M1if09 – Calculabilité & complexité

Sylvain Brandel
2017 – 2018
sylvain.brandel@univ-lyon1.fr

CM 2

MACHINES DE TURING LANGAGES ET FONCTIONS

Les machines de Turing pour reconnaître des langages

- Dans ce contexte, il faut modifier le concept de machine de Turing standard introduit au paragraphe précédent :

  $\rightarrow$ ajouter la notion d'état final.

- Une machine de Turing augmentée avec des états finaux est le <u>sextuplet</u>

  $$M = (K, \Sigma, \Gamma, \delta, q_0, F) \text{ où :}$$

  - $F \subseteq K$ est l'ensemble des états finaux.

(le reste ne change pas)

Les machines de Turing pour reconnaître des langages

- Soit $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ une machine de Turing. Une chaîne $w \in \Sigma^*$ est <u>acceptée</u> par M ssi :

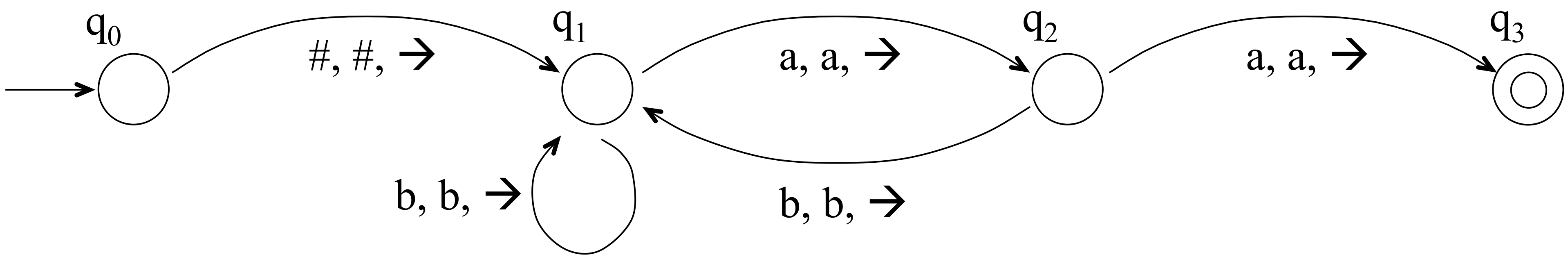
  $(q_0, \underline{\#}w) \vdash_M^* (q_f, w'\underline{a}w'')$ où :

  - $q_f \in F$,
  - $a \in \Gamma$,
  - $w', w'' \in \Gamma^*$,
  - $\delta(q_f, a)$ n'est pas défini.

- Le <u>langage accepté</u> par M, noté L(M), est l'ensemble de toutes les chaînes acceptées par M.

Les machines de Turing pour reconnaître des langages

- Exemple



- La suite de configurations associée au mot aabb est :

$$(q_0, \underline{\#}aabb) \vdash_M (q_1, \#\underline{a}abb) \vdash_M (q_2, \#a\underline{a}bb) \vdash_M (q_3, \#aa\underline{b}b)$$

  Comme l'état q_3 est final, et qu'il n'y a pas de transition depuis q_3, le mot w = aabb est accepté.

- Pour tout mot w ne contenant par aa, le calcul s'arrête sur le premier # à droite de w sur le ruban dans un état non final.

Les machines de Turing pour reconnaître des langages

- Le langage accepté par une machine de Turing est dit <u>Turing-acceptable</u> ou <u>récursivement énumérable</u>.

- Si la machine de Turing s'arrête sur toutes les entrées possibles (c-à-d pour tous les mots w, $w \in L$ ou $w \notin L$), alors le langage est dit <u>Turing-décidable</u> ou <u>récursif</u>.

  On dit que M <u>semi-décide</u> L, ou encore M <u>accepte</u> L.

- On a alors :
  - $\forall\, w \in L,\ (q_0, \underline{\#}w) \vdash_M^* (q_f, \underline{\#}Y)\ (q_f \in F) \rightarrow$ YES (accepté)
  - $\forall\, w \notin L,\ (q_0, \underline{\#}w) \vdash_M^* (q_f, \underline{\#}N)\ (q_f \in F) \rightarrow$ NO (rejeté)

Les machines de Turing pour reconnaître des langages

- Autre formulation : Une machine de Turing est le <u>septuplet</u> (?)

$$M = (K, \sum, \Gamma, \delta, q_0, q_{accept}, q_{reject}) \text{ où :}$$

  - $q_{accept} \in K$ est l'état d'acceptation de l'entrée,
  - $q_{reject} \in K$ est l'état de rejet de l'entrée.

(le reste ne change pas)

Les machines de Turing pour calculer des fonctions

- L'idée est d'utiliser les machines de Turing pour calculer des fonctions de chaînes vers chaînes.

- Soient
  - $\sum_0$ et $\sum_1$ deux alphabets ne contenant pas le symbole blanc (#),
  - f une fonction de $\sum_0{}^*$ vers $\sum_1{}^*$.
- Une machine de Turing M = (K, $\sum_0$, Γ, δ, q_0, F) calcule la fonction f ssi :

$\forall$ w $\in \sum_0{}^*$ tel que f(w) = u, on a

$(q_0, \underline{\#}w) \vdash_M^* (q_f, \underline{\#}u)$ où :

$q_f \in F$,

$\delta(q_f, \#)$ n'est pas défini.

Les machines de Turing pour calculer des fonctions

- Lorsqu'une telle machine de Turing existe, la fonction est dite <u>Turing-calculable</u>.

- La notion de Turing-calculable n'est pas restreinte aux fonctions de chaînes vers chaînes.

  → elle peut être étendue de plusieurs façons :

  - Nombre quelconque d'arguments
  - Pour des fonctions de **N** dans **N**

Les machines de Turing pour calculer des fonctions

- Fonctions avec un nombre quelconque d'arguments de la forme $f : (\Sigma_0^*)^k \rightarrow \Sigma_1^*$:

- Une machine de Turing $M = (K, \Sigma_0, \Gamma, \delta, q_0, F)$ calcule la fonction f ssi :
  - $\forall\ \sigma_1, \sigma_2, \ldots, \sigma_k \in \Sigma_0^*$ tels que $f(\sigma_1, \sigma_2, \ldots, \sigma_k) = u$, on a :

$$(q_0, \underline{\#}\sigma_1\#\sigma_2\#\ldots\#\sigma_k) \vdash_M^* (q_f, \underline{\#}u) \text{ où :}$$

$$q_f \in F,$$

$$\delta(q_f, \#) \text{ n'est pas défini.}$$

Les machines de Turing pour calculer des fonctions

- Fonctions de N dans N :

- Notons I un symbole fixé différent de #.
  - Tout entier naturel n peut être représenté par la chaîne I^n en notation unaire
  - (dans ce cas l'entier zéro est représenté par la chaîne vide)

- Une fonction $f : N \rightarrow N$ est calculée par une machine de Turing M, si M calcule la fonction $f' : \{I\}^* \rightarrow \{I\}^*$ telle que $f'(I^n) = I^{f(n)}$ pour tout $n \in N$.

Les machines de Turing pour calculer des fonctions

- La fonction successeur définie par succ(n) = n + 1, $\forall n \geq 0$, est calculée par la machine de Turing

$$M = (K, \Sigma, \Gamma, \delta, q_0, F) \text{ où :}$$

  - $K = \{q_0, q_1, q_2\}$,
  - $\Sigma = \{I\}$,
  - $\Gamma = \{I, \#\}$,
  - $F = \{q_2\}$

δ	#	I
q_0	$(q_1, \#, \rightarrow)$	
q_1	$(q_2, I, \leftarrow)$	$(q_1, I, \rightarrow)$
q_2		$(q_2, I, \leftarrow)$

~~Combinaison des machines de Turing~~

- machine de Turing = module ou sous-routine
- Ces modules peuvent être connectés entre eux en utilisant des règles de combinaisons :

$$\begin{tikzcd} \arrow[r] & M_1 \arrow[rr, bend left] & & M_2 \end{tikzcd} \qquad \begin{tikzcd} \arrow[r] & M_1 \arrow[rr, bend left, "a"] \arrow[d, "b"] & & M_2 \\ & M_3 & & \end{tikzcd}$$

- Les flèches ne représentent pas des transitions mais des branchements, conditionnels ou non

Extensions des machines de Turing

- Est-il possible d'accroitre la puissance des machines de Turing ?
- Examinons des extensions :

  (a) un ruban infini dans les deux directions,

  (b) plusieurs rubans,

  (c) plusieurs têtes sur le ruban,

  (d) un ruban bidimensionnel,

  (e) le non-déterminisme,

  (f) l'accès aléatoire.
- Et montrons que ces machines étendues peuvent être simulées par des machines standard

Extensions des machines de Turing

- Pour chaque type d'extension, nous montrons que l'opération de la machine étendue peut être simulée par une machine de Turing normale.
- La démonstration consiste dans chaque cas :
  (1) à montrer comment construire une machine normale à partir de la machine étendue considérée,
  (2) à prouver que la machine normale construite simule correctement le comportement de la machine de départ.

Machine de Turing à ruban infini dans les deux sens

- Soit une machine de Turing $M = (K, \Sigma, \Gamma, \delta, q_0, F)$ dont le ruban n'a pas de borne à gauche :
  - la chaîne d'entrée peut se trouver n'importe où sur le ruban,
  - la tête pointe sur le premier blanc à gauche de la chaîne.
- Dans cette machine, une configuration est de la forme : $(q, w\underline{a}u)$ avec $q \in K$, $w, u \in \Gamma^*$, $a \in \Gamma$, où :
  - w ne commence pas par un blanc
  - u ne finit pas par un blanc.
- On étend la relation entre configurations pour prendre en compte les déplacements à gauche :
  - si $\delta(q, a) = (p, b, G)$ alors $(q, \underline{a}u) \vdash_M (p, \underline{\#}bu)$.

Machine de Turing à ruban infini dans les deux sens

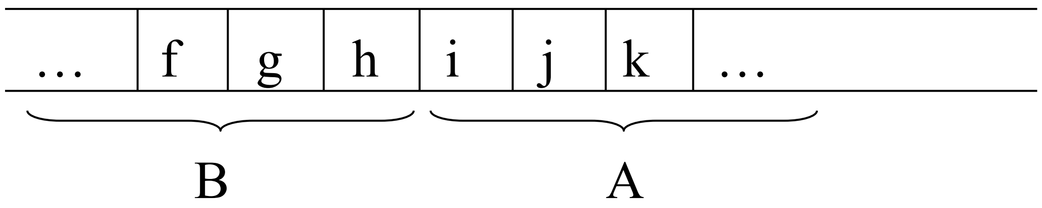
- Montrons qu'une machine M avec ruban infini dans les deux sens n'est <u>pas plus puissante</u> qu'une machine normale (dans le sens qu'elle ne permet pas de reconnaître plus de langages, ou calculer plus de fonctions).
- Pour cela montrons comment construire une machine M' $= (K', \Sigma, \Gamma', \delta', q_0', F')$, à partir de M et qui <u>simule</u> M :
  - si M s'arrête sur un mot w, alors M' s'arrête sur ce même mot w,
  - si M ne s'arrête pas sur un mot w, alors M' ne s'arrête pas non plus sur ce même mot w.

Machine de Turing à ruban infini dans les deux sens

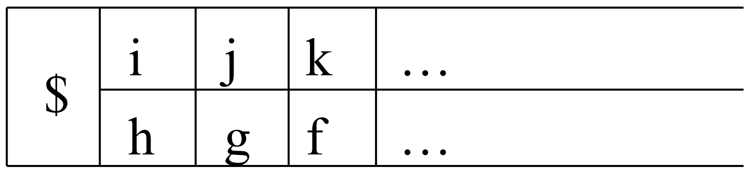
- Pour simuler le ruban doublement infini de M dans celui de M', on définit pour M' un ruban à <u>2 pistes</u> :
- Ce ruban est obtenu en coupant en 2 celui de M de façon arbitraire.

- <u>Exemple</u>

Ruban de M :



Ruban de M' :



Principe de la simulation au tableau…