Machines de Turing

Introduction

Modèle mathématique introduit par Alan Turing en 1936. Idée :

- décrire ce qu'est un calcul
- et même développer un modèle de calcul qui englobe tous les modèles connus
- le rendre aussi simple que possible

Une machine de Turing calcule des fonctions

- il existe des fonctions non calculables
- les machines de Turing peuvent donc calculer un sous-ensemble des fonctions existantes
- cependant tous les modèles de calcul décrits jusqu'ici calculent le même ensemble de fonctions (ou moins)

Attention, ne pas confondre

- calculable
- pouvant être calculé en un temps « raisonnable »

Idée intuitive

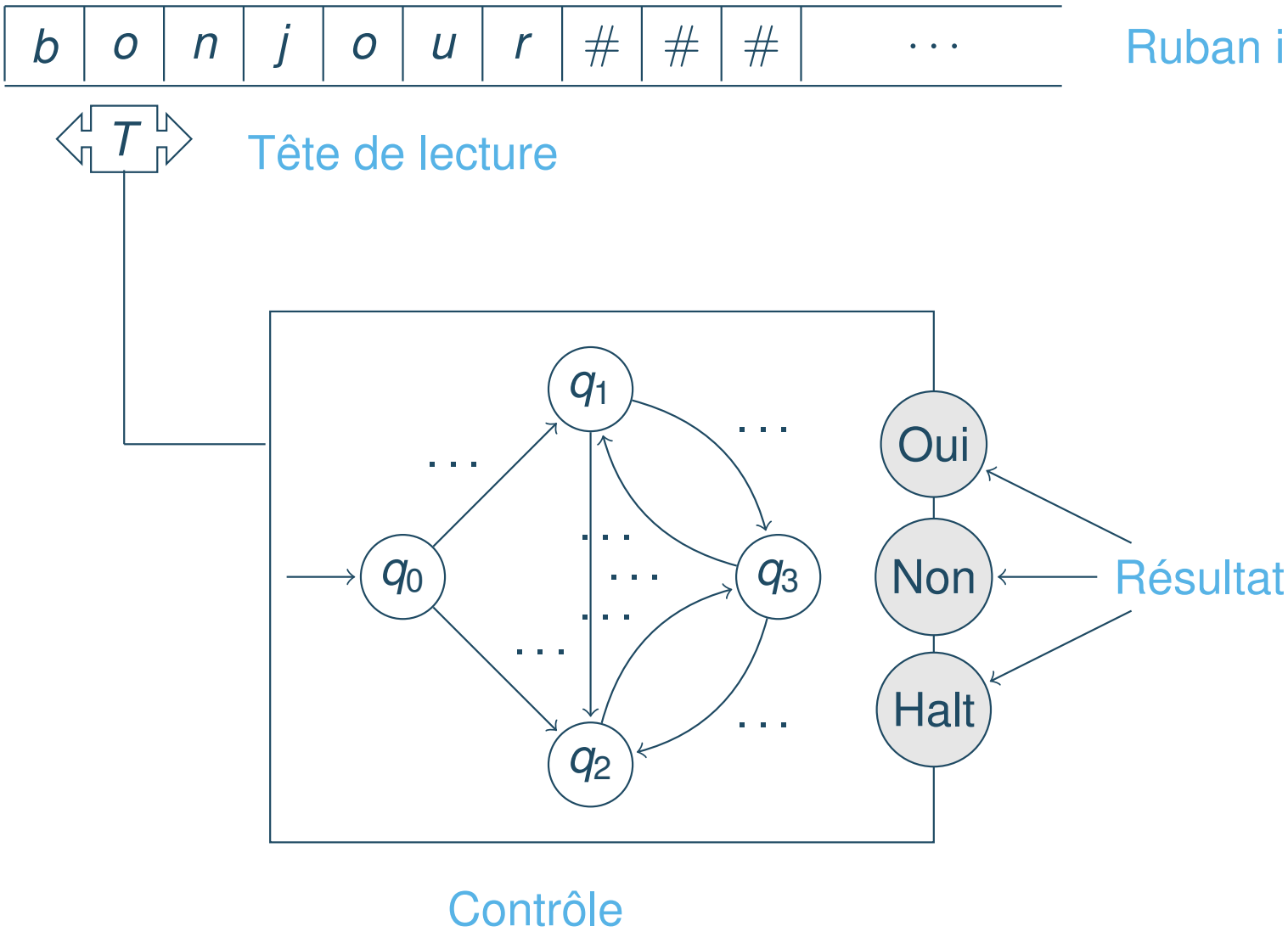
Une machine de Turing est une abstraction d'un ordinateur. Elle est constituée

- d'un automate de contrôle qui représente le processeur, le nombre d'états de ce contrôle est fini de même que le nombre d'états d'un processeur
- d'un ruban qui représente la mémoire de l'ordinateur, finie d'un côté (mémoire adressée à partir de 0) et infinie de l'autre côté (à discuter)
- d'une tête de lecture qui représente le bus qui relie le processeur et la mémoire

On peut aussi se représenter une machine de Turing comme une abstraction d'un humain qui réalise un calcul :

- l'automate de contrôle est le cerveau de l'humain
- le ruban est une feuille de papier quadrillée
- la tête de lecture est la pointe du stylo

Principe d'une machine de Turing

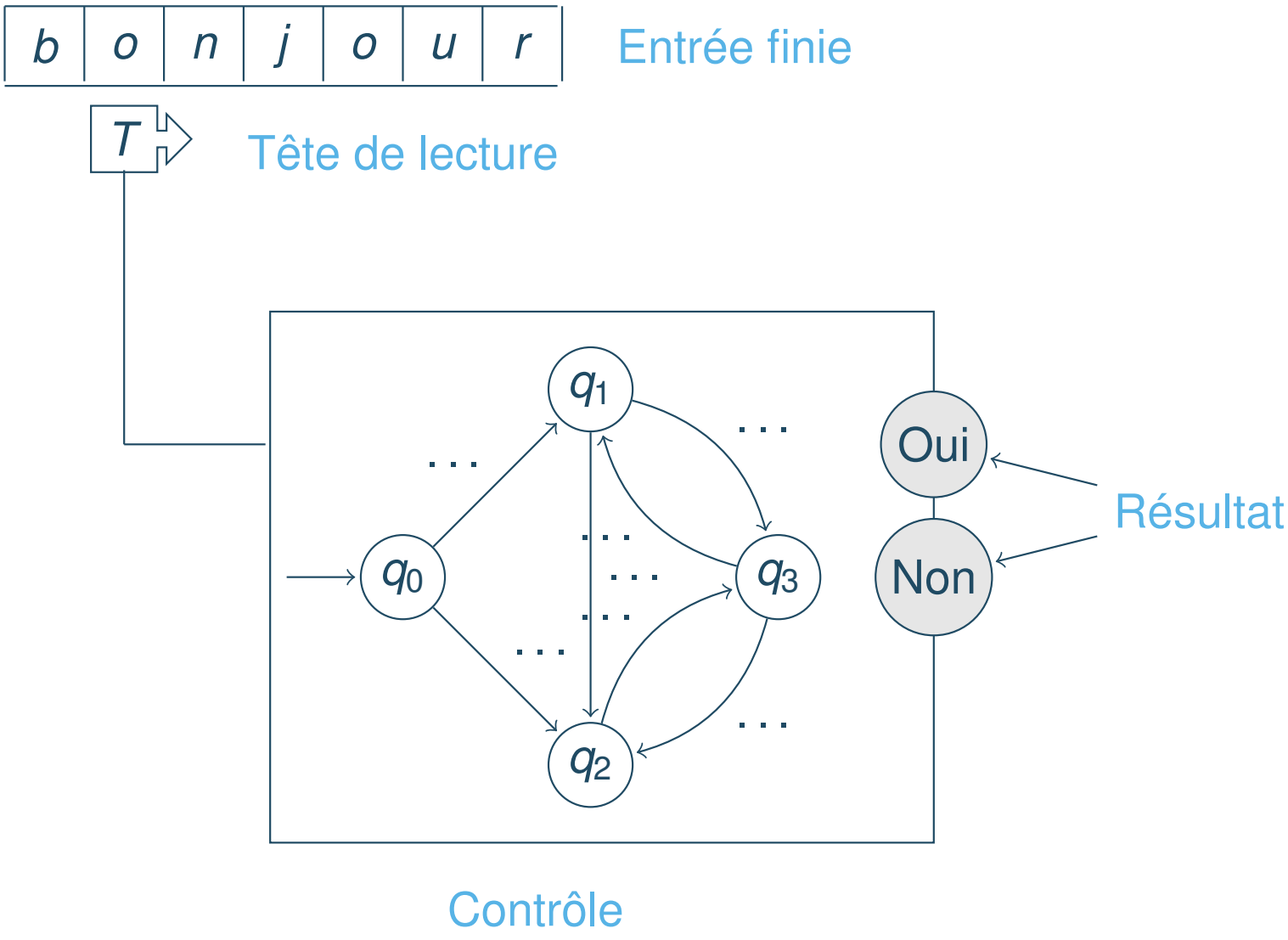


Machine de Turing - Définition

Fonctionnement

- L'entrée est placée sur le ruban (pas de limite de taille)
- La machine peut lire et écrire un caractère dans la case sous la tête de lecture
- La tête de lecture peut se déplacer vers la droite ou vers la gauche
- La machine peut répondre « oui » si l'entrée est reconnue ou « non » si elle ne l'est pas
- La machine peut s'arrêter en ayant calculé un résultat qui sera alors sur le ruban

Retour sur les automates finis (1/2)



Retour sur les automates finis (2/2)

Un automate fini est un outil moins puissant

- l'automate de contrôle a un nombre fini d'états
- l'entrée est une chaîne finie placée en mémoire
- la machine peut lire un caractère dans la case sous la tête de lecture
- la tête de lecture se déplace vers la droite
- la machine peut répondre « oui » si l'entrée est reconnue ou « non » si elle ne l'est pas
- Il n'y a pas d'autre résultat

Machine de Turing - Définition

Une **Machine de Turing** est un septuplet $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F, \#)$ tel que :

- Q est un ensemble fini d'états
- Σ est un alphabet fini, l'alphabet d'entrée
- Γ est un alphabet fini, l'alphabet du ruban, $\Sigma \subseteq \Gamma$ et $\# \in \Gamma$
- $\delta \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L, R\}$ est un ensemble fini de transitions
- $q_0 \in Q$ l'état initial de la machine
- $F \subseteq Q$ l'ensemble des états finaux
- $\# \in \Gamma$ le symbole qui remplit au départ toutes les cases du ruban autres que celles contenant l'entrée

Notations :

- une transition (q, a, q', b, x) est aussi notée $q, a \to q', b, x$
- les symboles désignant les déplacements de tête de lecture L et R sont aussi notés $\leftarrow$ et $\rightarrow$

Machine de Turing déterministe

Une machine de Turing est **déterministe** si $\forall q \in Q, \forall a \in \Gamma$,

$$((q, a, p, b, x) \in \delta \wedge (q, a, p', b', x') \in \delta) \Rightarrow (p = p' \wedge b = b' \wedge x = x')$$

i.e. pour un état de la machine et un symbole de ruban sous la tête de lecture, il existe une seule transition possible. Dans ce cas, δ est une fonction (partielle)

Fonctionnement

Le fonctionnement d'une machine de Turing est simple

- au départ :
  - L'entrée, un mot de Σ^*, est placé sur le ruban, une lettre par case, dans des cases successives, de gauche à droite
  - Le reste du ruban est rempli de symboles $\#$
  - Le contrôle est dans l'état q_0
  - La tête de lecture est placée sur le 1^{er} symbole de l'entrée
- durant le calcul :
  - dans un état q, lorsque le symbole a est sous la tête de lecture
  - s'il existe une transition $(q, a, q', b, x) \in \delta$ alors la machine passe dans l'état q', le symbole a est remplacé sur le ruban par b et la tête de lecture se déplace d'une case vers la gauche ou vers la droite selon la valeur de x

Configuration

La **configuration** d'une machine de Turing décrit son état global à un instant donné :

- l'état de contrôle (état de Q)
- le contenu du ruban
- la position de la tête de lecture

Formellement, une configuration de $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F, \#)$ est un triplet $(u, q, v) \in$ tel que :

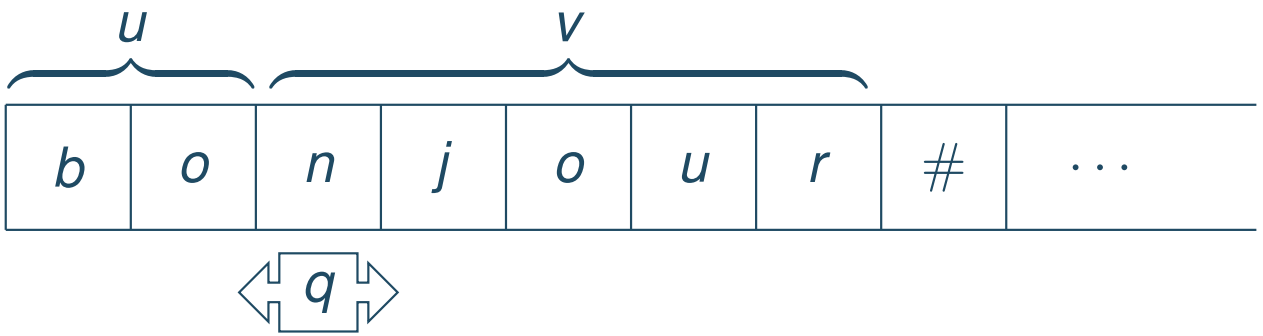
- $q \in Q$ est l'état de contrôle courant
- $u \in \Gamma^*$ est le contenu du ruban situé strictement à gauche de la tête de lecture
- $v \in \Gamma^*$ est le contenu du ruban situé à droite de la tête de lecture, dans lequel on omet l'infinité de $\#$ après le dernier symbole significatif

➥ la tête de lecture est placée sur le premier symbole de v

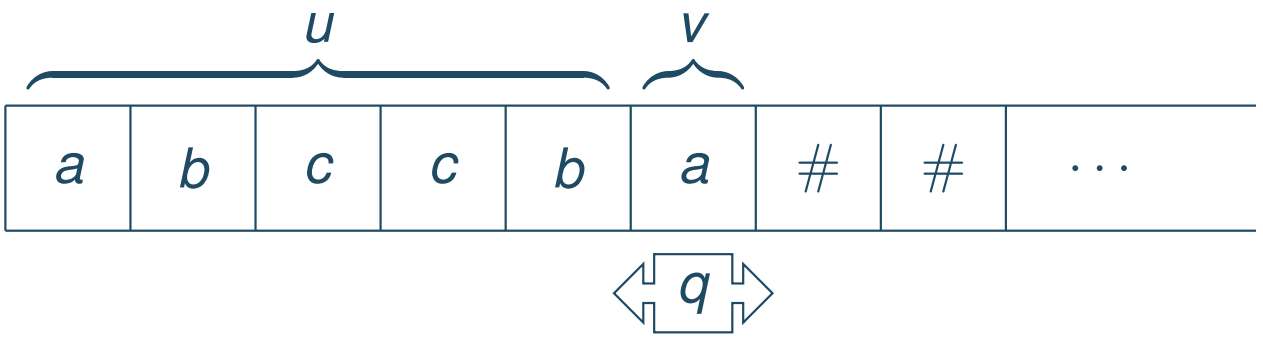
Configuration - Exemple

Exemples de configurations (u, q, v)

- configuration $(bo, q, njour)$



- configuration $(abccb, q, a)$

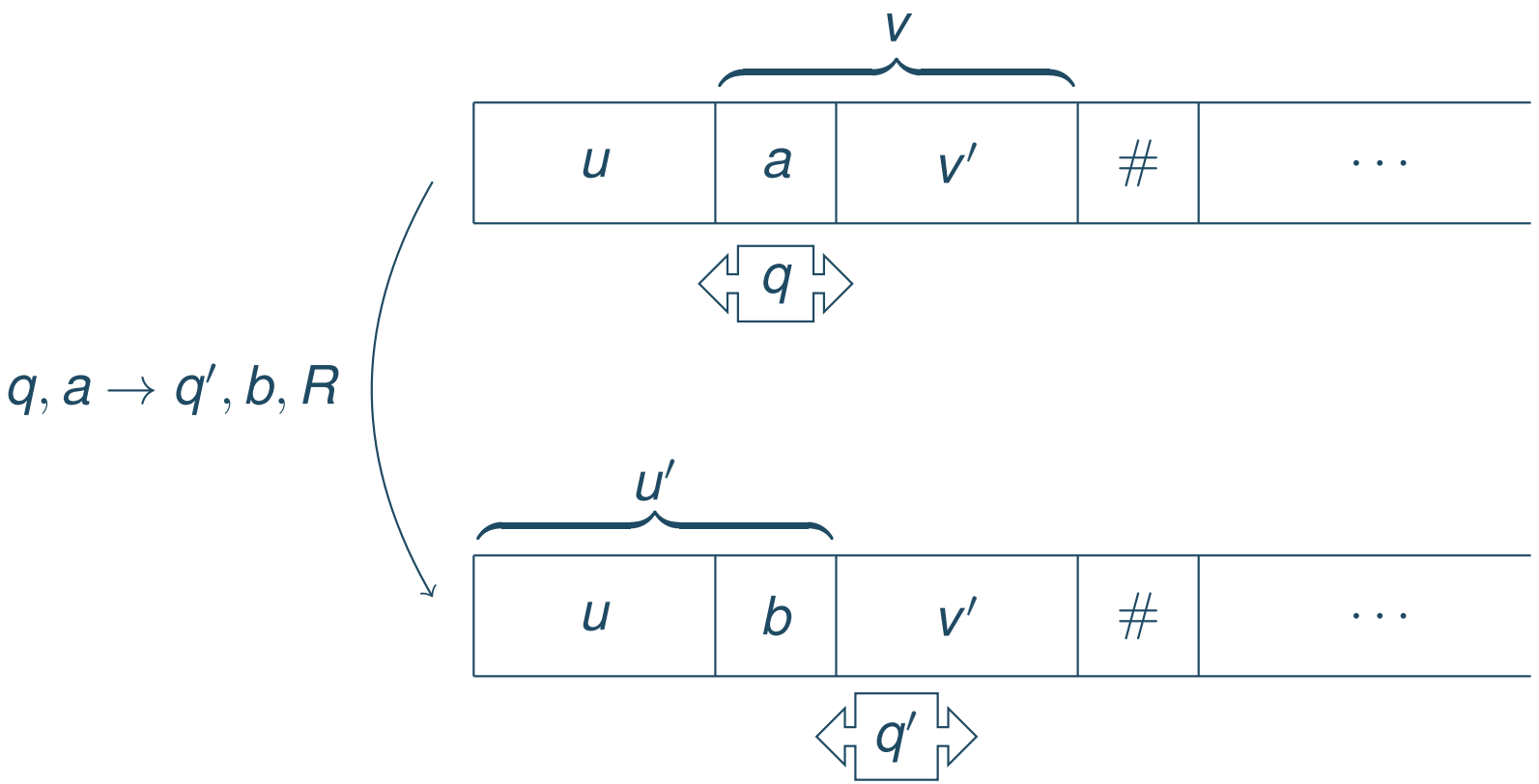


Étape de calcul (1/2)

Une **étape de calcul** fait passer la machine d'une configuration à une autre :

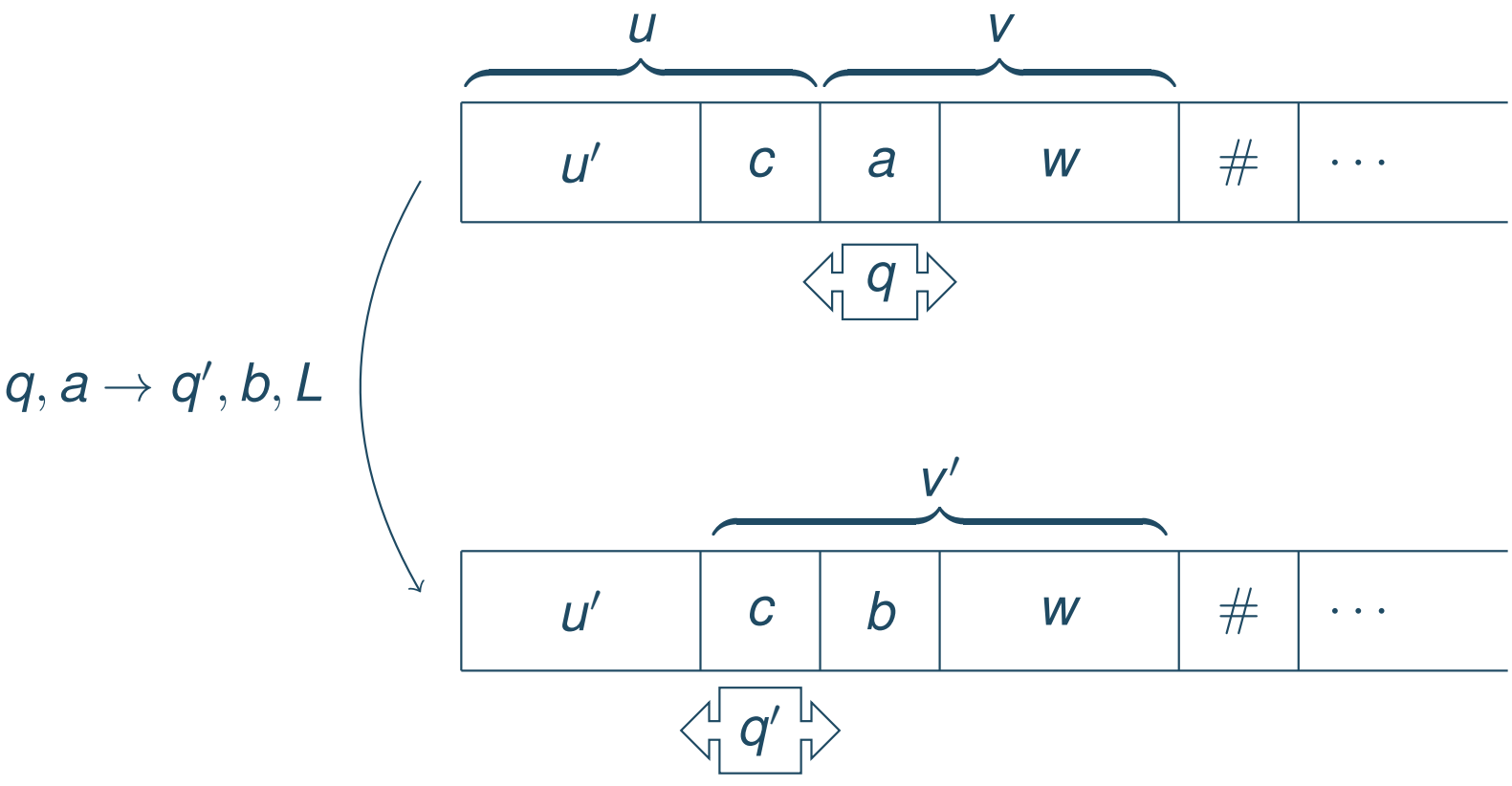
$$(u, q, v) \to (u', q', v')$$

1) si $v = av'$, $u' = ub$ et $q, a \to q', b, R \in \delta$



Étape de calcul (2/2)

2) si $u = u'c$, $v = aw$, $v' = cbw$ et $q, a \to q', b, L \in \delta$



3) si $v = aw$ et qu'il n'existe pas de transition (q, a, q', x, y) appartenant à δ, ou si un déplacement vers la gauche n'est pas possible, alors la machine est bloquée

Calcul

Un **calcul** d'une machine de Turing $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F, \#)$ est une suite d'étapes de calcul qui mènent d'une configuration c_1 à une configuration c_n

$$c_1 \to c_2 \to \cdots \to c_{n-1} \to c_n$$

on note

$$c_1 \xrightarrow{*} c_n$$

Un calcul est **acceptant** si

- $c_1 = (\varepsilon, q_0, u)$ avec $u \in \Sigma^*$
- $c_n = (u', q_F, v')$ avec $q_F \in F$

Langage accepté

Soit $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, F, \#)$ une machine de Turing. Le **langage accepté** par $\mathcal{M}$, noté $\mathcal{L}(\mathcal{M})$ est défini par

$$\mathcal{L}(\mathcal{M}) = \{u \in \Sigma^* \mid \exists q \in F, \exists v, w \in \Gamma^*, (\varepsilon, q_0, u) \xrightarrow{*} (v, q, w)\}$$

Rappel. Hierarchie de Chomsky

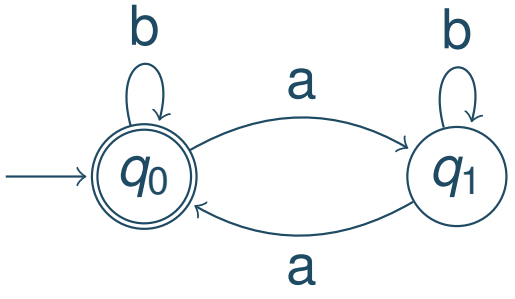
Type	Grammaires	Langages	Outil
3	régulières	réguliers	automates finis
2	algébriques	algébriques	automates à pile
1	contextuelles	langages contextuels	machines de Turing
0		décidables*	machines de Turing

(*)reconnus par une machine en un temps fini

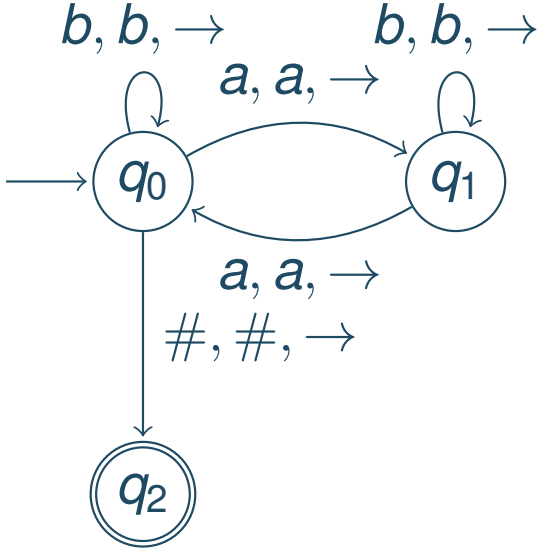
Exemple - Reconnaître un langage régulier

Langage sur $\Sigma = \{a, b\}$ des mots qui contiennent un nombre pair de a

Automate fini

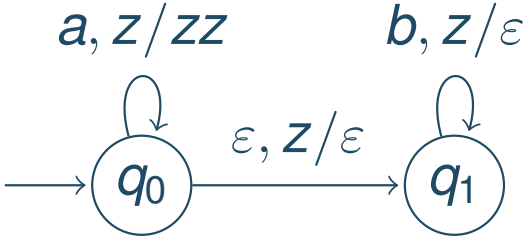


Machine de Turing

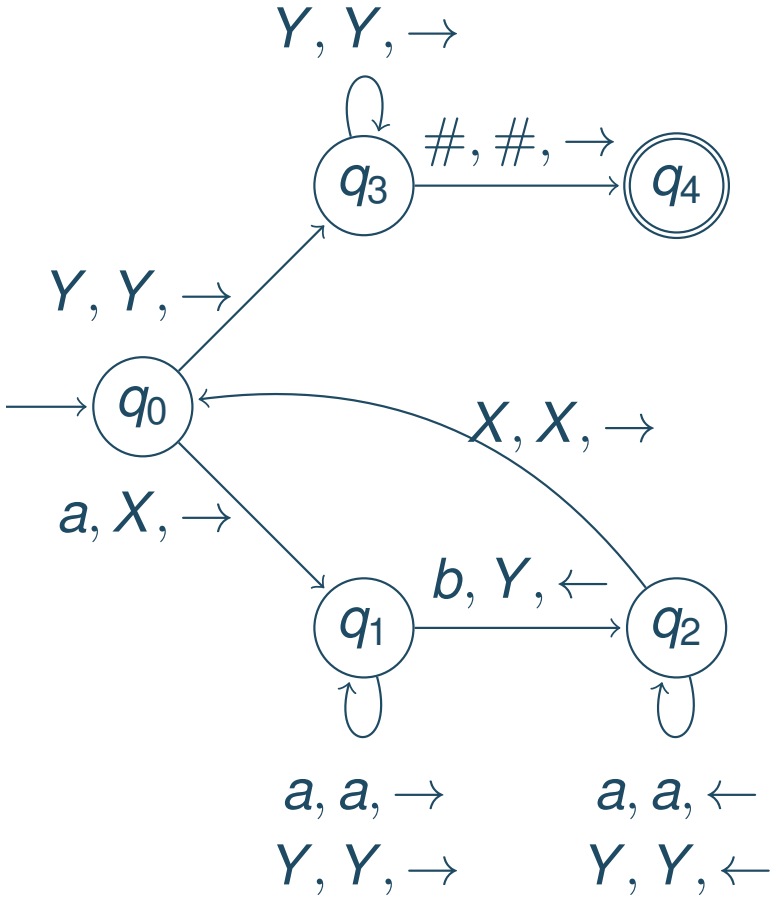


Exemple - Reconnaître un langage algébrique $a^n b^n$

Automate à pile

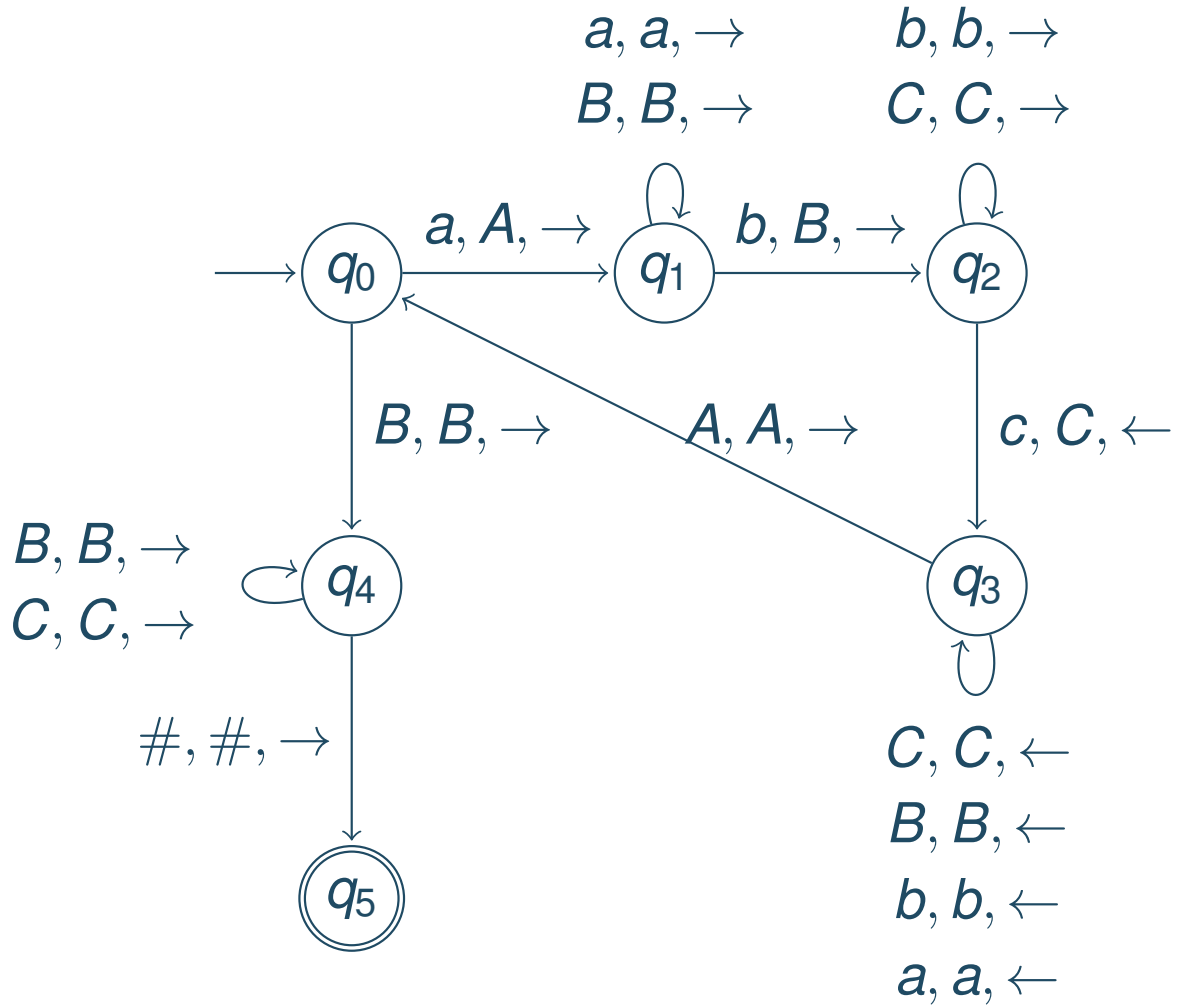


Machine de Turing



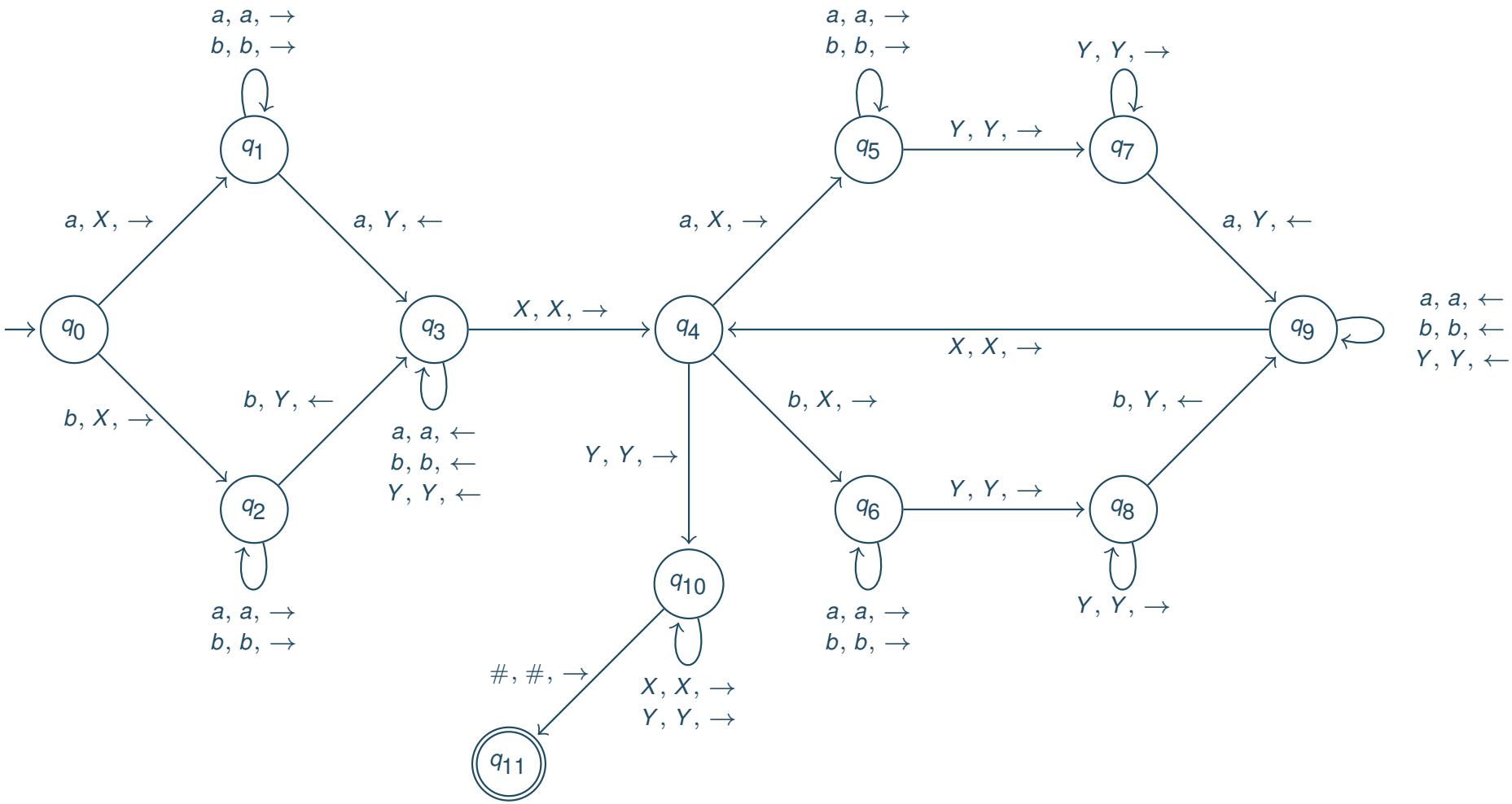
Exemple - Reconnaître un langage non algébrique

Langage $L = \{a^n b^n c^n \mid n \in \mathbb{N}\}$



Exemple - Reconnaître un langage non algébrique

Langage $\Sigma = \{a, b\} : L = \{uu \mid u \in \Sigma^*\}$



Langage accepté / Langage décidé

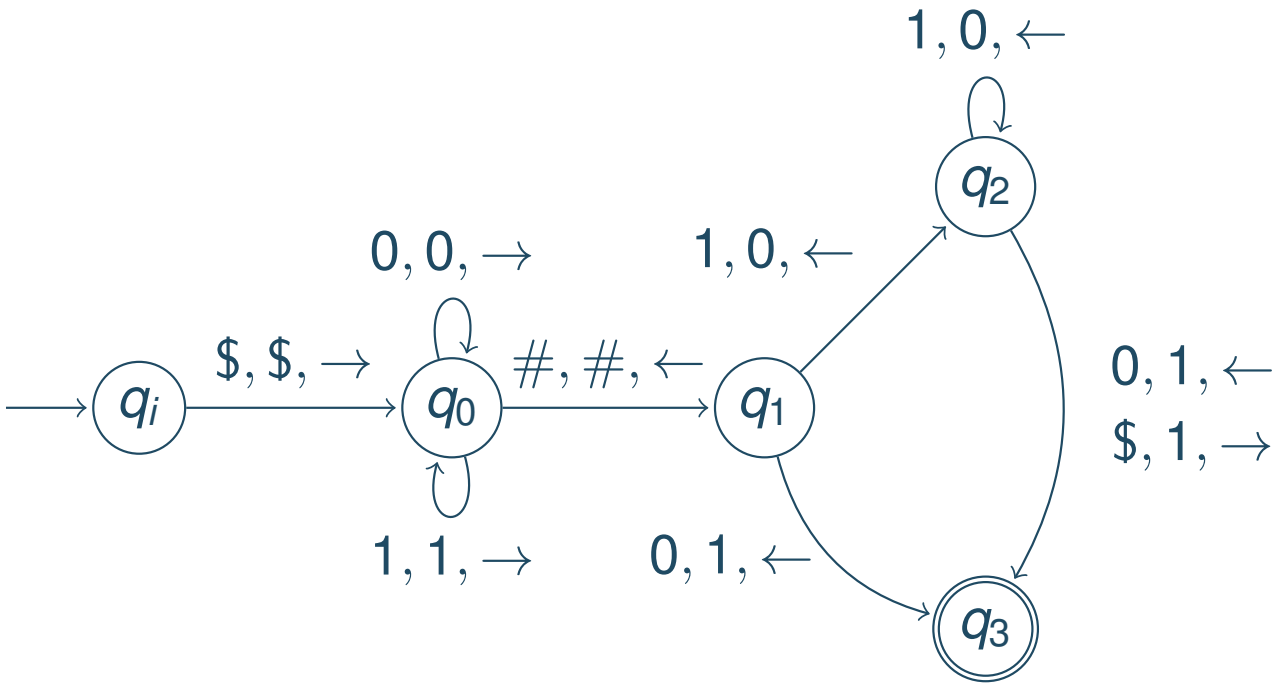
Lorsqu'un calcul est effectué sur une machine de Turing à partir d'une configuration initiale (ε, q_0, u) celui-ci peut :

- se terminer dans un état final et le mot u est reconnu
- se terminer dans un état de blocage parce qu'il n'y a pas de transition possible ou parce qu'un déplacement à gauche entraînerait le dépassement du début du ruban
- être infini car le calcul ne passe jamais par un état final

Un langage L est dit **récursivement énumérable** s'il est reconnu par une machine de Turing. Un langage est dit **décidable** s'il est reconnu par une machine de Turing qui s'arrête pour toutes les entrées

Exemple - Calculer un résultat

Addition de 1 à un nombre binaire qui commence par \$, poids forts à gauche



Variantes des machines de Turing

Il existe des variantes qui permettent de simplifier l'écriture de certaines fonctions :

- machine à ruban infini à droite et à gauche
- machine à multiples rubans

➥ Ces variantes sont montrées équivalentes, en particulier les machines de Turing déterministes

Proposition

Pour toute machine de Turing $\mathcal{M}$, il existe une machine de Turing déterministe $\mathcal{M}'$ qui reconnait le même langage. De plus si $\mathcal{M}$ n'admet pas de calcul infini, il en est de même pour $\mathcal{M}'$

Thèse de Church

- A. Church a postulé que tous les modèles de calcul définissent la même notion de calculable
- C'est un postulat, donc non démontré
- Il peut être énoncé de différentes manières
- Il est parfois appelé Thèse de Church-Turing

Thèse de Church

Les langages reconnus par une procédure effective sont les langages décidés par une machine de Turing

Thèse de Church

La Thèse de Church n'est pas démontrée... mais elle est admise :

- depuis les années 30, personne n'a réussi à démontrer qu'elle était fausse
- de nombreux autres modèles ont été introduits depuis et se sont avérés équivalents : les langages de programmation, les RAM (random access machines), les automates cellulaires, ... et même : les ordinateurs quantiques ou le DNA computing

Machine de Turing universelle

- Une machine de Turing n'est rien d'autre qu'une suite finie de symboles...
- Elle peut donc être donnée (accompagnée de son entrée) en entrée d'une machine de Turing !!!

➥ On peut utiliser une machine de Turing pour simuler le calcul d'une machine de Turing !

Théorème

Il existe des machines de Turing universelles

Idée

Il s'agit de machines qui simulent l'exécution de n'importe quelle machine de Turing sur n'importe quelle entrée

On construit une machine à 2 rubans, le premier contenant la machine que l'on veut simuler et son entrée, le second la configuration de la machine à un instant donné

Indécidabilité

Un problème (oui/non) est **indécidable** s'il n'existe pas de machine de Turing qui le résolve

- Turing a démontré en 1936 que le problème de l'arrêt d'un programme est indécidable
- Il existe donc des problèmes qu'une machine de Turing ne peut résoudre (celui-là et de nombreux autres)
- Il existe donc de (nombreux) problèmes qu'un ordinateur ne peut résoudre

Arrêt d'un programme (1/2)

Montrons par l'absurde que l'arrêt d'un programme est indécidable

- Supposons que l'arrêt d'un programme soit décidable
- Alors il existe un programme `int halt(char *p, char *x)` qui pour un programme `p` et une entrée `x` répond `0` (soit « oui ») si le programme s'arrête et `1` (soit « non »)s'il ne s'arrête pas

Considérons le programme

```
void inverse (char *p) {
   if (halt(p,p))
      while (true);
}
```

Que se passe-t-il si on appelle `inverse(inverse)` ?

Arrêt d'un programme (1/2)

Regardons ce qui se passe si on appelle `inverse(inverse)`

- si `inverse` s'arrête, alors `halt(inverse,inverse)` est vrai et donc le programme exécute `while (true);`

  ➥ `inverse` ne s'arrête pas

- si `inverse` ne s'arrête pas, alors `halt(inverse,inverse)` est faux et donc le programme n'exécute pas le corps du `if`

  ➥ `inverse` s'arrête

Arrêt d'un programme (1/2)

Regardons ce qui se passe si on appelle `inverse(inverse)`

- si `inverse` s'arrête, alors `halt(inverse,inverse)` est vrai et donc le programme exécute `while (true);`

  ➥ `inverse` ne s'arrête pas

- si `inverse` ne s'arrête pas, alors `halt(inverse,inverse)` est faux et donc le programme n'exécute pas le corps du `if`

  ➥ `inverse` s'arrête

➥Contradiction

Pourquoi s'intéresser à ce genre de questions ?

- Pour la « beauté » de la chose ☺
- Plus sérieusement : établir des liens et équivalences entre les différents modèles, étudier la consistance des théories, . . .
- en pratique, pourquoi est-ce si difficile de programmer, et de mettre au point un programme ? La plupart des problèmes rencontrés sont indécidables
  - Est-ce qu'une fonction donnée s'arrête pour une entrée ? pour chaque entrée ?
  - Est-ce qu'il existe une exécution du programme qui passe par une instruction donnée ?
  - Est-ce qu'une variable donnée est toujours initialisée avant d'être utilisée ?
  - . . .