Les mosaïques, ces ordinateurs qui s'ignorent

Séminaire mathématiques et société

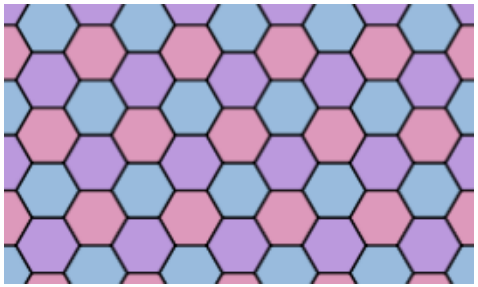
Nathalie AUBRUN

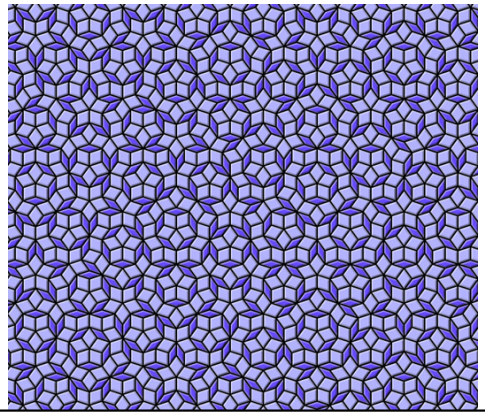
LIP, ENS de Lyon

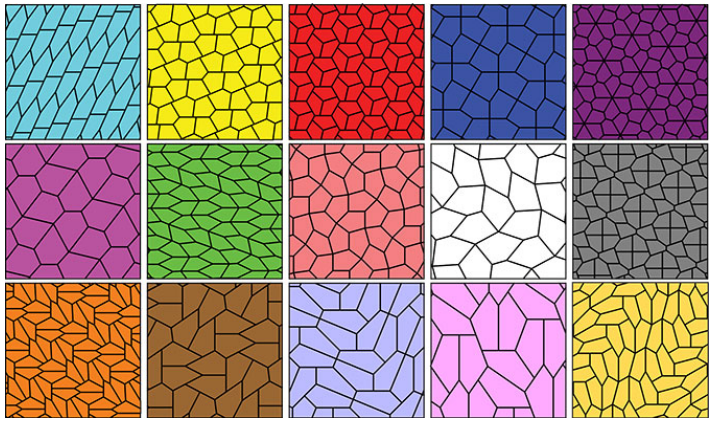
Mercredi 24 mai 2017

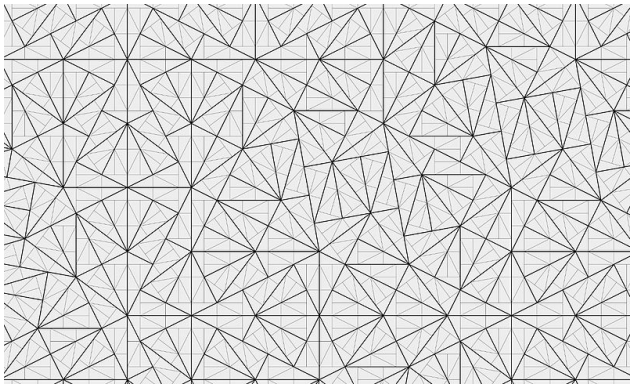
Les mosaïques, qu'est-ce que c'est ?

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



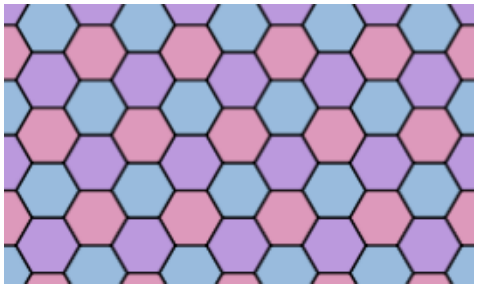


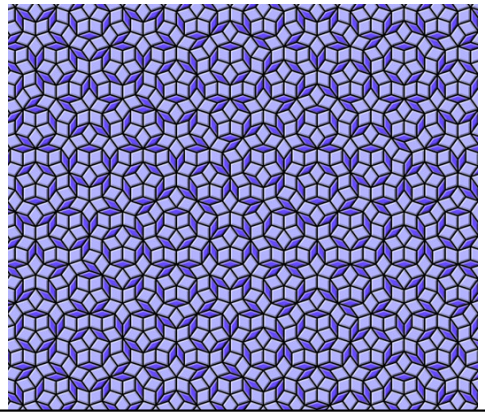


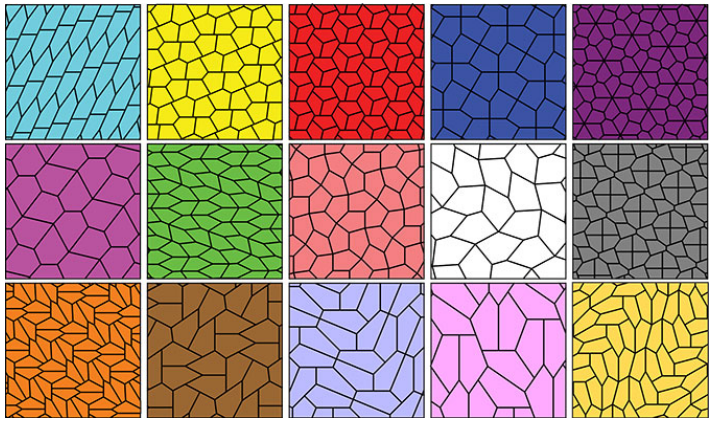


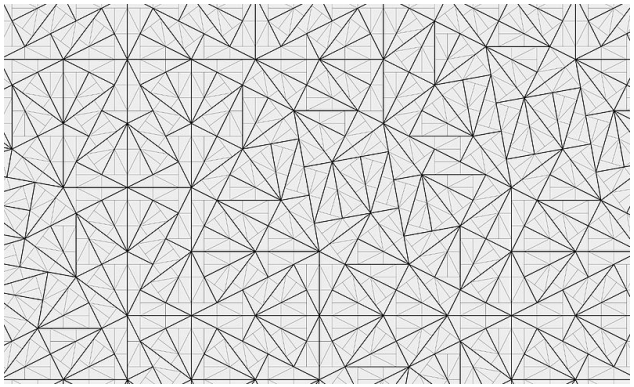
Les mosaïques, qu'est-ce que c'est ?

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









Dans la suite, on parlera de **pavages**.

Pour quoi faire ?

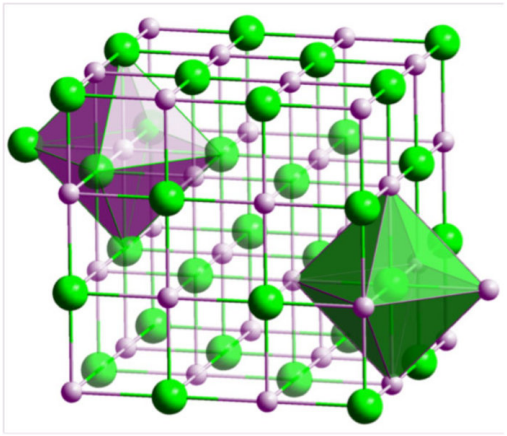
- Ornementer des bâtiments (alhambra de Grenade)
  (17 types de pavages périodiques du plan)

Pour quoi faire ?

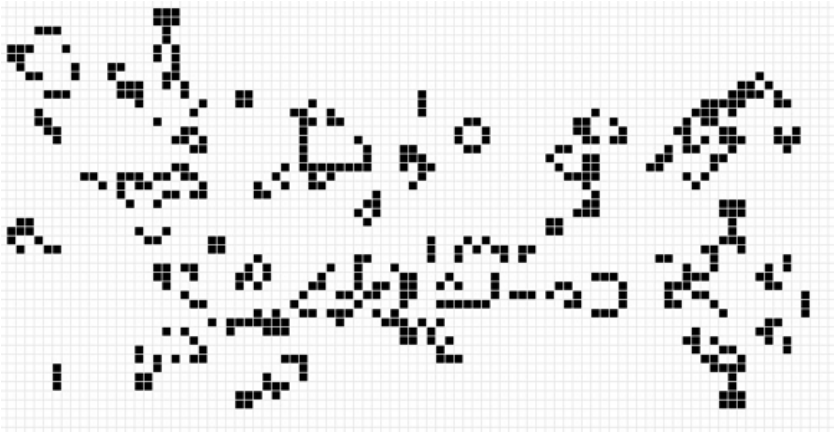
- Paver des rues (Helsinki)
  (pavage de Penrose)

Pour quoi faire ?

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



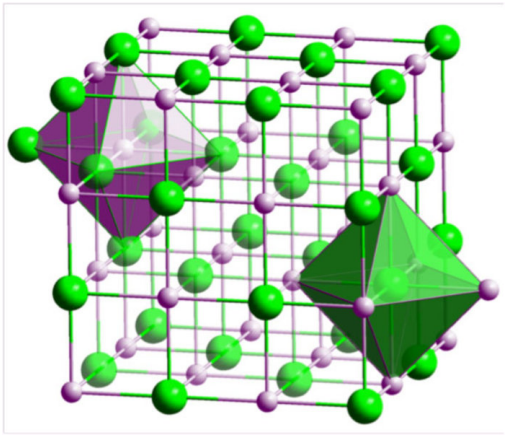
- Modélisation de phénomènes décrits localement (Automates Cellulaires)



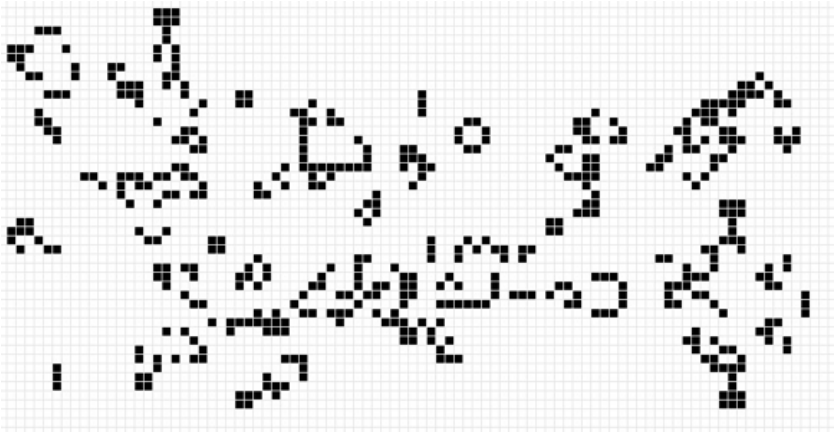
- Définir une manière de réaliser des calculs : tuiles de Wang

Pour quoi faire ?

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

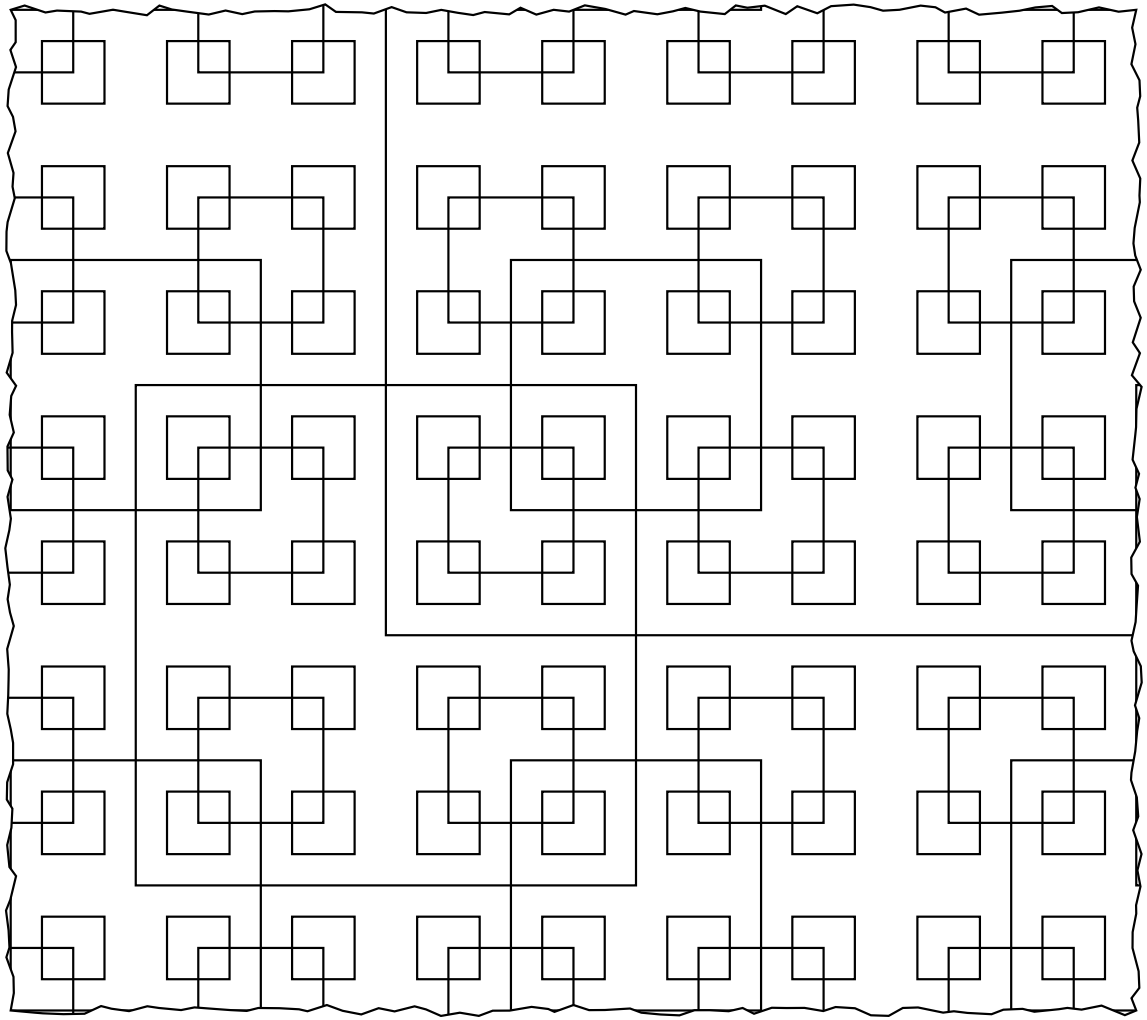


- Modélisation de phénomènes décrits localement (Automates Cellulaires)

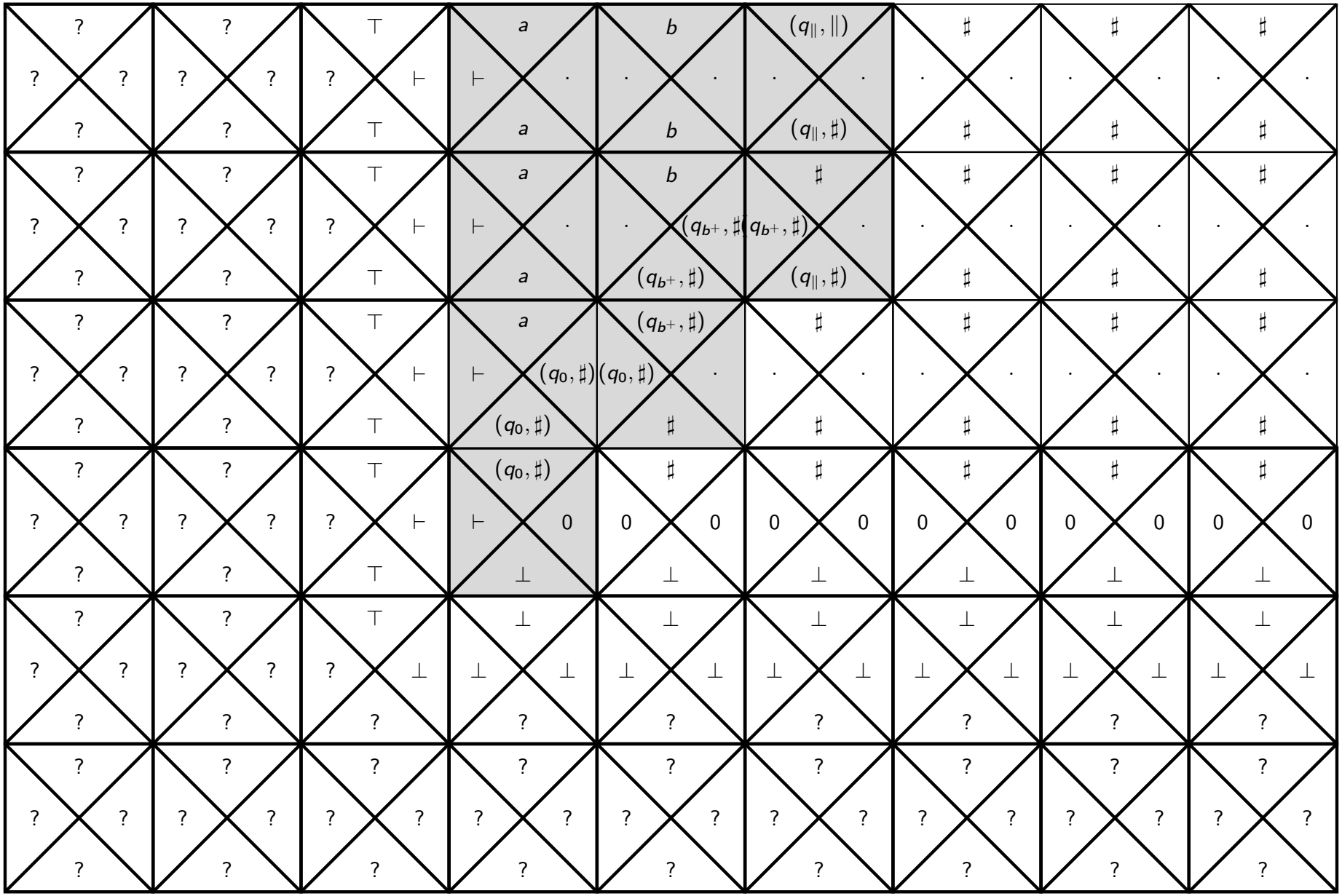


- **Définir une manière de réaliser des calculs : tuiles de Wang**

Comment construire des pavages compliqués avec peu de tuiles ?



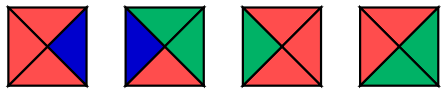
Comment réaliser des calculs dans un pavage ?



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,

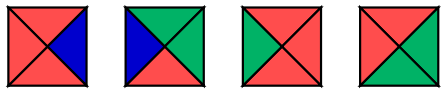


- où les tuiles sont placées sommet contre sommet,
- où deux tuiles voisines portent la même couleur sur leur arête commune.

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,



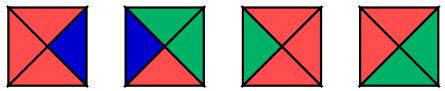
- où les tuiles sont placées sommet contre sommet,
- où deux tuiles voisines portent la même couleur sur leur arête commune.

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,



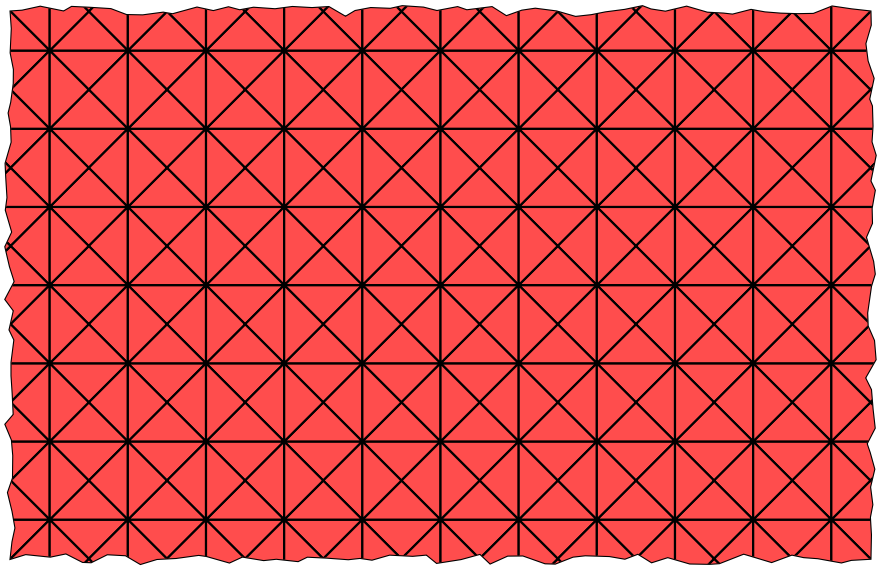
- où les tuiles sont placées sommet contre sommet,
- où deux tuiles voisines portent la même couleur sur leur arête commune.

Interdiction de tourner les tuiles ! !

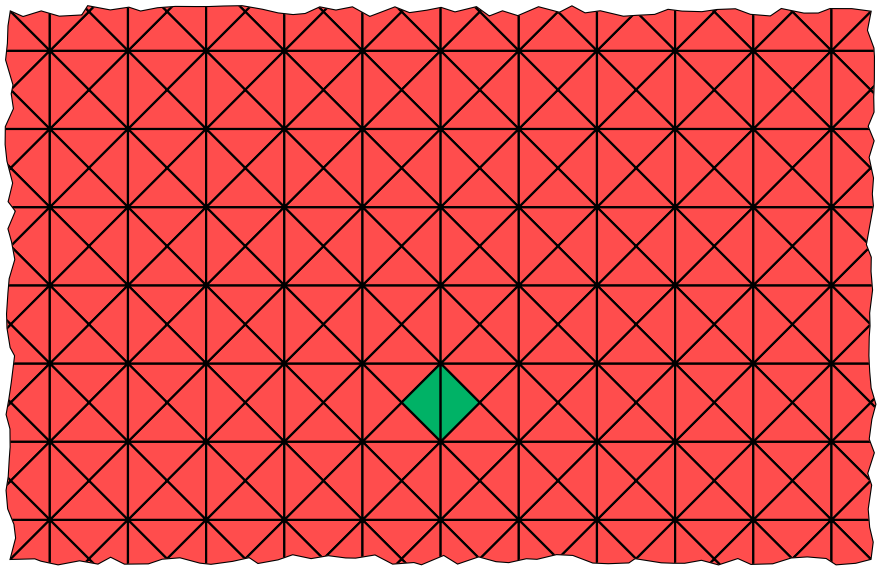
Étant donné un jeu de tuiles τ, on s'intéresse à **tous les pavages valides** par τ.

Premier exemple

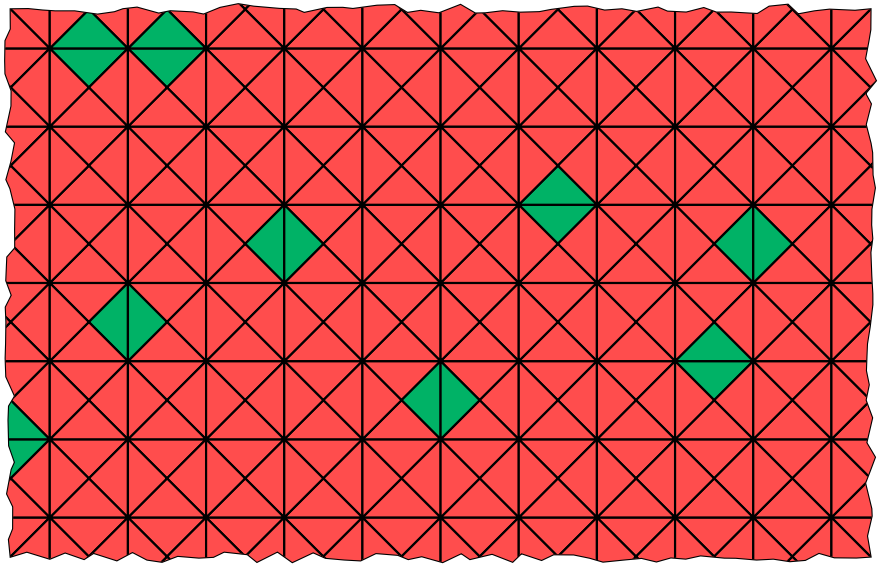
Premier exemple



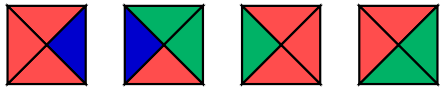
Premier exemple



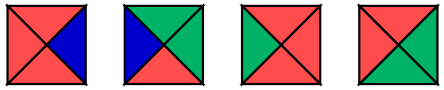
Premier exemple

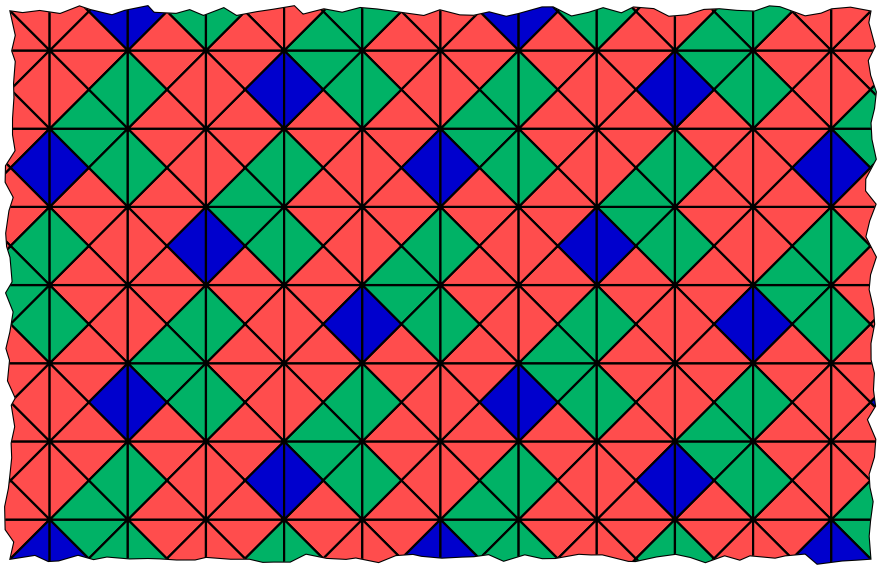


Deuxième exemple

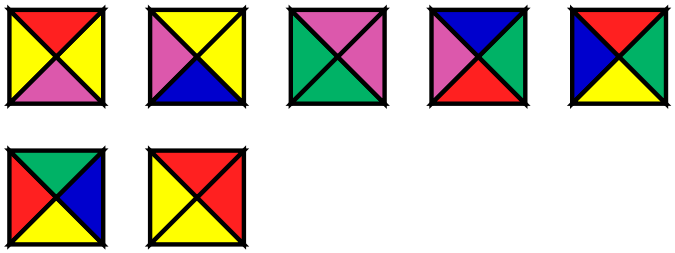


Deuxième exemple

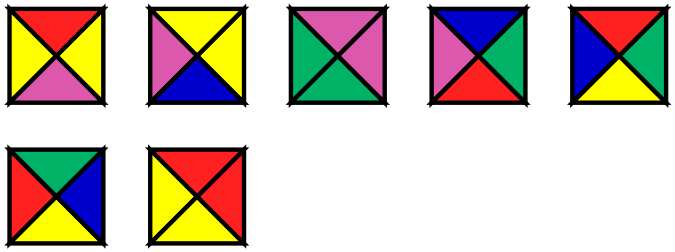




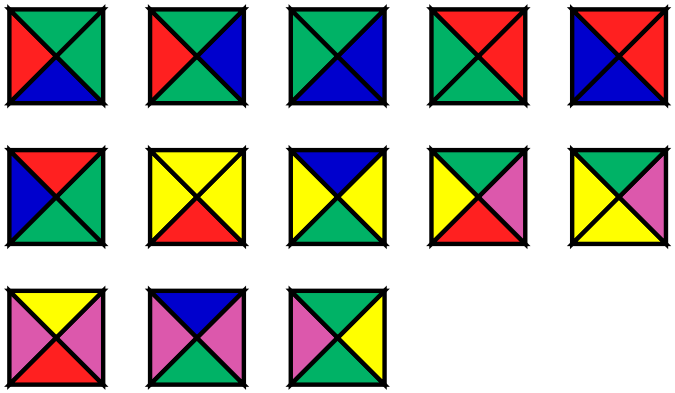
Et celui-ci ?



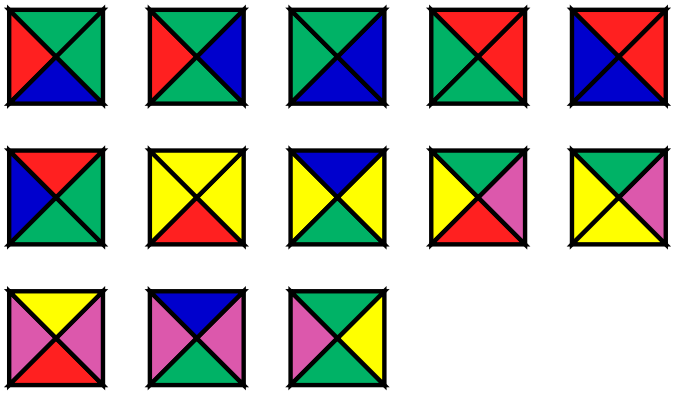
Et celui-ci ?

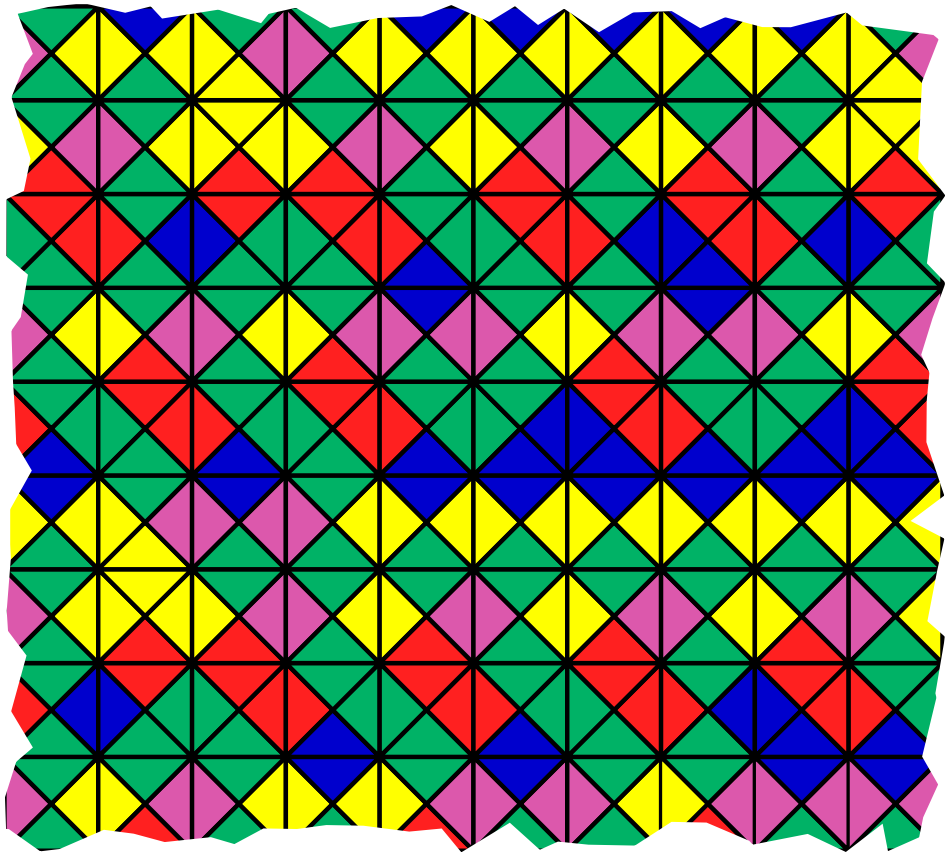


Et celui-là ?



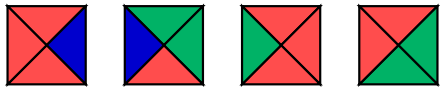
Et celui-là ?





Le modèle des tuiles de Wang

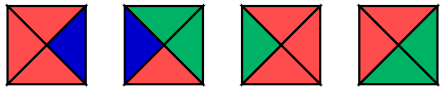
Ensemble fini de tuiles de Wang τ (une infinité de copies de chaque tuile)



Règle de voisinage

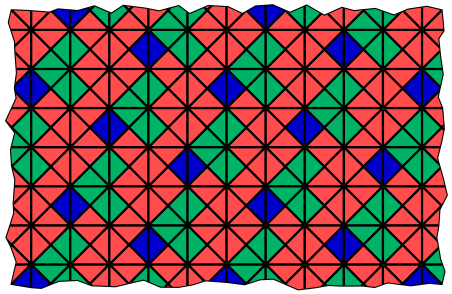
Le modèle des tuiles de Wang

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



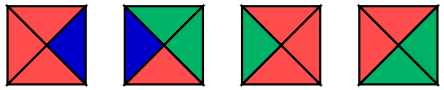
Règle de voisinage

Exemple de pavage par τ (X_τ l'ensemble de tous les pavages valides)



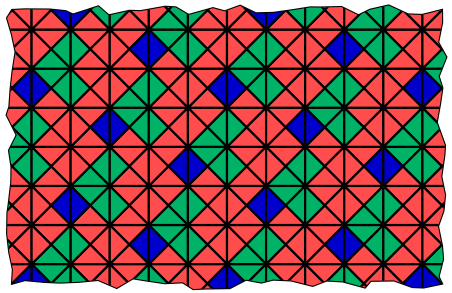
Le modèle des tuiles de Wang

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



Règle de voisinage

Exemple de pavage par τ (X_τ l'ensemble de tous les pavages valides)



Problème du Domino

Entrée : Un jeu fini de tuiles de Wang τ
Sortie : **Oui** s'il existe un pavage valide par τ ($X_\tau \neq \emptyset$), **Non** sinon ($X_\tau = \emptyset$).

Le problème du Domino : remarque

Problème du Domino

Entrée : Un jeu fini de tuiles de Wang τ
Sortie : **Oui** s'il existe un pavage valide par τ ($X_\tau \neq \emptyset$), **Non** sinon ($X_\tau = \emptyset$).

Le problème du Domino : remarque

Problème du Domino

Entrée : Un jeu fini de tuiles de Wang τ
Sortie : **Oui** s'il existe un pavage valide par τ ($X_\tau \neq \emptyset$), **Non** sinon ($X_\tau = \emptyset$).

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 fini de tuiles de Wang τ
Sortie : **Oui** s'il existe un pavage valide par τ ($X_\tau \neq \emptyset$), **Non** sinon ($X_\tau = \emptyset$).

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 fini de tuiles de Wang τ
Sortie : **Oui** s'il existe un pavage valide par τ ($X_\tau \neq \emptyset$), **Non** sinon ($X_\tau = \emptyset$).

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 fini de tuiles de Wang τ
Sortie : **Oui** s'il existe un pavage valide par τ ($X_\tau \neq \emptyset$), **Non** sinon ($X_\tau = \emptyset$).

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**.

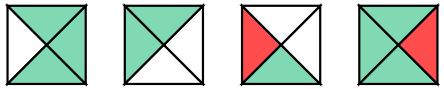
Partie difficile du problème : comment savoir qu'un jeu de tuiles permet de paver le plan ?

Conjecture (Wang, 1961)

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

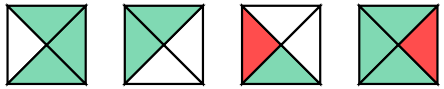
Pavages périodiques (I)

On se donne un jeu de tuiles τ.

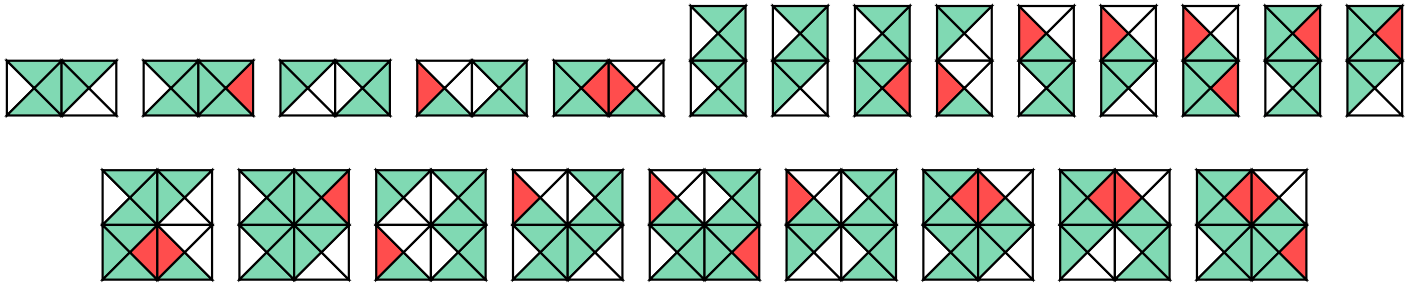


Pavages périodiques (I)

On se donne un jeu de tuiles τ.

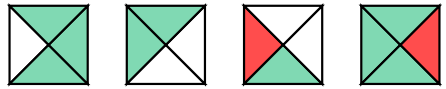


On regarde les motifs finis rectangulaires que l'on peut construire.

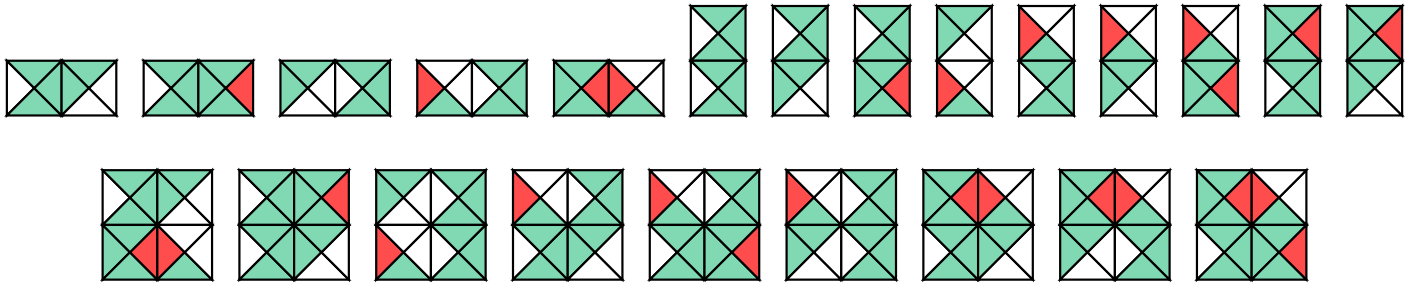


Pavages périodiques (I)

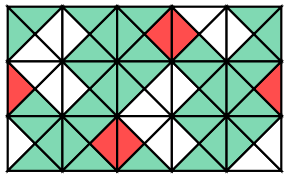
On se donne un jeu de tuiles τ.



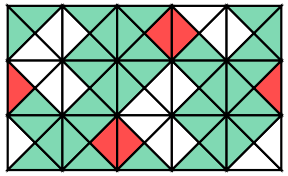
On regarde les motifs finis rectangulaires que l'on peut construire.



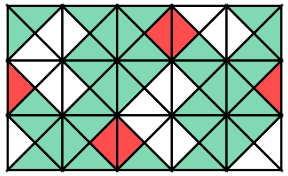
Parmi ceux-ci, on cherche les motifs qui peuvent être « collés à eux-mêmes ».

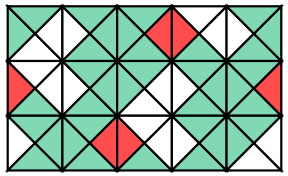


Pavages périodiques (II)

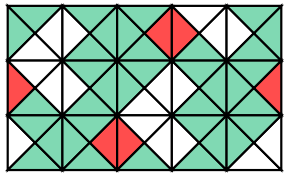


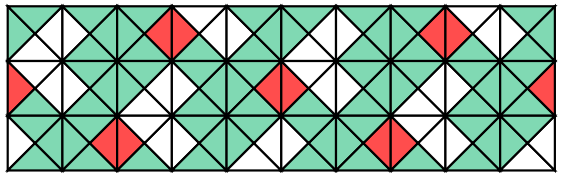
Pavages périodiques (II)



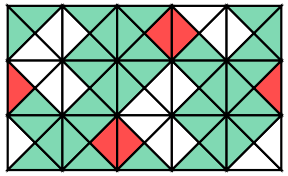


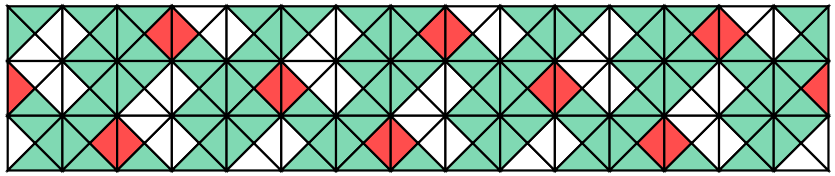
Pavages périodiques (II)



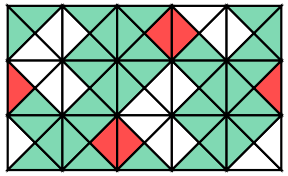


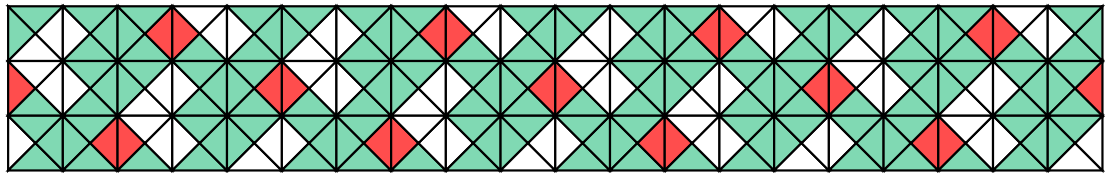
Pavages périodiques (II)



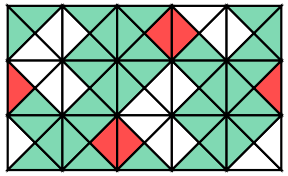


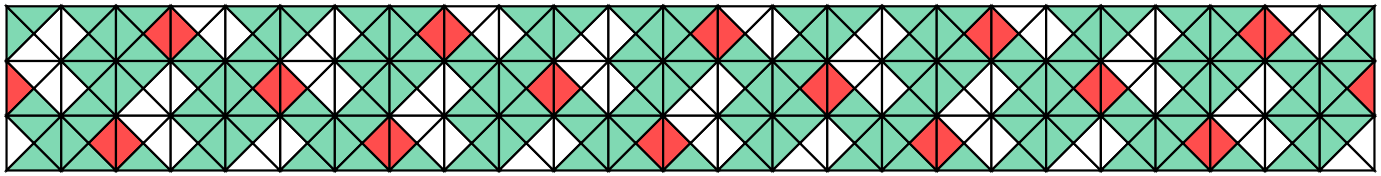
Pavages périodiques (II)



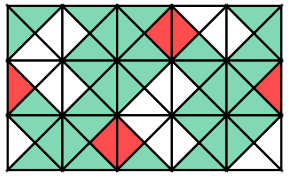


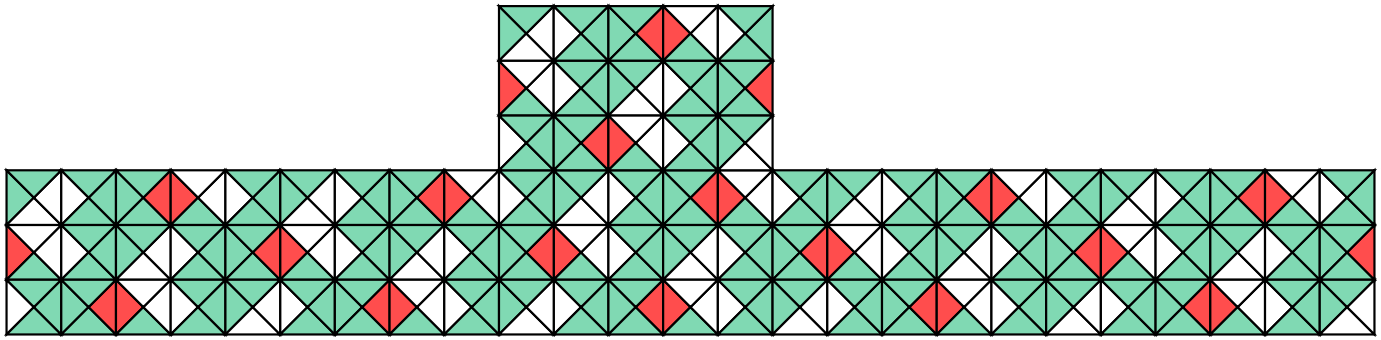
Pavages périodiques (II)



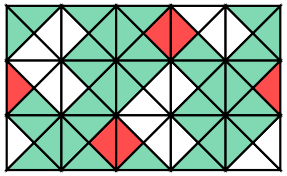


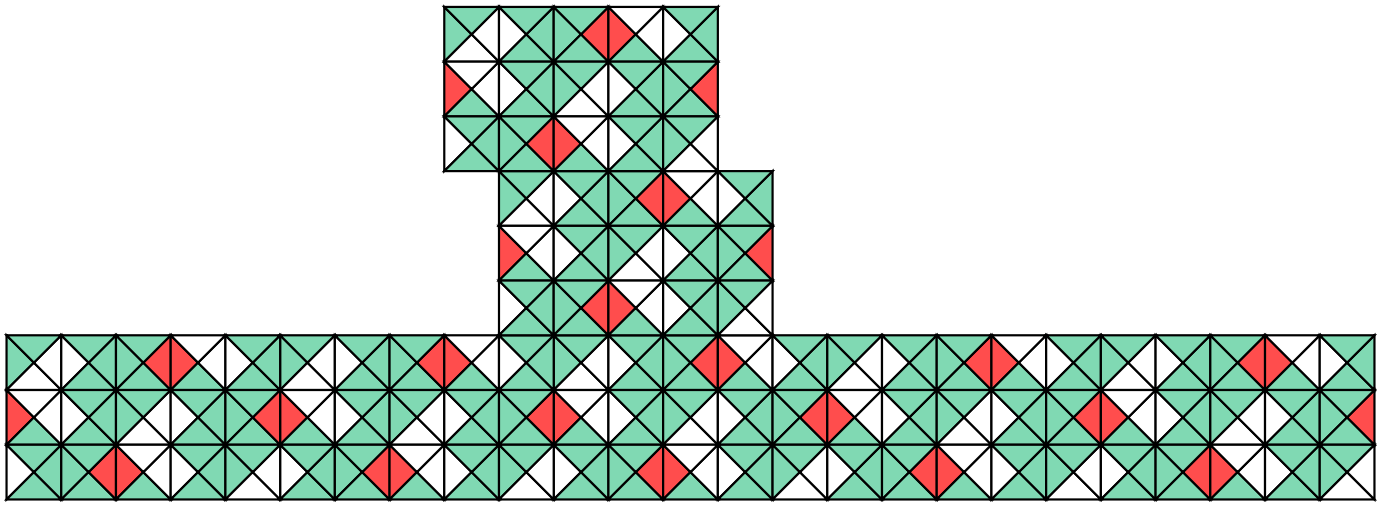
Pavages périodiques (II)



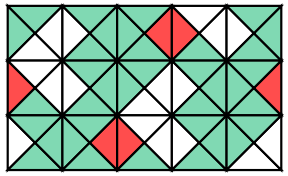


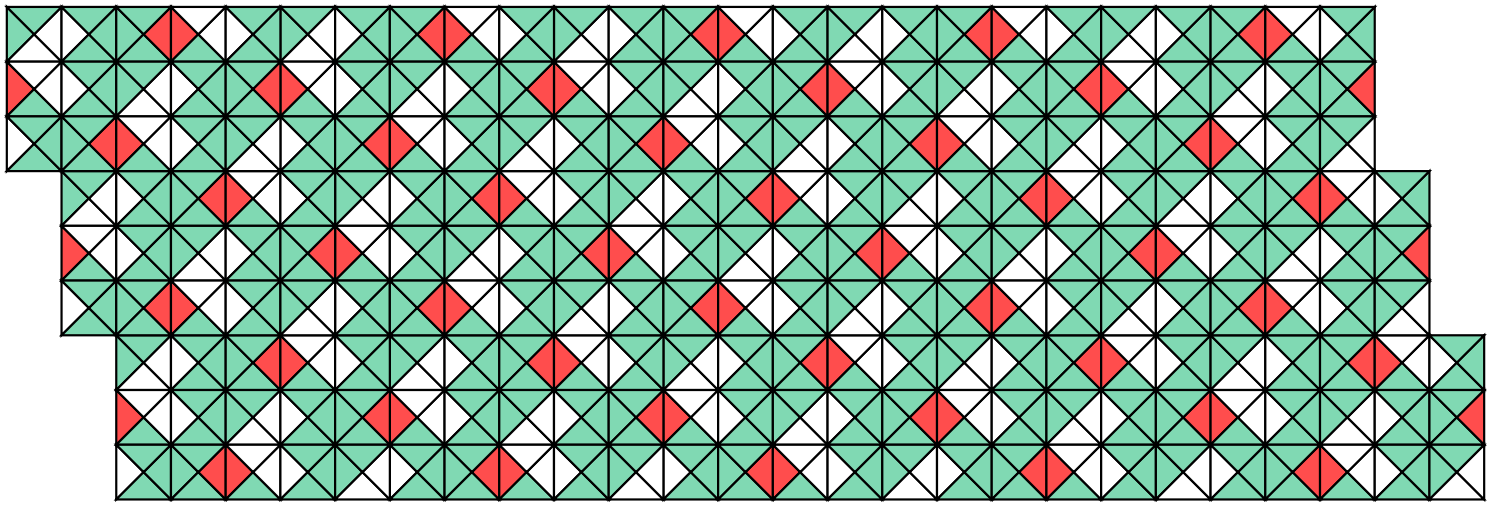
Pavages périodiques (II)





Pavages périodiques (II)



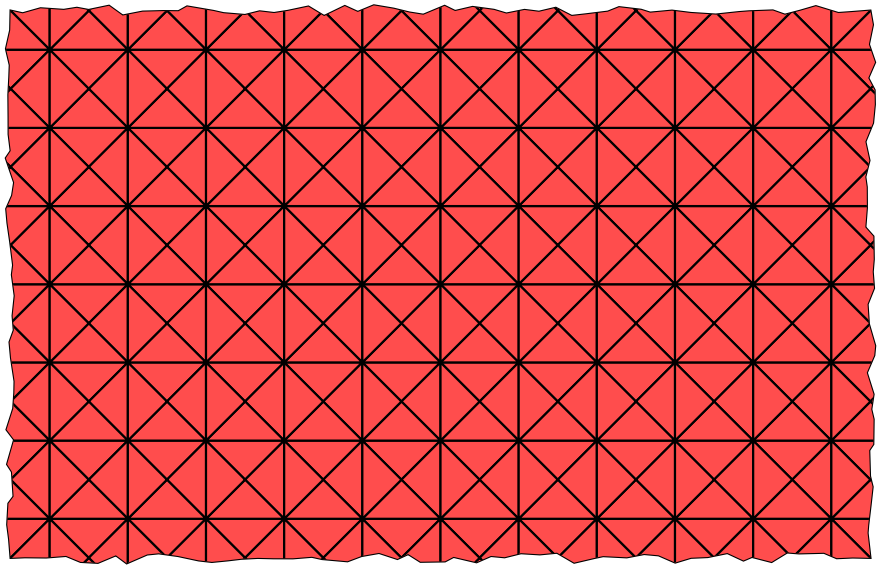


Premier exemple

Peut-on construire un pavage périodique ?

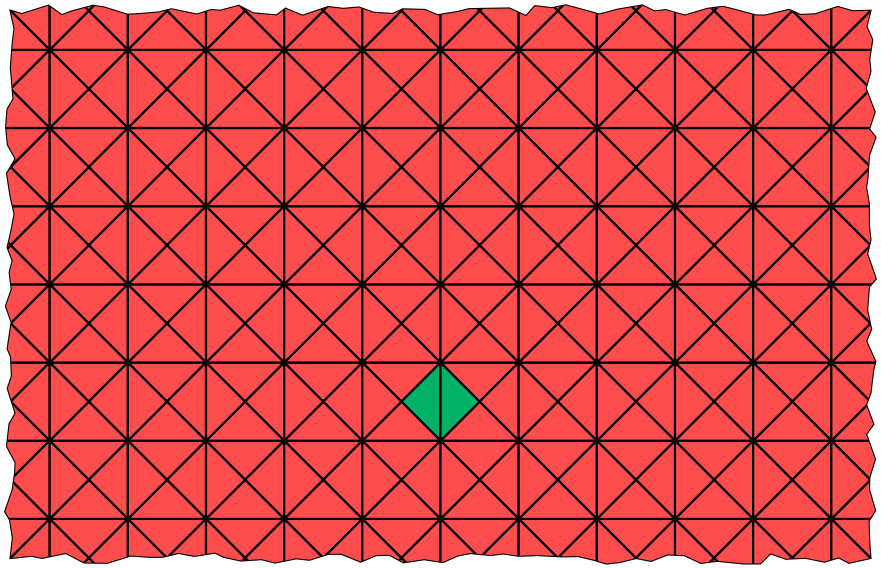
Premier exemple

Peut-on construire un pavage périodique ?



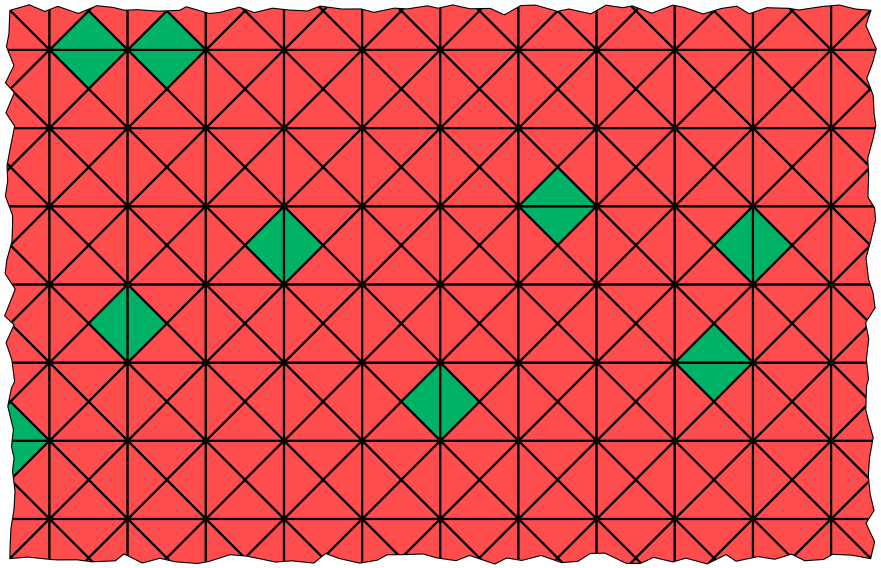
Premier exemple

Peut-on construire un pavage périodique ?

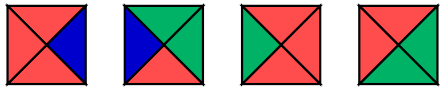


Premier exemple

Peut-on construire un pavage périodique ?

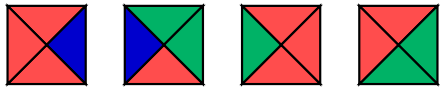


Deuxième exemple

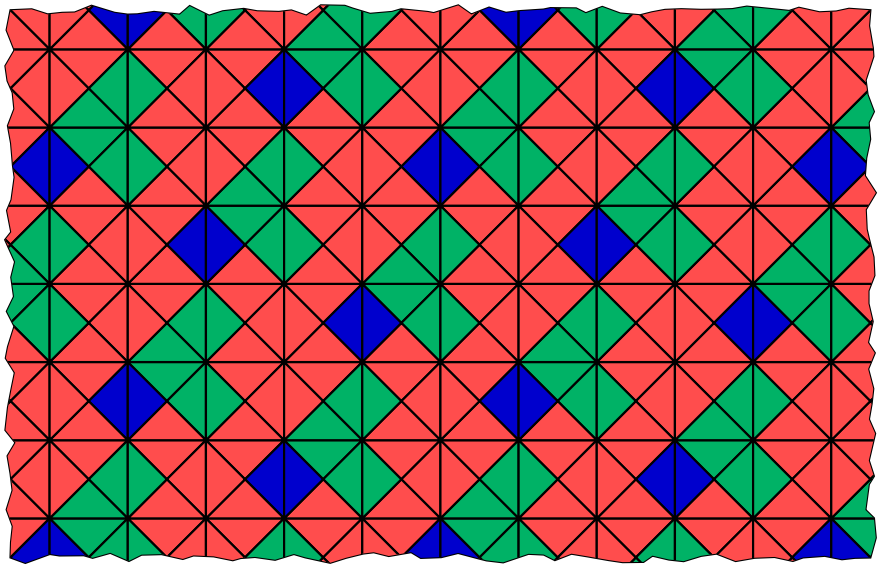


Peut-on construire un pavage périodique ?

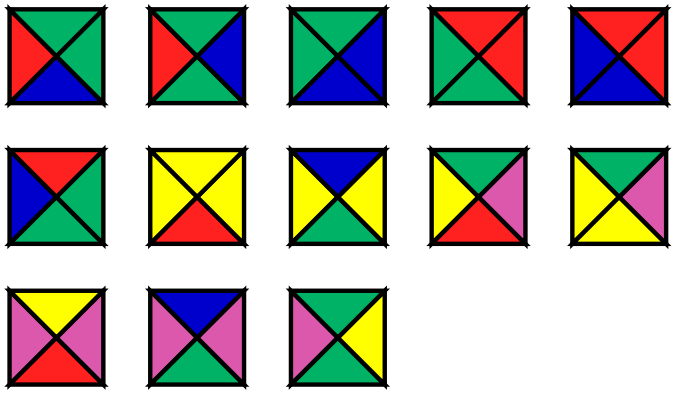
Deuxième exemple



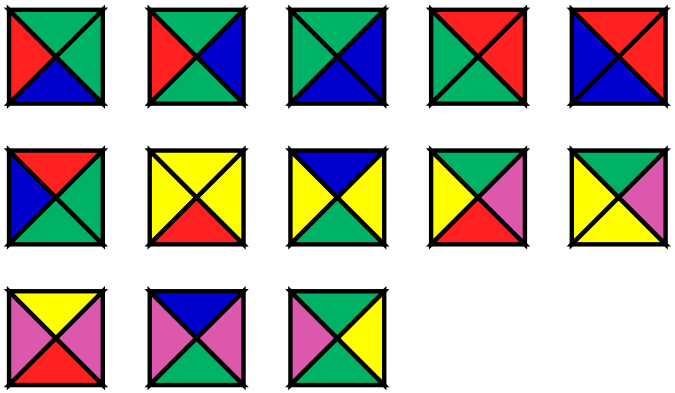
Peut-on construire un pavage périodique ?

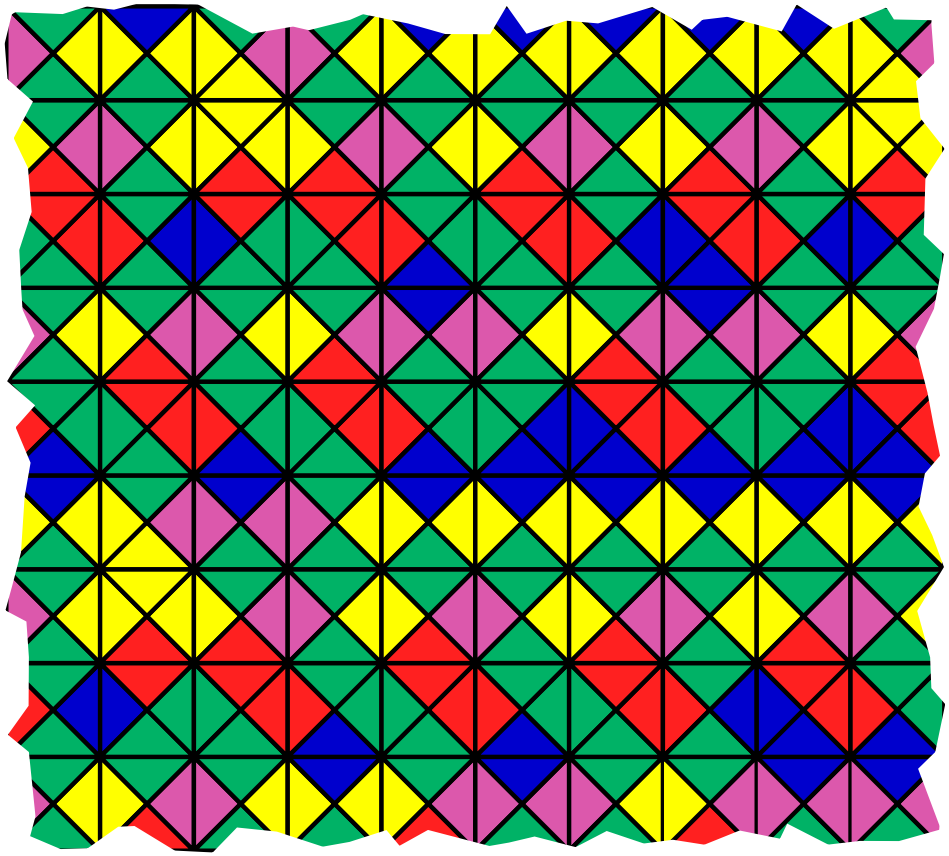


Et celui-là ?



Et celui-là ?





Existence de jeux de tuiles apériodiques ?

Conjecture (Wang, 1961)

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

Existence de jeux de tuiles apériodiques ?

Conjecture (Wang, 1961)

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

Si la conjecture de Wang est vraie, on peut décider le problème du Domino !

Existence de jeux de tuiles apériodiques ?

Conjecture (Wang, 1961)

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

Si la conjecture de Wang est vraie, on peut décider le problème du Domino !

Semi-algorithme 1 :

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

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

Existence de jeux de tuiles apériodiques ?

Conjecture (Wang, 1961)

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

Si la conjecture de Wang est vraie, on peut décider le problème du Domino !

Semi-algorithme 1 :

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

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

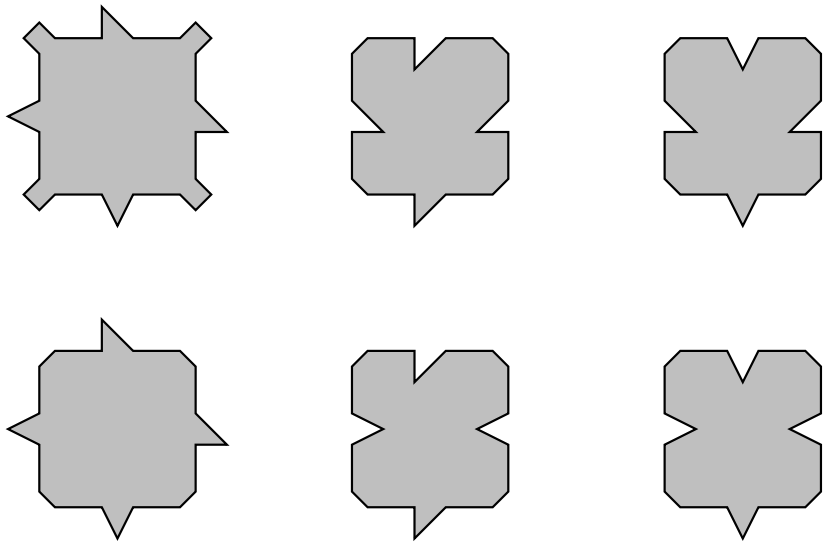
Un jeu de tuiles est **apériodique** s'il permet de paver le plan, et si aucun des pavages valides n'est périodique.

Question

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

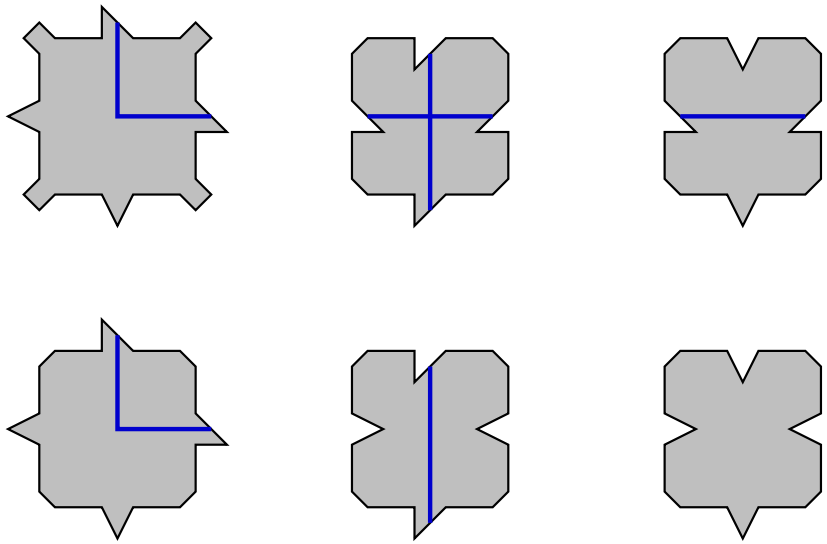
Le jeu de tuiles de Robinson

On s'autorise les rotations et réflexions des 6 tuiles suivantes :

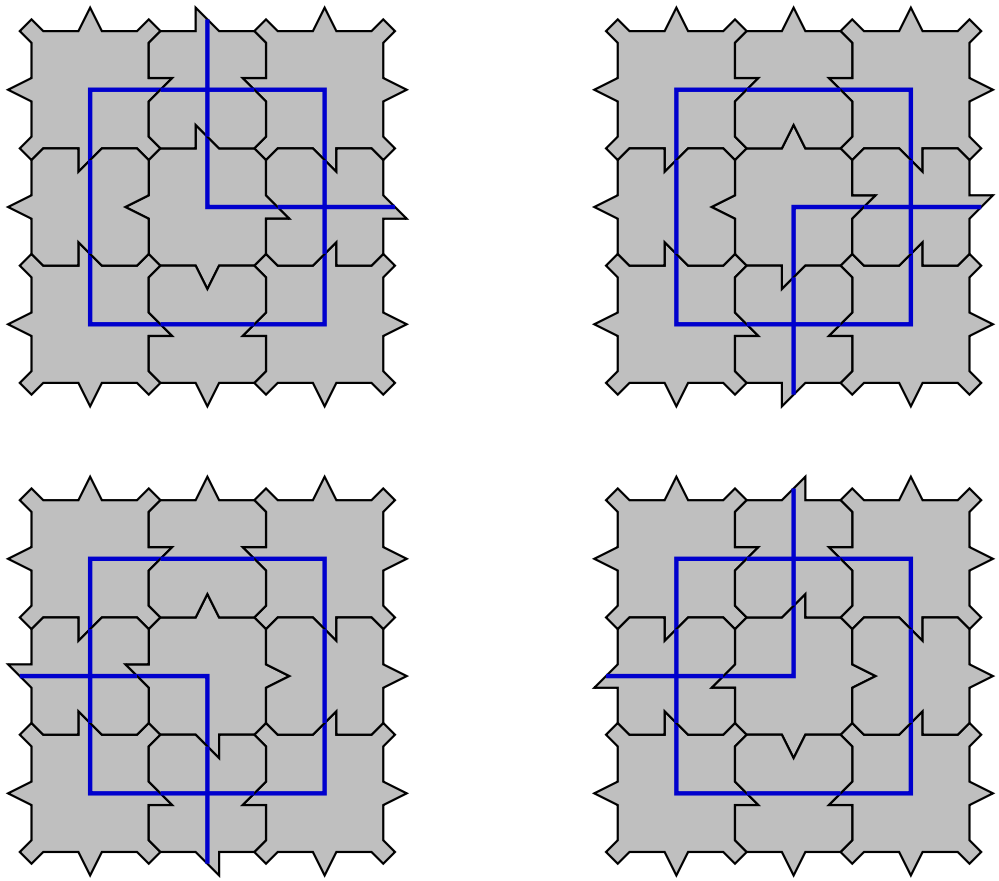


Le jeu de tuiles de Robinson

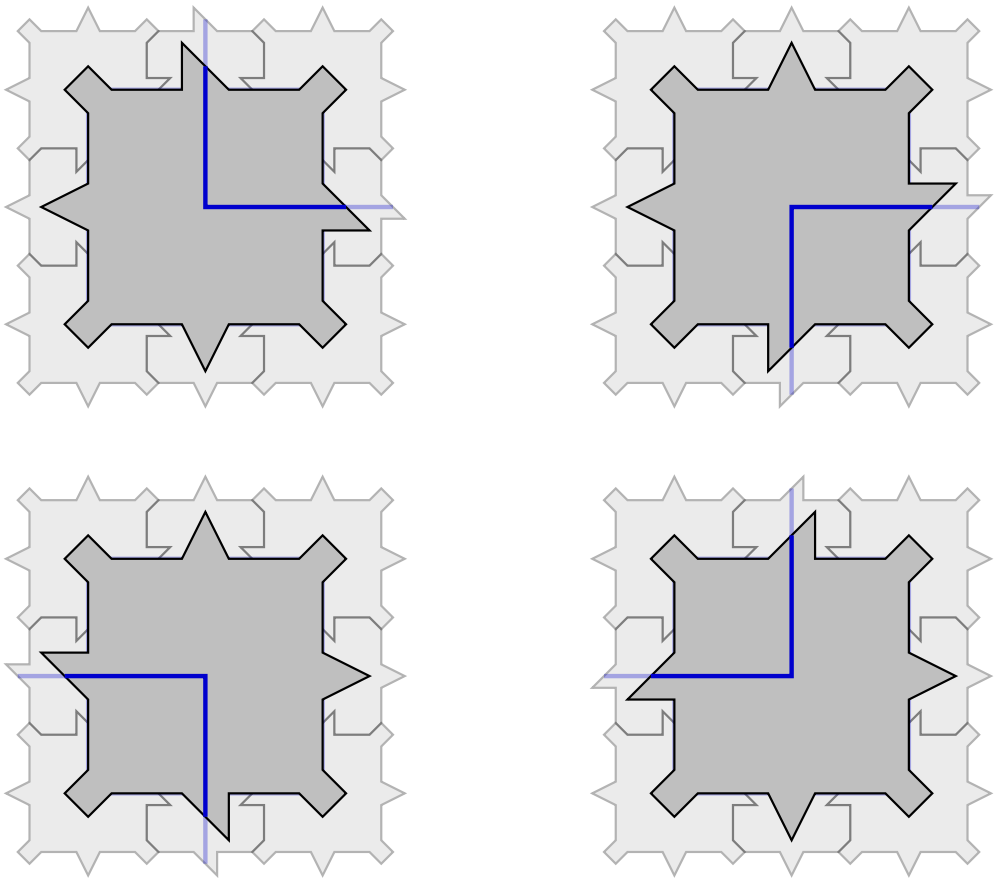
On s'autorise les rotations et réflexions des 6 tuiles suivantes :



Pavage par le jeu de Robinson (I)

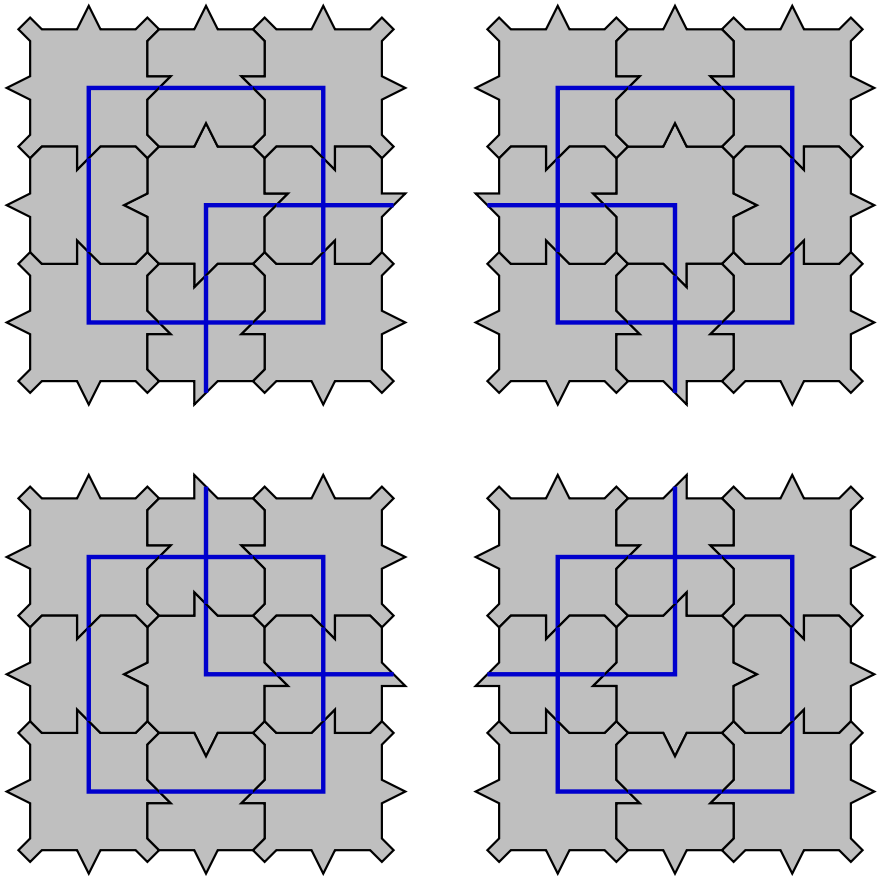


Pavage par le jeu de Robinson (I)

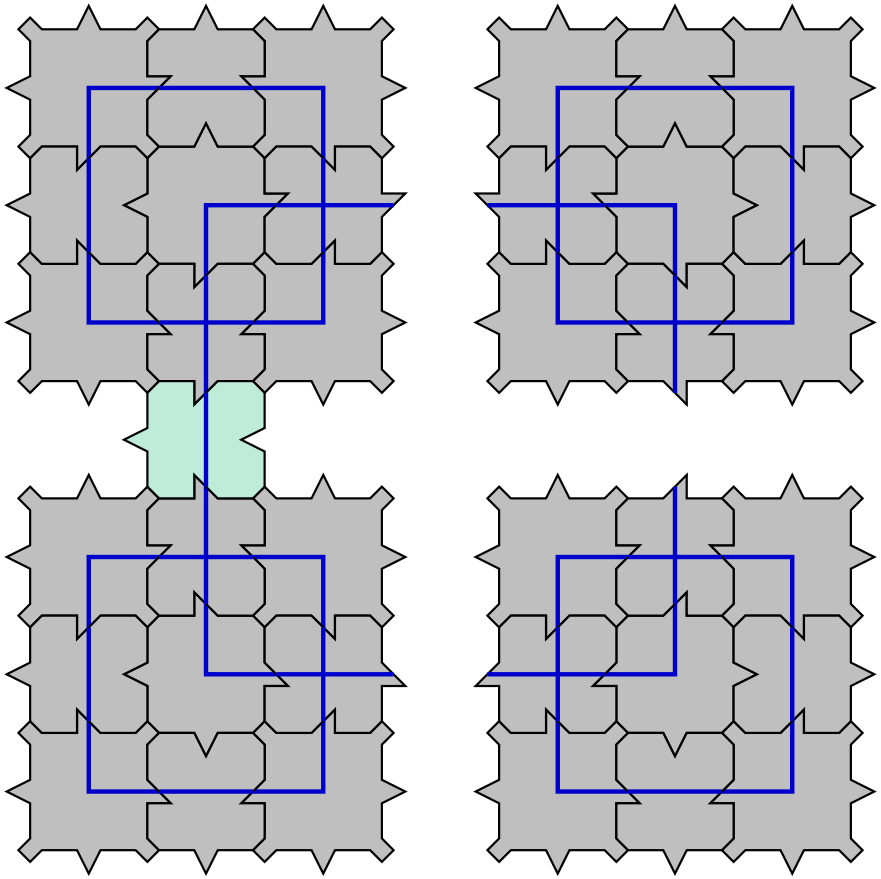


Ils se comportent comme de grands  .

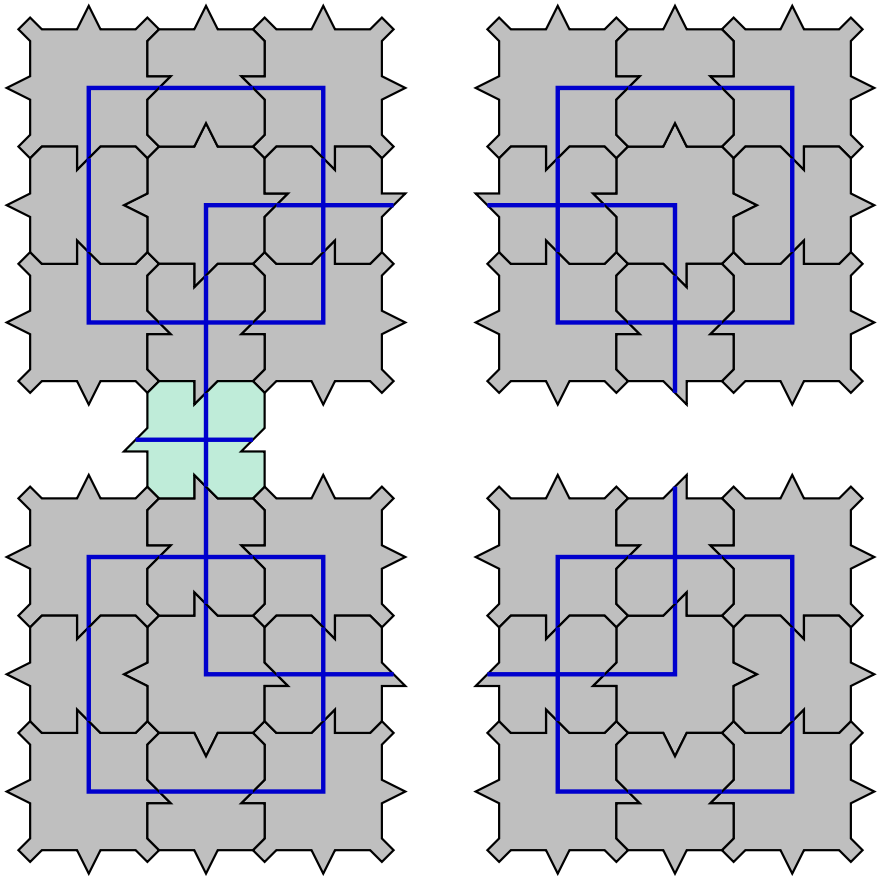
Pavage par le jeu de Robinson (II)



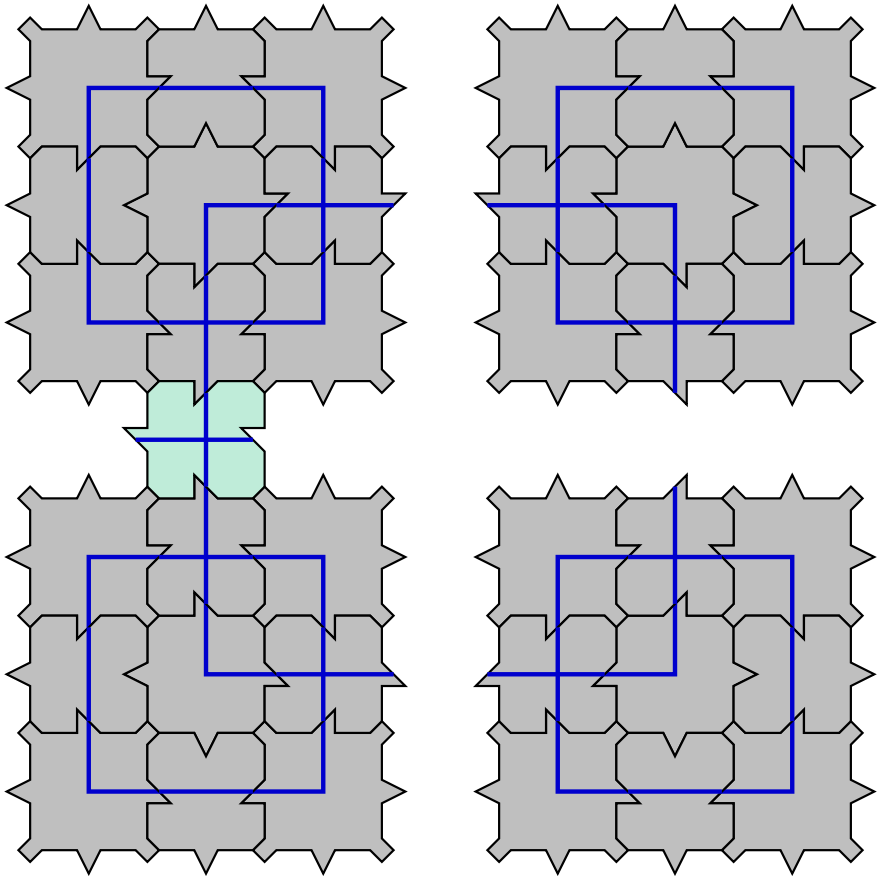
Pavage par le jeu de Robinson (II)



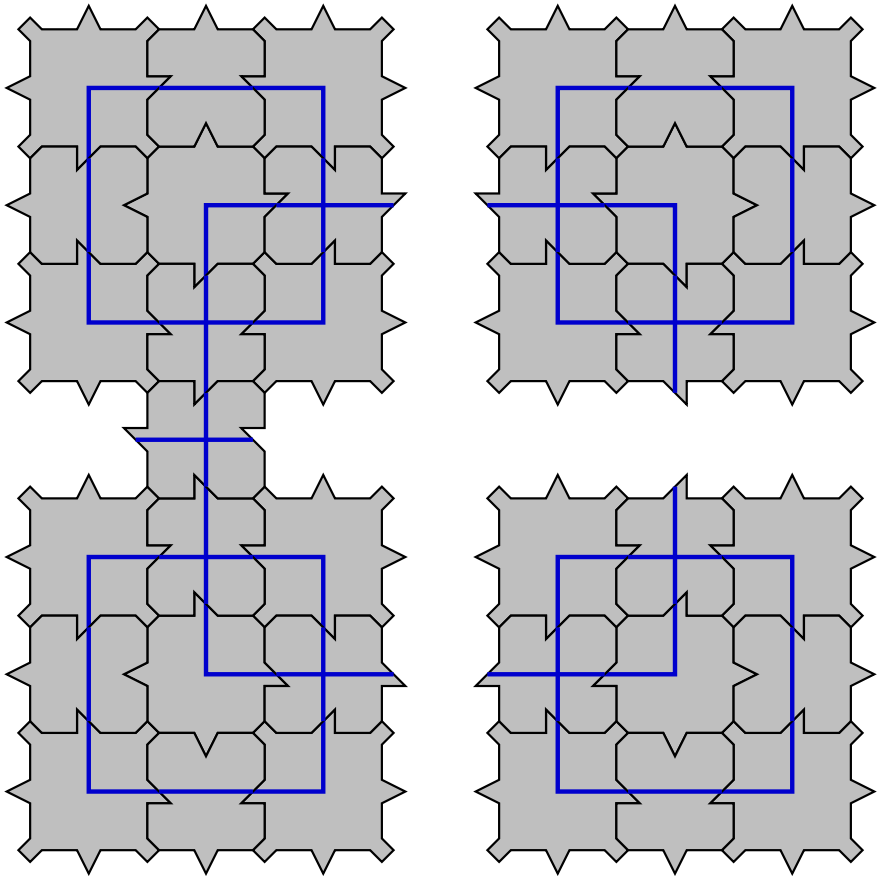
Pavage par le jeu de Robinson (II)



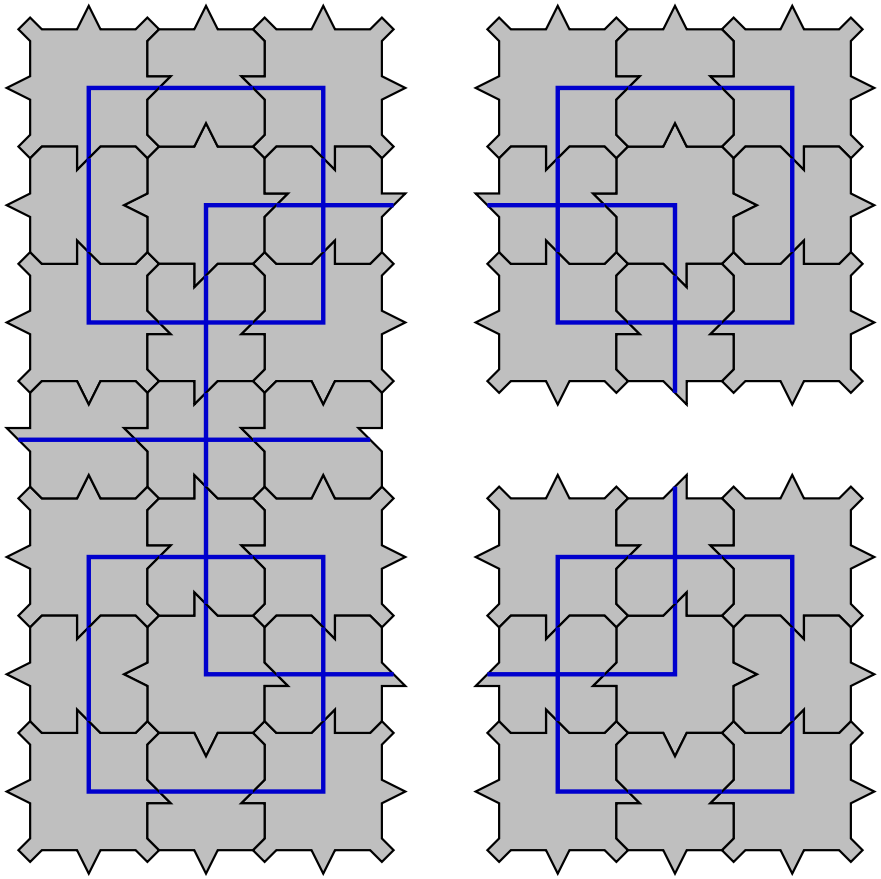
Pavage par le jeu de Robinson (II)



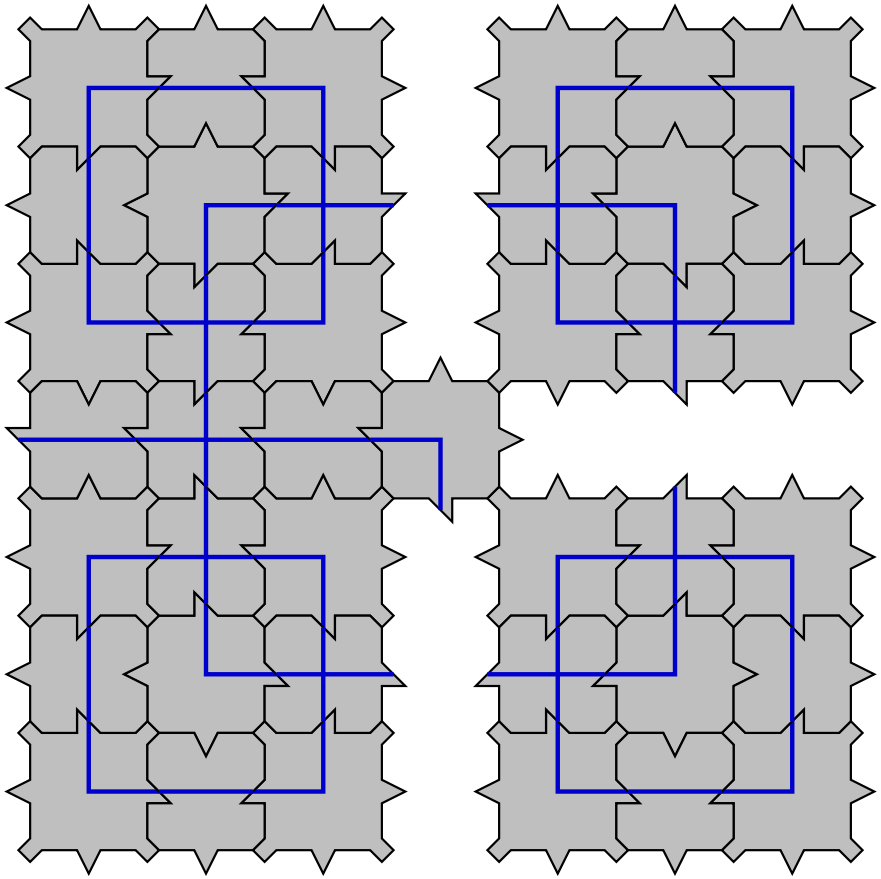
Pavage par le jeu de Robinson (II)



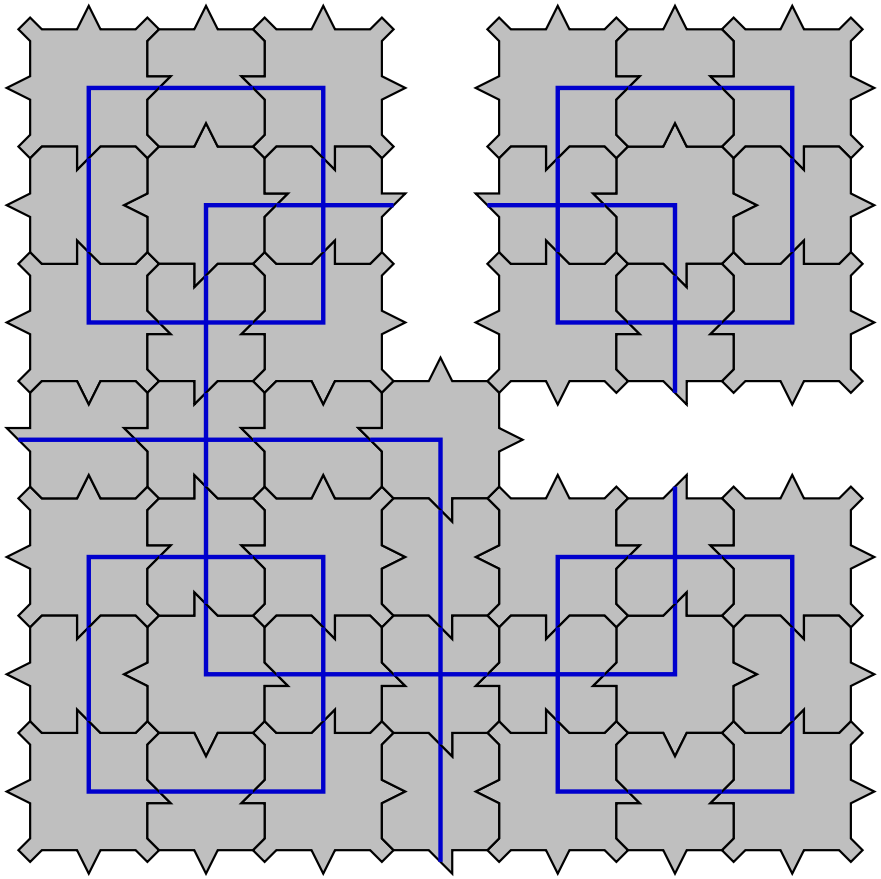
Pavage par le jeu de Robinson (II)



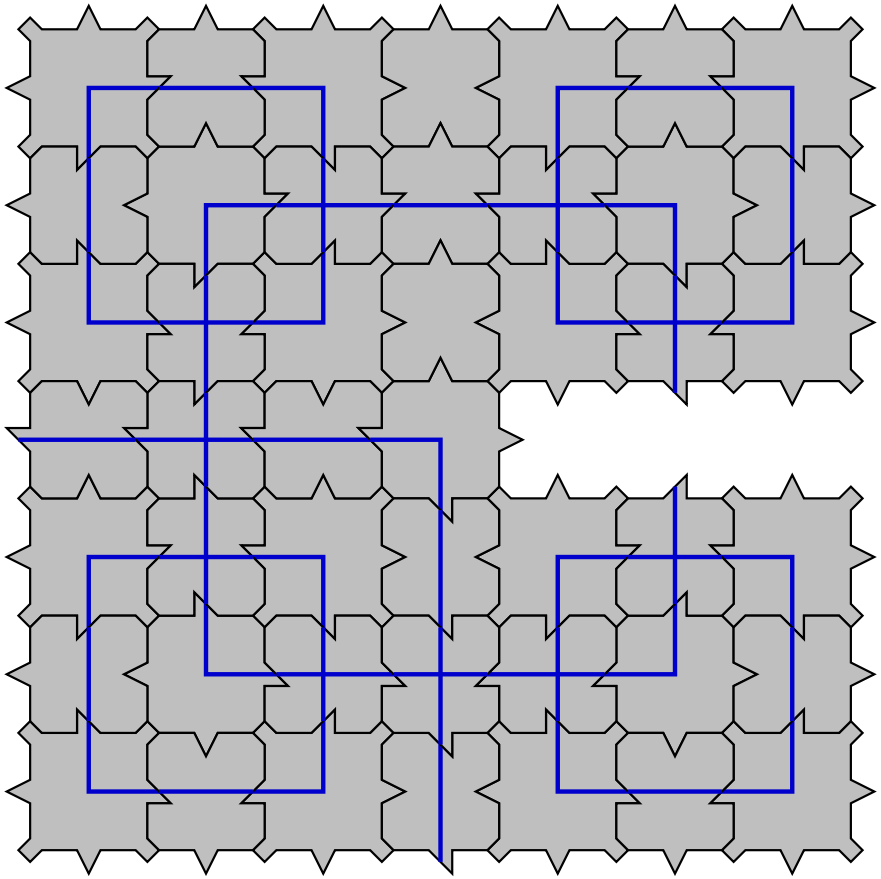
Pavage par le jeu de Robinson (II)



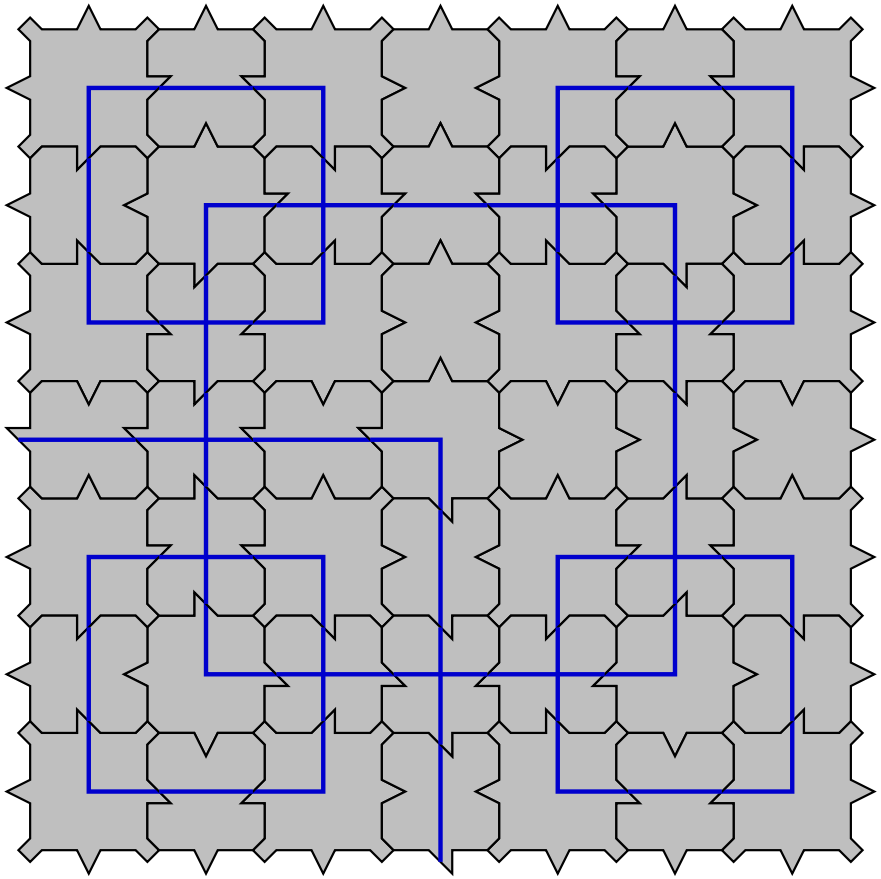
Pavage par le jeu de Robinson (II)



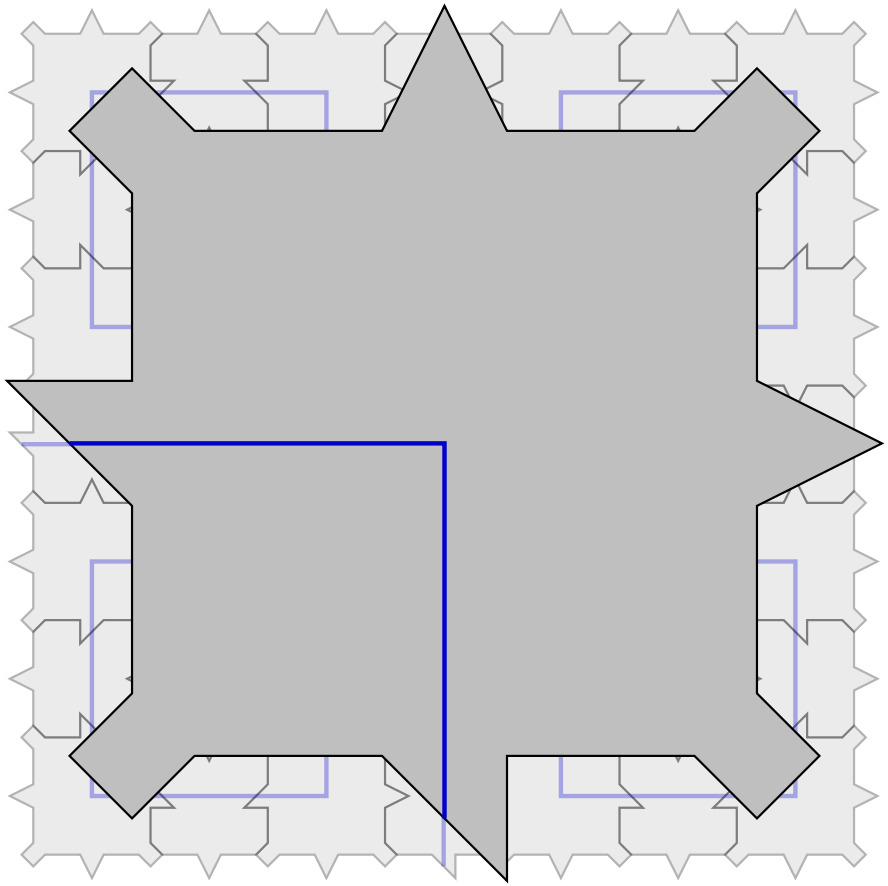
Pavage par le jeu de Robinson (II)



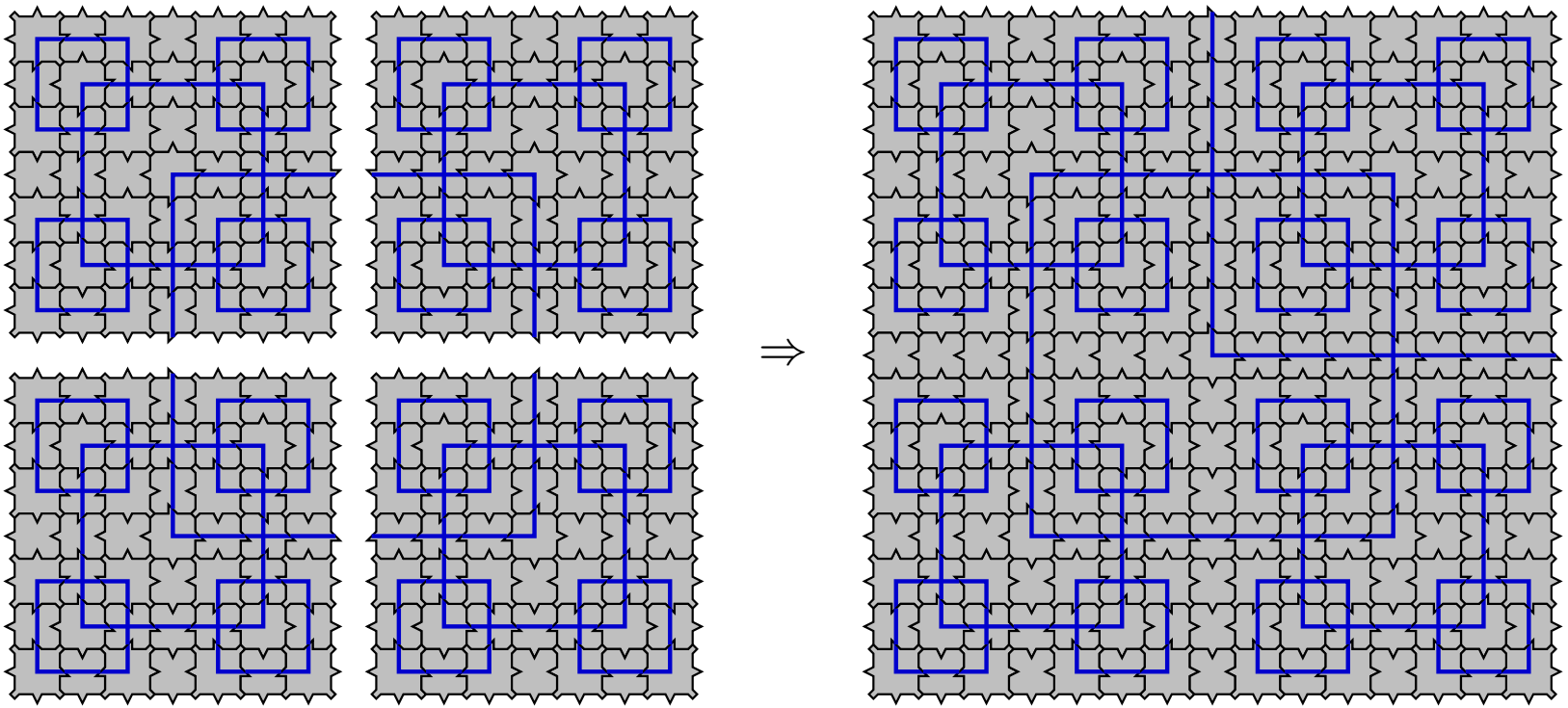
Pavage par le jeu de Robinson (II)



Pavage par le jeu de Robinson (II)

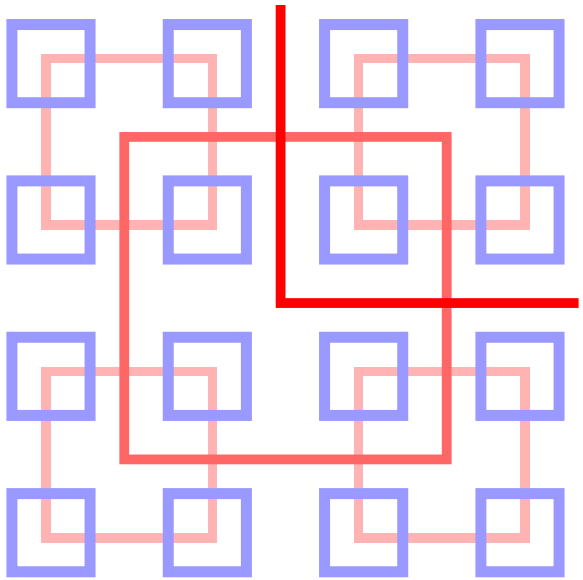


Pavage par le jeu de Robinson (III)



Sur la structure des pavages

Hiérarchie de carrés : les carrés de niveau n sont regroupés par 4 pour former les carrés de niveau $n+1$

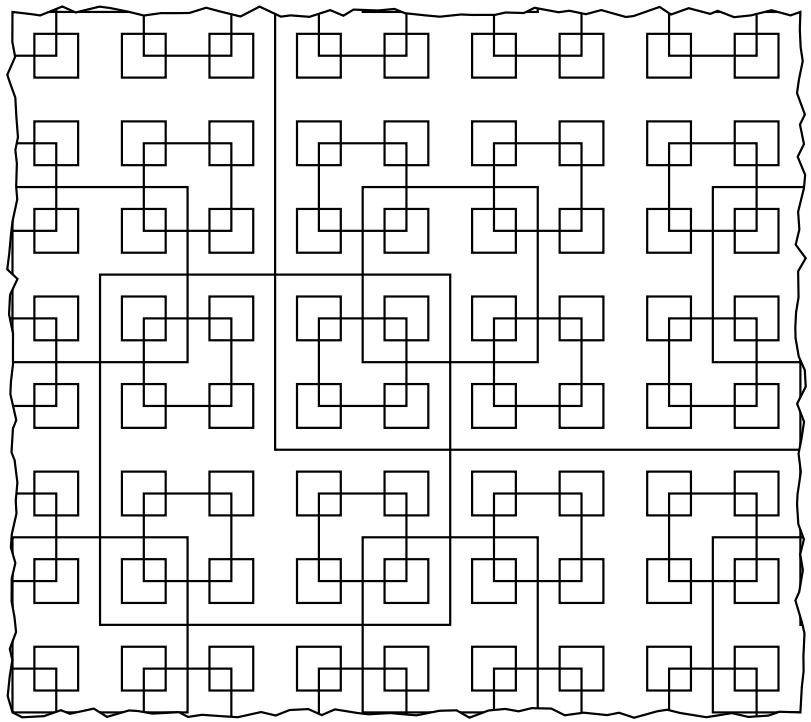


Un carré de niveau n intersecte 4 carrés de niveau $n-1$ et 1 carré de niveau $n+1$.

Apériodicité du jeu de tuiles de Robinson

Proposition

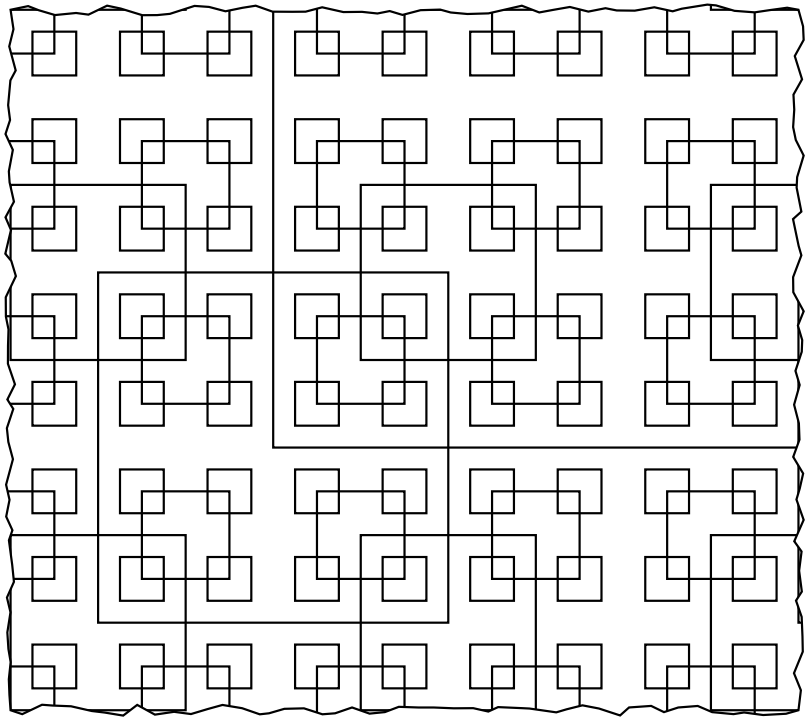
Le jeu de tuiles de Robinson est apériodique.



Apériodicité du jeu de tuiles de Robinson

Proposition

Le jeu de tuiles de Robinson est apériodique.



$\Rightarrow$ **la conjecture formulée par Wang est fausse !**

Et maintenant ?

La stratégie de Wang pour résoudre le problème du Domino a échoué. . .

Et maintenant ?

La stratégie de Wang pour résoudre le problème du Domino a échoué...

Et si en fait ce problème du Domino était ***vraiment*** difficile ?

Et maintenant ?

La stratégie de Wang pour résoudre le problème du Domino a échoué...

Et si en fait ce problème du Domino était ***vraiment*** difficile ?

Pour certains problèmes, il est **impossible** de trouver un algorithme qui les résout. On peut montrer qu'il **ne peut pas exister** de tel algorithme.

Et maintenant ?

La stratégie de Wang pour résoudre le problème du Domino a échoué...

Et si en fait ce problème du Domino était ***vraiment*** difficile ?

Pour certains problèmes, il est **impossible** de trouver un algorithme qui les résout. On peut montrer qu'il **ne peut pas exister** de tel algorithme.

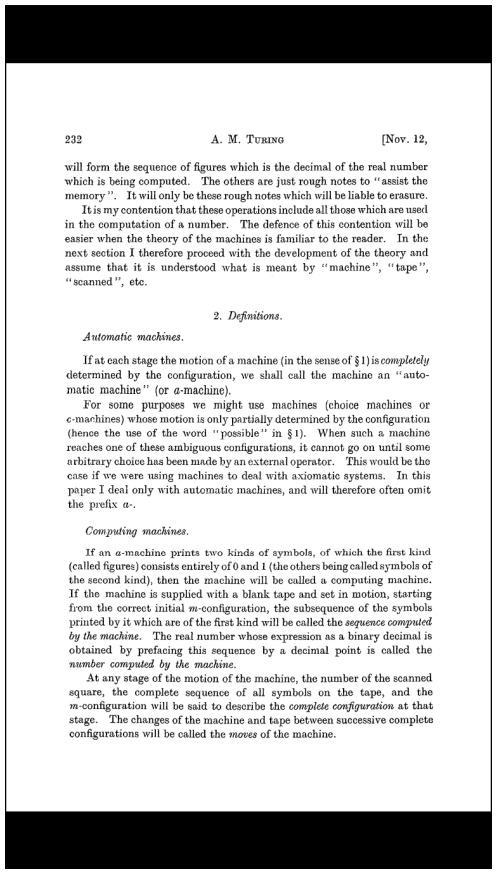
Ces problèmes sont dits **indécidables**.

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.

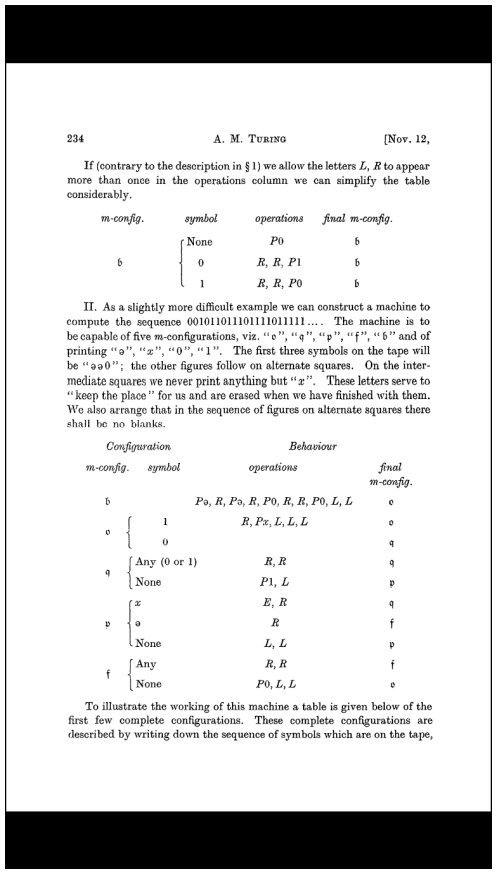
If at each stage the motion of a machine (in the sense of § 1) is *completely* determined by the configuration, we shall call the machine an "automatic machine" (or *a*-machine).

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.*
b	None	$P0$	b
	0	$R, R, P1$	b
	1	$R, R, P0$	b

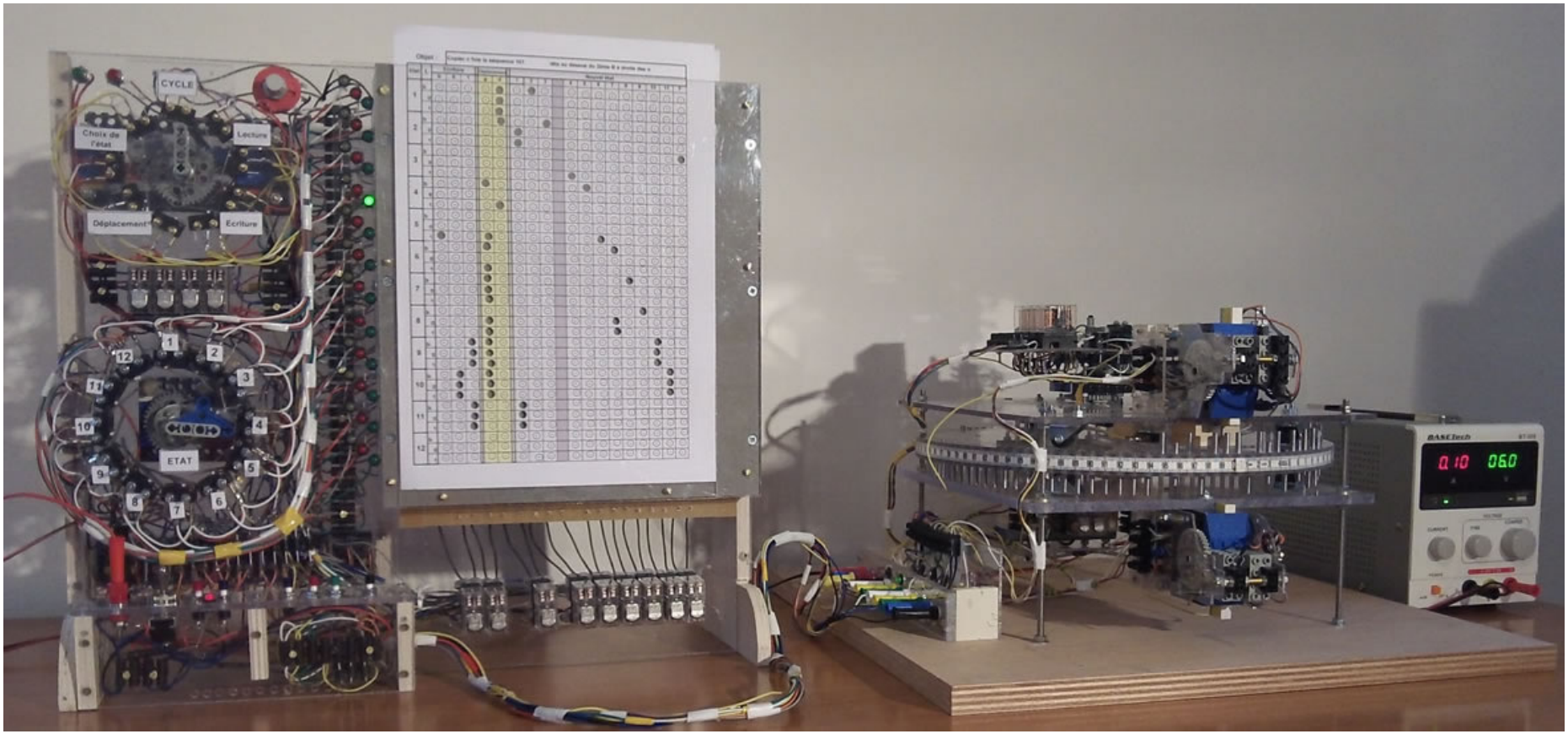
II. As a slightly more difficult example we can construct a machine to compute the sequence 001011011101111011111 .... The machine is to be capable of five *m*-configurations, viz. "e", "q", "p", "f", "b" and of printing "ə", "x", "0", "1". The first three symbols on the tape will be "ə ə 0"; the other 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.*
b		$Pə, R, Pə, R, P0, R, R, P0, L, L$	e
e	1	R, Px, L, L, L	e
	0		q
q	Any (0 or 1)	R, R	q
	None	$P1, L$	p
p	x	E, R	q
	ə	R	f
	None	L, L	p
f	Any	R, R	f
	None	$P0, L, L$	e

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)

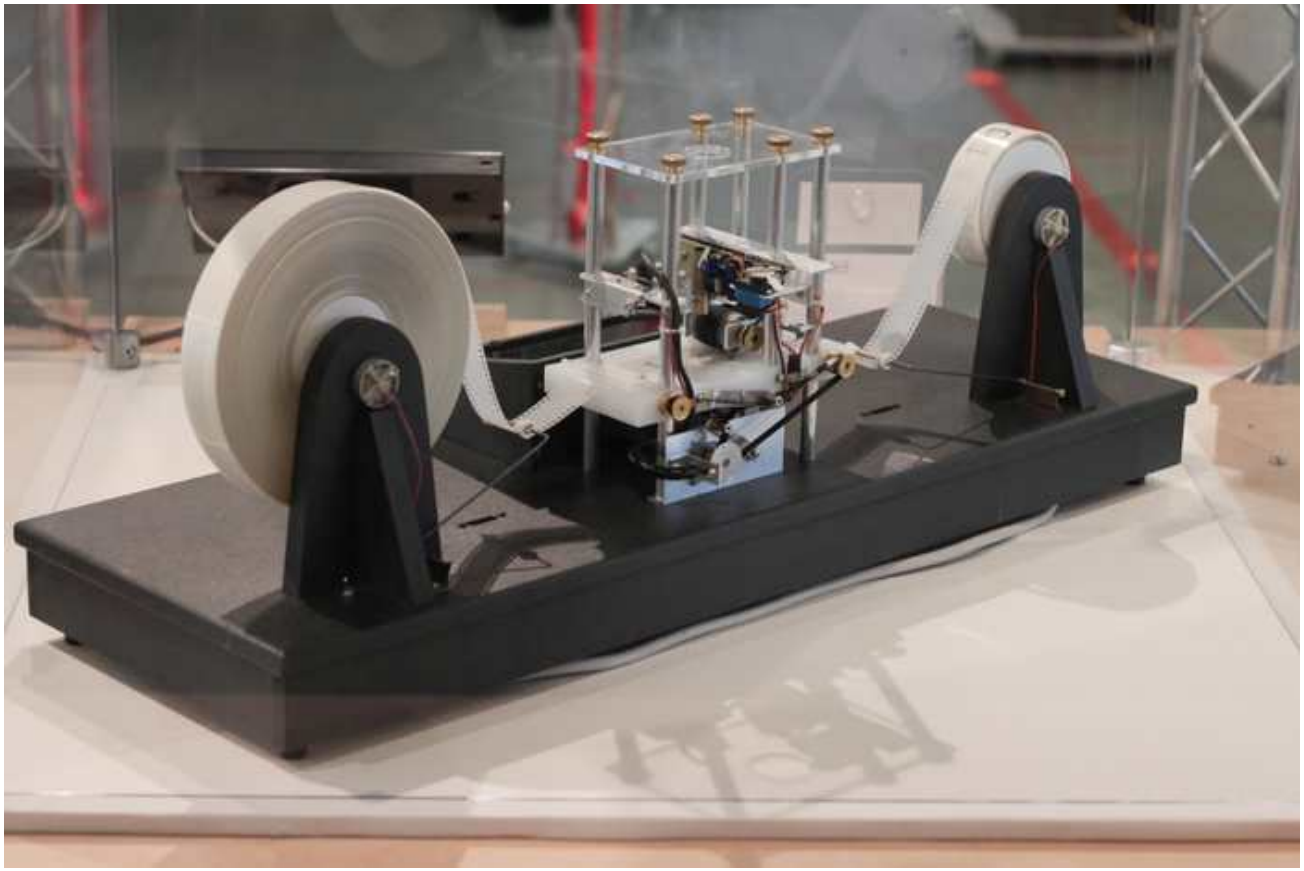
Imaginées par Alan Turing en 1936, pour délimiter ce qu'un d'algorithme est capable de faire ou non.



Modèle théorique : pas de limite ni en temps ni en espace de calcul.

Machines de Turing (I)

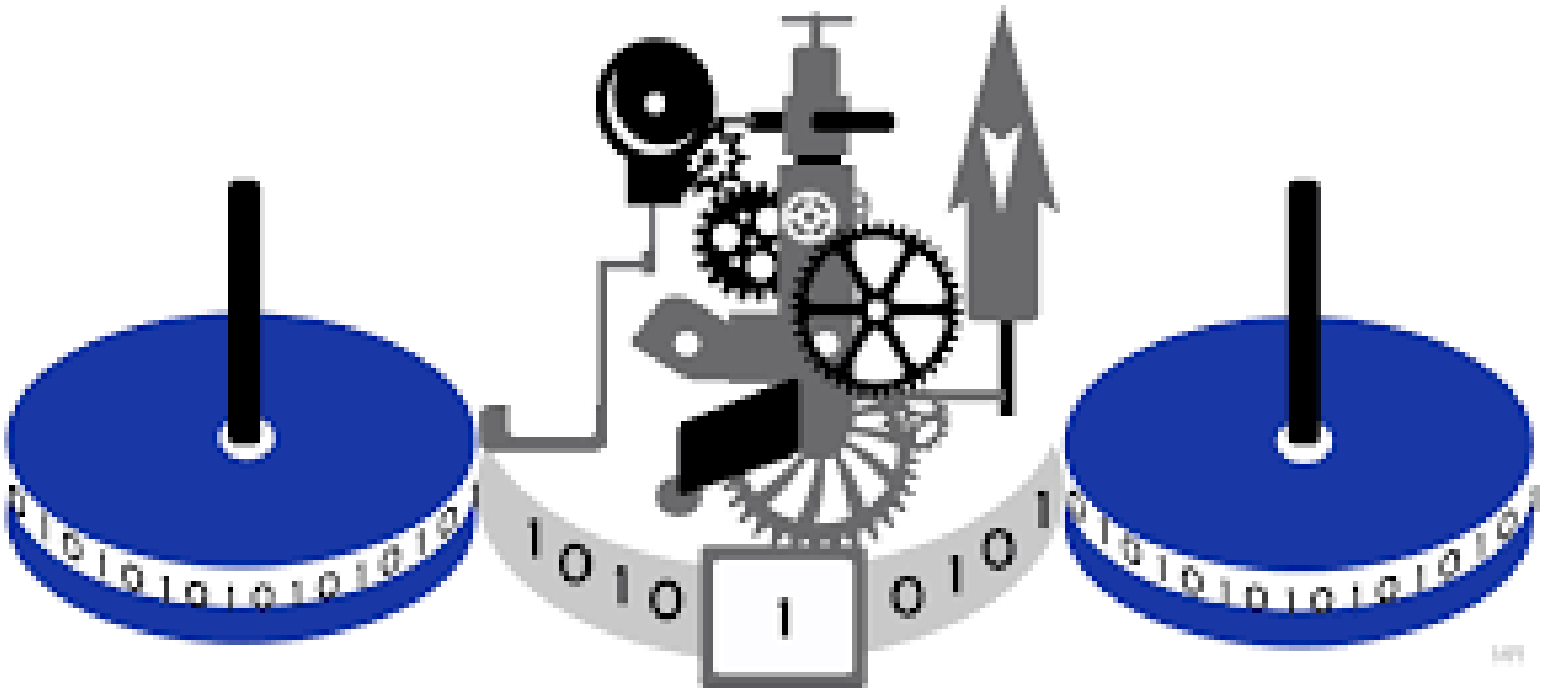
Imaginées par Alan Turing en 1936, pour délimiter ce qu'un d'algorithme est capable de faire ou non.



Modèle théorique : pas de limite ni en temps ni en espace de calcul.

Machines de Turing (I)

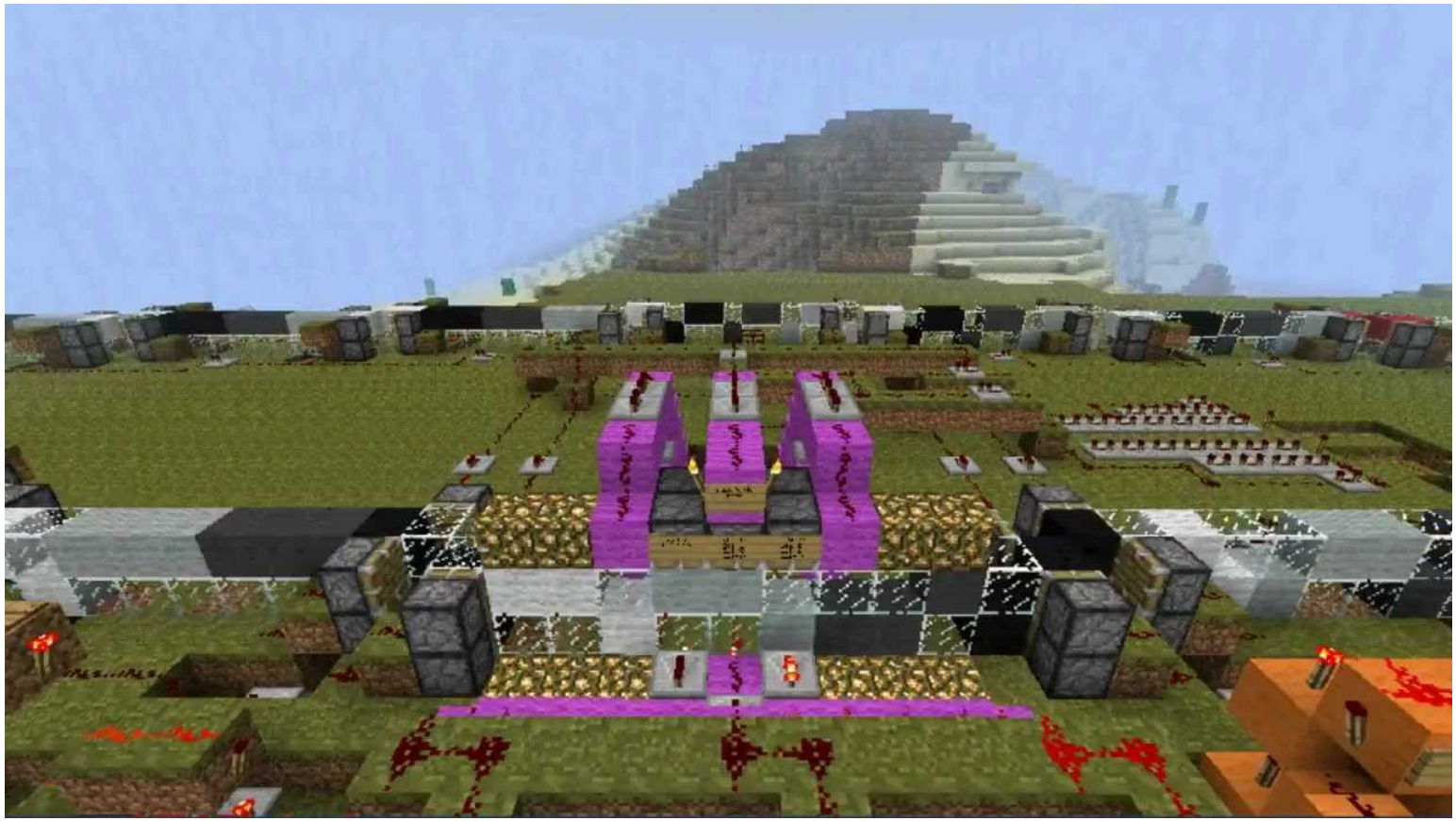
Imaginées par Alan Turing en 1936, pour délimiter ce qu'un d'algorithme est capable de faire ou non.



Modèle théorique : pas de limite ni en temps ni en espace de calcul.

Machines de Turing (I)

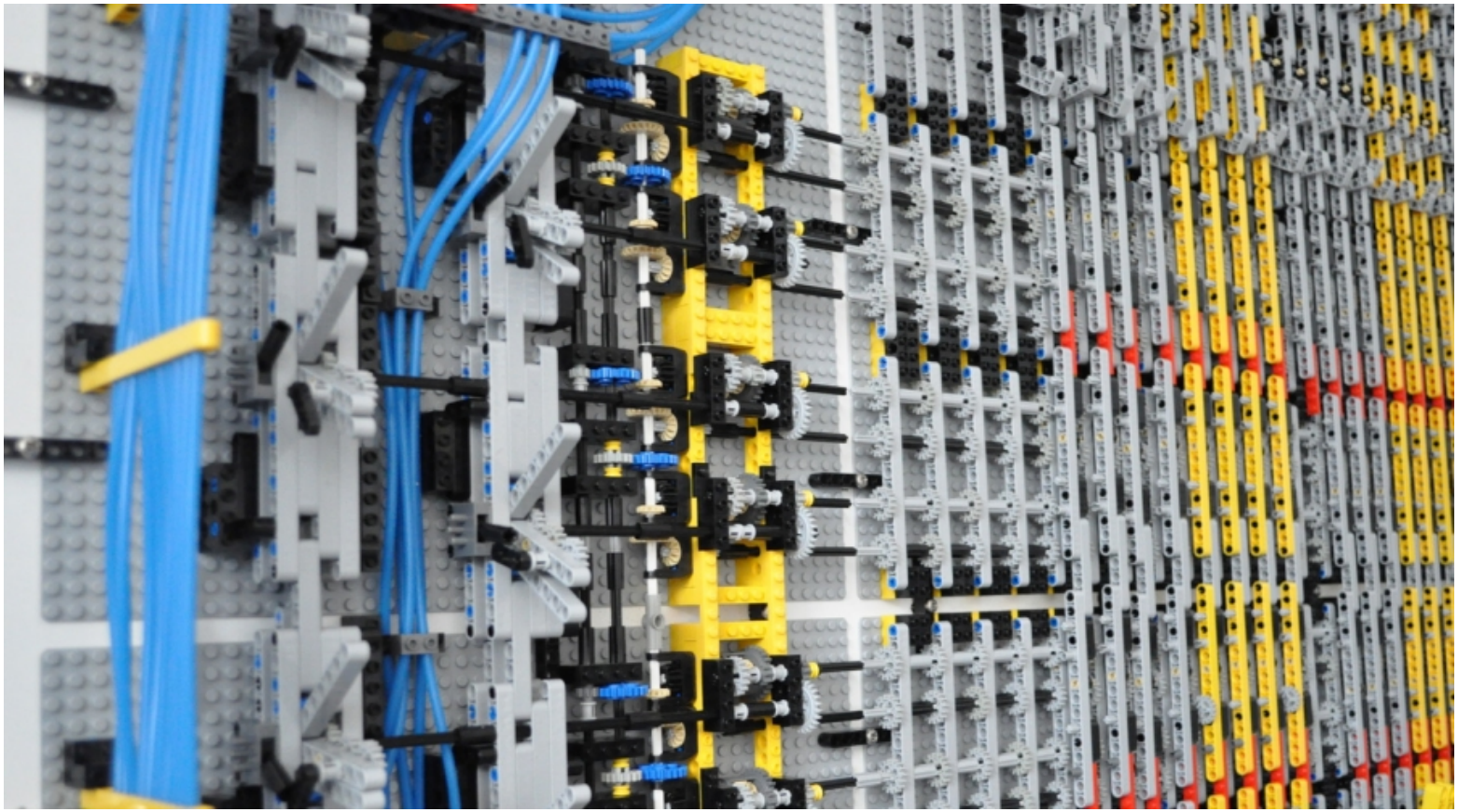
Imaginées par Alan Turing en 1936, pour délimiter ce qu'un d'algorithme est capable de faire ou non.



Modèle théorique : pas de limite ni en temps ni en espace de calcul.

Machines de Turing (I)

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

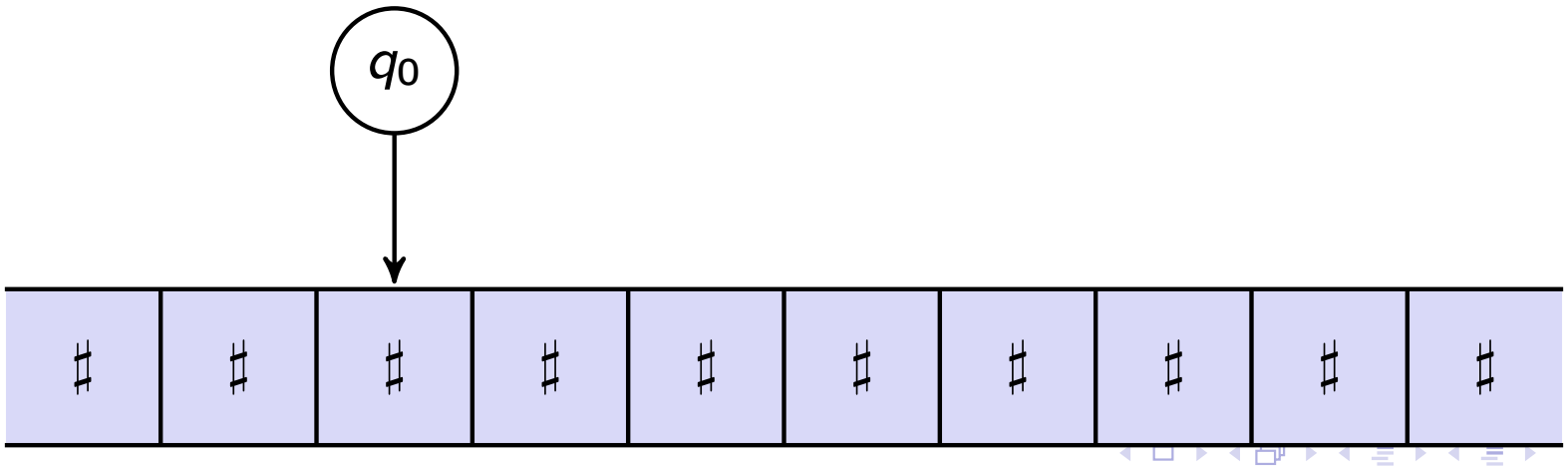


Modèle théorique : pas de limite ni en temps ni en espace de calcul.

Machines de Turing (I)

Un exemple :

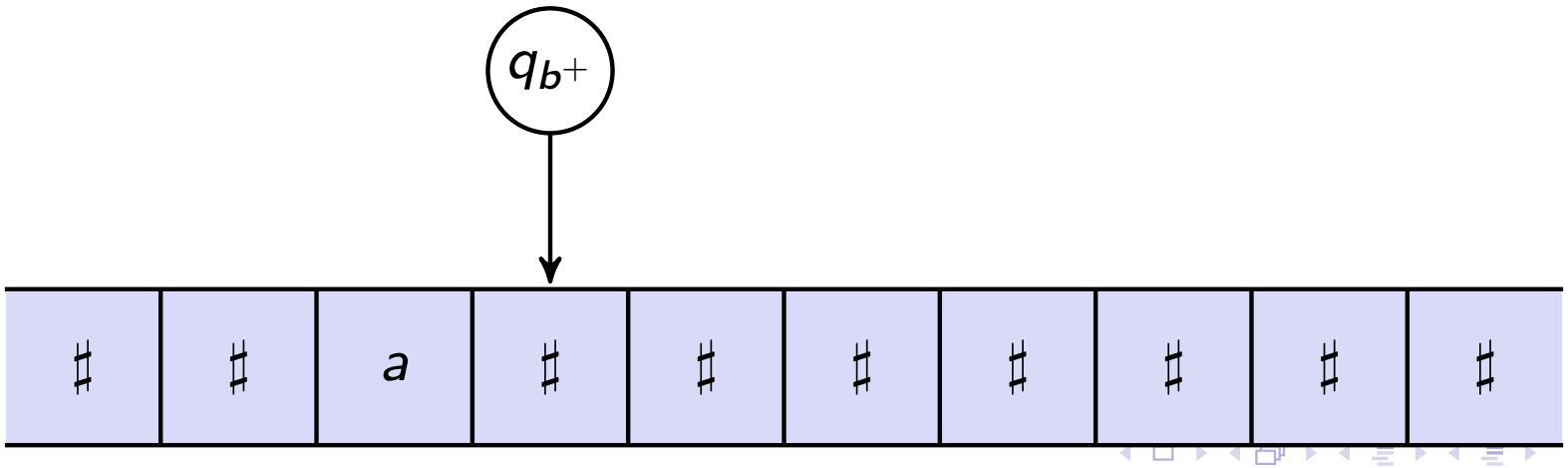
$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+},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



Machines de Turing (I)

Un exemple :

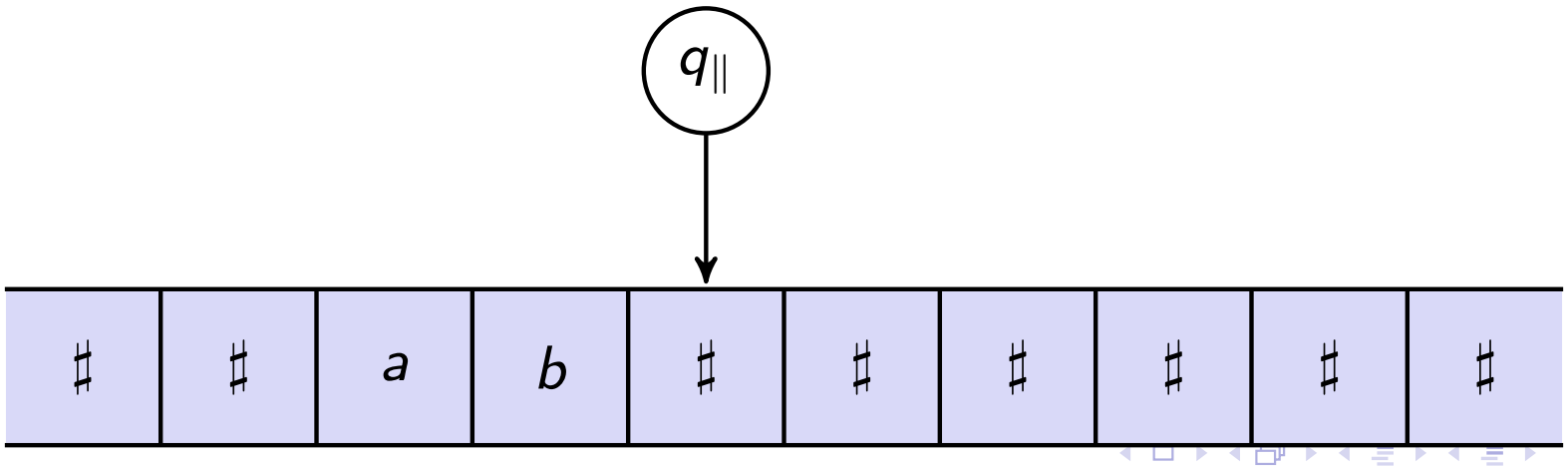
$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+},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



Machines de Turing (I)

Un exemple :

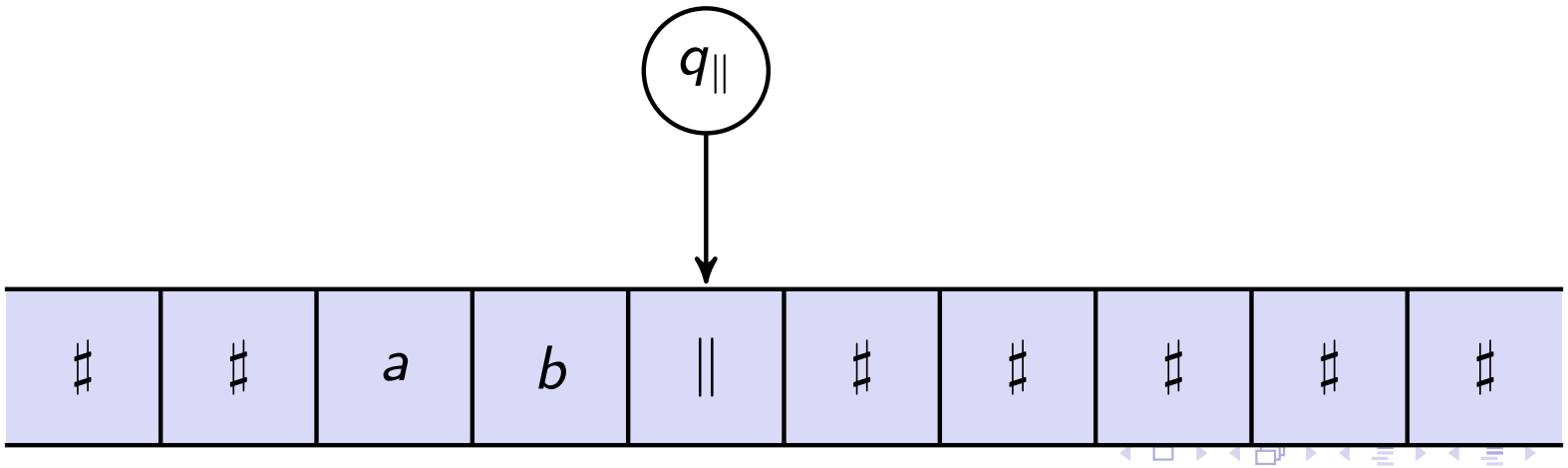
$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+},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



Machines de Turing (I)

Un exemple :

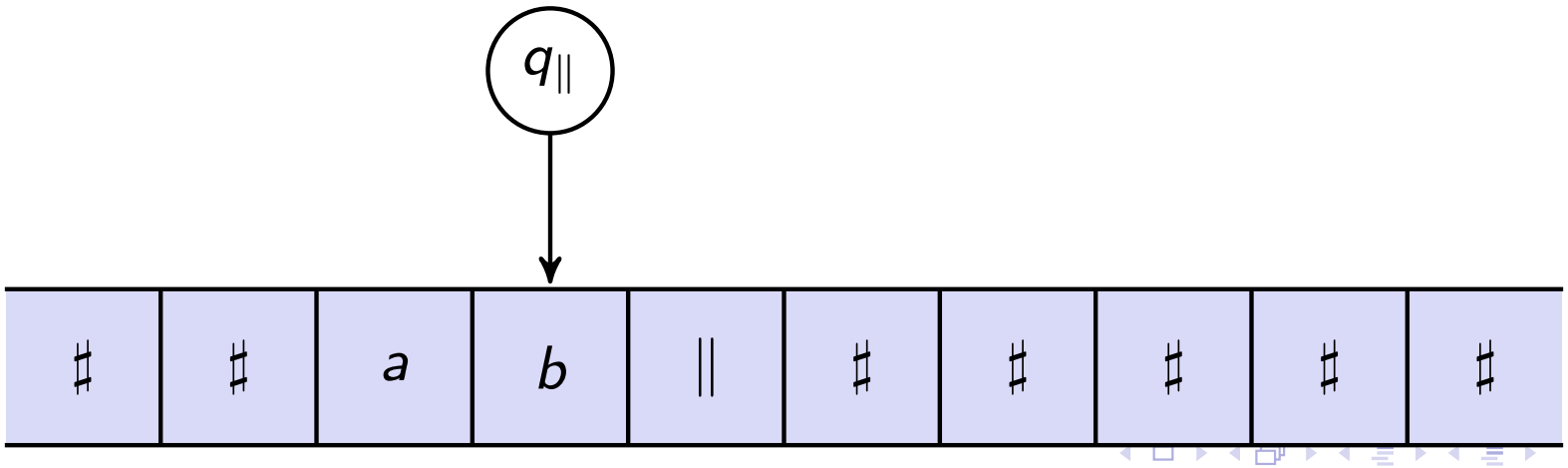
$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+},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



Machines de Turing (I)

Un exemple :

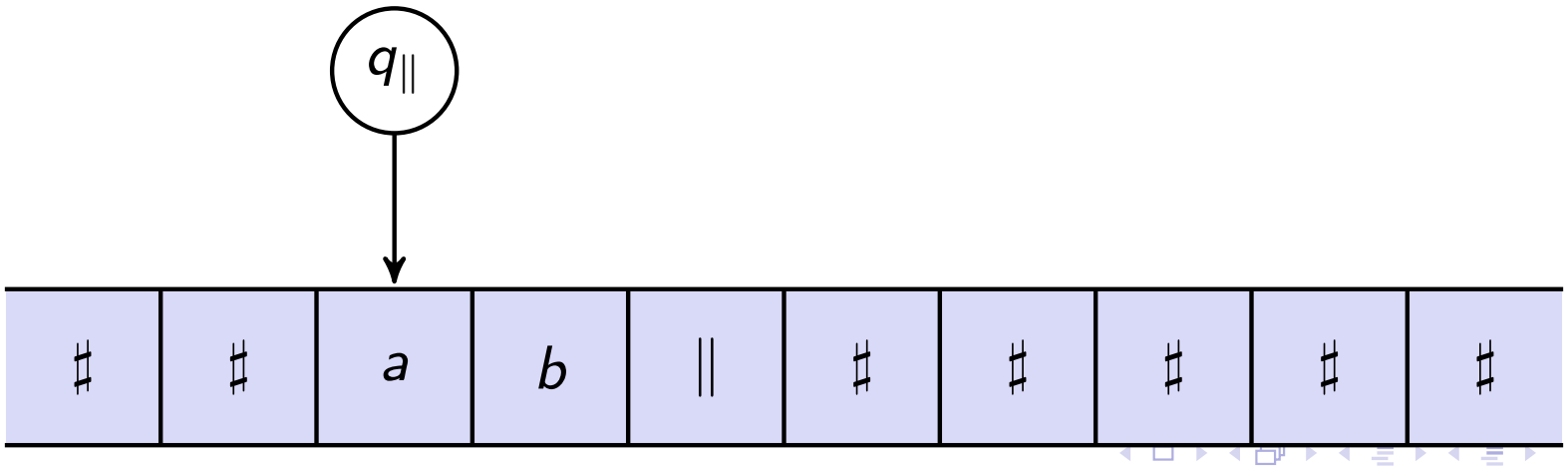
$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+},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



Machines de Turing (I)

Un exemple :

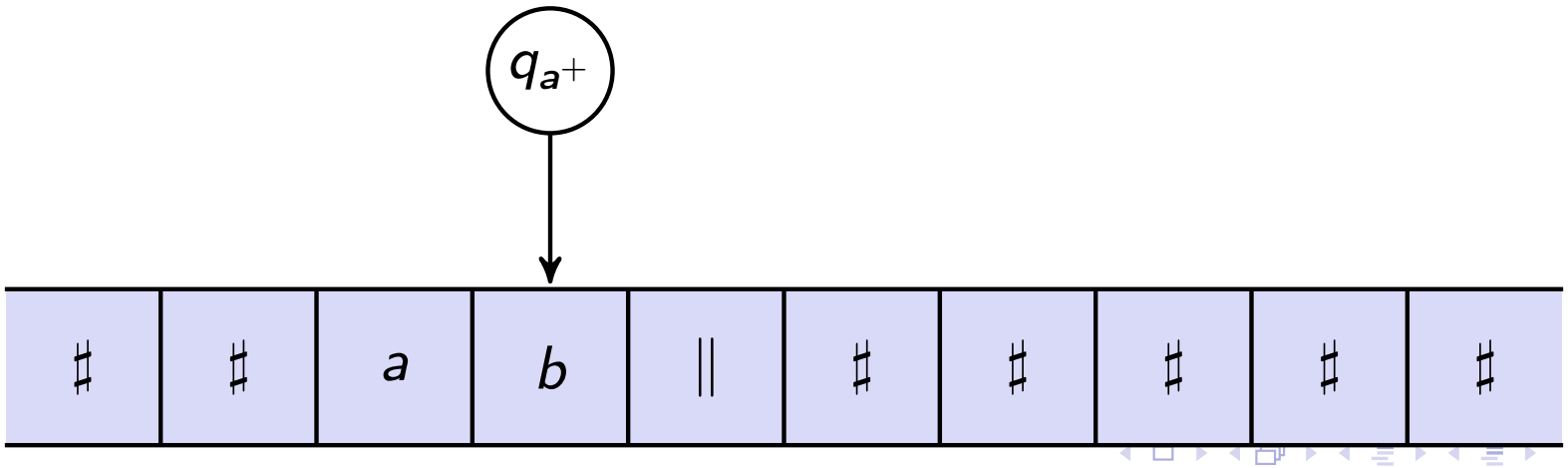
$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+},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



Machines de Turing (I)

Un exemple :

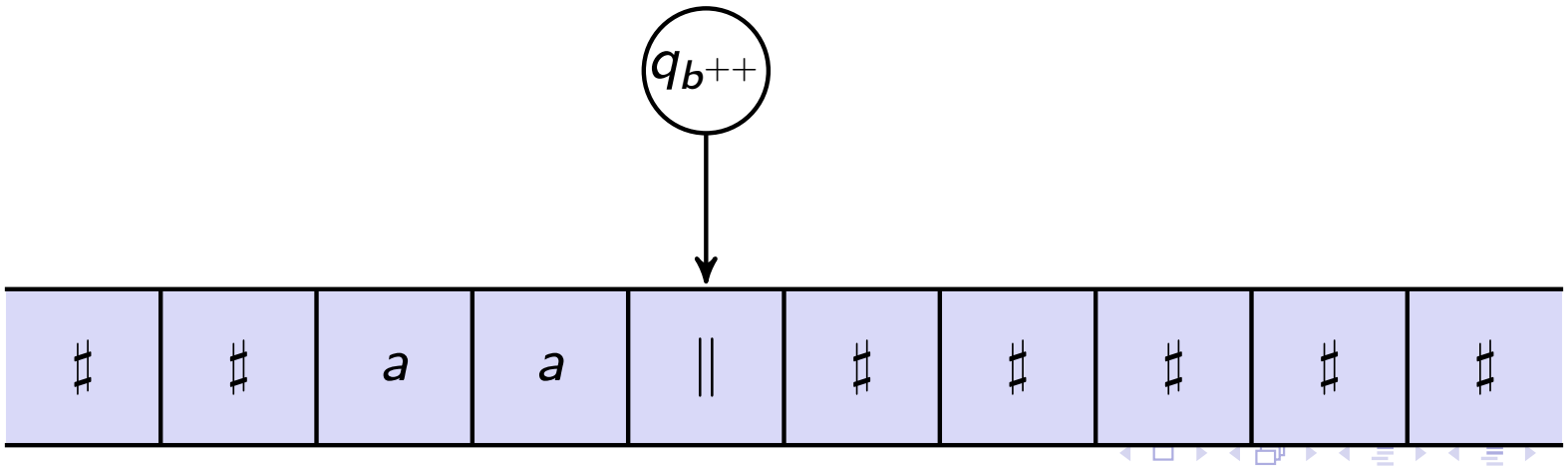
$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+},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



Machines de Turing (I)

Un exemple :

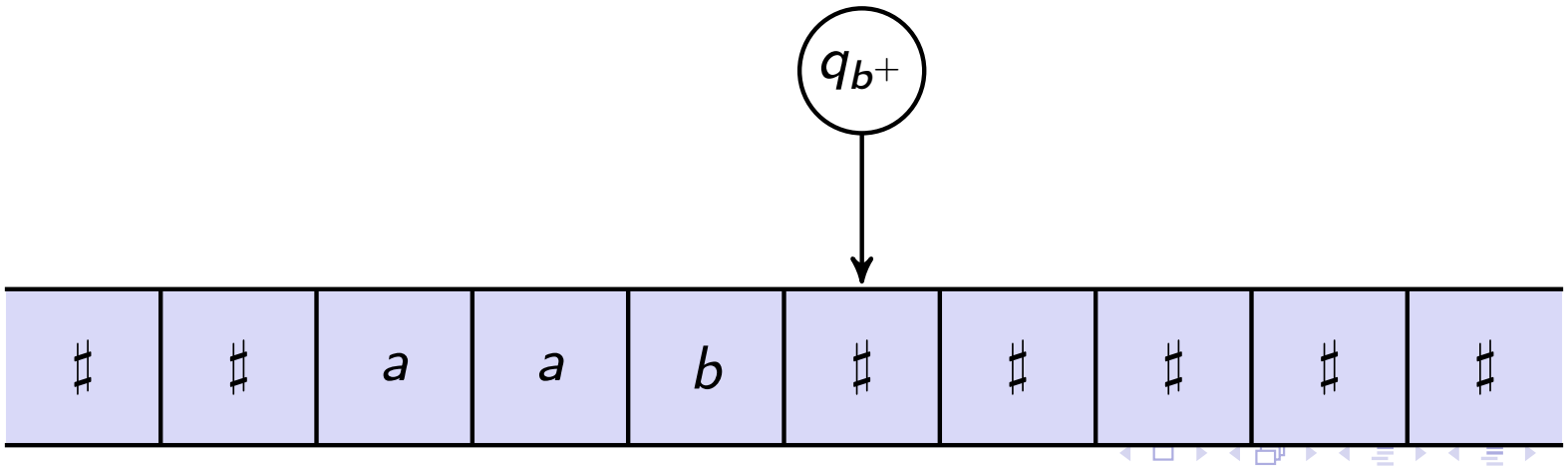
$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+},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



Machines de Turing (I)

Un exemple :

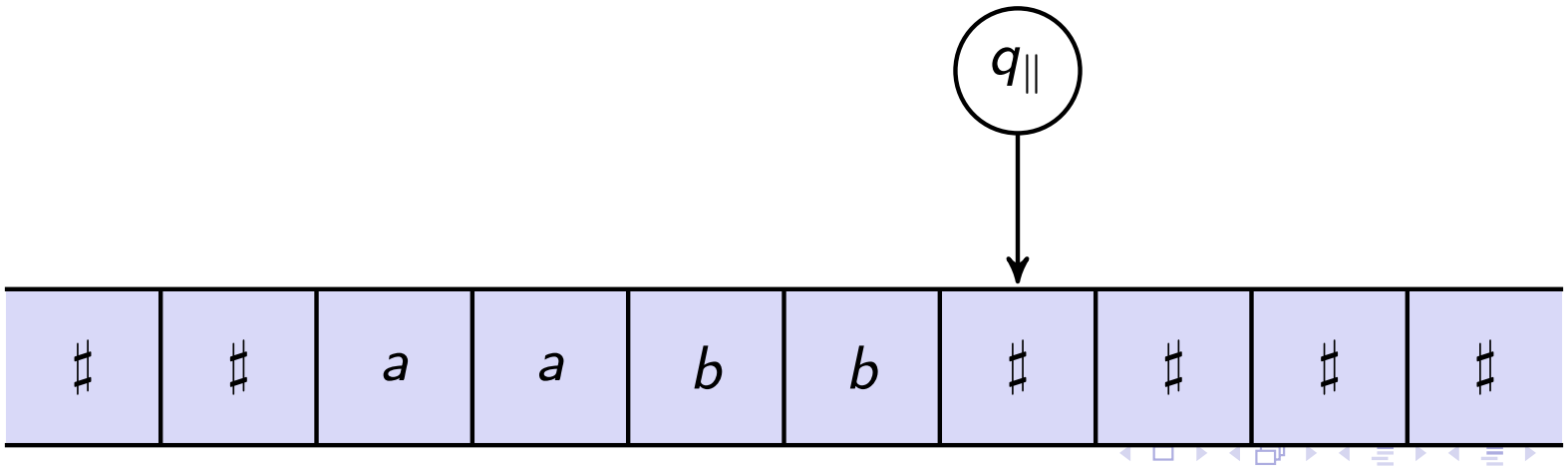
$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+},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



Machines de Turing (I)

Un exemple :

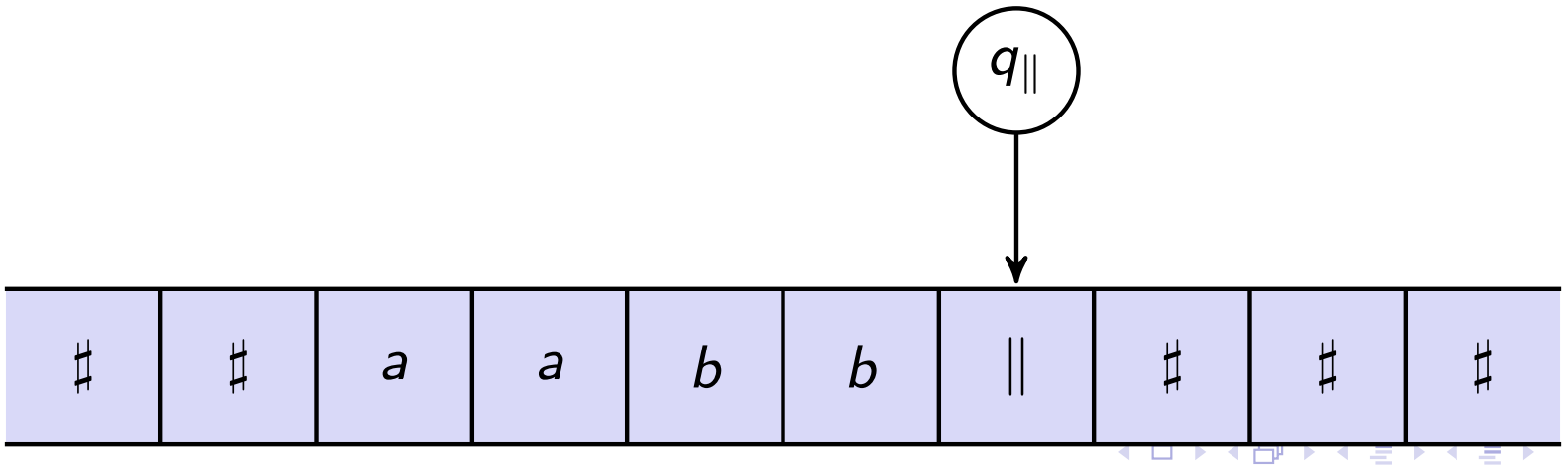
$\delta(q,x)$		Symbole x			
		a	b	$\|\|$	$\sharp$
État q	q_0	$\perp$	$\perp$	$\perp$	$(q_{b^+}, a, \rightarrow)$
	q_{a^+}	$\perp$	$(q_{b^{++}}, a, \rightarrow)$	$\perp$	$\perp$
	q_{b^+}	$\perp$	$\perp$	$\perp$	$(q_{\|\|}, b, \rightarrow)$
	$q_{b^{++}}$	$\perp$	$(q_{b^{++}}, b, \rightarrow)$	$(q_{b^+},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



Machines de Turing (I)

Un exemple :

$\delta(q,x)$		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)

Un autre exemple :

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

Machines de Turing (II)

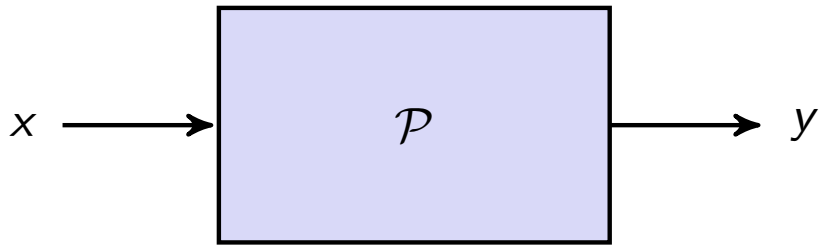
Un autre exemple :

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

Le problème de l'arrêt (I)

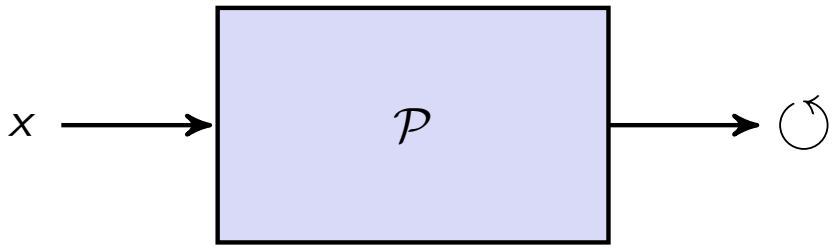
On oublie les machines de Turing, leur ruban, leurs états et leurs transitions. On parle maintenant de **programme**.



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)

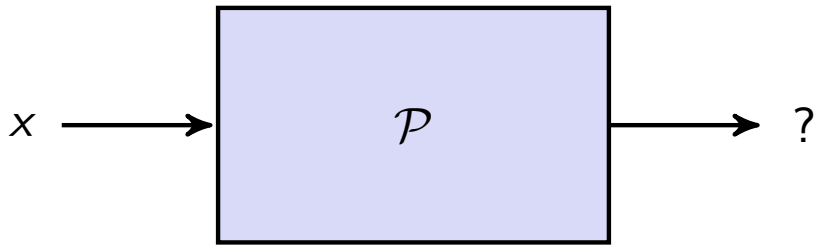
On oublie les machines de Turing, leur ruban, leurs états et leurs transitions. On parle maintenant de **programme**.



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 ↻.

Le problème de l'arrêt (I)

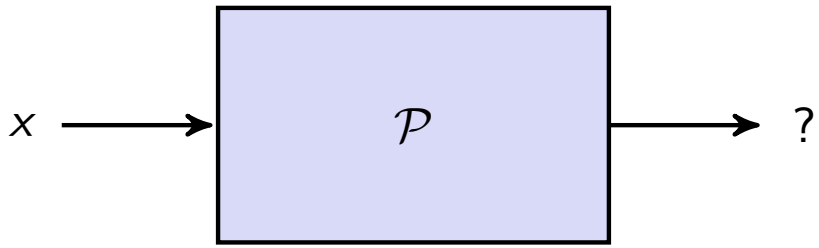
On oublie les machines de Turing, leur ruban, leurs états et leurs transitions. On parle maintenant de **programme**.



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 ↻.

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

Le problème de l'arrêt (II)

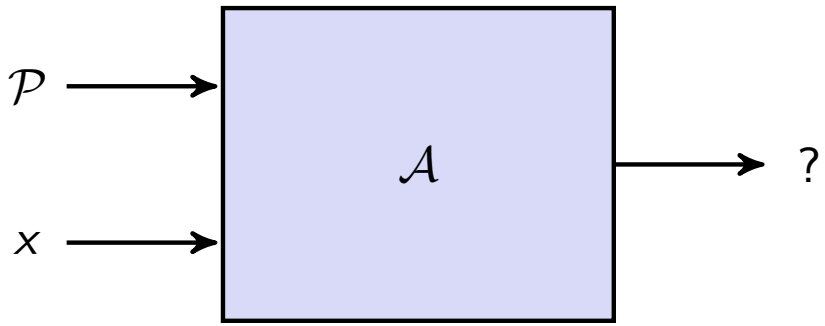
Question

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

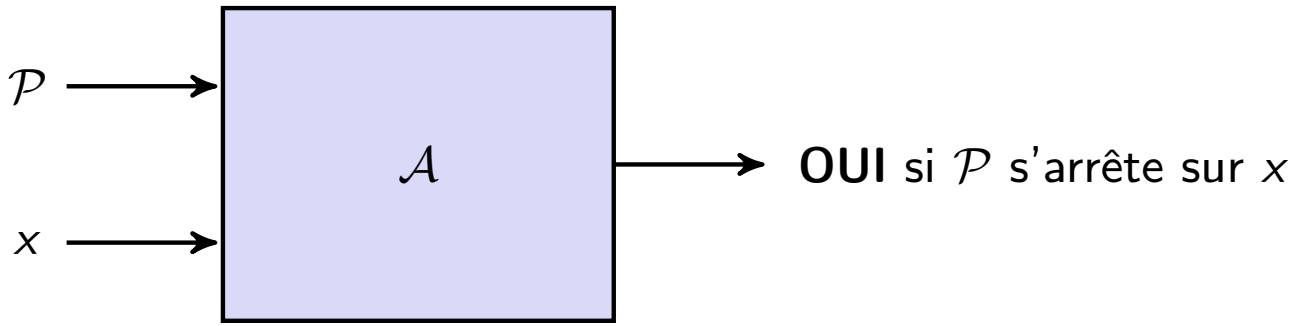
Est-ce qu'il existe un super programme $\mathcal{A}$ qui peut me dire si un programme $\mathcal{P}$ s'arrête sur une entrée x ?



Le problème de l'arrêt (II)

Question

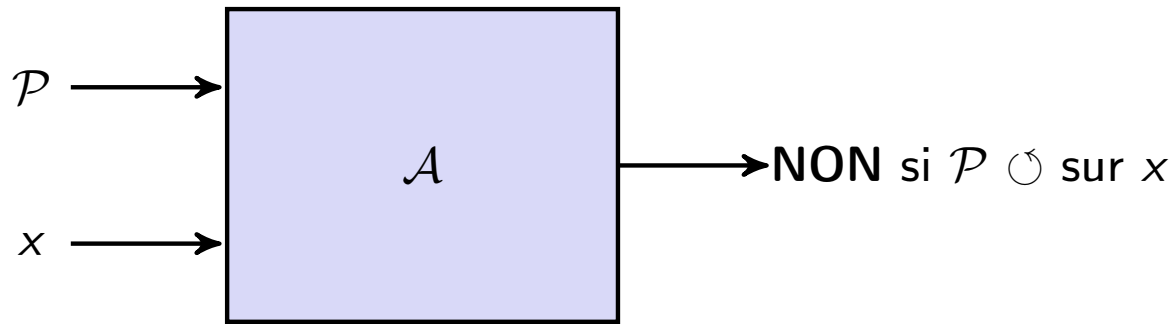
Est-ce qu'il existe un super programme $\mathcal{A}$ qui peut me dire si un programme $\mathcal{P}$ s'arrête sur une entrée x ?



Le problème de l'arrêt (II)

Question

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

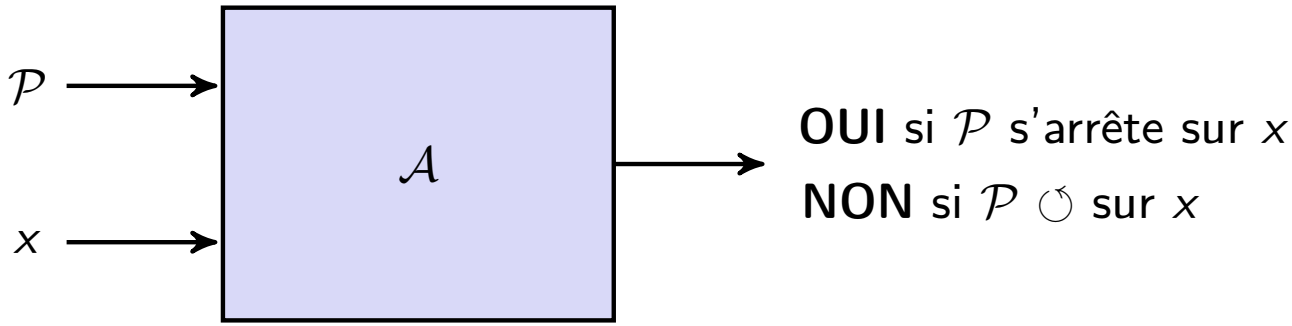


Le problème de l'arrêt (III)

On va montrer que le super-programme $\mathcal{A}$ n'existe pas, en raisonnant par **l'absurde**. On suppose que $\mathcal{A}$ existe,...

Le problème de l'arrêt (III)

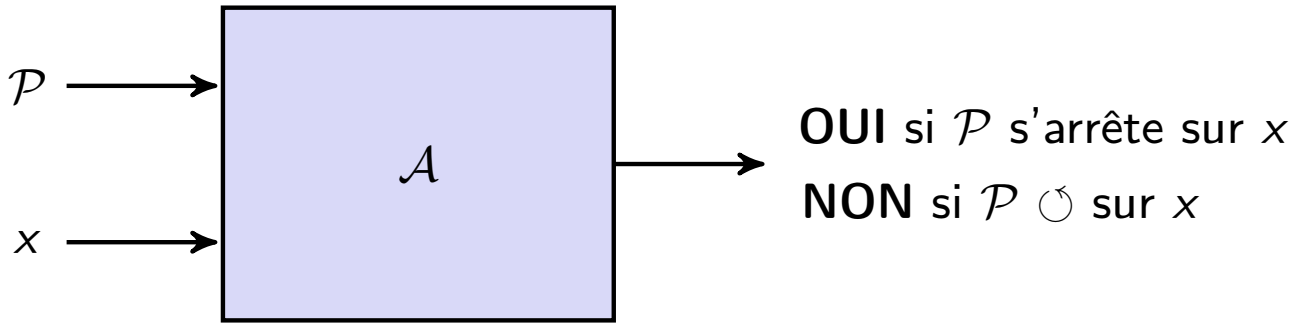
On va montrer que le super-programme $\mathcal{A}$ n'existe pas, en raisonnant par **l'absurde**. On suppose que $\mathcal{A}$ existe,...



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

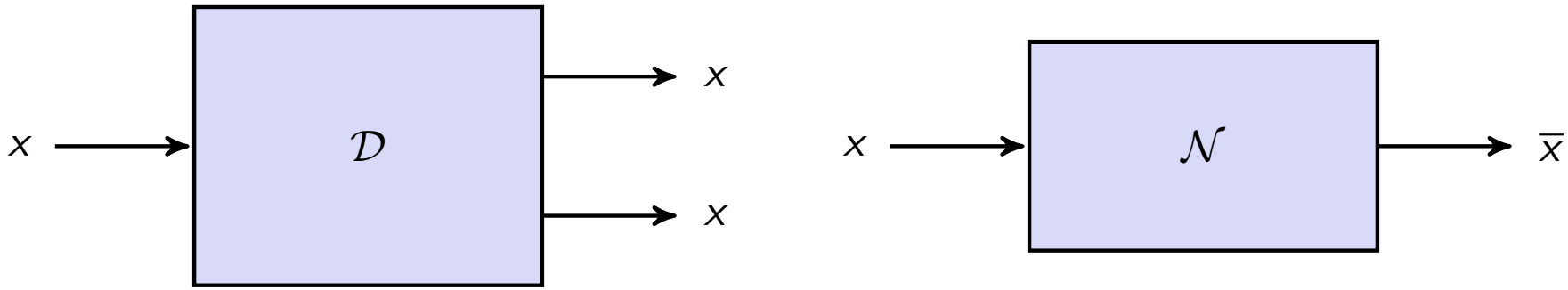
Le problème de l'arrêt (III)

On va montrer que le super-programme $\mathcal{A}$ n'existe pas, en raisonnant par **l'absurde**. On suppose que $\mathcal{A}$ existe,...



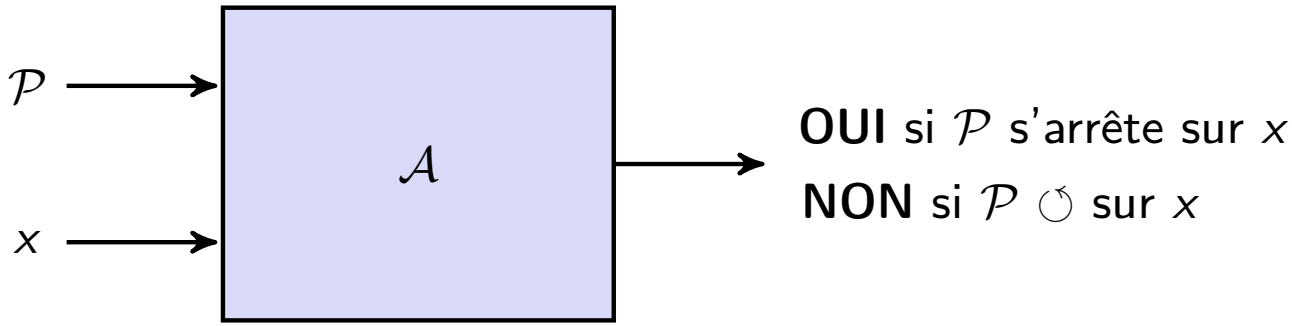
et que $\mathcal{A}$ est bien un super-programme, c'est-à-dire qu'il ne se trompe jamais !

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



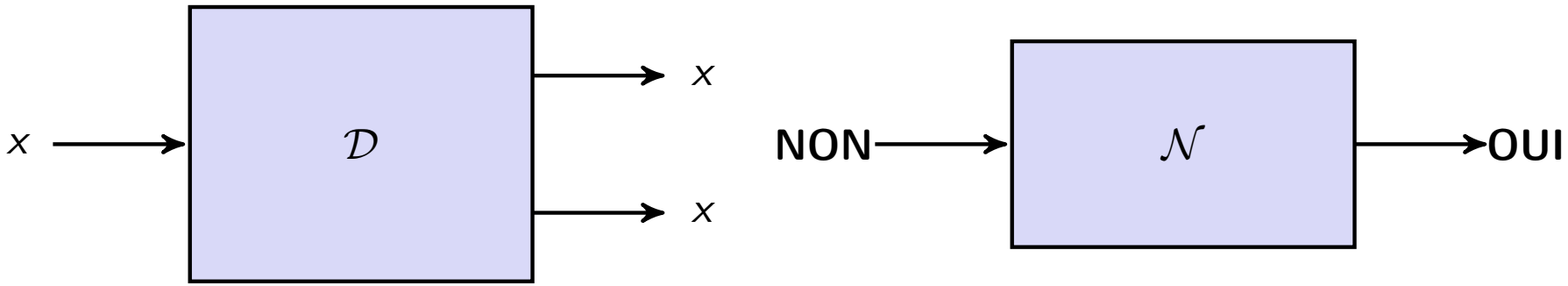
Le problème de l'arrêt (III)

On va montrer que le super-programme $\mathcal{A}$ n'existe pas, en raisonnant par **l'absurde**. On suppose que $\mathcal{A}$ existe,...



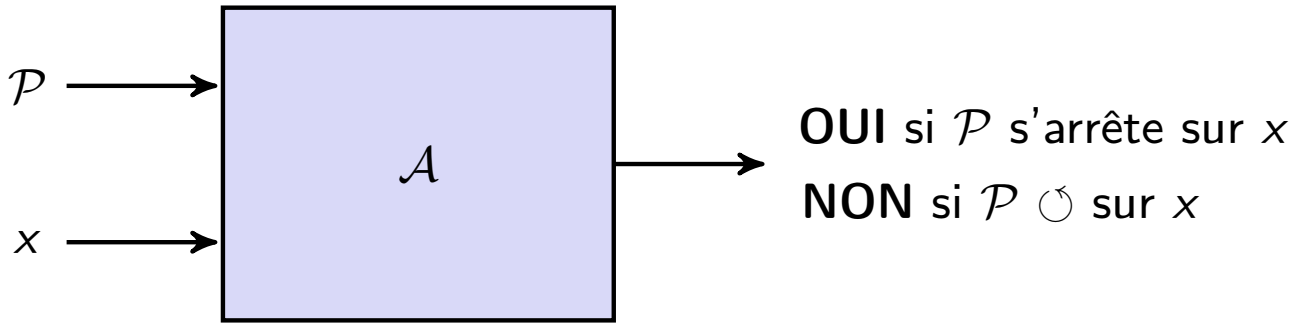
et que $\mathcal{A}$ est bien un super-programme, c'est-à-dire qu'il ne se trompe jamais !

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



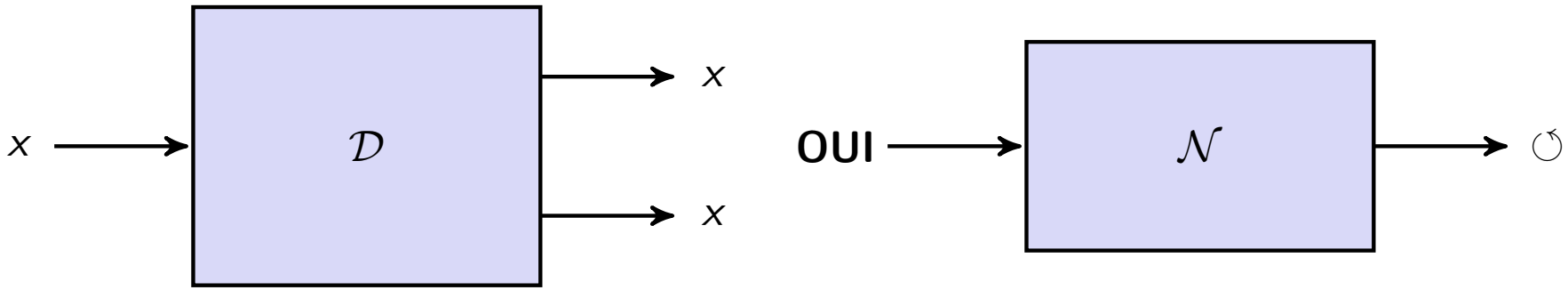
Le problème de l'arrêt (III)

On va montrer que le super-programme $\mathcal{A}$ n'existe pas, en raisonnant par **l'absurde**. On suppose que $\mathcal{A}$ existe,...



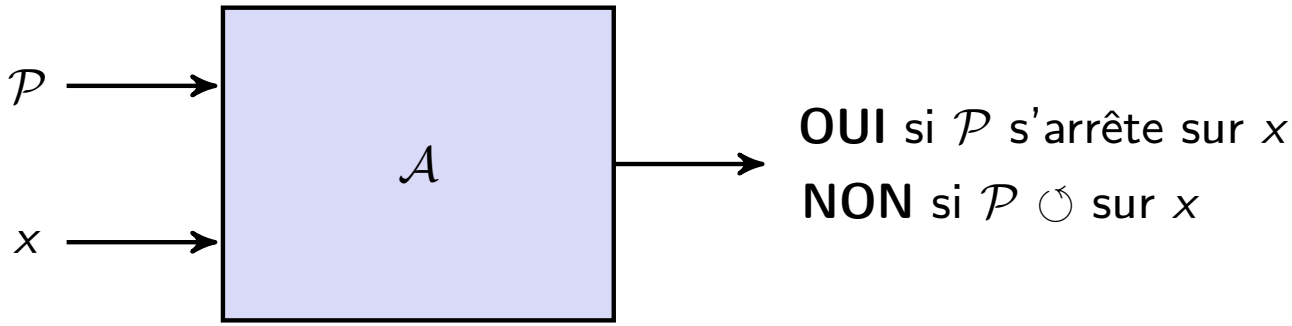
et que $\mathcal{A}$ est bien un super-programme, c'est-à-dire qu'il ne se trompe jamais !

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



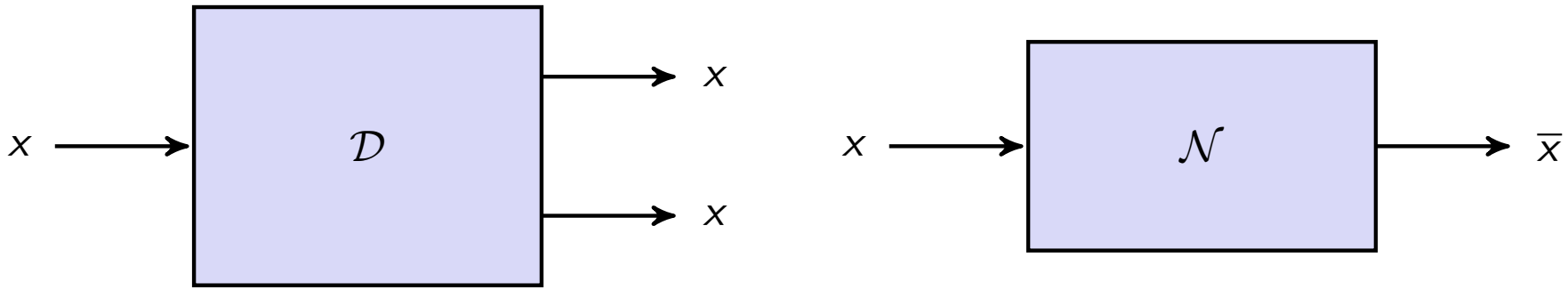
Le problème de l'arrêt (III)

On va montrer que le super-programme $\mathcal{A}$ n'existe pas, en raisonnant par **l'absurde**. On suppose que $\mathcal{A}$ existe,...



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

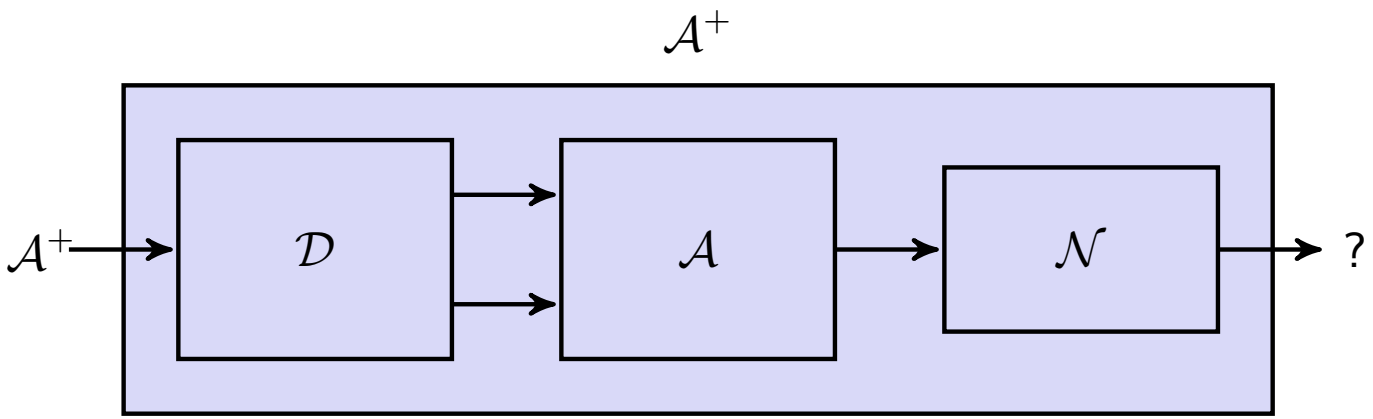
On utilise deux programmes auxiliaires $\mathcal{D}$ et $\mathcal{N}$



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

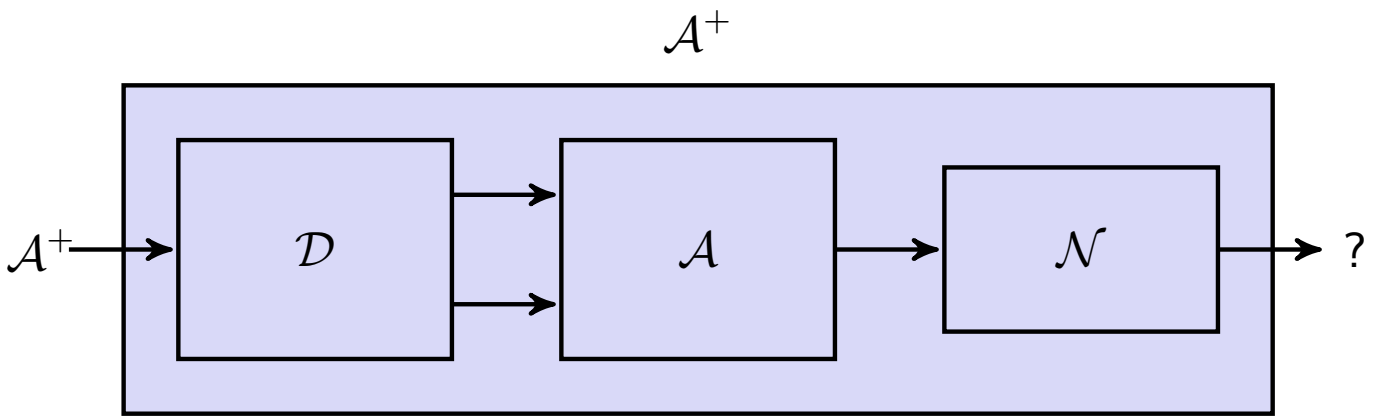
Le problème de l'arrêt (IV)

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



Le problème de l'arrêt (IV)

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

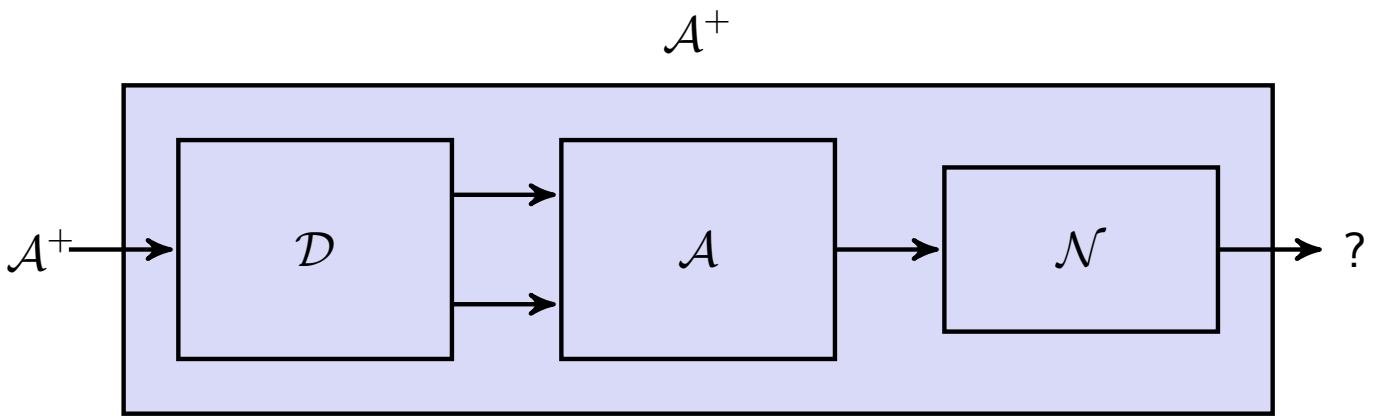


Conclusion :

- 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 (IV)

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



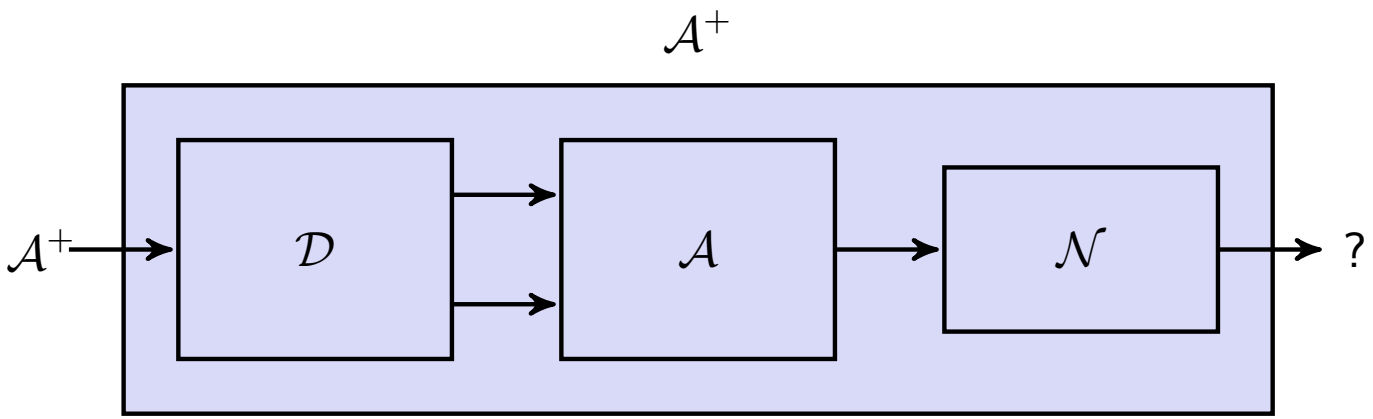
Conclusion :

- 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 (IV)

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**.

Théorème

Il n'existe pas de programme/d'algorithme qui décide si une machine de Turing machine $\mathcal{M}$ s'arrête sur l'entrée vide.

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.

Le problème du Domino : rappel

Problème du Domino

Entrée : Un jeu fini de tuiles de Wang τ
Sortie : **Oui** s'il existe un pavage valide par τ ($X_\tau \neq \emptyset$), **Non** sinon ($X_\tau = \emptyset$).

Le problème du Domino : rappel

Problème du Domino

Entrée : Un jeu fini de tuiles de Wang τ
Sortie : **Oui** s'il existe un pavage valide par τ ($X_\tau \neq \emptyset$), **Non** sinon ($X_\tau = \emptyset$).

On suppose qu'il existe un algorithme qui résout le problème du Domino.

Le problème du Domino : rappel

Problème du Domino

Entrée : Un jeu fini de tuiles de Wang τ
Sortie : **Oui** s'il existe un pavage valide par τ ($X_\tau \neq \emptyset$), **Non** sinon ($X_\tau = \emptyset$).

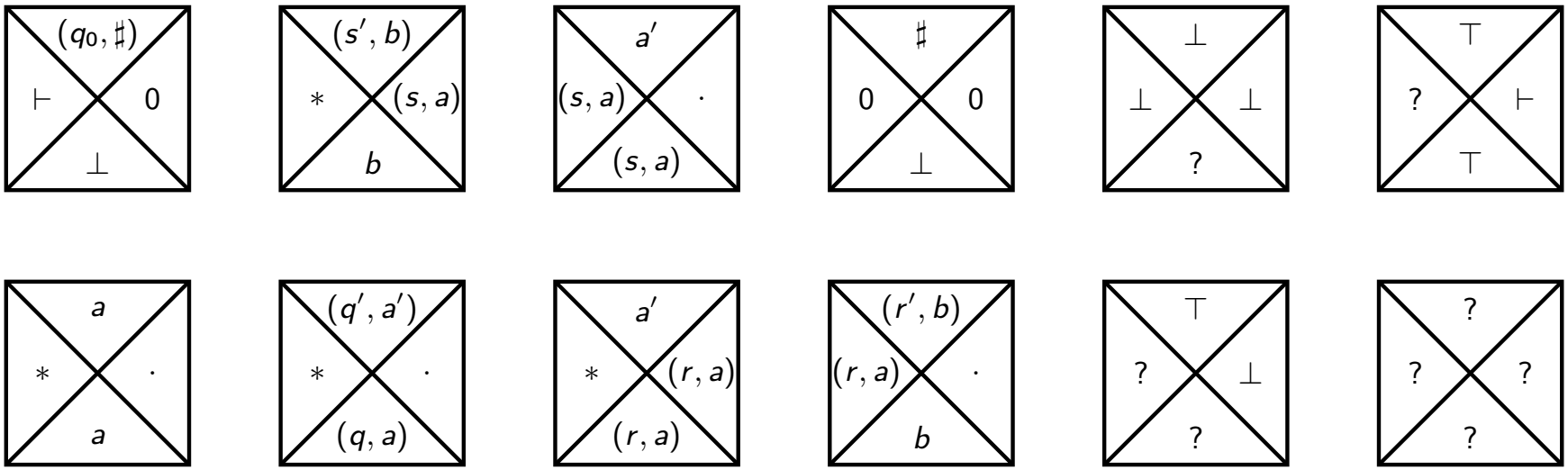
On suppose qu'il existe un algorithme qui résout le problème du Domino.

On veut montrer qu'alors, il existe un algorithme qui résout le problème de l'arrêt.

Machines de Turing et tuiles de Wang

On peut coder des calculs de machine de Turing dans des tuiles de Wang :

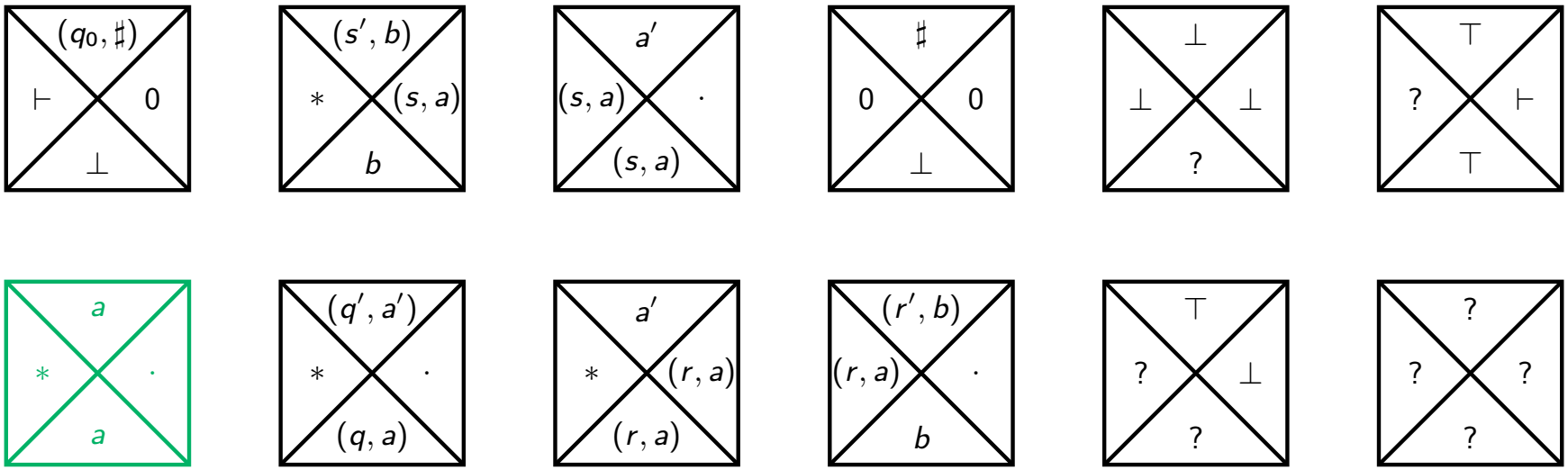
- pas de tête de calcul
- configuration initiale $({}^{\infty}\sharp^{\infty}, q_0)$
- $\delta(q, a) = (q', a', .)$
- $\delta(r, a) = (r', a', \rightarrow)$
- $\delta(s, a) = (s', a', \leftarrow)$



Machines de Turing et tuiles de Wang

On peut coder des calculs de machine de Turing dans des tuiles de Wang :

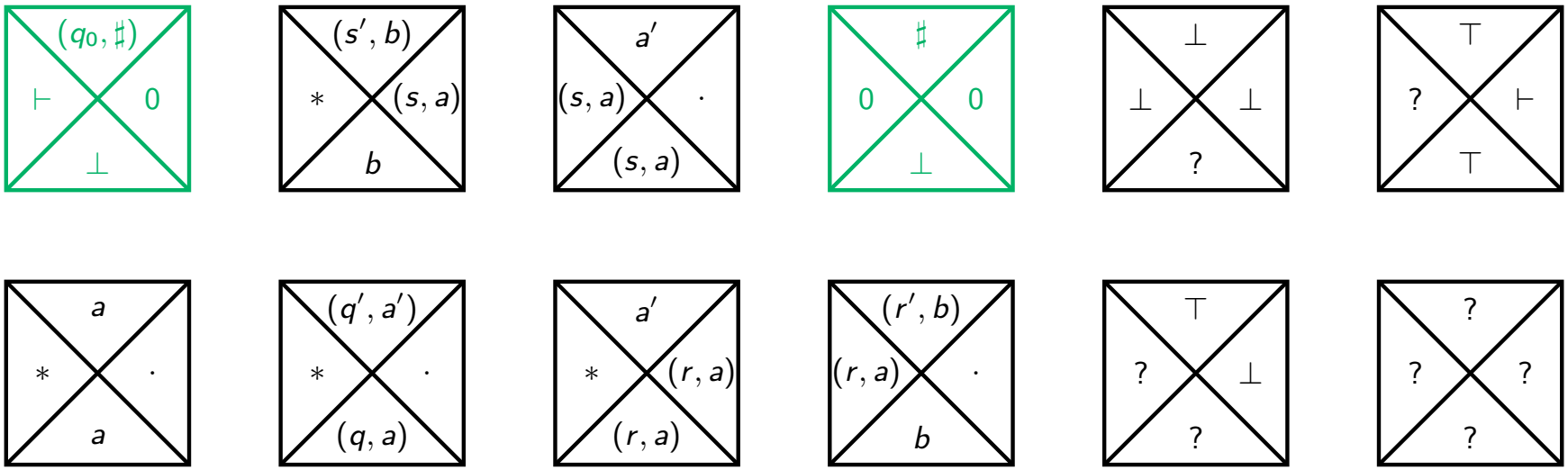
- pas de tête de calcul
- configuration initiale $({}^{\infty}\sharp^{\infty}, q_0)$
- $\delta(q, a) = (q', a', .)$
- $\delta(r, a) = (r', a', \rightarrow)$
- $\delta(s, a) = (s', a', \leftarrow)$



Machines de Turing et tuiles de Wang

On peut coder des calculs de machine de Turing dans des tuiles de Wang :

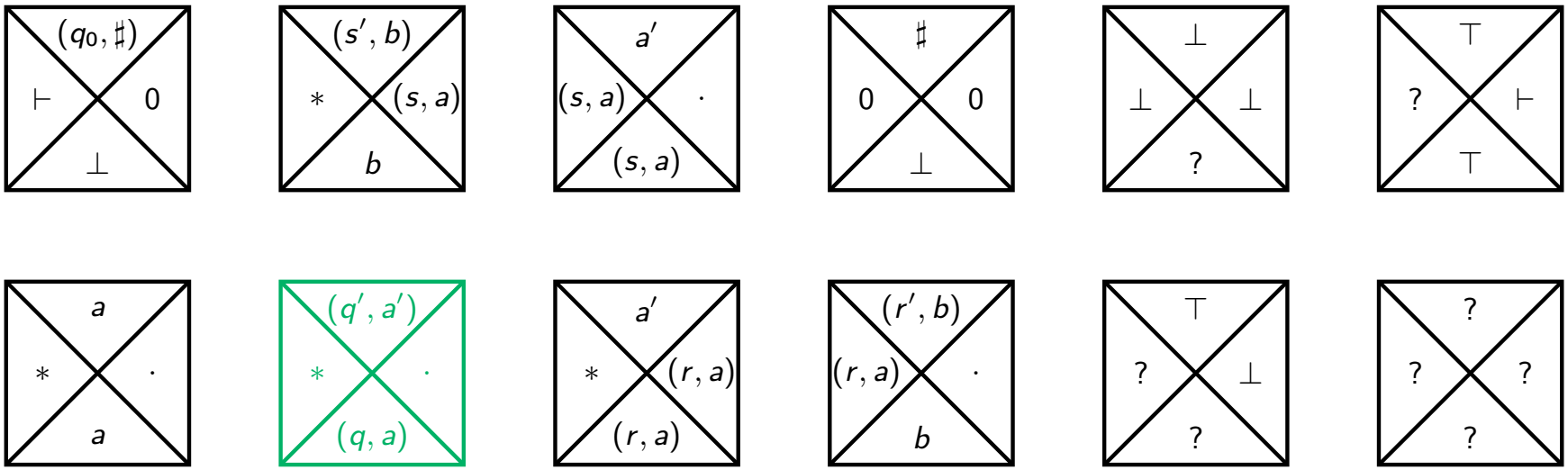
- pas de tête de calcul
- configuration initiale $({}^{\infty}\sharp^{\infty}, q_O)$
- $\delta(q, a) = (q', a', .)$
- $\delta(r, a) = (r', a', \rightarrow)$
- $\delta(s, a) = (s', a', \leftarrow)$



Machines de Turing et tuiles de Wang

On peut coder des calculs de machine de Turing dans des tuiles de Wang :

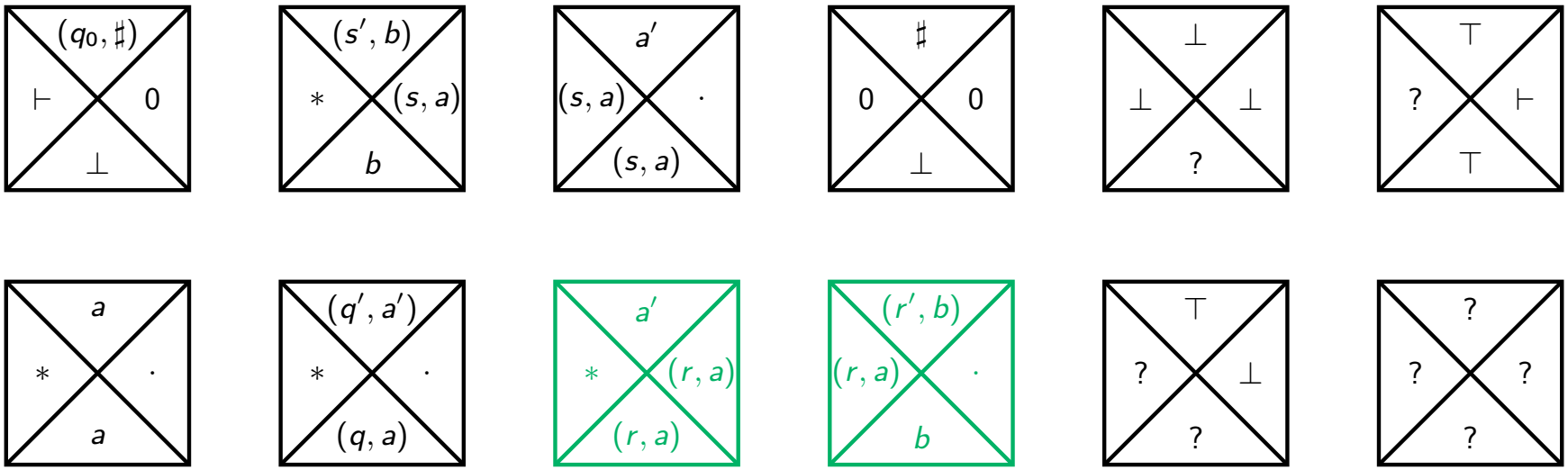
- pas de tête de calcul
- configuration initiale $({}^{\infty}\sharp^{\infty}, q_0)$
- $\delta(q, a) = (q', a', .)$
- $\delta(r, a) = (r', a', \rightarrow)$
- $\delta(s, a) = (s', a', \leftarrow)$



Machines de Turing et tuiles de Wang

On peut coder des calculs de machine de Turing dans des tuiles de Wang :

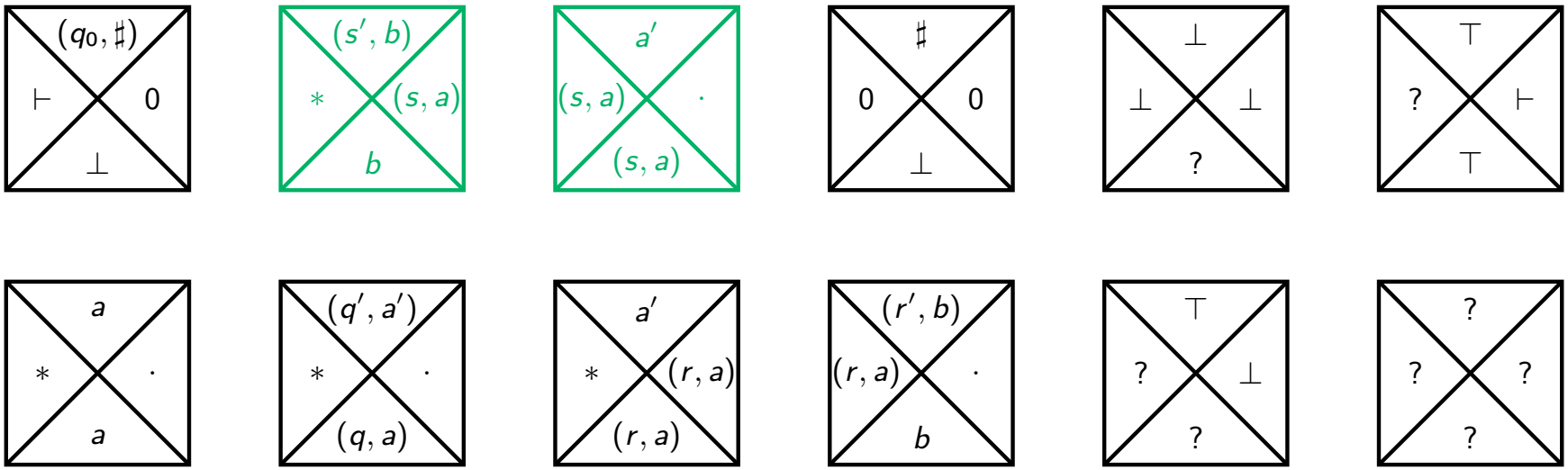
- pas de tête de calcul
- configuration initiale $({}^{\infty}\sharp^{\infty}, q_0)$
- $\delta(q, a) = (q', a', .)$
- $\delta(r, a) = (r', a', \rightarrow)$
- $\delta(s, a) = (s', a', \leftarrow)$



Machines de Turing et tuiles de Wang

On peut coder des calculs de machine de Turing dans des tuiles de Wang :

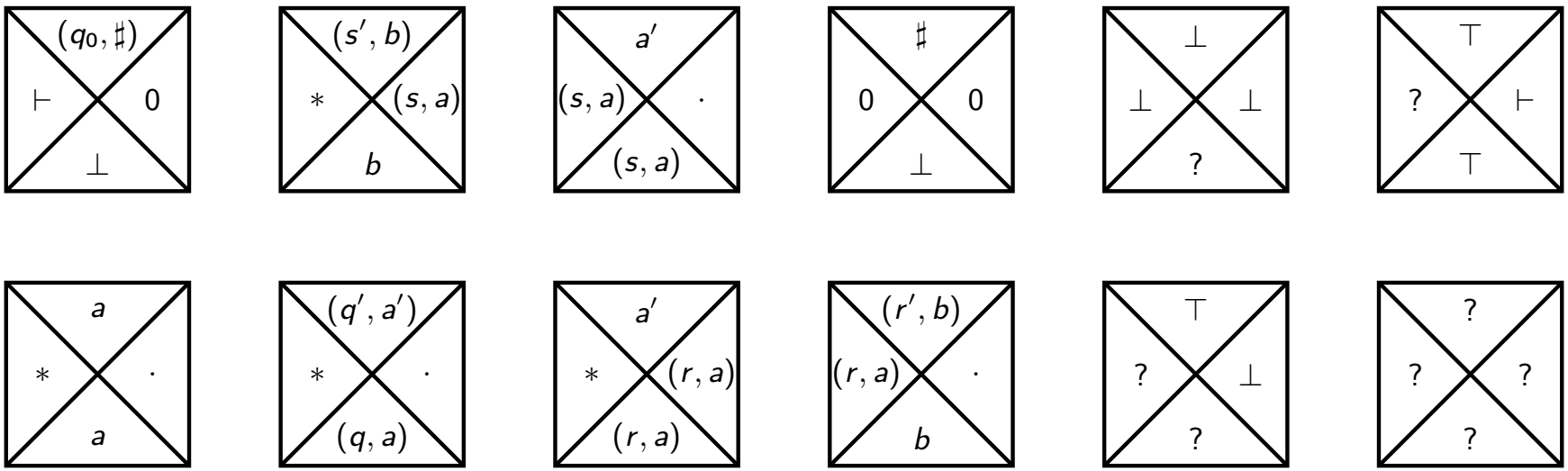
- pas de tête de calcul
- configuration initiale $({}^{\infty}\sharp^{\infty}, q_0)$
- $\delta(q, a) = (q', a', .)$
- $\delta(r, a) = (r', a', \rightarrow)$
- $\delta(s, a) = (s', a', \leftarrow)$



Machines de Turing et tuiles de Wang

On peut coder des calculs de machine de Turing dans des tuiles de Wang :

- pas de tête de calcul
- configuration initiale $({}^{\infty}\sharp^{\infty}, q_0)$
- $\delta(q, a) = (q', a', .)$
- $\delta(r, a) = (r', a', \rightarrow)$
- $\delta(s, a) = (s', a', \leftarrow)$



Ce qu'on veut : τ **admet un pavage ssi** $\mathcal{M}$ **ne s'arrête pas sur l'entrée vide.**

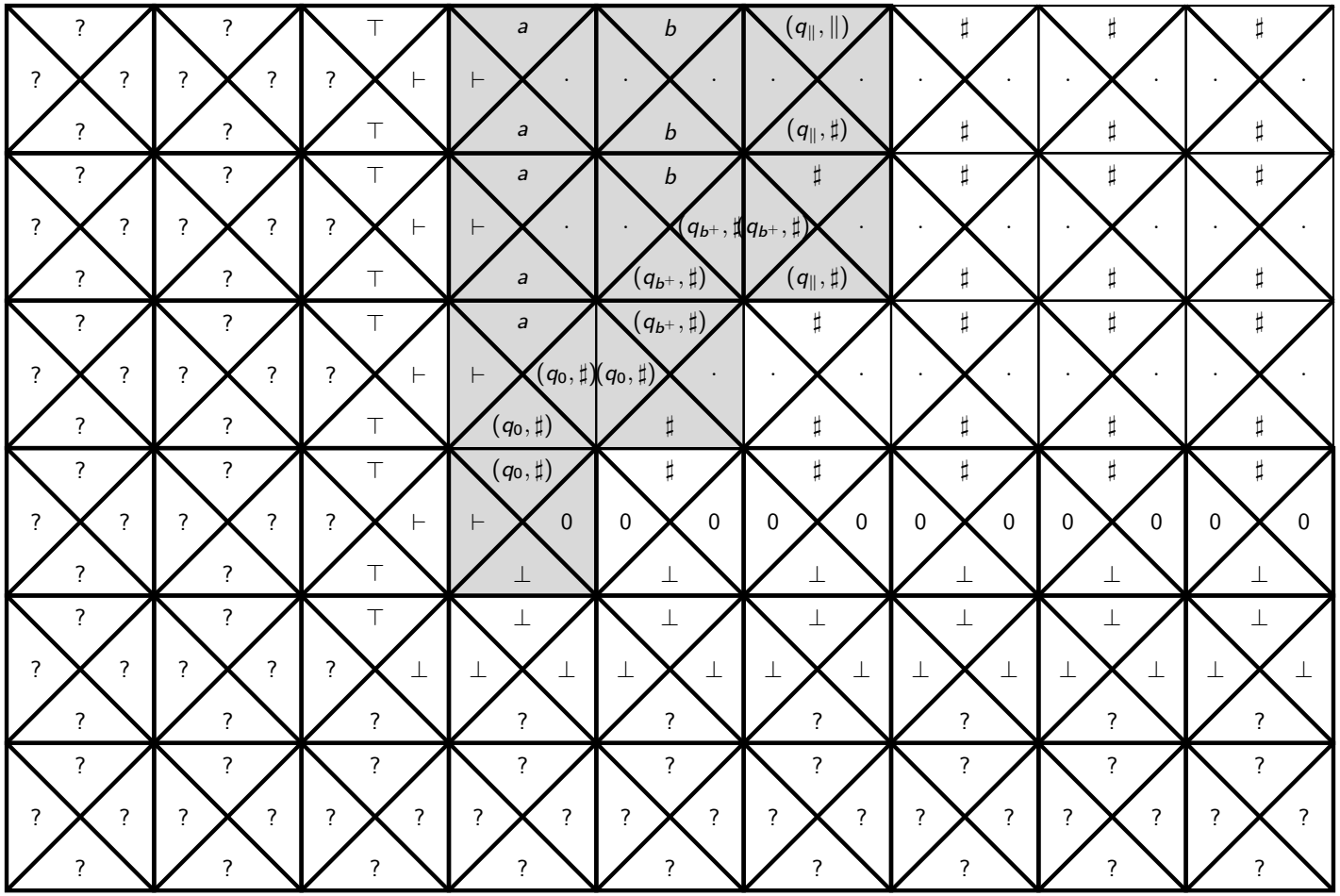
Pavages possibles ?

On **supprime** les tuiles sur lesquelles apparaît un état d'acceptation q_a.

Pavages possibles ?

On **supprime** les tuiles sur lesquelles apparaît un état d'acceptation q_a.

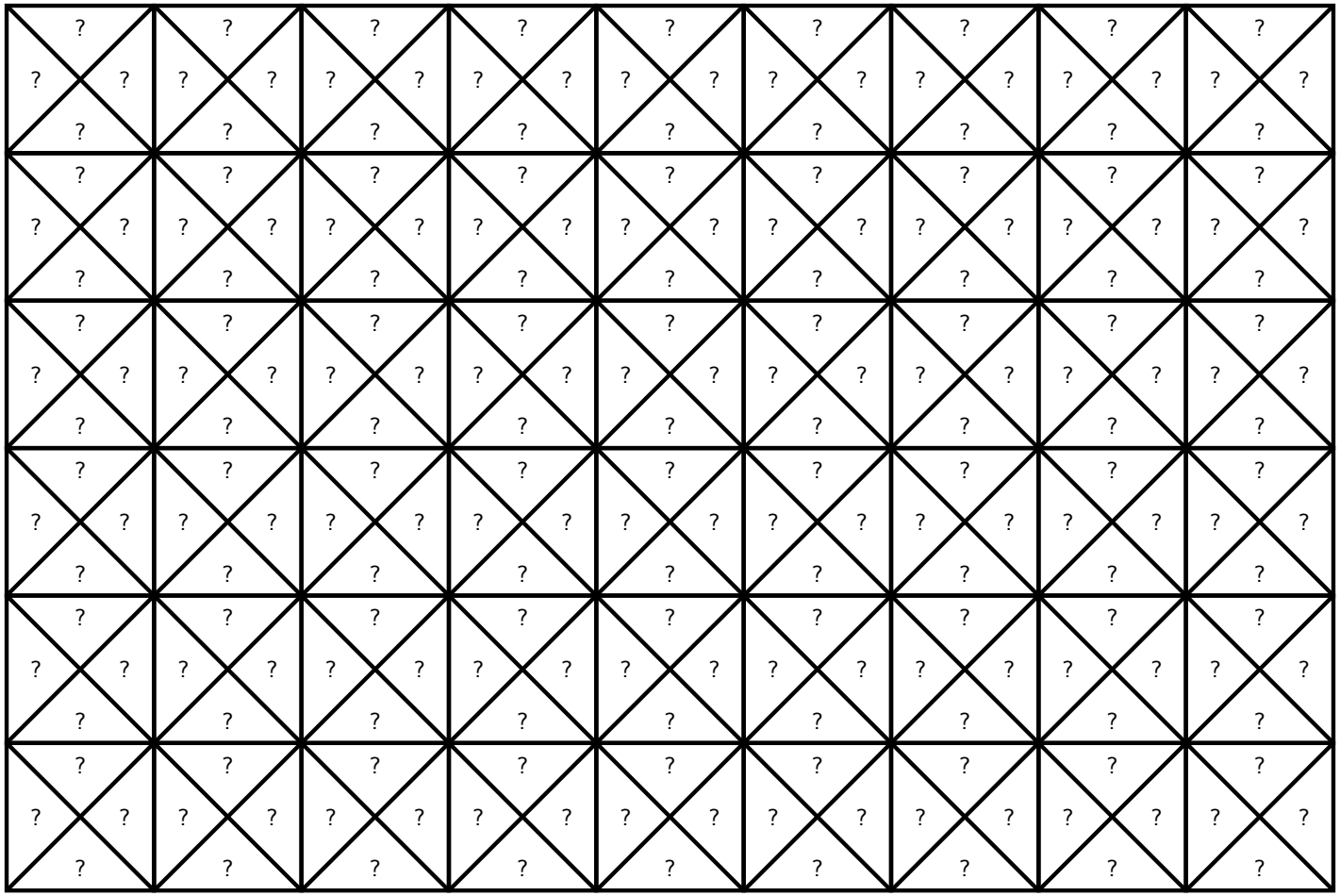
Si $\mathcal{M}$ ne s'arrête pas sur l'entrée vide, il existe un pavage.



Pavages possibles ?

On **supprime** les tuiles sur lesquelles apparaît un état d'acceptation q_a.

Si $\mathcal{M}$ ne s'arrête pas sur l'entrée vide, il existe un pavage. Mais...



Le problème du Domino à origine fixée

Ce que l'on n'a pas montré :

Pas encore un théorème

Le problème du Domino est indécidable.

Le problème du Domino à origine fixée

Ce que l'on n'a pas montré :

Pas encore un théorème

Le problème du Domino est indécidable.

Ce que l'on a montré :

Théorème (Kahr, Moore & Wang 1962, Büchi 1962)

Le problème du Domino à origine fixée est indécidable.

avec

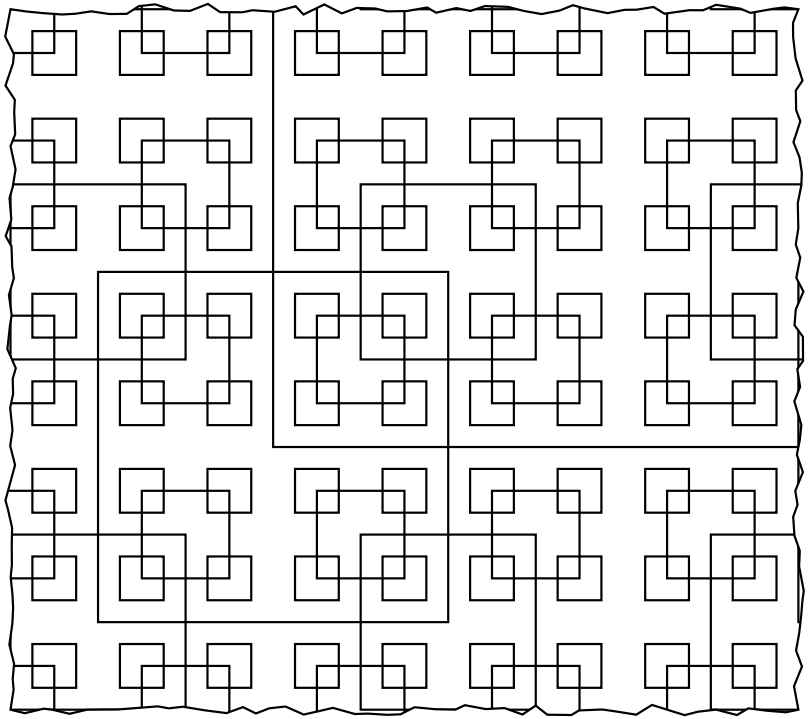
Problème du Domino à origine fixée

Input : Un jeu fini de tuiles de Wang τ, une tuile $t \in \tau$
Output : **Oui** s'il existe un pavage valide par τ avec la tuile t à l'origine, **Non** sinon.

Et le problème du Domino ?

Solution

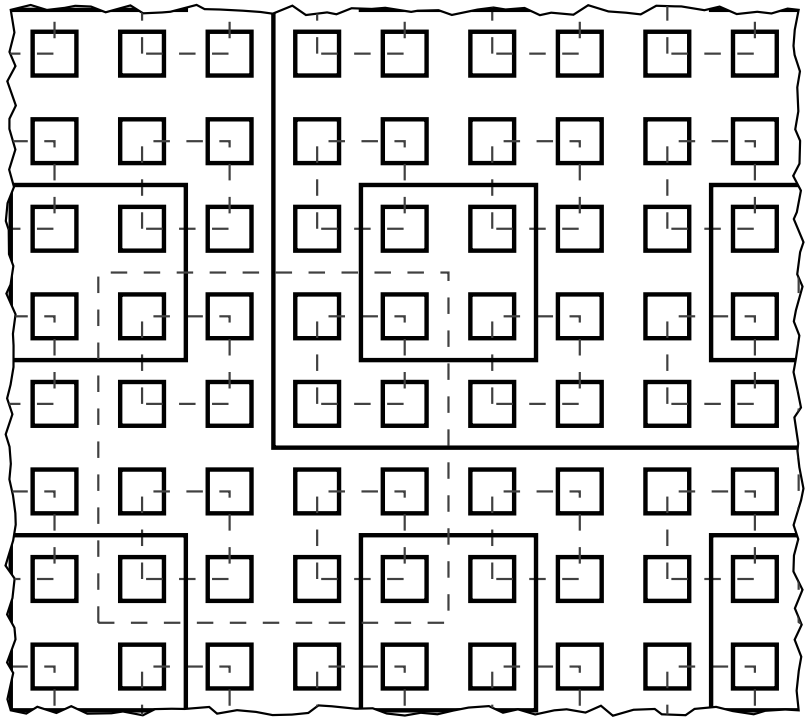
Un peu de machines de Turing, un peu de pavage de Robinson...



Et le problème du Domino ?

Solution

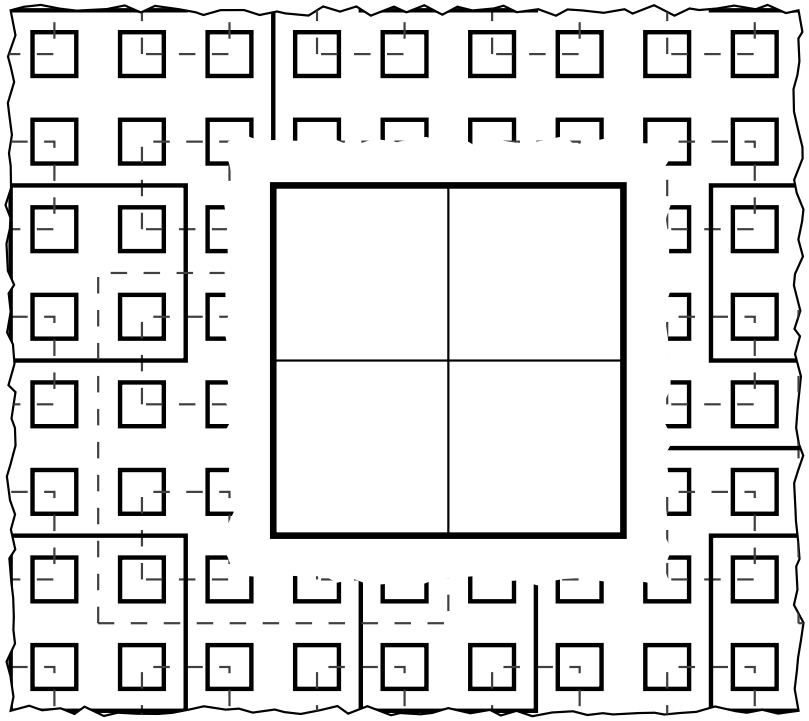
Un peu de machines de Turing, un peu de pavage de Robinson...



Et le problème du Domino ?

Solution

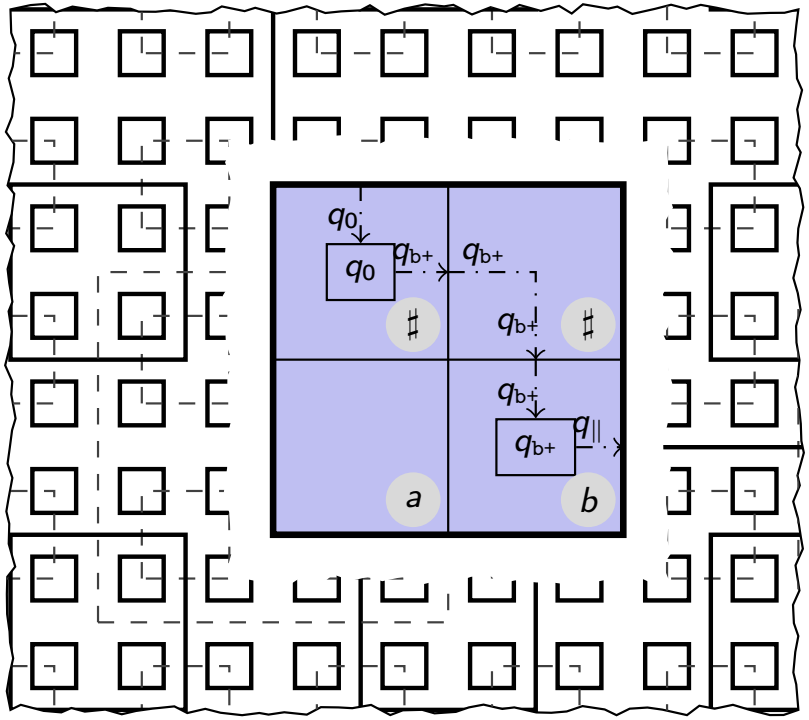
Un peu de machines de Turing, un peu de pavage de Robinson...



Et le problème du Domino ?

Solution

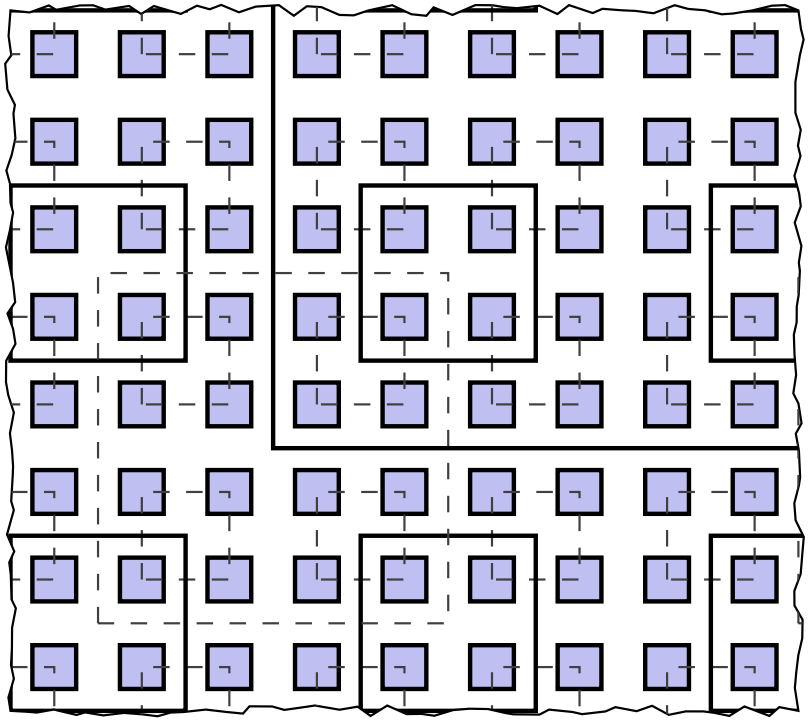
Un peu de machines de Turing, un peu de pavage de Robinson...



Et le problème du Domino ?

Solution

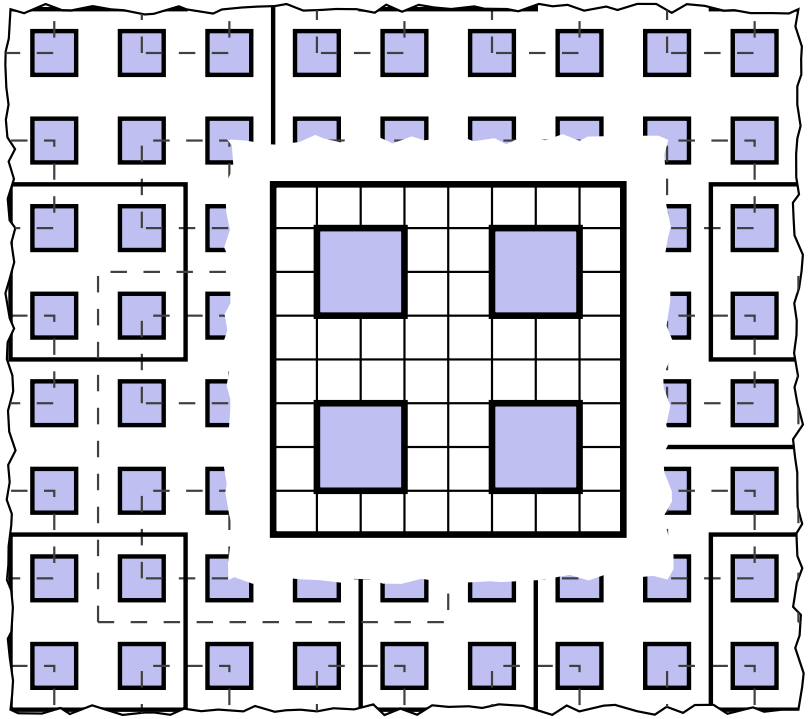
Un peu de machines de Turing, un peu de pavage de Robinson...



Et le problème du Domino ?

Solution

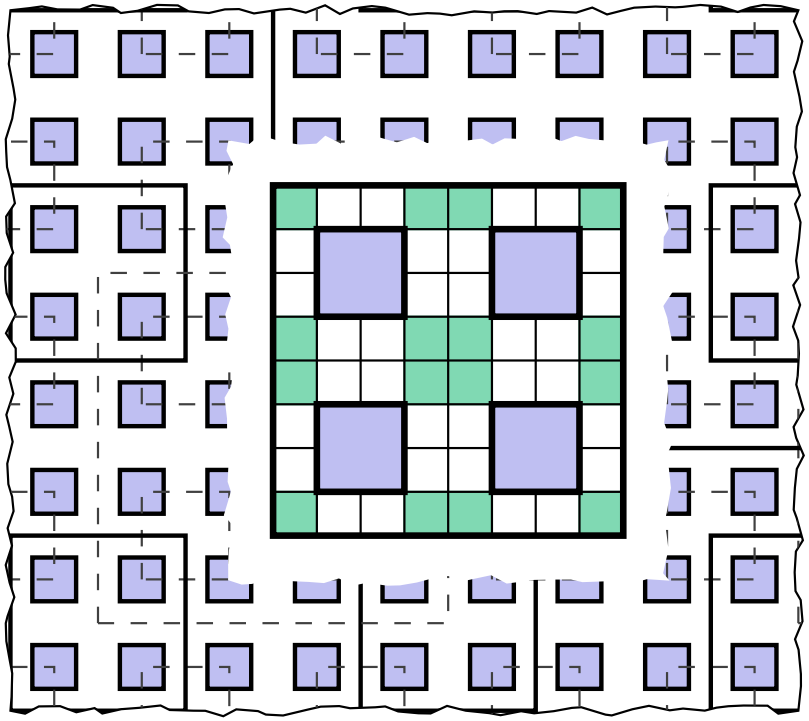
Un peu de machines de Turing, un peu de pavage de Robinson...



Et le problème du Domino ?

Solution

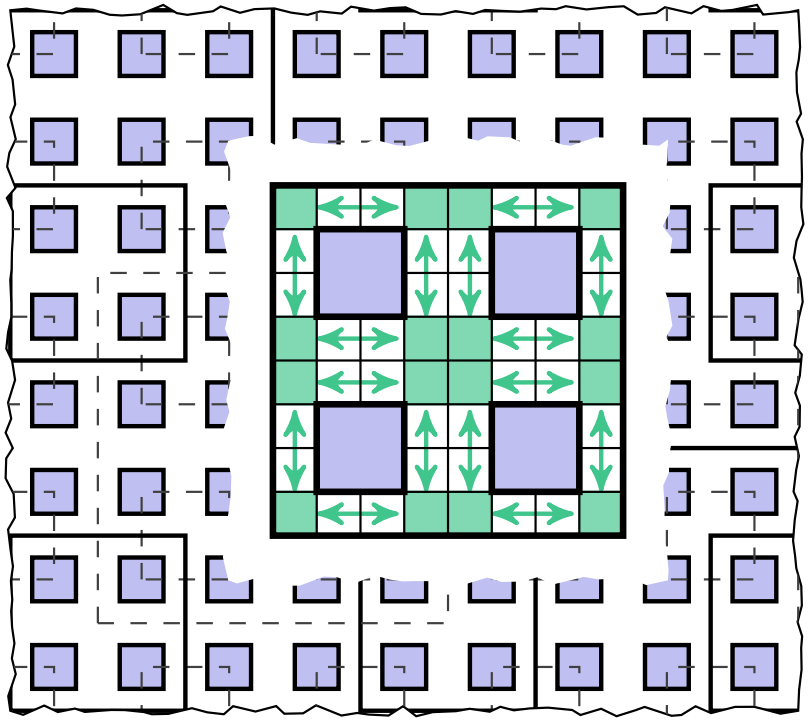
Un peu de machines de Turing, un peu de pavage de Robinson...



Et le problème du Domino ?

Solution

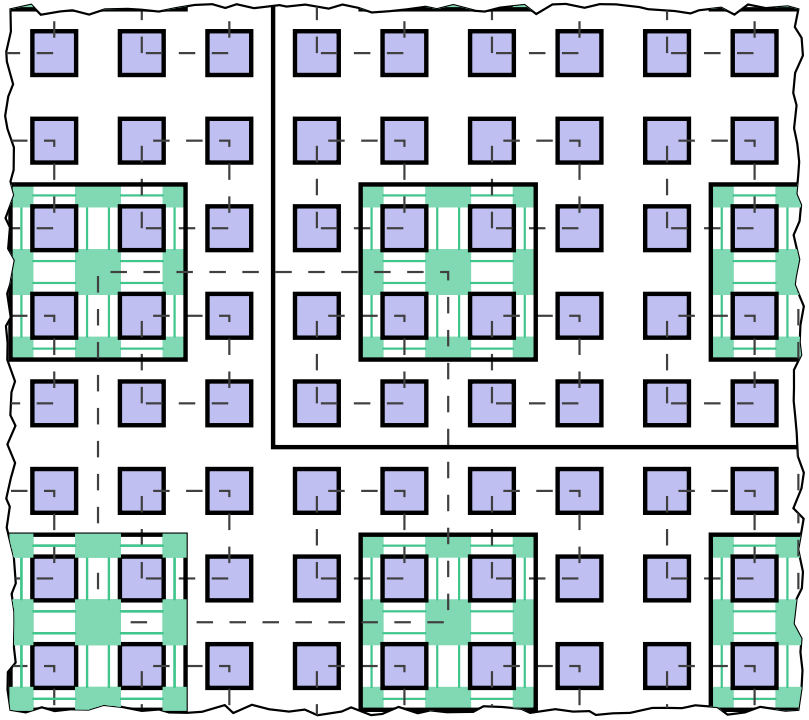
Un peu de machines de Turing, un peu de pavage de Robinson...



Et le problème du Domino ?

Solution

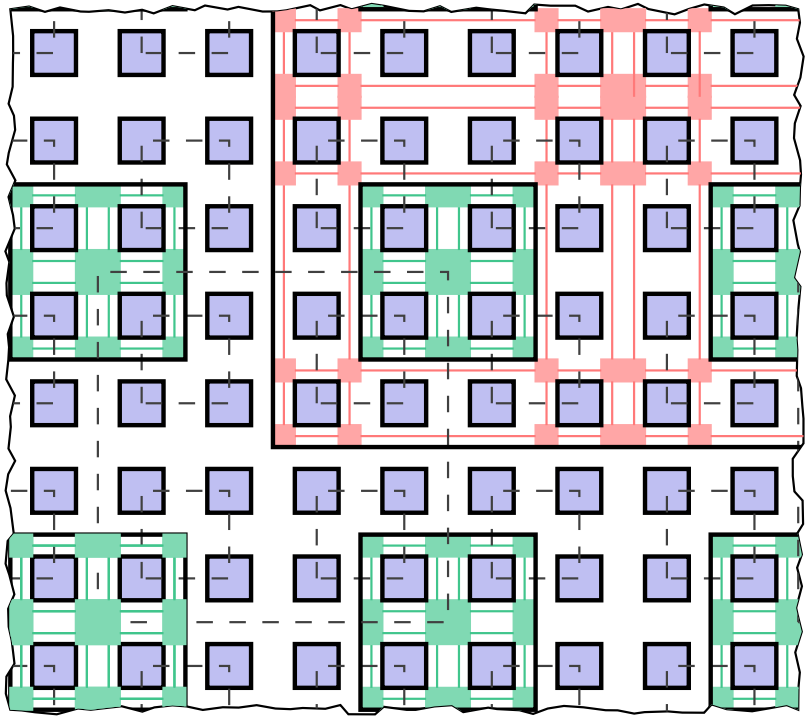
Un peu de machines de Turing, un peu de pavage de Robinson...



Et le problème du Domino ?

Solution

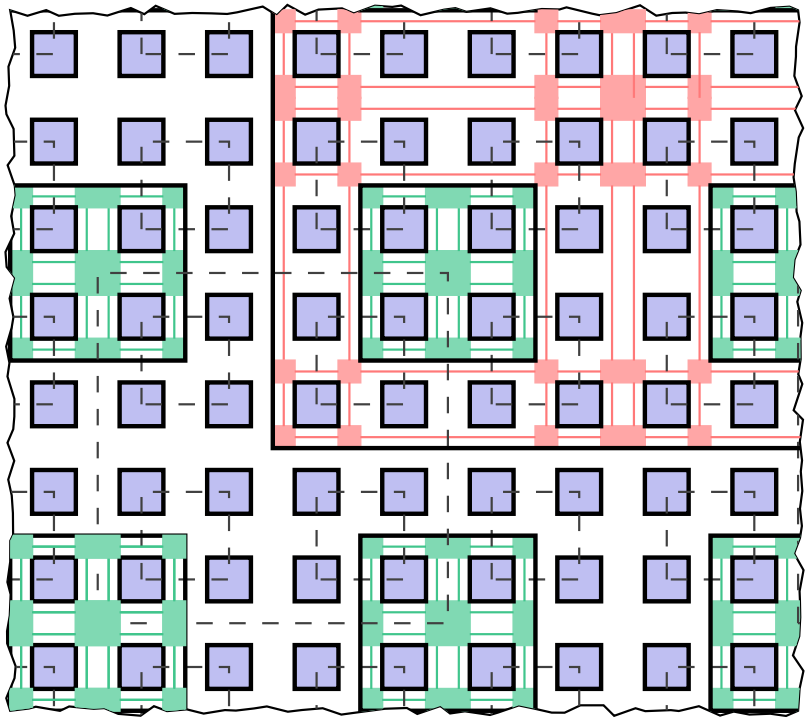
Un peu de machines de Turing, un peu de pavage de Robinson...



Et le problème du Domino ?

Solution

Un peu de machines de Turing, un peu de pavage de Robinson...



Théorème (Berger 1966, Robinson 1971)

Le problème du Domino est indécidable.

Merci ! !