Cours Algorithmique des systèmes parallèles et distribués
Exercices
Série : PlusCal pour la programmation répartie ou concurrente (II)
par Alessio Coltellacci et Dominique Méry
12 mars 2025

Exercice 1 *Compléter le module pluscalappaspd22.tla en proposant une assertion Q1 correcte.*

```
---------------------------- MODULE pluscalappaspd22 ----------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets
(*
--wf
--algorithm ex1{
variables x = 0;


process (one = 1)
variables u;
{
  A:
    u := x+1;
  AB:
    x := u;
  B:
    x := x +1;
};

process (two = 2)
{
  C:
    x := x - 1;
  D:
    assert E2;
};

}
end algorithm;

*)


====
```

Exercice 2 *Compléter le module pluscalappaspd33.tla en proposant deux as-*

sertions R1 et R2 correctes.

```
----------------------------- MODULE pluscalappaspd33 ------------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets
(*
--wf
--algorithm ex3{
variables x = 0, y = 2;


process (one = 1)
variable u;
{
  A:
  u := x+1;
  AB:
    x := u;
  B:
    y := y -1;
  C:
 assert E31;
};

process (two = 2)
{
  D:
    x := x - 1;
  E:
    y:=y+2;
   F:
    x:= x+2;
  G:
    assert E32;
};

}
end algorithm;

*)
\
====
```

Exercice 3 *voir Figure* **??**
On considère un système formé de deux processus one et two assurant les calculs suivants :

— *one : le processus envoie les entiers pairs entre 0 et N via un canal de communication à two.*
— *two : le processus reçoit les valeurs envoyées par one et ajoute la valeur reçue à la variable s.*
— *three : le processus fait un calcul de la somme des entiers de 0 à N/4.*

On suppose que N est divisible par 4..

Question 3.1 *Afin de vérifier que le calcul effectué par les deux processus est correct, on décide de vérifier que, quand tous les processus ont terminé la variable result contient la somme des entiers pairs entre 0 et N.*
En utilisant le fichier qquestion1a.tla, ajouter une propriété de sûreté safety1 qui énonce la correction de cet algorithme.

Question 3.2 *On décide de calculer avec le processus three la somme des entiers de 0 à* $N\%4$*. Proposer une propriété à vérifier afin de monter que le calcul du processus two est correct.*

Exercice 4 *voir Figure* **??**
Soit le petit module qquestion2a.tla.
Donner les deux expressions A1 et A2 à placer dans les parties `assert` *afin que la vérification ne détecte pas d'erreurs dans cette assertion. Par exemple, on pourrait proposer* $(x = 1 \vee x = 2) \wedge (y = 0 \vee y = 5)$ *mais il vous appartient de simuler le programme pluscal pour vérifier que jamais l'assertion que vous proposerez ne soit fausse. La solution TRUE fonctionne mais n'est pas autorisée et les expressions demandées doivent contenir une occurence de x au moins et une occurence de y.*

Exercice 5 *Voir figure* **??**
On considère des populations de clients $\mathcal{P}_i = \{P_{ij} : j \in \ \in 1..n_i\}$ *avec* $i \in 1..n$ *et* $Q_j, i \in 1..n$ *associé à chaque population : le processus* Q_i *est le serveur de la population* $\mathcal{P}_i$*. L'algorithme de la figure* **??** *met en place la gestion d'une ressource R partagée par les processus* $C_1 \dots C_p$ *via un serveur S. On décide d'utiliser cet algorithme pour gérer une ressource R partagée par toutes les populations et attribuée aux populations par leur serveur respectif quand ce serveur a le jeton. Le réseau en anneau dans la figure* **??** *explique les liens possibles entre les processus serveurs* Q_i *et les populations. La gestion de l'anneau est réalisé comme indiqué dans le fichier qring.tla de la figure* **??***. La gestion d'une population est assurée par le programme du fichier de la figure* **??***.*

Question 5.1 *On souhaite tester le protocole ring du fichier qring.tla de la figure* **??***. Expliquer pourquoi le réseau ring de qring.tla est correct et effecteur des tests que vous indiquerez dans votre fichier soit en exprimant des propriétés de sûreté ou d'invariance, soit en vérifiant l'absence de blocage. n*

Listing 1 – qquestion1.tla

```
----------------------------- MODULE question1a -----------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets
CONSTANTS N
ASSUME N % 4 = 0
(*
--algorithm algo {
variable
        canal = <<>>;
        witness = -1;
        result = -1;

\*  Macro for sending primitive: sending a message m on the fifo channel chan
macro Send(m, chan) {
    chan := Append(chan, m);
  };

\*  Macro for receivinbg  primitive: receiving
a message m on the fifo channel chan
macro Recv(v, chan) {
    await chan # <<>>;
    v := Head(chan);
    chan := Tail(chan);
  };

process (one = 1)
variable
         x = 0;
{
        w:while (x <= N) {
      a:x := x + 1;
      b:if ( x % 4 = 0) {
           c: Send(x,canal);
       };
};
d: Send(-1,canal);
};

process (two = 2)
variable s = 0,mes;
{
          w:while (TRUE) {
          a: if (canal # <<>>) {
             b:Recv(mes,canal);
                c:if (mes # -1) { d: s := s +mes;}
                else {e: goto f;};
          };
          };
          f: print <<s>>;
          g: result := s;
};

process (three = 3)
variable
         i = 0;
         s = 0;
         b = N \div 4;
{
        w:while ( i<= b) {
    a:i := i + 1;
        b: s := s +i;
       }
```

Listing 2 – qquestion2a.tla

```
---------------------------- MODULE qquestion2a ----------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets


(*
--wf
--algorithm ex3{
variables x = 0, y = 8;


process (one = 1)
{
  A:
    x := x + 1;
  B:
    y := y -1;
  C:
      assert A1;
};

process (two = 2)
{
  D:
    x := x - 1;
  E:
    y:=y+2;
   F:
    x:= x+2;
   assert A2;
};

}
end algorithm;

*)
====
```

FIGURE 2 – Programme

Question 5.2 *La ressource R ne peut être attribuée que par un processus serveur Q qui a le jeton c'est-à-dire que $v[Q] = TOKEN$ sinon NIL. Q est un des processus sur l'anneau et est numéroté de 0 à N. Modifier le processus Q afin de réaliser cette fonctionnalité d'attribution de la ressource R au processus de la population gérée par Q et répondant à une demande de la population selon le,protocole de la figure* **??**.

Question 5.3 *Détailler les propriétés de correction que doit satisfaire ce protocole et vérifier les. En particulier, il faudra montrer que le processus P d'une population donnée reçoit la ressource quand le serveur est en état de lui donner c'est-à-dire qu'il a le jeton.*

Exercice 6

La figure **??** *est un réseau de Petri modélisant le système des philosophes qui mangent des spaghetti.*

Question 6.1 *Traduire le réseau de Petri sous la forme d'un module TLA, en utilisant le fichier petri2023.tla. En particulier, il faut compléter l'initialisation.*

Question 6.2 *Est-ce que le réseau peut atteindre un point de deadlock ? Expliquez votre réponse.*

Question 6.3 *Proposer une propriété TLA pour répondre à la question suivante, en donnant des explications.*
Est-ce que deux philosophes voisins peuvent manger en même temps ?

```
------------------- MODULE examen2023q1 -----------------
EXTENDS Naturals,TLC
CONSTANTS  Places (* d\'esigne l'ensemble des places du r\'eseau de Petri  *)

VARIABLES  M  (*  la variable d'\'etat  indiquant o\`u se trouvent les jetons *)
------------------------
ASSUME
   Places \subseteq {"p11","p12","p13", ...}
------------------------
l1 ==
r1 ==
b1 ==
......

Init ==  M = [p \in Places |-> IF p \in {"p1","p2","p3","p4"} THEN 1 ELSE IF .... ]
Next == t1 \/ t2    \/ t3   \/ t4    \/ t5
```

Listing 3 – anneau1

```
--------------------- MODULE qring ----------------------
EXTENDS Integers, Naturals, Sequences, TLC
CONSTANT N,NIL,TOKEN

Remove(i, seq) == [j \in 1..(Len(seq)-1) |-> IF j < i THEN seq[j] ELSE seq[j+1]]

v0 == [i \in 0..N |->NIL]

(*
--algorithm algo {

variable
    v = v0;
    port = [i \in 0..N |-> IF i # 0 THEN <<>> ELSE <<TOKEN>>];

\* Macro for sending primitive: sending a message m on the fifo channel chan
macro Send(m, chan) {
    chan := Append(chan, m);
  };

\* Macro for receiving primitive: receiving
a message m on the fifo channel chan
macro Recv(v, chan) {
    await chan # <<>>;
    v := Head(chan);
    chan := Tail(chan);
  };

 process (q \in 0..N )
    variable mes;
  {
  s: while (TRUE) {
  cc1: Recv(mes,port[self]);
        test: if (self = 0) { pp: print <<"P[0]:",v>>;};
        cc4: v[self] := mes;
        rr: v[self]:= NIL;
        cc5: Send(TOKEN, port[(self+1) % N]);
        };
         };
         };



} \* end algorithm

*)
```



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

FIGURE 3 – Programme

Listing 4 – pop

```
---------------------- MODULE  qquestion3a
-----------------------
EXTENDS Naturals , Sequences , TLC
CONSTANT p

Remove(i , seq) == [j \in 1..(Len(seq)-1) |-> IF j < i THEN seq[j] ELSE seq[j+1]]

(*
--algorithm  algo {

variable
    requests = <<>>,  reply = [i \in 1..p |-><<>>], msgok = <<>>;

macro Send(m, chan) {
    chan := Append(chan, m);
  };

macro Recv(v, chan) {
    await chan # <<>>;  \* could also do Len(chan) > 0 ??
    v := Head(chan);
    chan := Tail(chan);
  };

 process (C \in 1..p )
    variable request = 0, mes, cs = 0;
  {
  s:  while (TRUE) {
    c1: request := 1;
    c2: Send(self,requests);
    c3: Recv(mes,reply[self]);
    c4: cs := 1;
    c5: request := 0;
    c6: Send(self,msgok);
    };
}


  process (Server = 0 )
    variables  cs=0,v;
  {
  r:  while (TRUE) {
      if (requests # <<>> /\ cs=0)
      {
            a: Recv(v,requests);
            b: cs := 1;
            c: Send(v,reply[v]);

      } else if (msgok # <<>>)
      {
            d: Recv(v,msgok);
            e: cs := 0;
      } else
      { v:skip;
      }
      ;


          };
  };
```

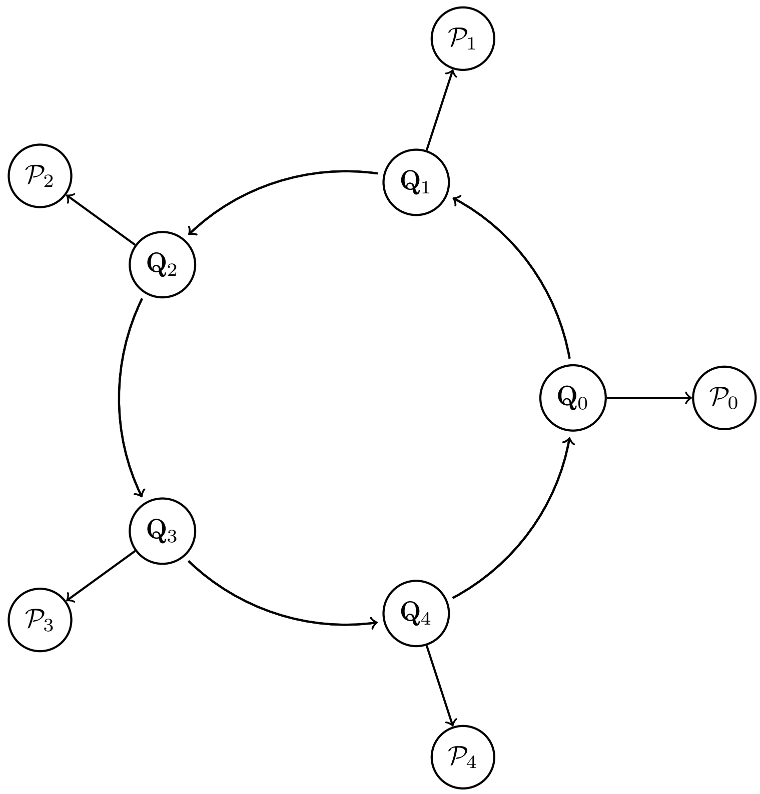


FIGURE 5 – Réseau global

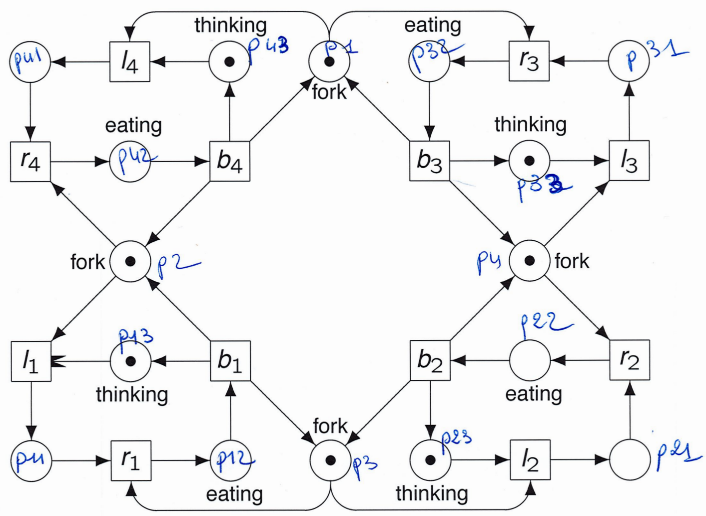


FIGURE 6 – Réseau de Petri

==

Cours Algorithmique des systèmes parallèles et distribués
Exercices
Série 3 : Exclusion Mutuelle - Dates - Estampilles
par Alessio Coltellacci et Dominique Méry
12 mars 2025

Exercice 1 *(dis-slidingwindow)*

Question 1.1 *Le protocole appelé Sliding Window Protocol est fondé sur un fenêtre qui glisse pour valider progressivement les envois reçus.*
Le protocole est donné sous la forme d'invariant avec des événements. Proposer un schéma de traduction pour cet algorithme réparti en un module TLA+.

EVENT e
  ANY
    x
  WHERE
    $G(x,u)$
  THEN
    $u := f(x,u)$
  END

$\leadsto$

$$BA(e) \stackrel{def}{=} \exists x. \begin{pmatrix} \wedge\ G(x,u) \\ \wedge\ u' = f(x,u) \end{pmatrix}$$

Question 1.2 *Proposer un schéma de traduction pour un algorithme réparti en TLA$^+$.*

Question 1.3 *Reprendre la solution précédente pour modéliser chan comme un buffer de taille la taille de la fenêtre.*

AXIOMS
  $axm1 : n \in \mathbb{N}_1$
  $axm2 : IN \in \mathbb{N} \mathrel{+\!\!\!\rightarrow} D$
  $axm3 : dom(IN) = 0 \mathinner{..} n$
  $axm4 : l \in \mathbb{N}$
  $axm5 : l \leq n$

VARIABLES OUT, i, ack, got, b

INVARIANTS

$inv1 : OUT \subseteq IN$

$inv2 : 0 \,..\, i{-}1 \lhd OUT = 0 \,..\, i{-}1 \lhd IN$

$inv3 : i \in 0 \,..\, n{+}1$

$inv4 : ack \cup got \subseteq i \,..\, i{+}l \cap 0 \,..\, n$

$inv5 : ack \subseteq dom(OUT)$

$inv1 : OUT \in \mathbb{N} \nrightarrow D$

$inv2 : i \in 0 \,..\, n{+}1$

$inv3 : 0 \,..\, i{-}1 \subseteq dom(OUT) \wedge dom(OUT) \subseteq 0 \,..\, n$

$inv8 : ack \subseteq \mathbb{N}$

$inv10 : got \subseteq \mathbb{N}$

$inv13 : got \subseteq dom(OUT)$

$inv14 : ack \subseteq dom(OUT)$

$inv16 : 0 \,..\, i{-}1 \lhd OUT = 0 \,..\, i{-}1 \lhd IN$

EVENT INITIALISATION

BEGIN

$act1 : OUT := \varnothing$

$act2 : i := 0$

$act5 : ack := \varnothing$

$act6 : got := \varnothing$

$act8 : b := \varnothing$

END

EVENT send

ANY

j

WHERE

$grd1 : j \in i \,..\, i{+}l$

$grd2 : j \leq n$

$grd3 : j \notin got$

$grd4 : j{-}i \in 0 \,..\, l$

THEN

$act3 : b(j{-}i) := IN(j)$

END

EVENT receive

ANY

j

WHERE

$grd2 : j \in i \,..\, i{+}l$

$grd3 : j{-}i \in dom(b)$

THEN

$act2 : ack := ack \cup \{j\}$

$act3 : OUT(j) := b(j{-}i)$

END

EVENT receiveack
ANY
k
WHERE
$grd1 : k \in ack$
THEN
$act1 : got := got \cup \{k\}$
$act2 : ack := ack \setminus \{k\}$
END

EVENT sliding
ANY
c
WHERE
$grd1 : got \neq \varnothing$
$grd3 : i \in got$
$grd4 : i+l < n$
$$grd5 : \left(\begin{array}{l} c \in 0\,..\,l \nrightarrow D \\ \wedge dom(c) = \{u | u \in 0\,..\,l-1 \wedge u+1 \in dom(b)\} \\ \wedge (\forall o \cdot o \in dom(b) \wedge o \neq 0 \Rightarrow o-1 \in dom(c) \wedge c(o-1) = b(o)) \end{array} \right)$$
THEN
$act1 : i := i+1$
$act2 : got := got \setminus \{i\}$
$act3 : ack := ack \setminus \{i\}$
$act5 : b := c$
END

EVENT emptywindow
ANY
c
WHERE
$grd1 : got \neq \varnothing$
$grd2 : i \in got$
$grd3 : i{+}l \geq n$
$grd4 : i \leq n$
$grd5 : \left(\begin{array}{l} c \in 0\,..\,l \nrightarrow D \\ \wedge dom(c) = \{u | u \in 0\,..\,l{-}1 \wedge u{+}1 \in dom(b)\}) \\ \wedge(\forall o \cdot o \in dom(b) \wedge o \neq 0 \Rightarrow o{-}1 \in dom(c) \wedge c(o{-}1) = b(o)) \end{array}\right)$
THEN
$act1 : i := i{+}1$
$act2 : got := got \setminus \{i\}$
$act3 : ack := ack \setminus \{i\}$
$act5 : b := c$
END

EVENT completion

WHEN
$grd1 : i = n{+}1 \wedge got = \varnothing$
THEN
skip
END

EVENT loosingchan

ANY
j
WHERE
$grd1 : j \in i\,..\,i{+}l$
$grd3 : j \notin got$
$grd4 : j{-}i \in dom(b)$
THEN
$act3 : b := \{j{-}i\} \lhd b$
END

EVENT loosingack

ANY
k
WHERE
$grd1 : k \in ack$
THEN
$act1 : ack := ack \setminus \{k\}$
END

Exercice 2 *(plusboulanger.tla)*
L'algorithme BAKERY *résout le problème de l'xclusion mutuelle pour un système centralisé. Vous pouvez récupérer le fichier tla crrespondant à cet exemple sur le*

site.

Question 2.1 *Poser une question sur l'accessibilité du processsus 1 en section critique.*

Question 2.2 *Poser une question sur l'accessibilité du processsus 2 en section critique.*

Question 2.3 *En bornant les valeurs de $y1$ et $y2$, montrer que la solution retenue satisfait la propriété d'exclusion mutuelle que vous énoncerez.*

Question 2.4 *Expliquez et justifiez expérimentalement que les valeurs de $y1$ et $y2$ croissent.*

Exercice 3 *dis-ricartagrawalav0*
Le fichier de l'algorithme de Ricart et Agrawala est sur le site.

Question 3.1 *Modéliser l'algorithme de Ricart et Agrawala en TLA$^+$.*

Question 3.2 *Enoncer la propriété à vérifier.*

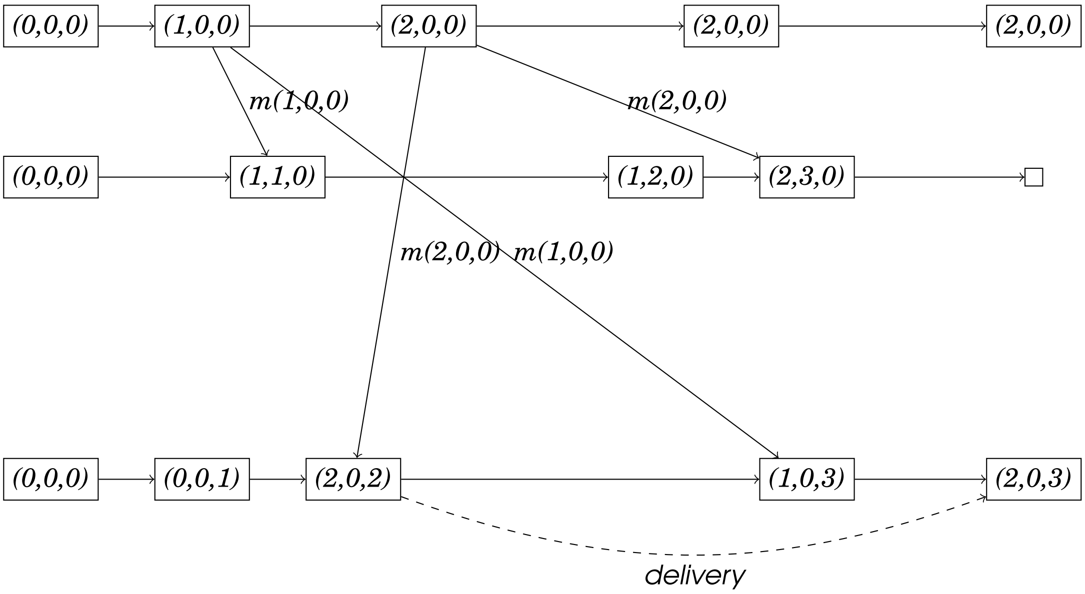
Question 3.3 *Poser une question montrant que toute demande de section critique par un processus P sera servie.*

Question 3.4 *Modifier les actions pour supprimer le sémaphore ey montrer qu'il y a un interblocage.*

Question 3.5 *En vous aidant de la version algorithmique de Ricart et Agrawala, écrire un algorithme PlusCal.*

Question 3.6 *En reprenant l'algorithme de Ricart et Agrawala, on peut le simplifier pour construire l'algorithme de Carvalho et Roucairol. Modéliser l'algoritme de Carvalho et Roucairol.*

Exercice 4 *(vecteurs d'horloge) (pluscal_vc.tla, vector_clock.tla)*
Soit un ensemble de processus P communiquant par messages en groupe. Cela signifie que les processus d'un groupe de P peuvent envoyer des messages à un groupe. On s'intéresse à l'ordre FIFO c'est-à-dire la propriété suivante : si un processus p d'un groupe g de P envoie un message m1 avant un message m2, aucun processus correct de g ne livrera le message m2 avant le message m1.



Dans cet exemple, le message 2 est reçu avant le message 1 et doit donc être livré plus trad après la livraison du message 1. L'algorithme FBCAST résout ce problème en livrant selon la règle des estampilles. En fait le processus 3 attend un message avec une estampille dont le champ de l'émetteur vaut cette valeur. Dans notre exemple, 3 attend un message de 1 avec 1 et quand il reçoit le second message avec la valeur 2, il attend.
Ecrire un ensemble d'opérations de communications mettant en oeuvre ce mécanisme.

Exercice 5 *(Protocole CBCAST)*
Le protocole CBCAST utilise les vecteurs d'horloges pour livrer les messages en respectant l'ordre FIFO des messages envoyés : si un processus P envoie un message m1 puis m2 à un processus Q, alors le protocole livrera d'abord m1 puis m2.
Le principe général est le suivant :
- *Le vecteur d'horloges* $VC \in 1..n \longrightarrow (1..n \longrightarrow \mathbb{N}$ *est initialisé à 0 pour toutes composantes.*
- *Si un processus i envoie un message m,* $VC(i)[i]$ *est incrémenté de 1.*
- *Tout message envoyé m est estampillé par* $VC(i)$ *:* $TM(m) = VC(i)$ *où* $TM \in MES \nrightarrow \mathbb{N}$
- *Quand un processus j reçoit un message estampillé* m*, il met à jour l'horloge de j comme suit :*
  $\forall k \in 1..n : VC(j)[k] = Max(VC(j)[k], TM(m)[k]$

Pour le protocole CBCAST, on a des restrictions sur la livraison effective des messages :

— *Si le processus i reçoit un message m, il le,place en file d'attente CB-QUEUE en attendant que le condition suivante soit vraie :*

$\forall k \in 1..n : \begin{cases} TM(m)[i] = VC(i)[k]{+}1 \\ TM(m)[k] \leq VC(i)[k] \end{cases}$

— *La livraison du message met à jour* $VC(i)$.