Module I41, Licence Informatique 2

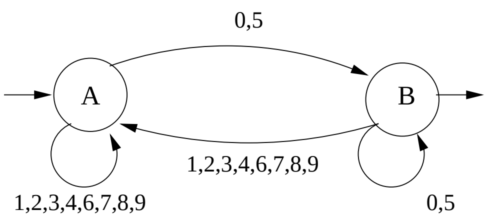
Septembre 2006

Tous les documents sont autorisés. Vous êtes invités à remettre une copie claire, concise, sans rature ni surcharge. Il est par ailleurs inutile de recopier l'énoncé… Le barème est donné à titre indicatif, la note finale tiendra compte de la présentation générale de la copie.

[1] Automates [5 pts]

Exercice 1. Considérons l'alphabet $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

1. On considère l'automate :



   Quel est le langage reconnu par cet automate ? Déterminer la forme générale des mots et leur propriété arithmétique.
2. On rappelle que la somme des chiffres décimaux d'un multiple de trois est encore un multiple de trois. Vérifier que si $n = (n_k \dots n_0)_{10} \equiv 0 \mod 3$ alors $(n_k \dots n_0 0)_{10} \equiv 0 \mod 3$. Pour $(n_k \dots n_0)_{10} \equiv 0 \mod 3$, compléter le tableau :

x	0	1	2	3	4	5	6	7	8	9
$(n_k \dots n_0 x)_{10} \mod 3$										

3. Faire un tableau similaire pour $(n_k \dots n_0)_{10} \equiv 1 \mod 3$, puis pour $(n_k \dots n_0)_{10} \equiv 2 \mod 3$.
4. Construire un automate à trois états (les restes modulo 3) qui reconnait le langage des multiples de 3 en base 10 et le mot vide.
5. Modifier l'automate précédent pour qu'il ne reconnaisse pas le mot vide.

```
Algorithme valuation( a : entier)
données r : entier;
  début
    r ← 0
    tant que (a   pair) faire
             r ← r + 1;
             a ← a div 2;
    fintq
    retourner(r)
  fin
```

Figure 1: Valuation dyadique

[2] Logique de Hoare [3 pts]

Soit a un entier naturel non nul. La valuation dyadique de a est l'unique entier k tel que $a = 2^k b$ avec b impair. L'objectif de l'exercice est de prouver que l'algorithme (1) calcule et retourne la valuation dyadique de d'un entier.

(a) Montrer $2^r a = \gamma$ est un invariant de boucle de l'algorithme (1).

(b) Faire une preuve partielle.

(c) Montrer que $a \geq 1$ est un invariant.

(d) Proposer un variant pertinent pour faire une preuve d'arrêt.

[3] Implantation [4 pts]

(a) Donner une implantation itérative en langage C de l'algorithme (1).

(b) Donner une implantation récursive en langage C de l'algorithme (1).

[4] Arithmétique [3 pts]

Soient a et b deux entiers positifs ou nul. On note $\mathrm{PGCD}(a, b)$ le plus grand diviseur commun de a et b. On rappelle que

$$\mathrm{PGCD}(a, b) = \mathrm{PGCD}(a, b + a), \quad \mathrm{PGCD}(a, a) = a.$$

(a) Montrer que $2\,\mathrm{PGCD}(a, b) = \mathrm{PGCD}(2a, 2b)$.

(b) Montrer que si a est un entier impair et b un entier pair alors alors $\mathrm{PGCD}(a, b) = \mathrm{PGCD}(a, b \text{ div } 2)$.

(c) Montrer que si a et b sont deux entiers impairs alors $\mathrm{PGCD}(a, b) = \mathrm{PGCD}(a, (a + b)/2)$.

[5] Algorithme [5 pts]

On considère les primitives :

$\mathrm{PAIR}(a)$ qui renvoie vrai ou faux suivant la parité de a.

$\mathrm{DICH}(a)$ qui renvoie le quotient de a par 2.

$\mathrm{MED}(a, b)$ qui renvoie le quotient de $a + b$ par 2.

$\mathrm{DIFF}(a, b)$ qui renvoie vrai si et seulement si $a \neq b$.

$\mathrm{SUP}(a, b)$ qui renvoie vrai si et seulement si $a \geq b$.

Ecrire un algorithme $\mathrm{PGCD\text{-}BIN}(a, b)$ de calcul du plus grand diviseur commun de deux entiers fondé sur les relation (a), (b) et (c) de l'exercice précédent. L'algorithme proposé sera obtenu en complétant les lignes de (2) en utilisant les primitives ci-dessus. Préciser le temps de calcul de l'algorithme en fonction de la taille binaire des valeurs initiales.

[6] Temps de calcul [1 pts]

Préciser le temps de calcul de l'algorithme en fonction de la taille binaire des valeurs initiales.

```
Algorithme PGCD-BINAIRE( a, b : entier)
données r : entier;
  début
    r ← 1;
    si SUP(b, a) alors a ↔ b;
    finsi
    tant que PAIR(a) ∧ PAIR(b) faire
             a ← DICH(a)
             b ← DICH(b)
             ...
    fintq
    tant que DIFF(a, b) faire
             ...
             ...
             ...
             ...
             ...
             ...
             ...
             ...
    fintq
    ...
    retourner(r)
  fin
```

Figure 2: PGCD binaire, à compléter !