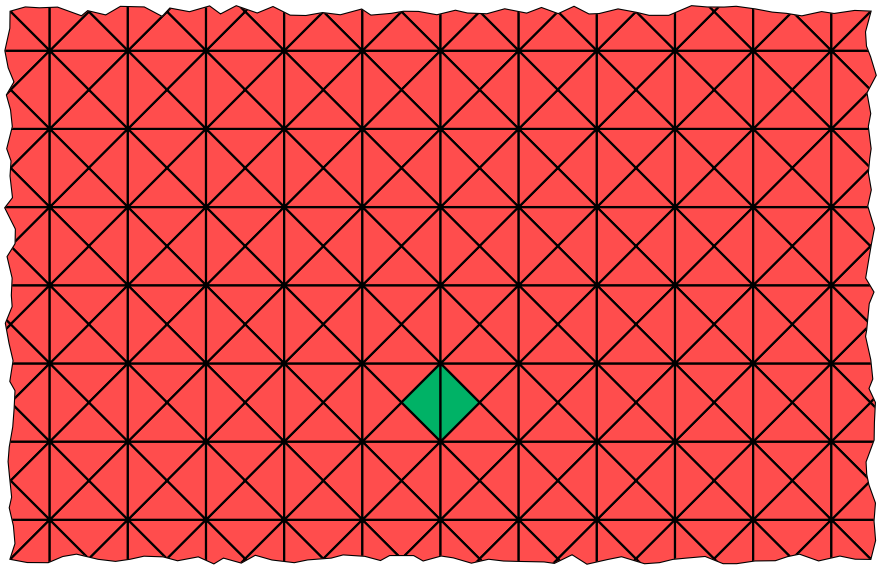
Etant donné un jeu de tuiles τ, on s'intéresse à **tous les pavages valides** par τ.

Premier exemple

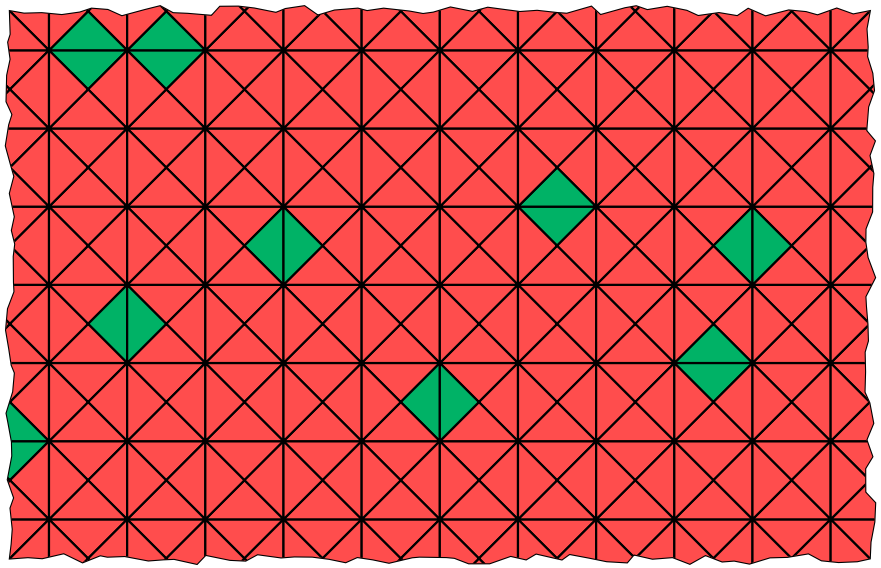
Premier exemple



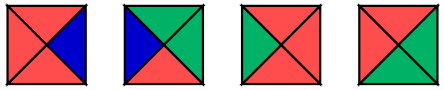
Premier exemple



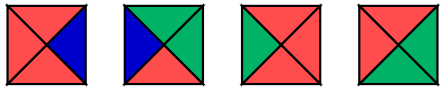
Premier exemple

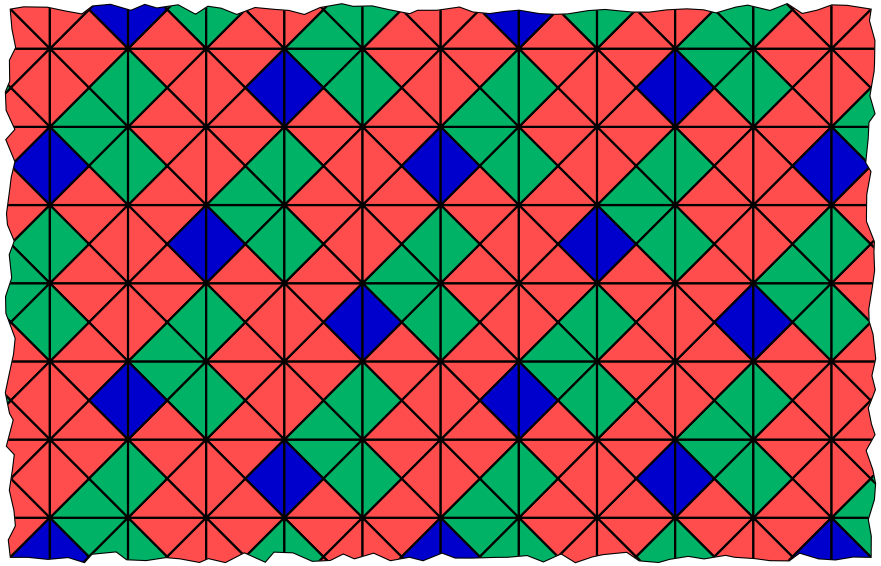


Deuxième exemple

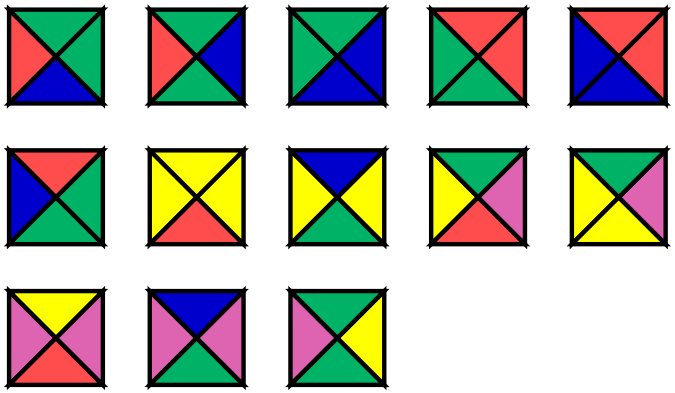


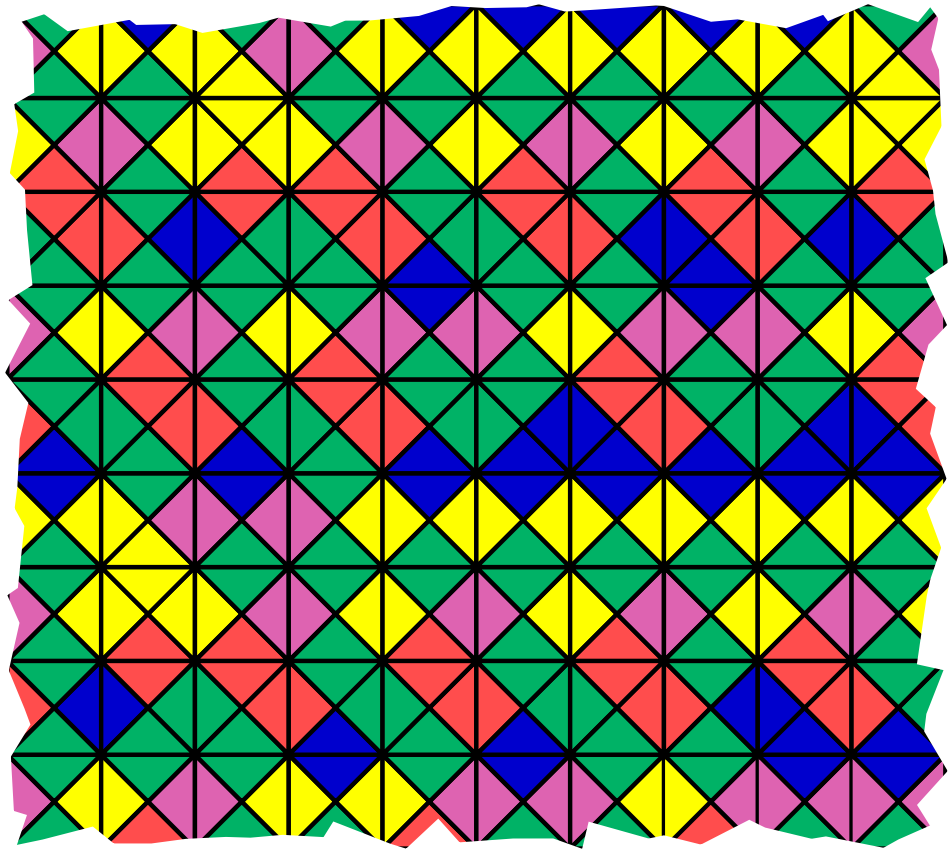
Deuxième exemple





Et celui-là ?



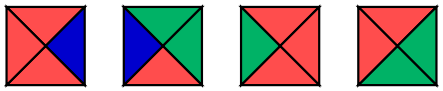


Plan de l'exposé

1. Des tuiles de Wang pour paver le plan
   - Définition
   - Quelques exemples
2. Deux problèmes classiques
   - Le problème du Domino
   - Jeux de tuiles apériodiques
3. Peut-on tout calculer ?
   - Algorithmes, programmes et calculs
   - Un modèle de calcul : les machines de Turing
   - Le problème de l'arrêt

Le problème du Domino : énoncé

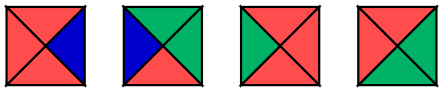
Jeu fini de tuiles de Wang τ (une infinité de copies de chaque tuile)



Règles d'adjacences

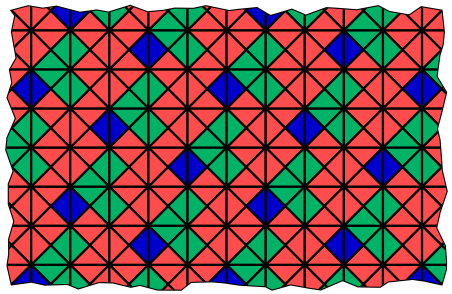
Le problème du Domino : énoncé

Jeu fini de tuiles de Wang τ (une infinité de copies de chaque tuile)



Règles d'adjacences

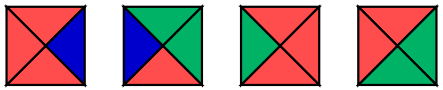
Exemple de pavage par τ



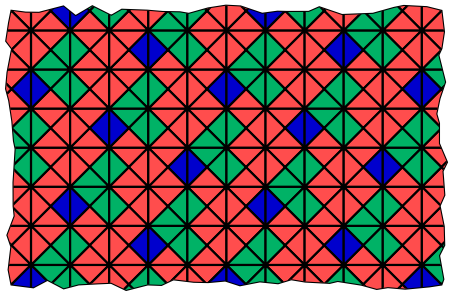
Le problème du Domino : énoncé

Jeu fini de tuiles de Wang τ (une infinité de copies de chaque tuile)

Règles d'adjacences



Exemple de pavage par τ



Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Le problème du Domino : remarque

Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Le problème du Domino : remarque

Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Si le jeu de tuiles τ ne permet pas de paver le plan, je suis capable de le détecter.
Comment ?

Le problème du Domino : remarque

Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Si le jeu de tuiles τ ne permet pas de paver le plan, je suis capable de le détecter.
Comment ?

$\Rightarrow$ Méthode par **essai-erreur**.

Le problème du Domino : remarque

Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Si le jeu de tuiles τ ne permet pas de paver le plan, je suis capable de le détecter.
Comment ?

$\Rightarrow$ Méthode par **essai-erreur**.

Partie difficile du problème : comment savoir qu'un jeu de tuiles permet de paver le plan ?

Le problème du Domino : remarque

Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Si le jeu de tuiles τ ne permet pas de paver le plan, je suis capable de le détecter.
Comment ?

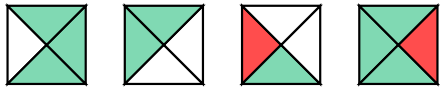
$\Rightarrow$ Méthode par **essai-erreur**.

Partie difficile du problème : comment savoir qu'un jeu de tuiles permet de paver le plan ?

Pour aller plus loin : voir le premier atelier proposé !

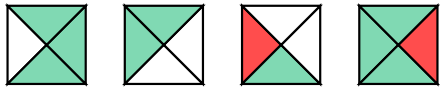
Pavages périodiques (I)

On se donne un jeu de tuiles τ.

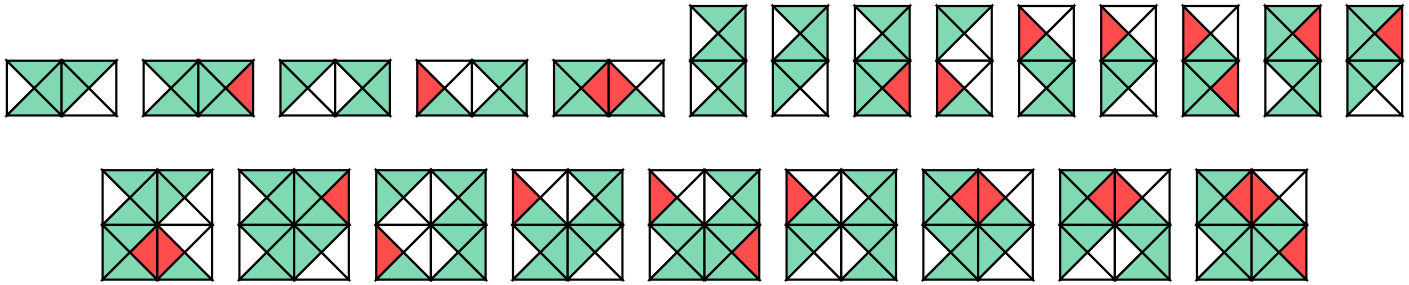


Pavages périodiques (I)

On se donne un jeu de tuiles τ.

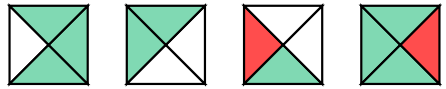


On regarde les motifs finis rectangulaires que l'on peut construire.

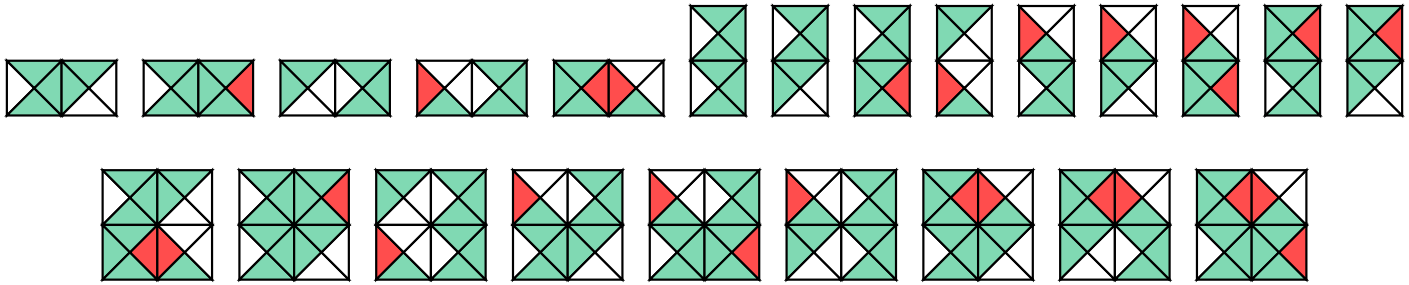


Pavages périodiques (I)

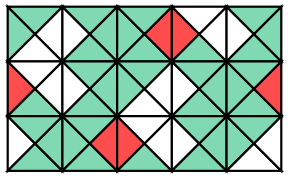
On se donne un jeu de tuiles τ.



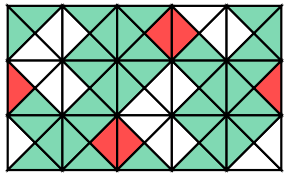
On regarde les motifs finis rectangulaires que l'on peut construire.



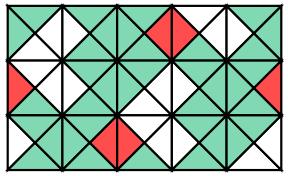
Parmi ceux-ci, on cherche les motifs qui peuvent être « collés à eux-mêmes ».

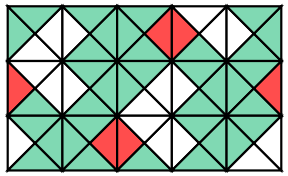


Pavages périodiques (II)

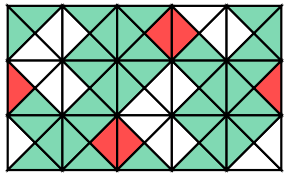


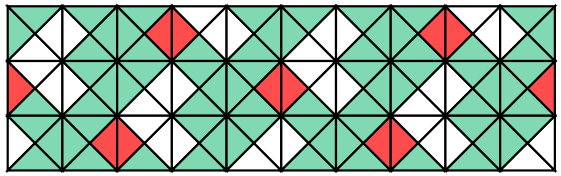
Pavages périodiques (II)



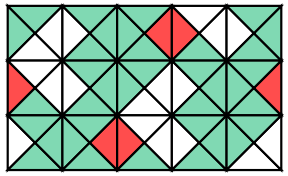


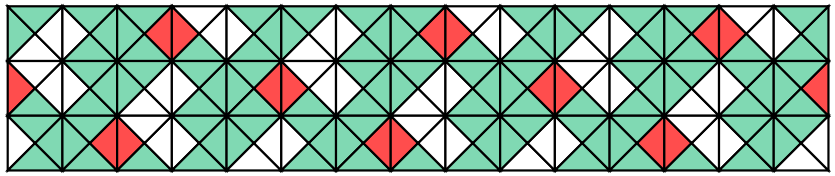
Pavages périodiques (II)



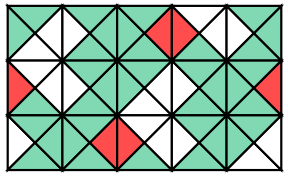


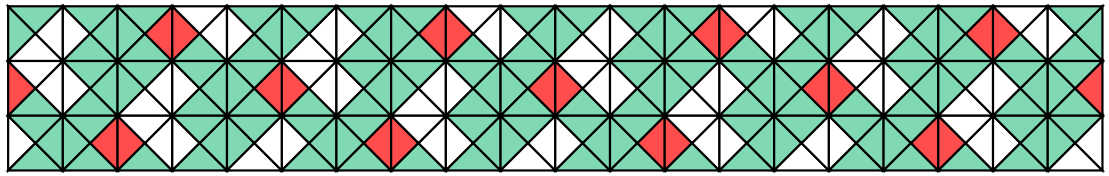
Pavages périodiques (II)



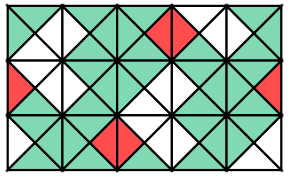


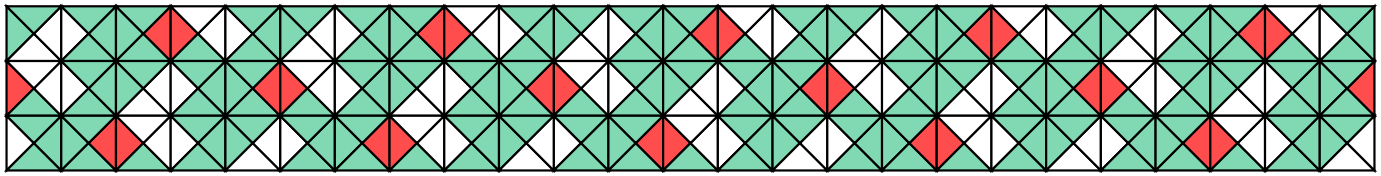
Pavages périodiques (II)



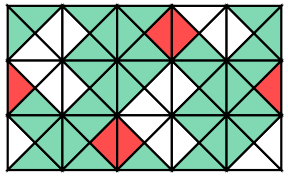


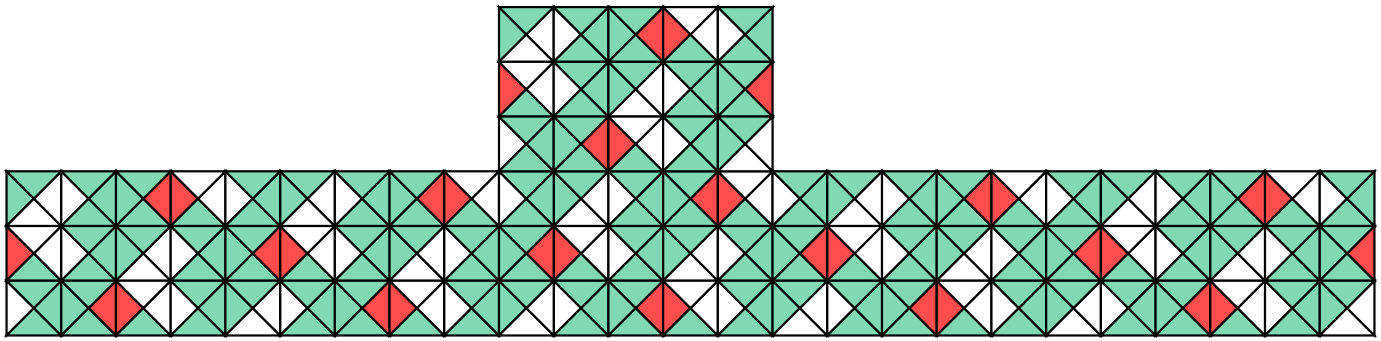
Pavages périodiques (II)



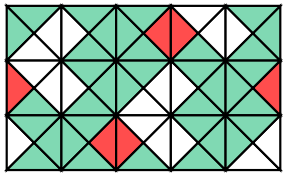


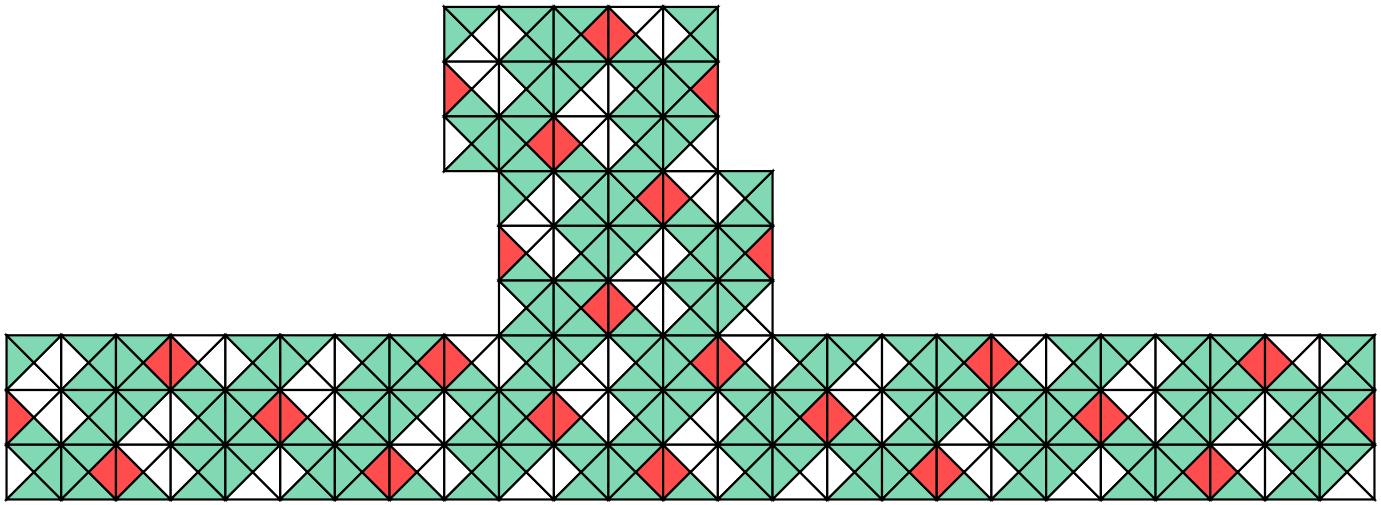
Pavages périodiques (II)



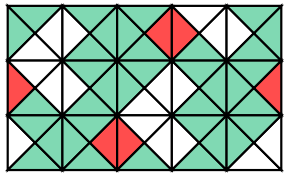


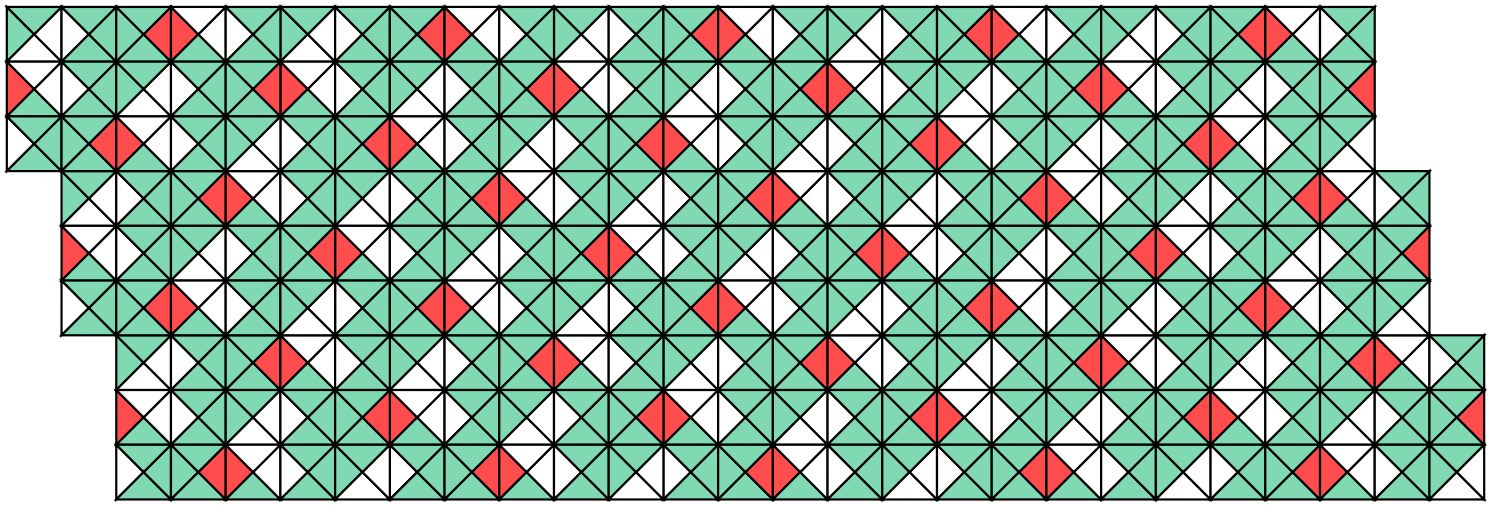
Pavages périodiques (II)





Pavages périodiques (II)



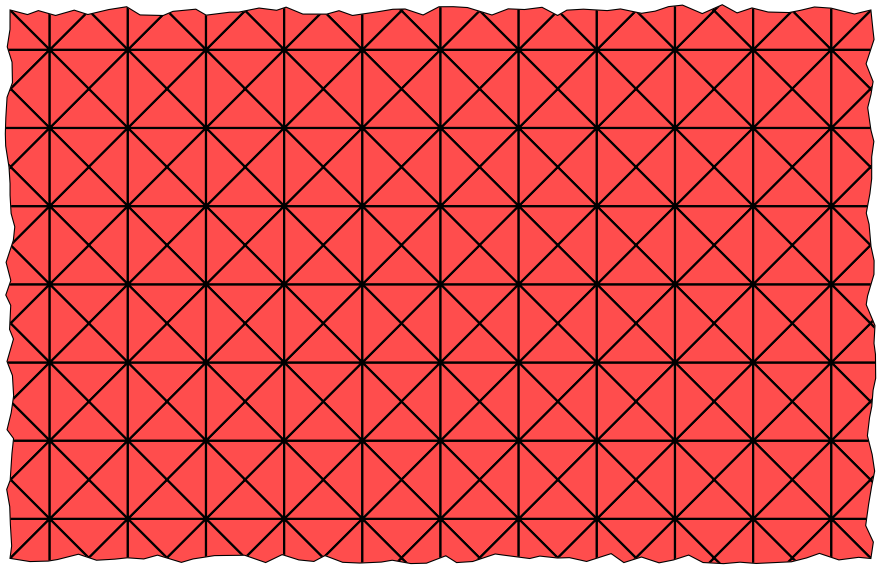


Premier exemple

Peut-on construire un pavage périodique ?

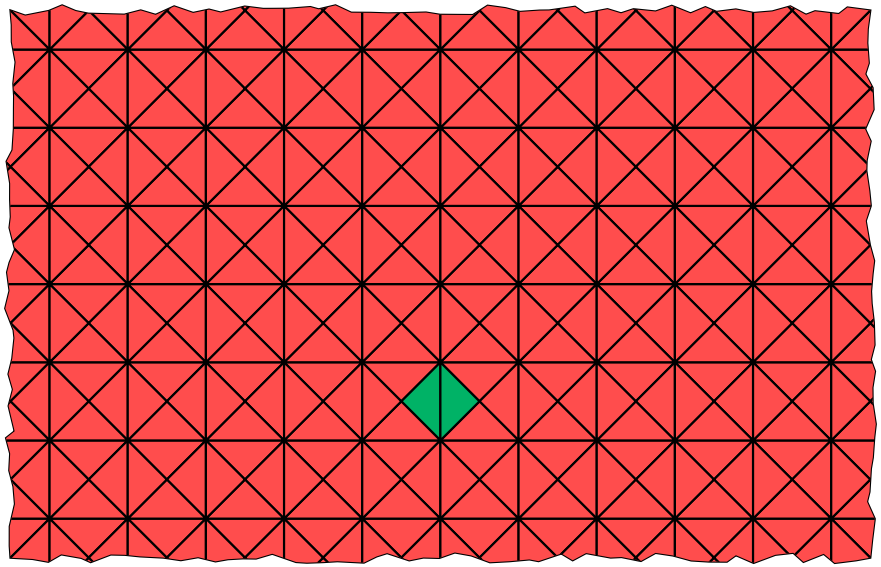
Premier exemple

Peut-on construire un pavage périodique ?



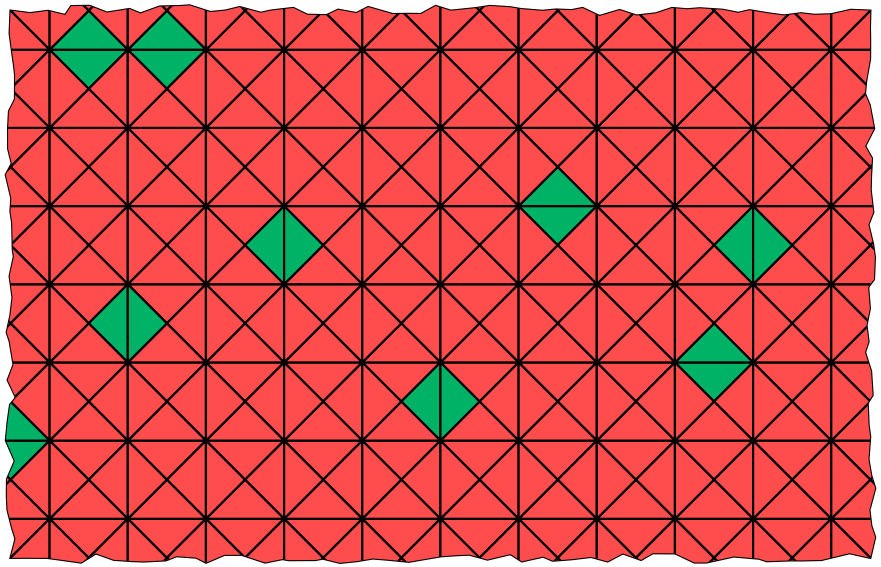
Premier exemple

Peut-on construire un pavage périodique ?

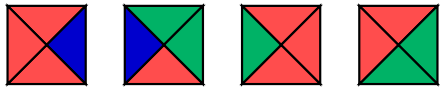


Premier exemple

Peut-on construire un pavage périodique ?

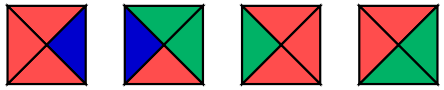


Deuxième exemple

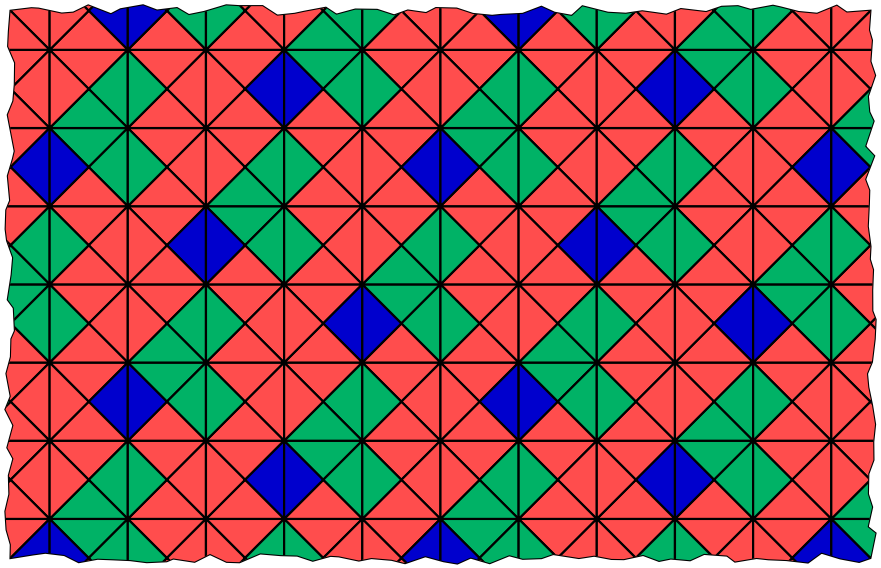


Peut-on construire un pavage périodique ?

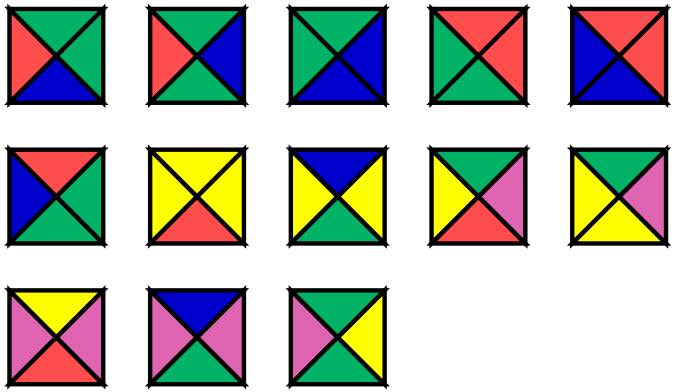
Deuxième exemple

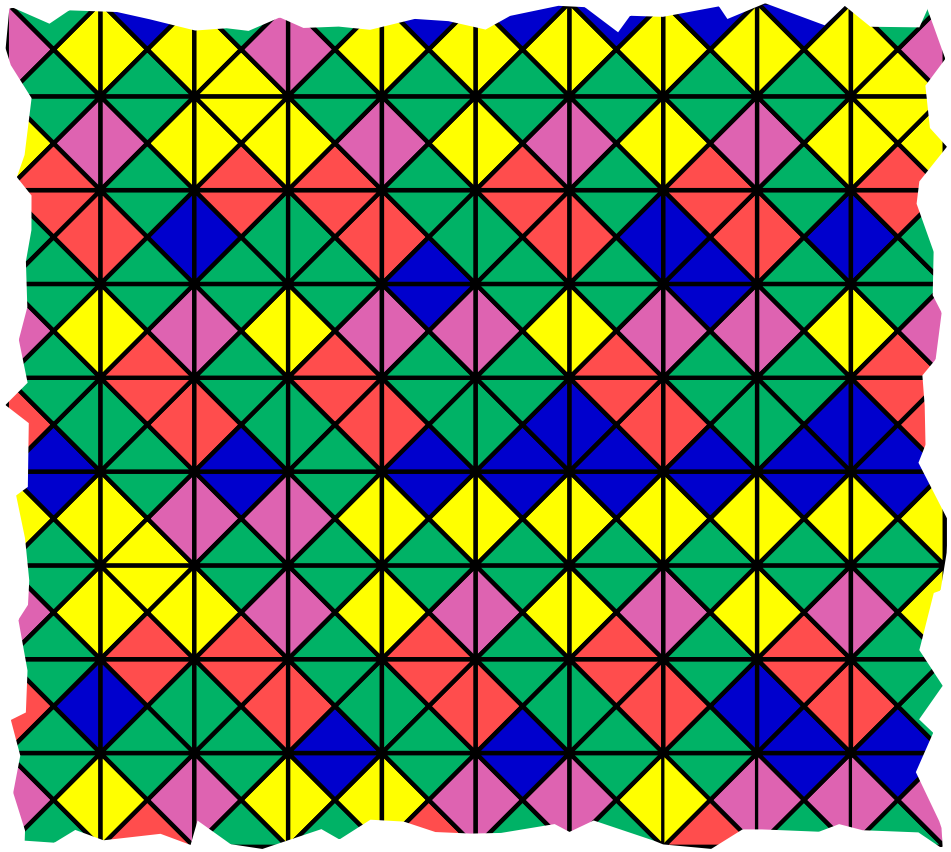


Peut-on construire un pavage périodique ?



Et celui-là ?





Existence de jeux de tuiles apériodiques ?

Un jeu de tuiles est **apériodique** s'il permet de paver le plan, et si aucun des pavages valides n'est périodique.

Conjecture (Wang, 1961)

Si un jeu de tuiles de Wang permet de paver le plan, alors il peut toujours le faire de manière périodique.

Question

Peut-on on construire un jeu de tuiles de Wang apériodique ?

Existence de jeux de tuiles apériodiques ?

Un jeu de tuiles est **apériodique** s'il permet de paver le plan, et si aucun des pavages valides n'est périodique.

Conjecture (Wang, 1961)

Si un jeu de tuiles de Wang permet de paver le plan, alors il peut toujours le faire de manière périodique.

Question

Peut-on on construire un jeu de tuiles de Wang apériodique ?

Pour aller plus loin : voir le deuxième atelier proposé !

Plan de l'exposé

1. Des tuiles de Wang pour paver le plan
   - Définition
   - Quelques exemples
2. Deux problèmes classiques
   - Le problème du Domino
   - Jeux de tuiles apériodiques
3. Peut-on tout calculer ?
   - Algorithmes, programmes et calculs
   - Un modèle de calcul : les machines de Turing
   - Le problème de l'arrêt

Algorithmes, programmes et calculs (I)

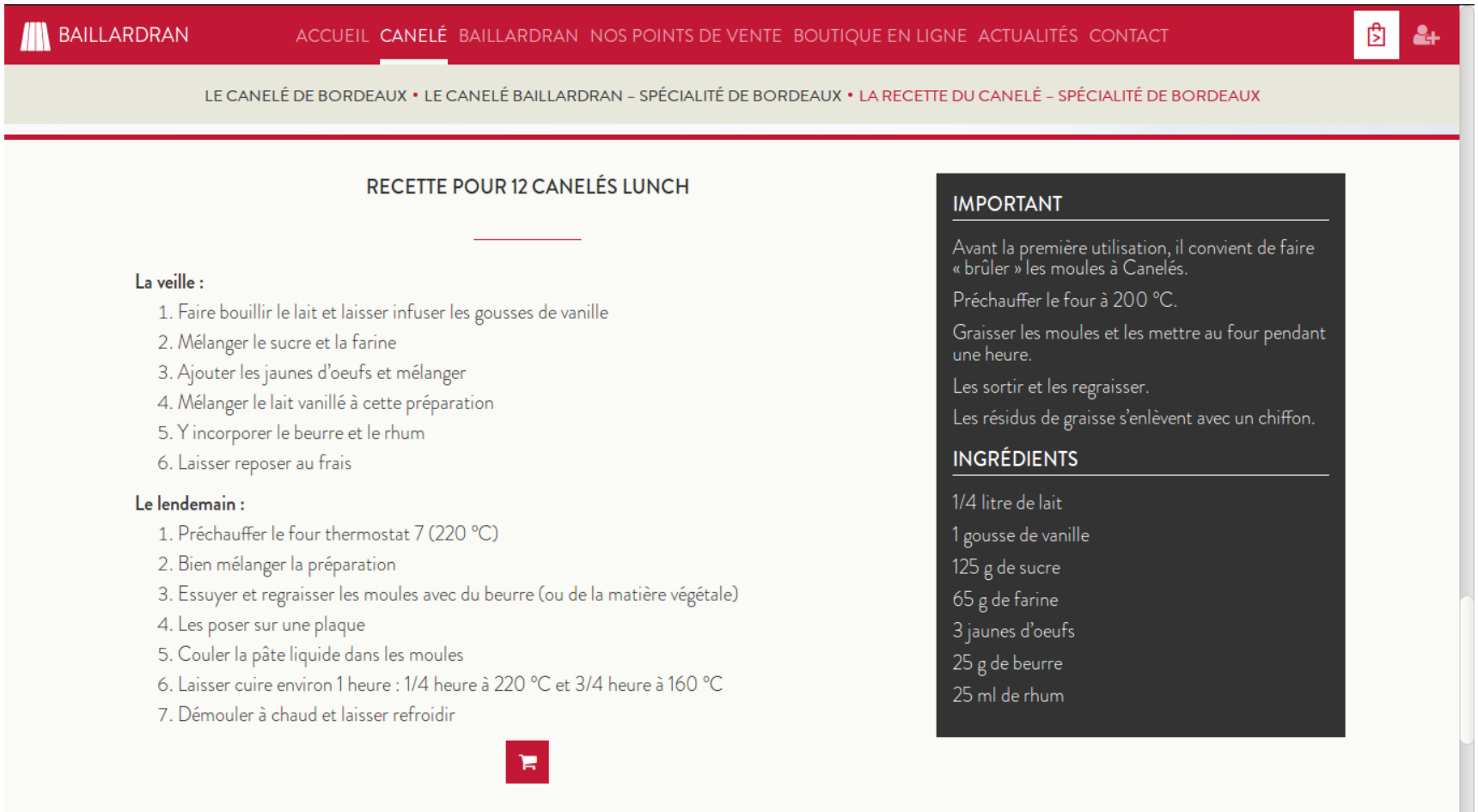
Un algorithme est une **suite d'instructions** qui conduit à un résultat. Par exemple :

- cuisiner des canelés,
- se rendre au musée des Confluences depuis ce bâtiment,
- calculer x^n où x est un réel positif et n est un entier positif.

Algorithmes, programmes et calculs (I)

Un algorithme est une **suite d'instructions** qui conduit à un résultat. Par exemple :

- cuisiner des canelés,
- se rendre au musée des Confluences depuis ce bâtiment,
- calculer x^n où x est un réel positif et n est un entier positif.



BAILLARDRAN ACCUEIL CANELÉ BAILLARDRAN NOS POINTS DE VENTE BOUTIQUE EN LIGNE ACTUALITÉS CONTACT

LE CANELÉ DE BORDEAUX • LE CANELÉ BAILLARDRAN – SPÉCIALITÉ DE BORDEAUX • LA RECETTE DU CANELÉ – SPÉCIALITÉ DE BORDEAUX

RECETTE POUR 12 CANELÉS LUNCH

La veille :

1. Faire bouillir le lait et laisser infuser les gousses de vanille
2. Mélanger le sucre et la farine
3. Ajouter les jaunes d'oeufs et mélanger
4. Mélanger le lait vanillé à cette préparation
5. Y incorporer le beurre et le rhum
6. Laisser reposer au frais

Le lendemain :

1. Préchauffer le four thermostat 7 (220 °C)
2. Bien mélanger la préparation
3. Essuyer et regraisser les moules avec du beurre (ou de la matière végétale)
4. Les poser sur une plaque
5. Couler la pâte liquide dans les moules
6. Laisser cuire environ 1 heure : 1/4 heure à 220 °C et 3/4 heure à 160 °C
7. Démouler à chaud et laisser refroidir

IMPORTANT

Avant la première utilisation, il convient de faire « brûler » les moules à Canelés.

Préchauffer le four à 200 °C.

Graisser les moules et les mettre au four pendant une heure.

Les sortir et les regraisser.

Les résidus de graisse s'enlèvent avec un chiffon.

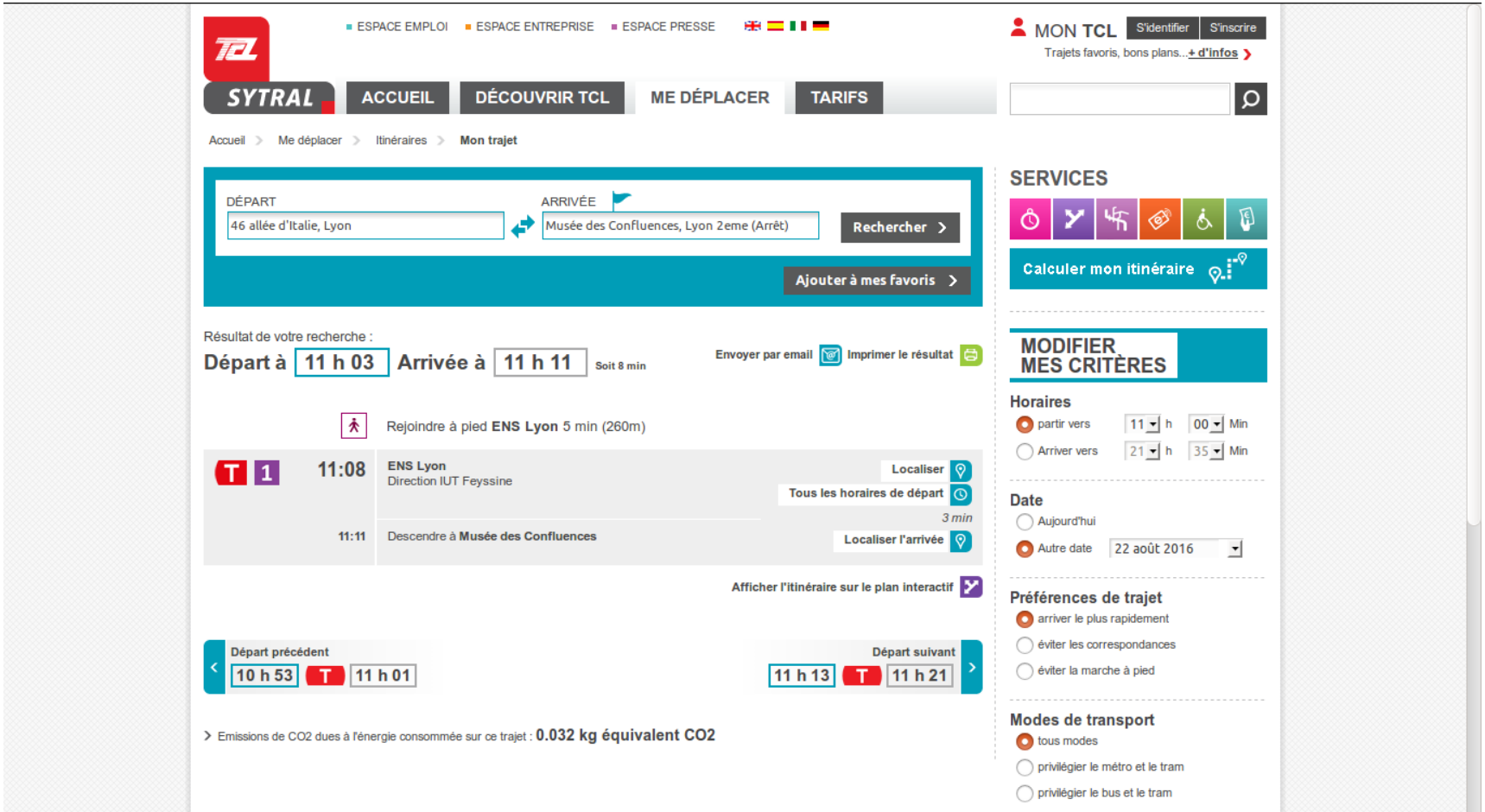
INGRÉDIENTS

1/4 litre de lait
1 gousse de vanille
125 g de sucre
65 g de farine
3 jaunes d'oeufs
25 g de beurre
25 ml de rhum

Algorithmes, programmes et calculs (I)

Un algorithme est une **suite d'instructions** qui conduit à un résultat. Par exemple :

- cuisiner des canelés,
- se rendre au musée des Confluences depuis ce bâtiment,
- calculer x^n où x est un réel positif et n est un entier positif.



Algorithmes, programmes et calculs (I)

Un algorithme est une **suite d'instructions** qui conduit à un résultat. Par exemple :

- cuisiner des canelés,
- se rendre au musée des Confluences depuis ce bâtiment,
- calculer x^n où x est un réel positif et n est un entier positif.

```
def puissance(x,n):
    resultat=1
    for k in range(n):
        resultat = resultat*x
return resultat
```

Algorithmes, programmes et calculs (I)

Un algorithme est une **suite d'instructions** qui conduit à un résultat. Par exemple :

- cuisiner des canelés,
- se rendre au musée des Confluences depuis ce bâtiment,
- calculer x^n où x est un réel positif et n est un entier positif.

```
def puissancebis(x,n):
    if n == 0: return 1
    if n%2 == 0: return puissancebis(x,n//2)**2
    return x*puissancebis(x,(n-1)//2)**2
```

Un algorithme peut servir à **résoudre un problème**. L'algorithme est **correct** si le résultat est celui attendu, incorrect sinon.

Algorithmes, programmes et calculs (II)

Dans votre calculatrice, ordinateur, téléphone, etc. . . une information est codée par une suite de nombres.

Algorithmes, programmes et calculs (II)

Dans votre calculatrice, ordinateur, téléphone, etc. . . une information est codée par une suite de nombres.

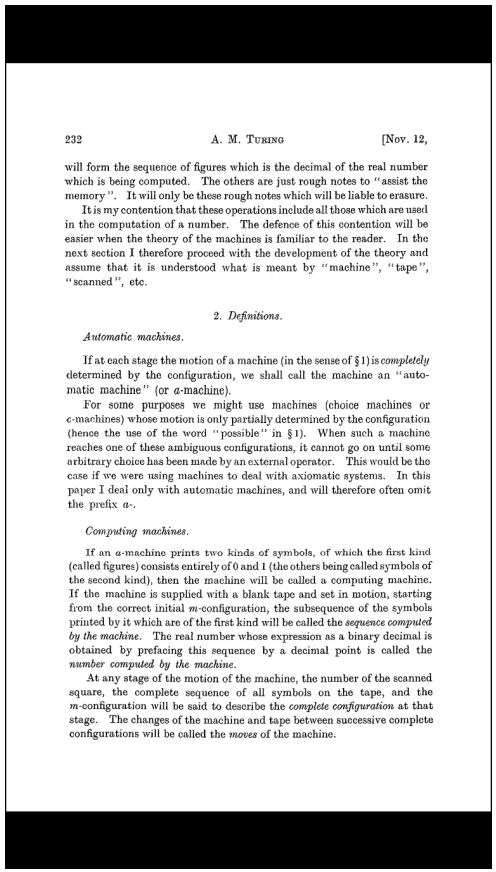
Un **calcul**, c'est donc une transformation d'une suite de nombres en une autre.

Machines de Turing (I)

Imaginées par Alan Turing en 1936, pour délimiter ce qu'un d'algorithme est capable de faire ou non.

Machines de Turing (I)

Imaginées par Alan Turing en 1936, pour délimiter ce qu'un d'algorithme est capable de faire ou non.



232 A. M. TURING [Nov. 12,

will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to "assist the memory". It will only be these rough notes which will be liable to erasure.

It is my contention that these operations include all those which are used in the computation of a number. The defence of this contention will be easier when the theory of the machines is familiar to the reader. In the next section I therefore proceed with the development of the theory and assume that it is understood what is meant by "machine", "tape", "scanned", etc.

2. *Definitions.*

Automatic machines.

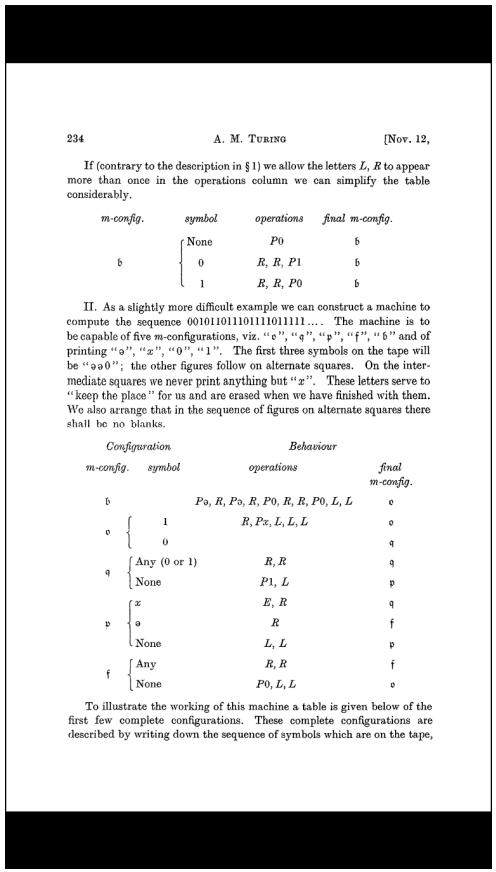
If at each stage the motion of a machine (in the sense of § 1) is *completely* determined by the configuration, we shall call the machine an "automatic machine" (or *a*-machine).

For some purposes we might use machines (choice machines or *c*-machines) whose motion is only partially determined by the configuration (hence the use of the word "possible" in § 1). When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator. This would be the case if we were using machines to deal with axiomatic systems. In this paper I deal only with automatic machines, and will therefore often omit the prefix *a*-.

Computing machines.

If an *a*-machine prints two kinds of symbols, of which the first kind (called figures) consists entirely of 0 and 1 (the others being called symbols of the second kind), then the machine will be called a computing machine. If the machine is supplied with a blank tape and set in motion, starting from the correct initial *m*-configuration, the subsequence of the symbols printed by it which are of the first kind will be called the *sequence computed by the machine*. The real number whose expression as a binary decimal is obtained by prefacing this sequence by a decimal point is called the *number computed by the machine*.

At any stage of the motion of the machine, the number of the scanned square, the complete sequence of all symbols on the tape, and the *m*-configuration will be said to describe the *complete configuration* at that stage. The changes of the machine and tape between successive complete configurations will be called the *moves* of the machine.



234 A. M. TURING [Nov. 12,

If (contrary to the description in § 1) we allow the letters L, R to appear more than once in the operations column we can simplify the table considerably.

m-config.	*symbol*	*operations*	*final m-config.*
$\mathfrak{b}$	None	$P0$	$\mathfrak{b}$
	0	$R, R, P1$	$\mathfrak{b}$
	1	$R, R, P0$	$\mathfrak{b}$

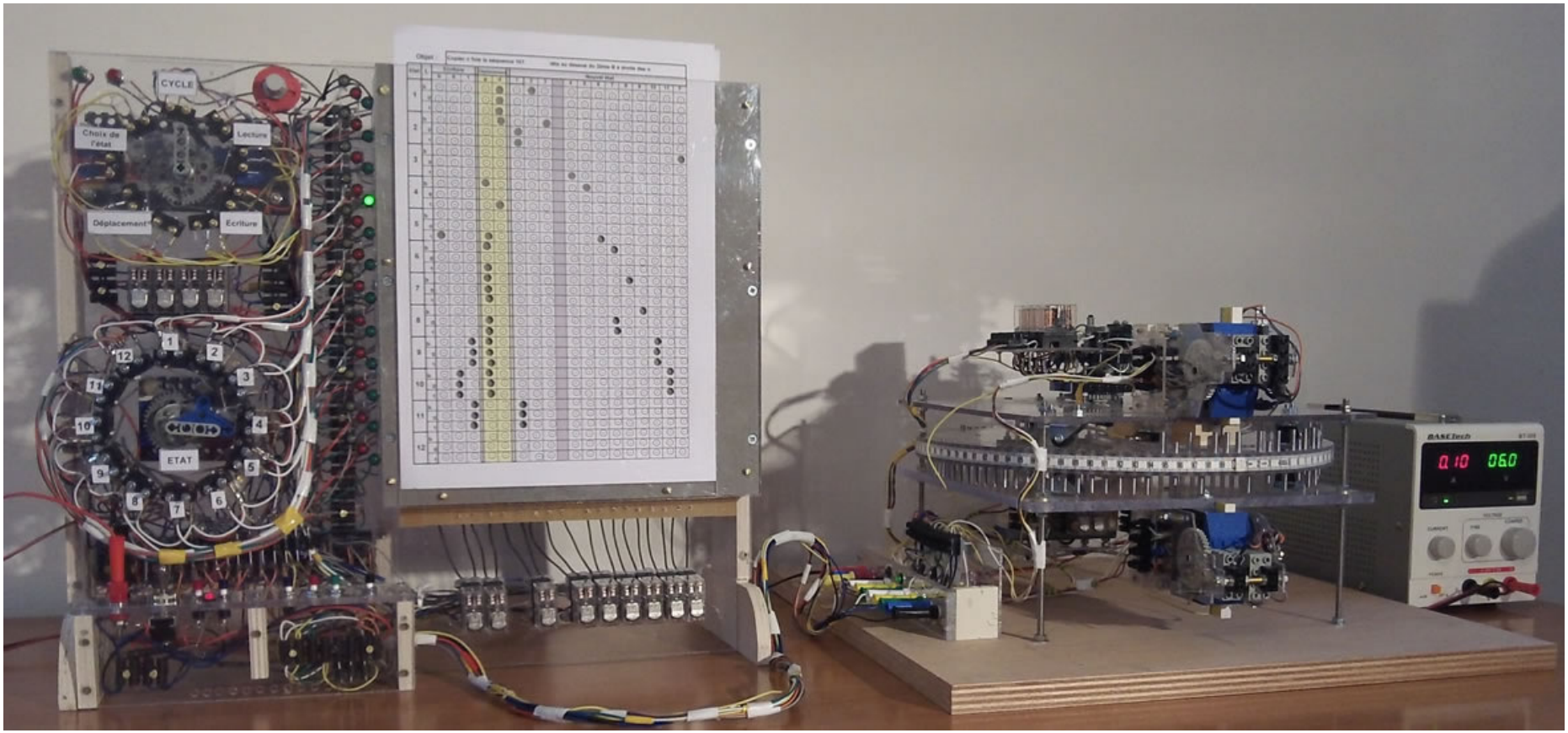
II. As a slightly more difficult example we can construct a machine to compute the sequence 0010110111011111011111 .... The machine is to be capable of five *m*-configurations, viz. "$\mathfrak{o}$", "$\mathfrak{q}$", "$\mathfrak{p}$", "$\mathfrak{f}$", "$\mathfrak{b}$" and of printing "$\mathfrak{d}$", "x", "0", "1". The first three symbols on the tape will be "$\mathfrak{d}\mathfrak{d}0$"; the other figures follow on alternate squares. On the intermediate squares we never print anything but "x". These letters serve to "keep the place" for us and are erased when we have finished with them. We also arrange that in the sequence of figures on alternate squares there shall be no blanks.

Configuration		*Behaviour*	
m-config.	*symbol*	*operations*	*final m-config.*
$\mathfrak{b}$		$P\mathfrak{d}, R, P\mathfrak{o}, R, P0, R, R, P0, L, L$	$\mathfrak{o}$
$\mathfrak{o}$	1	R, Px, L, L, L	$\mathfrak{o}$
	0		$\mathfrak{q}$
$\mathfrak{q}$	Any (0 or 1)	R, R	$\mathfrak{q}$
	None	$P1, L$	$\mathfrak{p}$
$\mathfrak{p}$	x	E, R	$\mathfrak{q}$
	$\mathfrak{d}$	R	$\mathfrak{f}$
	None	L, L	$\mathfrak{p}$
$\mathfrak{f}$	Any	R, R	$\mathfrak{f}$
	None	$P0, L, L$	$\mathfrak{o}$

To illustrate the working of this machine a table is given below of the first few complete configurations. These complete configurations are described by writing down the sequence of symbols which are on the tape,

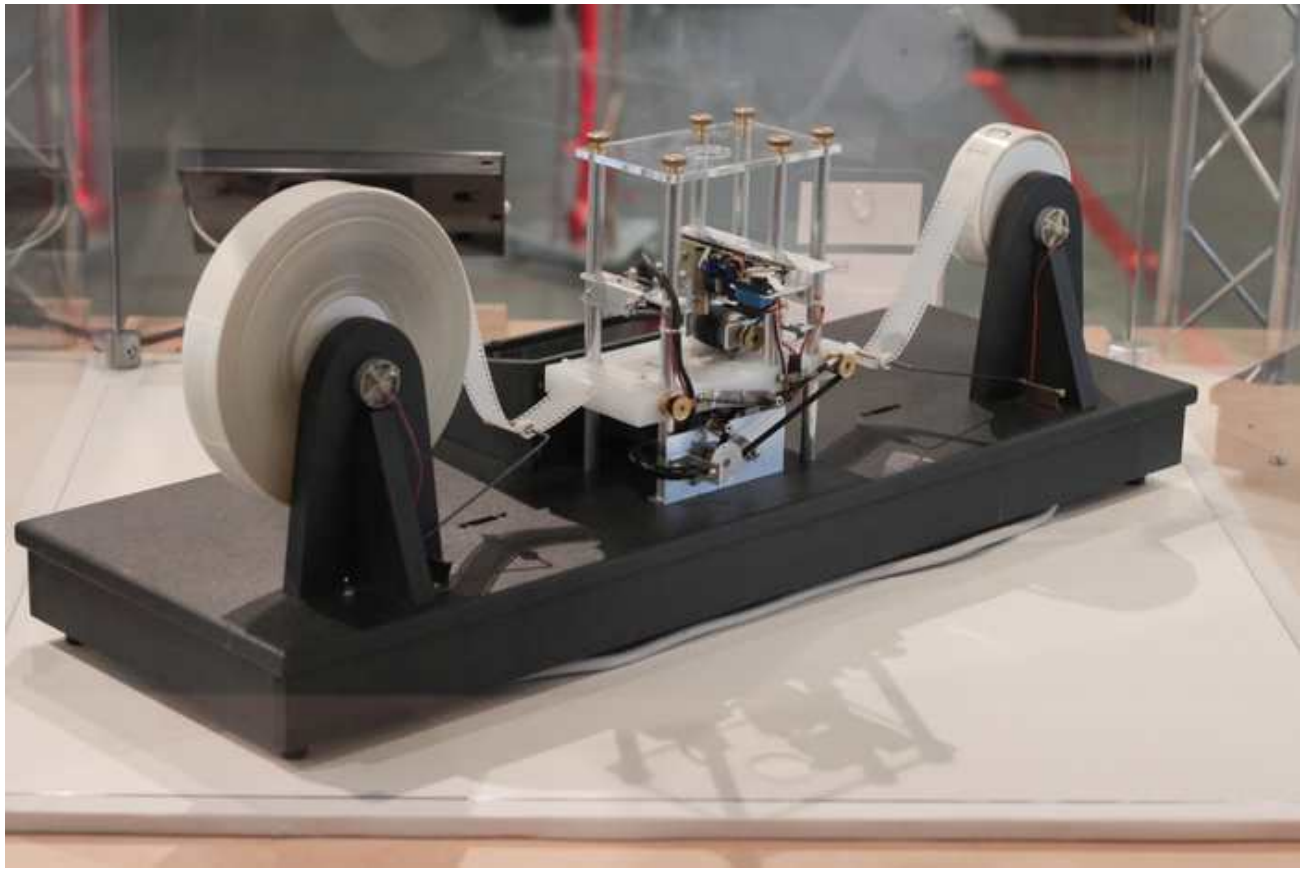
Machines de Turing (I)

Imaginées par Alan Turing en 1936, pour délimiter ce qu'un d'algorithme est capable de faire ou non.



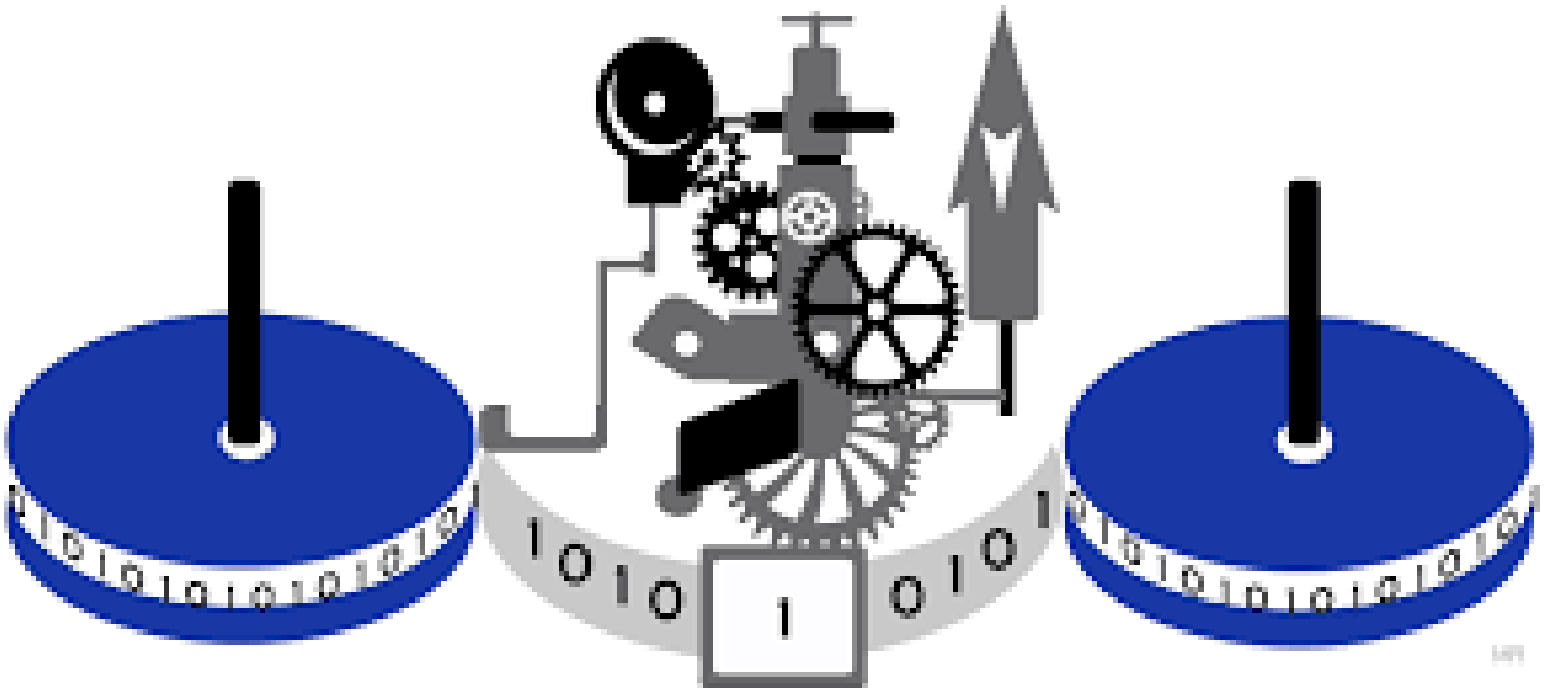
Machines de Turing (I)

Imaginées par Alan Turing en 1936, pour délimiter ce qu'un d'algorithme est capable de faire ou non.



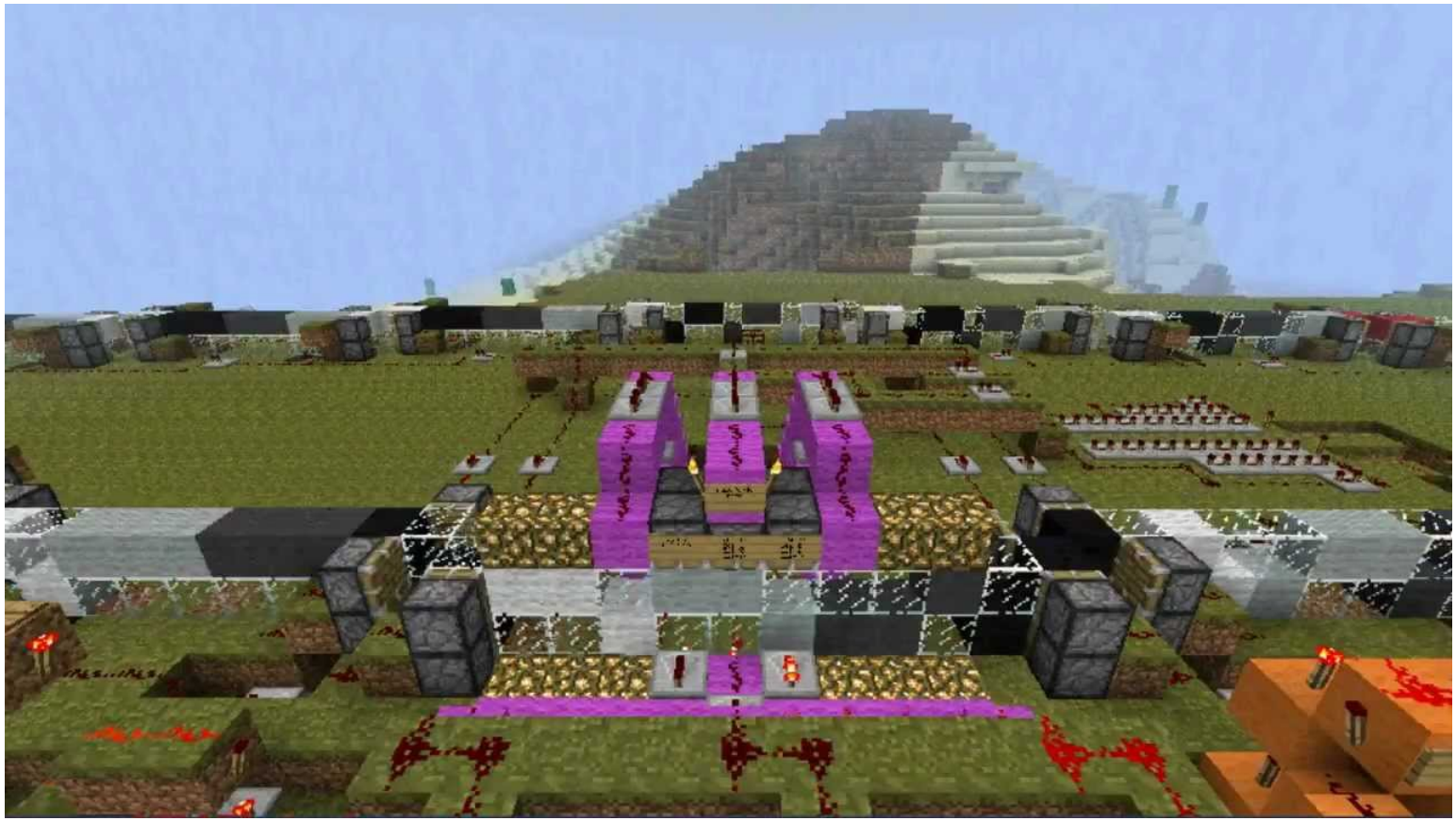
Machines de Turing (I)

Imaginées par Alan Turing en 1936, pour délimiter ce qu'un d'algorithme est capable de faire ou non.



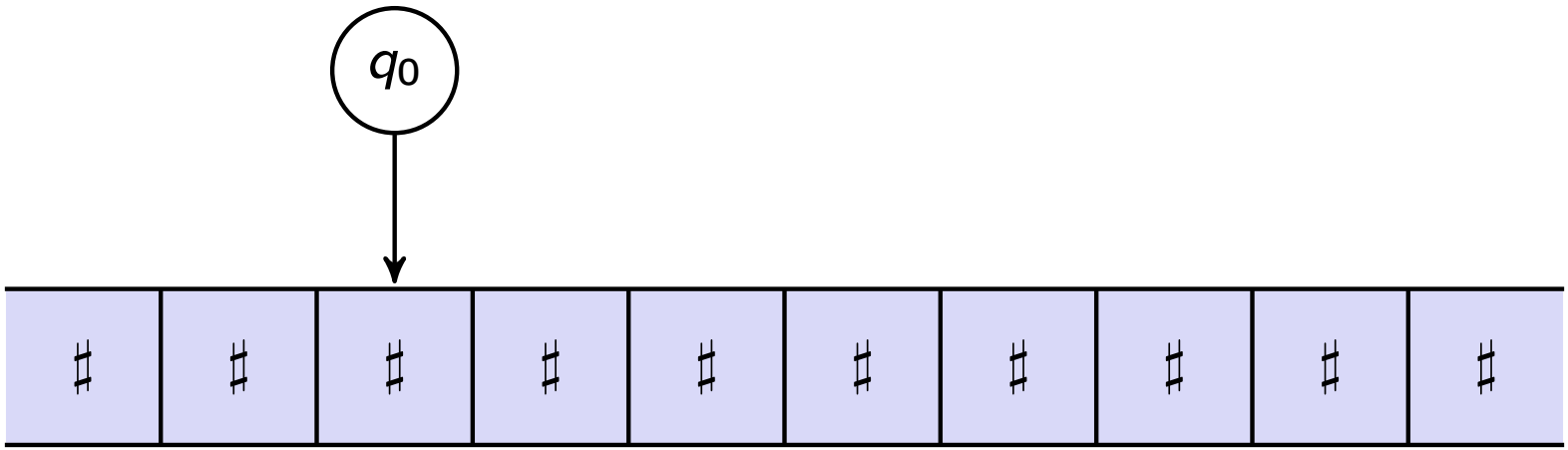
Machines de Turing (I)

Imaginées par Alan Turing en 1936, pour délimiter ce qu'un d'algorithme est capable de faire ou non.



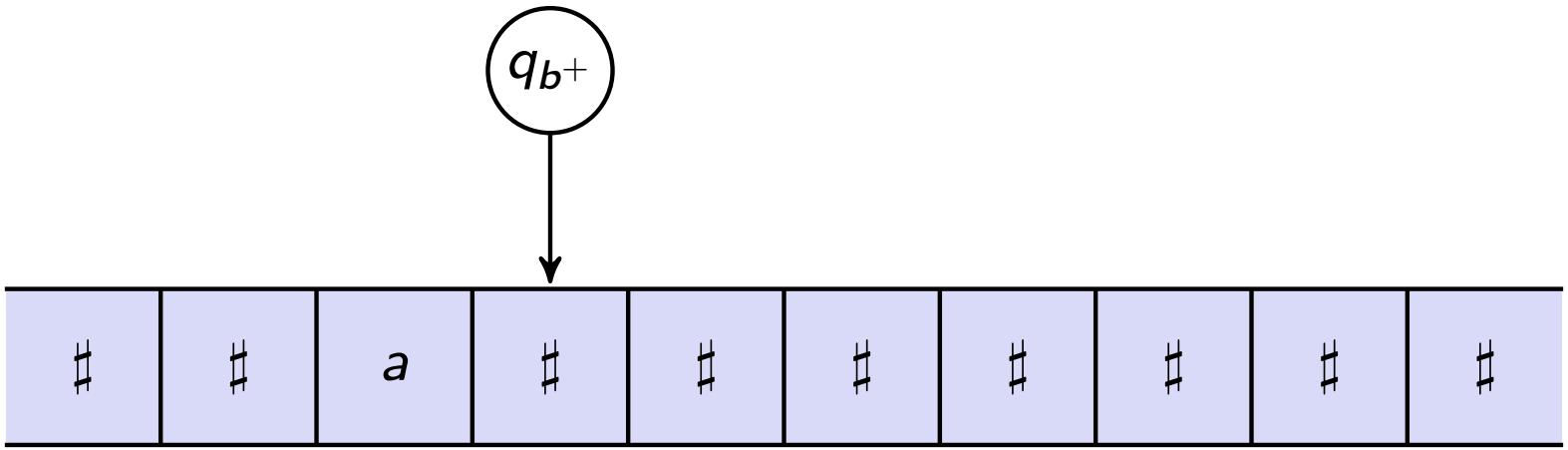
Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



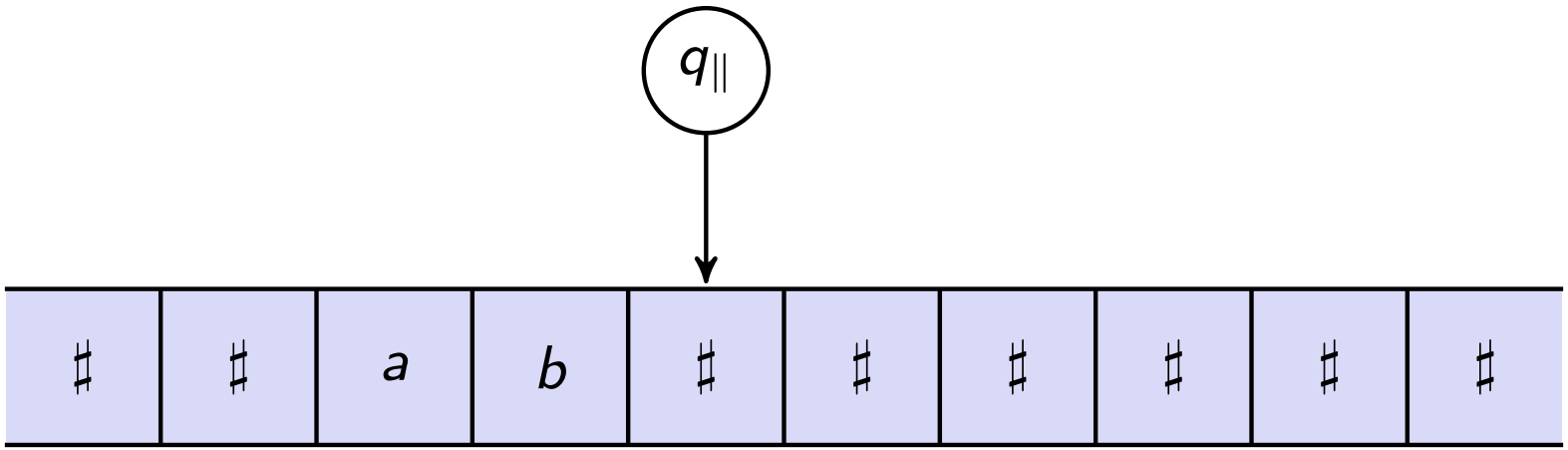
Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



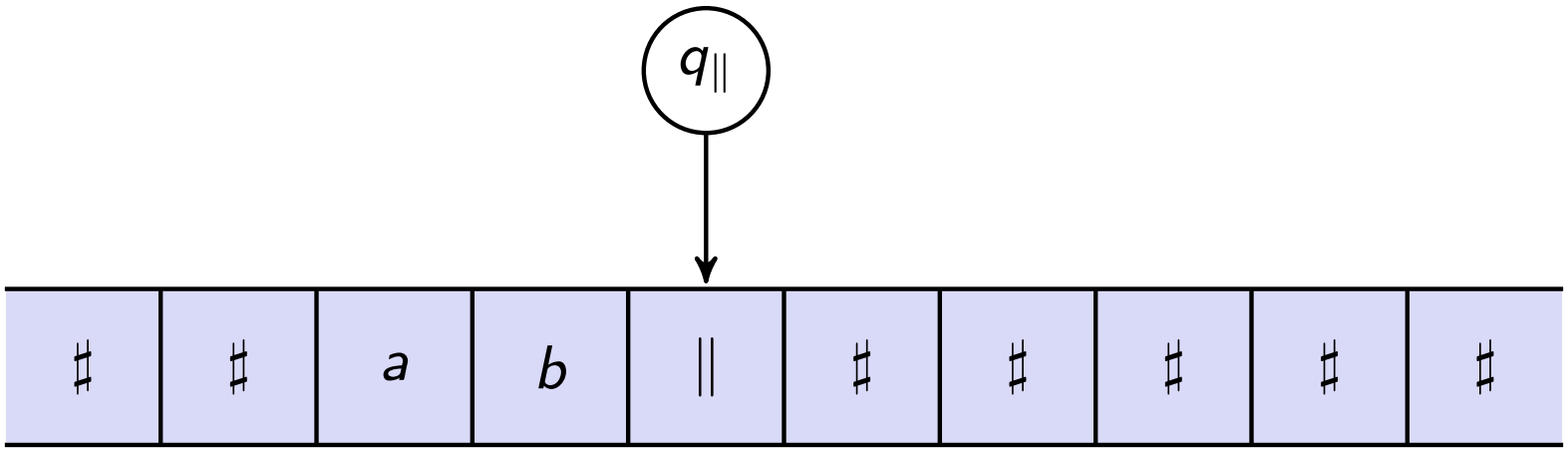
Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



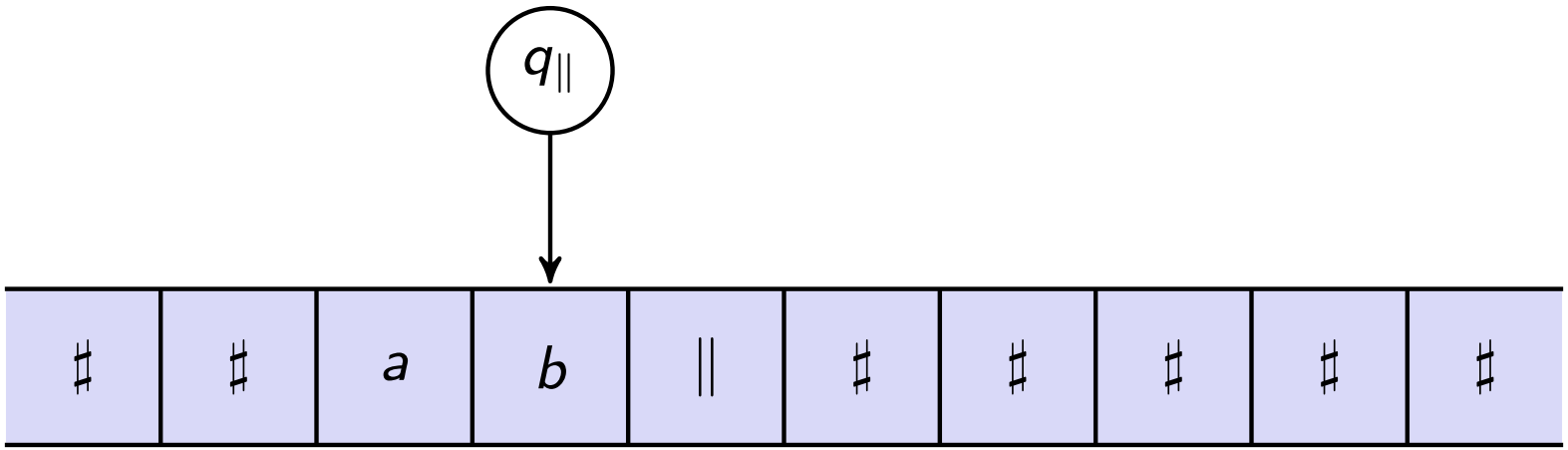
Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



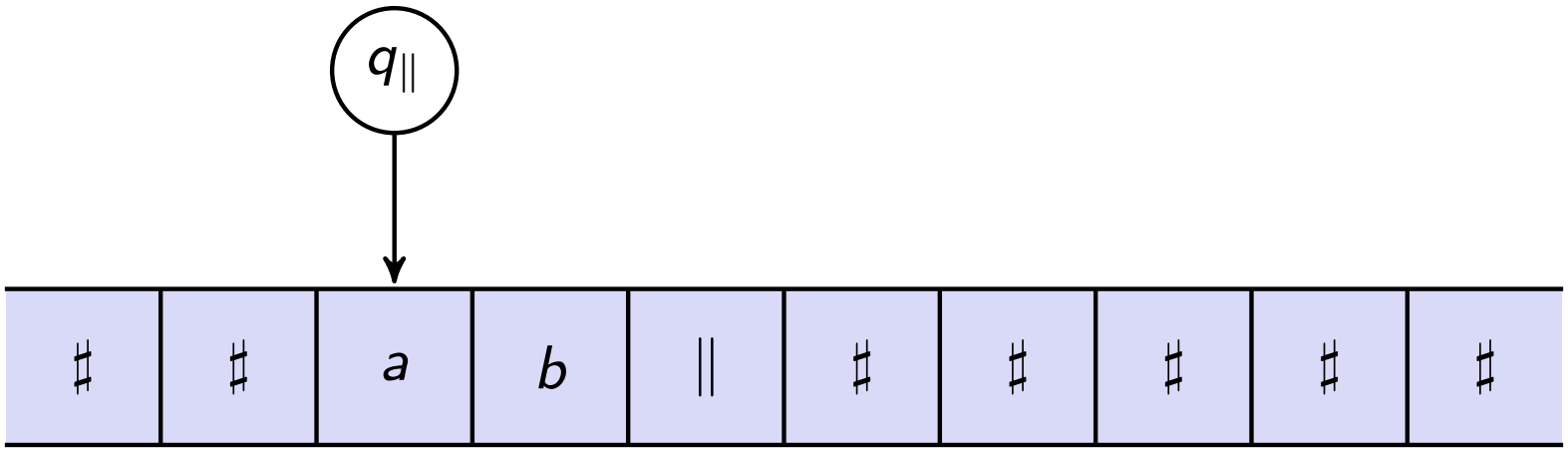
Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



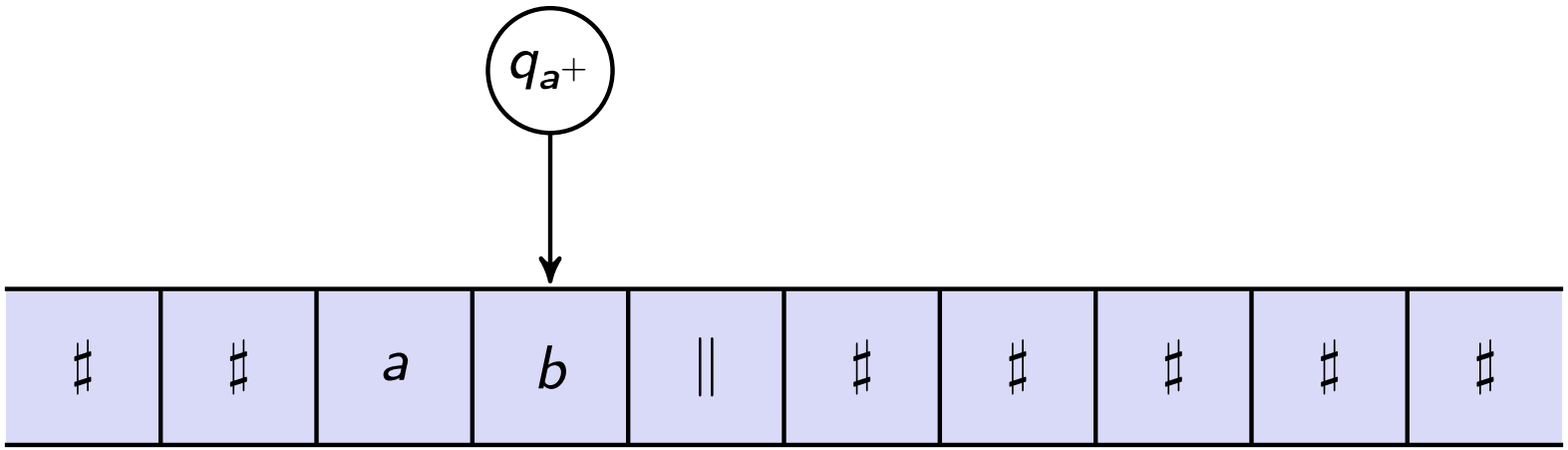
Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



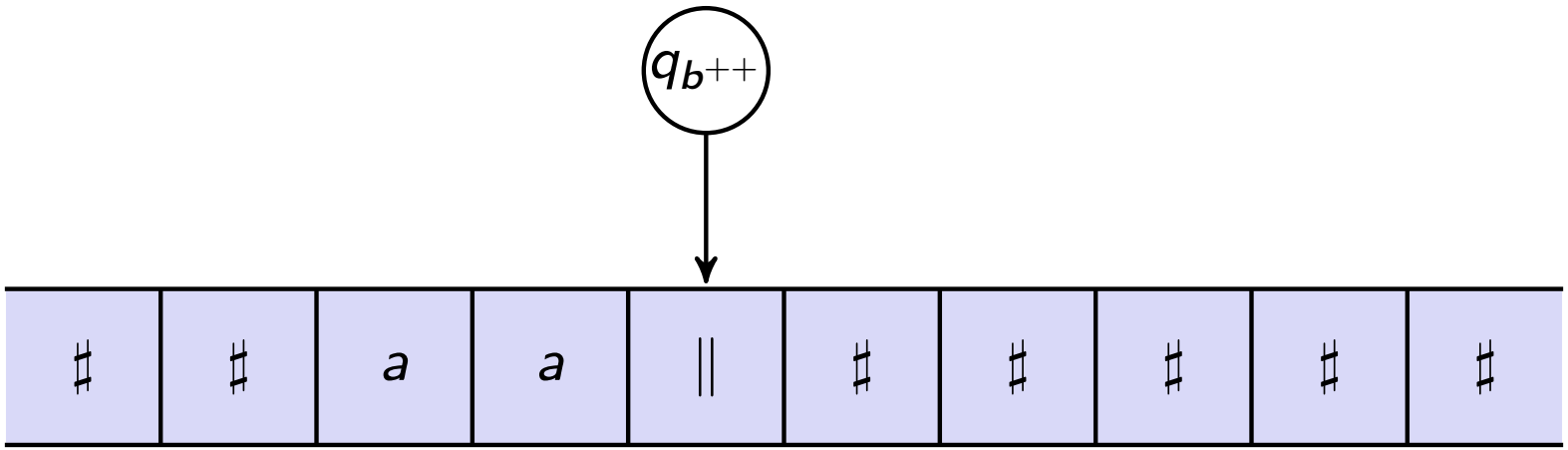
Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



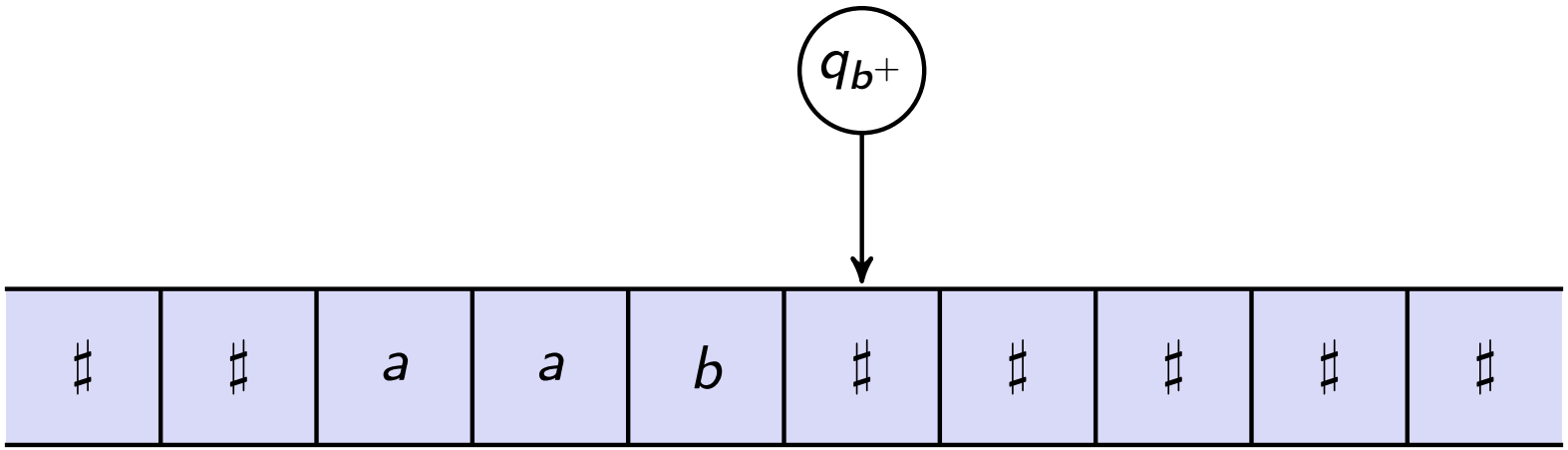
Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



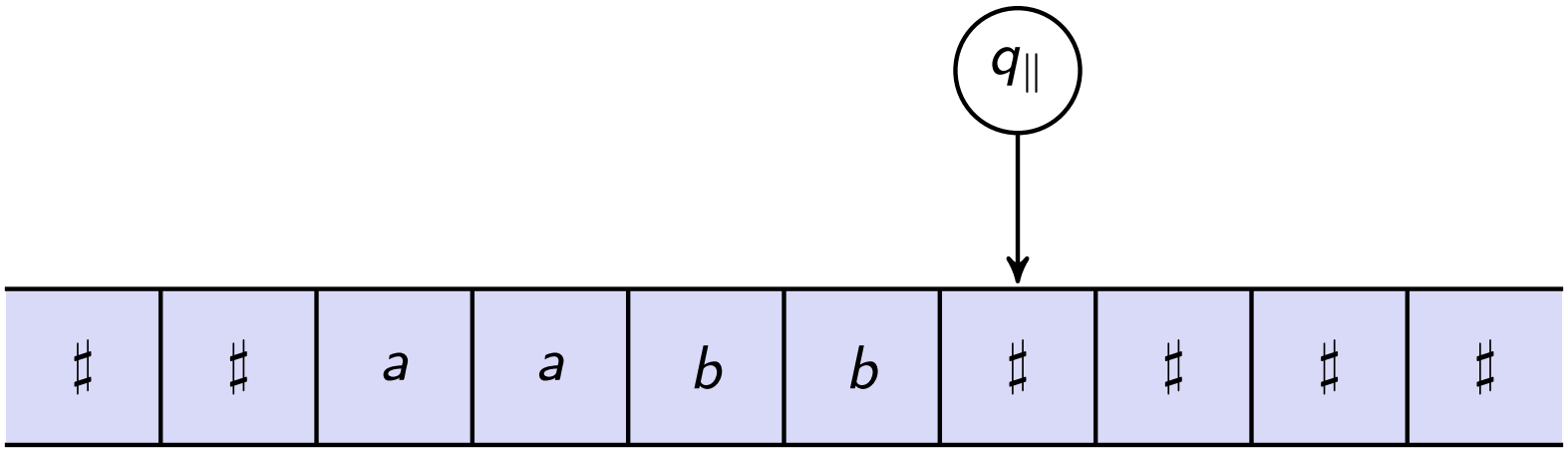
Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



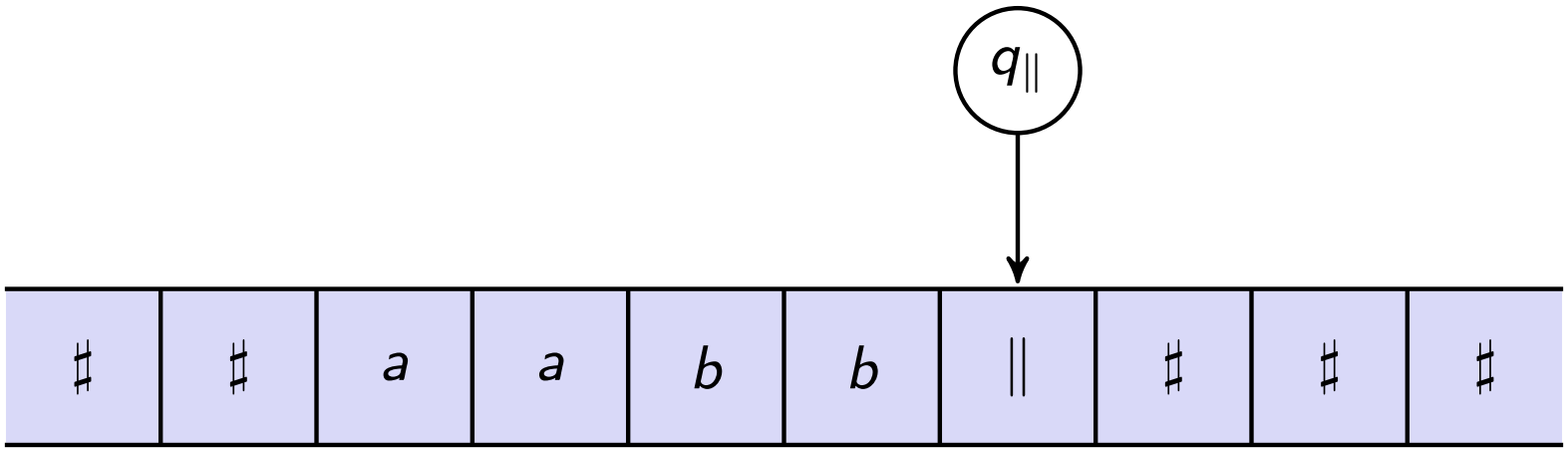
Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



Machines de Turing (II)

$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+}, b, \rightarrow)$	$\perp$
	$q_{\|\|}$	$(q_{a^+}, a, \rightarrow)$	$(q_{\|\|}, b, \leftarrow)$	$(q_{\|\|}, \|\|, \leftarrow)$	$(q_{\|\|}, \|\|, \cdot)$



Machines de Turing (III)

Autre exemple :

$\delta(q,x)$		Symbole x		
		0	1	♯
État q	q_0	$(q_0, 0, \rightarrow)$	$(q_0, 1, \rightarrow)$	$(q_+, 0, \leftarrow)$
	q_+	$(q_+, 0, \leftarrow)$	$(q_+, 1, \leftarrow)$	$(q_f, \sharp, \rightarrow)$
	q_f	$(q_f, 0, .)$	$(q_f, 1, .)$	$(q_f, \sharp, .)$

Machines de Turing (III)

Autre exemple :

$\delta(q,x)$		Symbole x		
		0	1	♯
État q	q_0	$(q_0, 0, \rightarrow)$	$(q_0, 1, \rightarrow)$	$(q_+, 0, \leftarrow)$
	q_+	$(q_+, 0, \leftarrow)$	$(q_+, 1, \leftarrow)$	$(q_f, \sharp, \rightarrow)$
	q_f	$(q_f, 0, .)$	$(q_f, 1, .)$	$(q_f, \sharp, .)$

- L'état q_f est un état particulier appelé **état final**.
- S'il n'y a qu'un nombre fini de 0 et de 1 sur le ruban autour de la tête de lecture, la machine remplace le premier ♯ à droite par un 0 et s'arrête.
- Sinon, la machine ne s'arrête jamais et continue indéfiniment à calculer.

Machines de Turing (IV)

Encore un exemple :

$\delta(q,x)$		Symbole x		
		0	1	♯
État q	q_0	$(q_0, 0, \rightarrow)$	$(q_0, 1, \rightarrow)$	$(q_+, 0, \leftarrow)$
	q_+	$(q_f, 1, .)$	$(q_+, 0, \leftarrow)$	$(q_f, 1, .)$
	q_f	$(q_f, 0, .)$	$(q_f, 1, .)$	$(q_f, \sharp, .)$

Machines de Turing (IV)

Encore un exemple :

$\delta(q,x)$		Symbole x		
		0	1	♯
État q	q_0	$(q_0, 0, \rightarrow)$	$(q_0, 1, \rightarrow)$	$(q_+, 0, \leftarrow)$
	q_+	$(q_f, 1, .)$	$(q_+, 0, \leftarrow)$	$(q_f, 1, .)$
	q_f	$(q_f, 0, .)$	$(q_f, 1, .)$	$(q_f, \sharp, .)$

- S'il n'y a qu'un nombre fini de 0 et de 1 sur le ruban autour de la tête de lecture, la machine ajoute 1 au nombre codé en binaire sur son ruban et s'arrête.
- Sinon, la machine ne s'arrête jamais et continue indéfiniment à calculer.

Machines de Turing (IV)

Encore un exemple :

$\delta(q,x)$		Symbole x		
		0	1	♯
État q	q_0	$(q_0, 0, \rightarrow)$	$(q_0, 1, \rightarrow)$	$(q_0, \sharp, \rightarrow)$
	q_+	$(q_f, 1, .)$	$(q_+, 0, \leftarrow)$	$(q_f, 1, .)$
	q_f	$(q_f, 0, .)$	$(q_f, 1, .)$	$(q_f, \sharp, .)$

Machines de Turing (IV)

Encore un exemple :

$\delta(q,x)$		Symbole x		
		0	1	♯
État q	q_0	$(q_0, 0, \rightarrow)$	$(q_0, 1, \rightarrow)$	$(q_0, \sharp, \rightarrow)$
	q_+	$(q_f, 1, .)$	$(q_+, 0, \leftarrow)$	$(q_f, 1, .)$
	q_f	$(q_f, 0, .)$	$(q_f, 1, .)$	$(q_f, \sharp, .)$

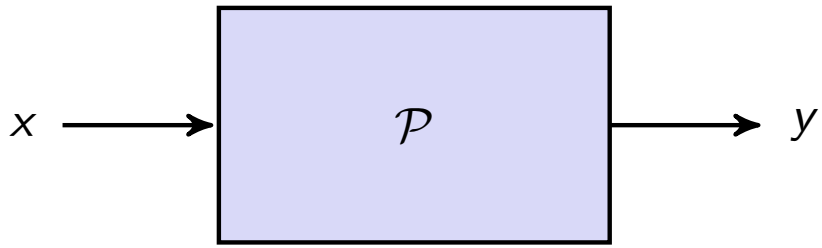
- Quelle que soit l'entrée, la machine boucle !

Machines de Turing (V)

- L'entrée de la machine de Turing est initialement écrite sur le ruban (symboles de l'alphabet et symbole $\sharp$).
- La machine calcule étape par étape en respectant les règles de transition.
- Dès que l'état final est atteint, la machine arrête son calcul et renvoie le résultat écrit sur son ruban.
- Si l'état final n'est jamais atteint, la machine n'arrête pas son calcul : elle **boucle** (à cause de l'entrée ou à cause des règles de transition).

Le problème de l'arrêt (I)

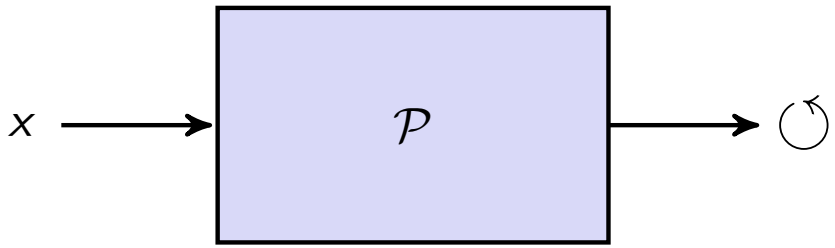
On oublie les machines de Turing, leur ruban, leurs états et leurs transitions. On parle maintenant de **programme**.



Un programme reçoit en **entrée** une information codée sous forme de suite de nombres x, et soit renvoie en **sortie** un résultat lui aussi codé par une suite de nombres y, soit ne s'arrête jamais ↻.

Le problème de l'arrêt (I)

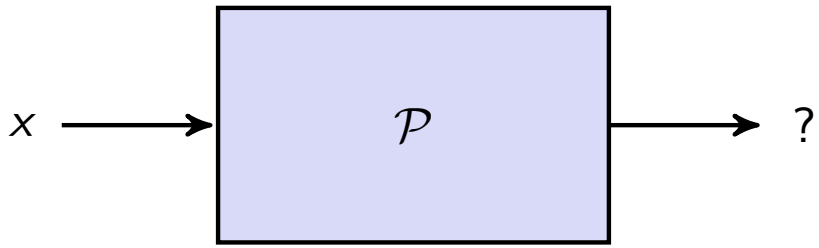
On oublie les machines de Turing, leur ruban, leurs états et leurs transitions. On parle maintenant de **programme**.



Un programme reçoit en **entrée** une information codée sous forme de suite de nombres x, et soit renvoie en **sortie** un résultat lui aussi codé par une suite de nombres y, soit ne s'arrête jamais $\circlearrowleft$.

Le problème de l'arrêt (I)

On oublie les machines de Turing, leur ruban, leurs états et leurs transitions. On parle maintenant de **programme**.



Un programme reçoit en **entrée** une information codée sous forme de suite de nombres x, et soit renvoie en **sortie** un résultat lui aussi codé par une suite de nombres y, soit ne s'arrête jamais $\circlearrowleft$.

Remarque : On peut coder sous forme de suite de nombres n'importe quelle information finie. En particulier, x peut coder un entier... **ou un programme ! !**

Le problème de l'arrêt (II)

Question

Est-ce que le programme $\mathcal{P}$ s'arrête (atteint un état final) sur l'entrée x ?

Le problème de l'arrêt (II)

Question

Est-ce que le programme $\mathcal{P}$ s'arrête (atteint un état final) sur l'entrée x ?

Question

Est-ce que le programme $\mathcal{P}$ s'arrête toujours (sur toutes les entrées possibles) ?

Le problème de l'arrêt (II)

Question

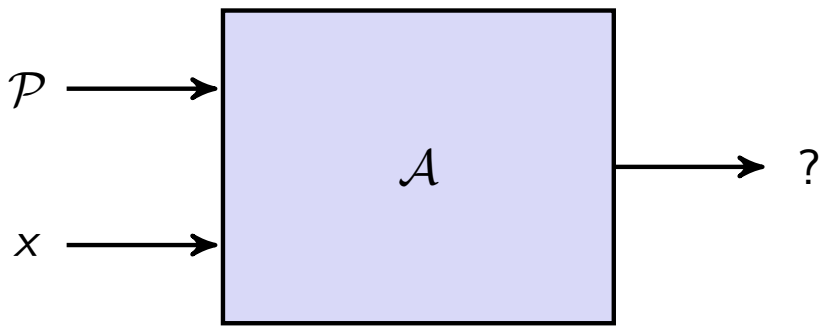
Est-ce que le programme $\mathcal{P}$ s'arrête (atteint un état final) sur l'entrée x ?

Question

Est-ce que le programme $\mathcal{P}$ s'arrête toujours (sur toutes les entrées possibles) ?

Question

Est-ce qu'il existe un super programme $\mathcal{A}$ qui peut me dire si un programme $\mathcal{P}$ s'arrête sur une entrée x ?



Le problème de l'arrêt (II)

Question

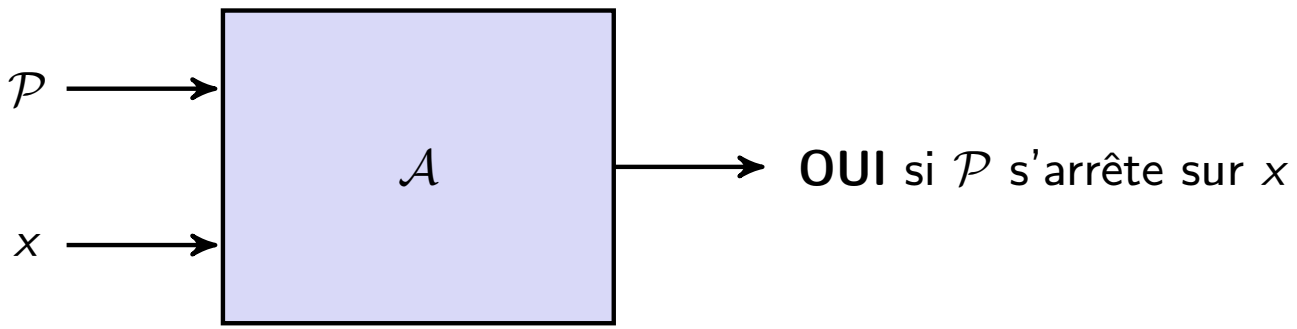
Est-ce que le programme $\mathcal{P}$ s'arrête (atteint un état final) sur l'entrée x ?

Question

Est-ce que le programme $\mathcal{P}$ s'arrête toujours (sur toutes les entrées possibles) ?

Question

Est-ce qu'il existe un super programme $\mathcal{A}$ qui peut me dire si un programme $\mathcal{P}$ s'arrête sur une entrée x ?



Le problème de l'arrêt (II)

Question

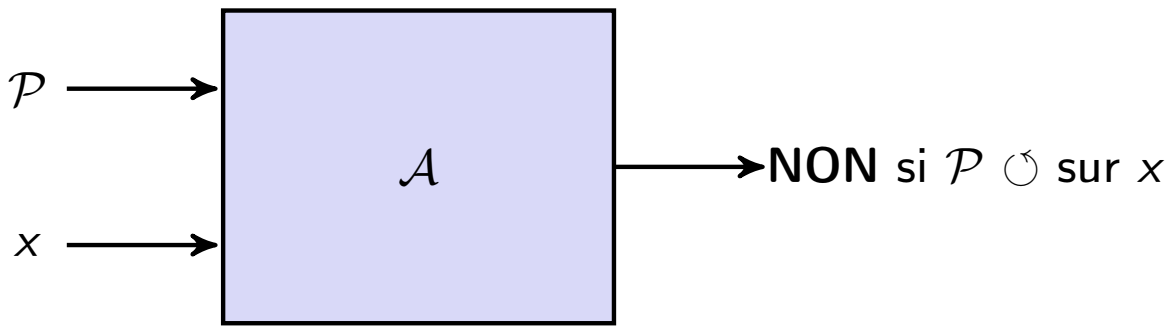
Est-ce que le programme $\mathcal{P}$ s'arrête (atteint un état final) sur l'entrée x ?

Question

Est-ce que le programme $\mathcal{P}$ s'arrête toujours (sur toutes les entrées possibles) ?

Question

Est-ce qu'il existe un super programme $\mathcal{A}$ qui peut me dire si un programme $\mathcal{P}$ s'arrête sur une entrée x ?



Le problème de l'arrêt (III)

On va montrer que le super-programme $\mathcal{A}$ n'existe pas...

Le problème de l'arrêt (III)

On va montrer que le super-programme $\mathcal{A}$ n'existe pas...

Pour cela, on raisonne par **l'absurde** : on suppose que $\mathcal{A}$ existe,...

Le problème de l'arrêt (III)

On va montrer que le super-programme $\mathcal{A}$ n'existe pas...

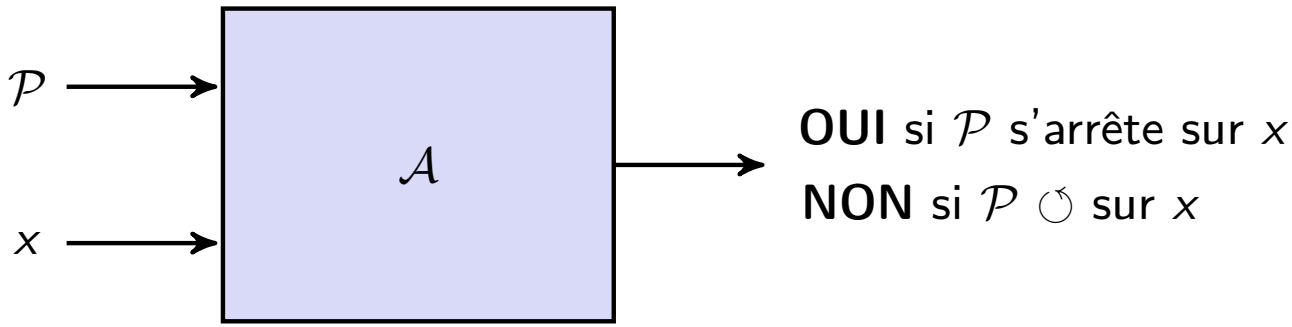
Pour cela, on raisonne par **l'absurde** : on suppose que $\mathcal{A}$ existe,... et on aboutit à une contradiction !

Le problème de l'arrêt (III)

On va montrer que le super-programme $\mathcal{A}$ n'existe pas...

Pour cela, on raisonne par **l'absurde** : on suppose que $\mathcal{A}$ existe,... et on aboutit à une contradiction !

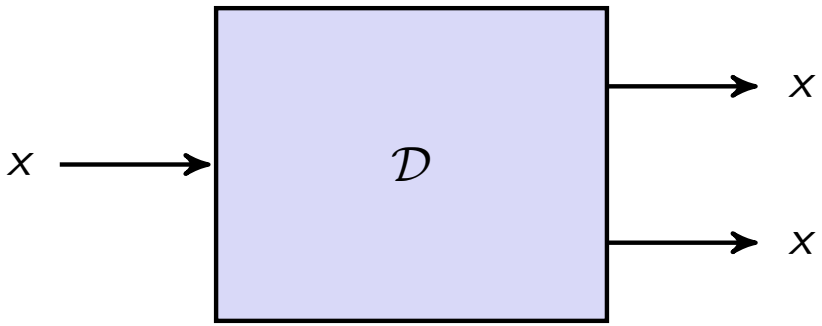
On suppose donc qu'on a notre super-programme $\mathcal{A}$

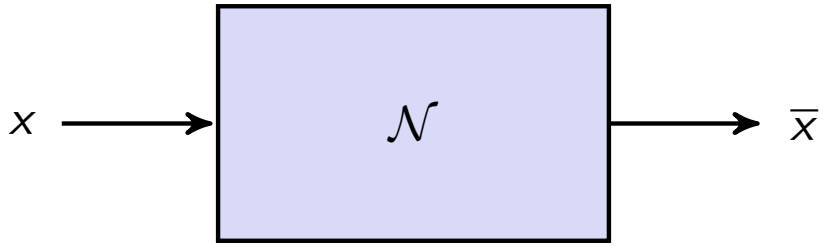


et que $\mathcal{A}$ est bien un super-programme, c'est-à-dire qu'il ne se trompe jamais !

Le problème de l'arrêt (IV)

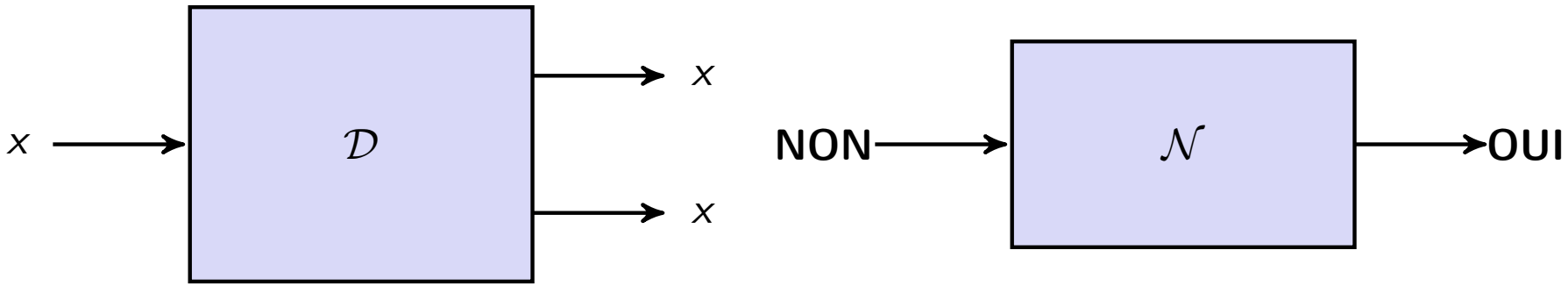
On utilise deux programmes auxiliaires $\mathcal{D}$ et $\mathcal{N}$





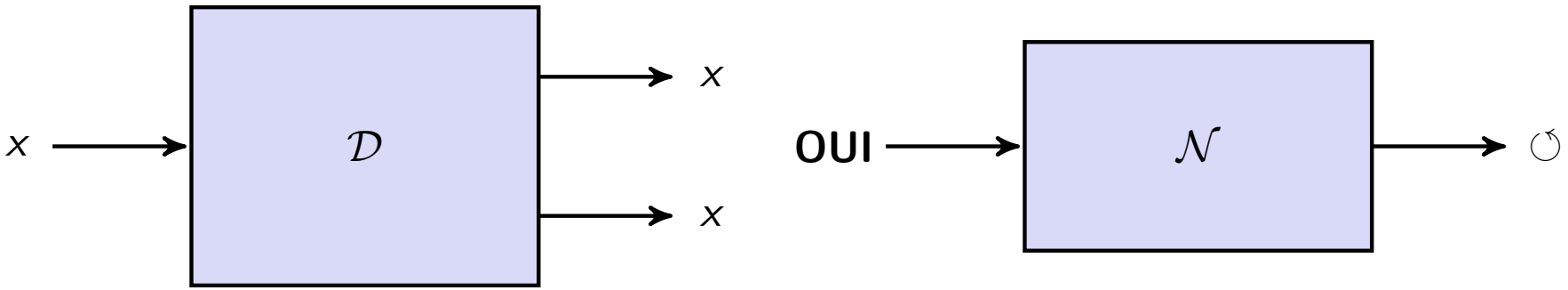
Le problème de l'arrêt (IV)

On utilise deux programmes auxiliaires $\mathcal{D}$ et $\mathcal{N}$



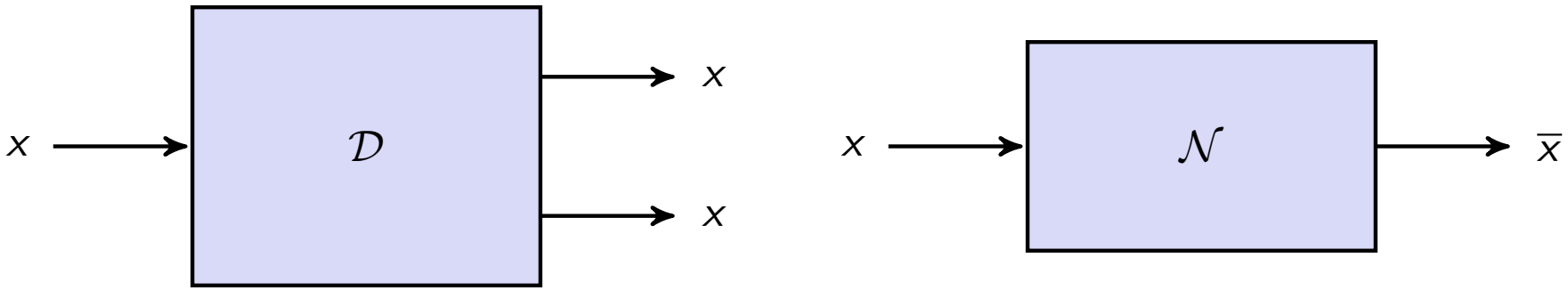
Le problème de l'arrêt (IV)

On utilise deux programmes auxiliaires $\mathcal{D}$ et $\mathcal{N}$

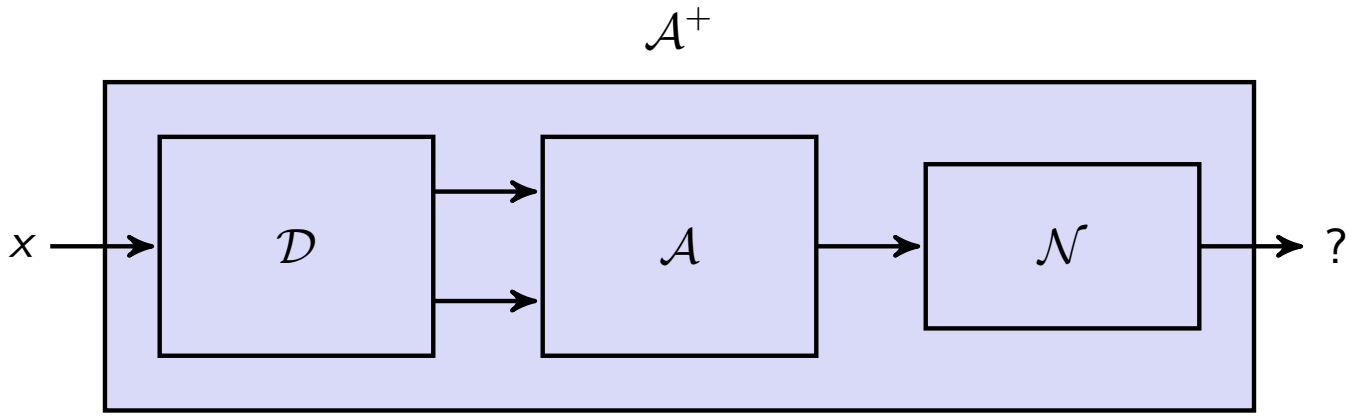


Le problème de l'arrêt (IV)

On utilise deux programmes auxiliaires $\mathcal{D}$ et $\mathcal{N}$

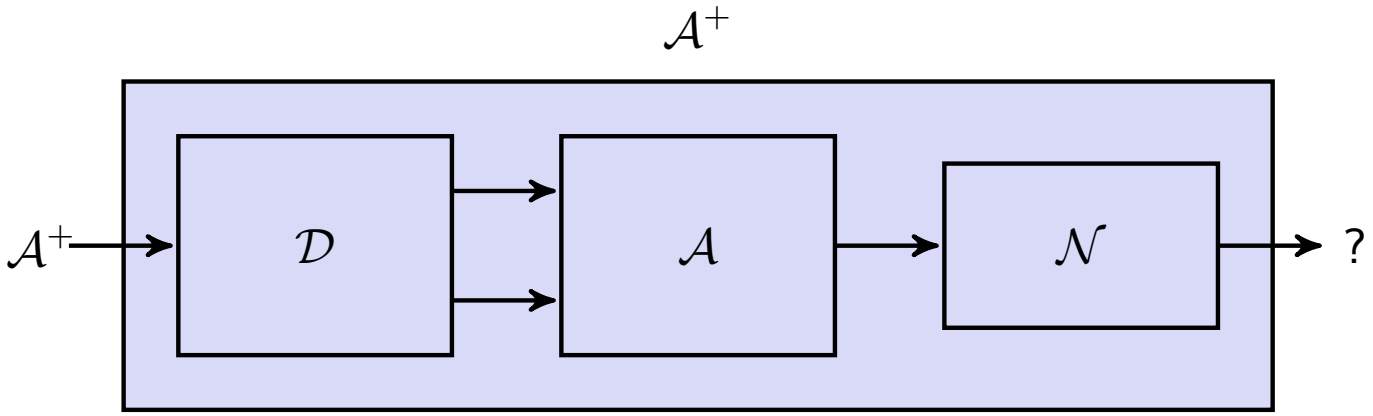


que l'on combine avec $\mathcal{A}$ pour construire le programme $\mathcal{A}^+$.



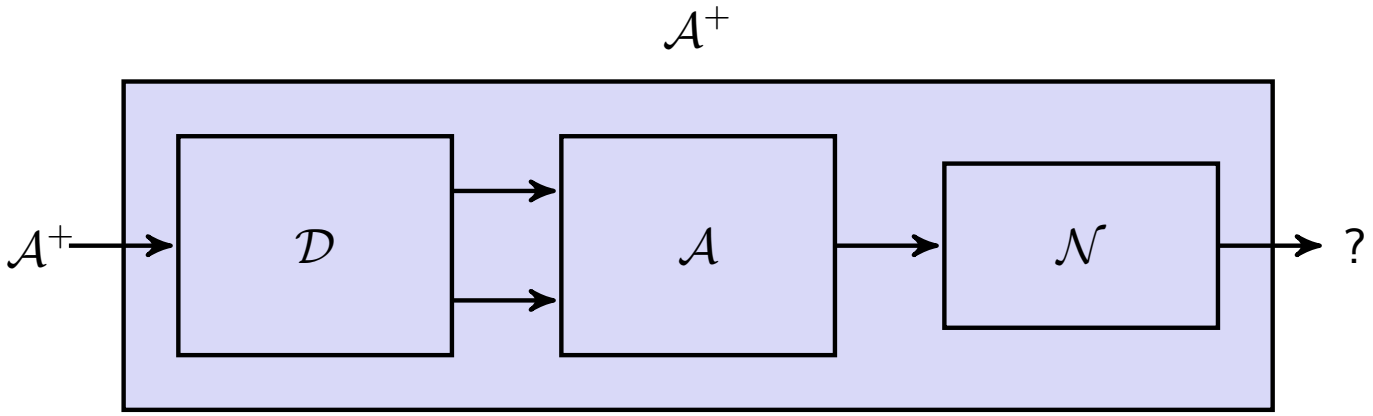
Le problème de l'arrêt (V)

On donne en entrée $\mathcal{A}^+$ à notre programme $\mathcal{A}^+$. Que se passe-t-il ?



Le problème de l'arrêt (V)

On donne en entrée $\mathcal{A}^+$ à notre programme $\mathcal{A}^+$. Que se passe-t-il ?

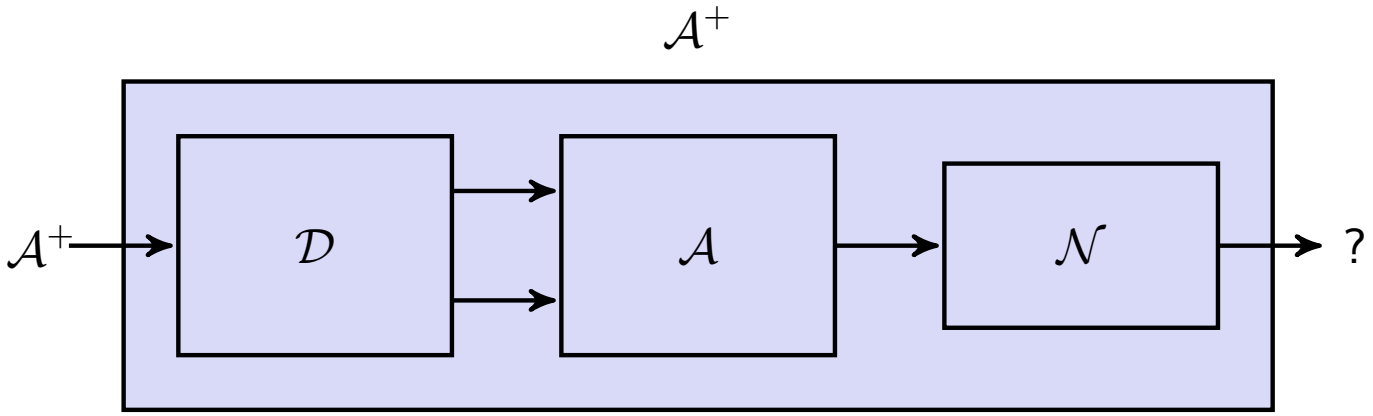


Conclusion :

- Si le programme $\mathcal{A}^+$ répond **OUI**, cela signifie que le programme $\mathcal{A}^+$ boucle.
- Mais si le programme $\mathcal{A}^+$ boucle, cela signifie le programme $\mathcal{A}^+$ répond **OUI**.
- ...

Le problème de l'arrêt (V)

On donne en entrée $\mathcal{A}^+$ à notre programme $\mathcal{A}^+$. Que se passe-t-il ?



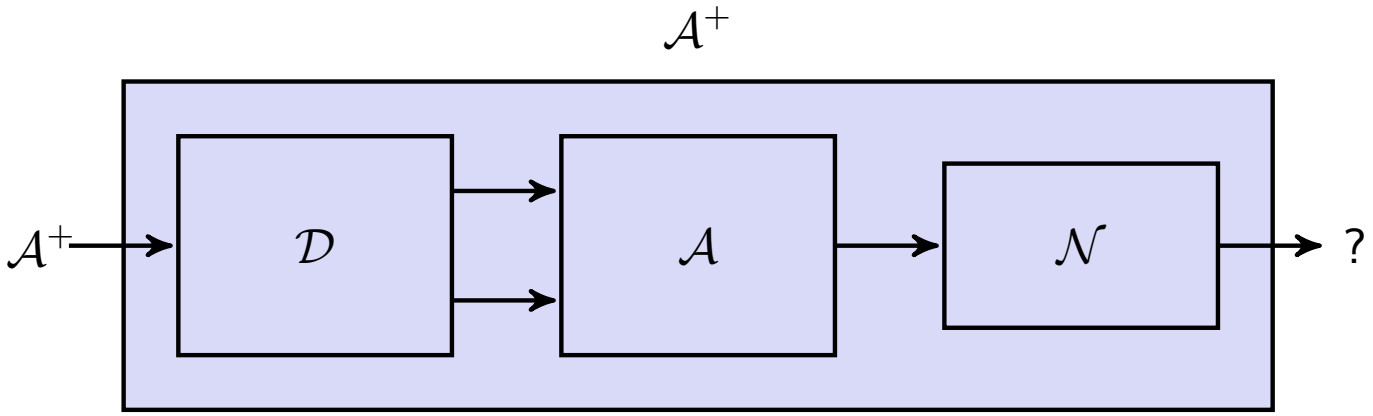
Conclusion :

- Si le programme $\mathcal{A}^+$ répond **OUI**, cela signifie que le programme $\mathcal{A}^+$ boucle.
- Mais si le programme $\mathcal{A}^+$ boucle, cela signifie le programme $\mathcal{A}^+$ répond **OUI**.
- ...

$\Rightarrow$ contradiction ! ! le super-programme $\mathcal{A}$ ne peut pas exister.

Le problème de l'arrêt (V)

On donne en entrée $\mathcal{A}^+$ à notre programme $\mathcal{A}^+$. Que se passe-t-il ?



Conclusion :

- Si le programme $\mathcal{A}^+$ répond **OUI**, cela signifie que le programme $\mathcal{A}^+$ boucle.
- Mais si le programme $\mathcal{A}^+$ boucle, cela signifie le programme $\mathcal{A}^+$ répond **OUI**.
- ...

$\Rightarrow$ contradiction ! ! le super-programme $\mathcal{A}$ ne peut pas exister.

Théorème (Turing, 1936)

Il n'existe pas de programme/d'algorithme qui résout le problème de l'arrêt. On dit que le problème de l'arrêt est **indécidable**.

Comment créer ce type de « monstres » mathématiques ?

On construit un objet, et on le donne à manger à lui-même.

- Le barbier rase tous ceux qui ne se rasent pas eux-mêmes. Qui rase le barbier ?
- L'ensemble de tous les ensembles qui ne se contiennent pas à eux-mêmes se contient-il lui-même ? (Russell)
- ...

Comment créer ce type de « monstres » mathématiques ?

On construit un objet, et on le donne à manger à lui-même.

- Le barbier rase tous ceux qui ne se rasent pas eux-mêmes. Qui rase le barbier ?
- L'ensemble de tous les ensembles qui ne se contiennent pas à eux-mêmes se contient-il lui-même ? (Russell)
- ...

Le raisonnement « tourne en rond » à l'infini $\Rightarrow$ **paradoxe**.

Comment créer ce type de « monstres » mathématiques ?

On construit un objet, et on le donne à manger à lui-même.

- Le barbier rase tous ceux qui ne se rasent pas eux-mêmes. Qui rase le barbier ?
- L'ensemble de tous les ensembles qui ne se contiennent pas à eux-mêmes se contient-il lui-même ? (Russell)
- ...

Le raisonnement « tourne en rond » à l'infini $\Rightarrow$ **paradoxe**.

On parle d'**argument diagonal** (Cantor, 1891).

Comment montrer qu'un problème est indécidable ?

Valable pour les problèmes de **décision** (ceux pour lesquels la réponse ne peut être que **OUI** ou **NON**).

Un problème est **indécidable** s'il ne peut pas exister d'algorithme/programme qui le résout (qui donne la bonne réponse quelle que soit l'entrée).

Comment montrer qu'un problème est indécidable ?

Valable pour les problèmes de **décision** (ceux pour lesquels la réponse ne peut être que **OUI** ou **NON**).

Un problème est **indécidable** s'il ne peut pas exister d'algorithme/programme qui le résout (qui donne la bonne réponse quelle que soit l'entrée).

Pour montrer qu'un problème P est indécidable, une stratégie de démonstration est :

1. On suppose que l'on sait résoudre le problème P à l'aide d'un programme $\mathcal{P}$.
2. On transforme une entrée du problème de l'arrêt en entrée du problème P. La transformation est telle que les réponses **OUI**/**NON** restent inchangées.
3. Pour résoudre le problème de l'arrêt, il suffirait donc de transformer l'entrée et d'appliquer $\mathcal{P}$.
4. On conclut : comme le programme $\mathcal{P}$ ne peut pas exister, alors P est indécidable.

Comment montrer qu'un problème est indécidable ?

Valable pour les problèmes de **décision** (ceux pour lesquels la réponse ne peut être que **OUI** ou **NON**).

Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Comment montrer qu'un problème est indécidable ?

Valable pour les problèmes de **décision** (ceux pour lesquels la réponse ne peut être que **OUI** ou **NON**).

Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Un problème est **indécidable** s'il ne peut pas exister d'algorithme/programme qui le résout (qui donne la bonne réponse quelle que soit l'entrée).

Comment montrer qu'un problème est indécidable ?

Valable pour les problèmes de **décision** (ceux pour lesquels la réponse ne peut être que **OUI** ou **NON**).

Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Un problème est **indécidable** s'il ne peut pas exister d'algorithme/programme qui le résout (qui donne la bonne réponse quelle que soit l'entrée).

Pour montrer qu'un problème P est indécidable, une stratégie de démonstration est :

1. On suppose que l'on sait résoudre le problème P à l'aide d'un programme $\mathcal{P}$.
2. On transforme une entrée du problème de l'arrêt en entrée du problème P. La transformation est telle que les réponses **OUI**/**NON** restent inchangées.
3. Pour résoudre le problème de l'arrêt, il suffirait donc de transformer l'entrée et d'appliquer $\mathcal{P}$.
4. On conclut : comme le programme $\mathcal{P}$ ne peut pas exister, alors P est indécidable.

Lien entre les deux problèmes

Conjecture (Wang, 1961)

Si un jeu de tuiles de Wang pave le plan, alors il peut le faire de manière périodique.

Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Lien entre les deux problèmes

Conjecture (Wang, 1961)

Si un jeu de tuiles de Wang pave le plan, alors il peut le faire de manière périodique.

Problème du Domino

Entrée : Un jeu de tuiles τ
Sortie : Oui s'il existe un pavage valide par τ, **Non** sinon.

Supposons que la conjecture de Wang est vraie. Alors on peut résoudre le problème du Domino !

Semi-algorithme 1 :

1. renvoie un motif qui pave le plan de manière périodique s'il existe,
2. boucle sinon.

Semi-algorithme 2 :

1. renvoie un entier n tel que le carré $[1; n] \times [1; n]$ ne peut pas être pavé, s'il existe,
2. boucle sinon.

Pour aller plus loin...

Deux ateliers :

1. Autour du problème du Domino.
2. A propos de jeux de tuiles apériodiques.

Pour aller plus loin...

Deux ateliers :

1. Autour du problème du Domino.
2. A propos de jeux de tuiles apériodiques.

Merci ! !