TME 10 - Implantation sous SPIN du calcul de minimum sur un arbre

Exercice(s)

Exercice 1 – Calcul de minimum sur un arbre

On souhaite utiliser SPIN pour vérifier l'algorithme de calcul de minimum sur un arbre sans annonce. On choisit pour cela d'appliquer l'algorithme sur l'arbre présenté dans la figure 1, où la valeur du minimum local de chaque site est indiquée en italique.

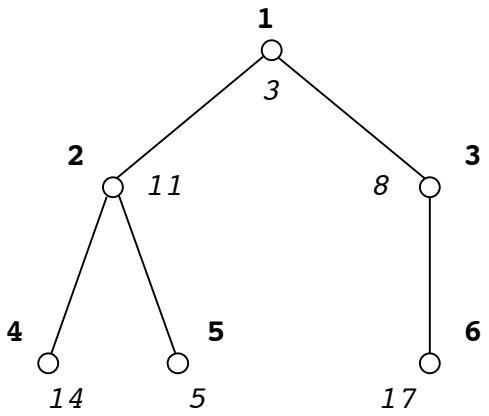

FIGURE 1 – Topologie en arbre

Question 1

Écrivez un programme promela réalisant le calcul de minimum sur un arbre. Vous pourrez pour cela compléter le squelette suivant qui se trouve sur la page web de l'UE.

```
#define N 6            /* Number of processes */
#define NB_V_MAX 3     /* maximum degree of a node */

mtype = {calcul};

chan channel_IN[N] = [N] of {mtype, byte, byte};
   /*  All messages to process i are received on channel_IN[i]
       a message contains information <type, sender, value>   */

/*************************************************************/

inline Simulateur() {
   if
```

```
      :: (id == 0) ->
         minloc = 3; nb_voisins = 2;
         voisins[0] = 1; voisins[1] = 2;
      :: (id == 1) ->
         minloc = 11; nb_voisins = 3;
         voisins[0] = 0; voisins[1] = 3; voisins[2] = 4;
      :: (id == 2) ->
         minloc = 8; nb_voisins = 2;
         voisins[0] = 0; voisins[1] = 5;
      :: (id == 3) ->
         minloc = 14; nb_voisins = 1;
         voisins[0] = 1;
      :: (id == 4) ->
         minloc = 5; nb_voisins = 1;
         voisins[0] = 1;
      :: (id == 5) ->
         minloc = 17; nb_voisins = 1;
         voisins[0] = 2;
   fi;
   mincal = minloc
}

/*************************************************************/
inline Test_Emission (q) {
   /* determine si l'on peut emettre et identifie le destinataire q */
}

inline Test_Decision () {
   /* determine si l'on peut decider */
}

/*************************************************************/

proctype node( byte id ) {

byte nb_voisins;
byte voisins[NB_V_MAX];

chan canal_IN = channel_IN[id];
xr canal_IN;            /* only id reads on this channel */

mtype type;
byte i, nb_recus;
byte sender, receiver, minloc, mincal;

/* tableau initialise a 0 ; recu[i] = 1 si on a recu de i */
byte recu[N];
bool emission, wait = 0, decision, deja_emis = 0;

   Simulateur();

/* Each process executes a finite loop */
   do
      :: /* test emission et traitement correspondant
            - on ne doit emettre qu'une fois
            - si on ne peut pas emettre, on attend (wait = 1) que la
              condition ait pu etre modifiee                        */
```

```
      :: /* test reception et traitement correspondant */

      :: /* test terminaison */

   od;
   printf("%d : le minimum est %d\n", _pid, mincal);
}

/*************************************************************/
/* Creates a network of 6 nodes
   The structure of the network is given by array voisins */

init{
   byte proc;

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

Une fois l'algorithme traduit en promela, on souhaite s'assurer qu'il fonctionne correctement. En particulier, on voudrait vérifier qu'il y a toujours au moins un processus qui parvient à une décision.

Question 2

Écrivez une formule de logique temporelle exprimant cette propriété et vérifiez-la.