

# Théorie des Langages

Yannick Le Nir    Gaspard Férey

CY tech

ING1



## Problèmes de Hilbert

- David Hilbert : 1862-1943
- Liste de 23 problèmes  
(Exposition universelle de 1900  
à Paris)
- Problème numéro 10 :  
*"Trouver un algorithme  
déterminant si une équation  
diophantienne à des solutions."*



## Machine de Turing

- Alan Matheson Turing :  
1912-1954
- Rôle actif pour décrypter la machine Enigma pendant la seconde guerre mondiale
- Inventeur de la machine de Turing (1936)



# Machine de Turing

## Généralités

- Tout algorithme peut être traduit en un programme pour la machine de Turing
- Modèle théorique de l'ordinateur (réalisations physiques dès 1940).

## Description

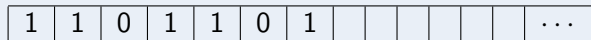
- Ruban infini : suite de cases portant chacune un élément d'un alphabet
- Semblable aux automates finis, sauf qu'elle peut lire, écrire et se déplacer sur le ruban
- Déplacement d'une seule case (droite ou gauche) à la fois
- Diagramme de transition pour modéliser son comportement

## Historique

- En 1935, Turing essaye de résoudre la question posée par Hilbert, reformulée par :  
*"Existe-t-il, au moins théoriquement, une méthode ou un processus moyen duquel toutes les questions mathématiques peuvent être décidées"*
- *Thèse de Church-Turing* : "Aucune procédure de calcul ne peut être considérée comme un algorithme à moins qu'on puisse la représenter comme une machine de Turing"
- *Thèse de Turing* : "Tout ce qui est calculable peut être calculé par une machine de Turing"
- *Vrai si calculer signifie manipuler un nombre fini de symboles et produire une réponse après un nombre fini d'étapes*

# Machine de Turing

## Représentation de la "mémoire"



Mécanisme de contrôle : nombre d'états fini

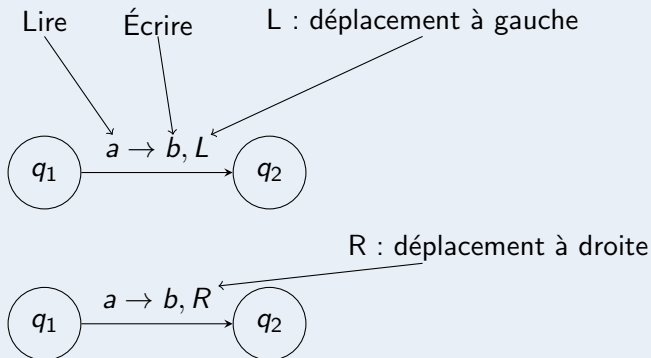
- La tête de lecture
  - ▶ commence à gauche du ruban
  - ▶ déplaçable librement dans les deux sens
- Le ruban
  - ▶ contient le mot en entrée (canal d'entrée)
  - ▶ est réinscriptible à volonté (mémoire)
  - ▶ contient le mot retour (canal de sortie)
- Règles de transitions :
  - ▶ état initial
  - ▶ caractère lu
  - ▶ état final
  - ▶ caractère écrit
  - ▶ déplacement

## Fonctionnement

- Initialisation :
  - ▶ Le mot d'entrée est inscrit sur le ruban
  - ▶ La tête de lecture est positionnée sur le caractère le plus à gauche
- A chaque étape, la machine de Turing :
  - ▶ lit le symbole sous la tête de lecture
  - ▶ écrit un symbole sous la tête de lecture
  - ▶ déplace la tête vers la droite ou vers la gauche
  - ▶ effectue une transition d'état

# Machine de Turing

## Etats et Transitions



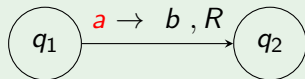
# Machine de turing

## Exemple de transition



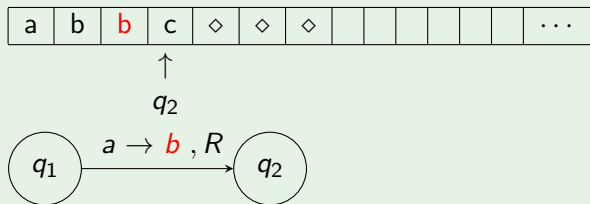
↑

$q_1$



# Machine de turing

## Exemple de transition



# Machine de Turing

## Arrêt

- La machine s'arrête s'il n'y a plus de transition possible

## Acceptation et Langage

- Mot accepté si la machine s'arrête dans un état final :

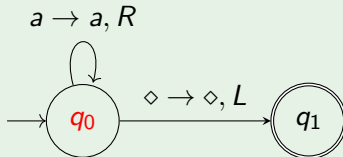


- Mot rejeté si la machine
  - ▶ s'arrête dans un état non final
  - ▶ ne s'arrête pas (boucle)
- L'ensemble des mots acceptés constitue le langage de la machine de Turing

# Machine de Turing

## Exemple de Machine de Turing

Soit la machine suivante qui accepte le langage  $L = a^*$  :

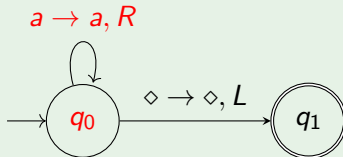


$T = 0$

# Machine de Turing

## Exemple de Machine de Turing

Soit la machine suivante qui accepte le langage  $L = a^*$  :

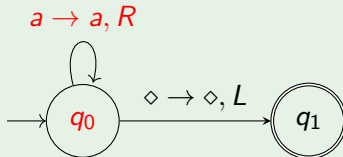


$q_0$   
 $T = 1$

# Machine de Turing

## Exemple de Machine de Turing

Soit la machine suivante qui accepte le langage  $L = a^*$  :

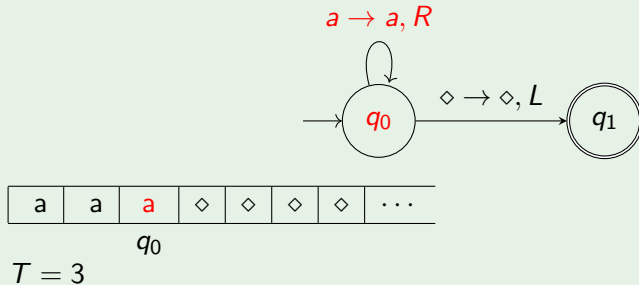


$q_0$   
 $T = 2$

# Machine de Turing

## Exemple de Machine de Turing

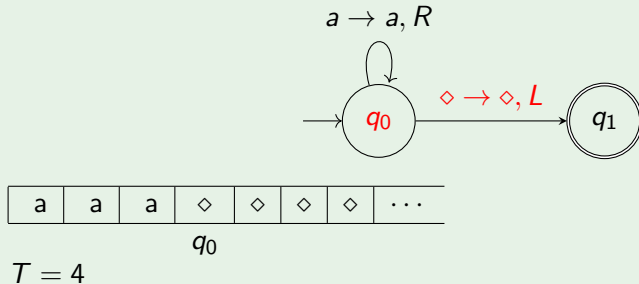
Soit la machine suivante qui accepte le langage  $L = a^*$  :



# Machine de Turing

## Exemple de Machine de Turing

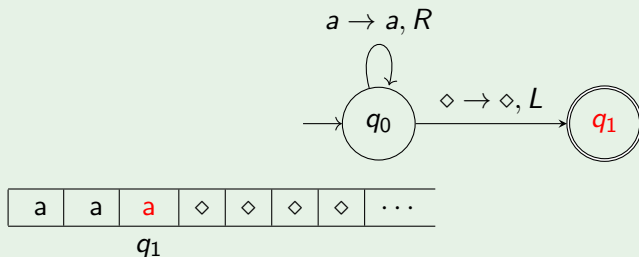
Soit la machine suivante qui accepte le langage  $L = a^*$  :



# Machine de Turing

## Exemple de Machine de Turing

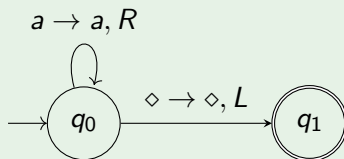
Soit la machine suivante qui accepte le langage  $L = a^*$  :



$T = 5 \dots$  : état final ; arrêt et acceptation.

# Machine de Turing

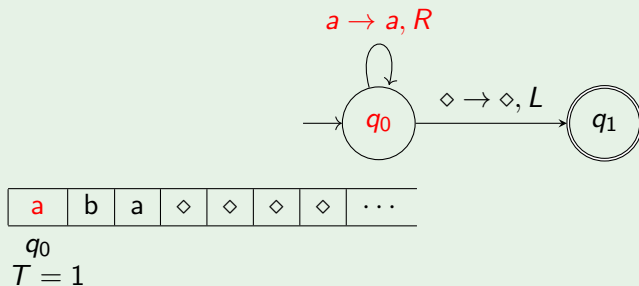
## Exemple de rejet



$T = 0$

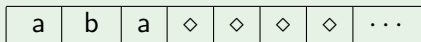
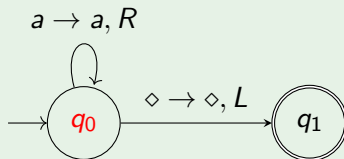
# Machine de Turing

## Exemple de rejet



# Machine de Turing

## Exemple de rejet

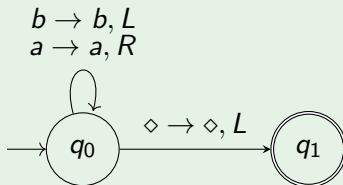


$q_0$

$T = 2$  : pas de transition possible ; arrêt et rejet.

# Machine de Turing

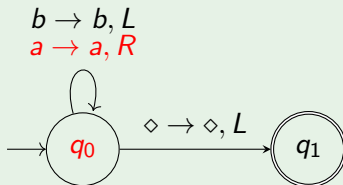
## Exemple de boucle infinie



$T = 0$

# Machine de Turing

## Exemple de boucle infinie

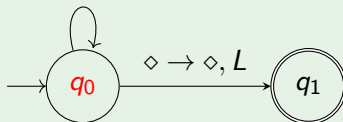


$q_0$   
 $T = 1$

# Machine de Turing

## Exemple de boucle infinie

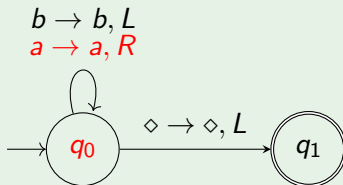
$b \rightarrow b, L$   
 $a \rightarrow a, R$



$q_0$   
 $T = 2$

# Machine de Turing

## Exemple de boucle infinie

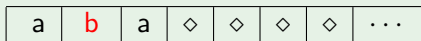
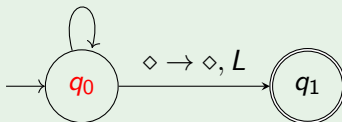


$q_0$   
 $T = 3$

# Machine de Turing

## Exemple de boucle infinie

$b \rightarrow b, L$   
 $a \rightarrow a, R$

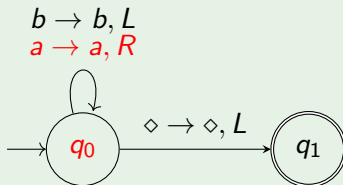


$q_0$

$T = 4$

# Machine de Turing

## Exemple de boucle infinie

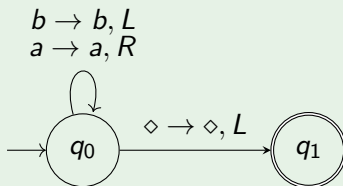


$q_0$

$T = 5 \dots$  : boucle infinie.

# Machine de Turing

## Exemple de boucle infinie



$T =$

## Définition formelle

$$M = (\mathcal{Q}, \Sigma, \Gamma, \delta, q_0, \diamond, F)$$

avec

- $\mathcal{Q}$  : Etats
- $\Sigma$  : Alphabet d'entrée
- $\Gamma$  : Alphabet du ruban
- $\delta$  : fonction de transition (ex :  $\delta(q_1, a) = (q_2, b, R)$ )
- $q_0$  : état initial
- $\diamond$  : blanc
- $F$  : Etat final

# Machine de Turing

## Utilisation

Une machine de Turing peut-être utilisée de deux manière différentes :

- Comme machine à reconnaître des langages
  - ▶ Seul importe l'état de la machine à l'arrêt.
- Comme transformateur (ou calculateur)
  - ▶ Le mot écrit sur le ruban quand la machine termine est interprété comme le résultat d'une fonction calculée par la machine.
  - ▶ Le rejet est interprété comme une exception/erreur.
  - ▶ Plusieurs machines de Turing peuvent être utilisées successivement. Semblable à la programmation usuelle.
  - ▶ Attention : la position finale de la tête peut-être importante.

# Machine de Turing

## Langage accepté

Pour toute machine de Turing  $M$ ,

$$L(M) = \{w : q_0 w \mapsto^* x_1 q_f x_2\}$$

avec

- $q_0 w$  : configuration initiale (état  $q_0$  et tête sur première lettre de  $w$ )
- $q_1 xv \mapsto x q_2 v$  : déplacement de la tête en lisant la lettre  $x$  et en passant de l'état  $q_1$  à l'état  $q_2$
- $\mapsto^*$  : occurrence multiple du déplacement  $\mapsto$

# Machine de Turing

Soit  $L$  un langage.

## Décidabilité

- Langage décidable :  $L$  est décidable ssi il existe un algorithme qui permet de reconnaître en un temps fini si un mot  $w$  appartient ou non à  $L$
- $L$  est décidé par une machine de Turing  $M$  si
  - ▶  $M$  accepte  $L$
  - ▶  $M$  n'a pas d'exécution infinie

## Semi-décidabilité

- Si  $M$  peut avoir des exécutions infinies mais accepte  $L$ , alors  $L$  est dit semi-décidable ou recursivement énumérable.

# Machine équivalentes

## Extensions

- La définition d'une machine de Turing peut-être étendue pour des raisons pratiques.
- On peut prouver que les machines de Turing étendues ne permettent pas de calculer plus de choses.

# Machine équivalentes

## Exemples

Les machine abstraites suivantes sont équivalentes aux machines de Turing :

- Machine de Turing autorisant les transitions avec lectures multiples
- Machine de Turing autorisant les transitions sans déplacement de la tête de lecture
- Machine de Turing à ruban infini dans les deux directions
- Machine de Turing à 2 rubans (et 2 têtes de lectures indépendantes)
- Machine de Turing à  $n$  rubans
- Machine de Turing à  $k$  têtes de lectures
- Machine de Turing multidimensionnelles
- Machine de Turing non déterministes

# Grammaires générales

## Exemple

$$\begin{array}{lll} S \rightarrow TZ & T \rightarrow 0T1C & T \rightarrow \varepsilon \\ C1 \rightarrow 1C & CZ \rightarrow Z2 & 1Z \rightarrow 1 \end{array}$$

Langage engendré :

# Grammaires générales

## Exemple

$$\begin{array}{lll} S \rightarrow TZ & T \rightarrow 0T1C & T \rightarrow \varepsilon \\ C1 \rightarrow 1C & CZ \rightarrow Z2 & 1Z \rightarrow 1 \end{array}$$

Langage engendré :  $0^i 1^i 2^i$

## Exemple

$$\begin{array}{lll} S \rightarrow TZ & T \rightarrow 0T1C & T \rightarrow \varepsilon \\ C1 \rightarrow 1C & CZ \rightarrow Z2 & 1Z \rightarrow 1 \end{array}$$

Langage engendré :  $0^i 1^i 2^i$

Langage non algébrique, non analysable via les automates à piles.

## Théorème (Chomsky 1959)

Le langage  $L$  est engendré par une grammaire générale si et seulement si il est accepté par une machine de Turing (automate à deux piles).

## Utilisation pratique

Nous verrons l'an prochain (cours de décidabilité), que malheureusement, ces langages sont en général indécidables, donc inexploitable dans la pratique.

# Grammaires contextuelles

Restriction des grammaires générales, en contraignant les parties droites des règles à être au moins aussi long que les parties gauches. Ceci exclu évidemment le mot vide.

## Exemple

$$\begin{array}{lll} S \rightarrow TZ & T \rightarrow 0U1 & T \rightarrow 01 \\ U \rightarrow 0U1C & U \rightarrow 01C & \\ C1 \rightarrow 1C & CZ \rightarrow Z2 & 1Z \rightarrow 12 \end{array}$$

Langage engendré :

# Grammaires contextuelles

Restriction des grammaires générales, en contraignant les parties droites des règles à être au moins aussi long que les parties gauches. Ceci exclu évidemment le mot vide.

## Exemple

$$\begin{array}{lll} S \rightarrow TZ & T \rightarrow 0U1 & T \rightarrow 01 \\ U \rightarrow 0U1C & U \rightarrow 01C & \\ C1 \rightarrow 1C & CZ \rightarrow Z2 & 1Z \rightarrow 12 \end{array}$$

Langage engendré :  $0^i 1^i 2^i$

# Grammaires contextuelles

Restriction des grammaires générales, en contraignant les parties droites des règles à être au moins aussi long que les parties gauches. Ceci exclu évidemment le mot vide.

## Exemple

$$\begin{array}{lll} S \rightarrow TZ & T \rightarrow 0U1 & T \rightarrow 01 \\ U \rightarrow 0U1C & U \rightarrow 01C & \\ C1 \rightarrow 1C & CZ \rightarrow Z2 & 1Z \rightarrow 12 \end{array}$$

Langage engendré :  $0^i 1^i 2^i$

# Propriété des grammaires contextuelles

## Propriétés

- Soit une grammaire contextuelle et un mot de longueur  $n$ . Il est possible de vérifier si ce mot est engendré par la grammaire : fabrication de toutes les dérivations à partir de l'axiome. Problème dit "*PSPACE*-complet".
- Reconnaissance des langages contextuels via les machine de Turing linéairement bornées : machine de Turing non déterministe qui n'utilise de sa mémoire infinie qu'une portion dépendant linéairement de la taille du mot testé.

# Propriété des grammaires contextuelles (suite)

## Théorème

Un langage est contextuel si et seulement si il est accepté par une machine de Turing linéairement bornée.

## Limitations

- On ne sait pas si l'on peut se passer de l'hypothèse de non déterminisme
- Beaucoup de problèmes indécidables, notamment pour déterminer si un langage contextuel est vide

# Hiérarchie de Chomsky

## Outils algorithmique

Manipulation des langages de type 0 et 1 via les machines de Turing :

Langages	Grammaires	Machine abstraite
Langages quelconque		
Langages récursivement énumérables	Type 0	Machine de Turing
Langages décidables		Machine de Turing terminante
Langages contextuels	Type 1	Machine de Turing linéairement bornée
Langages hors contexte	Type 2	Automate à pile
		Automate à pile déterministe
Langages rationnels	Type 3	Automate d'états finis