

Cours 1': Algorithmes. Modèles de calcul.

Olivier Bournez

bournez@lix.polytechnique.fr
LIX, Ecole Polytechnique

Au menu

Quelques modèles séquentiels et leur équivalence

Forme générale des modèles

- On va considérer des modèles qui correspondent à des programmes du type

```
ctl_state ←  $q_0$   
repeat  
  seq <instructions>  
  if ctl_state =  $q_a$  then out ← vrai  
  if ctl_state =  $q_r$  then out ← faux  
  endseq  
until out ≠ undef  
write out
```

où <instructions> est une suite finie d'instructions parmi un type d'instructions autorisées.

Sous menu

Quelques modèles séquentiels et leur équivalence

Machines de Turing sur un alphabet fini

Machines de Turing sur une structure \mathfrak{M}

Machines à k piles

Machines à k compteurs

Machines RAM

Algorithme versus Machine de Turing

Résumé

Machine de Turing à k rubans sur un alphabet M :

- Les instructions autorisées sont du type :

```
if  $ctl\_state=q$  and  $tape_\ell(head_\ell) = a$  then  
par  
   $tape_{\ell'}(head_{\ell'}) \leftarrow a'$   
   $head_{\ell'} \leftarrow head_{\ell'} + m$   
   $ctl\_state \leftarrow q'$   
endpar
```

où $m \in \{-1, 0, 1\}$, $a' \in M$, $\ell' \in \{1, 2, \dots, k\}$.

- Ici
 - ▶ $head_1, \dots, head_k$ sont des symboles de constantes à valeurs entières, qui codent la position de chacune des têtes de lecture.
 - ▶ $tape_1, \dots, tape_k$ sont des symboles de fonctions d'arité 1, qui codent le contenu de chacun des k rubans.

Vision alternative (classique)

- Une machine de Turing à k -rubans est un 8-uplet

$$M = (Q, \Sigma, \Gamma, \mathbf{B}, \delta, q_0, q_a, q_r)$$

où

1. Q est l'ensemble fini des états.
2. Σ est un alphabet *fini*.
3. Γ est l'alphabet de travail *fini* : $\Sigma \subset \Gamma$.
4. $\mathbf{B} \in \Gamma$ est le caractère blanc.
5. $q_0 \in Q$ est l'état initial.
6. $q_a \in Q$ est l'état d'acceptation.
7. $q_r \in Q$ est l'état de rejet (ou d'arrêt).
8. δ est la fonction de transition : δ est constitué d'une fonction δ_1 de Q dans $\{1, 2, \dots, k\}$, et d'une fonction δ_2 de $Q \times \Gamma$ dans $Q \times \{1, 2, \dots, k\} \times \Gamma \times \{\leftarrow, |, \rightarrow\}$.

- Une configuration est donnée par la description des rubans, par les positions des têtes de lecture/écriture, et par l'état interne :
 - ▶ elle peut se noter $C = (q, u_1 \# v_1, \dots, u_k \# v_k)$, avec $u_1, \dots, u_n, v_1, \dots, v_n \in (\Gamma - \{\mathbf{B}\})^*$, $q \in Q$: u_i et v_i désignent le contenu respectivement à gauche et à droite de la tête de lecture du ruban i , la tête de lecture du ruban i étant sur la première lettre de v_i , et u_i écrit de droite à gauche.

- Une telle configuration est dite *acceptante* si $q = q_a$, *rejetante* si $q = q_r$.

- Pour $w \in \Sigma^*$, la configuration initiale correspondante à w est la configuration $C[w] = (q_0, \#w, \#, \dots, \#)$.

- On note : $C \vdash C'$ si la configuration C' est le successeur direct de la configuration C par le programme (donné par δ) de la machine de Turing.

Formellement, si $C = (q, u_1 \# v_1, \dots, u_k \# v_k)$, et si a_1, \dots, a_k désignent les premières lettres de v_1, \dots, v_k , et si

$$\delta_1(q) = \ell$$

$$\delta_2(q, a_\ell) = (q', \ell', a', m')$$

alors $C \vdash C'$ si

- $C' = (q', u'_1 \# v'_1, \dots, u'_k \# v'_k)$, et
- pour $i \neq \ell'$, $u'_i = u_i$, $v'_i = v_i$
- si $i = \ell'$,
 - ▶ si $m' = |$, $u'_i = u_i$, et v'_i est obtenu en remplaçant la première lettre a_i de v_i par a' .
 - ▶ si $m' = \leftarrow$, $v'_i = a' v_i$, et u'_i est obtenu en supprimant la première lettre de u_i .
 - ▶ si $m' = \rightarrow$, $u'_i = u_i a'$, et v'_i est obtenu en supprimant la première lettre a_i de v_i .

- On appelle *calcul de M sur un mot $w \in \Sigma^*$* , une suite de configurations $(C_i)_{i \in \mathbb{N}}$ telle que $C_0 = C[w]$ et pour tout i , $C_i \vdash C_{i+1}$.
- Le mot w est dit *accepté* (resp. *refusé*) si le calcul sur ce mot est tel qu'il existe un entier t avec C_t acceptante (resp. rejetante).
- Un langage $L \subset \Sigma^*$ est dit *accepté par M* si pour tout $w \in \Sigma^*$, $w \in L$ si et seulement si w est accepté.
- Un langage $L \subset \Sigma^*$ est dit *décidé par M* si pour tout $w \in \Sigma^*$, $w \in L$ si et seulement si w est accepté, et $w \notin L$ si et seulement si w est rejeté.

Sous menu

Quelques modèles séquentiels et leur équivalence

Machines de Turing sur un alphabet fini

Machines de Turing sur une structure \mathfrak{M}

Machines à k piles

Machines à k compteurs

Machines RAM

Algorithme versus Machine de Turing

Résumé

Machine de Turing à k rubans sur une structure

$$\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$$

- Les instructions autorisées sont du type :

```
if  $ctl\_state=q$  and  
     $r_j(tape_\ell(head_\ell), tape_\ell(head_\ell + 1), \dots, tape_\ell(head_\ell + k))$  then  
par  
     $tape_{\ell'}(head_{\ell'}) \leftarrow$   
         $f_i(tape_\ell(head_\ell), tape_\ell(head_\ell + 1), \dots, tape_\ell(head_\ell + k'))$   
     $head_{\ell'} \leftarrow head_{\ell'} + m$   
     $ctl\_state \leftarrow q'$   
endpar
```

où $m \in \{-1, 0, 1\}$, $\ell' \in \{1, 2, \dots, k\}$.

- ▶ $head_1, \dots, head_k$ sont des symboles de la signature d'arité 0 à valeurs entières, qui codent la position de chacune des têtes de lecture.
- ▶ $tape_1, \dots, tape_k$ sont des symboles de fonctions d'arité 1, qui codent le contenu de chacun des k rubans : le contenu de chaque case est un élément de M .

Sous menu

Quelques modèles séquentiels et leur équivalence

Machines de Turing sur un alphabet fini

Machines de Turing sur une structure \mathfrak{M}

Machines à k piles

Machines à k compteurs

Machines RAM

Algorithme versus Machine de Turing

Résumé

Machines à k piles sur un alphabet fini M :

- Les instructions autorisées sont du type :

1. $push_i(a)$
2. pop_i
3. **if** $top_i = a$ **then** $ctl_state := q$ **else** $ctl_state := q'$

où

- ▶ $i \in \{1, 2, \dots, k\}$,
- ▶ a est un symbole de l'alphabet M ,
- ▶ $push_i(a)$, pop_i , top_i désignent respectivement
 - empiler le symbole a sur la pile i ,
 - dépiler le sommet de la pile i ,
 - et le symbole au sommet de la pile i .

Theorem

Toute machine de Turing à k rubans peut être simulée par une machine à $2k$ piles.

Principe :

- la machine utilise 2 piles par ruban : une pile pour le contenu à droite, écrit de gauche à droite, et une pile pour le contenu à gauche, écrit de droite à gauche.
- Par exemple, se déplacer à droite correspond à dépiler le symbole de la pile “contenu à droite”, et à l’empiler sur la pile “contenu à gauche”.

Sous menu

Quelques modèles séquentiels et leur équivalence

Machines de Turing sur un alphabet fini

Machines de Turing sur une structure \mathfrak{M}

Machines à k piles

Machines à k compteurs

Machines RAM

Algorithme versus Machine de Turing

Résumé

Machines à k compteurs

■ Les instructions autorisées sont du type :

1. $x_i \leftarrow x_i + 1$
2. $x_i \leftarrow x_i \ominus 1$
3. **if** $x_i = 0$ **then** $ctl_state := q$ **else** $ctl_state := q'$

où

- ▶ chacun des registres x_i contient un entier naturel.
- ▶ $i \in \{1, 2, \dots, k\}$,
- ▶ et $x \ominus 1$ vaut $x - 1$ si $x \neq 0$, et 0 pour $x = 0$.

Theorem

Toute machine à k -piles sur un alphabet fini peut être simulée par une machines à $k + 1$ compteurs.

Principe :

- voir une pile (donc un mot) $w = a_1 a_2 \cdots a_n$ sur l'alphabet $\Sigma = \{0, 1, \dots, r - 1\}$ comme l'entier $i = a_n r^{n-1} + a_{n-1} r^{n-2} + \cdots + a_2 r + a_1$.
- Dépiler correspond à remplacer i par $i \text{ div } r$. Empiler le symbole a correspond à remplacer i par $i * r + a$. Lire le sommet d'une pile i correspond à calculer $i \text{ mod } r$.
- Par exemple, pour $i \text{ div } r$: en partant avec le compteur supplémentaire (celui d'indice $k + 1$) à 0, on décrémente le compteur i de r et on incrémente le compteur supplémentaire de 1. On répète cette opération jusqu'à ce que le compteur i atteigne 0. On décrémente alors le compteur supplémentaire de 1 en incrémentant le compteur i de 1 jusqu'à ce que le premier soit 0.

Theorem

Toute machine à $k \geq 3$ compteurs se simule par une machine à 2 compteurs.

Principe :

- Supposons $k = 3$. L'idée est coder trois compteurs i, j et k par l'entier $m = 2^i 3^j 5^k$. L'un des compteurs stocke cet entier. L'autre compteur est utilisé pour faire des multiplications, divisions, calculs modulo m , pour m valant 2, 3, ou 5, comme dans la preuve précédente.
- Pour $k > 3$, on utilise le même principe, mais avec les k premiers nombres premiers.

Sous menu

Quelques modèles séquentiels et leur équivalence

Machines de Turing sur un alphabet fini

Machines de Turing sur une structure \mathfrak{M}

Machines à k piles

Machines à k compteurs

Machines RAM

Algorithme versus Machine de Turing

Résumé

Machines RAM/RISC

■ Les instructions autorisées sont du type :

1. $x_0 \leftarrow 0$
2. $x_0 \leftarrow x_0 + 1$
3. $x_0 \leftarrow x_0 \ominus 1$
4. **if** $x_0 = 0$ **then** $ctl_state \leftarrow j$
5. $x_0 \leftarrow x_i$
6. $x_i \leftarrow x_0$
7. $x_0 \leftarrow x_{x_i}$
8. $x_{x_0} \leftarrow x_i$

où

- ▶ les x_i sont des registres à valeurs entières.

Theorem

Toute machine RAM/RISC peut être simulée par une machine de Turing.

- La machine de Turing possède 4 rubans. Les deux premiers rubans codent les couples (i, x_i) pour x_i non nul. Le troisième ruban code l'accumulateur x_0 et le quatrième est un ruban de travail.
- Plus concrètement, le premier ruban code un mot de la forme

$$\dots \mathbf{BB} \langle i_0 \rangle \mathbf{B} \langle i_1 \rangle \dots \mathbf{B} \dots \langle i_k \rangle \mathbf{BB} \dots$$

- Le second ruban code un mot de la forme

$$\dots \mathbf{BB} \langle x_{i_0} \rangle \mathbf{B} \langle x_{i_1} \rangle \dots \mathbf{B} \dots \langle x_{i_k} \rangle \mathbf{BB} \dots$$

- Les têtes de lecture des deux premiers rubans sont sur le deuxième **B**.
- Le troisième ruban code $\langle x_0 \rangle$, la tête de lecture étant tout à gauche. On appelle *position standard* une telle position des têtes de lecture.
- Au départ, la donnée du programme RAM est copiée sur le second ruban, et **0** est placé sur le ruban 1 signifiant que x_0 contient la donnée du programme.

La simulation est décrite pour trois exemples.

■ $x_0 \leftarrow x_0 + 1$:

- ▶ on déplace la tête de lecture du ruban 3 tout à droite jusqu'à atteindre un symbole **B**.
- ▶ On se déplace alors d'une case vers la gauche, et on remplace les **1** par des **0**, en se déplaçant vers la gauche tant que possible.
- ▶ Lorsqu'un **0** ou un **B** est trouvé, on le change en **1** et on se déplace à gauche pour revenir en position standard.

■ $x_{23} \leftarrow x_0$:

- ▶ on parcourt les rubans 1 et 2 vers la droite, bloc délimité par **B** par bloc, jusqu'à atteindre la fin du ruban 1, ou ce que l'on lise un bloc **B10111B** (**10111** correspond à 23 en binaire).
- ▶ Si la fin du ruban 1 a été atteinte, alors l'emplacement 23 n'a jamais été vu auparavant. On l'ajoute en écrivant **10111** à la fin du ruban 1, et on recopie le ruban 3 (la valeur de x_0) sur le ruban 2. On retourne alors en position standard.
- ▶ Sinon, c'est que l'on a trouvé **B10111B** sur le ruban 1. On lit alors $\langle x_{23} \rangle$ sur le ruban 2. Dans ce cas, il doit être modifié. On fait cela de la façon suivante :
 1. On copie le contenu à droite de la tête de lecture numéro 1 sur le ruban 4.
 2. On copie le contenu du ruban 3 (la valeur de x_0) à la place de x_{23} sur le ruban 2.
 3. On écrit **B**, et on recopie le contenu du ruban 4 à droite de la tête de lecture du ruban 2, de façon à restaurer le reste du ruban 2.
 4. On retourne en position standard.

■ $x_0 \leftarrow x_{x_{23}}$:

- ▶ En partant de la gauche des rubans 1 et 2, on parcourt les rubans 1 et 2 vers la droite, bloc délimité par **B** par bloc, jusqu'à atteindre la fin du ruban 1, ou ce que l'on lise un bloc **B10111B** (10111 correspond à 23 en binaire).
- ▶ Si la fin du ruban 1 a été atteinte, on ne fait rien, puisque x_{23} vaut 0 et le ruban 3 contient déjà $\langle x_0 \rangle$.
- ▶ Sinon, c'est que l'on a trouvé **B10111B** sur le ruban 1. On lit alors $\langle x_{23} \rangle$ sur le ruban 2, que l'on recopie sur le ruban 4. Comme ci-dessus, on parcourt les rubans 1 et 2 en parallèle jusqu'à trouver **B** $\langle x_{23} \rangle$ **B** où atteindre la fin du ruban 1. Si la fin du ruban 1 est atteinte, alors on écrit **0** sur le ruban 3, puisque $x_{x_{23}} = x_0$. Sinon, on copie le bloc correspondant sur le ruban 1 sur le ruban 3, puisque le bloc sur le ruban 2 contient $x_{x_{23}}$, et on retourne en position standard.

Sous menu

Quelques modèles séquentiels et leur équivalence

Machines de Turing sur un alphabet fini

Machines de Turing sur une structure \mathfrak{M}

Machines à k piles

Machines à k compteurs

Machines RAM

Algorithme versus Machine de Turing

Résumé

Theorem

Tout algorithme sur une structure $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$ peut se simuler par machine de Turing sur la structure \mathfrak{M} .

Principe :

- Nous savons qu'à un algorithme est associé un ensemble fini T de termes critiques
- Vu le théorème de forme normale, pour simuler un tel algorithme, il suffit d'être capable d'évaluer chacun des emplacements (f, \bar{m}) , pour f un symbole de fonction de la structure.
- On utilise le principe de la preuve précédente, en utilisant 2 rubans par tel symbole f d'arité ≥ 1 (comme pour le symbole x dans la preuve de ce théorème), un ruban par symbole d'arité 0 (comme pour le symbole x_0), en plus de rubans de travail en nombre fini.
- Pour un symbole f d'arité $r \geq 1$, les deux rubans codent des mots de la forme

$$\dots \mathbf{B}\mathbf{B}\bar{a}_0\mathbf{B}\bar{a}_1\mathbf{B}\dots\mathbf{B}\dots\bar{a}_k\mathbf{B}\mathbf{B}\dots$$

Le second ruban code un mot de la forme

$$\dots \mathbf{B}\mathbf{B}[[f, \bar{a}_0]]\mathbf{B}[[f, \bar{a}_1]]\mathbf{B}\dots[[f, \bar{a}_k]]\mathbf{B}\mathbf{B}\dots$$

où \bar{a}_i désigne un r -uplet d'éléments de M , et $[[f, \bar{a}_i]]$ le contenu de l'emplacement (f, \bar{m}) (et donc désigne un élément de M).

- Comme dans la preuve précédente, tout emplacement qui n'est pas dans la liste codée par ces deux rubans correspond à une valeur indéfinie, car jamais encore accédée.

Sous menu

Quelques modèles séquentiels et leur équivalence

Machines de Turing sur un alphabet fini

Machines de Turing sur une structure \mathfrak{M}

Machines à k piles

Machines à k compteurs

Machines RAM

Algorithme versus Machine de Turing

Résumé

Sur un alphabet fini

Theorem

Sur une structure finie, ou sur un alphabet fini, les modèles suivants se simulent deux à deux

1. *Les machines de Turing*
2. *Les machines à $k \geq 2$ piles*
3. *Les machines à compteurs*
4. *Les machines à 2 compteurs*
5. *Les machines RAM*
6. *Les algorithmes*

Sur une structure arbitraire

Theorem

Sur une structure arbitraire $\mathfrak{M} = (M, f_1, \dots, f_u, r_1, \dots, r_v)$, les modèles suivants se simulent deux à deux

- 1. Les machines de Turing*
- 2. Les machines à $k \geq 2$ piles*
- 3. Les machines RAM*
- 4. Les algorithmes*