

Théorie et pratique de la concurrence – Master 1 II

TP 5 : Algorithmes distribués pour les sections critiques

www.liafa.jussieu.fr/~sighirea/cours/concur/

Promela permet de modéliser les algorithmes distribués grâce à des constructions comme les tableaux de processus et les canaux 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 inséré 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 :

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 `muNum`, `requestCS`, `highestNum` et `deferred`), on utilisera des tableaux globaux. En effet, ceci nous permettra la communication entre les processus du même site. Pour modéliser l'ensemble `deferred`, 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 canal i , les processus du site i lisent les messages envoyés par les autres processus. Définir le type des messages échangés dans le type `mtype`. Déclarer le tableau de canaux `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". Comment peut-on obtenir l'identificateur du site sur lequel chaque processus s'exécute ?
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 formule 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.

Exercice 2 :

Algorithme d'élection en Promela

On considère un anneau unidirectionnel avec N noeuds reliés entre eux par des canaux de communication (chaque noeud reçoit des messages de son voisin de droite et envoie des messages à son voisin de gauche). Chaque noeud a un identifiant, qui est une valeur naturelle, et on désire élire un leader (i.e., un unique processus du groupe).

Un protocole simple est le suivant : Il y a deux types de messages. Le premier type est utilisé pour détecter le leader, le second pour notifier aux différents noeuds l'identité de ce leader. Chaque processus envoie un premier message de type 1 avec son identifiant. Ensuite, le processus attend des messages envoyés par son voisin de droite : Si il reçoit un message de type 1, plusieurs cas se présentent :

- Si le message contient une valeur d'identifiant plus grande que son identifiant, il envoie le message à son voisin de gauche.
- Si le message contient une valeur d'identifiant plus petite que son identifiant, il ne fait rien et attend un nouveau message.
- Si la valeur d'identifiant du message est son propre identifiant, le noeud a détecté que c'est lui le leader et il envoie un message de type 2 avec son identifiant pour signaler aux autres noeuds qu'il est le leader. Ensuite, le noeud continue à attendre des messages.

Lorsqu'un noeud reçoit un message de type 2, il interprète la valeur de l'identifiant comme l'identité du nouveau leader. Si ce noeud n'est pas le leader, il envoie le message à son voisin de gauche. Sinon, il n'envoie pas le message. Dans les deux cas, le noeud termine le protocole.

1. Ecrire en Promela le protocole de l'élection de leader pour un anneau à trois noeuds dont les identifiants sont 1, 2 et 3. Comme les canaux ont une taille finie et bornée en Promela, on fixe leur taille à 3.
2. Vérifier à l'aide de Spin que les propriétés suivantes sont satisfaites par le modèle :
 - (a) Tous les noeuds terminent le protocole au bout d'un temps fini,
 - (b) Tous les noeuds déterminent à la fin du protocole que le noeud 3 est le leader,
 - (c) Lorsqu'un noeud termine le protocole il ne peut plus recevoir de messages.