Automates avancés

Master 1 II et MI

Partiel du 24 mars 2009 – petit corrigé

1 Rappelez-vous, le pumping lemma... 1+3+6

On considère les [ensembles de] nombres binaires qui correspondent aux puissances de

1. *8*
2. *10*
3. *3*

Est-ce qu'ils correspondent aux langages réguliers ? Justifiez bien vos réponses.

J'utilise le codage binaire little-endian avec les 0 qui trainent (comme dans la preuve de Presburger).

1. $8^k = 2^{3k}$ s'écrit en binaire comme 1 et $3k$ zéros. Ça correspond à un langage régulier,

$$(000)^*10^*.$$

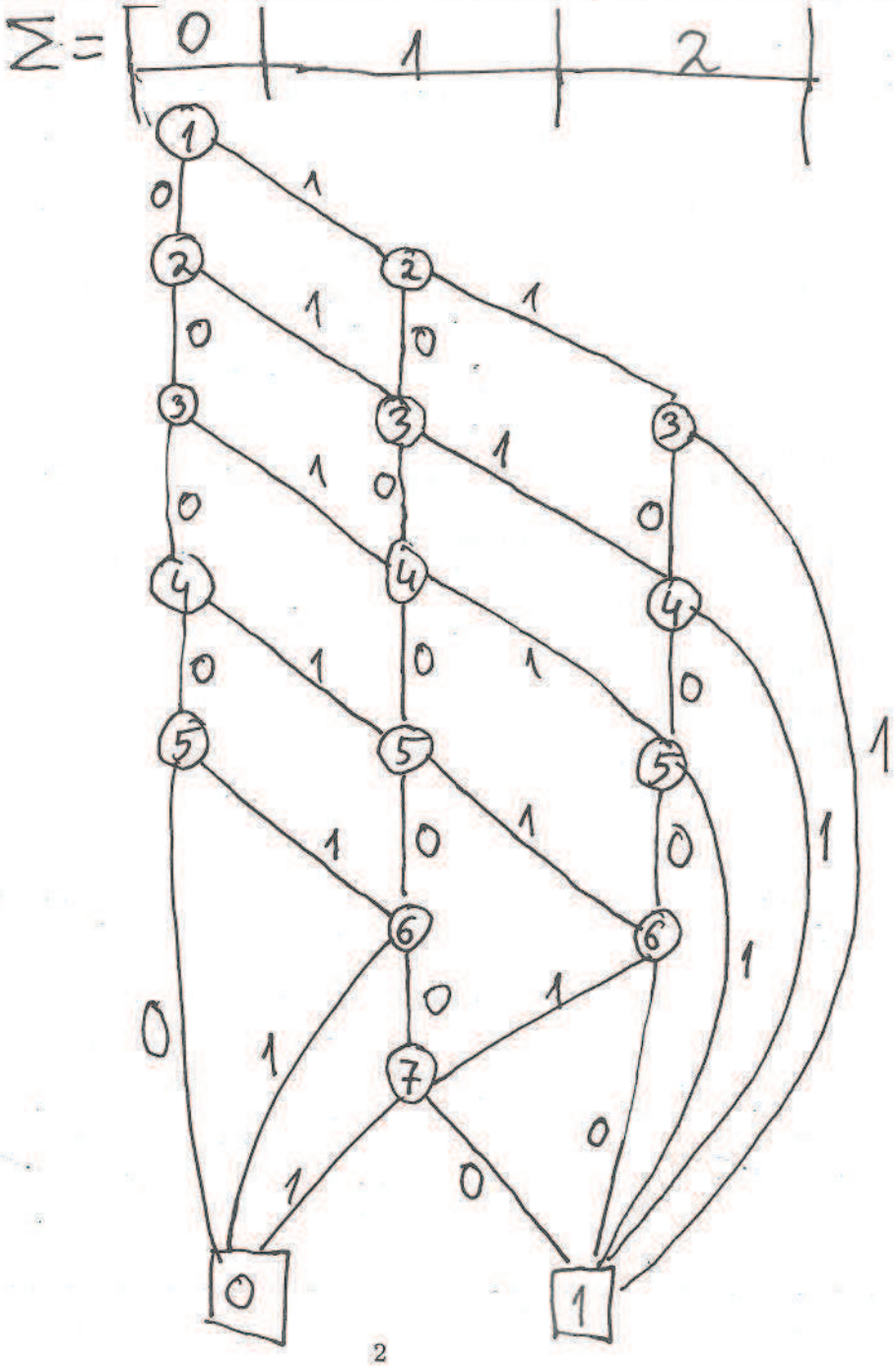
2. Supposons que le langage $L = \{10^k \textit{ en binaire}\}$ est régulier et reconnu par un automate déterministe A avec N états. Prenons le mot w qui correspond à $x = 10^{N+1}$. Ce dernier nombre est multiple de 2^{N+1} et commence donc par $N+1$ zéros en binaire. Par pumping lemma, quand l'automate A parcourt ces zéros il doit passer 2 fois par un même état p, et le sous-mot v qu'il parcourt entre deux visites de p peut être supprimé (ou gonflé) dans w tout en restant dans L. Évidemment le sous-mot v est une série de zéros, soit k leur nombre. Quand on supprime v dans w on obtient l'écriture binaire de $y = 10^{N+1}/2^k$. Cet y ne peut pas être la puissance de 10, et donc son écriture binaire ne doit pas être dans L. Contradiction. On déduit que le langage n'est pas régulier.
3. Le langage de puissances de 3 en binaire n'est pas régulier, mais la preuve par pumping lemma que je connais est un peu trop lourde.

2 Appliquez des algos de cours 2+4

1. *Construisez un BDD (ordonné réduit) pour la fonction* $G : \{0,1\}^7 \to \{0,1\}$ *définie comme suit*

$$G(x_1,\ldots,x_7) = \begin{cases} 1 & si \quad x_1+x_2+x_3+x_4+x_5 > x_6+x_7 \\ 0 & sinon \end{cases}$$

Le principe est simple : on calcule, bit par bit, la somme : x_1, puis x_1+x_2, ..., $x_1+x_2+x_3+x_4+x_5$. Ensuite on soustrait les 2 derniers bits $x_1+x_2+x_3+x_4+x_5-x_6$ et finalement $x_1+x_2+x_3+x_4+x_5-x_6-x_7$. Si le résultat final est positif on accepte, sinon on rejette. Pour obtenir un BDD réduit, on fait attention pour accepter ou rejetter dès que possible, sans terminer le calcul. Par exemple, si la somme cumulée atteint 3, on sait tout de suite que $G=1$. Si au niveau 5,6 ou 7, la somme est 0, on sait que $G=0$. On obtient le BDD suivant.



2. *Appliquez l'algorithme d'apprentissage d'Angluin au langage suivant sur $\{a, b\}$: Tous les mots qui commencent par a et qui ont les b sur toutes les positions paires.*

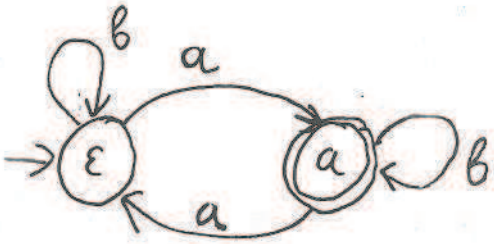
On fait la table :

	ε
ε	0
a	1
b	0

Elle n'est pas complète et on ajoute a aux préfixes :

	ε
ε	0
a	1
b	0
aa	0
ab	1

On obtient un automate :



Le Prof donne un contre-exemple *ba* (accepté par notre automate, mais pas dans le langage).

On ajoute $\{b, ba\}$ aux préfixes :

	ε
ε	0
a	1
b	0
ba	0
aa	0
ab	1
bb	0
baa	0
bab	0

Inconsistance : ε est équivalent à b, mais en ajoutant a on obtient a et ba non-équivalents.

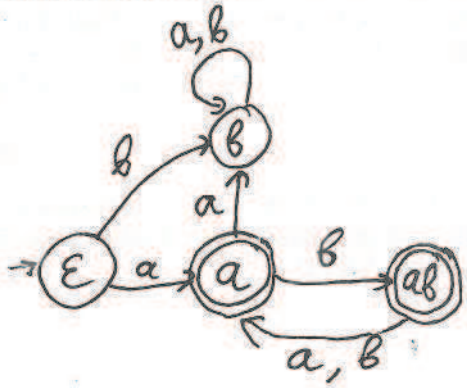
On ajoute a aux suffixes :

	ε	a
ε	0	1
a	1	0
b	0	0
ba	0	0
aa	0	0
ab	1	1
bb	0	0
baa	0	0
bab	0	0

et ensuite ab aux préfixes pour compléter :

	ε	a
ε	0	1
a	1	0
b	0	0
ba	0	0
ab	1	1
aa	0	0
bb	0	0
baa	0	0
bab	0	0
aba	1	0
abb	1	0

La table est complète est consistante et on obtient un automate :



Le Prof confirme qu'il est correct.

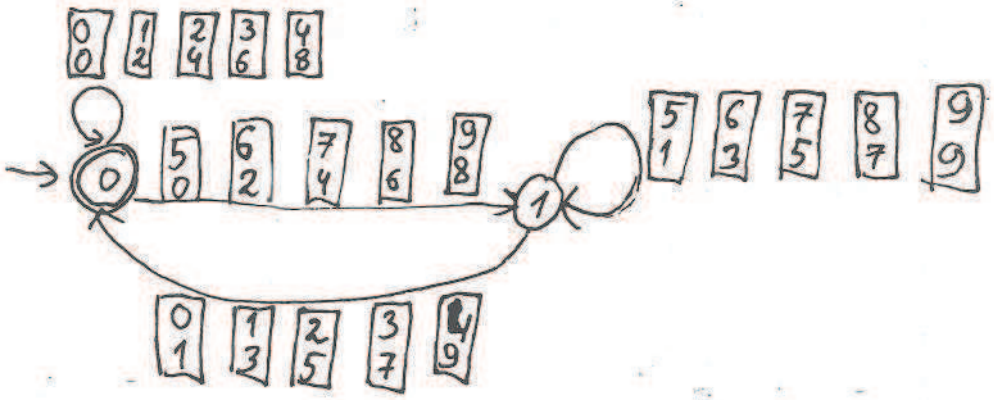
3 Adaptez les méthodes de cours 2+6

1. On considère les mots de l'alphabet $\{0,1,2,\ldots,9\}^2$. Un tel mot sera interprété comme deux entiers naturels x, y en système décimal (le digit le moins important d'abord) écrit sur 2 pistes. Ainsi, pour représenter $x = 981, y = 1962$ on écrira

1	8	9	0
2	6	9	1

Trouvez un automate qui reconnaît tous les mots qui correspondent à x, y tels que $2x = y$. Expliquez brièvement son principe de fonctionnement.

On code l'algorithme scolaire pour multiplier un nombre décimal par 2. Les états de l'automate correspondent à la retenue.



2. *Démontrez qu'il existe un langage régulier fini G sur $\{a,b\}$ tel que*
   - *tous les mots de G contiennent < 100 lettres ;*
   - *ET tout automate qui reconnaît G contient > 1000000 états*

   Soit A l'ensemble de tous les mots de < 100 lettres. Il existe 2^n mots de n lettres. Donc la cardinalité de A est $\#A = 1 + 2 + 4 + \ldots + 2^9 9$. On fait une approximations grossière de ce nombre :

$$\#A = 2^{100} - 1 = \left(2^{10}\right)^{10} - 1 = 1024^{10} - 1 > 1000^{10} = 10^{30}.$$

   Les langages dont tous les mots contiennent < 100 lettres sont tous les sous-ensembles de A. Le nombre total de tels sous-ensemble est

$$\#\text{langages} = 2^{\#A} > 2^{10^{30}}.$$

   Comptons maintenant tous les automates avec 1000000 états. On suppose que les états sont $1, 2, \ldots, 1000000$, et que l'état initial est 1. Pour spécifier la relation de transition il faut remplir un tableau booléen $\Delta[i,j,c]$ de taille $1000000 \times 1000000 \times 2$ qui renseigne pour chaque i, chaque j et $c \in \{a,b\}$ est-ce que la transition $i \xrightarrow{c} j$ est possible. Pour spécifier les états finals il faut remplir un tableau booléen $F[i]$ de de taille 1000000 qui renseigne pour chaque i est-ce que l'état i est final. En tout, pour spécifier complètement un automate il faut $B = 10^6 \cdot 10^6 \cdot 2 + 10^6 = 2 \cdot 10^{12} + 10^6 < 10^{13}$ bits. Il y a 2^B possibilités de donner des valeurs à ces bits. On a donc au plus 2^B automates différents avec 1000000 états, avec

$$\#\text{automates} = 2^B < 2^{10^{13}}.$$

   En comparant le nombre de langages et le nombre d'automates, on voit qu'il y a beaucoup plus de langages que d'automates. Donc certains langages ne correspondent à aucun automate. CQFD

4 Les congruences 2+2+2+4+3+2

On cherche un algorithme pour résoudre les systèmes de congruences linéaires entières avec une inconnue. Un tel système a la forme suivante (avec x l'inconnue, c_i, m_i des entiers donnés) :

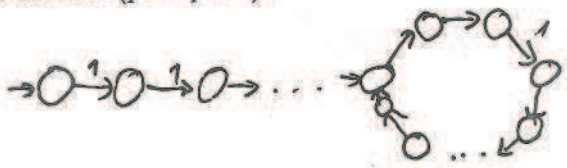
$$\begin{cases} x & \equiv & c_1 (\mathrm{mod}\, m_1) \\ \dots & \dots & \dots \\ x & \equiv & c_k (\mathrm{mod}\, m_k) \end{cases} \quad (1)$$

On propose de représenter x en système unaire (comme séquence de x chiffres 1).

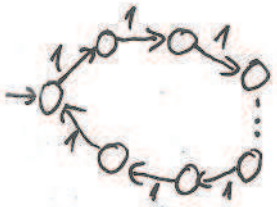
1. Montrez que l'ensemble de solutions d'une congruence $x \equiv c (\mathrm{mod}\, m)$ correspond à un langage régulier.
2. Montrez que l'ensemble de solutions du système (1) correspond à un langage régulier.
3. Déduisez un algorithme pour décider si le système (1) a une solution.
4. Comment caractériser l'ensemble de toutes ses solutions ?
5. Est-il possible de suivre le même schéma en utilisant le système binaire ? Détaillez.
6. Illustrez vos raisonnements pour tous les points précédents sur l'exemple :

$$\begin{cases} x & \equiv & 1 (\mathrm{mod}\, 2) \\ x & \equiv & 2 (\mathrm{mod}\, 3) \end{cases}$$

1. L'expression régulière $(1^m)^* 1^c$. L'automate est un cycle de m états avec état c accepteur.
2. L'ensembles de solutions d'un système correspond à l'intersection des ensembles de solutions de chaque équation. On construit donc un automate pour chaque équation, on les intersecte, et on teste si le langage de l'automate de l'intersection est non-vide.
3. Le raisonnement du point 2 montre que toutes les solutions en unaire constituent un langage régulier reconnu par l'automate produit déterministe avec un seul état final A. Deux cas sont possible :
   - L'état final est inaccessible de l'état initial. Dans ce cas il n'y a pas de solutions.
   - L'état final est accessible. Dans ce cas, en effaçant des états inutiles (inatteignables) on obtient un automate déterministe sur l'alphabet singleton $\{1\}$ avec tous les états atteignables à partir de q_0 (il reste toujours un état final et un seul). Un tel automate a toujours une structure très simple de lasso (pourquoi ?) :



En fait après M transitions on revient à l'état initial. (où M est le produit de tous les modulos m de toutes les congruences). Donc notre lasso n'a pas de poignée, c'est juste une boucle :



Soit B la longueur de la boucle, et C la position de l'état final dans la boucle. On obtient l'expression suivante pour l'ensemble de solutions :

$$\{C + Bn | n \in \mathbb{N}\}.$$

Autrement dit, il y a une solution périodiques $C + Bn$ et c'est tout.
On conclut qu'un système de congruences soit n'a pas de solutions, soit a une seule série de solutions $C + Bn$.

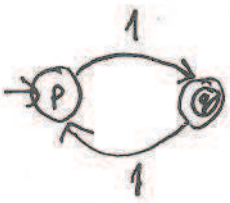
4. Une congruence $x \equiv c (\mathrm{mod}\, m)$ peut s'écrire comme ceci :

$$C(x) = \exists y \left(x = \underbrace{1 + \ldots + 1}_{c \text{ fois}} + \underbrace{y + \ldots + y}_{m \text{ fois}} \right).$$

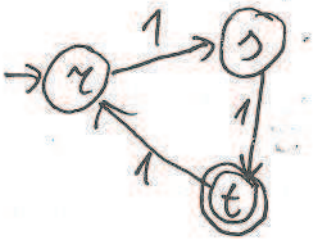
C'est une formule de Presburger (pour m et c fixes). Par théorème du cours, l'ensemble de ses solutions en binaire est régulier, et on sait en principe construire l'automate. Pour résoudre le système on passe à l'automate intersection. Je ne sais pas caractériser l'ensemble de solutions (comme dans le point 4) à partir de l'automate binaire.
Conclusion : on peut faire 1 et 2 en binaire.

5. Je le fais en unaire seulement :

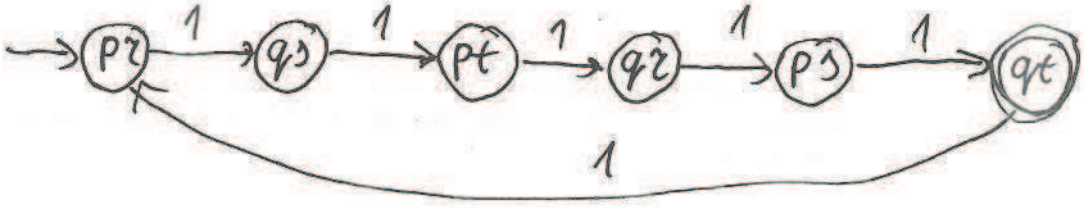
$X \equiv 1 \pmod 2$



$X \equiv 2 \pmod 3$



produit :



solution : $\{5 + 6n \mid n \in \mathbb{N}\}$