Travaux Dirigés n°4

Cours de Parallélisme

—Maîtrise Informatique—

Ce TD a pour objet de mettre en évidence les problèmes d'exclusion d'accès à des données partagées par plusieurs processus concurrents.

On restreint à deux processus P_0 et P_1 notre étude du problème de la section critique et on considère une exécution concurrente du type `parbegin P0; P1; parend`. On supposera par ailleurs que chaque processus a une vitesse d'exécution non nulle et que l'on utilise une machine ne disposant que des opérations atomiques classiques comme `load` et `store`.

▸ **Exercice 1.** (propositions d'algorithmes) *Dans la suite, on décrit l'algorithme effectué par le processus $P_{\mathtt{i}}$ et on dénote par* `j` *le numéro de l'autre processus* ($\mathtt{j} = 1 - \mathtt{i}$)*. Critiquer chacun des algorithmes suivants sensés résoudre le problème de la section critique :*

1. *Les processus partagent une variable* `tour` *initialisée à* 0 *:*

```
int tour=0;
for (;;)
{ while (tour!=i);
  section_critique();
  tour=j;
  section_non_critique();
}
```

2. *L'algorithme suivant utilise une variable* `drapeau` *de type tableau de deux booléens :*

```
typedef enum {faux, vrai} Bool;
Bool drapeau[2]={faux, faux};
for (;;)
{ while (drapeau[j]==vrai);
  drapeau[i]=vrai;
  section_critique();
  drapeau[i]=faux;
  section_non_critique();
}
```

3. *L'algorithme suivant est une légère modification de la proposition précédente :*

```
for (;;)
{ drapeau[i]=vrai;
  while (drapeau[j]==vrai);
```

```
    section_critique();
    drapeau[i]=faux;
    section_non_critique();
  }
```

4. *L'algorithme suivant reprend les variables partagées de l'algorithme précédent :*

```
  for (;;)
  { drapeau[i]=vrai;
    while (drapeau[j]==vrai)
    { drapeau[i]=faux;
      while (drapeau[j]==vrai);
      drapeau[i]=vrai;
    }
    section_critique();
    drapeau[i]=faux;
    section_non_critique();
  }
```

Exercice 2. (algorithme de Dekker) *On considère deux processus P_0 et P_1 qui exécutent l'algorithme suivant dû à Dekker :*

```
typedef enum {faux, vrai} Bool;
Bool drapeau[2] = {faux, faux};
int tour=0;
for (;;)
{ j=1-i;
  drapeau[i]=vrai;
  while (drapeau[j]==vrai)
  { if (tour==j)
    { drapeau[i]=faux;
      while (tour==j);
      drapeau[i]=vrai;
    }
  }
  section_critique();
  tour=j;
  drapeau[i]=faux;
  section_non_critique();
}
```

Prouver que l'algorithme de Dekker satisfait bien les trois contraintes du problème de la section critique pour deux processus, c'est-à-dire l'exclusion mutuelle en section critique, l'atteignabilité de la section critique et l'attente bornée.