Républiquealgérienne démocratique et populaire
Ministre de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – Biskra
Faculté des Sc.Exactes, et SNV.
Département d'Informatique

Formalisation de la procédure effective

Niveau: 1er année Master
Option: Génie Logiciel et Systèmes Distribués

2023/2024

Rappel

- Problème vs Algorithme
- Problème de décision
- Procédure effective
- Codage d un problème

Introduction

Exécution d'un programme sur un ordinateur ?

Un ordinateur qui exécute **un programme fixé**, est composé principalement de:

- Un processeur + un ensemble de registres dont il contient le compteur ordinal (CO).
- Une mémoire contenant les instructions du programme (ROM: car programme fixé).
- Une mémoire (RAM) contenant les données.

Exécution d'un programme sur un ordinateur?

L'exécution d'un programme sur un ordinateur est une succession de cycles contenant les étapes suivantes:

1. L'instruction indiquée par le CO est transféré de ROM vers le processeur (Registre instruction).
2. Le processeur exécute cette instruction et modifie, par conséquent, ses registres et/ou la mémoire RAM.
3. Le CO est incrémenté (ou modifié en cas de branchement).

Notion d'état

- D'une façon plus abstraite:

Si on connais le contenu des registres et de la mémoire des données,

➔ On peut connaitre sans ambiguïté, leurs contenus après l'exécution du cycle suivant.

- L'ensemble des informations nécessaires et suffisantes pour prédire son évolution future

➔ ETAT

Chaque contenu de la mémoire et des registres constitue un état différent.

NB: Il n'est pas nécessaire de connaître toutes la succession des instructions qui a abouti à l'état présent pour prédire l'état futur.

Fonction de transition

- Chaque cycle de l'exécution transforme l'état de la machine.
- Cette transformation ne dépend que du programme et de la machine.

Si le programme et la machine sont fixés

➔ transformation: état → état

= une fonction: États → États

- Elle est appelée une **fonction de transition.**
- Pour savoir ce qui va se passer durant l'exécution, il suffit de connaître:
  - La fonction de transition
  - L'état initial

Modélisation d'un programme exécuté par une machine

- Un programme exécuté par une machine est modélisé par:
  - Un ensemble ***fini*** d'états.
  - Une fonction de transition.
  - Un état initial.

- Une exécution du programme est représentée par la ***séquence des états*** obtenue par applications successives de la fonction de transition en partant de l'état initial.

Adaptation sur Les automates

- Automates finis : première modélisation de la notion de procédure effective.(Ont aussi d'autres applications).
- Dérivation de la notion d'automate fini de celle de programme exécutée sur un ordinateur : état, état initial, fonction de transition.
- Hypothèse du nombre d'états fini.

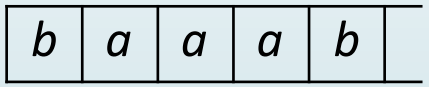
  → Conséquence : séquences d'états finies ou cycliques.
- Problème de la représentation des données : nombre de données différentes limitées car nombre d'états initiaux possibles fini.

Adaptation sur Les automates

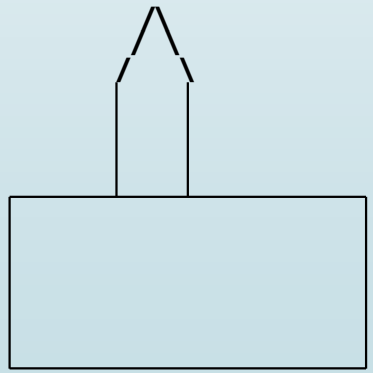
- Représentation des données
  1. Problème : reconnaître un langage.
  2. Données : mot.
  3. On supposera le mot fourni caractère par caractère, la machine traitant un caractère à chaque cycle et s'arrêtant à la fin du mot.

Description générale des automates

- Ruban d'entrée.
- Ensemble d'états :
  - état initial,
  - états accepteurs.
- Mécanisme d'exécution.
  - ruban :



  - Tête de lecture



Formalisation des automates fini

Un automate fini déterministe est défini par un quintuplait $M = (Q, \Sigma, \delta, s, F)$, où

1. Q est un ensemble fini d'états,
2. Σ est un alphabet,
3. $\delta : Q \times \Sigma \rightarrow Q$ est la fonction de transition,
4. $s \in Q$ est l'état initial,
5. $F \subseteq Q$ est l'ensemble des états accepteurs.

Langage accepté

- Configuration : $(q, w) \in Q \times \Sigma^*$.
- Configuration dérivable en une étape : $(q, w) \vdash_M (q', w')$.
- Configuration dérivable (en plusieurs étapes) : $(q, w) \vdash^*_M (q', w')$.
- Exécution d'un automate :

$$(s, w) \vdash (q_1, w_1) \vdash (q_2, w_2) \vdash \ldots \vdash (q_n, \varepsilon),$$

- Mot accepté :

$$(s, w) \vdash^*_M (q, \varepsilon) \text{ et } q \in F .$$

- Langage accepté $L(M)$:

$$\{ w \in \Sigma^* \mid (s, w) \vdash^*_M \quad (q, \varepsilon) \text{ avec } q \in F \}.$$

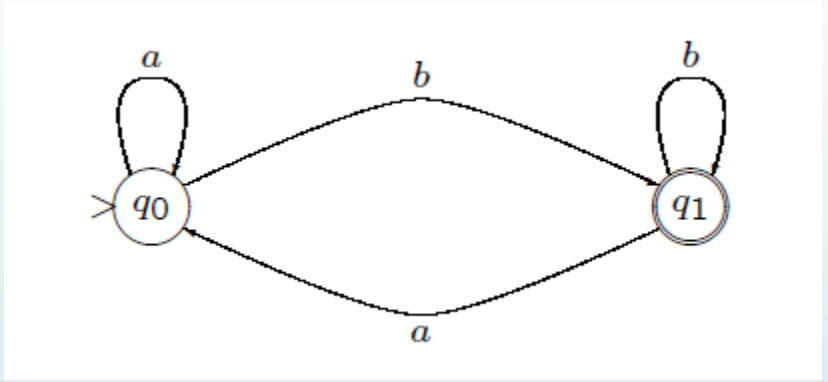
Exemple

- Mots se terminant par b :

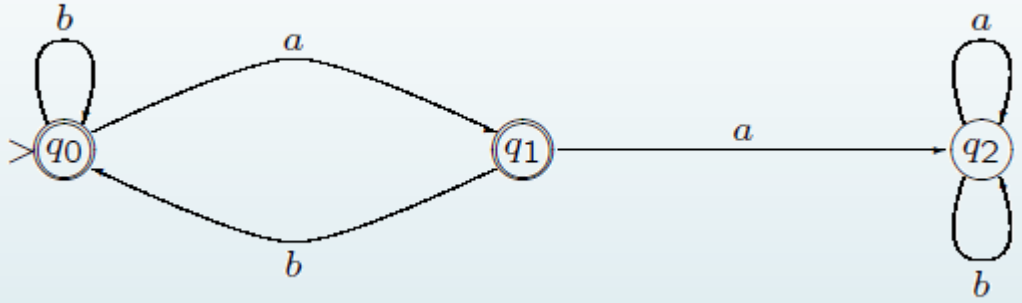
δ :

q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_0
q_1	b	q_1

$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$s = q_0$
$F = \{q_1\}$



Exemple



{w | w ne contient pas 2 a consécutifs}.

Les automates vs langage accepté

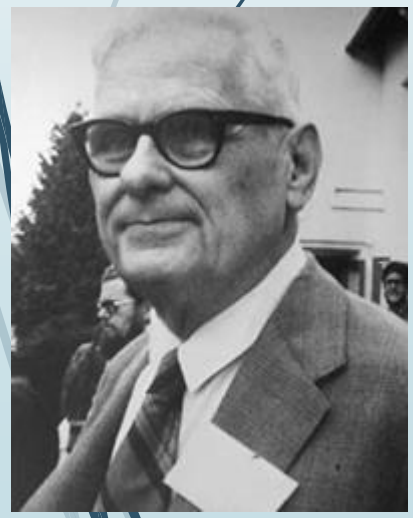
- Automates à états finis: ne peuvent pas reconnaitre les mots comme: $a^n b^n$.
- Automates à pile: ne peuvent pas reconnaitre les mots comme: $a^n b^n c^n$.
- Nous allons adopter les machines de Turing, car elles permettent de reconnaître tous les langages intuitivement reconnaissable par une procédure effective.

Machine de Turing

Un peu de histoire

Church (1930) et Turing (1936)

Alan Turing (1912 – 1954) et Alonzo Church (1903 – 1995) montrent indépendamment, en 1936, l'indécidabilité de **l'Entscheidungs** problème. Turing propose la machine de Turing comme modèle formel de calcul, et Church le lambda-calcul. Ils énoncent le principe selon lequel tout ce qui est calculable peut être calculé sur un de ces deux modèles ("thèse de Church-Turing").

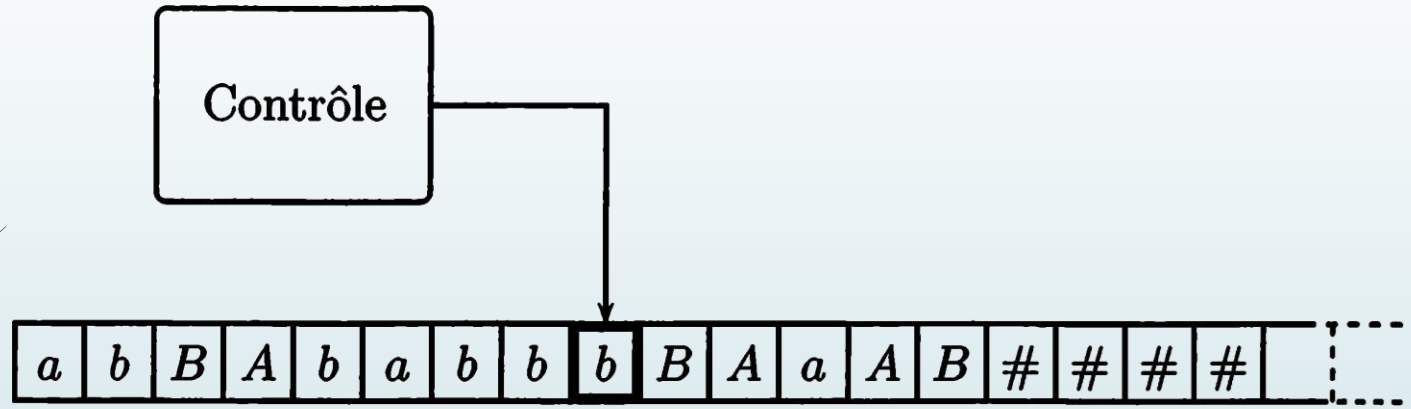


Machine de Turing

Une machine de Turing est un objet mathématique, défini en 1936 par Alan Turing, qui a pour but de décrire ce qu'est un calcul.

http://www.dailymotion.com/video/xrmfie/

Machine de Turing



Une machine de Turing est composée de deux parties:

- Contrôle: constitué de
  - un nombre fini d'états possibles.
  - les transitions qui régissent les calcules.
- Tête de lecture/écriture: pour lire et écrire sur la bande mémoire infinie (bande)

Machine de Turing déterministe

- Une machine de Turing déterministe est composée de:
  - Une mémoire infinie (ruban) divisée en cases. Chaque case peut contenir un symbole de l'alphabet du ruban.
  - Une tête de lecture se déplaçant sur le ruban.
  - Un ensemble fini d'états, dont l'état initial et un ensemble d'états accepteurs.
  - Une fonction de transition qui pour chaque état de la machine et symbole se trouvant sous la tête de lecture précise:
    - L'état suivant
    - Un caractère écrit sur le ruban à la position courante.
    - Un sens de déplacement de la tête de lecture.

Formalisation

Une machine de Turing est formellement décrite par un septuplet

$$M = (Q, \Gamma, \Sigma, \delta, s, B, F),$$

où :

- Q est un ensemble fini d'états,
- Γ est l'alphabet de ruban (l'alphabet utilisé sur le ruban),
- $\Sigma \subseteq \Gamma$ est l'alphabet d'entrée (l'alphabet utilisé pour le mot d'entrée),
- $s \in Q$ est l'état initial,
- $F \subseteq Q$ est l'ensemble des états accepteurs,
- $B \in \Gamma - \Sigma$ est le « symbole blanc » (souvent dénoté #),
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ est la fonction de transition (L et R sont utilisés pour représenter respectivement un déplacement de la tête de lecture vers la gauche (left) et vers la droite (right)).

Principe de fonctionnement

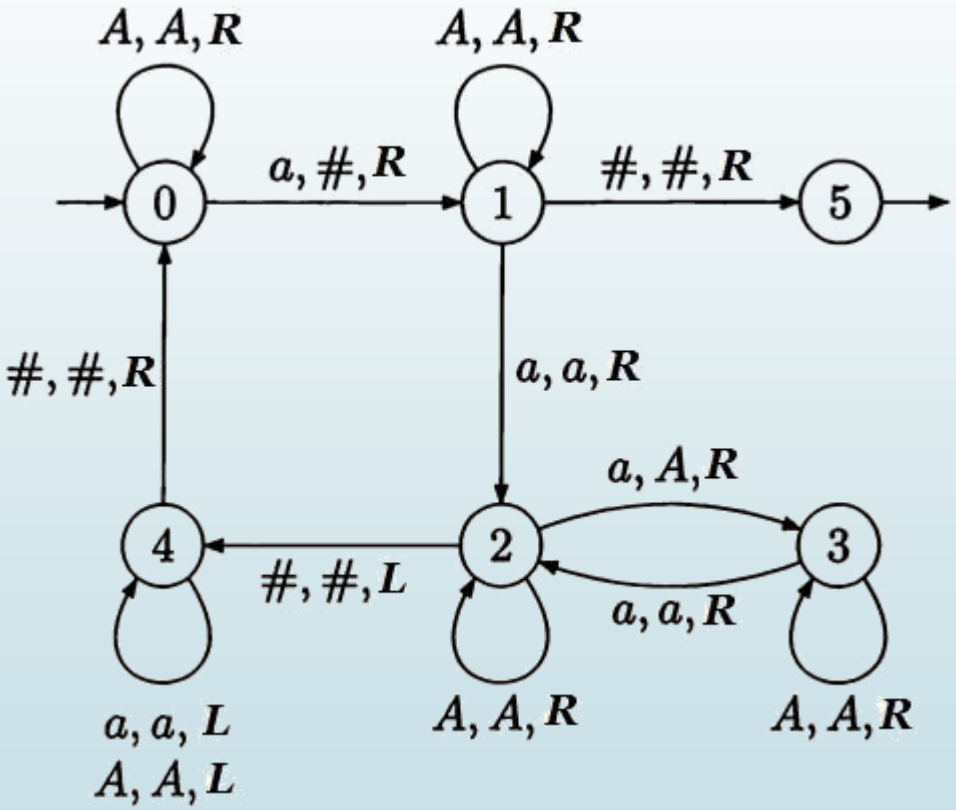
- Le mot d'entrée est écrit sur la bande à partir de la première case (la plus à gauche). Le reste de la bande est rempli par des blancs (#).
- La tête de lecture est placée sur le premier symbole du mot d'entrée.
- Le contrôle de la machine est dans l'état initial.
- La machine exécute son calcul (selon les transitions) jusqu'à éventuellement arriver dans un état acceptant (l'entrée).

Principe de fonctionnement

- Initialement, la tète de lecture est sur la première ce du ruban (la machine se trouve dans son état initial),
- À chaque étape de l exécution :
  - Lit le symbole se trouve sous sa tète de lecture
  - Remplace ce symbole par celui précisé par la fonction de transition
  - Déplace sa tète de lecture selon le sens de fonction de transition
  - Change l état
- Un mot est accepte par la machine lorsque l exécution atteint un état accepteur

Exécution d'une machine de Turing

L'exécution d'une machine de Turing peut être vue comme un automate



Notion de configuration

- Nous avons dit précédemment que:

  transition: état → état

  tel que un état: contient toute l'information nécessaire et suffisante pour prédire son évolution future

  L'état (selon la définition précédente) d'une machine de Turing, n'est pas seulement **l'état de contrôle** mais aussi:

  - **l'information présente sur le ruban, et**
  - **la position de la tête de lecture.**

- On appelle cette information dans le cas d'automates: ***configuration***

Définition d'une configuration

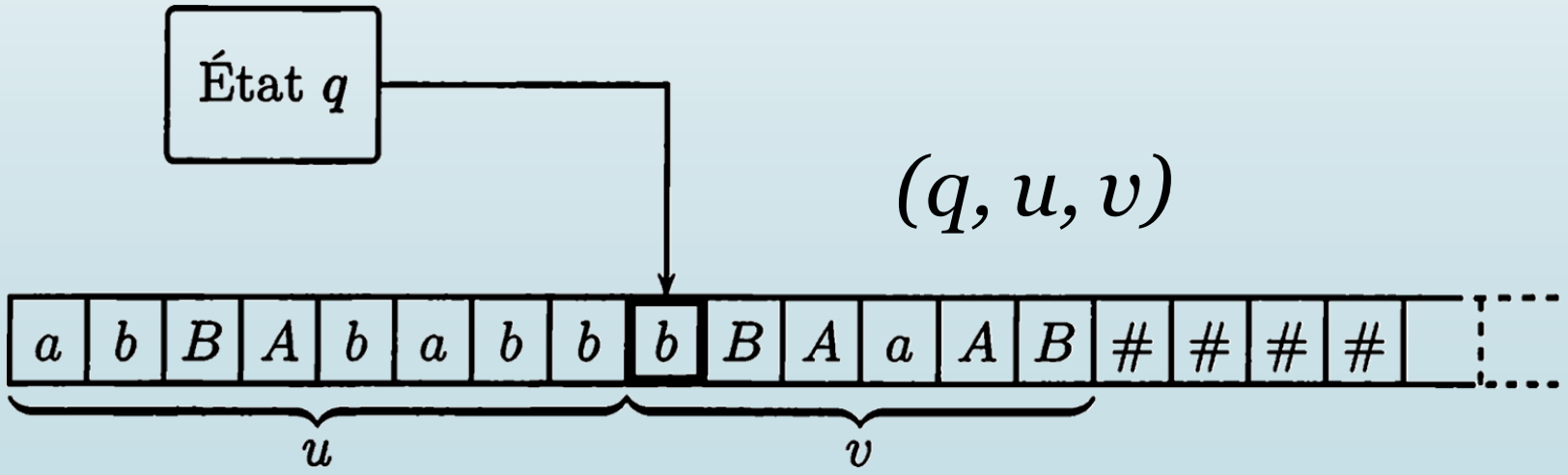
- Le ruban est infini mais à tout moment de l'exécution, seule une partie finie est effectivement utilisée par la machine (symboles non blancs).
- Ainsi une configuration d'une machine de Turing est un triplet :
  - L'état de contrôle de la machine.
  - Le mot apparaissant sur le ruban avant la tête de lecture (strictement à gauche).
  - Le mot se trouvant entre la tête et le dernier caractère non blanc.

Définition d'une configuration

Plus formellement, une configuration est un élément de:

$$Q \times \Gamma^* \times (\varepsilon \cup \Gamma^*(\Gamma - \{B\}))$$

Tel que $\boldsymbol{B} = \#$



Notion de dérivation

- Une exécution d'une machine de Turing (calcul) est une suite d'étapes.
- Une étape de calcul consiste à passer d'une configuration à une autre configuration en appliquant une transition.
- Ce passage est appelé : ***dérivation***.

Dérivation en une étape:

***Définition*:** Soit une configuration (q, α_1, α_2)
Notons cette configuration $(q, \alpha_1, b\alpha_2')$

(***b*** est le caractère sous la tête de lecture)

- Si $\alpha_2 = \varepsilon$ alors $\boldsymbol{b = \#}$.
- Si $\delta(q, b) = (q', b', R)$ alors:

$$(q, \alpha_1, b\alpha_2') \vdash_M (q', \alpha_1 b', \alpha_2').$$

- Si $\delta(q, b) = (q', b', L)$ et $\alpha_1 \neq \varepsilon$ et donc $\alpha_1' a$, alors:

$$(q, \alpha_1' a, b\alpha_2') \vdash_M (q', \alpha_1', ab'\alpha_2')$$

Dérivation en plusieurs étapes:

***Définition*:** une C' est dérivable en plusieurs étapes de la conconfiguration figuration C par la machine M:

M $(C \vdash_M^* C')$ s'il existe $k \geqslant 0$ et des configuration intermédiaires $C_0, C_1, C_2, \ldots, C_k$ telles que:

- $C = C_0$,
- $C' = C_k$,
- $C_i \vdash_M C_{i+1}$ pour $0 \leqslant i < k$.

Définition du langage accepté pat MT

Définition: Le langage *L(M)* accepté par une machine de Turing est l'ensemble des mots w tels que

$$(s, \varepsilon, w) \vdash_M^* (p, \alpha_1, \alpha_2), \text{ avec } p \in F$$

Exemple

$a^n b^n$ $n>0$

La suite des configurations obtenues pour le mot ***aaabbb***

$(q_0, \varepsilon, aaabbb)$	$(q_1, XXXYY, b)$
$(q_1, X, aabbb)$	$(q_2, XXXY, YY)$
$(q_1, Xa, abbb)$	(q_2, XXX, YYY)
(q_1, Xaa, bbb)	$(q_2, XX, XYYY)$
$(q_2, Xa, aYbb)$	(q_0, XXX, YYY)
$(q_2, X, aaYbb)$	$(q_3, XXXY, YY)$
$(q_2, \varepsilon, XaaYbb)$	$(q_3, XXXYY, Y)$
$(q_0, X, aaYbb)$	$(q_3, XXXYYY, \varepsilon)$
$(q_1, XX, aYbb)$	$(q_4, XXXYYY\#, \varepsilon)$

Décider Vs Accepter

Pour une suite de configuration à partir d'une configuration initiale, plusieurs cas peuvent se présenter:

- Elle contient une configuration où l'état est accepteur

  → **Le mot est accepté**

- Elle se termine par une configuration pour laquelle il n'y a pas de configuration suivante, car:
  - La fonction de transition n'est pas définie pour cette transition.
  - La configuration indique un déplacement à gauche alors que la tête de lecture se trouve à la première case.

  → **Le mot est rejeté**

- Une suite infinie de configurations où il n'y a jamais un état acceptant.

  → **Pas de réponse**

Décider Vs Accepter

- Dans les deux premiers cas, nous obtenons une réponse (positive ou négative).
- Dans le dernier cas, on n'obtient aucune réponse et on ne peut jamais confirmer que la machine ne s'arrêtera pas!

- → c'est ce dernier cas qui fait qu'une machine de Turing **acceptant** un langage ne définit pas une procédure effective pour **reconnaitre** ce langage.

Conclusion: Langage décidé par une MT

***Définition*:** Un langage L est décidé par une machine de Turing M si:

- M accepte L,
- M n'a pas d'exécution infinie.

➔Un langage décidé par une machine de Turing peut être calculé par une procédure effective.