SCIENCES SORBONNE UNIVERSITÉ

TME 5 - Implantation sous SPIN de l'algorithme de Naimi & Tréhel

Exercice(s)

Exercice 1 – L'algorithme de Naimi & Tréhel

On souhaite utiliser SPIN pour vérifier que l'algorithme de Naimi & Tréhel réalise bien une exclusion mutuelle et que de plus, il ne crée pas de problème de famine : si un processus demande la section critique, il finira par l'obtenir.

Dans un premier temps, nous devons donc traduire l'algorithme en un programme promela.

Question 1

Complétez le squelette suivant, accessible à partir de la page web du module.

```
#define N      3    /* Number of processes */
#define L     10    /* Buffer size */
#define NIL (N+1)   /* for an undefined value */

mtype = {req, tk};

/* le processus i recoit ses messages dans canal[i] */
chan canal[N] = [L] of {mtype, byte};    /* pour un message de type tk,
                                            on mettra la valeur associee a 0 */

byte SharedVar = 0;                      /* la variable partagee */

inline Initialisation ( ) {
   /* initialise les variables locales, sauf reqId, val et typ_mes */
}

inline Request_CS ( ) {
  /* operations effectuees lors d'une demande de SC */
}

inline Release_CS ( ) {
  /* operations effectuees en sortie de SC */
}

inline Receive_Request_CS ( ) {
  /* traitement de la reception d'une requete */
}

inline Receive_Token () {
  /* traitement de la reception du jeton */
}
```

```
proctype node(byte id; byte Initial_Token_Holder){

   bit requesting;    /* indique si le processus a demande la SC ou pas  */
   bit token;         /* indique si le processus possede le jeton ou non */

   byte father;       /* probable owner */
   byte next;         /* next node to whom send the token */
   byte val;          /* la valeur contenue dans le message */
   mtype typ_mes;     /* le type du message recu */
   byte reqId;        /* l'Id du demandeur, pour une requete */

   chan canal_rec = canal[id];     /* un alias pour mon canal de reception */
   xr canal_rec;                   /* je dois etre le seul a le lire */

   /* Chaque processus execute une boucle infinie */

   Initialisation ( );
   do
       :: ((token == true) && empty(canal_rec) && (requesting == true)) ->
          /* on oblige le detenteur du jeton a consulter les messages recus */
          in_SC :
               /* acces a la ressource critique (actions sur SharedVar),
                  puis sortie de SC */

       :: canal_rec ? typ_mes(val) ->
               /* traitement du message recu */

       :: (requesting == false) -> /* demander la SC */
               Request_CS ( );
   od ;
}

/* Cree un ensemble de N processus */

init {
   byte proc;
   atomic {
      proc=0;
      do
         :: proc <N ->
            run node(proc, 0);
            proc++
         :: proc == N -> break
      od
   }
}
```

Question 2

Vérifiez que l'algorithme réalise bien une exclusion mutuelle.

L'algorithme de Naimi & Tréhel est symétrique : tous les processus exécutent le même code. Pour vérifier l'absence de famine, il suffit donc de vérifier *pour un processus donné* que si celui-ci demande la section critique, il finira par l'obtenir.

Question 3

Vérifiez que cette propriété est réalisée pour le processus `node[1]`.