IUT Orsay

DUT Informatique
Module Système S4-L
2010 / 2011

Travaux Dirigés nº 1 : Section Critique

Objectifs : comprendre les principes de l'exclusion mutuelle et savoir déterminer si un algorithme respecte ces principes, par exemple à l'aide de systèmes de transitions étiquetées.

1 Notes de cours

Plusieurs processus peuvent vouloir accéder à une même ressource (CPU, mémoire centrale, I/O, variables partagées etc.), on dit alors qu'ils sont en *compétition*. Pour garantir la validité des résultats, il est parfois nécessaire qu'une partie de code soit effectuée de façon ininterrompue relativement à une ou plusieurs ressources, on appelle cela une *section critique*.
On dit que deux sections critiques sont en *exclusion mutuelle* si l'une ne peut pas être exécutée en même temps que l'autre (même si on a plusieurs unités centrales). À chaque fois qu'on a compétition sur une ressource, on a une section critique.

Schéma général du passage en section critique pour un processus :

```
tant_que vrai faire
  actions avant section critique
  demande d'entrée en section critique
  section critique
  sortie de section critique
  actions après section critique
```

Spécifications de l'exclusion mutuelle :

1. *exclusion mutuelle* ;
2. *progression du système* : si aucun processus n'est en section critique et qu'un processus demande à y entrer, alors ce processus entre en section critique ;
3. *absence de blocage* : au moins un processus peut évoluer ;
4. *absence de famine* : après avoir demandé une entrée en section critique, un processus devra attendre un nombre *fini* de fois avant d'y entrer.

2 Exercices

2.1 Exercice 1 : exclusion mutuelle

Dans un système informatique, on dispose de trois fichiers $F1$, $F2$ et $F3$ et de trois processus dont les programmes A, B et C ont les structures suivantes :

Programme A	Programme B	Programme C
actions $A1$	actions $B1$ (écrire $F3$)	actions $C1$ (lire $F3$)
actions $A2$ (lire $F2$)	actions $B2$	actions $C2$
actions $A3$	actions $B3$ (lire $F1$)	actions $C3$
actions $A4$ (écrire $F3$)	actions $B4$	actions $C4$ (écrire $F2$)
actions $A5$	actions $B5$	actions $C5$

Chaque fichier ne peut ni être lu et modifié en même temps, ni modifié par plusieurs processus en même temps.

1. Réécrivez le tableau ci-dessus en ajoutant entre les actions, où ce sera utile, des commandes `debut_protection` et `fin_protection` pour mettre en évidence les sections critiques de A, B et C. Vous veillerez à ne pas protéger inutilement une action.
2. En déduire les sections en exclusion mutuelle. Mettez-les en évidence en précisant à la suite de vos commandes `debut_protection` une liste d'actions contre qui on doit se protéger (par exemple `debut_protection(B1,B2)` signifie que $B1$ et $B2$ ne devront pas s'exécuter en même temps que les actions protégées).

2.2 Exercice 1 : exclusion mutuelle par variables partagées

On dispose de deux processus (`P0` et `P1`) dont le squelette de programme est celui donné dans les notes de cours. L'objectif de l'exercice est d'établir un algorithme utilisant des variables partagées par les processus pour réaliser l'exclusion mutelle des sections critiques en respectant ses spécifications. On ne fait aucune hypothèse sur l'atomicité des manipulations des variables.

1. Préciser pourquoi une simple attente active sur une variable partagée `tour` qui vaudrait vrai ou faux selon qu'un processus est ou non en section critique, immédiatement suivie d'une mise à jour de `tour` ne convient pas. L'algorithme (1) de `Pi` (dans cet algorithme, remplacer `i` par `0` pour avoir `P0` ou par `1` pour avoir `P1`) serait le suivant :

```
tant_que vrai faire
| actions avant section critique
| tant_que tour faire
| | rien
| tour ← vrai
| section critique i
| tour ← faux
| actions après section critique
```

2. On propose dans un premier algorithme, la solution suivante. La variable partagée `tour` prend les valeurs `0` ou `1`. L'algorithme (2) de `Pi` est le suivant :

```
tant_que vrai faire
| actions avant section critique
| tant_que tour <> i faire
| | rien
| section critique i
| tour ← 1-i
| actions après section critique
```

   Noter que `1-i = 0` si `i = 1` et 1 si `i = 0`. La phase d'initialisation du système est réduite à : `tour` $\leftarrow$ `0`. Construire le système de transitions étiquetées de cette solution. Montrer que l'exclusion mutuelle est bien réalisée mais que la condition de progression n'est pas satisfaite.

3. Pour remédier à cet inconvénient, on emploie au lieu de `tour`, deux variables booléennes partagées `D0` et `D1`. `Di` est vraie ssi `Pi` demande à passer en section critique. L'algorithme (3) de `Pi` est alors :

```
tant_que vrai faire
| actions avant section critique
| Di ← vrai
| tant_que D(i-1) faire
| | rien
| section critique i
| Di ← faux
| actions après section critique
```

   `D(1-i)` désigne `D1` si `i = 0` et `D0` si `i = 1`. Les variables *Di* sont initialisées à faux. Construire le système de transitions étiquetées de cette solution. Montrer que la progression est maintenant assurée. Montrer que le système possède un état d'interblocage.

4. Voir suite en exercice corrigé.

3 Entraînement : exercice corrigé

3.1 Énoncé : algorithme de Peterson *

Une dernière solution au problème d'exclusion mutuelle par variables partagées est due à G.I. Peterson (1981). Elle vérifie les propriétés attendues d'une solution d'exclusion mutuelle. On utilise une variable `tour` et les deux variables `Di`.

```
    tant_que vrai faire
    | actions avant section critique
1:  | Di ← vrai
2:  | tour ← 1-i
3:  | tant_que (D(1-i) et (tour = 1-i)) faire
    | | rien
4:  | section critique i
5:  | Di ← faux
    | actions après section critique
```

L'initalisation met `tour` à `0` et les `Di` à faux. Montrer que les quatre spécifications de l'exclusion mutuelle sont réalisées. Combien de passages en section critique de `P(1-i)`, `Pi` doit-il attendre au plus lorsqu'il a demandé à passer en section critique ?

3.2 Correction (essayez d'abord ! ! !)

1. Exclusion mutuelle : raisonnons par l'absurde, si l'exclusion mutuelle n'est pas respectée, $P0$ et $P1$ se trouvent au même moment en section critique. Alors les trois propriétés suivantes sont vraies en même temps :
   - $((D1 = faux) OU (tour = 0))$, c'est à dire rien ne bloque $P0$.
   - $((D0 = faux) OU (tour = 1))$, c'est à dire rien ne bloque $P1$.
   - $((D0 = vrai) ET (D1 = vrai))$, car les deux processus ont mis à vrai leur variable *Di* avant d'entrer en section critique.

   Cela implique que *tour* vaut en même temps 1 et 0, ce qui est impossible, l'exclusion mutuelle est donc respectée.

2. Non blocage : toujours par l'absurde, si les deux processus sont bloqués en même temps, les deux propriétés suivantes sont vraies en même temps :
   – $((D1 = vrai)ET(tour = 1))$, c'est la condition de blocage de $P0$.
   – $((D0 = vrai)ET(tour = 0))$, c'est la condition de blocage de $P1$.
   Cela implique que *tour* vaut en même temps 1 et 0, ce qui est impossible, le non-blocage est donc respecté.
3. Progression : si $P0$ n'est pas en section critique ou en demande d'entrée, alors $D0$ vaut $faux$. Si dans le même temps $P1$ est bloqué (donc il y a non-progression), c'est que $((D0 = vrai)ET(tour = 0))$. Donc $D0$ vaudrait $faux$ et *vrai* en même temps ce qui est impossible. De même si $P1$ n'est pas en section critique et que $P0$ est bloqué. La progression est donc respectée.
4. Non famine : on doit se placer dans le cas où les deux processus ont demandé leur entrée en section critique et ou l'un l'obtient (cela arrive forcément car la progression est respectée), par exemple $P0$ (le problème est symétrique avec $P1$). Alors on a :
   – $((D1 = vrai)ET(D2 = vrai))$, car les deux ont fait la demande d'entrée.
   – $(tour = 0)$ car $P0$ a pu entrér en section critique.
   – $P1$ est en attente.
   Si $P0$ demande une nouvelle fois l'entrée en section critique, alors il placera $D1 = vrai$ et $tour = 1$, s'interdisant l'entrée en section critique et laissant passer $P1$. Donc la non-famine est repectée et un processus attend au plus un passage de l'autre processus pour entrer en section critique.

Il est aussi possible d'analyser le système de transitions étiquetées, mais il serait ici particulièrement grand (44 états !). À partir de ce graphe on pourrait démontrer :

1. Exclusion mutuelle : pas de sommets de type $(v, v, D0, D1, tour)$.
2. Non blocage : il n'y a pas d'état terminal.
3. Progression : à partir d'arcs de type $(f, f, v, f, tour)$ on arrive à $(v, f, v, D1, tour)$.
4. Non famine : À partir d'états du type $(f, q1, v, D1, tour)$, on atteint par un nombre toujours fini d'arcs des états du type $(v, q1', v, D1', tour)$. On voit qu'on attend au plus un passage en section critique de l'autre processus.