Cours Algorithmique des systèmes parallèles et distribués
Exercices
Série :5 Algorithmes répartis
par Dominique Méry
2 février 2026

Exercice 1 *Soit le fichier exieee.tla contenant la description de l'élection du leader dans un graphe connexe acyclique.*

```
---------------- MODULE ieee   --------------------
EXTENDS Naturals

VARIABLES
        nb,sn,bm,bt,ba,root,msg,ack,tr,cnt,con
-------------------------------------------------
NODES == {1,2,3,4}

(*  le noeud i  sait qu'il est le leader, puisque  tous ses voisins sont ses enfant
election(i) ==
 /\ i \in NODES
 /\ nb[i]=sn[i]
        /\ root'=[root EXCEPT![i]= TRUE]
        /\ UNCHANGED <<nb,sn,bm,bt,ba,msg,ack,tr,cnt,con>>
-------------------------------------------------------
(* le noeud x  envoie un message au noeud y, si le message n'est pas dï¿½jï¿½ envoy
(* si $x$  n'a pas dï¿½jï¿½  envoyï¿½  un message d'accord  pour ï¿½tre le parent d

sending_msg(x,y) ==
 /\ x \notin bm
 /\ y \notin ba[x]
 /\ nb[x]=sn[x] \cup {y}
        /\ msg' = msg \cup {<<x,y>>}
 /\ bm' = bm \cup {x}
        /\ UNCHANGED <<nb,sn,bt,ba,root,ack,tr,cnt,con>>
-------------------------------------------------------
(* x a envoyï¿½ un message ï¿½ y; y n'a pas encore envoyï¿½ son accord ï¿½ $x$; y n
(* de message ï¿½ x pour lui demander   d'ï¿½tre le chef.
sending_ack(x,y) ==
 /\ <<x,y>> \in msg
 /\ x \notin ba[y]
 /\ y \notin bm
        /\ ba'=[ba EXCEPT![y]= @ \cup {x}]
 /\ ack' = ack \cup {<<x,y>>}
        /\ UNCHANGED <<nb,sn,bm,bt,root,msg,tr,cnt,con>>
-------------------------------------------------------
```

```
progress(x,y) ==
 /\ <<x,y>> \in ack
 /\ x \notin bt
        /\ tr'=tr \cup {<<x,y>>}
 /\ bt' = bt \cup {x}
        /\ UNCHANGED <<nb,sn,bm,ba,root,msg,ack,cnt,con>>


rcv_cnf(x,y) ==
 /\ <<x,y>> \in tr
 /\ x \notin sn[y]
        /\ sn'=[sn EXCEPT![y]= @ \cup {x}]
        /\ UNCHANGED <<nb,bm,bt,ba,root,msg,ack,tr,cnt,con>>


decontention(x,y) ==
 /\ <<x,y>> \in cnt
 /\  <<y,x>> \in cnt
        /\ msg'=msg - cnt
        /\ bm'= bm - {x,y}
        /\ cnt'= {}
        /\ UNCHANGED <<nb,sn,bt,ba,root,ack,tr,con>>


contention(x,y) ==
 /\ con = 0
 /\ <<x,y>> \in msg
 /\ <<x,y>> \notin ack
 /\ x \notin ba[y]
 /\ y \in bm
        /\ cnt'=cnt\cup {<<x,y>>}
 /\ con'= 1
        /\ UNCHANGED <<nb,sn,bm,bt,ba,root,msg,ack,tr>>

solvecon(x,y) ==
 /\ con = 1
 /\ <<x,y>> \in msg
 /\ x \notin ba[y]
 /\ y \in bm
        /\ ba'=[ba EXCEPT![y]= @ \cup {x}]
 /\ ack' = ack \cup {<<x,y>>}
        /\ UNCHANGED <<nb,sn,bm,bt,root,msg,tr,cnt,con>>
```

```
----------------------------------------------------------
(*  modï¿½lisation *)
Init ==
 /\ nb = [i \in NODES  |-> IF i=1 THEN {2,3} ELSE IF i = 3 THEN  {1,4} ELSE IF i =
 /\ sn = [i \in NODES  |-> {}]
 /\ bm ={}
 /\ bt ={}
 /\ ack ={}
 /\ ba = [i \in NODES  |-> {}]
 /\ root = [i \in NODES  |-> FALSE]
 /\ msg ={}
 /\ cnt ={}
 /\ tr ={}
 /\ con = 0
Next ==
\/ \E i \in NODES: election(i)
\/ \E x,y \in NODES: sending_msg(x,y)
\/ \E x,y \in NODES: sending_ack(x,y)
\/ \E x,y \in NODES: progress(x,y)
\/ \E x,y \in NODES: rcv_cnf(x,y)
\/ \E x,y \in NODES: contention(x,y)
\/ \E x,y \in NODES: decontention(x,y)
\/ \E x,y \in NODES: solvecon(x,y)


==========================================================================
```

Question 1.1 *Montrer que ce modèle TLA vérifie les propriétés attendues de l'élection.*

Question 1.2 *Montrer que les invariants de présentation de la solution sont vérifiés.*

Question 1.3 *Modifier l'algorithme pour résoudre la contention en considérant un choix entre deux nœuds x et y oar exemple le plus petit.*

Question 1.4 *Traduire cette dernière version en utilisant le langage PlusCal.*

Exercice 2 *Soit le fichier exdijkstra.tla contenant la description de l'autostabilisation dans un anneau.*

```
----------------MODULE exdijkstra   --------------------
EXTENDS Naturals
CONSTANTS
        N,M
VARIABLES
        V
-----------------------------------------------------
DOMAINE  == 0 .. N
IMAGE == 0 .. M
-----------------------------------------------------
(* actions *)

NToZero ==
 /\  V[0] = V[N]
 /\  V' = [V EXCEPT![0] = (V[0] + 1) % (M+1)]


Others(I) ==
 /\ I \in  0..N
 /\ I # N
 /\ V[I+1] #  V[I]
    /\ V' = [V EXCEPT ![I+1] = V[I]]

-----------------------------------------------------------
Init == V = [i \in 0..N |-> (IF i # N THEN 9  ELSE 0)]
Next == NToZero \/ (\E I \in 0..N-1:Others(I))

Prop1 ==   (V[0] = V[N])  => (\A i \in 1..N-1: V[i+1] = V[i])

Prop2 ==
       \/  (\E i,j  \in 0..N-1: i # j /\ V[i+1] #  V[i] /\ V[j+1] #  V[j])
       \/ (V[0] = V[N] /\ (\E i  \in 0..N-1:  V[i+1] #  V[i]))

Prop == Prop2

test == Prop2



     (* /\  ( (V[I+1] #  V[I]) /\ (I \in 1..N)) => (V[0] # V[N]) /\ ( \A i \in 1..N

=============================================================
```

Question 2.1 *Compléter le module et vérifier des propriétés attendues comme la stabilité.*

Question 2.2 *Traduire ce module en un module PlusCal qui définit cet algorithme.*