Université Paris Diderot — Année 2010–2011

Théorie et pratique de la concurrence

Contrôle Continu

Exercice 1 [2 points] Soit une fonction f pour laquelle il existe une valeur i telle que $f(i) = 0$. Les algorithmes concurrents ci-dessous cherchent une telle valeur i (appelée aussi le point zéro) en divisant l'espace de recherche en deux parties : les points zéro positifs $i \geq 0$ et les points zéro négatifs $i < 0$. Un algorithme est correct si, pour toutes les exécutions, les deux processus terminent après que l'un d'entre eux a trouvé un point zéro.

L'algorithme suivant est-il correct, justifiez votre réponse.

Algorithme Zéro

```
boolean found:=false
```

```
process P
    integer i:=0;
p1: while not found
p2:    i:=i+1
p3:    found:=(f(i)==0)
```

```
process Q
    integer j:=1;
q1: while not found
q2:    j:=j-1
q3:    found:=(f(j)==0)
```

Exercice 2 [4 points] Soit la solution suivante pour l'implémentation d'une section critique :

Algorithme 1

```
boolean D1 = False, D2 = False; // variables partagées
```

```
Process P1
  loop forever:
  p1: section NC
  p2: D1:=True
  p3: await (D2==false)
  p4: section critique
  p5: D1:=False
```

```
Process Q
  loop forever:
  q1: section NC
  q2: D2:=True
  q3: await (D1==false)
  q4: section critique
  q5: D2:=False
```

1. Construisez le diagramme d'états de cet algorithme
2. Quelles sont les propriétés vérifiées par cet algorithme (exclusion mutuelle, absence de famine, absence d'interblocage) ?

Exercice 3 [5 points] Considérons l'Algorithme 2 pour l'implémentation d'une section critique :

1. Numérotez les lignes de cet algorithme dans l'idée de pouvoir construire ensuite le diagramme d'états

Algorithme 2

```
int wantP = 0, wantQ = 0;   // variable partagée
```

```
Process P
  loop forever:
    if (wantQ = −1)
        wantP := −1
    else
        wantP := 1
    await wantP != wantQ
    section critique
    wantP := 0
```

```
Process Q
  loop forever:
    if (wantP = −1)
        wantQ := −1
    else
        wantQ := 1
    await wantQ == wantP
    section critique
    wantQ := 0
```

2. Exprimez à l'aide d'une formule LTL la propriété d'exclusion mutuelle
3. Exprimez à l'aide d'une formule LTL la propriété d'absence de famine
4. En utilisant le diagramme d'états, indiquez si cet algorithme vérifie la propriété d'exclusion mutuelle (dans le cas d'une réponse négative, seulement une partie du diagramme d'états exhibant le comportement incorrect est demandée).

Exercice 4 [**5 points**] Soit la solution suivante pour l'implémentation d'une section critique :

Algorithme 3

```
int NP = 0, NQ = 0; // variables partagées
```

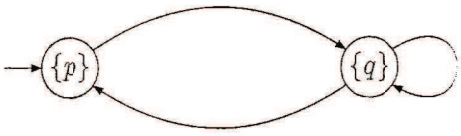
```
Process P
  loop forever:
  p1: section NC
  p2: NP := NQ + 1
  p3: await (NQ==0 or NP<=NQ)
  p4: section critique
  p5: NP:=0
```

```
Process Q
  loop forever:
  q1: section NC
  q2: NQ := NP + 1
  q3: await (NP==0 or NQ<NP)
  q4: section critique
  q5: NQ:=0
```

1. Les valeurs prises par les entiers NP et NQ sont-elles bornées ? (Justifiez votre réponse)
2. Quelle conséquence cela a-t-il sur le diagramme d'états ?
3. Cet algorithme respecte-t-il la propriété d'exclusion mutuelle ? (Justifiez votre réponse)
4. Cet algorithme respecte-t-il la propriété d'absence d'interblocage ? (Justifiez votre réponse)

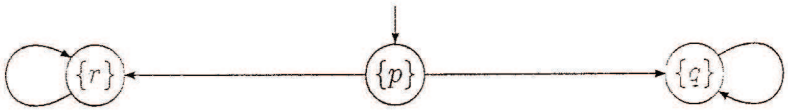
Exercice 5 [4 points] Dans les différents exemples donnés ci-dessous, parmi la liste des propriétés LTL fournies, donnez celles que vérifient le diagramme d'états correspondant. Cette question n'est pas notée comme un QCM mais trop de réponses fausses pourront faire baisser la note.

1.



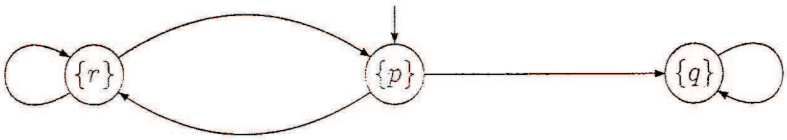
(a) $\Box p$
(b) $\Box(p \Rightarrow \bigcirc q)$
(c) $(\Box\Diamond p) \Rightarrow (\Box\Diamond q)$

2.



(a) $\Box(\bigcirc(q \vee r))$
(b) $\Box(p \Rightarrow \Diamond q)$
(c) $\Diamond(q \vee r)$

3.



(a) $\bigcirc(q \vee r)$
(b) $\Box(p \Rightarrow \Diamond q)$
(c) $\Box(p \Rightarrow \bigcirc(q \vee r))$

4.

(a) p
(b) $\Box(p \vee q)$
(c) $\Box(p \Rightarrow \Diamond q)$