Théorie et pratique de la concurrence – Master 1 II
TP 3 : Algorithmes distribués pour les sections critiques

Promela permet de modéliser les algorithmes distribués grâce à des constructions comme les tableaux de processus et les cannaux de communication.

Les canaux de communication en Promela : Le type de donnée `chan` permet de modéliser des canaux de communication. La déclaration d'un canal de communication `ch` de taille `N` où les composantes des messages sont de type `type1`,..., `typen` est :

```
chan ch = [N] of { type1, ... , typen };
```

Si $N = 0$ la communication correspond à un rendez-vous, mais dans le cadre de ce TP, on suppose que $N > 0$. Dans ce cas, la communication par canal se fait de façon asynchrone. Plus précisément, les processus utilisent le canal de communication comme une file :

1. l'opération `ch? x1,...,xn` lit le message (contenant n valeurs) en tête de la file et place le i^{eme} élément du message dans la variable `xi`. Attention : Si le canal est vide, un processus lecteur est bloqué jusqu'à ce qu'un message soit inseré dans le canal de communication. Si une des variables `xi` est remplacée par une constante, le message en tête de file peut être lu que si la i^{eme} composante du message à une valeur égale à la constante ;
2. l'opération `ch! e1,...,en` écrit un message de n valeurs (correspondant à la séquence d'expressions `e1,...,en`) en queue de file. Attention : Si le canal est plein, un processus écrivain est bloqué jusqu'à ce que le canal ne soit plus plein ;
3. l'opération `empty(ch)` permet de tester si un canal est vide ;
4. une série d'opérations existent pour contourner la politique stricte de la file :
   - `ch?[x1,...,xn]` renvoie une valeur non nulle si l'opération de réception est possible,
   - `ch?<x1,...,xn>` fait la lecture du message sans l'enlever de la file,
   - `ch!! e1,...,en` insère le message dans la file juste après le message qui le succède dans l'ordre lexicographique,
   - `ch?? x1,...,xn` réception aléatoire d'un message de ce type, le plus vieux étant enlevé,
   - `ch??[x1,...,xn]` renvoie vrai si l'opération de reception est exécutable,
   - `ch??<x1,...,xn>` réception aléatoire mais sans effacer le message de la file.

Exercice 1 :

Pour vous apprendre à utiliser les cannaux de communication, écrivez en promela le code de deux processus, le premier envoie un message de type `request` avec son identité au second, qui lui répond par un message de type `ack` avec son identité et finalement le premier processus envoie un message de type `ok` pour mettre fin à la communication. Simulez votre code avec Spin. Comment pouvez vous récupérer l'identifiant d'un processus ?

Exercice 2 : *Algorithme de Ricart-Agrawala en Promela*

Dans cet exercice, il s'agit de modéliser avec Promela l'algorithme de Ricart-Agrawala pour `NPROC` sites.

1. Pour modéliser la mémoire locale à chaque site (variables `myNum`, `requestCS`, `highestNum` et `waiting`), on utilisera des tableaux globaux. En effet, ceci nous permettra la communication entre les processus du même site. Pour modéliser l'ensemble `waiting`, on peut utiliser un canal de taille `NPROCS` qui memorise les identificateurs des processus retardés par le récepteur. Déclarer les variables globales.
2. Pour modéliser la communication entre les sites, on utilisera des canaux de taille `NPROCS`, un canal par site. Sur le cannal i, les processus du site i lisent les messages envoyés par les autres processus. Définir le type des messages echangés dans le type `mtype`. Déclarer le tableau de cannaux `ch` pour la communication entre processus.
3. Chaque site de l'algorithme exécute deux processus : le processus qui accède à la section critique (`Main`) et le processus qui reçoit les messages (`Receive`). Déclarer les deux tableaux de processus et rendez-les "actifs".
4. Ecrire le modèle Promela pour le processus `Main`. Afin d'assurer l'atomicité de certaines opérations dans ce processus, il faut utiliser la construction **atomic**.
5. Ecrire le modèle Promela pour le processus `Receive`.
6. En utilisant le simulateur de Spin, construisez une séquence d'exécution dans laquelle les numéros d'ordre utilisés par l'algorithme croissent à l'infini.
7. Ecrire la formula LTL qui exprime l'exclusion mutuelle pour cette modélisation. Tester votre formule sur le modèle construit.
8. Ecrire la formule LTL pour l'absence de famine dans cette modélisation. Tester votre formule sur le modèle construit.