

TP n°2: Généralités sur les automates

Application aux équations diophantiennes

BATOG Guillaume

19 octobre 2006

1 Implémentation des automates déterministes complets

Un automate est une structure de données $A = \langle Q, \Sigma, \delta, F, I \rangle$ où Q est un ensemble fini d'états, $F \subset Q$ est l'ensemble des états finaux, $I \subset Q$ est l'ensemble des états initiaux, Σ est un alphabet fini et δ est l'ensemble des transitions de l'automate, ie un sous-ensemble de $Q \times \Sigma \times Q$. On étend la relation δ sur $Q \times \Sigma^*$ par la relation δ^* définie par récurrence par :

- $\forall q, q' \in Q, (q, \varepsilon, q') \in \delta^* \Leftrightarrow q = q'$,
- $\forall a \in \Sigma, \forall u \in \Sigma^*, \forall q, q' \in Q,$
 $(q, ua, q') \in \delta^* \Leftrightarrow \exists q'' \in Q, (q, u, q'') \in \delta^* \wedge (q'', a, q) \in \delta.$

Un mot $u \in \Sigma^*$ est accepté par l'automate A si et seulement s'il existe $q \in I$ et $q' \in F$ tels que $(q, u, q') \in \delta^*$. Le langage des mots acceptés par l'automate A noté $\mathcal{L}(A)$ est l'ensemble des mots acceptés par A .

Un automate est dit déterministe si et seulement s'il possède un unique état initial et si δ définit une fonction de $Q \times \Sigma$ sur Q . Un automate déterministe est dit complet si et seulement si δ définit une application de $Q \times \Sigma$ sur Q . Pour un automate déterministe, on définit δ^* l'extension de δ sur $Q \times \Sigma^*$ par

- $\forall q \in Q, \delta^*(q, \varepsilon) = q,$
- $\forall q \in Q, \forall a \in \Sigma, \forall u \in \Sigma^*, \delta^*(q, ua) = \delta(\delta^*(q, u), a).$

On note alors de façon plus concise $\delta^*(q, u) = q \cdot u$.

Un état q' est dit accessible à partir d'un état q si et seulement s'il existe un mot $u \in \Sigma^*$ tel que $q \cdot u = q'$ ($(q, u, q') \in \delta^*$) si et seulement si q est co-accessible à partir d'un état q' . Un état est dit accessible si et seulement s'il est accessible à partir d'un état initial. On note $\text{Acc}(A)$ l'ensemble des états accessibles de A . On note également $A[q]$ l'automate A dont q est l'unique état initial. On implémente ici les automates déterministes complets sous Caml à l'aide du type enregistrement suivant :

```
type automate = { etats : int ; delta : int vect vect ;
                  final : bool vect ; initial : int };;
```

Exercice 1 [Construction d'exemples pour les tests]

Dans toute la suite, on considère la suite $(u_i)_{i \geq 0}$ définie par récurrence par $u_0 = 5$ et $u_{i+1} = 3321 \cdot u_i + 5701$. Soit N un entier. On définit l'automate déterministe complet $A_N = \langle \{0, \dots, N-1\}, \{0, 1\}, \delta, F, 0 \rangle$ par

$$q \cdot a = u_{2q+a} \pmod N$$

$$q \in F \text{ ssi } (u_q \pmod N) \leq \frac{N}{3}.$$

L'automate B_N est obtenu de la même manière en prenant $u_0 = 16$.

a) Construire les automates A_N et B_N avec $N = 19, 97$ et 503 .

```
val automateTest : int -> int -> automate = <fun>
```

- b) Calculer le cardinal des ensembles F , $U = \{q \in Q \mid \exists a \in \{0, 1\}, q \cdot a \in F\}$, $V = \{q \in Q \mid \forall a \in \{0, 1\}, q \cdot a \in F\}$ pour les automates de la question a).

```
val cardF, cardU, cardV : automate -> int : <fun>
```

Exercice 2 [Mots de longueur n acceptés par un automate]

- a) Soit $H(q, n)$ le nombre de mots de longueur n acceptés par $A[q]$, trouver une relation de récurrence entre les $H(q, i)$. En déduire le nombre de mots de longueur $n = 0, 1, 2, 3, 4$ et 20 acceptés par les automates de l'exercice 1.

```
val hqn : automate -> int -> hMatrice : <fun>
```

- b) On considère que les mots sont triés par leur longueur et que deux mots de même longueur sont triés par ordre lexicographique où $0 < 1$. Calculer la longueur du 10^e , 100^e , 1000^e et 10000^e mot acceptés par les automates de l'exercice 1.

```
val longueurNieme : hMatrice -> int -> int : <fun>
```

2 Opérations sur les automates

Exercice 3 [Complément, union, intersection]

- a) Écrire une fonction Caml `complement` qui retourne le complément d'un automate déterministe complet. Quel en est la complexité? Répondre à la question b) de l'exercice 1 pour les compléments.

```
val complement : automate -> automate = <fun>
```

- b) Écrire une fonction Caml `intersection` qui retourne l'automate reconnaissant le langage intersecté de deux automates donnés. Quel en est la complexité? Répondre à la question b) de l'exercice 1 pour les automates $A_N \cap B_N$ pour $N = 19, 97$ et 503 .

```
val intersection : automate -> automate -> automate = <fun>
```

- c) Même question pour l'union.

```
val union : automate -> automate -> automate = <fun>
```

Exercice 4 [Partie accessible d'un automate]

L'ensemble des états accessibles d'un automate déterministe complet est caractérisé par $\text{Acc}(A) = \{q \in Q \mid \exists u \in \Sigma^*, q_0 \cdot u = q\}$. On maintient un ensemble R d'états accessibles réduit à $\{q_0\}$ initialement, puis on ajoute les états $q \cdot a$ pour toute lettre a et tout $q \in R$ jusqu'à obtenir un ensemble clos par transition.

- a) Écrire une fonction Caml `etatsAcc` qui retourne la liste des états accessibles d'un automate déterministe complet.

```
val etatsAcc : automate -> int list = <fun>
```

- b) Calculer le cardinal des ensembles $\text{Acc}(A_N)$, $\text{Acc}(B_N)$, $\text{Acc}(\overline{A_N})$, $\text{Acc}(\overline{B_N})$, $\text{Acc}(A_N \cup B_N)$, $\text{Acc}(A_N \cap B_N)$ pour $N = 19, 97$ et 503 .

- c) On note $\text{Réduit}(A)$ l'automate A réduit aux états $\text{Acc}(A)$. Montrer que $\mathcal{L}(\text{Réduit}(A)) = \mathcal{L}(A)$ et que $\text{Réduit}(A)$ est un automate déterministe complet. Écrire une fonction Caml `reduit` calculant le réduit $\text{Réduit}(A)$ d'un automate A . Quel est la complexité de l'algorithme?

```
val réduit : automate -> automate = <fun>
```

- d) Répondre à la question b) de l'exercice 1 pour les automates $\text{Réduit}(A_N)$, $\text{Réduit}(B_N)$, $\text{Réduit}(\overline{A_N})$, $\text{Réduit}(\overline{B_N})$, $\text{Réduit}(A_N \cup B_N)$, $\text{Réduit}(A_N \cap B_N)$ pour $N = 19, 97$ et 503 .

3 Équations diophantiennes

Exercice 5 [Codage d'un n -uplet d'entiers]

Soit $\sigma = \sigma_0 \dots \sigma_{k-1}$ un mot binaire (ie $\sigma \in \{0, 1\}^*$). On définit $\text{Entier}(\sigma)$ comme étant l'entier dont la représentation binaire est σ où σ_0 est le bit de poids le plus faible. Par extension, on définit la fonction $\text{Entier}_n : \{0, 1\}^* \rightarrow \mathbb{N}^n$ par $\text{Entier}_n(\sigma) = (x_0, \dots, x_{n-1})$ avec

$$x_u = \sum_{i, in+u < k} \sigma_{in+u} \cdot 2^i.$$

Par exemple, pour $\sigma = 1011000010$, $\text{Entier}(\sigma) = 269$, $\text{Entier}_2(\sigma) = (19, 2)$, $\text{Entier}_3(\sigma) = (3, 0, 5)$, $\text{Entier}_4(\sigma) = (5, 0, 1, 1)$.

- a) Écrire une fonction Caml `entier` qui à un entier n et un mot binaire σ retourne le n -uplet $\text{Entier}_n(\sigma)$. Quelle est la complexité de l'algorithme ?

```
val entier : int -> motBin -> int vect = <fun>
```

- b) On considère le mot σ de longueur 20 défini par $\sigma_i = 1$ si et seulement si $u_i \bmod 819 < 409$. Donner la valeur de $\text{Entier}_n(\sigma)$ pour $n = 1, \dots, 5$.

Ainsi pour un entier n donné, un automate binaire A code un sous-ensemble R de \mathbb{N}^n lorsque $\mathcal{L}(A) = \{\sigma \in \{0, 1\}^* \mid \text{Entier}_n(\sigma) \in R\}$. À présent, pour tout k entier positif, on définit l'équation diophantienne E_k à n variables x_i par $\sum_{i=0}^{n-1} c_i \cdot x_i = c_n$ avec $c_i = (u_i - 6) \bmod 13$ et $u_0 = 5 + 17k$.

Exercice 6 [Automate codant les solutions entiers positifs d'une équation linéaire]

Soit $\sum_{i=0}^{n-1} c_i \cdot x_i = d$ une équation linéaire (E) à coefficients c_i dans \mathbb{Z} . On définit l'automate $A_{(E)} = \langle Q, \delta, F, q_0 \rangle$ défini par

- $Q = \{\emptyset\} \cup (\{0, \dots, n-1\} \times \mathbb{Z})$,
- $q_0 = (0, -d)$,
- $F = \{0, \dots, n-1\} \times \{0\}$,
- pour $a = 0, 1$, $\delta(\emptyset, a) = \emptyset$,
- pour tout $a = 0, 1$ et $(i, e) \in \{0, \dots, n-1\} \times \mathbb{Z}$,

$$\delta((i, e), a) = \begin{cases} (i+1, e+a \cdot c_i) & \text{si } i < n-1, \\ (0, \frac{e+a \cdot c_{n-1}}{2}) & \text{si } i = n-1 \text{ et } e+a \cdot c_{n-1} = 0 \pmod{2}, \\ \emptyset & \text{sinon.} \end{cases}$$

Bien que par définition l'automate a un nombre infini d'états, l'ensemble de ses états accessibles est fini. On peut montrer que $\text{Acc}(A_{(E)}) \subset \{\emptyset\} \cup (\{0, \dots, n-1\} \times \{-M, \dots, M\})$ avec $M = 2 \sum_{i=0}^{n-1} |c_i|$.

- a) Écrire une fonction Caml `diophVersAutomate` qui à une équation diophantienne (E) donnée construit l'automate $A_{(E)}$ qui code ses solutions. Quelle est la complexité de l'algorithme en fonction des paramètres de l'équation ?

```
val diophVersAutomate : eqDioph -> automate = <fun>
```

- b) Donner le cardinal de $\text{Acc}(A_{(E_0)})$ pour $n = 4, \dots, 8$.
- c) Pour l'équation (E_0) avec $n = 4, \dots, 8$, donner la valeur de Entier_n du 10^e , 100^e , 1000^e , 10000^e mot reconnu par l'automate $A_{(E_0)}$.

Exercice 7 [Automate codant les solutions d'un système d'équations diophantiennes]

- a) Soit S_k le système d'équations E_0, \dots, E_k . Calculer les automates correspondant à S_2, \dots, S_5 avec $n = 4, \dots, 8$ et donner pour chacun le nombre d'états accessibles.

```
val diophSysVersAutomate : eqDioph vect -> automate = <fun>
```

- b) Déterminer la plus petite valeur k telle que le système S_k n'a pas de solutions pour $n = 4, \dots, 8$.