Complexité algorithmique

Université de Toulouse

Année 2018/2019

Problèmes algorithmiques

Codage

Un objet informatique est codé par une suite de caractères appelé mot.

Alphabet Ensemble fini (par exemple : $\mathbb{B} = \{0, 1\}$ est l'alphabet binaire, $\mathcal{A} = \{a, b, c, \ldots, z\}$ un alphabet à 26 lettres.)

Mot Suite finie d'éléments de $\mathcal{A}$ on le note $u = u_1 u_2 \ldots u_n$ et n est la longueur du mot u, notée $|u|$.

Mot vide Le mot vide est noté ε.

Exemples de codage

Exemples usuels :

- **entiers** codés en **binaire** (l'entier 12 sera coder 1100) ;
- **rationnels** : le numérateur et le dénominateur sont séparé par un symbole spécial ($\frac{4}{3}$ sera coder 100#11) ;
- **matrices** : liste de coefficients séparés par des délimiteurs ;
- **graphes** : matice d'adjacence ;
- **polynôme** : liste des coefficients séparés par des délimiteurs...

Exemples moins usuels :

- **entiers** codés en **unaire** (l'entier 5 sera coder 11111) ;
- un **polynôme** de degré d codé par la liste des valeurs qu'il prend sur les entiers de 0 à d.

Codage d'un couple

On aura régulièrement besoin de parler de codage d'un couple (ou d'un n-uplets) de mots. Soit $x, y \in \mathcal{A}^*$, il y a plusieurs façon de coder (x, y) :

- Si l'alphabet peut être augmenter :
  - (x, y) peut être coder par le mot $x\#y$ sur l'alphabet $\mathcal{A} \cup \{\#\}$ la taille est alors $|x| + |y| + 1$;
  - on peut obtenir la taille $|x| + |y|$ en doublant l'alphabet (par exemple si $x = aab$ et $y = babb$ on code (x, y) par $aabBABB$.
- Si l'alphabet ne peut pas être augmenter (par exemple $\mathcal{A} = \{0, 1\}$) :
  - si $x = x_1 x_2 \dots x_n$ et $y = y_1 \dots y_m$ alors (x, y) est codé par

    $$x_1 0 x_2 0 x_3 0 \dots 0 x_n 1 y_1 \dots y_m$$

    Le codage de (x, y) est de taille $2|x| + |y|$.
  - pour un codage plus compact on code la taille $t = t_1 t_2 \dots t_k$ de x en binaire et (x, y) est codé par $t_1 0 t_2 0 t_3 0 \dots 0 t_k 1 xy$. Le codage de (x, y) est de taille $|x| + |y| + 2\lfloor \log(1 + |x|) \rfloor$.

Définition : Problèmes

Problème

Un **problème** P est composé d'une **entrée** (ou **instance**) et d'une question ou d'une tâche à réaliser dont la sortie est codée par un mot.

$$P : \mathcal{A}^* \to \mathcal{B}^*$$

Lorsque la sortie est $\{\texttt{true}, \texttt{false}\}$ on à un **problème de décision**.

Tri

Entrée : une liste d'entiers l

Tâche : trier l par ordre croissant

Primalité (`Prime`)

Entrée : un entier N

Question : N est il premier ?

Exemples de problèmes

Longueur de chemin

Entrée : Un graphe orienté $G = (V, E)$ et deux sommets s et t

Tâche : Donner la longueur du plus court chemin entre s et t (éventuellement ∞)

Accessibilité

Entrée : Un graphe orienté $G = (V, E)$ et deux sommets s et t

Question : Existe-t-il un chemin reliant s à t ?

Plusieurs codages possibles de $G = (V, E)$:

- on code n le nombre de sommets puis les m arrêtes comme un couple de sommet : $\simeq (2m + 1) \log(n + 1)$ bits ;
- On code n le nombre de sommets puis la matrice d'adjacence : $\simeq \log(n + 1) + n^2$.

Problèmes de décision vs langages

Définitons : Langage

Un langage $\mathcal{L}$ sur un alphabet $\mathcal{A}$ est un ensemble de mot.
Un langage $\mathcal{L} \subset \mathcal{A}^*$ définit le problème de décision dont les instances positives sont exactement les mots de $\mathcal{L}$.

Dans la suite, nous étudierons les problèmes de décisions mais il est souvent possible de passer d'un problème de décision à un problème dévaluation.

Exemple : Pour "trouver le plus petit diviseur non trivial de N", on met en place un algorithme pour résoudre le problème de décision "existe-t-il un diviseur de N inférieur à k" puis on approche la valeur par dichotomie.

Des problèmes non-calculables, ça existe !

- **Combien de mots binaires y a-t-il ?**

Des problèmes non-calculables, ça existe !

- **Combien de mots binaires y a-t-il ?**
  Autant que des entiers positifs ($\mathbb{N} = \{0, 1, 2, \ldots\}$). On dit que l'ensemble $\{0, 1\}^*$ est dénombrable.
  Les mots binaires peuvent être énumérés, par exemple :
  $0, 1, 00, 01, 10, 11, 000, \ldots$

Des problèmes non-calculables, ça existe !

- **Combien de mots binaires y a-t-il ?**
  Autant que des entiers positifs ($\mathbb{N} = \{0, 1, 2, \ldots\}$). On dit que l'ensemble $\{0, 1\}^*$ est dénombrable.
  Les mots binaires peuvent être énumérés, par exemple :
  $0, 1, 00, 01, 10, 11, 000, \ldots$
- **Combien de programmes/algorithmes existe-t-il ?**

Des problèmes non-calculables, ça existe !

- **Combien de mots binaires y a-t-il ?**
  Autant que des entiers positifs ($\mathbb{N} = \{0, 1, 2, \ldots\}$). On dit que l'ensemble $\{0, 1\}^*$ est dénombrable.
  Les mots binaires peuvent être énumérés, par exemple :
  $0, 1, 00, 01, 10, 11, 000, \ldots$
- **Combien de programmes/algorithmes existe-t-il ?**
  On peut coder chaque programme P par un mot en binaire $w_P \in \{0, 1\}^*$, il suffit de choisir son codage préféré. Ensuite, on peut énumérer les programmes, en énumérant les mots $\{0, 1\}^*$ représentant des programmes.
  L'ensemble des programmes est donc dénombrable également

Des problèmes non-calculables, ça existe !

- **Combien de mots binaires y a-t-il ?**
  Autant que des entiers positifs ($\mathbb{N} = \{0, 1, 2, \ldots\}$). On dit que l'ensemble $\{0, 1\}^*$ est dénombrable.
  Les mots binaires peuvent être énumérés, par exemple :
  $0, 1, 00, 01, 10, 11, 000, \ldots$
- **Combien de programmes/algorithmes existe-t-il ?**
  On peut coder chaque programme P par un mot en binaire $w_P \in \{0, 1\}^*$, il suffit de choisir son codage préféré. Ensuite, on peut énumérer les programmes, en énumérant les mots $\{0, 1\}^*$ représentant des programmes.
  L'ensemble des programmes est donc dénombrable également
- **Combien de fonctions $f : \mathbb{N} \to \mathbb{N}$ existe-t-il ?**

Des problèmes non-calculables, ça existe !

- **Combien de mots binaires y a-t-il ?**
  Autant que des entiers positifs ($\mathbb{N} = \{0, 1, 2, \ldots\}$). On dit que l'ensemble $\{0, 1\}^*$ est dénombrable.
  Les mots binaires peuvent être énumérés, par exemple :
  $0, 1, 00, 01, 10, 11, 000, \ldots$
- **Combien de programmes/algorithmes existe-t-il ?**
  On peut coder chaque programme P par un mot en binaire $w_P \in \{0, 1\}^*$, il suffit de choisir son codage préféré. Ensuite, on peut énumérer les programmes, en énumérant les mots $\{0, 1\}^*$ représentant des programmes.
  L'ensemble des programmes est donc dénombrable également
- **Combien de fonctions $f : \mathbb{N} \to \mathbb{N}$ existe-t-il ?**
  Autant que des nombres réels (ensemble non-dénombrable).

Des problèmes non-calculables, ça existe !

- **Combien de mots binaires y a-t-il ?**
  Autant que des entiers positifs ($\mathbb{N} = \{0, 1, 2, \ldots\}$). On dit que l'ensemble $\{0, 1\}^*$ est dénombrable.
  Les mots binaires peuvent être énumérés, par exemple :
  $0, 1, 00, 01, 10, 11, 000, \ldots$
- **Combien de programmes/algorithmes existe-t-il ?**
  On peut coder chaque programme P par un mot en binaire $w_P \in \{0, 1\}^*$, il suffit de choisir son codage préféré. Ensuite, on peut énumérer les programmes, en énumérant les mots $\{0, 1\}^*$ représentant des programmes.
  L'ensemble des programmes est donc dénombrable également
- **Combien de fonctions $f : \mathbb{N} \to \mathbb{N}$ existe-t-il ?**
  Autant que des nombres réels (ensemble non-dénombrable).

Proposition

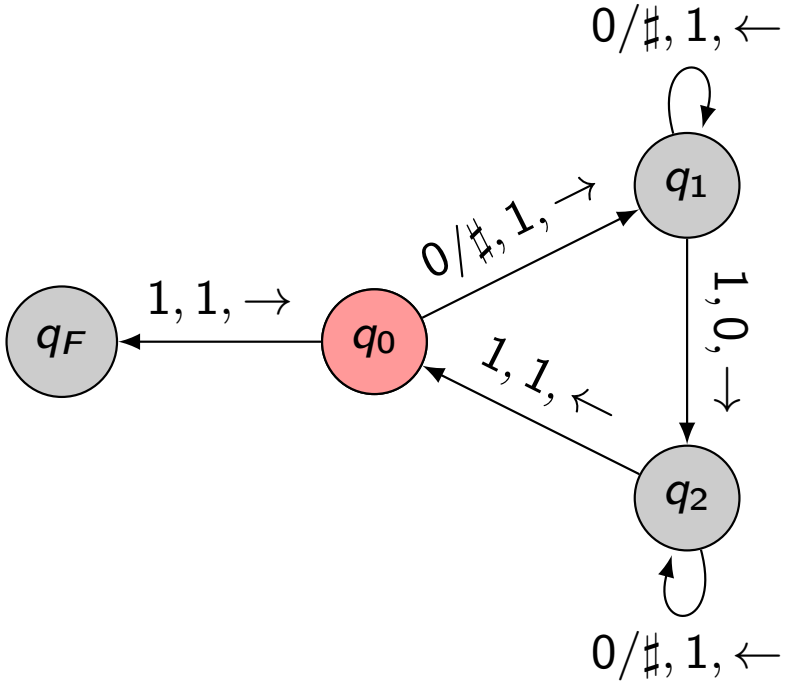
Il existe des fonctions $f : \mathbb{N} \to \mathbb{N}$ non-calculables.

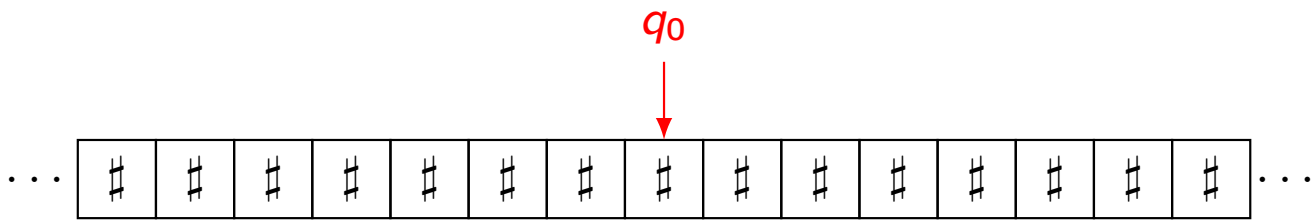
Modèles de calculs

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



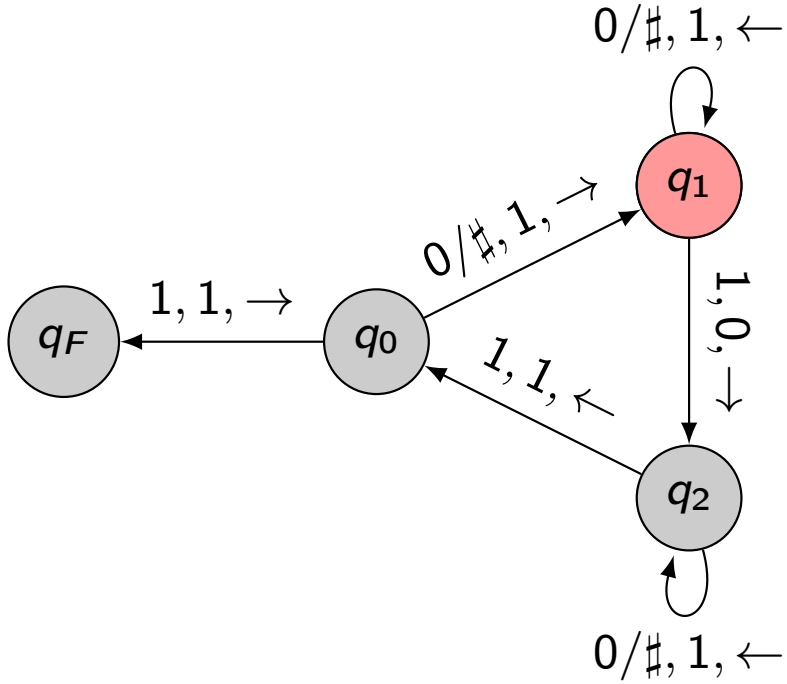


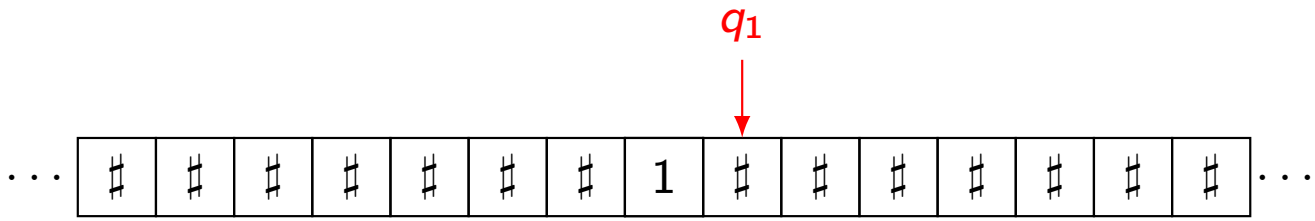
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



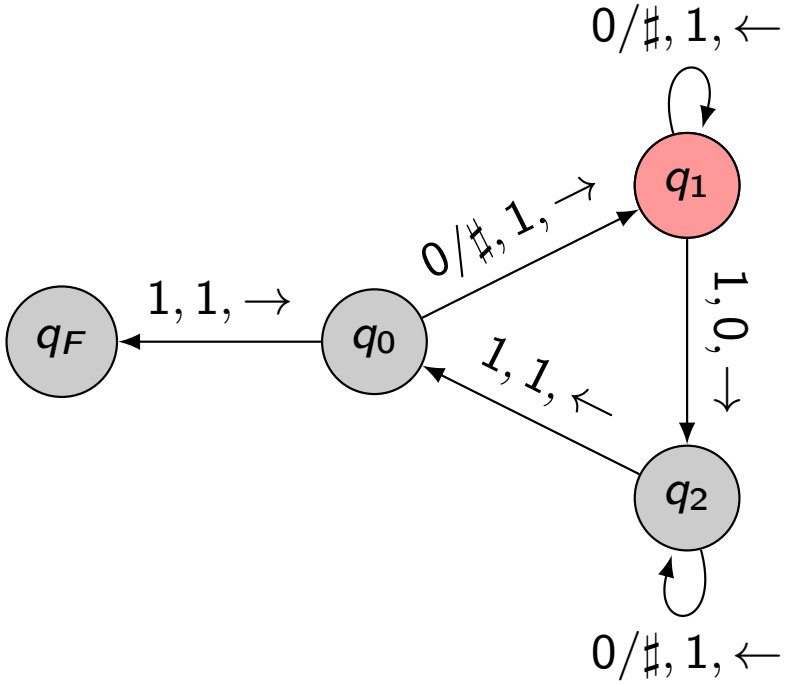


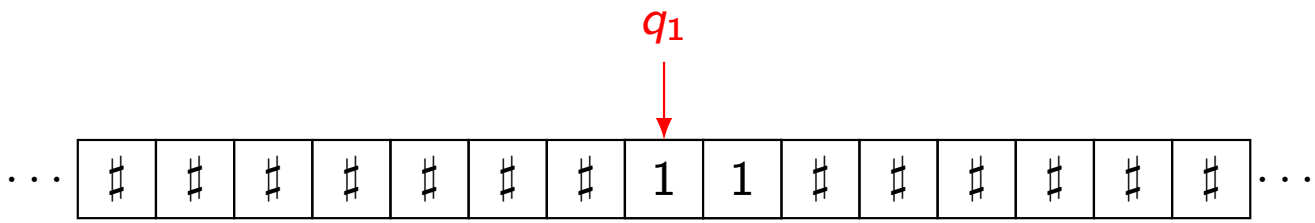
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



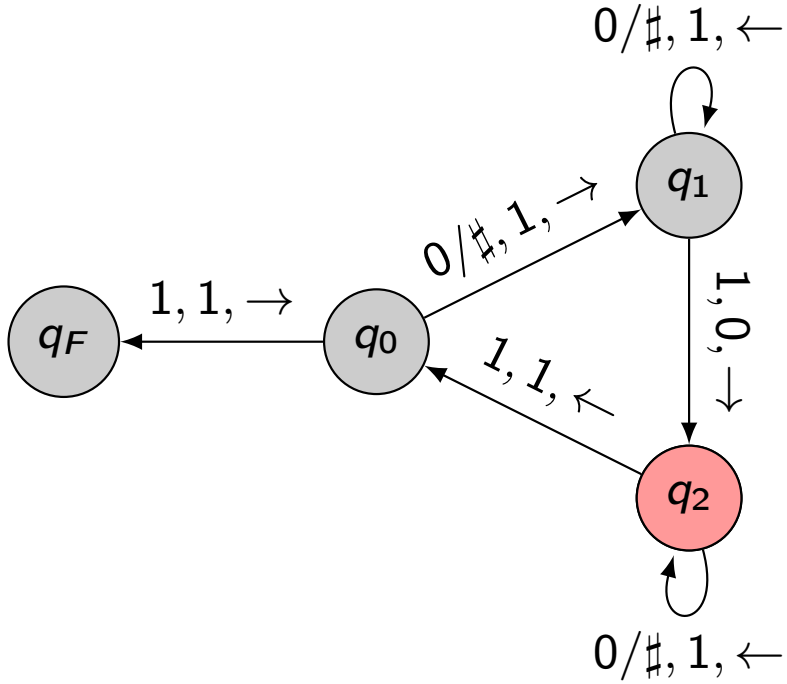


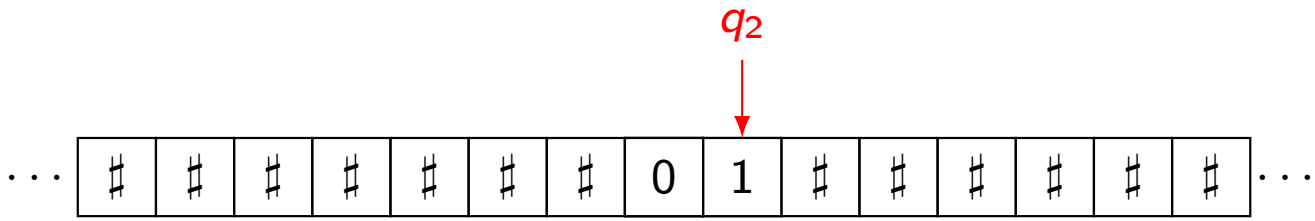
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



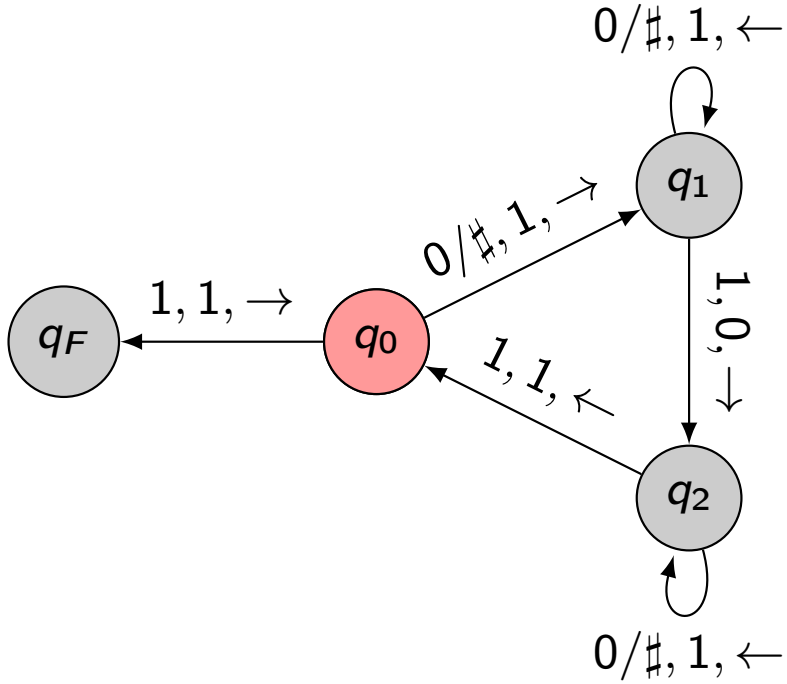


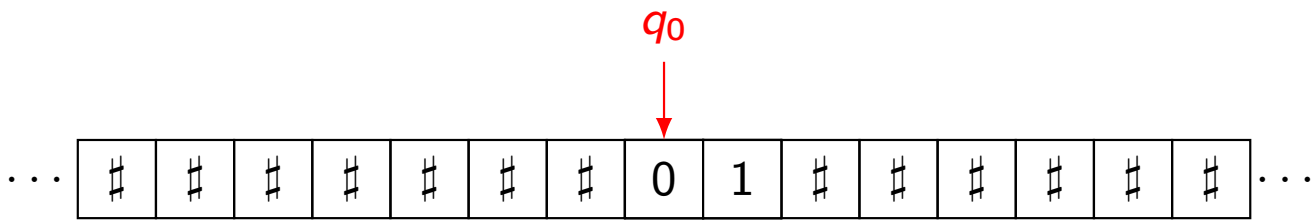
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



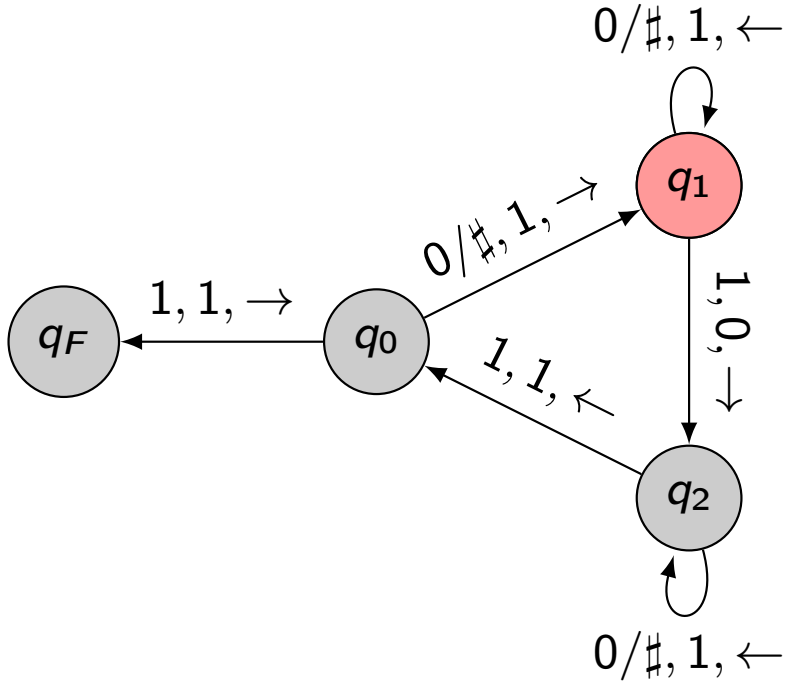


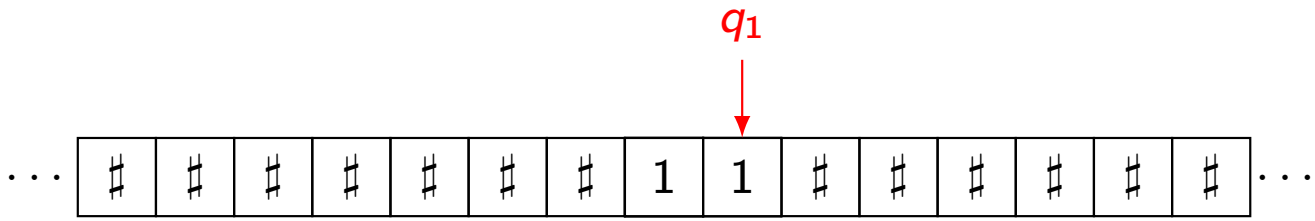
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



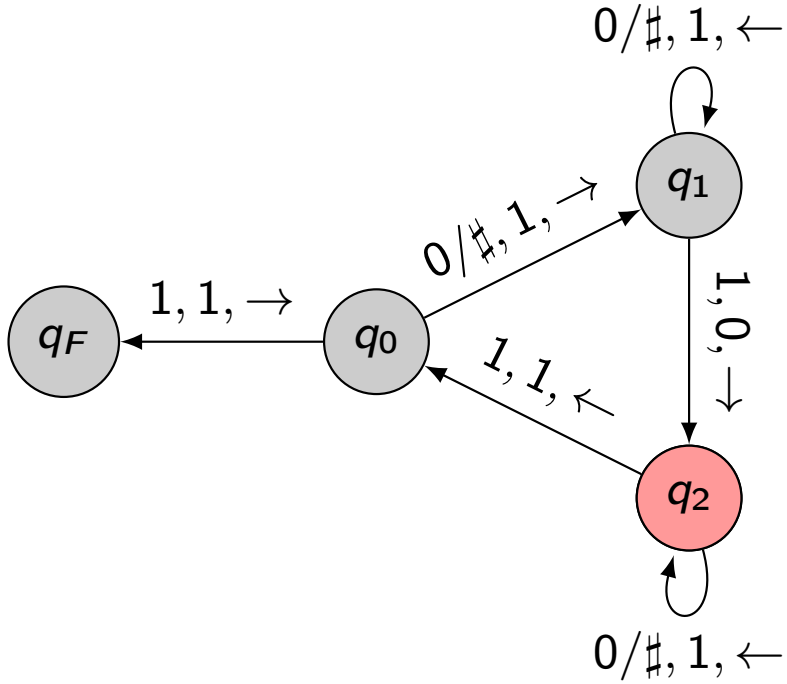


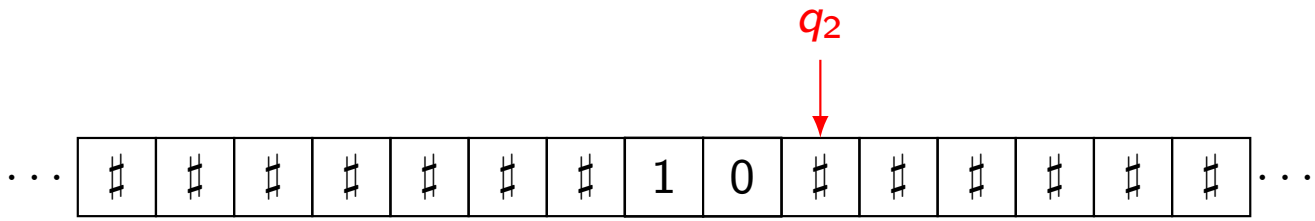
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



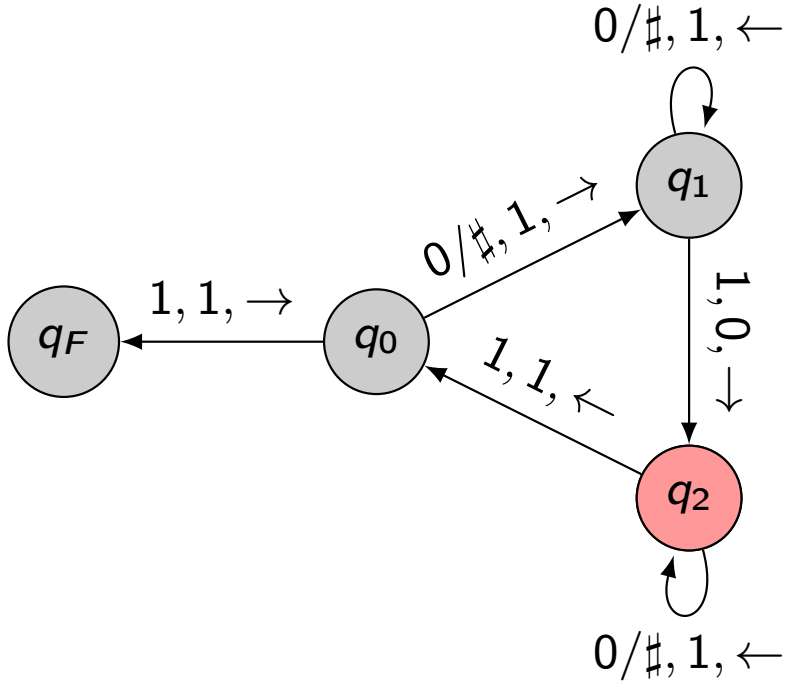


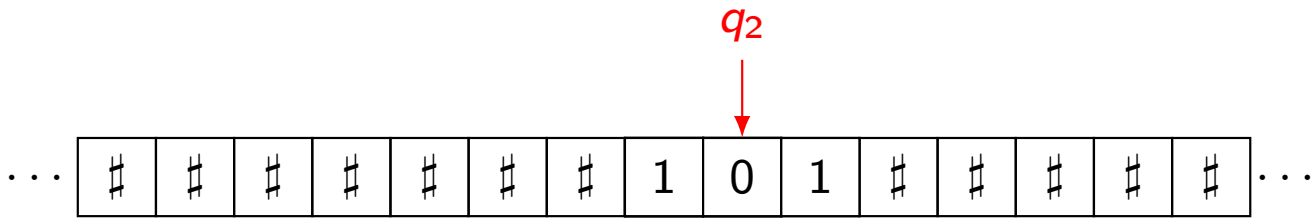
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



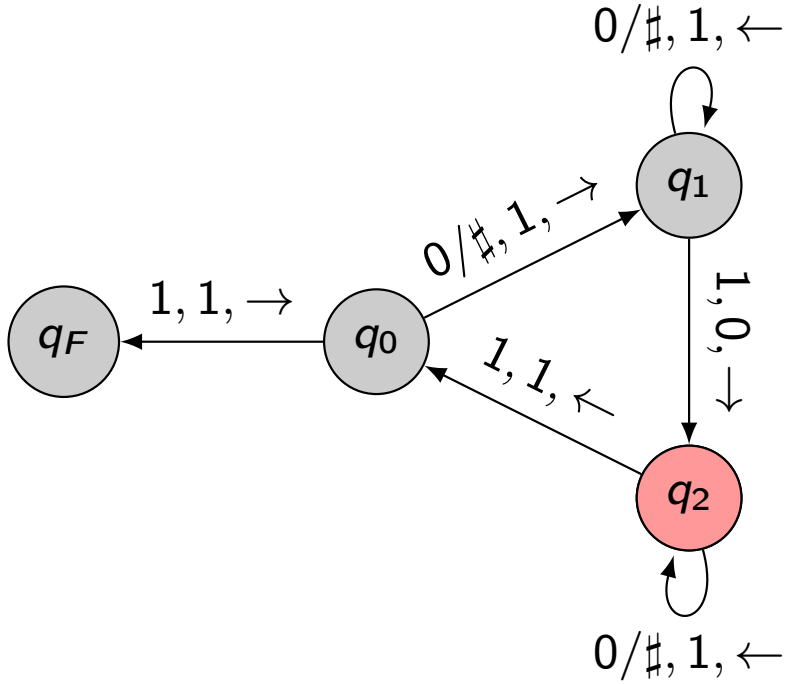


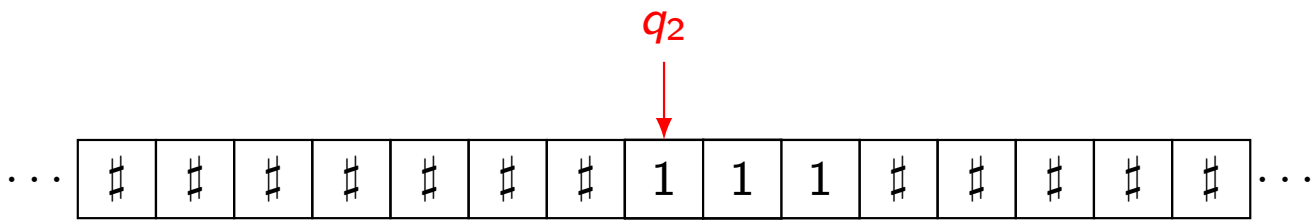
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



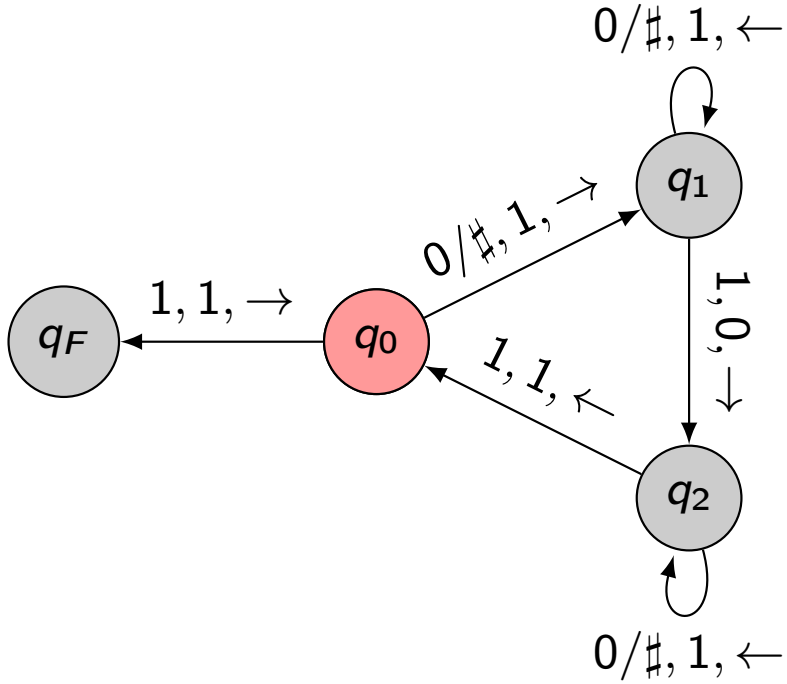


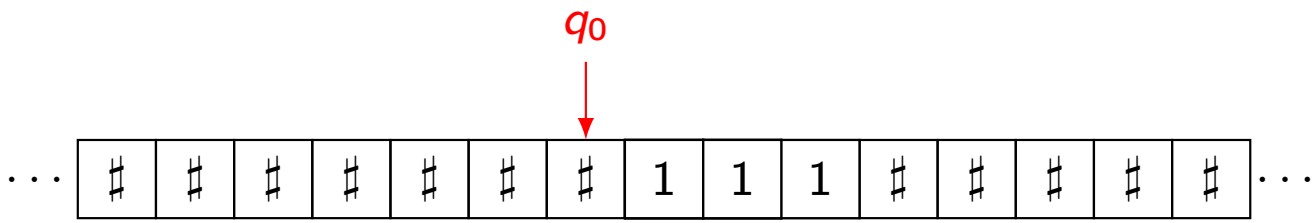
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



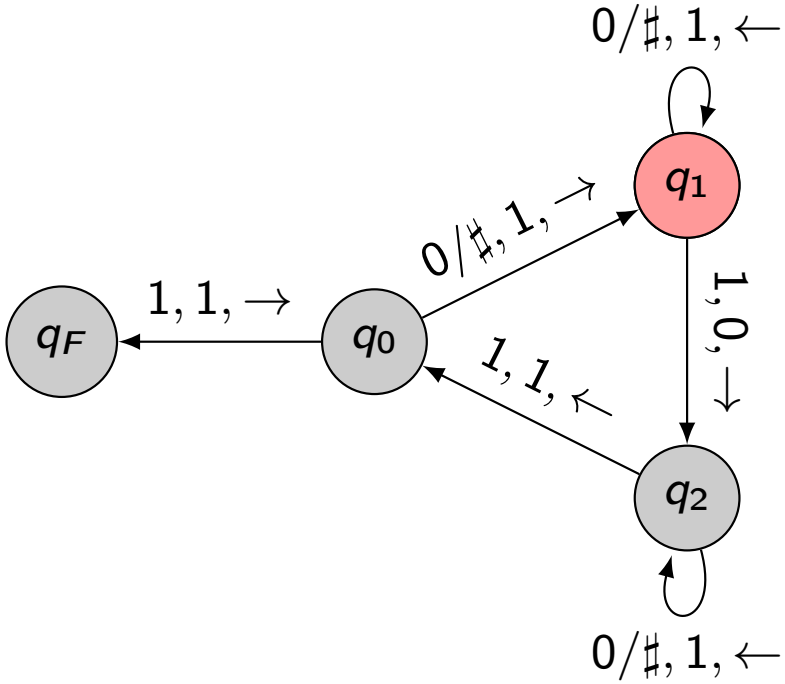


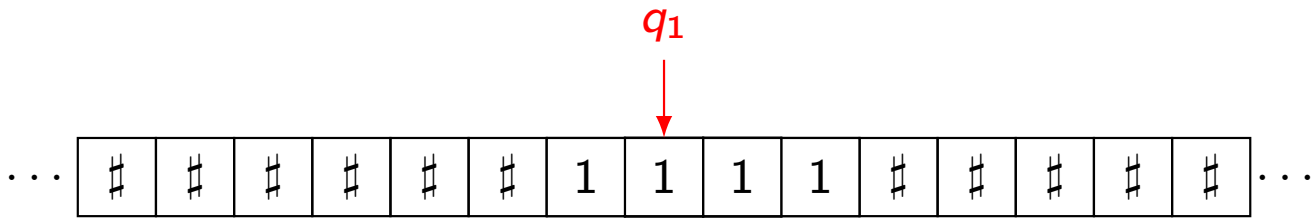
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



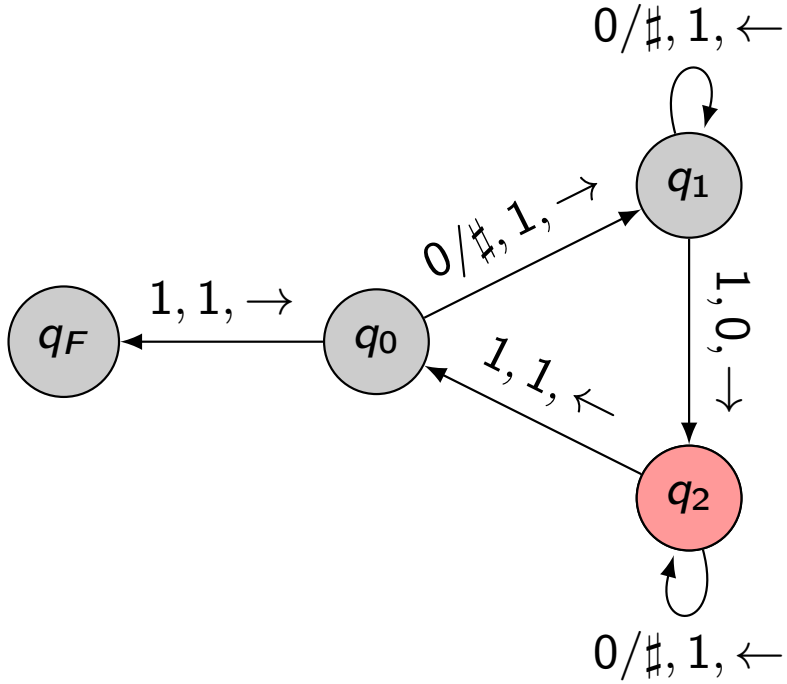


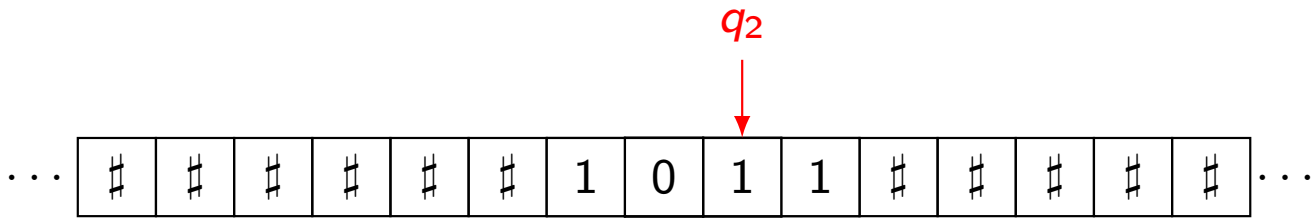
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



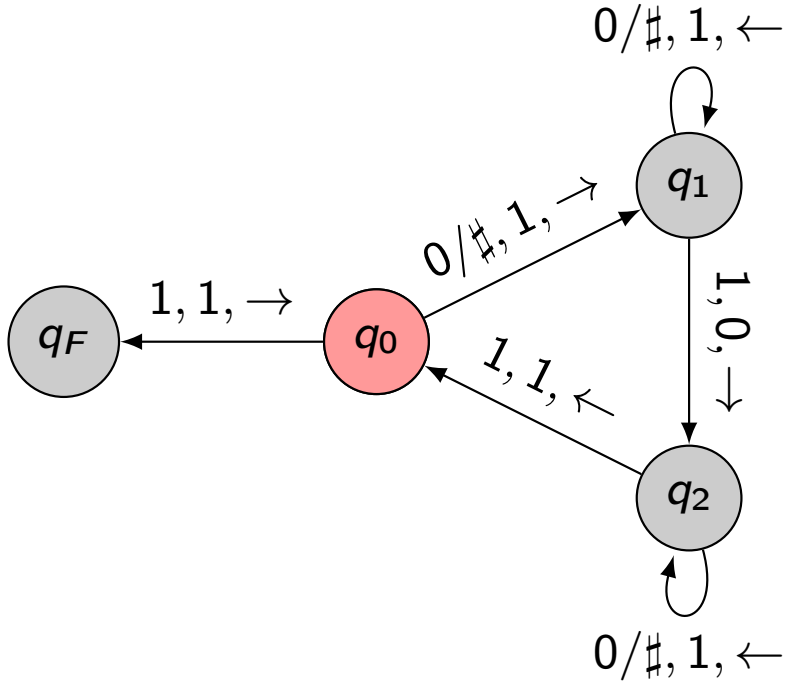


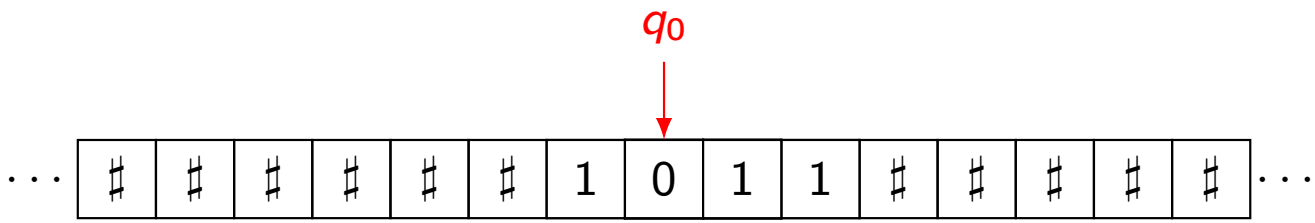
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



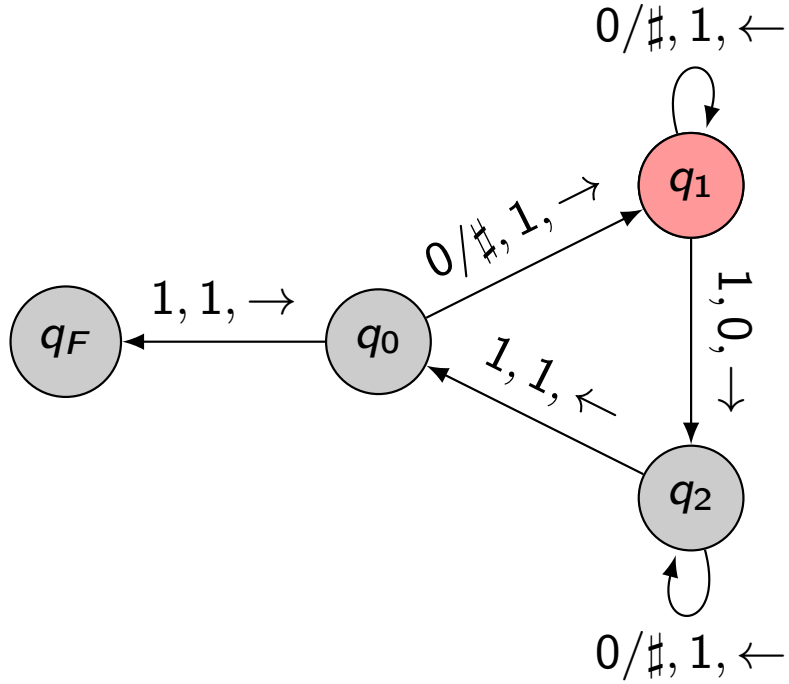


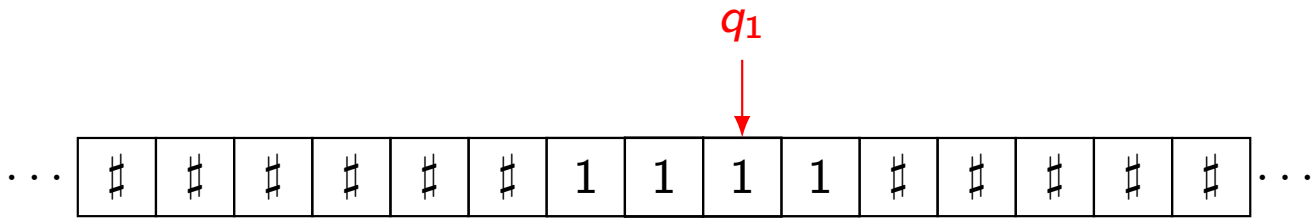
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



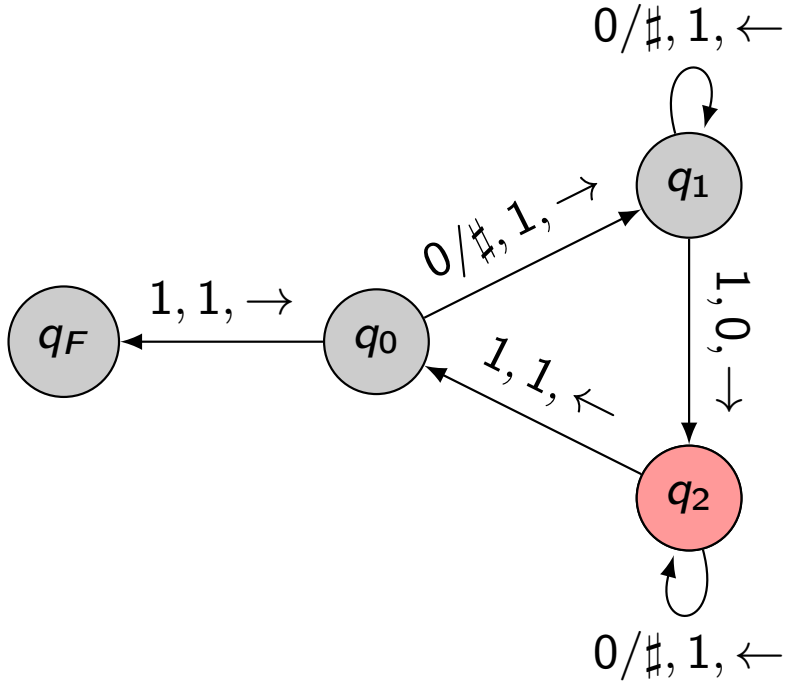


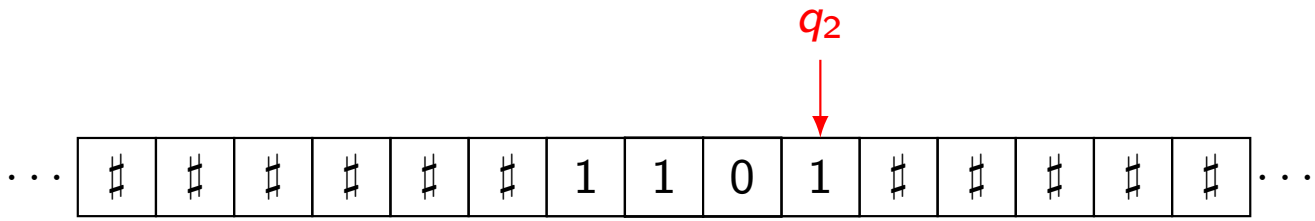
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



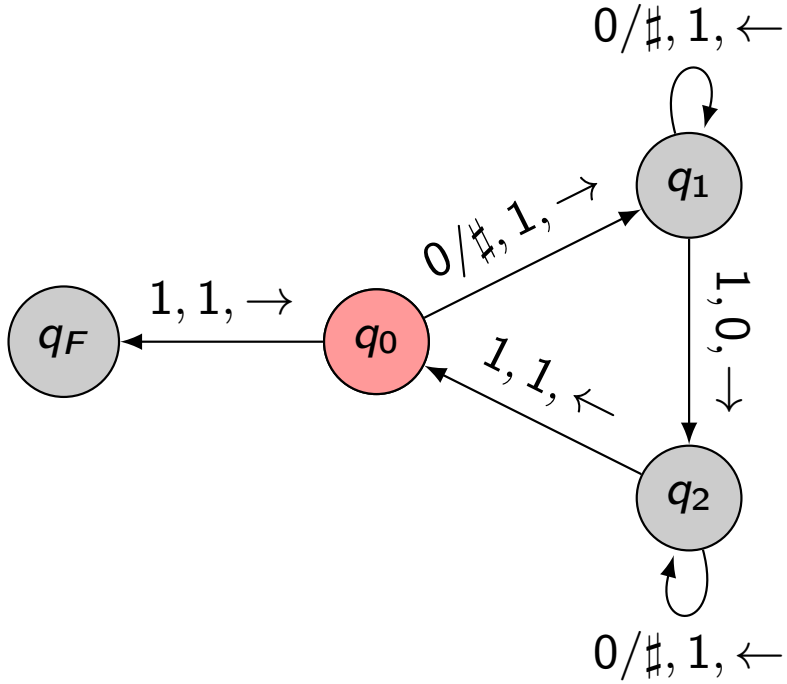


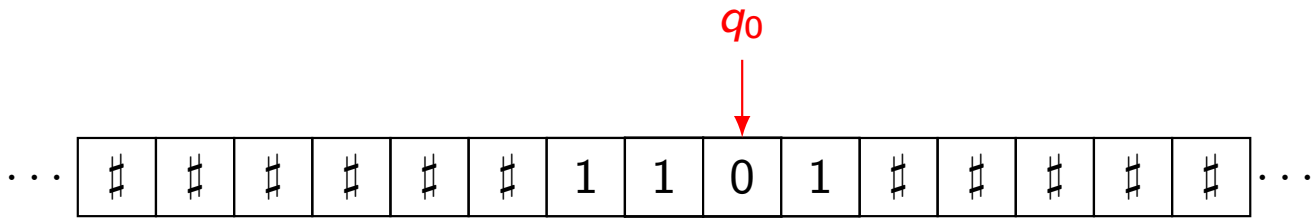
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



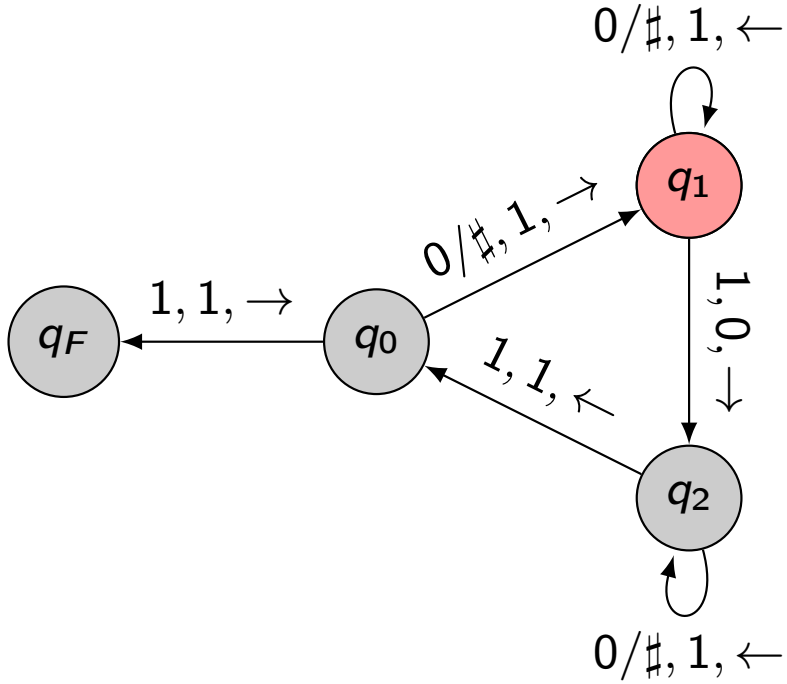


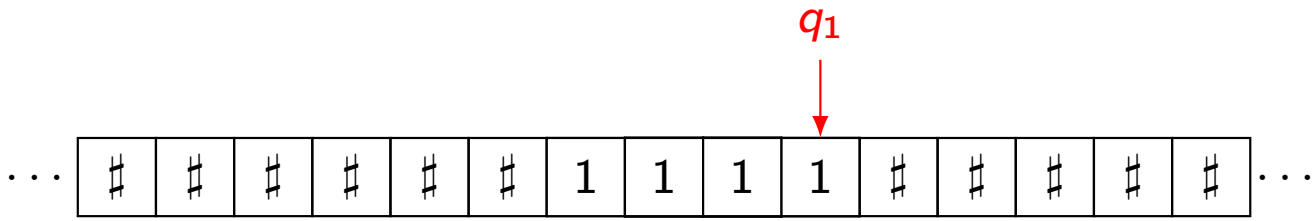
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



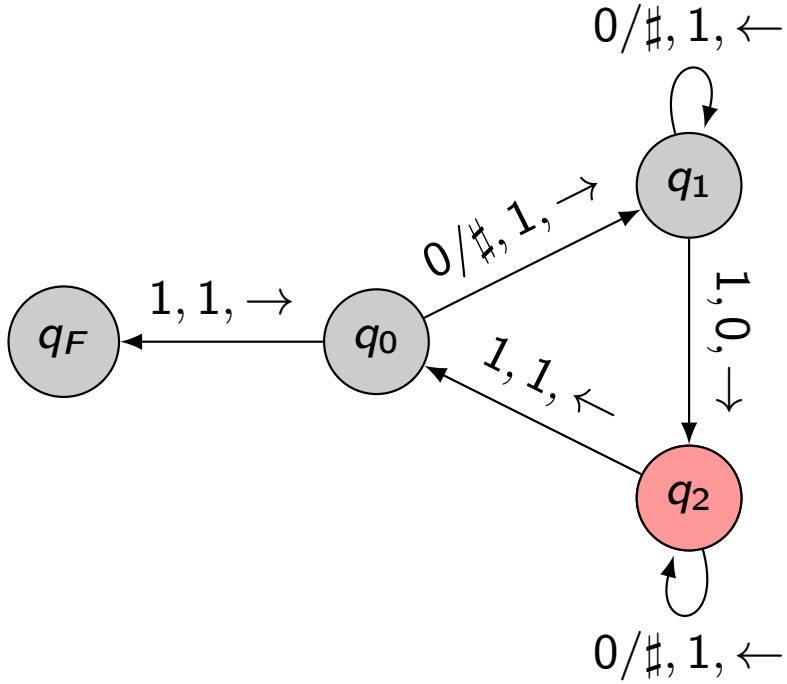


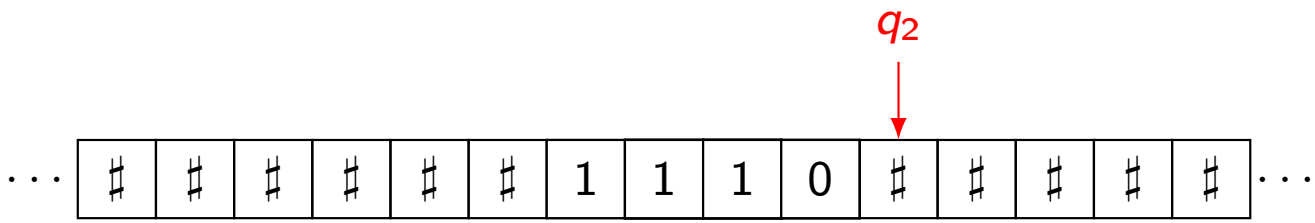
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



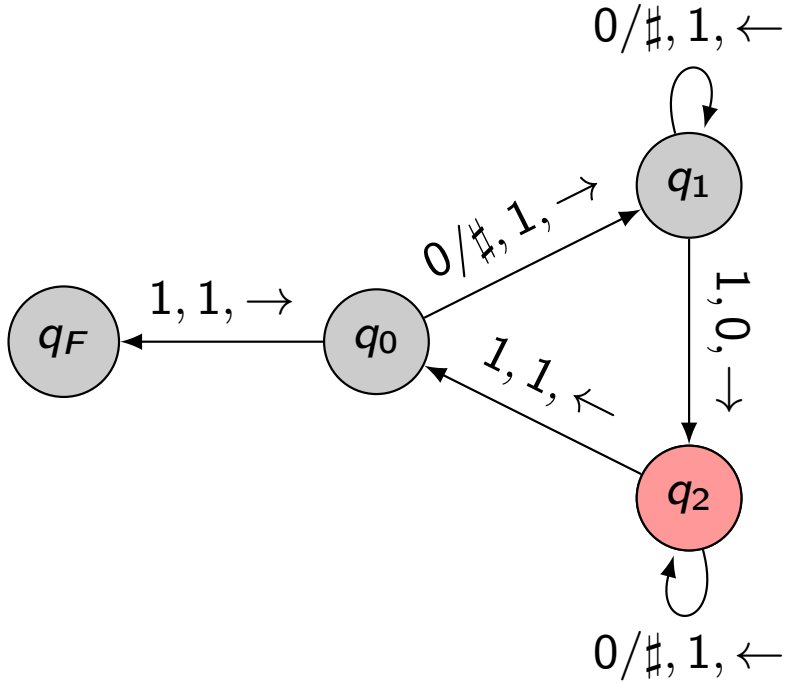


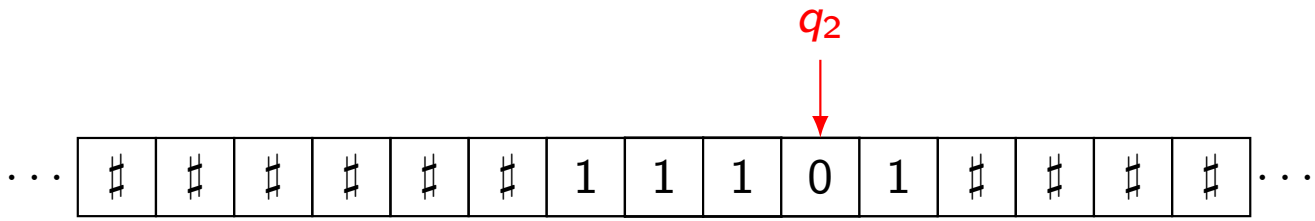
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



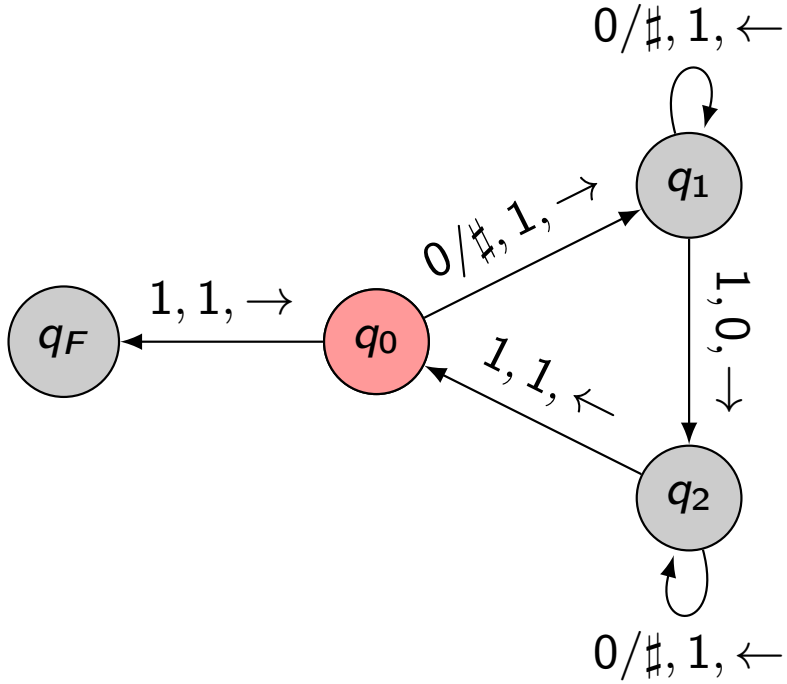


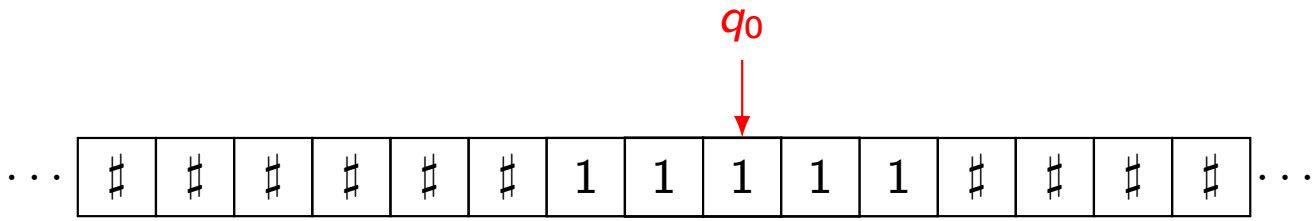
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.



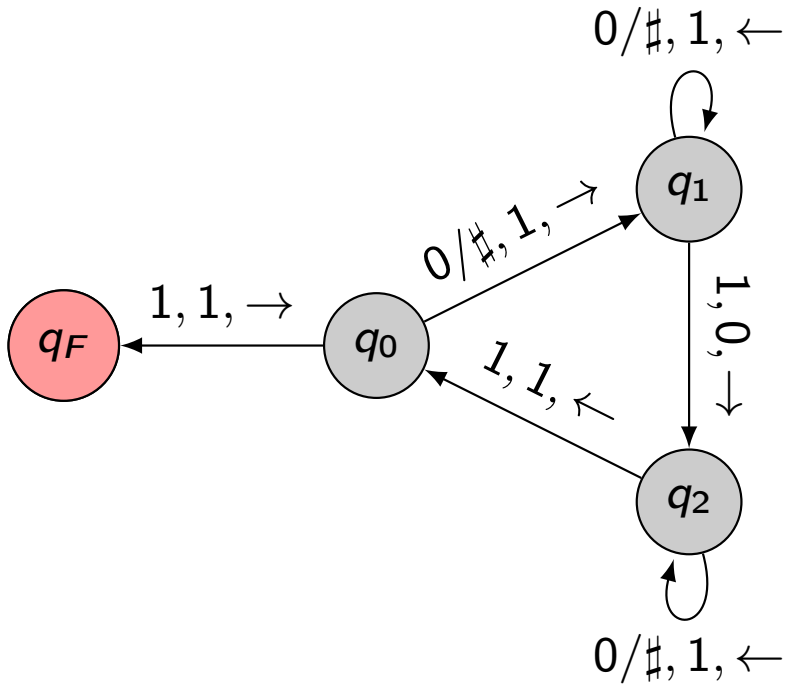


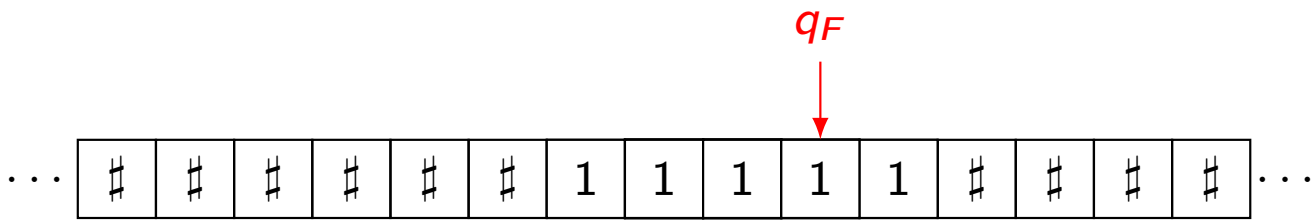
Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Calcul d'une machine de Turing

Une **machine de Turing** $\mathcal{M}$ est définit par :

- Q : un nombre fini d'état ;
- q_0 : l'état initial ;
- q_F : l'état final ;
- $\mathcal{A}$ un alphabet fini ;
- $\sharp \notin \mathcal{A}$ un symbole blanc
- $\delta : Q \times \mathcal{A} \to Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\}$ une fonction de transition.





Soit un ruban $R \in \Gamma^{\mathbb{Z}}$, la tête est dans la position $i \in \mathbb{Z}$ et l'état $q \in Q$. La machine de Turing réalise la transition $\delta(q, R_i) = (q', a, \epsilon)$, elle écrit donc a à la position i, ce déplace à la position $i + \epsilon$ et rentre dans l'état q'.

Cette machine s'arête après 20 étapes de calculs et visiter 5 cases.

Quelques remarques

- Le nombre d'états d'une machine de Turing est fini : un ordinateur a un nombre fini de registres.
- La bande représente la mémoire de la machine. Elle est infinie : sur un ordinateur, on peut ajouter des périphériques mémoire (disques. . .) de façon quasi-illimitée.
- L'accès à la mémoire est séquentiel : la machine peut bouger sa tête à droite ou à gauche d'une case à chaque étape.
- Un compilateur peut être vu comme une machine de **Turing universelle**

Exemples de machines de Turing

- Machine qui effectue `while(true)`.
- Machine qui efface son entrée et s'arrête.
- Machine qui accepte 0^*1^*
- Machine qui accepte $\{a^{2^n}; n \geq 0\}$.
- Machine qui accepte $\{a^n b^n; n \geq 0\}$.
- Machine qui accepte $\{a^n b^n c^n; n \geq 0\}$.
- Machine qui accepte $\{ww; w \in \mathcal{A}^*\}$.
- Machine qui interprête son entrée comme un entier n, le remplace par $\lfloor \frac{n}{2} \rfloor$ et s'arrête.
- Machine qui effectue l'incrément en binaire.
- Machine qui effectue l'addition de deux entiers en unaire.
- Machine qui effectue la multiplication de deux entiers en unaire.

Thèse de Church-Turing

Il existe de nombreux autres modèles équivalent :

- Machine de Turing sur l'alphabet $\{0, 1\}$
- Machine de Turing à k rubans
- Automate à deux piles
- Machine RAM
- Fonctions récursives primitives et non primitives...

Thèse de Church-Turing

Il existe de nombreux autres modèles équivalent :

- Machine de Turing sur l'alphabet $\{0, 1\}$
- Machine de Turing à k rubans
- Automate à deux piles
- Machine RAM
- Fonctions récursives primitives et non primitives...

Proposition

Tout programme écrit dans un langage composé de variables entières x_i (en nombre non borné), d'opération d'affectation $\leftarrow$, d'addition $+$, de multiplication $\times$, de tests d'égalité $=$ et d'inégalité $\leq$, d'instruction conditionnelles `Si...alors...Sinon...` et de boucles `tant que`, peut être simulé par une machine de Turing.

Problèmes indécidables

Problème de l'arrêt

Problème de l'arrêt (`Halt`)

Entrée : Un programme $\mathcal{M}$ et une entrée x

Question : Est ce que l'exécution de $\mathcal{M}$ sur x s'arrête ?

Langage associé : $\texttt{Halt} = \{\langle \mathcal{M}, x \rangle : \mathcal{M} \text{ s'arrête sur l'entrée } x\}$

Théorème *(Turing 1937)*

Le problème de l'arrêt est indécidable (mais semi-décidable).

Problème de l'arrêt

Problème de l'arrêt (Halt)

Entrée : Un programme $\mathcal{M}$ et une entrée x

Question : Est ce que l'exécution de $\mathcal{M}$ sur x s'arrête ?

Langage associé : $\texttt{Halt} = \{\langle \mathcal{M}, x \rangle : \mathcal{M} \text{ s'arrête sur l'entrée } x\}$

Théorème *(Turing 1937)*

Le problème de l'arrêt est indécidable (mais semi-décidable).

Supposons qu'un programme `SuperProg` résolve le problème :

$$\texttt{SuperProg}(\texttt{P}, w) = \begin{cases} \texttt{"yes"} & \text{si } \texttt{P}(\text{w}) \text{ s'arrête} \\ \texttt{"no"} & \text{sinon} \end{cases}$$

Considérons le programme : $\texttt{Bug}(\texttt{P}) = \begin{cases} \texttt{"yes"} & \text{si } \texttt{SuperProg}(\texttt{P},\texttt{P})=\texttt{"no"} \\ \text{fait une boucle} & \text{sinon} \end{cases}$

Problème de l'arrêt

Problème de l'arrêt (Halt)

Entrée : Un programme $\mathcal{M}$ et une entrée x

Question : Est ce que l'exécution de $\mathcal{M}$ sur x s'arrête ?

Langage associé : $\texttt{Halt} = \{\langle \mathcal{M}, x\rangle : \mathcal{M} \text{ s'arrête sur l'entrée } x\}$

Théorème *(Turing 1937)*

Le problème de l'arrêt est indécidable (mais semi-décidable).

Supposons qu'un programme SuperProg résolve le problème :

$$\texttt{SuperProg(P}, w) = \begin{cases} \text{"yes"} & \text{si P(w) s'arrête} \\ \text{"no"} & \text{sinon} \end{cases}$$

Considérons le programme : $\texttt{Bug(P)} = \begin{cases} \text{"yes"} & \text{si SuperProg(P,P)="no"} \\ \text{fait une boucle} & \text{sinon} \end{cases}$

Bug(Bug)="yes" $\Longleftrightarrow$ SuperProg(Bug,Bug)="no" $\Longleftrightarrow$ Bug(Bug) ne s'arrête pas

Problème de l'arrêt

Problème de l'arrêt (Halt)

Entrée : Un programme $\mathcal{M}$ et une entrée x

Question : Est ce que l'exécution de $\mathcal{M}$ sur x s'arrête ?

Langage associé : $\texttt{Halt} = \{\langle \mathcal{M}, x \rangle : \mathcal{M} \text{ s'arrête sur l'entrée } x\}$

Théorème *(Turing 1937)*

Le problème de l'arrêt est indécidable (mais semi-décidable).

Problème de l'arrêt sur l'entrée vide ($\texttt{Halt}\emptyset$)

Entrée : Un programme $\mathcal{M}$

Question : Est ce que l'exécution de $\mathcal{M}$ sur s'arrête sur l'entrée vide ?

Langage associé : $\texttt{Halt}\emptyset = \{\langle \mathcal{M} \rangle : \mathcal{M} \text{ s'arrête sur l'entrée } \emptyset\}$

Théorème *(Turing 1937)*

Le problème de l'arrêt sur l'entrée vide est indécidable.

Notion de réduction

Réduction

Soient $\mathcal{L} \subset \mathcal{A}^*$ et $\mathcal{L}' \subset \mathcal{B}^*$ deux langages. On dit que $\mathcal{L}$ se réduit à $\mathcal{L}'$, noté $\mathcal{L} \leq_m \mathcal{L}'$, s'il existe une fonction calculable totale $f : \mathcal{A}^* \to \mathcal{B}^*$ telle que

$$x \in \mathcal{L} \iff f(x) \in \mathcal{L}'$$

Proposition

- $\leq_m$ est un préordre sur l'ensemble des langages
- Si $\mathcal{L} \leq_m \mathcal{L}'$ et $\mathcal{L}'$ décidable alors $\mathcal{L}$ est décidable.
- Si $\mathcal{L} \leq_m \mathcal{L}'$ et $\mathcal{L}$ indécidable alors $\mathcal{L}$ est indécidable.

Application :

$$\texttt{Halt} \leq_m \texttt{Halt}\emptyset$$

Donc $\texttt{Halt}\emptyset$ est indécidable.

Quelques problèmes indécidables sur les programmes

- HALT : étant donnés un programme P et un entier n, est-ce que P s'arrête sur n ?
- $\text{HALT}_\emptyset$: étant donné un programme P, est-ce que P s'arrête sur 0 ?
- UTILE : étant donnés un programme P et une instruction I, est-ce que P utilise l'instruction I sur l'entrée 0 ?
- $\text{HALT}_\exists$: étant donné un programme P, est-ce que P s'arrête sur au moins une entrée ?
- $\text{HALT}_\forall$: étant donné un programme P, est-ce que P s'arrête sur toute entrée ?
- EQUIV : étant donnés deux programmes P_1 et P_2, est-ce que P_1 et P_2 acceptent les mêmes mots ?

Remarques : Les 4 premières questions sont **semi-décidables**, les deux dernières ne le sont pas.

Deux problèmes décidables sur les programmes

Problème de l'arrêt borné (`HaltExp`)

Entrée : Un programme $\mathcal{M}$, une entrée x et un entier k codé en binaire

Question : Est ce que l'exécution de $\mathcal{M}$ sur x s'arrête en k étapes ?

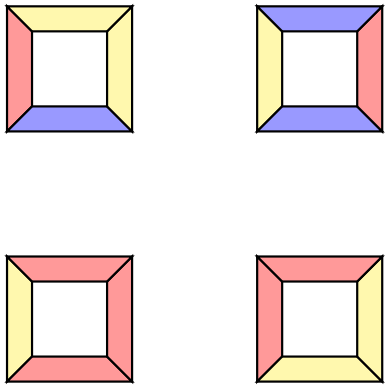
Problème des valeurs bornées

Entrée : Un programme $\mathcal{M}$, une entrée x et un entier k codé en binaire

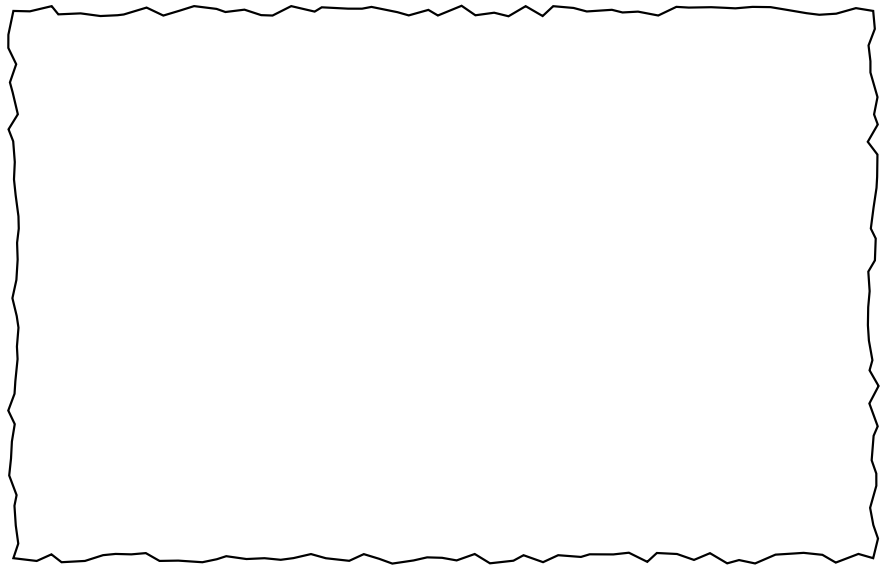
Question : Est ce que l'exécution de $\mathcal{M}$ sur x avant que la taille des variables dépassent la valeur k ?

Problème du domino

Jeu de tuile



Un pavage associé

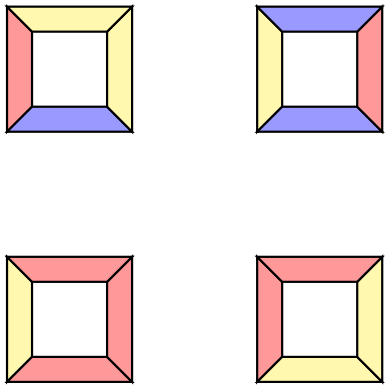


Problème du domino

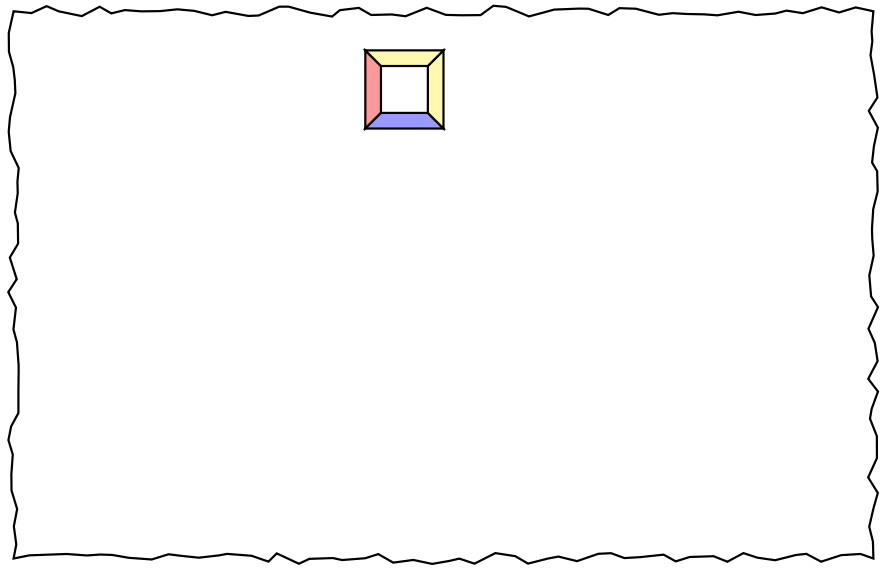
Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

Problème du domino

Jeu de tuile



Un pavage associé

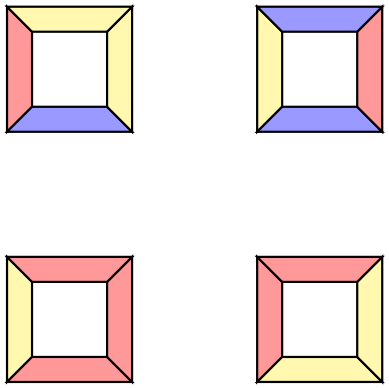


Problème du domino

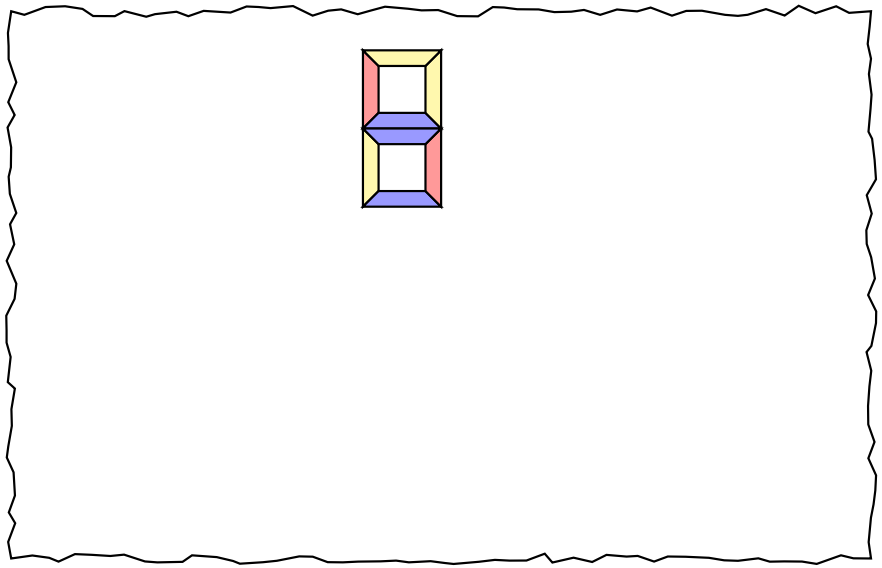
Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

Problème du domino

Jeu de tuile



Un pavage associé

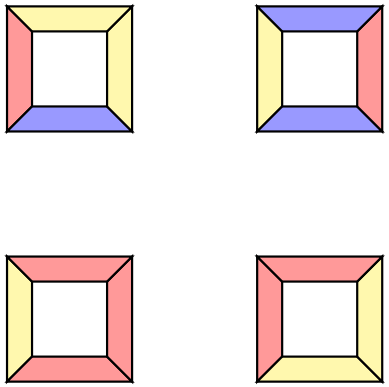


Problème du domino

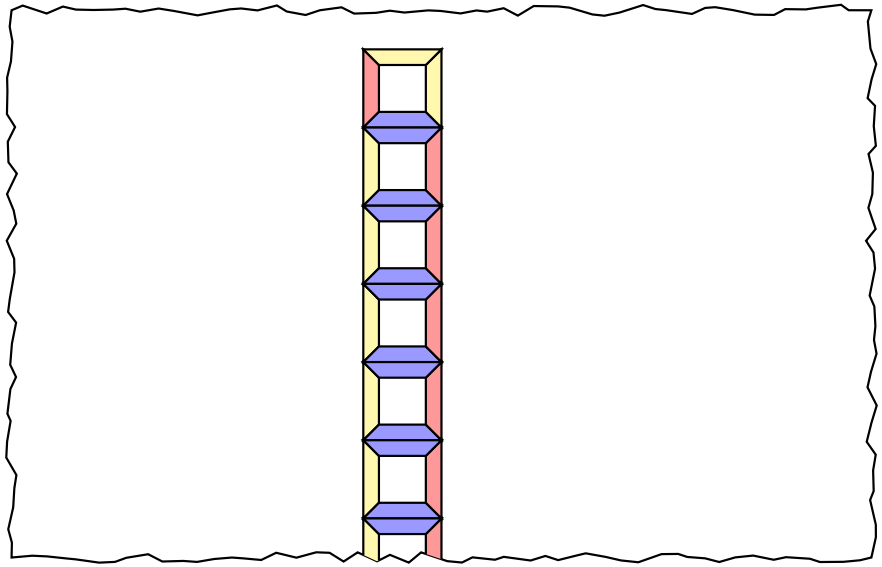
Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

Problème du domino

Jeu de tuile



Un pavage associé

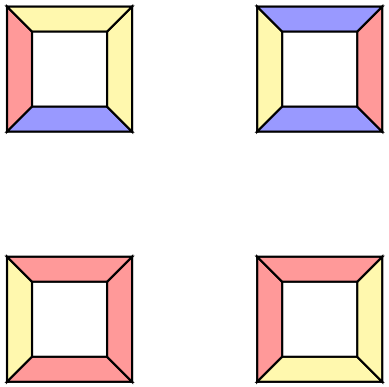


Problème du domino

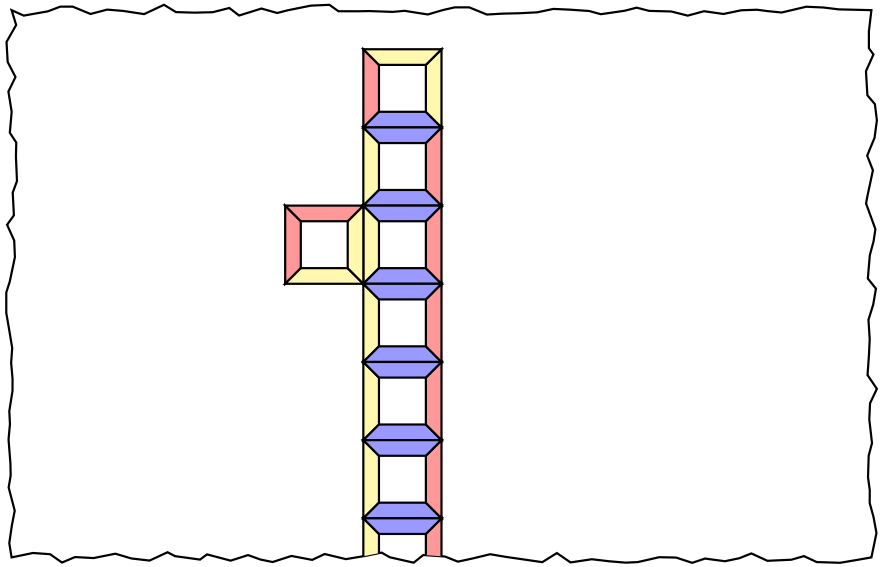
Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

Problème du domino

Jeu de tuile



Un pavage associé

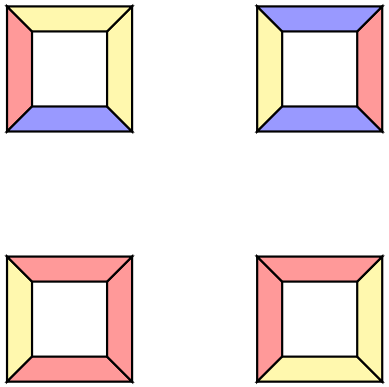


Problème du domino

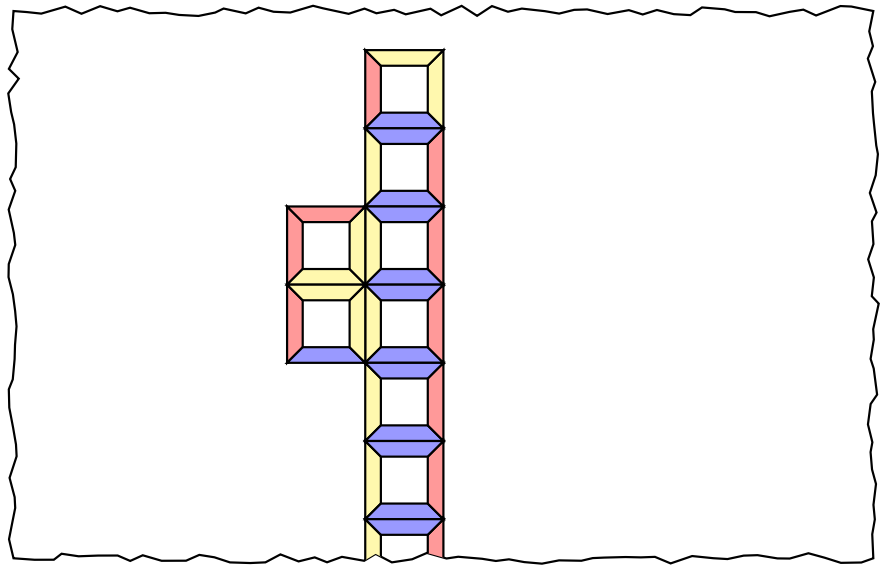
Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

Problème du domino

Jeu de tuile



Un pavage associé

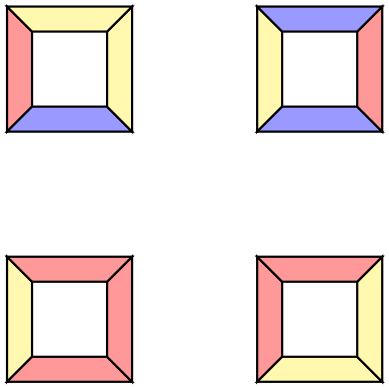


Problème du domino

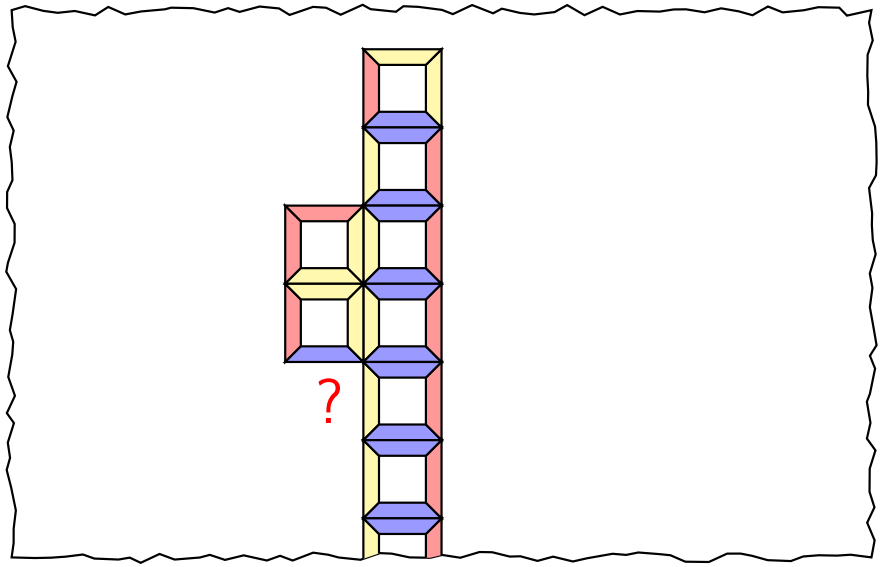
Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

Problème du domino

Jeu de tuile



Un pavage associé



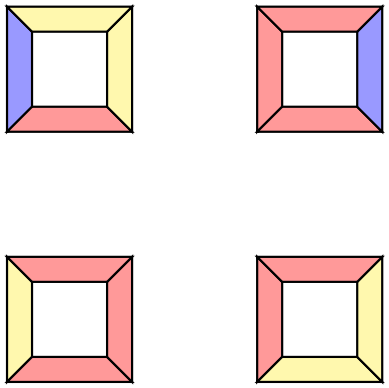
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

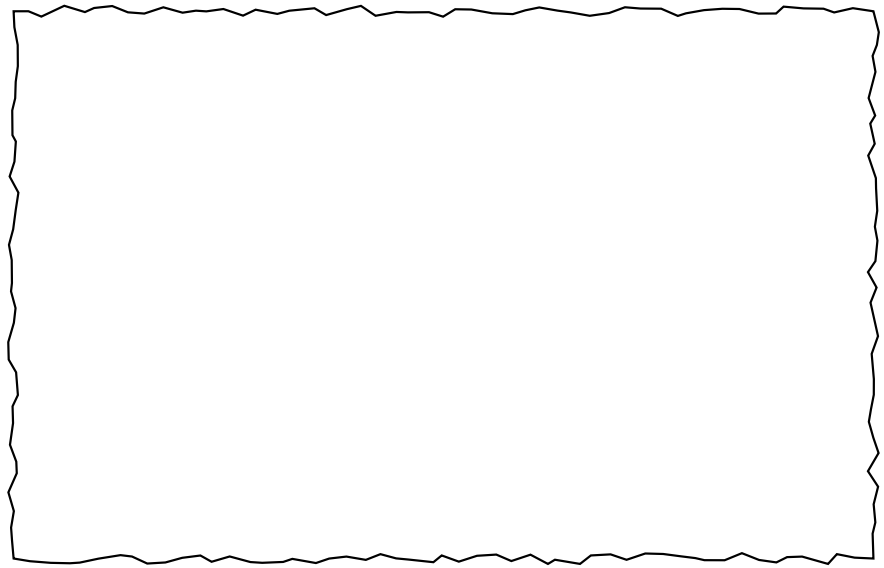
- Le complémentaire du problème du domino est semi-decidable.

Problème du domino

Jeu de tuile



Pavage associé



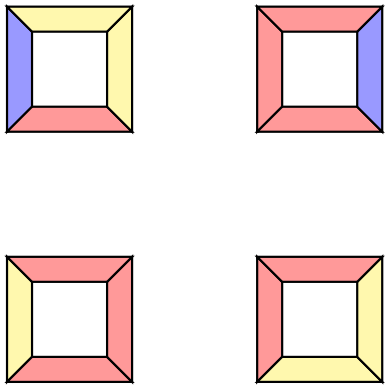
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

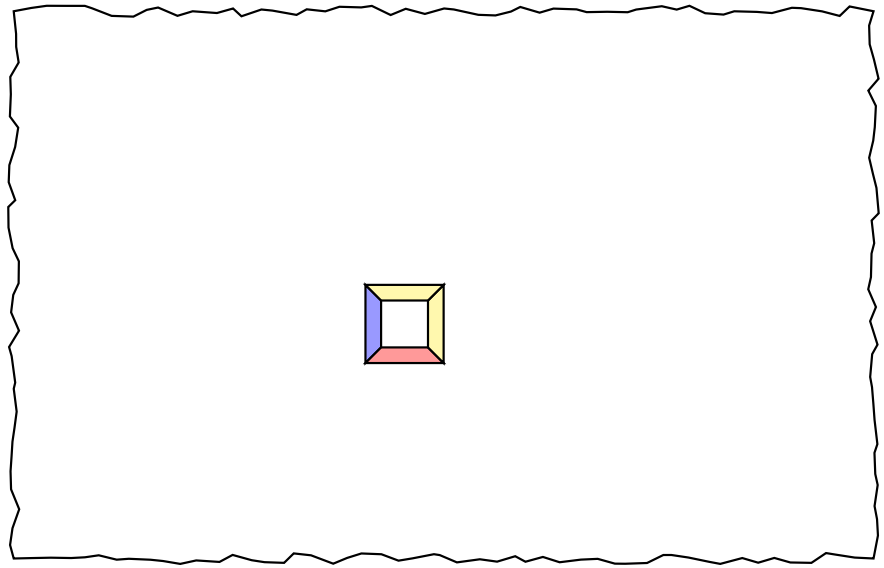
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



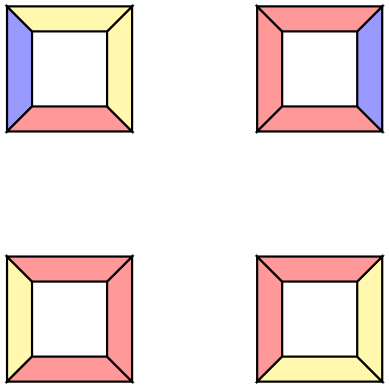
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

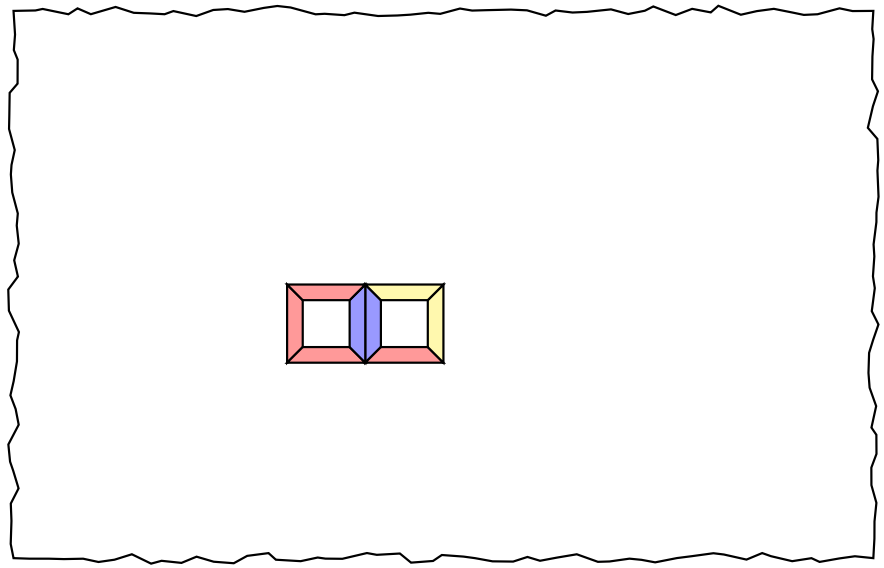
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



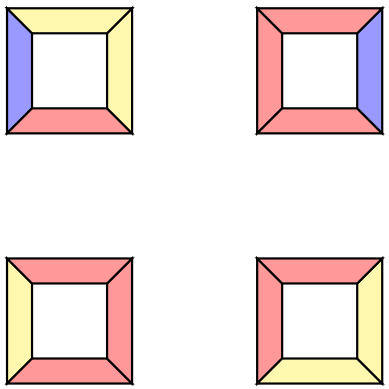
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

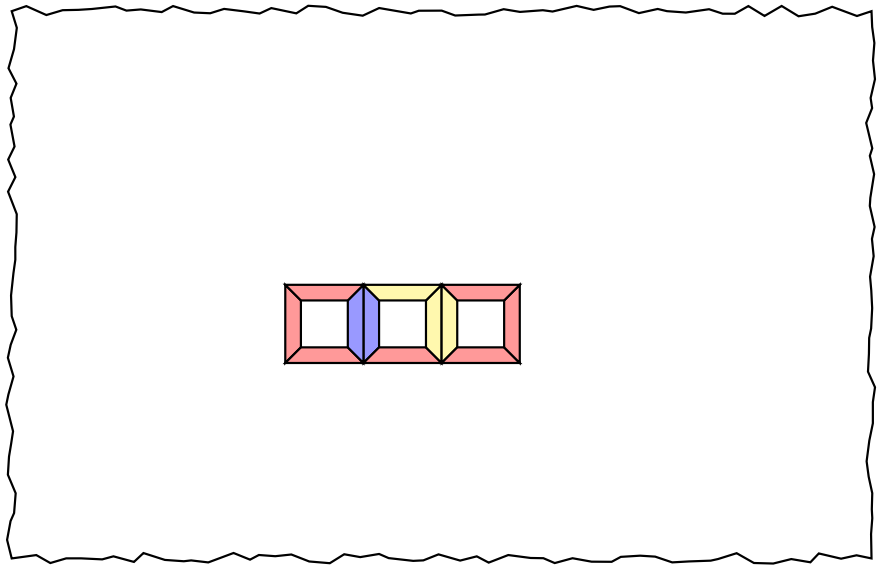
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



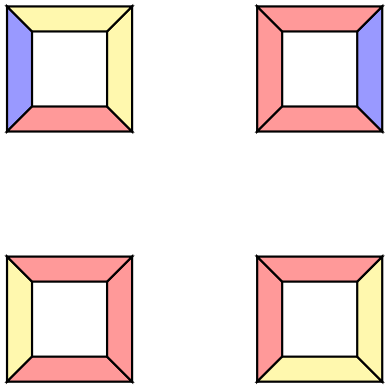
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

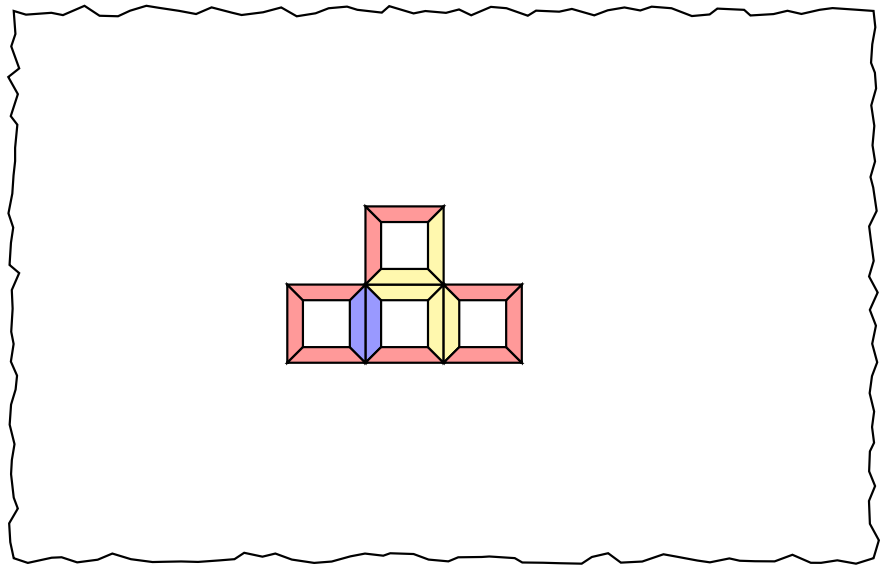
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



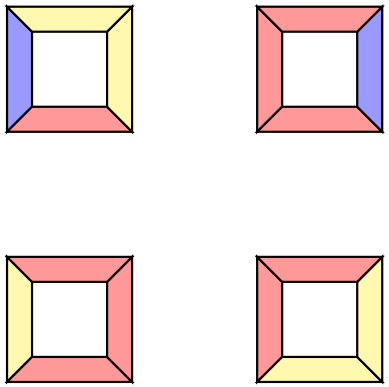
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

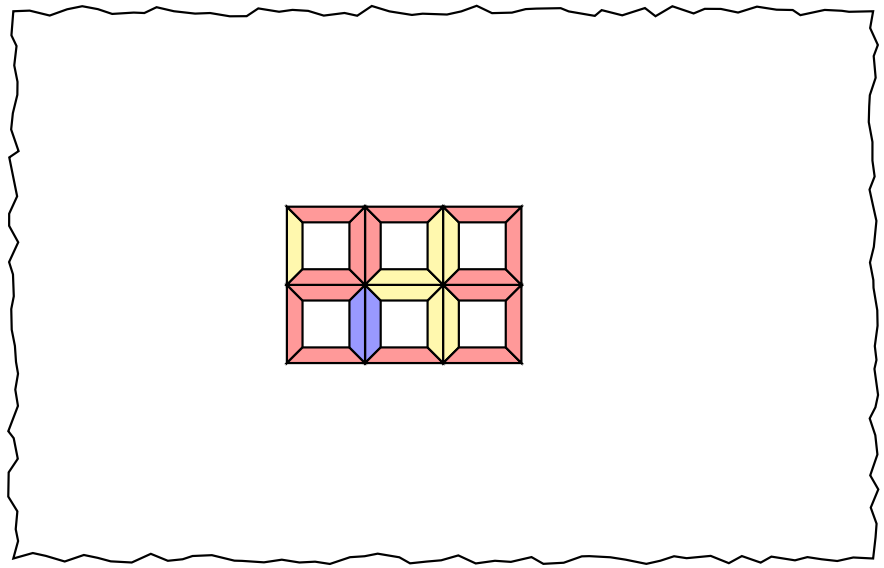
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



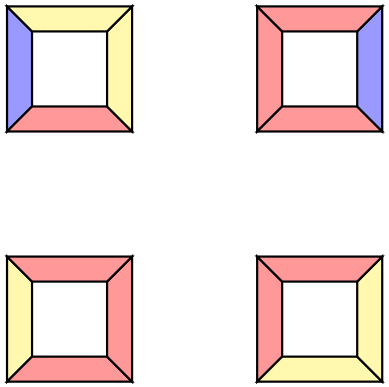
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

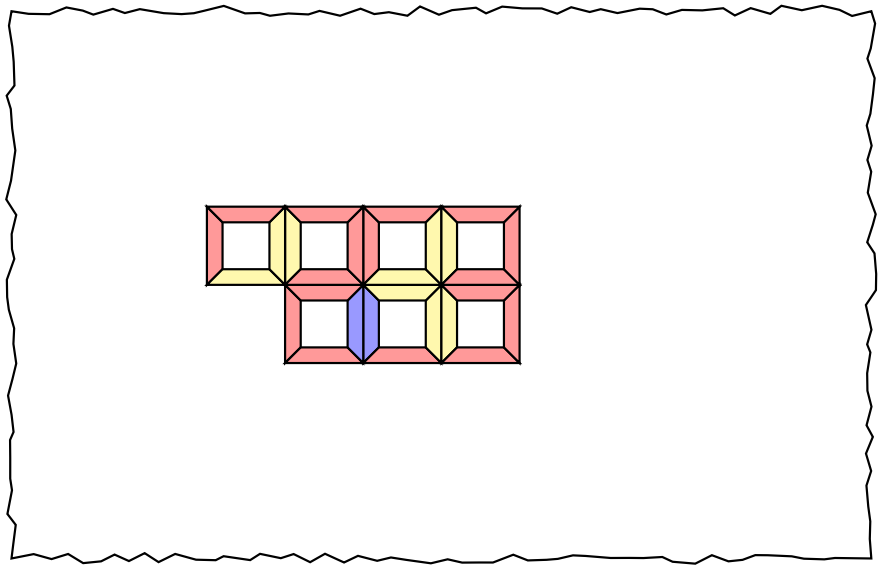
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



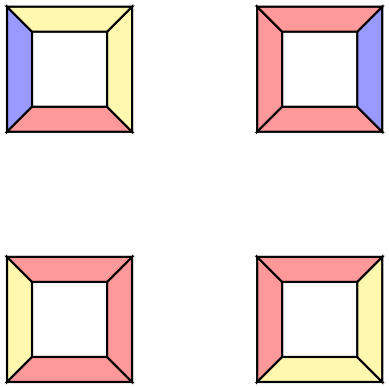
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

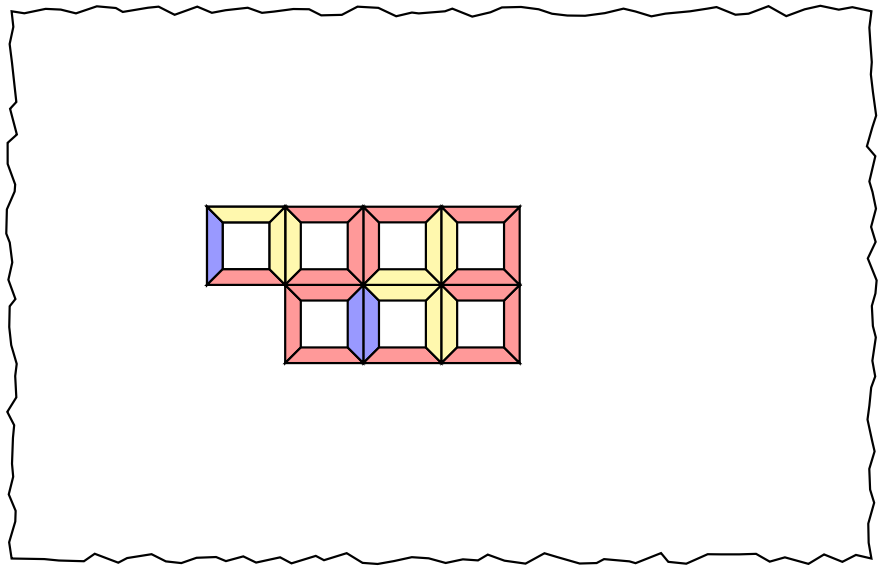
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



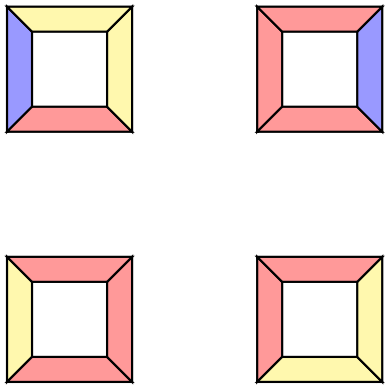
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

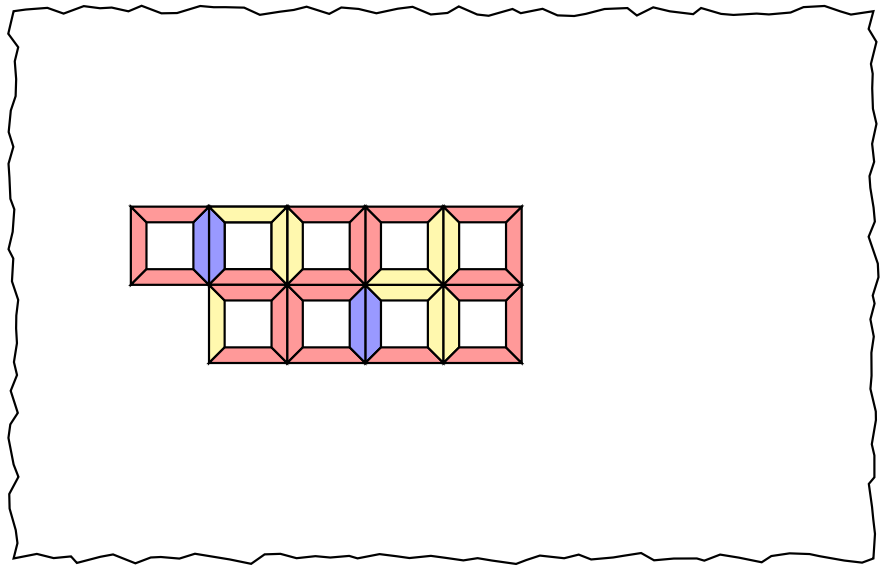
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



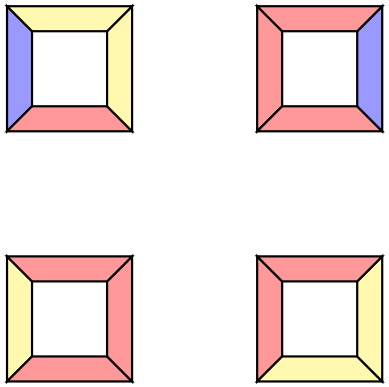
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

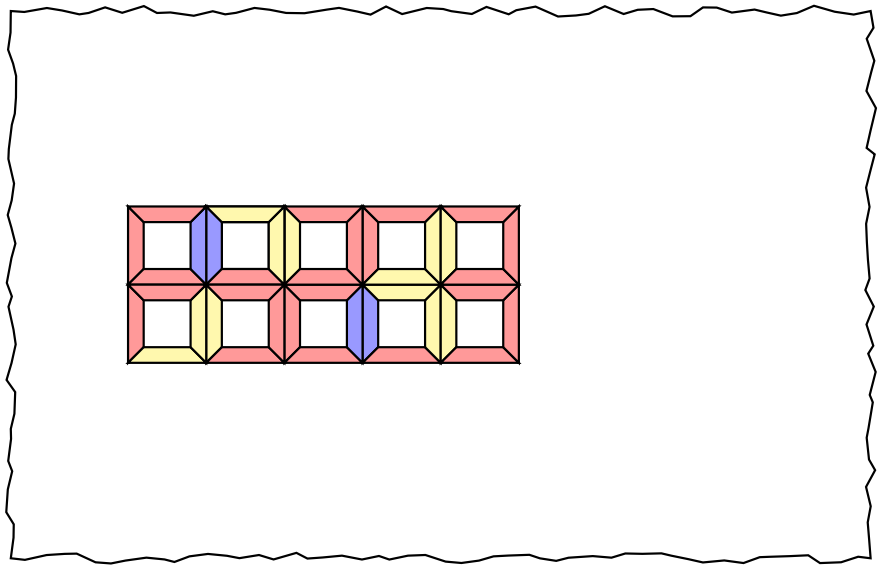
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



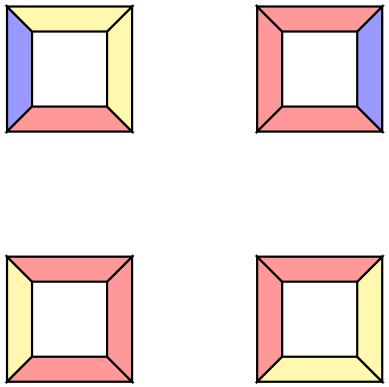
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

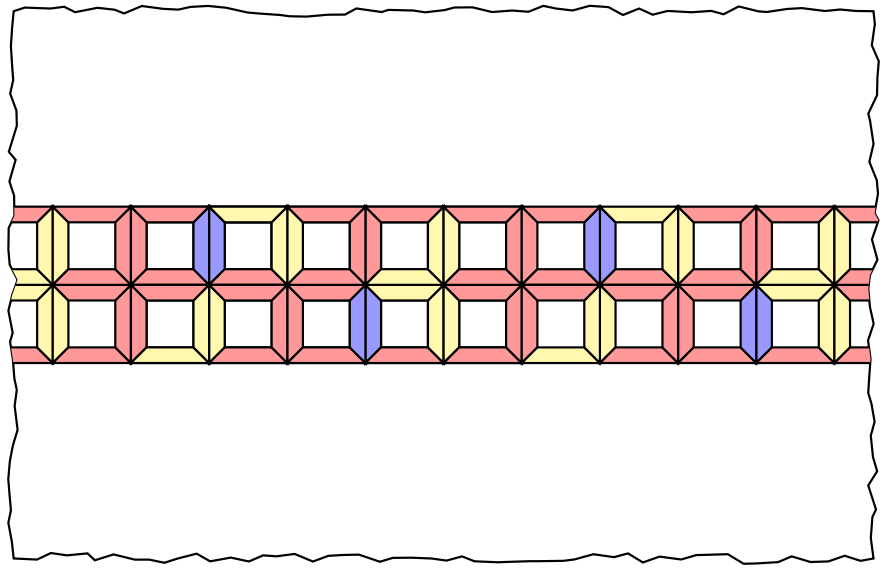
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



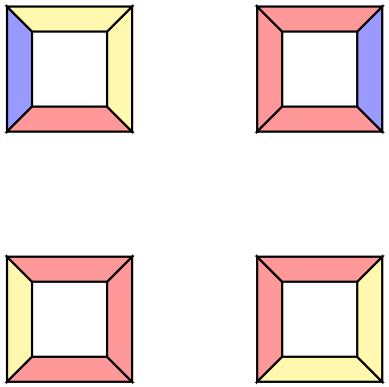
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

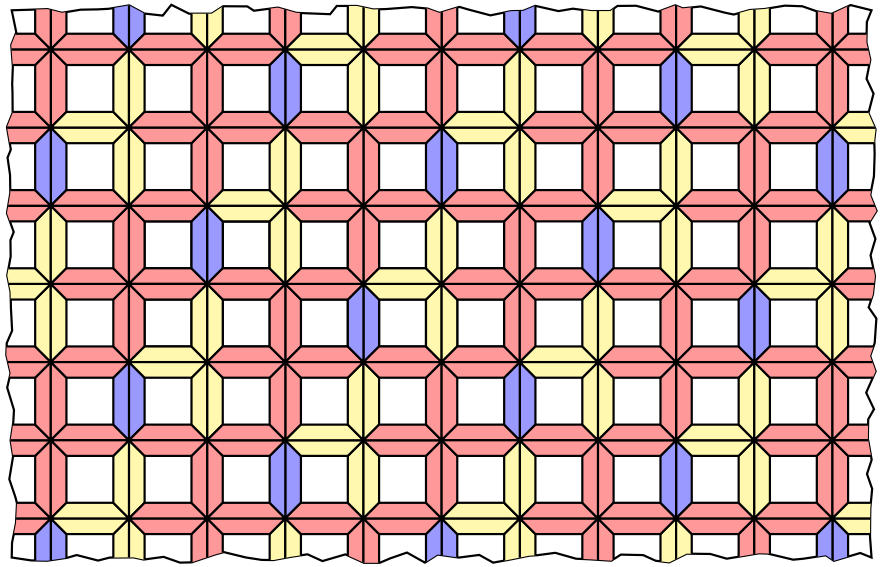
- Le complémentaire du problème du domino est semi-décidable.

Problème du domino

Jeu de tuile



Pavage associé



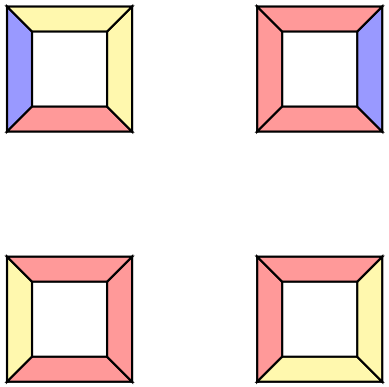
Problème du domino

Etant donné un jeu de tuile $\mathcal{P}$, est ce qu'il est possible de paver le plan ?

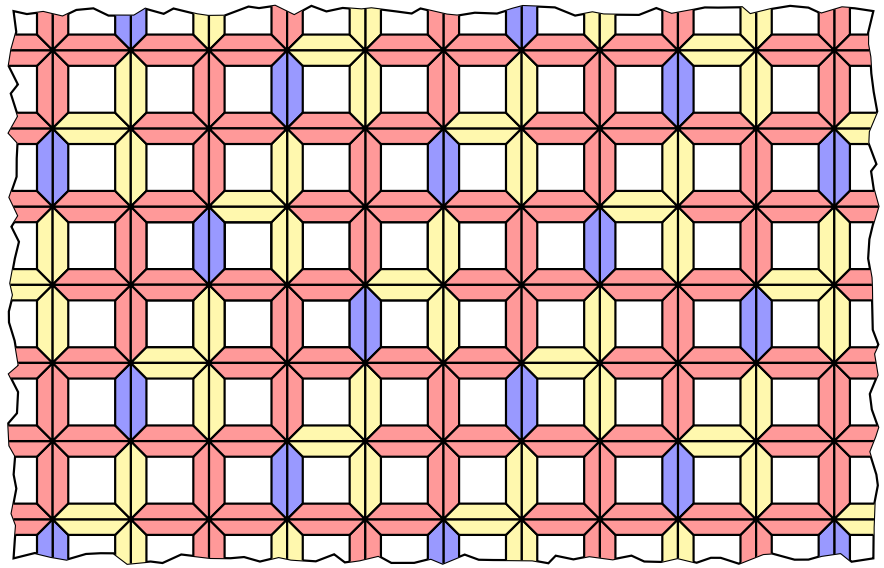
- Le complémentaire du problème du domino est semi-décidable.
- S'il existe un pavage périodique il est facile de trouver

Problème du domino

Jeu de tuile



Pavage associé

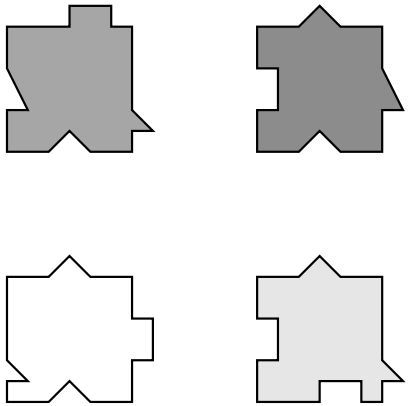


Théorèm *(Berger-66, Robinson-71, Mozes-89, Kari-96...)*

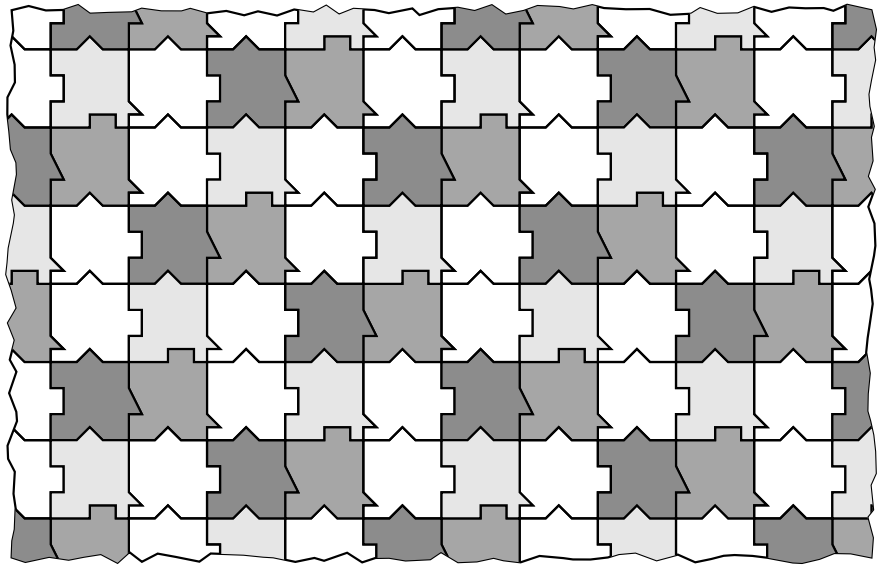
- Il existe des jeux de tuiles qui pavent uniquement périodiquement.
- Le domino problème est indécidable.

Problème du domino

Jeu de tuile



Pavage associé

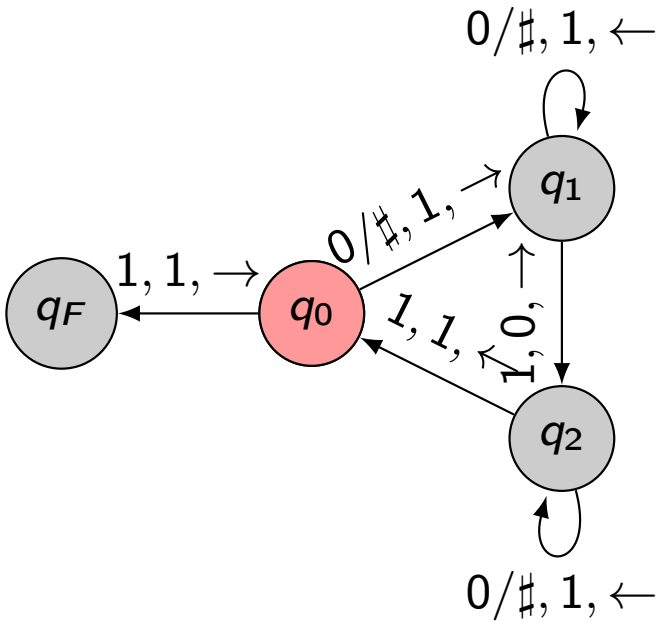


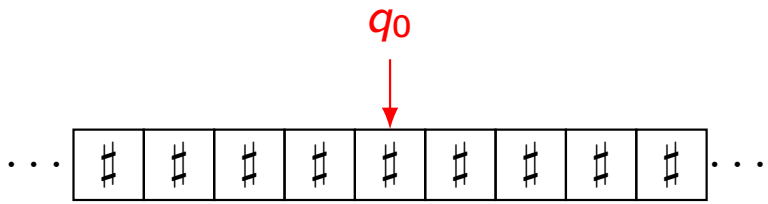
Théorèm *(Berger-66, Robinson-71, Mozes-89, Kari-96...)*

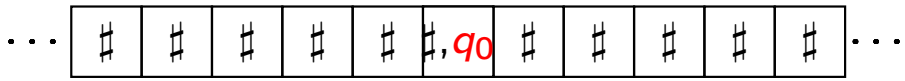
- Il existe des jeux de tuiles qui pavent uniquement périodiquement.
- Le domino problème est indécidable.

Calcul d'une machine de Turing et pavage

Un exemple de modèle de calcul :

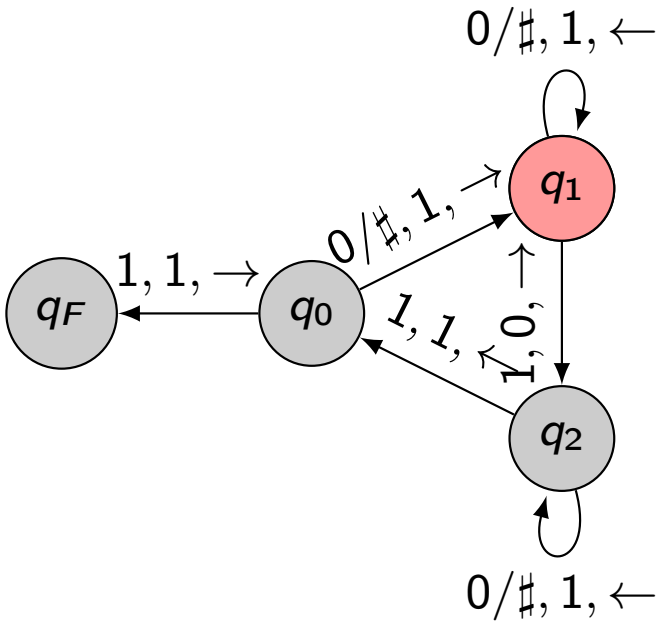


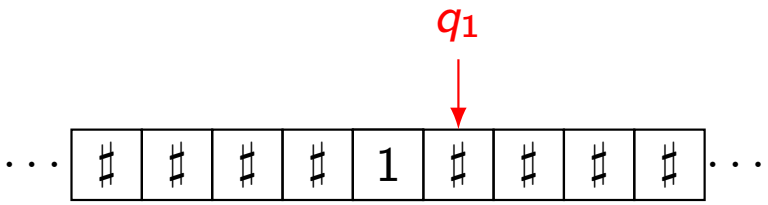


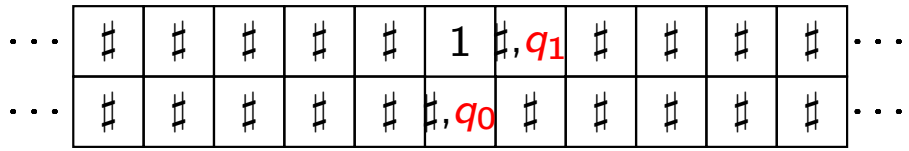


Calcul d'une machine de Turing et pavage

Un exemple de modèle de calcul :

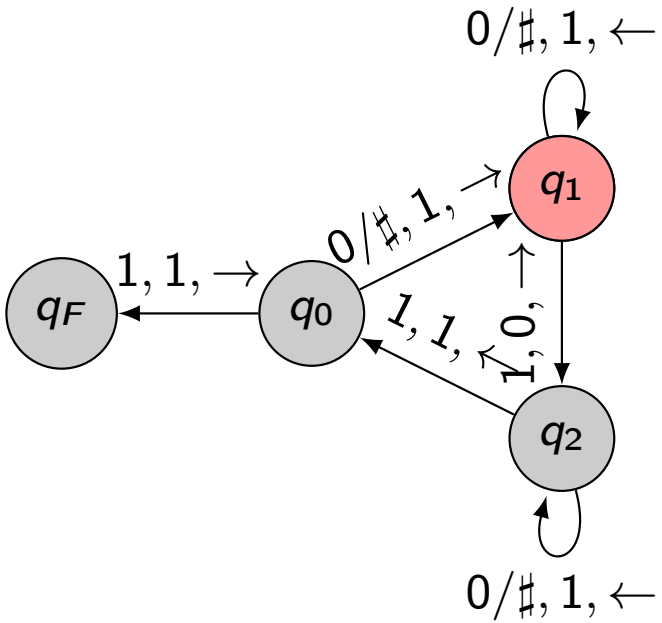


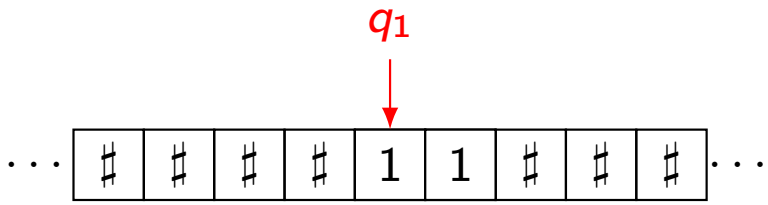


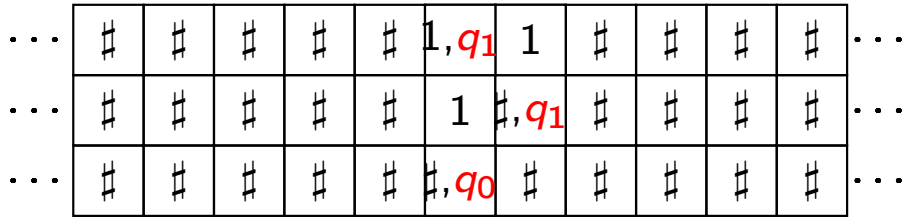


Calcul d'une machine de Turing et pavage

Un exemple de modèle de calcul :

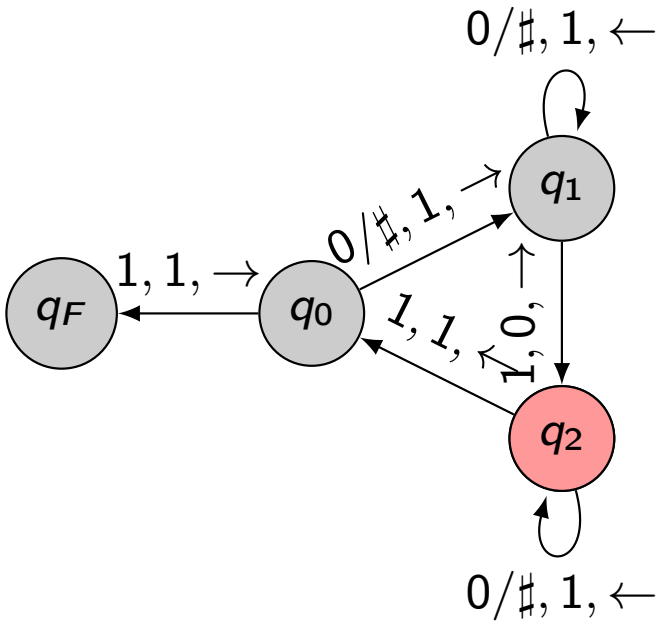


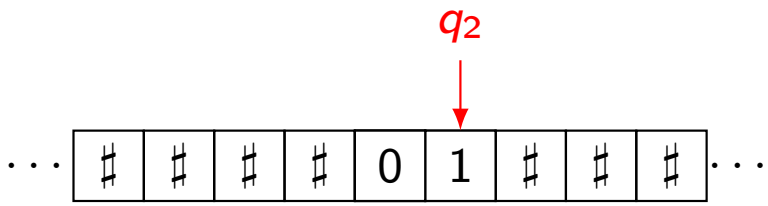




Calcul d'une machine de Turing et pavage

Un exemple de modèle de calcul :

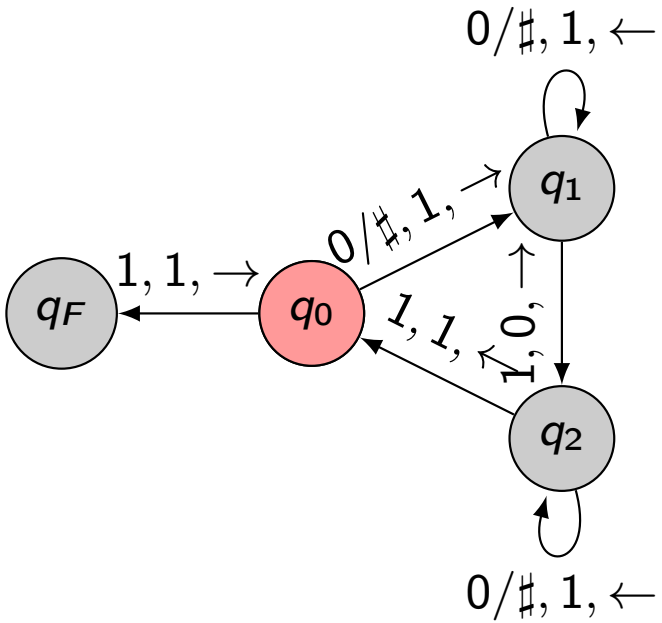


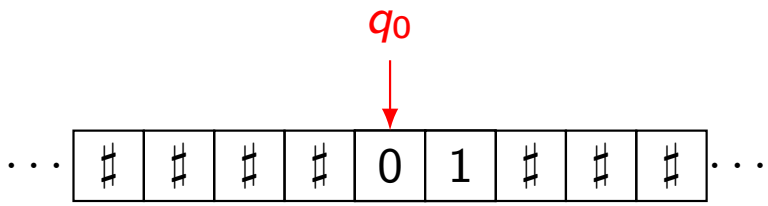


...	♯	♯	♯	♯	♯	0	1, q_2	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1, q_1	1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1	♯, q_1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	♯, q_0	♯	♯	♯	♯	♯ ...

Calcul d'une machine de Turing et pavage

Un exemple de modèle de calcul :

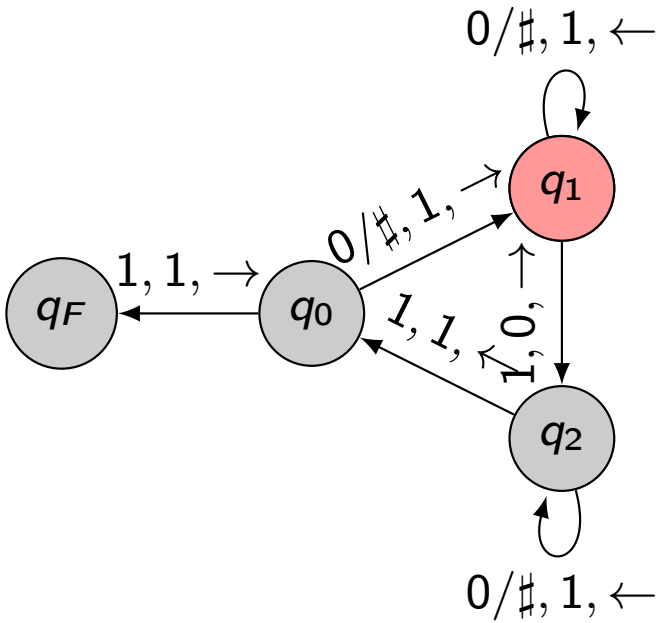


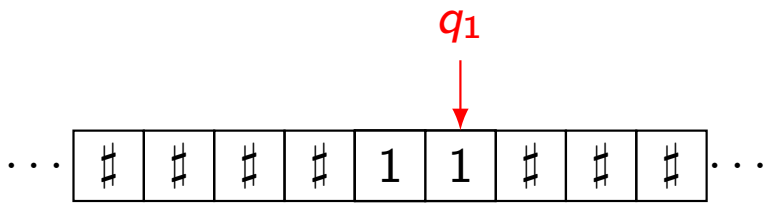


...	♯	♯	♯	♯	♯	0, q_0	1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	0	1, q_2	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1, q_1	1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1	♯, q_1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	♯, q_0	♯	♯	♯	♯	♯ ...

Calcul d'une machine de Turing et pavage

Un exemple de modèle de calcul :

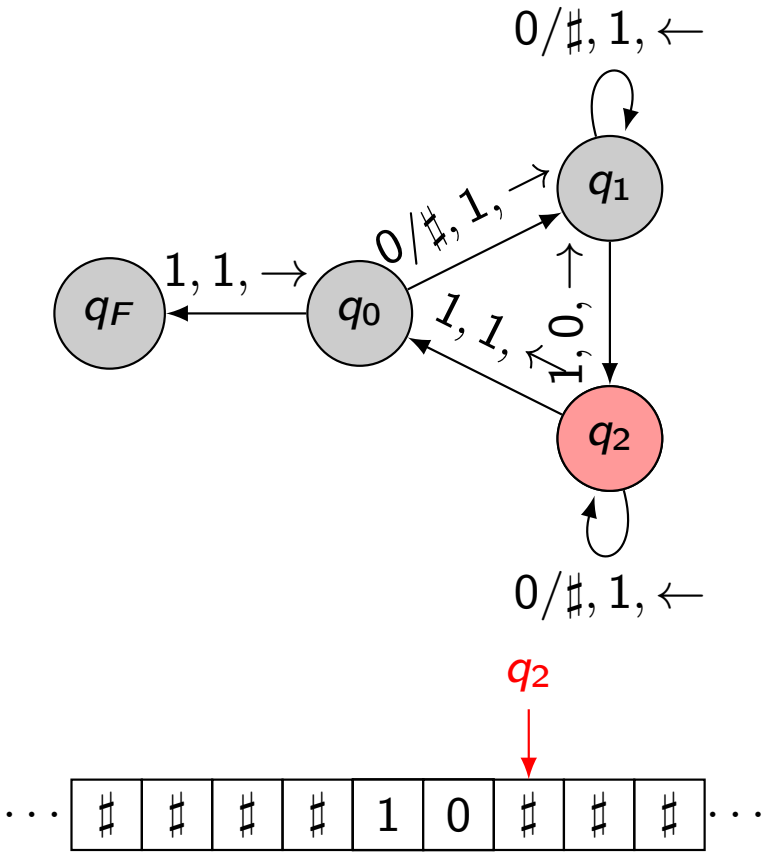




♯	♯	♯	♯	♯	1	1, q_1	♯	♯	♯	♯	
♯	♯	♯	♯	♯	0, q_0	1	♯	♯	♯	♯	
♯	♯	♯	♯	♯	0	1, q_2	♯	♯	♯	♯	
♯	♯	♯	♯	♯	1, q_1	1	♯	♯	♯	♯	
♯	♯	♯	♯	♯	1	♯, q_1	♯	♯	♯	♯	
♯	♯	♯	♯	♯	♯, q_0	♯	♯	♯	♯	♯	

Calcul d'une machine de Turing et pavage

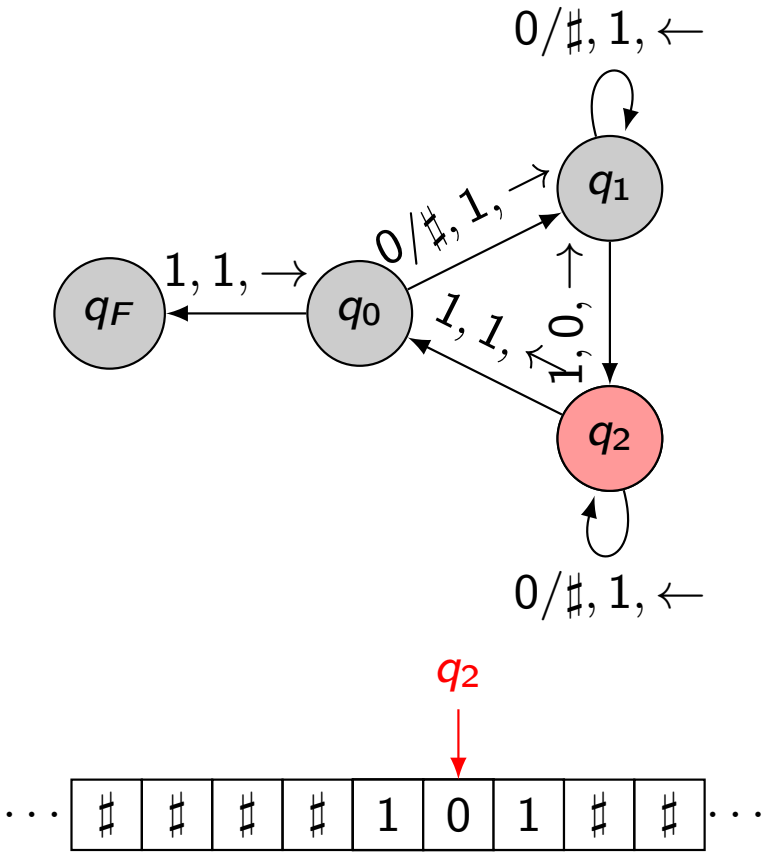
Un exemple de modèle de calcul :



♯	♯	♯	♯	♯	1	0	♯, q_2	♯	♯	♯	
♯	♯	♯	♯	♯	1	1, q_1	♯	♯	♯	♯	
♯	♯	♯	♯	♯	0, q_0	1	♯	♯	♯	♯	
♯	♯	♯	♯	♯	0	1, q_2	♯	♯	♯	♯	
♯	♯	♯	♯	♯	1, q_1	1	♯	♯	♯	♯	
♯	♯	♯	♯	♯	1	♯, q_1	♯	♯	♯	♯	
♯	♯	♯	♯	♯	♯, q_0	♯	♯	♯	♯	♯	

Calcul d'une machine de Turing et pavage

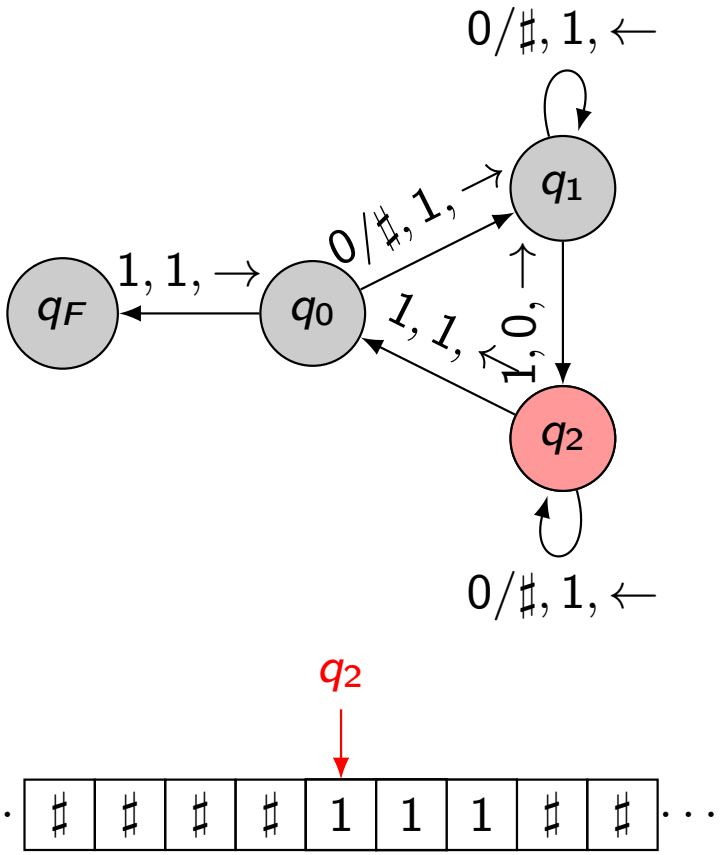
Un exemple de modèle de calcul :

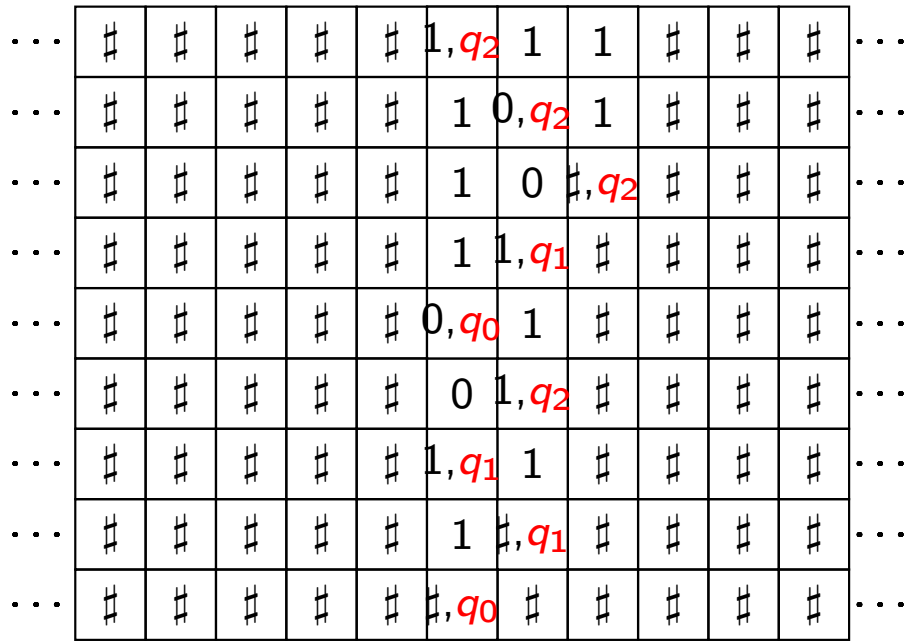


...	♯	♯	♯	♯	♯	1	0, q_2	1	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1	0	♯, q_2	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1	1, q_1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	0, q_0	1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	0	1, q_2	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1, q_1	1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1	♯, q_1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	♯, q_0	♯	♯	♯	♯	♯ ...

Calcul d'une machine de Turing et pavage

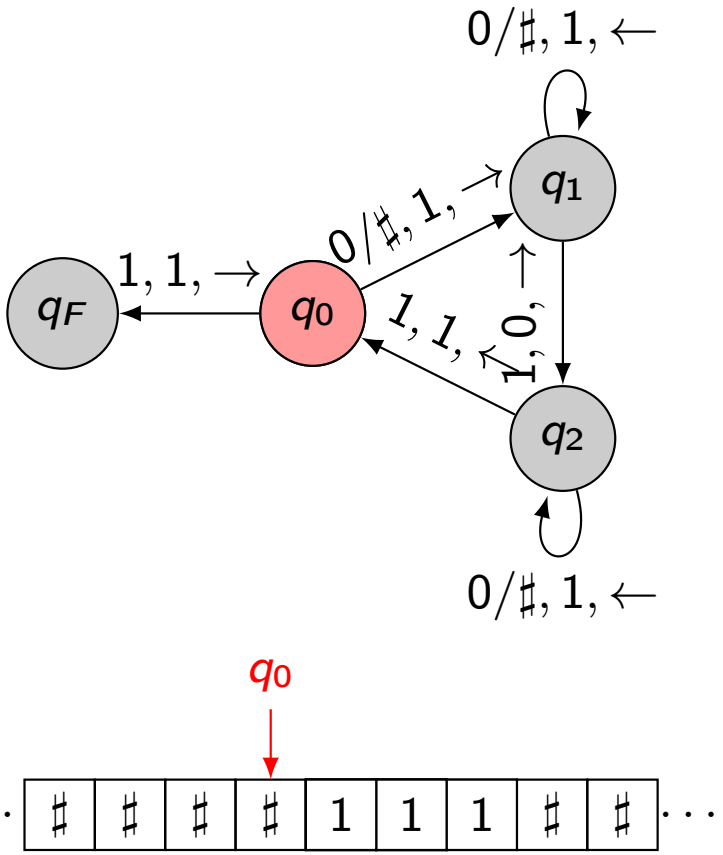
Un exemple de modèle de calcul :





Calcul d'une machine de Turing et pavage

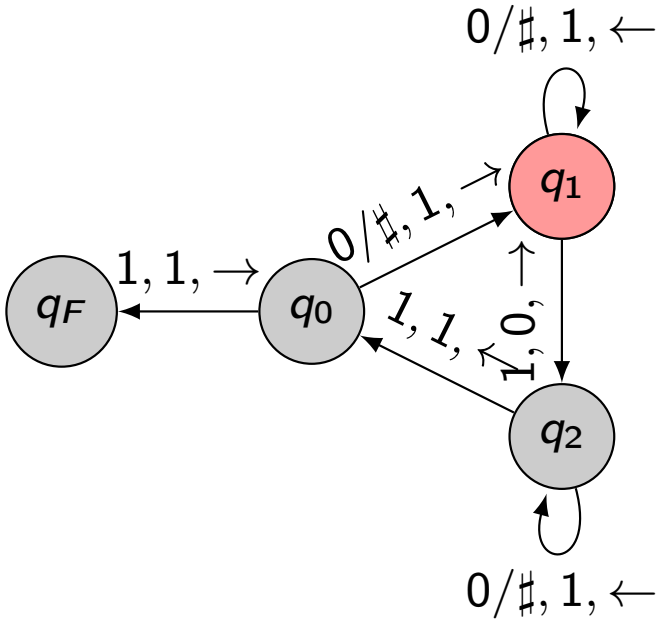
Un exemple de modèle de calcul :

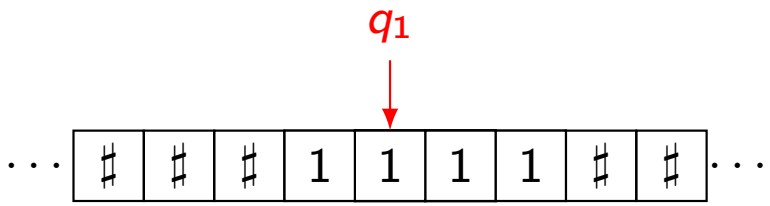


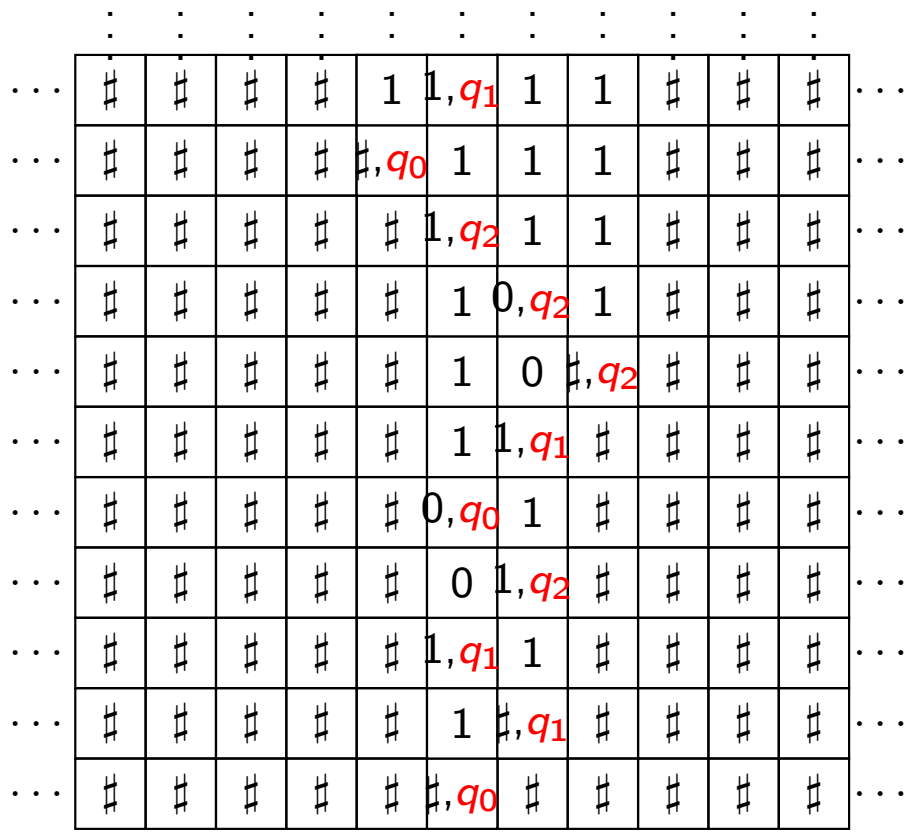
...	♯	♯	♯	♯	♯,q_0	1	1	1	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1,q_2	1	1	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1	0,q_2	1	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1	0	♯,q_2	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1	1,q_1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	0,q_0	1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	0	1,q_2	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1,q_1	1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	1	♯,q_1	♯	♯	♯	♯ ...
...	♯	♯	♯	♯	♯	♯,q_0	♯	♯	♯	♯	♯ ...

Calcul d'une machine de Turing et pavage

Un exemple de modèle de calcul :

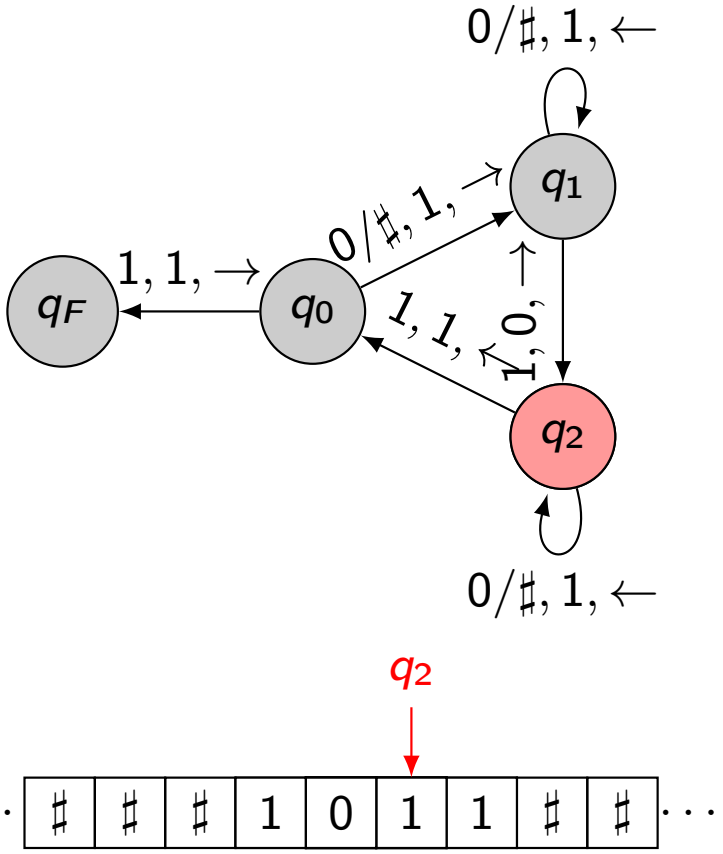


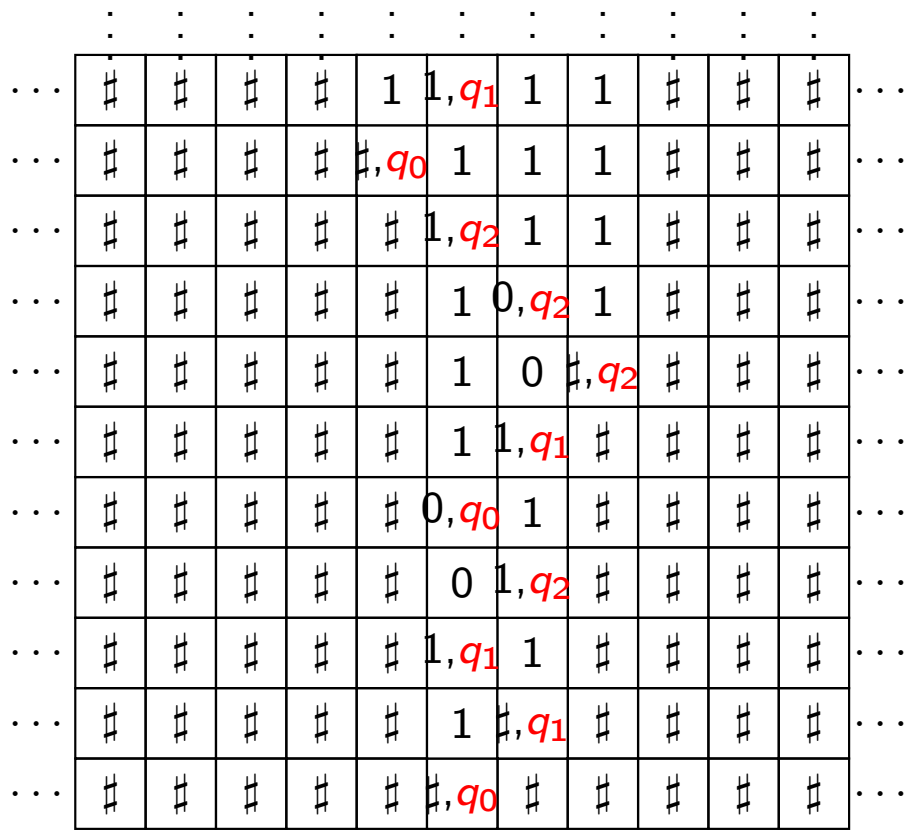




Calcul d'une machine de Turing et pavage

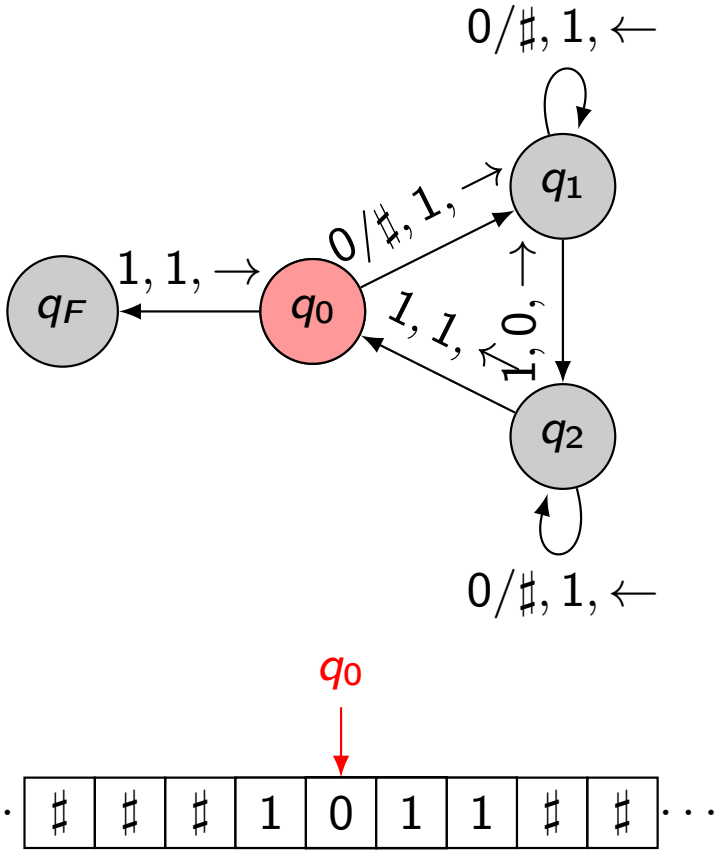
Un exemple de modèle de calcul :

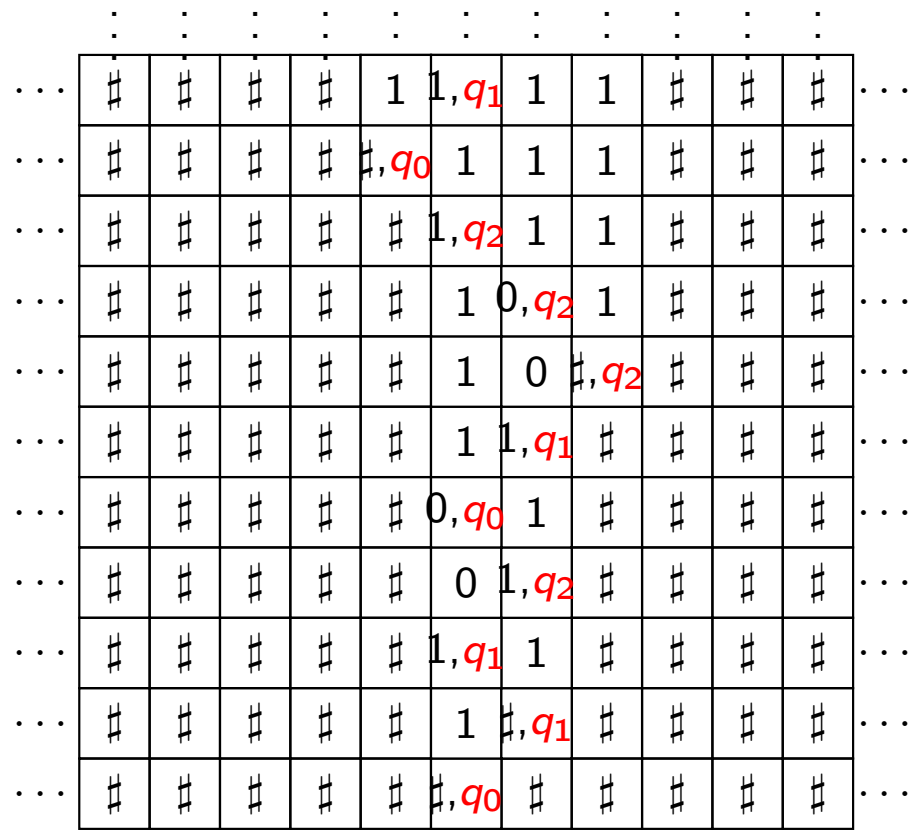




Calcul d'une machine de Turing et pavage

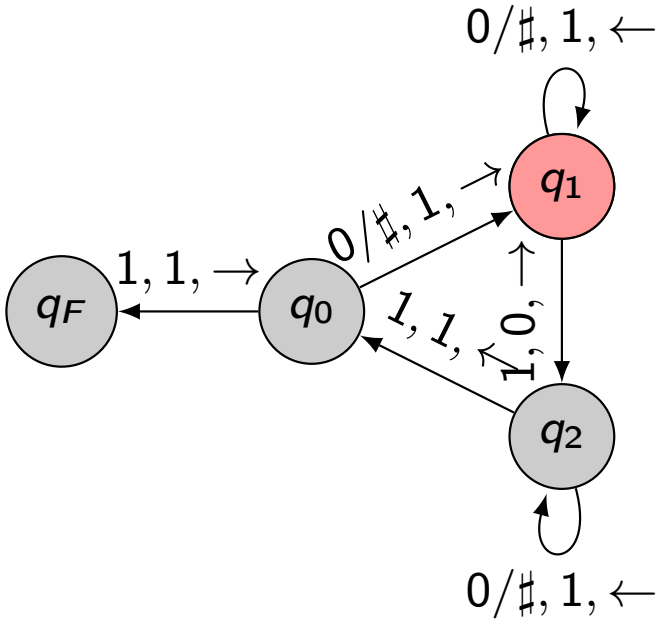
Un exemple de modèle de calcul :

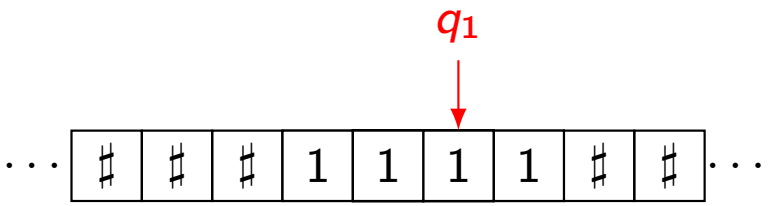


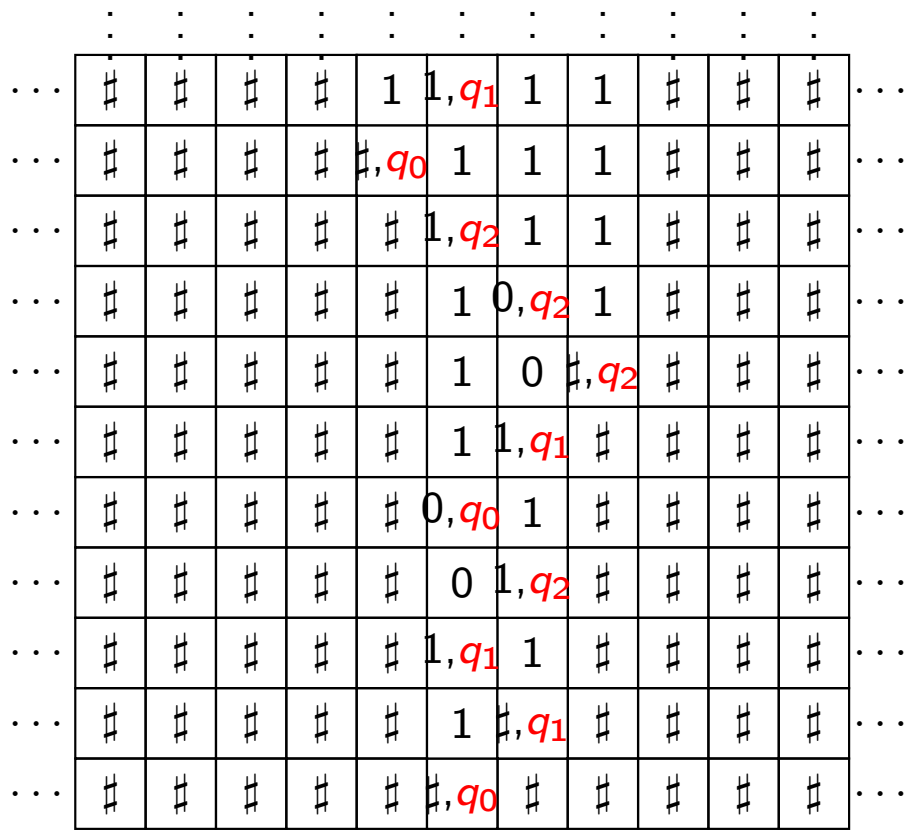


Calcul d'une machine de Turing et pavage

Un exemple de modèle de calcul :

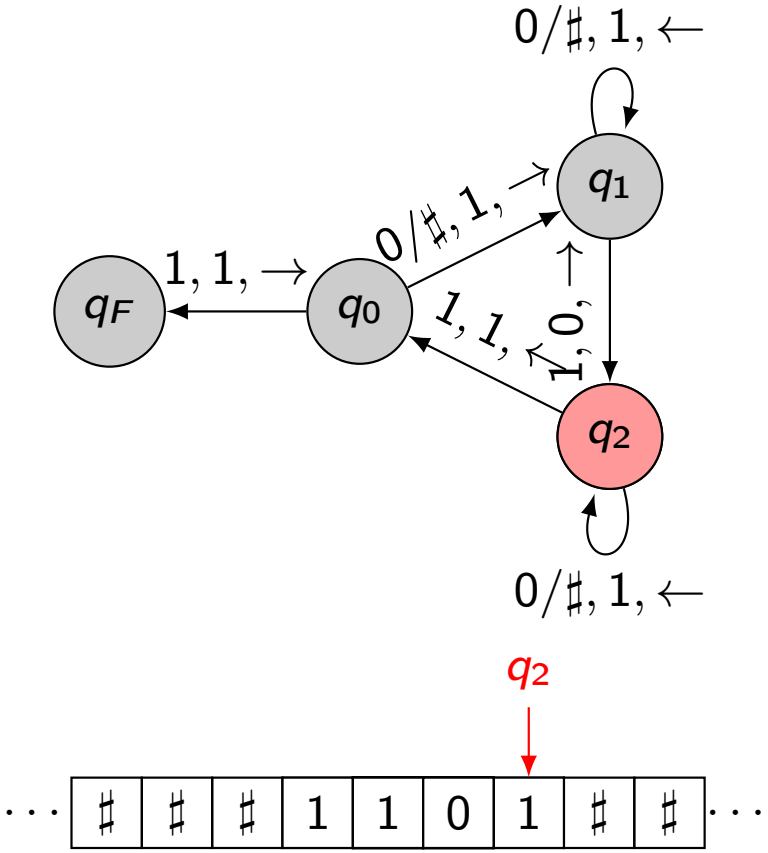


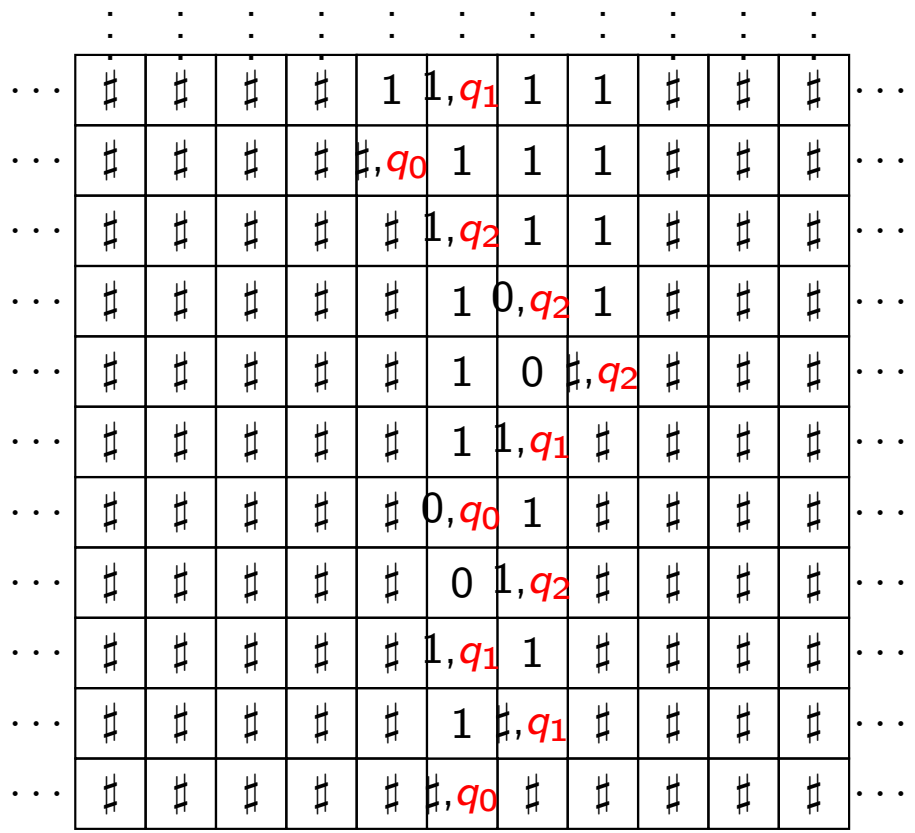




Calcul d'une machine de Turing et pavage

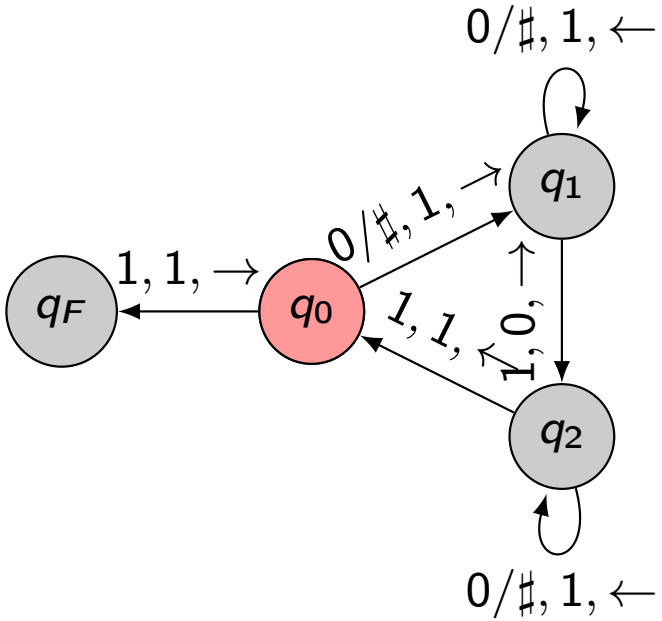
Un exemple de modèle de calcul :

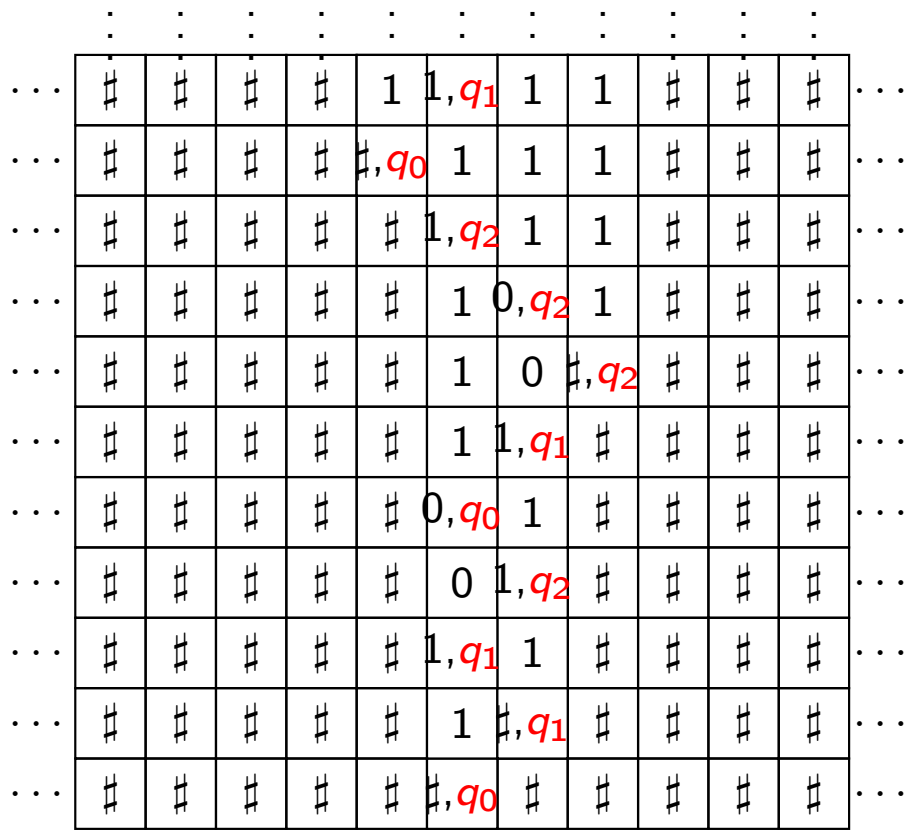




Calcul d'une machine de Turing et pavage

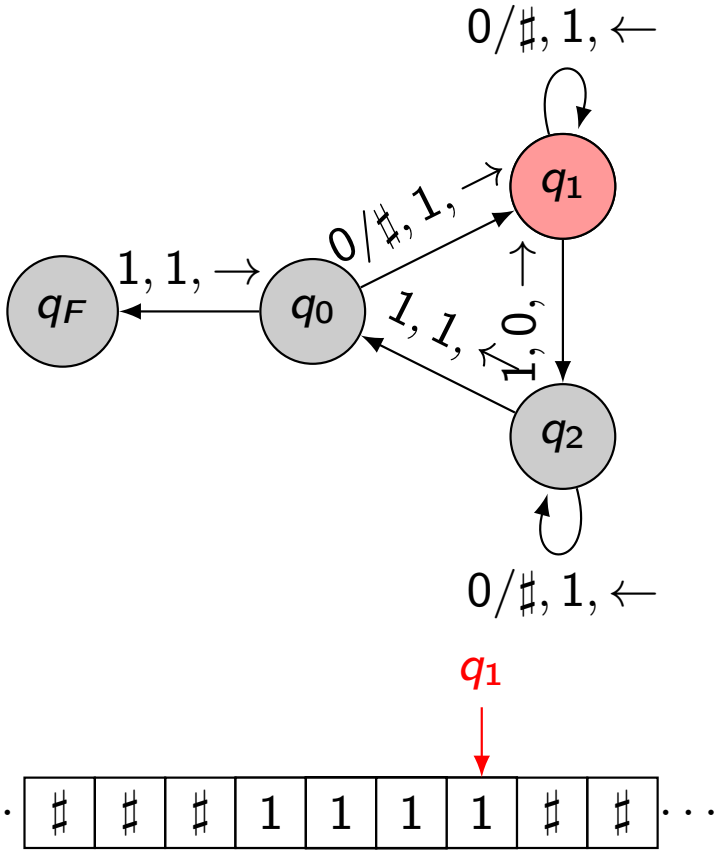
Un exemple de modèle de calcul :

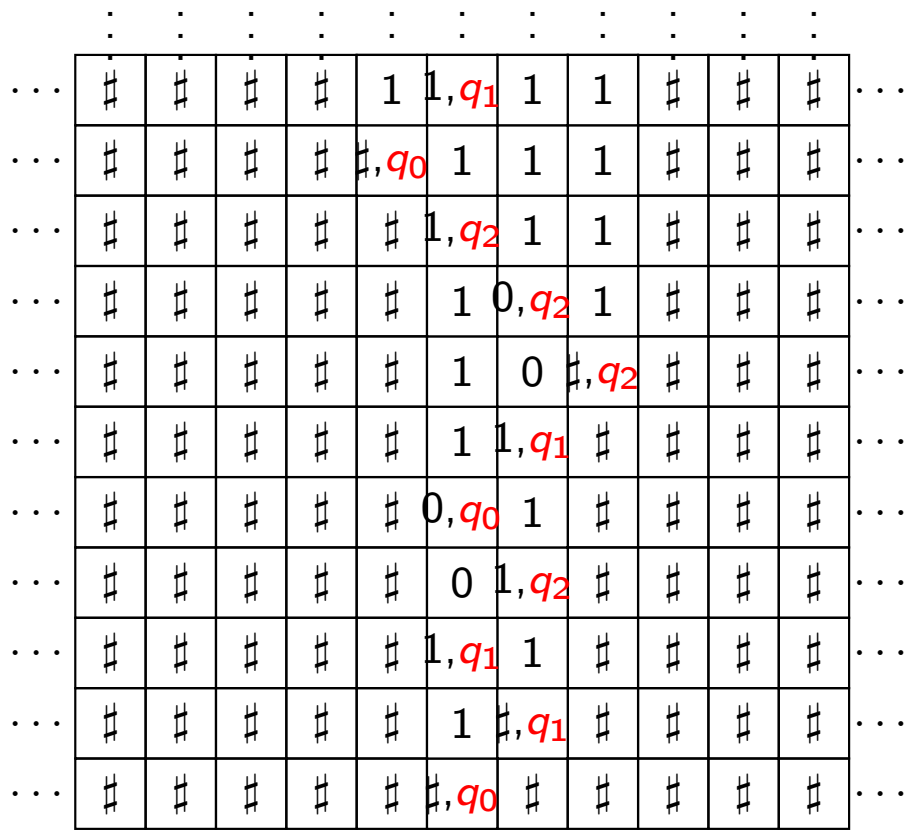




Calcul d'une machine de Turing et pavage

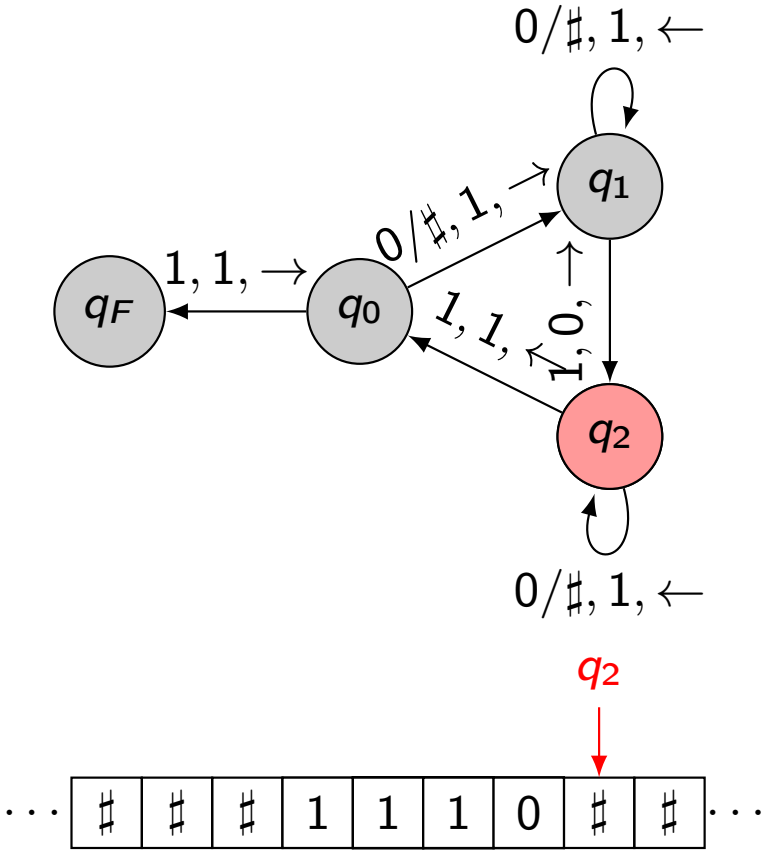
Un exemple de modèle de calcul :

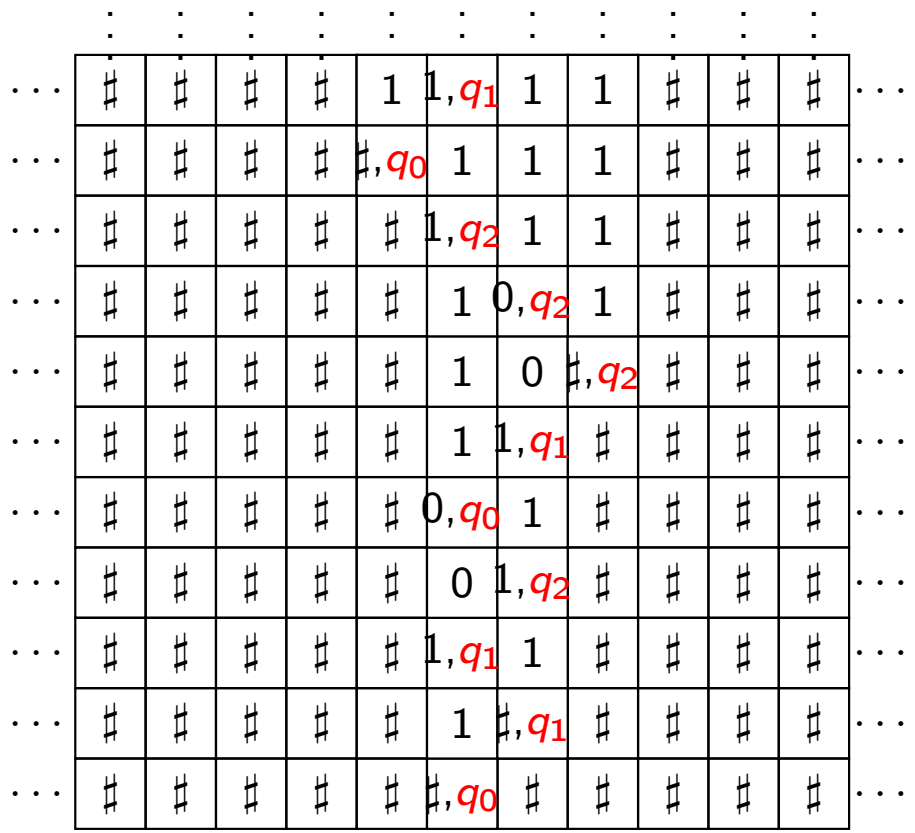




Calcul d'une machine de Turing et pavage

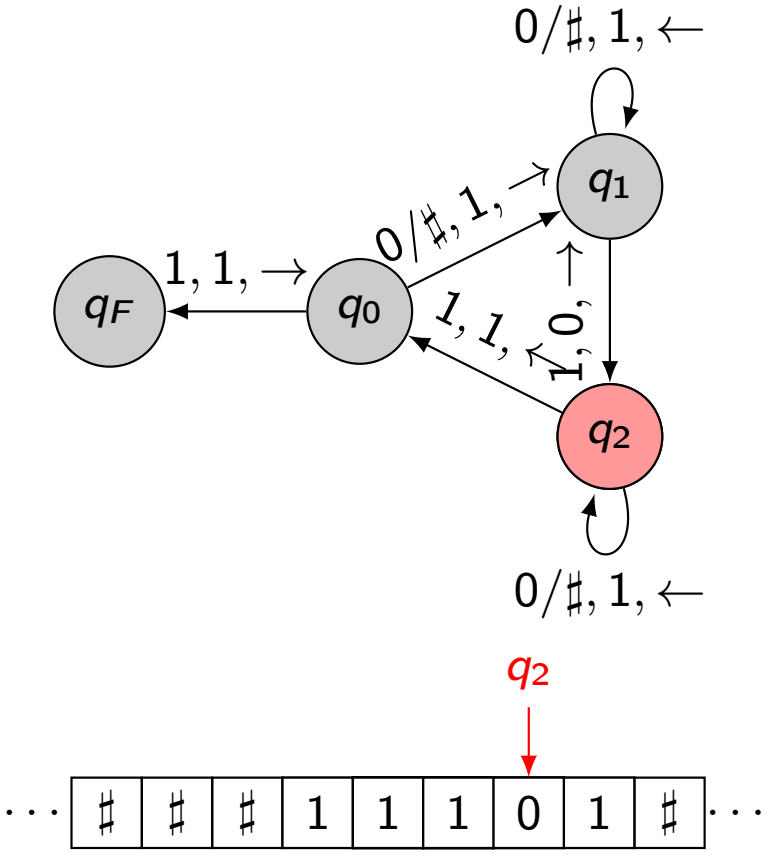
Un exemple de modèle de calcul :

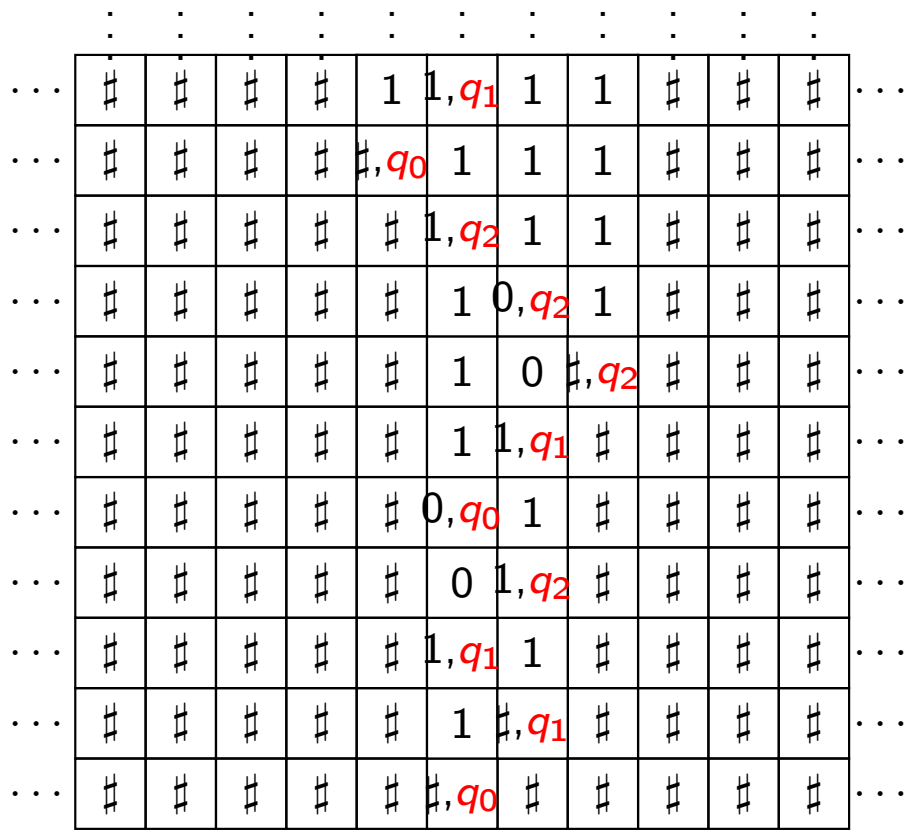




Calcul d'une machine de Turing et pavage

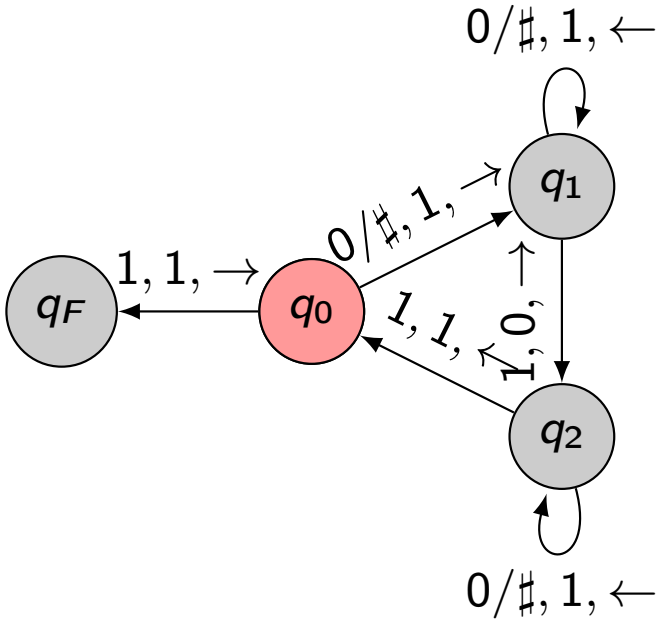
Un exemple de modèle de calcul :

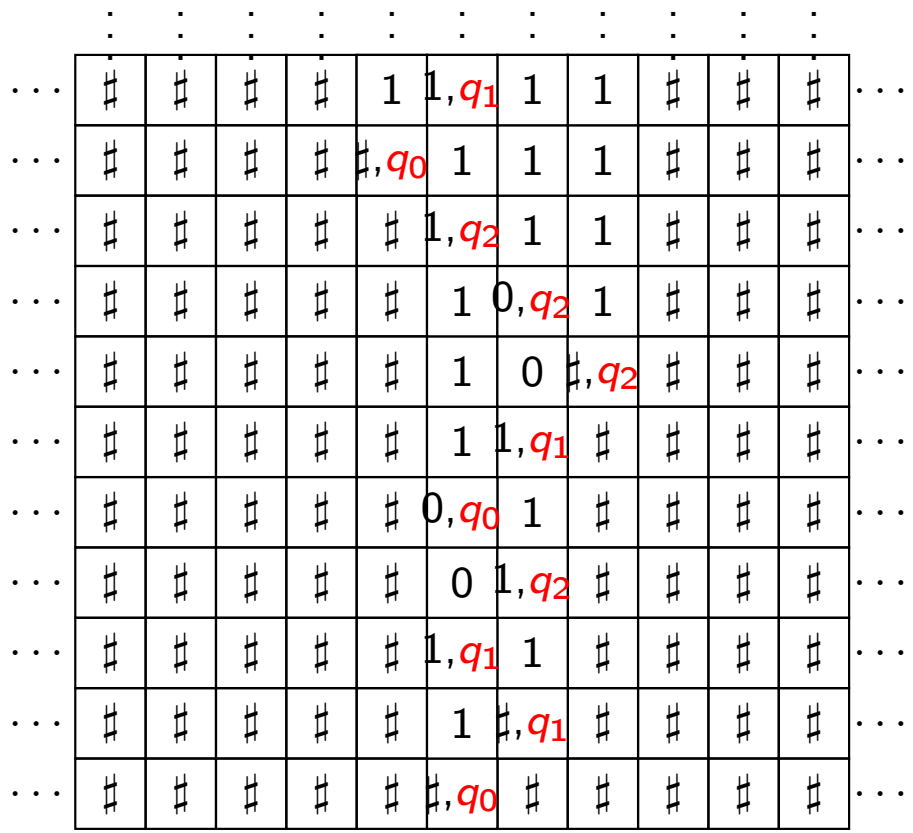




Calcul d'une machine de Turing et pavage

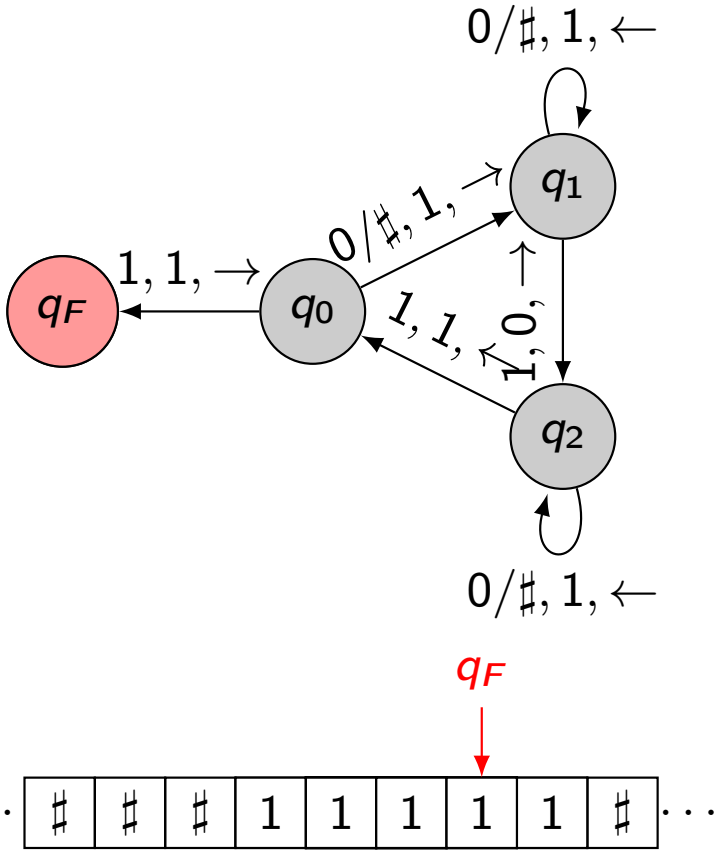
Un exemple de modèle de calcul :

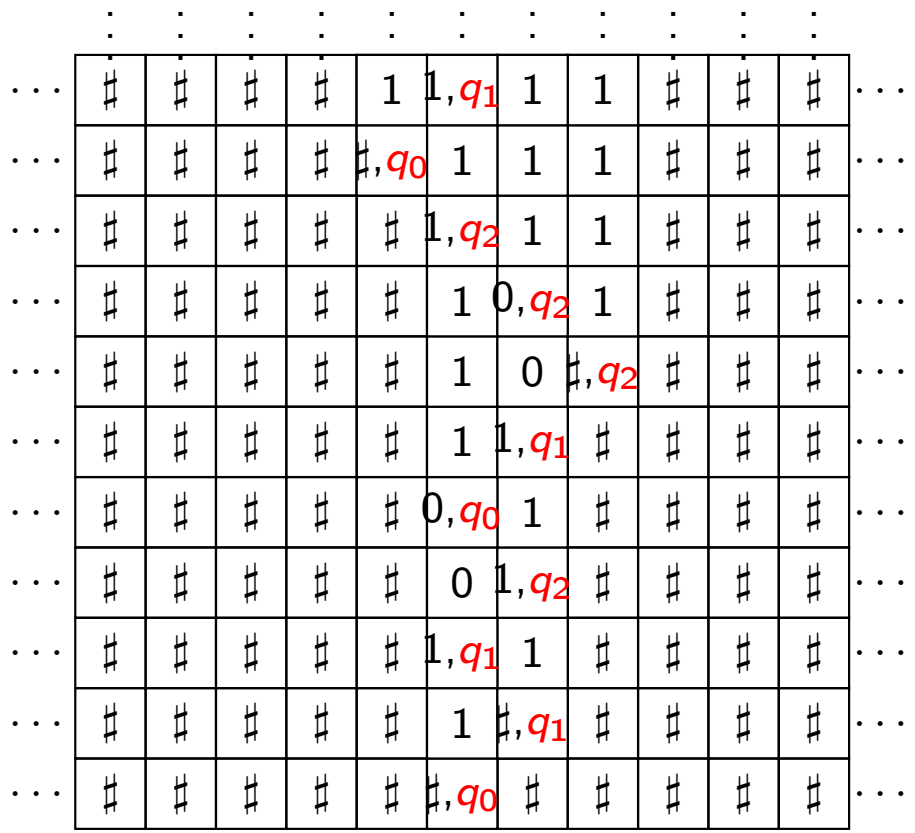




Calcul d'une machine de Turing et pavage

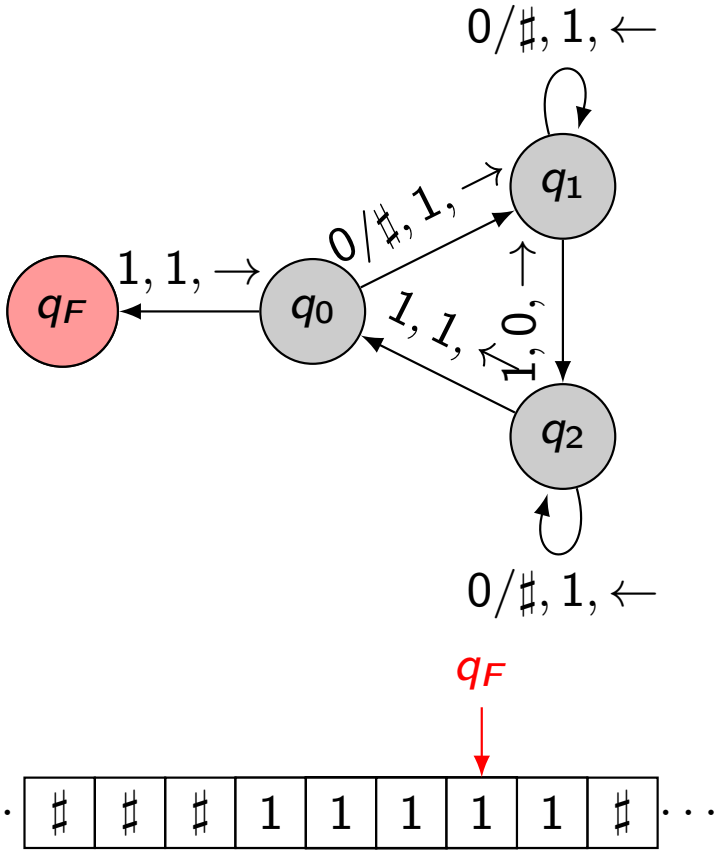
Un exemple de modèle de calcul :

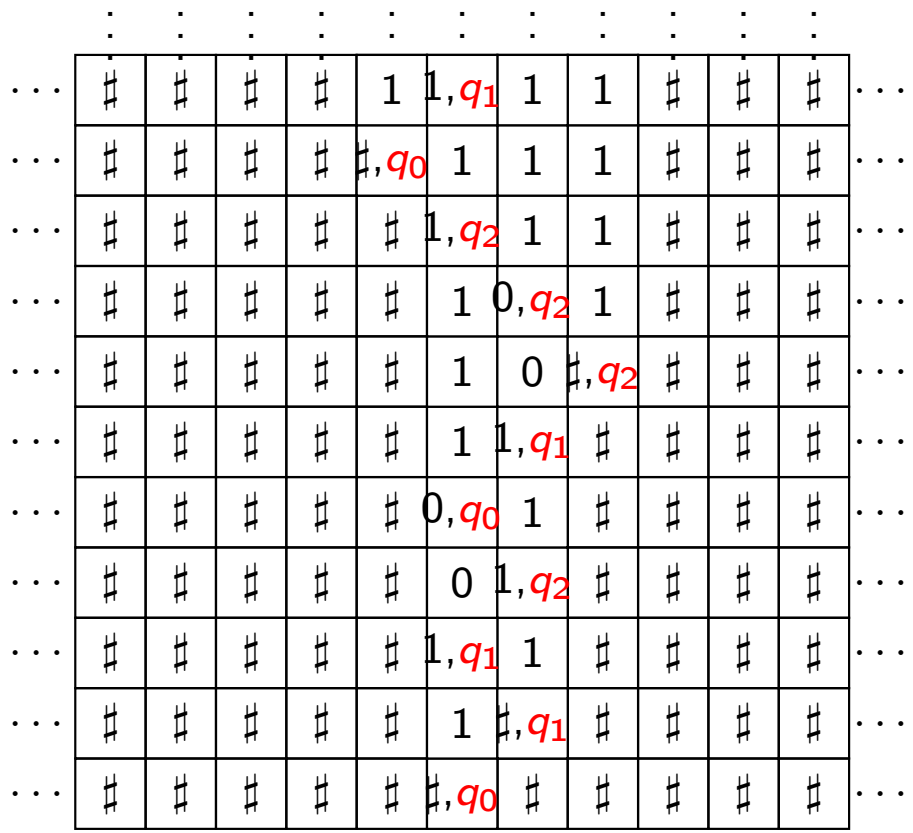




Calcul d'une machine de Turing et pavage

Un exemple de modèle de calcul :





Theorem *(Turing 1937)*

Le problème de l'arrêt est indécidable (mais il est semi-decidable).

Problème de complétion

Problème de complétion

Etant donné un jeu de tuile et un motif p, est-il possible de paver avec le jeu de tuile partant du motif p ?

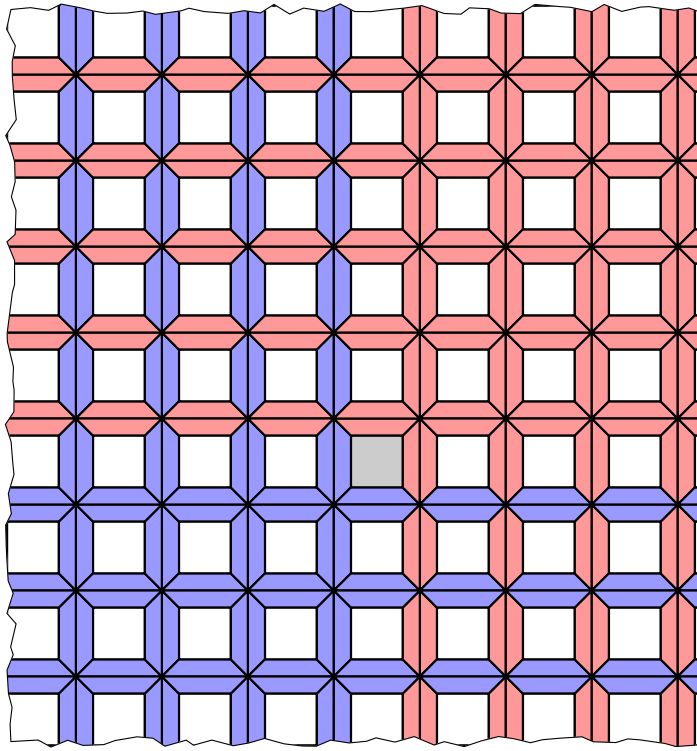
Problème de complétion

Problème de complétion

Etant donné un jeu de tuile et un motif p, est-il possible de paver avec le jeu de tuile partant du motif p ?

Considérons

$$\mathbf{T}(\mathcal{F}_{\leq 1}) \subset \mathcal{A}_{\leq 1}$$



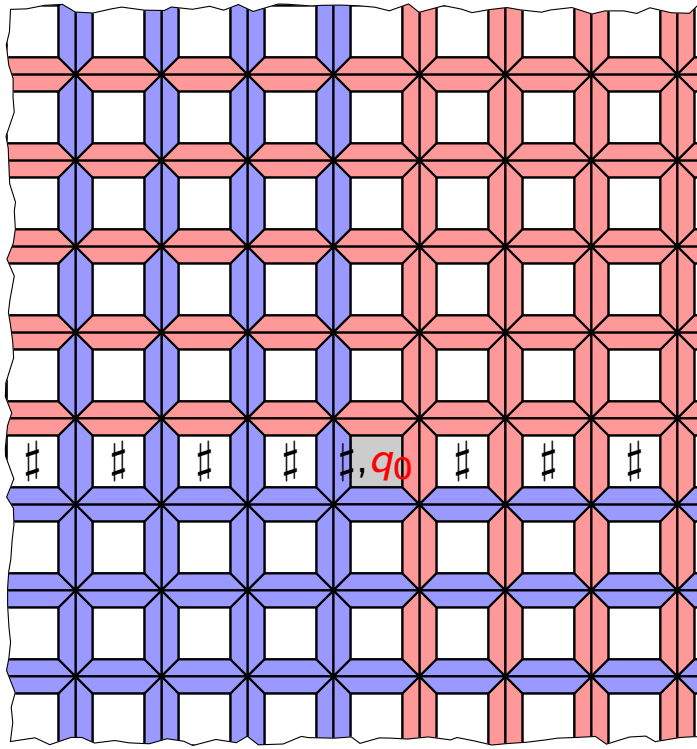
Problème de complétion

Problème de complétion

Etant donné un jeu de tuile et un motif p, est-il possible de paver avec le jeu de tuile partant du motif p ?

Considérons

$$\mathbf{T}(\mathcal{F}_{\leq 1} \cup \mathcal{F}_{\mathtt{Ini}}) \subset \mathcal{A}_{\leq 1} \times \mathcal{A}_{\mathcal{M}}$$



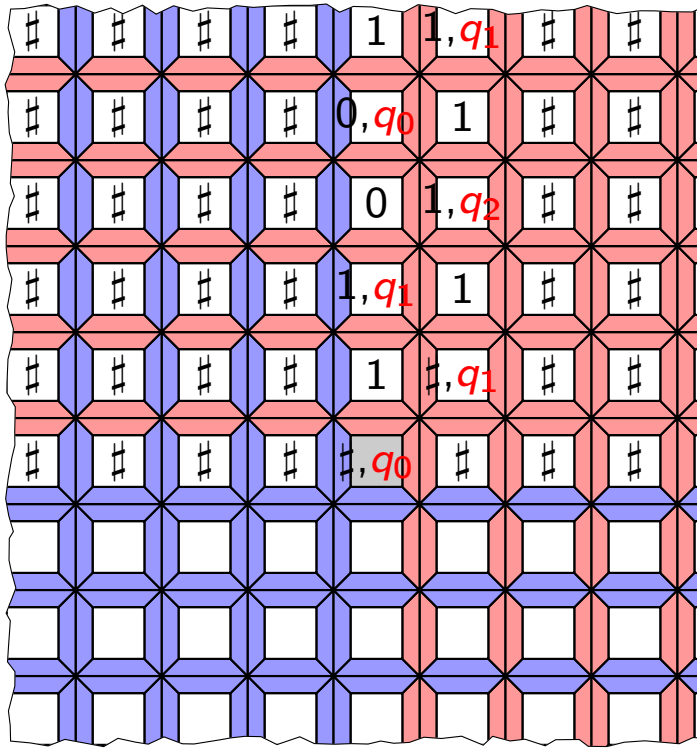
Problème de complétion

Problème de complétion

Etant donné un jeu de tuile et un motif p, est-il possible de paver avec le jeu de tuile partant du motif p ?

Considérons

$$\mathbf{T}(\mathcal{F}_{\leq 1} \cup \mathcal{F}_{\mathtt{Ini}} \cup \mathcal{F}_{\mathcal{M}}) \subset \mathcal{A}_{\leq 1} \times \mathcal{A}_{\mathcal{M}}$$



Problème de complétion

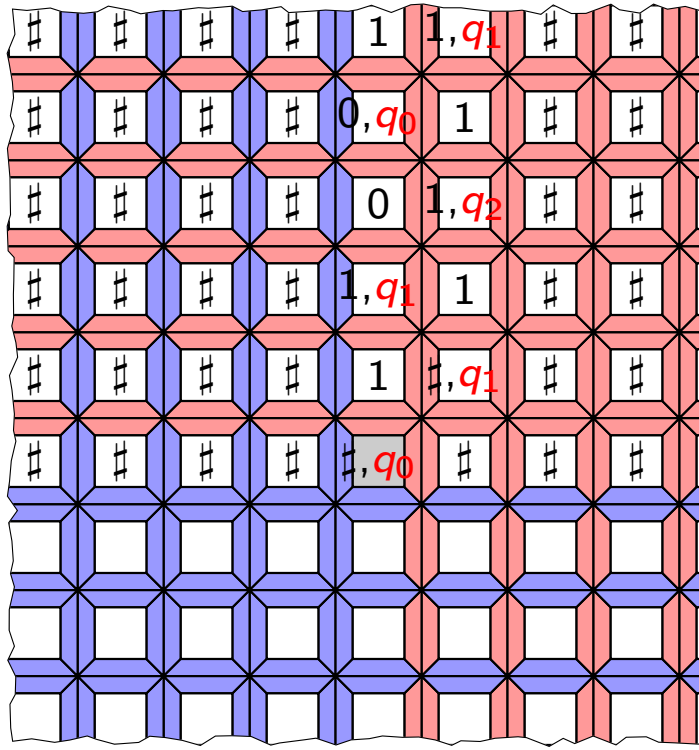
Problème de complétion

Etant donné un jeu de tuile et un motif p, est-il possible de paver avec le jeu de tuile partant du motif p ?

Considérons

$$\mathbf{T}(\mathcal{F}_{\leq 1} \cup \mathcal{F}_{\texttt{Ini}} \cup \mathcal{F}_{\mathcal{M}} \cup \{q_F\}) \subset \mathcal{A}_{\leq 1} \times \mathcal{A}_{\mathcal{M}}$$

peut être complété $\iff$ $\mathcal{M}$ ne s'arrête pas



Problème de complétion

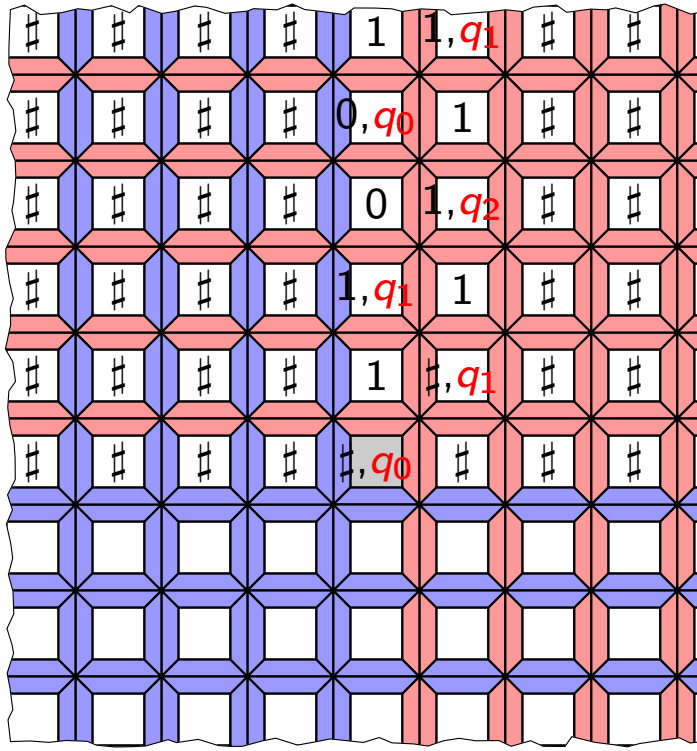
Problème de complétion

Etant donné un jeu de tuile et un motif p, est-il possible de paver avec le jeu de tuile partant du motif p ?

Considérons

$$\mathbf{T}(\mathcal{F}_{\leq 1} \cup \mathcal{F}_{\texttt{Ini}} \cup \mathcal{F}_{\mathcal{M}} \cup \{q_F\}) \subset \mathcal{A}_{\leq 1} \times \mathcal{A}_{\mathcal{M}}$$

peut être complété $\iff$ $\mathcal{M}$ ne s'arrête pas



Théorèm *(Wang 1961)*

Le problème de complétion est indécidable.

Quelques problèmes indécidables

Les problèmes suivants sont indécidables :

- `POST` : Étant donné n paires de mots $(u_1, v_1), \ldots, (u_n, v_n)$, existe-il une suite finie $i_1, \ldots, i_k$ telle que $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$.
- `DOMINO` : Étant donné un jeu fini de tuiles carrées, avec conditions de compatibilité entre côtés (gauche/droite/haut/bas), déterminer si on peut paver le 1/4 de plan.
- `GRAM` : Étant donnée une grammaire hors-contexte, déterminer si elle est ambiguë.
- `MATRICE` : Étant donné un nombre fini de matrices 3×3 à coefficients entiers, déterminer si un produit permet d'annuler la composante $(3, 2)$.
- `CONV` : Étant donnée une suite calculable d'entiers, déterminer si elle converge.
- `Hilbert` : Étant donné un polynôme à coefficients entiers, déterminer s'il a des racines entières (10ème problème de Hilbert).

Importance du problème de l'arrêt

- Il faut prouver qu'un algorithme s'arrête et c'est difficile même pour un programme de 10 lignes !

Importance du problème de l'arrêt

- Il faut prouver qu'un algorithme s'arrête et c'est difficile même pour un programme de 10 lignes !
- Difficulté d'écrire des programmes corrects !

Théorème de Rice

Toutes propriétés non triviale sur un programme est indécidable.

Importance du problème de l'arrêt

- Il faut prouver qu'un algorithme s'arrête et c'est difficile même pour un programme de 10 lignes !
- Difficulté d'écrire des programmes corrects !

Théorème de Rice

Toutes propriétés non triviale sur un programme est indécidable.

- L'indécidabilité logique en mathématiques (K. Gödel) : il existe des énoncés indécidables.

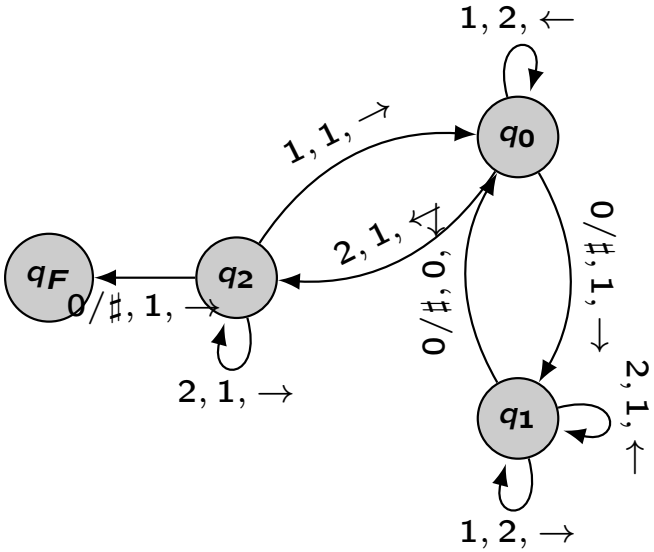
Importance du problème de l'arrêt

- Il faut prouver qu'un algorithme s'arrête et c'est difficile même pour un programme de 10 lignes !
- Difficulté d'écrire des programmes corrects !

Théorème de Rice

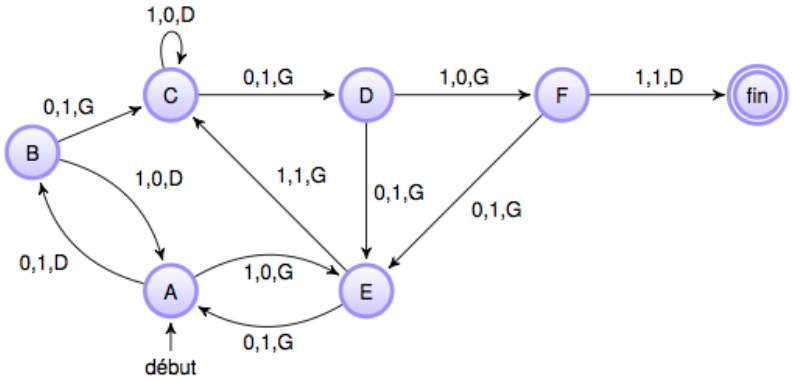
Toutes propriétés non triviale sur un programme est indécidable.

- L'indécidabilité logique en mathématiques (K. Gödel) : il existe des énoncés indécidables.
- Il existe des monstres combinatoires



Partant d'une entrée vide, cette machine de Turing écrit 374×10^6 lettres en 119×10^{15} étapes de calcul.

Monstre combinatoire



Partant d'une entrée vide, cette machine de Turing s'arrête après 10^{2879} étapes de calcul.

Ce nombre est physiquement hors d'atteinte

A âge estimé de l'univers $\simeq 10^{18}$ secondes

T temps de Planck $\simeq 10^{-44}$

N nombre de particules dans l'univers observable $\simeq 10^{90}$

$$\frac{NA}{T} \simeq 10^{152}$$

Classes de complexité

Pourquoi s'intéresser à la complexité ?

Objectifs :

- Prédiction du temps d'exécution
- Comparaison entre algorithmes

Méthodes :

- Expérimentation : lancer le(s) algorithme(s) sur des entrées
- Analyse de complexité théorique

Exemples :

- De combien de temps ai-je besoin pour calculer a^n ou $1000!$?
- Recherche dichotomique ou recherche séquentielle ?

Limites

Complexité expérimentale :

- Dépend des langages, machines, systèmes d'expoitation, . . .
- Et donc varie au cours du temps !
- Mais utile et utilisée en pratique

Limites

Complexité expérimentale :

- Dépend des langages, machines, systèmes d'expoitation, . . .
- Et donc varie au cours du temps !
- Mais utile et utilisée en pratique

4 idées pour la complexité théorique :

1. Compter le nombre d'opérations élémentaires...
2. ...en fonction des entrées...
3. ...de manière asymptotique...
4. ...dans le pire cas.

Limites

Complexité expérimentale :

- Dépend des langages, machines, systèmes d'expoitation, . . .
- Et donc varie au cours du temps !
- Mais utile et utilisée en pratique

4 idées pour la complexité théorique :

1. Compter le nombre d'opérations élémentaires...
2. ...en fonction des entrées...
3. ...de manière asymptotique...
4. ...dans le pire cas.

Discutions :

1. Autres mesures : temps parallèle, espace mémoire, etc.
2. Taille en pratique des entrées
3. Constantes cachées, comportement pratique
4. Analyse en moyenne, analyse lissée...

Temps et espace de calcul

Lorsque le programme s'arrête sur une entrée donnée, on va s'intéresser à quantifier le temps et l'espace qu'a nécessité le calcul.

Définition : Temps et espace de calcul

Soit $\mathcal{M}$ une machine de Turing et x une entrée. Si $M(x)$ s'arrête alors :

- le temps de calcule de $\mathcal{M}(x)$ est le nombre d'étapes effectuées par le calcul $\mathcal{M}(x)$;
- l'espace utilisé par $\mathcal{M}(x)$ est le nombre total de cases différentes visitées sur les rubans de travail au cours du calcul.

Attention : Le temps et l'espace de calcul dépendent du modèle de calcul et du codage de l'entrée.

Thèse de Church-Turing et complexité

- On peut transformer une machine de Turing $\mathcal{M}$ travaillant sur l'alphabet $\mathcal{A}$ en une machine $\mathcal{M}'$ sur l'alphabet $\{0,1\}$ telle que :
  - $\mathcal{M}$ accepte $x \in \mathcal{A}^*$ si et seulement si $\mathcal{M}'$ accepte $\varphi(x) \in \mathcal{B}^*$;
  - $\mathcal{M}(x)$ s'arrête en temps t alors $\mathcal{M}'(\varphi(x))$ s'arrête en temps au plus $6t \log(|\mathcal{A}|)$;
  - $\mathcal{M}(x)$ utilise un espace s alors $\mathcal{M}'(\varphi(x))$ utilise un espace au plus $2s \log(|\mathcal{A}|)$.
- On peut transformer une machine de Turing $\mathcal{M}$ à k rubans en une machine $\mathcal{M}'$ à un telle que :
  - $\mathcal{M}$ accepte $x \in \mathcal{A}^*$ si et seulement si $\mathcal{M}'$ accepte $\varphi(x) \in \mathcal{B}^*$;
  - $\mathcal{M}(x)$ s'arrête en temps t alors $\mathcal{M}'(\varphi(x))$ s'arrête en temps au plus $C\, t^2$;
  - $\mathcal{M}(x)$ utilise un espace s alors $\mathcal{M}'(\varphi(x))$ utilise un espace au plus $k\, s$.
- ...

Machine de Turing universelle

Théorème : Machine de Turing universelle

Il existe une machine de Turing universelle $\mathcal{U}$ à 5 rubans sur l'alphabet $\{0,1\}$ sur qui prend en argument le code $\langle \mathcal{M} \rangle$ de la machine de Turing $\mathcal{M}$ et le code $\langle x \rangle$ d'une entrée x de $\mathcal{M}$ en binaire telle que :

- $\mathcal{M}(x)$ s'arrête si et seulement si $\mathcal{U}(\langle \mathcal{M} \rangle, \langle x \rangle)$ s'arrête ;
- $\mathcal{U}(\langle \mathcal{M} \rangle, \langle x \rangle)$ est un codage de $\mathcal{M}(x)$;
- si $\mathcal{M}(x)$ s'arrête en temps t et espace s alors $\mathcal{U}(\langle \mathcal{M} \rangle, \langle x \rangle)$ s'arrête en temps inférieur à $\alpha_{\mathcal{M}}(1 + st)$ et utilise un espace inférieur à $\alpha_{\mathcal{M}}(s + 1)$.

$\alpha_{\mathcal{M}}$ est une constante dépendant de la machine $\mathcal{M}$ simulée.

Machine de Turing universelle

Théorème : Machine de Turing universelle

Il existe une machine de Turing universelle $\mathcal{U}$ à 5 rubans sur l'alphabet $\{0,1\}$ sur qui prend en argument le code $\langle \mathcal{M} \rangle$ de la machine de Turing $\mathcal{M}$ et le code $\langle x \rangle$ d'une entrée x de $\mathcal{M}$ en binaire telle que :

- $\mathcal{M}(x)$ s'arrête si et seulement si $\mathcal{U}(\langle \mathcal{M} \rangle, \langle x \rangle)$ s'arrête ;
- $\mathcal{U}(\langle \mathcal{M} \rangle, \langle x \rangle)$ est un codage de $\mathcal{M}(x)$;
- si $\mathcal{M}(x)$ s'arrête en temps t et espace s alors $\mathcal{U}(\langle \mathcal{M} \rangle, \langle x \rangle)$ s'arrête en temps inférieur à $\alpha_{\mathcal{M}}(1 + t \log(t))$ et utilise un espace inférieur à $\alpha_{\mathcal{M}}(s + \log(t))$.

$\alpha_{\mathcal{M}}$ est une constante dépendant de la machine $\mathcal{M}$ simulée.

Classes de complexité en temps déterministe

Définition (Classes de complexité en temps déterministe)

Pour une fonction $t : \mathbb{N} \to \mathbb{N}$, la classe $\mathrm{Dtime}(t(n))$ est l'ensemble des langages tels que $\mathcal{L} \in \mathrm{Dtime}(t(n))$ s'il existe une machine de Turing $\mathcal{M}$ et une constante α telles que

$$x \in \mathcal{L} \iff x \text{ est accepté par } \mathcal{M} \text{ en temps inférieur à } \alpha\, t(|x|)$$

Remarques :

- Une classe de complexité dépend de la taille de l'entrée.
- On considère ici que les machines s'arrêterons sur toutes entrées.
- Pour résoudre la plupart des problèmes no triviaux, il faut lire la totalité de l'entrée, donc les classes $\mathrm{Dtime}(t(n))$ où $t(n) = o(n)$ ont peu d'intérêt.

Premières propriétés

Inclusion

Si pour $n \in \mathbb{N}$ on a $f(n) \leq g(n)$ alors $\mathrm{Dtime}(f(n)) \subset \mathrm{Dtime}(g(n))$

Clôture par opérations usuelles

Soient $\mathcal{L}_1, \mathcal{L}_2 \in \mathrm{Dtime}(f(n))$ alors

- $\mathcal{L}_1 \cap \mathcal{L}_2 \in \mathrm{Dtime}(f(n))$
- $\mathcal{L}_1 \cup \mathcal{L}_2 \in \mathrm{Dtime}(f(n))$
- $\overline{\mathcal{L}_1} \in \mathrm{Dtime}(f(n))$

Théorème de la hiérarchie

Définitons : Fonctions constructibles en temps

Une fonction $t : \mathbb{N} \to \mathbb{N}$ est **constructible en temps** s'il existe une constante α et une machine de Turing $\mathcal{M}$ qui, sur l'entrée 1^n (l'entier n en unaire) renvoie $1^{t(n)}$ (l'entier $t(n)$ en unaire) en temps inférieur à $\alpha\, t(n)$.

Exemples : fonctions polynômes, $t(n) = 2^{n^k}$

Théorème de la hiérarchie

Soit $f : \mathbb{N} \to \mathbb{N}$ et $g : \mathbb{N} \to \mathbb{N}$ des fonctions vérifiant :

- $f(n) \neq 0$ pour tout $n \in \mathbb{N}$;
- g est constructible en temps ;
- $f(n)\log(f(n)) = o(g(n))$.

Il existe un langage $\mathcal{L}$ tel que $\mathcal{L} \in \mathrm{Dtime}\,(g(n))$ et $\mathcal{L} \notin \mathrm{Dtime}\,(f(n))$.

Deux classes de complexité importantes

Définitons : Classes $\mathbf{P}$ et $\mathbf{Exp}$

$$\mathbf{P} = \bigcup_{k \in \mathbb{N}} \mathrm{Dtime}\left(n^k\right)$$

$$\mathbf{Exp} = \bigcup_{k \in \mathbb{N}} \mathrm{Dtime}\left(2^{n^k}\right)$$

Remarques :

Ces classes de complexité sont stable si on change le modèle de calcul.

Deux classes de complexité importantes

Définitons : Classes $\mathbf{P}$ et $\mathbf{Exp}$

$$\mathbf{P} = \bigcup_{k \in \mathbb{N}} \mathrm{Dtime}\left(n^k\right)$$

$$\mathbf{Exp} = \bigcup_{k \in \mathbb{N}} \mathrm{Dtime}\left(2^{n^k}\right)$$

Remarques :

Ces classes de complexité sont stable si on change le modèle de calcul.

Corollaire

- $\mathbf{P} \subset \mathbf{Exp}$
- $\mathbf{P} \neq \mathbf{Exp}$

Exemples

Multiplication d'entiers (`MultEnt`)

Entrée : Deux entiers a, b coder en binaire, un entier k

Question : Le $k^{\text{ème}}$ bit de $a\,b$ vaut-il 1 ?

Exemples

Multiplication d'entiers (`MultEnt`)

Entrée : Deux entiers a, b coder en binaire, un entier k

Question : Le $k^{\text{ème}}$ bit de $a\,b$ vaut-il 1 ?

`MultEnt` $\in \mathbf{P}$: il suffit d'effectuer la multiplication $a\,b$ grâce à l'algorithme de l'école primaire et regarder le $k^{\text{ème}}$ bit du résultat.

Exemples

Multiplication d'entiers (`MultEnt`)

Entrée : Deux entiers a, b coder en binaire, un entier k

Question : Le $k^{\text{ème}}$ bit de $a\,b$ vaut-il 1 ?

`MultEnt` $\in \mathbf{P}$: il suffit d'effectuer la multiplication $a\,b$ grâce à l'algorithme de l'école primaire et regarder le $k^{\text{ème}}$ bit du résultat.

Multiplication de matrices (`MultMat`)

Entrée : Deux matrices A et B de taille $m \times m$ à coefficients entiers codés en binaire, un couple $(i, j) \in [1, m]^2$, un entier k.

Question : Le $k^{\text{ème}}$ bit du coefficient (i, j) du produit AB vaut-il 1 ?

Exemples

Multiplication d'entiers (`MultEnt`)

Entrée : Deux entiers a, b coder en binaire, un entier k

Question : Le $k^{\text{ème}}$ bit de $a\,b$ vaut-il 1 ?

`MultEnt` $\in \mathbf{P}$: il suffit d'effectuer la multiplication $a\,b$ grâce à l'algorithme de l'école primaire et regarder le $k^{\text{ème}}$ bit du résultat.

Multiplication de matrices (`MultMat`)

Entrée : Deux matrices A et B de taille $m \times m$ à coefficients entiers codés en binaire, un couple $(i, j) \in [1, m]^2$, un entier k.

Question : Le $k^{\text{ème}}$ bit du coefficient (i, j) du produit AB vaut-il 1 ?

`MultMat` $\in \mathbf{P}$: le coefficient (i, j) de AB n'est rien d'autre que la somme des m produits $a_{i,k}\,b_{k,j}$.

Exemples

Accessibilité (Acc)

Entrée : Un graphe orienté $G = (V, E)$ et deux sommets s et t

Question : Existe-t-il un chemin reliant s à t ?

Exemples

Accessibilité (Acc)

Entrée : Un graphe orienté $G = (V, E)$ et deux sommets s et t

Question : Existe-t-il un chemin reliant s à t ?

$\texttt{Acc} \in \mathbf{P}$: il suffit de faire un parcours de graphe en partant de s et de voir que l'on atteint t.

Exemples

Accessibilité (Acc)

Entrée : Un graphe orienté $G = (V, E)$ et deux sommets s et t

Question : Existe-t-il un chemin reliant s à t ?

$\texttt{Acc} \in \mathbf{P}$: il suffit de faire un parcours de graphe en partant de s et de voir que l'on atteint t.

Couplage parfait (Couplage)

Entrée : Un graphe non orienté $G = (V, E)$

Question : G admet un couplage parfait ?

Exemples

Accessibilité (Acc)

Entrée : Un graphe orienté $G = (V, E)$ et deux sommets s et t

Question : Existe-t-il un chemin reliant s à t ?

$\texttt{Acc} \in \mathbf{P}$: il suffit de faire un parcours de graphe en partant de s et de voir que l'on atteint t.

Couplage parfait (Couplage)

Entrée : Un graphe non orienté $G = (V, E)$

Question : G admet un couplage parfait ?

Un couplage parfait est un ensemble d'arêtes tel que chaque sommet est relié à exactement une arrête de cet ensemble. Ce problème est dans $\mathbf{P}$ mais l'algorithme n'est pas trivial.

Exemples

Primalité (`Prime`)

Entrée : Un entier N

Question : N est il premier ?

Exemples

Primalité (`Prime`)

Entrée : Un entier N

Question : N est il premier ?

L'algorithme classique du crible consiste à essayer tous les diviseurs potentiels jusqu'à $\sqrt{N}$.

Exemples

Primalité (`Prime`)

Entrée : Un entier N

Question : N est il premier ?

L'algorithme classique du crible consiste à essayer tous les diviseurs potentiels jusqu'à $\sqrt{N}$.

- Si N est coder en binaire cet algorithme montre que $\texttt{Prime} \in \mathbf{Exp}$ puisque la taille de l'entrée est $n = \log(n)$.

Exemples

Primalité (`Prime`)

Entrée : Un entier N

Question : N est il premier ?

L'algorithme classique du crible consiste à essayer tous les diviseurs potentiels jusqu'à $\sqrt{N}$.

- Si N est coder en binaire cet algorithme montre que $\texttt{Prime} \in \mathbf{Exp}$ puisque la taille de l'entrée est $n = \log(n)$.
- Si N est coder en unaire on montre que $\texttt{Prime} \in \mathbf{P}$: on transforme l'entrée en binaire et on utilise l'algorithme précédent.

Exemples

Primalité (`Prime`)

Entrée : Un entier N

Question : N est il premier ?

L'algorithme classique du crible consiste à essayer tous les diviseurs potentiels jusqu'à $\sqrt{N}$.

- Si N est coder en binaire cet algorithme montre que $\texttt{Prime} \in \mathbf{Exp}$ puisque la taille de l'entrée est $n = \log(n)$.
- Si N est coder en unaire on montre que $\texttt{Prime} \in \mathbf{P}$: on transforme l'entrée en binaire et on utilise l'algorithme précédent.

En 2004, Agrawal, Kayal et Saxena montrent que $\texttt{Prime} \in \mathbf{P}$

Exemples

Rangement (Rangement)

Entrée : Des entiers $p_1, \ldots, p_n \in \mathbb{N}$ en binaire et deux entiers m, p.

Question : Peut-on ranger les n objets dans les m boites en respectant le poids maximal p de chaque boîte ?

$\texttt{Rangement} \in \mathbf{Exp}$

Exemples

Rangement (Rangement)

Entrée : Des entiers $p_1, \ldots, p_n \in \mathbb{N}$ en binaire et deux entiers m, p.

Question : Peut-on ranger les n objets dans les m boites en respectant le poids maximal p de chaque boîte ?

$\texttt{Rangement} \in \mathbf{Exp}$ car il suffit d'essayer toutes les possibilités pour ranger les objets et vérifier si la condition est vérifiée. Il y a $m^n = 2^{n \log(m)}$ possibilités.

Exemples

Rangement (Rangement)

Entrée : Des entiers $p_1, \ldots, p_n \in \mathbb{N}$ en binaire et deux entiers m, p.

Question : Peut-on ranger les n objets dans les m boites en respectant le poids maximal p de chaque boîte ?

$\texttt{Rangement} \in \mathbf{Exp}$ car il suffit d'essayer toutes les possibilités pour ranger les objets et vérifier si la condition est vérifiée. Il y a $m^n = 2^{n \log(m)}$ possibilités.

Actuellement on ne sait pas si $\texttt{Rangement} \in \mathbf{P}$ mais on verra que $\texttt{Rangement} \in \mathbf{NP}$ où $\mathbf{NP}$ est une classe intermédiaire entre $\mathbf{P}$ et $\mathbf{Exp}$.

Exemples

Arrêt d'une machine en temps exponentiel (`Arrêt Exp`)

Entrée : Le code d'une machine $\mathcal{M}$ et un entier k en binaire.

Question : Est-ce que $\mathcal{M}$ sur l'entrée vide s'arrête en temps au plus k ?

`Arrêt Exp` $\in \mathbf{Exp}$ car on peut simuler $\mathcal{M}$ sur l'entrée vide pendant k étapes et la taille de l'entrée k est $\log(k)$, cela prend donc un temps exponentiel.

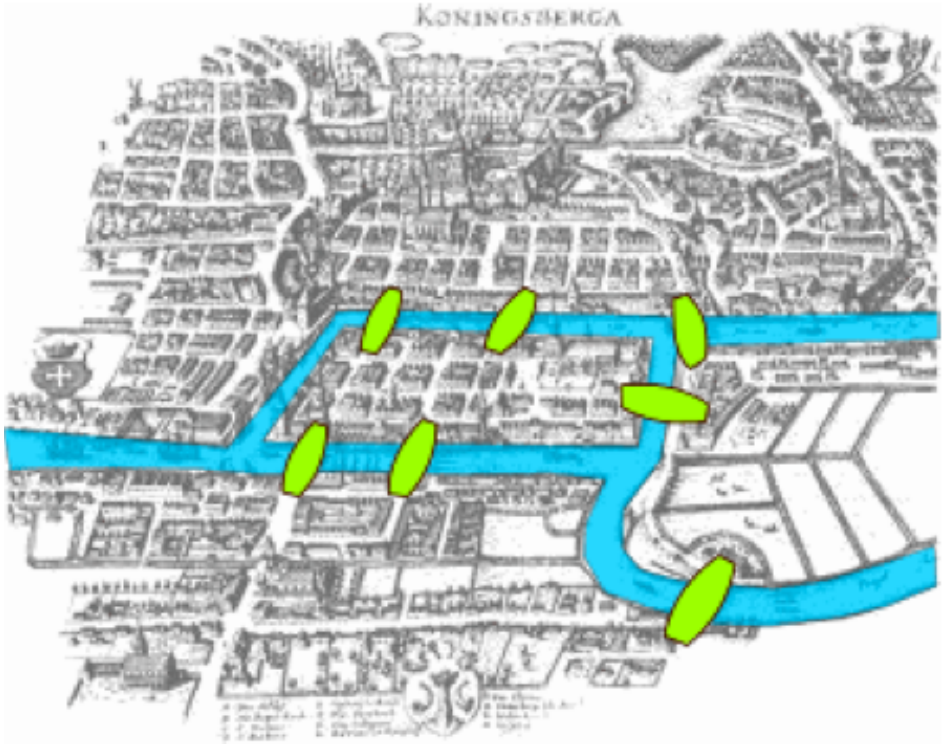
Remarque : Si k est donné en unaire le problème devient polynomial.

Exemples

Chemin Eulerien (`Chemin eulerien`)

Entrée : Un graphe $G = (V, E)$.

Question : Existe t-il un chemin qui passe une fois par chaque arête ?



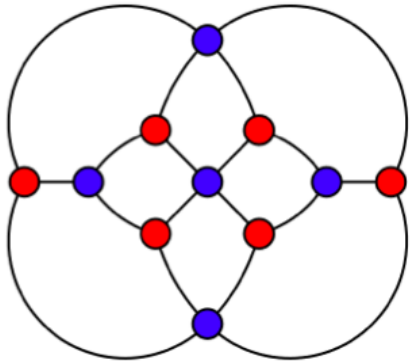
Exemples

Chemin Hamiltonien (`Chemin Hamiltonien`)

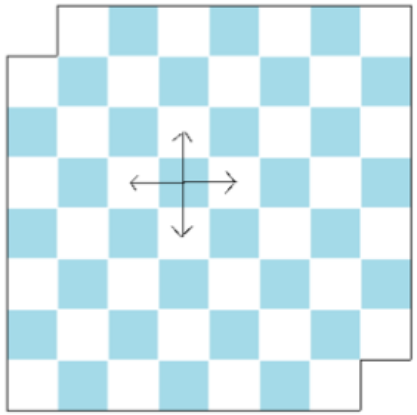
Entrée : Un graphe $G = (V, E)$.

Question : Existe t-il un chemin qui passe une fois par chaque sommet ?

Problème du voyageur de commerce.



Peut on trouver un cycle Hamiltonien ?



La tour peut elle passer une fois par chaque case ?

Exemples

Coloriage avec 2 couleurs (2-colors)

Entrée : Un graphe $G = (V, E)$.

Question : Est-il possible de le colorier avec 2 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Exemples

Coloriage avec 2 couleurs (2-colors)

Entrée : Un graphe $G = (V, E)$.

Question : Est-il possible de le colorier avec 2 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Coloriage avec 3 couleurs (3-colors)

Entrée : Un graphe $G = (V, E)$.

Question : Est-il possible de le colorier avec 3 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Exemples

Coloriage avec 2 couleurs (2-colors)

Entrée : Un graphe $G = (V, E)$.

Question : Est-il possible de le colorier avec 2 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Coloriage avec 3 couleurs (3-colors)

Entrée : Un graphe $G = (V, E)$.

Question : Est-il possible de le colorier avec 3 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Coloriage avec 4 couleurs (4-ColorPlanaire)

Entrée : Un graphe $G = (V, E)$ planaire.

Question : Est-il possible de le colorier avec 4 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Exemples

Coloriage avec 2 couleurs (2-colors)

Entrée : Un graphe $G = (V, E)$.

Question : Est-il possible de le colorier avec 2 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Coloriage avec 3 couleurs (3-colors)

Entrée : Un graphe $G = (V, E)$.

Question : Est-il possible de le colorier avec 3 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Coloriage avec 4 couleurs (4-ColorPlanaire)

Entrée : Un graphe $G = (V, E)$ planaire.

Question : Est-il possible de le colorier avec 4 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Coloriage de cartes avec 3 couleurs (3-ColorPlanaire)

Entrée : Un graphe $G = (V, E)$ planaire.

Question : Est-il possible de le colorier avec 3 couleurs ?

Classes de complexité en temps non déterministe

Machine de Turing non déterministe

- On définit une machine de Turing non déterministe comme $\mathcal{N} = (Q, q_0, q_a, q_r, \mathcal{A}, \#, \Delta)$ où Δ n'est plus une fonction de transition mais un ensemble de transitions possibles $\Delta \subset (Q \times \mathcal{A}) \times (Q \times \mathcal{A} \times \{\leftarrow, \rightarrow\})$.
- Autrement dit, à chaque itération la machine de Turing a le choix entre plusieurs itérations.
- On peut le voir comme une instruction supplémentaires qui consiste à deviner des bits.

Autres définitions pour $\mathbf{NP}$ et $\mathbf{NExp}$

Caractérisation existentielle de $\mathbf{NP}$

Un langage $\mathcal{L}$ est dans $\mathbf{NP}$ si et seulement si il existe un polynôme p et un langage $\mathcal{L}' \in \mathbf{P}$ tels que

$$x \in \mathcal{L} \iff \exists y \in \{0,1\}^{p(|x|)} \ (x,y) \in \mathcal{L}'$$

Caractérisation existentielle de $\mathbf{NExp}$

Un langage $\mathcal{L}$ est dans $\mathbf{NExp}$ si et seulement si il existe un polynôme p et un langage $\mathcal{L}' \in \mathbf{P}$ tels que

$$x \in \mathcal{L} \iff \exists y \in \{0,1\}^{2^{p(|x|)}} \ (x,y) \in \mathcal{L}'$$

Exemples de problème reconnu de manière non déterministe

Clique (`Clique`)

Entrée : Un graphe orienté $G = (V, E)$ et un entier k codé en binaire

Question : Existe-t-il une clique de taille k ?

Indépendant (`Indépend`)

Entrée : Un graphe orienté $G = (V, E)$ et un entier k codé en binaire

Question : Existe-t-il k sommet indépendants ?

Exemples de problème reconnu de manière non déterministe

Classes de complexité en temps non déterministe

Définition (Classes de complexité en temps non déterministe)

$\mathrm{Ntime}(t(n))$ est la classe des langages tels que $\mathcal{L} \in \mathrm{Ntime}(t(n))$ s'il existe une machine de Turing non déterministe $\mathcal{N}$ et une constante α telles que

$$x \in \mathcal{L} \Longleftrightarrow x \text{ est accepté par une exécution de } \mathcal{N} \text{ en temps } \leq \alpha\, t(|x|)$$

Inclusion

Si pour $n \in \mathbb{N}$ on a $f(n) \leq g(n)$ alors $\mathrm{Ntime}(f(n)) \subset \mathrm{Ntime}(g(n))$

Clôture par opérations usuelles

Soient $\mathcal{L}_1, \mathcal{L}_2 \in \mathrm{Ntime}(f(n))$ alors

- $\mathcal{L}_1 \cap \mathcal{L}_2 \in \mathrm{Ntime}(f(n))$
- $\mathcal{L}_1 \cup \mathcal{L}_2 \in \mathrm{Ntime}(f(n))$

Non déterminisme vs déterminisme

$$\mathrm{Dtime}(t(n)) \subset \mathrm{Ntime}(t(n)) \subset \mathrm{Dtime}\left(2^{C\, t(n)}\right)$$

Théorème de la hiérarchie

Théorème de la hiérarchie

Soit $f : \mathbb{N} \to \mathbb{N}$ et $g : \mathbb{N} \to \mathbb{N}$ des fonctions vérifiant :

- $f(n) \neq 0$ pour tout $n \in \mathbb{N}$;
- g est constructible en temps et croissante ;
- $f(n+1) = o(g(n))$.

Il existe un langage $\mathcal{L}$ tel que $\mathcal{L} \in \mathrm{Ntime}\,(g(n))$ et $\mathcal{L} \notin \mathrm{Ntime}\,(f(n))$.

Deux classes de complexité importantes

Définitons : Classes **NP** et **NExp**

$$\mathbf{NP} = \bigcup_{k \in \mathbb{N}} \mathrm{Ntime}\left(n^k\right)$$

$$\mathbf{NExp} = \bigcup_{k \in \mathbb{N}} \mathrm{Ntime}\left(2^{n^k}\right)$$

Deux classes de complexité importantes

Définitons : Classes $\mathbf{NP}$ et $\mathbf{NExp}$

$$\mathbf{NP} = \bigcup_{k \in \mathbb{N}} \mathrm{Ntime}\left(n^k\right)$$
$$\mathbf{NExp} = \bigcup_{k \in \mathbb{N}} \mathrm{Ntime}\left(2^{n^k}\right)$$

Inclusions importantes

- $\mathbf{P} \subset \mathbf{NP} \subset \mathbf{Exp} \subset \mathbf{NExp}$
- $\mathbf{P} \neq \mathbf{Exp}$ et $\mathbf{NP} \neq \mathbf{NExp}$ (obtenus avec les théorèmes de hiérarchie déterministe et non déterministe)

Deux classes de complexité importantes

Définitons : Classes $\mathbf{NP}$ et $\mathbf{NExp}$

$$\mathbf{NP} = \bigcup_{k \in \mathbb{N}} \mathrm{Ntime}\left(n^k\right)$$

$$\mathbf{NExp} = \bigcup_{k \in \mathbb{N}} \mathrm{Ntime}\left(2^{n^k}\right)$$

Inclusions importantes

- $\mathbf{P} \subset \mathbf{NP} \subset \mathbf{Exp} \subset \mathbf{NExp}$
- $\mathbf{P} \neq \mathbf{Exp}$ et $\mathbf{NP} \neq \mathbf{NExp}$ (obtenus avec les théorèmes de hiérarchie déterministe et non déterministe)

Savoir si $\mathbf{P} = \mathbf{NP}$ est le grand problème de l'informatique théorique !

De même on ne sait pas si $\mathbf{Exp} = \mathbf{NExp}$

Complexité du complémentaire

Définition

Soit $\mathcal{C}$ une classe de complexité. La classe $co\mathcal{C}$ est l'ensemble des complémentaires des langages de $\mathcal{C}$. Autrement dit $co\mathcal{C} = \{\mathcal{L} : \overline{\mathcal{L}} \in \mathcal{C}\}$

Différentes définitions de co-NP

Les propositions suivantes sont équivalentes :

- $\mathcal{L} \in \mathbf{co\text{-}NP}$
- $\mathcal{A}^* \setminus \mathcal{L} \in \mathbf{NP}$
- Il existe $\mathcal{N}$ une machine de Turing non déterministe travaillant en temps polynomial telle que $x \in \mathcal{L}$ ssi tous les chemins de calcul de $\mathcal{N}(x)$ sont acceptant.
- il existe un polynôme p et un langage $\mathcal{L}' \in \mathbf{P}$ tels que

$$x \in \mathcal{L} \iff \forall y \in \{0,1\}^{p(|x|)} \ (x, y) \in \mathcal{L}'$$

Il est ouvert de savoir si $\mathbf{NP} = \mathbf{co\text{-}NP}$.

On a $\mathbf{P} = \mathbf{NP} \Longrightarrow \mathbf{NP} = \mathbf{co\text{-}NP}$

Exemple de problème dans co-NP

Primalité (`Prime`)

Entrée : un entier N

Question : N est il premier ?

On a vu que $\overline{\texttt{Prime}} \in \mathbf{NP}$ donc $\texttt{Prime} \in co\mathbf{NP}$.
On peut aussi le prouver directement en utilisant l'algorithme suivant :

- Entrée n en binaire.
- deviner un entier $p \leq n$
- Si $n\%p \neq 0$ alors accepté n.

Cet algorithme fonctionne en temps polynomial et $n \in \texttt{Prime}$ si toute exécution de cet algorithme est acceptante.
On verra en TD que l'on a aussi $\texttt{Prime} \in \mathbf{NP}$.

Problèmes $\mathbf{NP}$-complets

Notion de réduction

Réduction

Soient $\mathcal{L} \subset \mathcal{A}^*$ et $\mathcal{L}' \subset \mathcal{B}^*$ deux langages. $\mathcal{L}$ se réduit à $\mathcal{L}'$, noté $\mathcal{L} \leq_m^P \mathcal{L}'$, s'il existe une fonction $f : \mathcal{A}^* \to \mathcal{B}^*$ de complexité polynomiale telle que

$$x \in \mathcal{L} \iff f(x) \in \mathcal{L}'$$

Proposition

- $\leq_m^P$ est un préordre sur l'ensemble des langages
- Si $\mathcal{L} \leq_m^P \mathcal{L}'$ alors
  - $\mathcal{L}' \in \mathbf{P} \Longrightarrow \mathcal{L} \in \mathbf{P}$.
  - $\mathcal{L}' \in \mathbf{NP} \Longrightarrow \mathcal{L} \in \mathbf{NP}$.
  - $\mathcal{L}' \in \mathbf{Exp} \Longrightarrow \mathcal{L} \in \mathbf{Exp}$.

Application :

$$\texttt{Clique} \leq_m^P \texttt{Indépendant} \quad \text{et} \quad \texttt{Indépendant} \leq_m^P \texttt{Clique}$$

$$\texttt{k-Color} \leq_m^P \texttt{k+1-Color}$$

Problèmes $\mathbf{NP}$-complets

Définition

Un langage $\mathcal{L}$ est $\mathbf{NP}$-complet si :

- $\mathcal{L} \in \mathbf{NP}$;
- $\mathcal{L}$ est $\mathbf{NP}$-difficile c'est à dire que pour tout $\mathcal{L}' \in \mathbf{NP}$ on a

$$\mathcal{L}' \leq_m^P \mathcal{L}$$

Problèmes $\mathbf{NP}$-complets

Définition

Un langage $\mathcal{L}$ est $\mathbf{NP}$-complet si :

- $\mathcal{L} \in \mathbf{NP}$;
- $\mathcal{L}$ est $\mathbf{NP}$-difficile c'est à dire que pour tout $\mathcal{L}' \in \mathbf{NP}$ on a

$$\mathcal{L}' \leq_m^P \mathcal{L}$$

Proposition

Si on résout un problème $\mathbf{NP}$-complet en temps polynomial alors

$$\mathbf{P} = \mathbf{NP}$$

Problèmes $\mathbf{NP}$-complets

Définition

Un langage $\mathcal{L}$ est $\mathbf{NP}$-complet si :

- $\mathcal{L} \in \mathbf{NP}$;
- $\mathcal{L}$ est $\mathbf{NP}$-difficile c'est à dire que pour tout $\mathcal{L}' \in \mathbf{NP}$ on a

$$\mathcal{L}' \leq_m^P \mathcal{L}$$

Proposition

Si on résout un problème $\mathbf{NP}$-complet en temps polynomial alors

$$\mathbf{P} = \mathbf{NP}$$

Conséquences :

- Si on doit résoudre un problème $\mathbf{NP}$-complet, il y a peu de chances qu'on trouve un algorithme efficace pour le résoudre !
- Tout n'est pas négatif, on peut utiliser les problèmes $\mathbf{NP}$-complet pour faire de la cryptographie.

Eternity

Paver une région fini

Entrée : Une région fini du plan et un ensemble de tuiles.

Question : Peut on paver la région avec des copies de ces tuiles ?

Eternity

Paver une région fini

Entrée : Une région fini du plan et un ensemble de tuiles.

Question : Peut on paver la région avec des copies de ces tuiles ?

Théorème

Le problème de décision paver une région fini du plan est $\mathbf{NP}$-complet

Littéraux, clauses, et formules CNF

Étant données des variables booléennes $x_1, x_2, \ldots$

- Un **littéral** est soit une variable x_i, soit la négation d'une variable $\neg x_i$.
- Une **clause** est une disjonction de littéraux.
  Exemple : $x1 \vee x_2 \vee \neg x_4 \vee x_5$.
- Une **3-clause** est une clause avec 3 littéraux différents.
  Exemple : $x1 \vee \neg x_2 \vee x_4$
- Une **formule CNF** est une conjonction de clauses.
- Une **formule 3-CNF** est une conjonction de 3-clauses.
  Exemple : $(x1 \vee x_2 \vee x_4) \wedge (x_3 \vee \neg x_1 \vee \neg x_4)$.
- Une formule est **satisfiable** s'il existe une affectation valide.

SAT

SAT

Entrée : Une formule CNF

Question : Cette formule est elle satisfiable ?

3-SAT

Entrée : Une formule 3-CNF

Question : Cette formule est elle satisfiable ?

2-SAT

Entrée : Une formule 2-CNF

Question : Cette formule est elle satisfiable ?

SAT

SAT

Entrée : Une formule CNF

Question : Cette formule est elle satisfiable ?

3-SAT

Entrée : Une formule 3-CNF

Question : Cette formule est elle satisfiable ?

2-SAT

Entrée : Une formule 2-CNF

Question : Cette formule est elle satisfiable ?

Théorème *(Cook-1971,Levin-1973)*

SAT et 3-SAT sont $\mathbf{NP}$-complet.

$2 - \texttt{SAT} \in \mathbf{P}$

Examples de problèmes $\mathbf{NP}$-complet

Clique (`Clique`)

Entrée : Un graphe orienté $G = (V, E)$ et un entier k codé en binaire

Question : Existe-t-il une clique de taille k ?

Indépendant (`Indépend`)

Entrée : Un graphe orienté $G = (V, E)$ et un entier k codé en binaire

Question : Existe-t-il k sommet indépendants ?

Couplage parfait (`Couplage`)

Entrée : Un graphe non orienté $G = (V, E)$

Question : G admet un couplage parfait ?

Rangement (`Rangement`)

Entrée : Des entiers $p_1, \ldots, p_n \in \mathbb{N}$ en binaire et deux entiers m, p.

Question : Peut-on ranger les n objets dans les m boites en respectant le poids maximal p de chaque boîte ?

Examples de problèmes $\mathbf{NP}$-complet

Coloriage avec 3 couleurs (`3-colors`)

Entrée : Un graphe $G = (V, E)$.

Question : Est-il possible de le colorier avec 3 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Coloriage avec 4 couleurs (`4-ColorPlanaire`)

Entrée : Un graphe $G = (V, E)$ planaire.

Question : Est-il possible de le colorier avec 4 couleurs tel que deux sommets voisins soient de couleurs différentes ?

Coloriage de cartes avec 3 couleurs (`3-ColorPlanaire`)

Entrée : Un graphe $G = (V, E)$ planaire.

Question : Est-il possible de le colorier avec 3 couleurs ?

Mais `3-colors` et `4-ColorPlanaire` sont dans $\mathbf{P}$.

Autres classes de complexité