Cours Algorithmique des systèmes parallèles et distribués
Exercices
Série :PlusCal pour la programmation répartie ou concurrente (I)
par Dominique Méry
5 février 2026

Exercice 1 *(pluscaltut1.tla)*
Etudier, complÃ©ter et analyser le programme PlusCal suivant :

```
------------------------------ MODULE pluscaltut1q ------------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets
(*
--wf
--algorithm Tut1 {
variables x = 0;

process (one = 1)
{
  A: assert x \in ???;
     x := x - 1;
  B: assert x \in ???? ;
     x := x * 3;
   BB: assert x \in ???;
};

process (two = 2)
{
  C: assert x \in ???;
     x := x + 1;
  D:
     assert x \in ??;
};

}
end algorithm;

*)

safepc == pc[1]="Done" /\ pc[2]="Done"  => ??


====
```

Exercice 2 *(pluscaltut2.tla)*
Etudier, complÃ©ter et analyser le programme PlusCal suivant :

```
------------------------------ MODULE pluscaltut2q ------------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets

(*
--algorithm Tut2 {
variables x = 0;

process (one = 1)

variables temp
{

 A:
         temp := x + 1;


         x := temp;

};


process (two = 2)

variables temp
{
  B:
         temp := x + 1;

         x:= temp;
};
}
end algorithm;

*)

safepc == pc[1]="Done" /\ pc[2]="Done"  => ??
====
```

Exercice 3 *(pluscaltut3.tla)*
Etudier le programme PlusCal suivant :

```
------------------------------ MODULE pluscaltut3q ------------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets
(*
--algorithm Tut3 {
```

```
variables x = 0;

process (one = 1)
{
  A:
     x := x + 1;
  B:
     await  x = 1;
  C:
     print <<"x=",x>>;
};

process (two = 2)
{
  D:
     await x = 1;
  E:
     assert x = 1;
  F:
     x := x -2;

};

}
end algorithm;

*)
```

==

Exercice 4 *pluscaltut4.tla*
Ecrire un programme PlusCal qui traduit le protocole suivant :

- *S envoie une valeur val à R*
- *R reÃ§oit la mÃªme valeur val*

Exercice 5 *pluscaltut5.tla*
Ecrire un programme PlusCal qui calcule la fonction factorielle de la façon suivante :

- *Un processus P1 calcule* $1 \times 2 \times 3 .... \times k1$
- *Un processus P2 calcule* $k2 \times (k2+1) \times \ldots \times N$
- *Les processus stoppent quand la condition* $k1 < k2$ *est fausse*

Exercice 6 *pluscaltut6.tla*
Ecrire un programme PlusCal qui calcule la fonction L^K *la façon suivante :*

- *Un processus P1 calcule* $L\times\ldots\times L$ ***k1*** *fois.*
- *Un processus P2 calcule* $L\times\ldots\times L$ ***k2*** *fois.*
- *Les processus P3 stoppent quand la condition* $k1 + K2 < L$ *est fausse*

Cours Algorithmique des systèmes parallèles et distribués
Exercices
Série : PlusCal pour la programmation répartie ou concurrente (II)
par Dominique Méry
5 février 2026

Exercice 1 *pluscaltut7.tla*
Compléter le module pluscaltut7q.tla en proposant une assertion Q1 correcte.

```
---------------------------- MODULE pluscaltut7q ----------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets
(*

--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 *pluscaltut8.tla*

Compléter le module pluscalappaspd33.tla en proposant deux assertions R1 et R2 correctes.

```
----------------------------- MODULE pluscaltut8q -----------------------------
EXTENDS Integers, Sequences, TLC, FiniteSets
(*

--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 *pluscaltut9.tla Fig : 1*
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 *pluscaltut10.tla voir Figure 2*
Soit le petit module pluscaltut10.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.*

Cours Algorithmique des systèmes parallèles et distribués Exercices Série 1 Modélisation, programmation et vérification en TLA$^+$ par Dominique Méry 5 février 2026

Modélisation et vérification avec TLA$^+$

RAPPELS

Un réseau de Petri est un uple R=(S,T,F,K,M,W) tel que

— S est l'ensemble (fini) des places.
— T est l'ensemble (fini) des transitions.
— $S \cap T = \varnothing$
— F est la relation du flôt d'exécution : $F \subseteq S \times T \cup T \times S$
— K représente la capacité de chaque place : $K \in S \to Nat$.

Listing 1 – pluscaltut9.tla

```
---------------------------- MODULE pluscaltut09 ----------------------------
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 – pluscaltut10.tla

```
---------------------------- MODULE pluscaltut10 ----------------------------
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

— M représente le initial marquage chaque place :
M $\in$ S $\rightarrow$ Nat et vérifie la condition $\forall$ s $\in$ S : M(s) $\leq$ K(s).
— W représente le poids de chaque arc : W $\in$ F $\rightarrow$ Nat
— un marquage M pour R est une fonction de S dans Nat :
M $\in$ S $\rightarrow$ Nat et respectant la condition $\forall$ s $\in$ S : M(s) $\leq$ K(s).
— une transition t de T est activable à partir de M un marquage de R si

1. $\forall$ s $\in$ { s' $\in$ S | (s',t) $\in$ F } : M(s) $\geq$ W(s,t).
2. $\forall$ s $\in$ { s' $\in$ S | (t,s') $\in$ F } : M(s) $\leq$ K(s) - W(s,t).

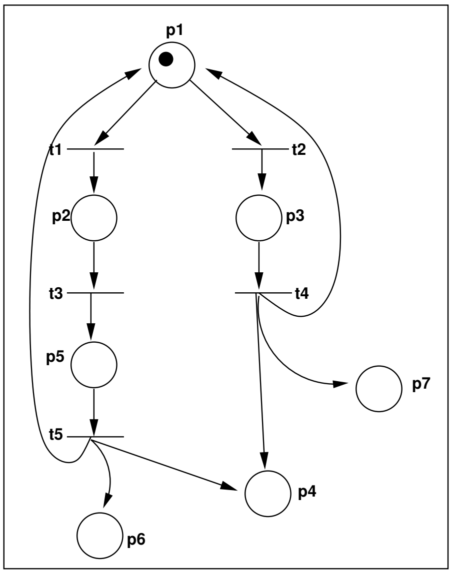
— Pour chaque transition t de T, Pre(t) est l'ensemble des places conduisant à t et Post(t) est l'ensemble des places pointées par un lien depuis t :
Pre(t) = { s' $\in$ S : (s',t) $\in$ F } et Post(t) = { s' $\in$ S : (t,s') $\in$ F }
— Soit une transition t de T activable à partir de M un marquage de R :

1. $\forall$ s $\in$ { s' $\in$ S | (s',t) $\in$ F } : M(s) $\geq$ W(s,t).
2. $\forall$ s $\in$ { s' $\in$ S | (t,s') $\in$ F } : M(s) $\leq$ K(s) - W(s,t).

— un nouveau marquage M' est défini à partir de M par : ' $\forall$ s $\in$ S,

$$M'(s) = \begin{cases} M(s)-W(s,t), \text{ si } s \in Pre(t) \text{ - } Post(t) \\ M(s)+W(t,s), \text{ si } s \in Post(t) \text{ - } Pre(t) \\ M(s)-W(s,t)+W(t,s), \text{ si } s \in Pre(t) \cap Post(t) \\ M(s), \text{ sinon} \end{cases}$$

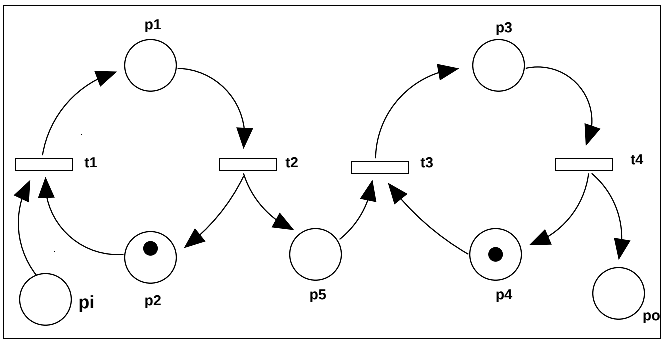
Exercice 1 *(petri13.tla)*
Soit le réseau de Petri suivant :



Question 1.1 *Modéliser ce réseau de Petri avec TLA⁺.*

Question 1.2 *Etudier ce réseau en proposant et en vérifiant des invariants Ã l'aide des outils.*

Exercice 2 *(petri10.tla)*
On considère le réseau suivant :



Question 2.1 *Traduire ce réseau en un module TLA$^+$.. Pour cela, on donnera la définition des quatre transitions* $t1$, $t2$, $t3, t4$. *On ne tiendra pas compte de la capacité des places : les places ont une capacité d'au plus un jeton, sauf la place pi qui peut contenir* N *jetons, la place p5 peut contenir au plus B jetons et la place po peut contenir au plus* Q.

Question 2.2 *Donner une relation liant les places po,p1,p3,p5,pi et la valeur* N. *Justifier la réponse.*

Question 2.3 *Si on suppose que la place po peut contenir au plus* Q *jetons, donnez une condition sur* Q *pour que tous les jetons de pi soient consommés un jour. Justifier la réponse.*

Question 2.4 *Expliquer ce que modélise ce réseau de Petri.*

Exercice 3 *(petri14.tla)*
La figure 3 est un réseau de Petri modélisant le système des philosophes qui mangent des spaghetti.

Question 3.1 *Traduire le réseau de Petri sous la forme d'un module TLA.*

Question 3.2 *Est-ce que le réseau peut atteindre un point de deadlock ? Expliquezr votre réponse.*

Question 3.3 *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 ?

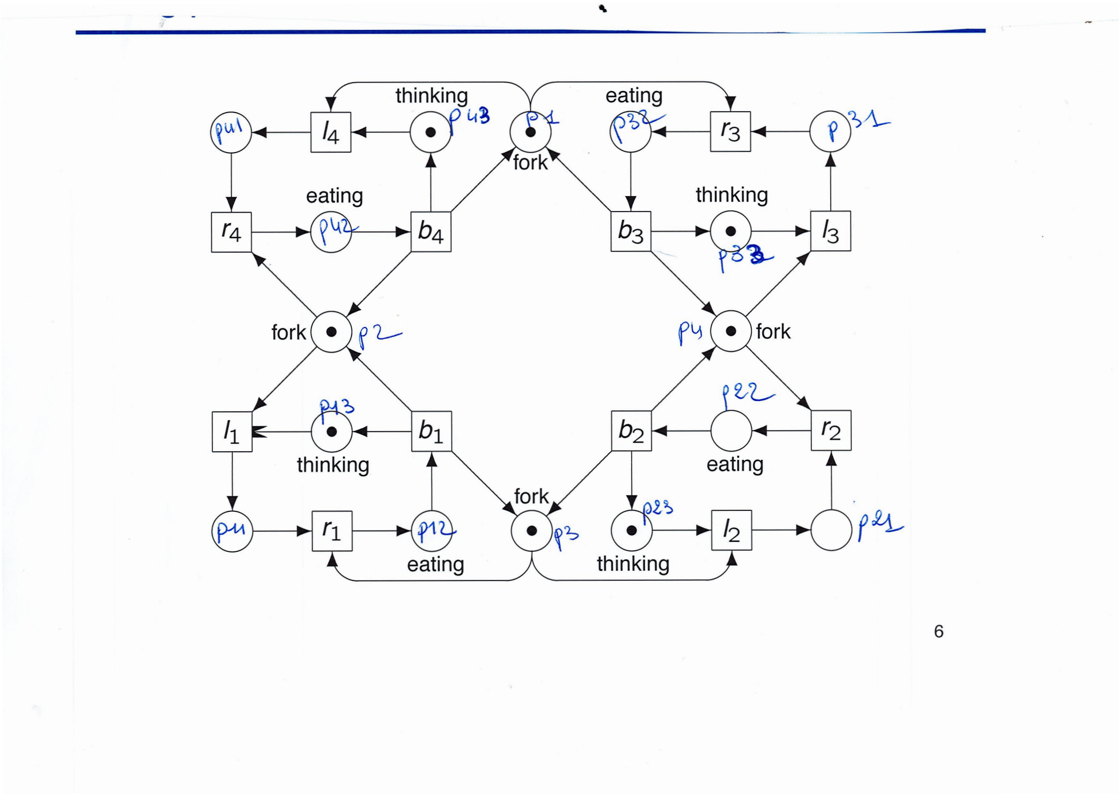


FIGURE 3 – Réseau de Petri

Exercice 4 *(disapp_td1_ex1.tla)*

Question 4.1 *Modéliser sous forme d'un module TLA$^+$ le réseau de Petri de la figure 4. Donner une instanciation possible des constantes. Préciser les conditions initiales correspondant à un système avec cinq (5) processeurs et deux (2) bus.*

Question 4.2 *On désire analyser le comportement de ce réseau et, pour cela, on souhaite savoir si la place $p5$ contiendra au moins un jeton. Expliquer comment on doit procéder pour obtenir une réponse en un temps fini. Préciser le message donné par le système TLAPS.*

Question 4.3 *Est-ce que le réseau peut atteindre un point de deadlock ?*

Question 4.4 *Enoncez trois propriétés de sûreté de ce réseau établissant une relation entre au moins deux places.*

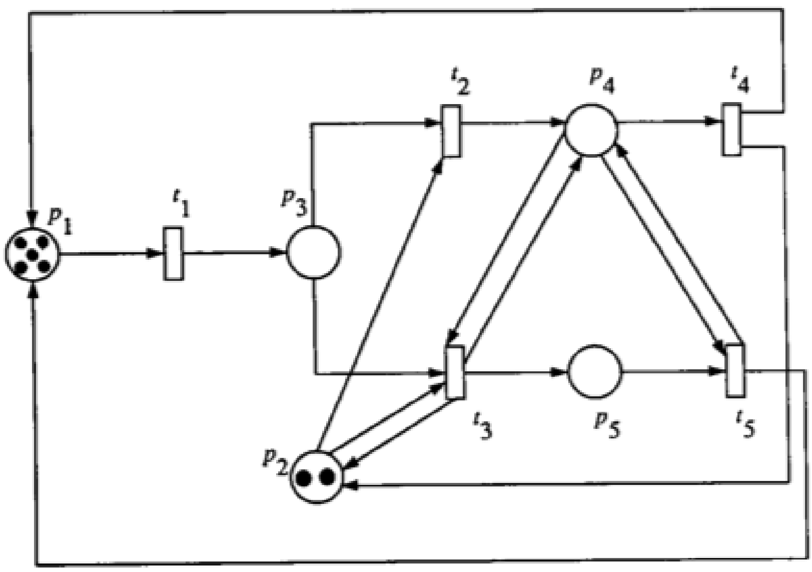


Fig. 14. A Petri-net model of a multiprocessor system, where tokens in p_1 represent active processors, p_2 available buses, p_3, p_4, and p_5 processors waiting for, having access to, queued for common memories, respectively.

FIGURE 4 – Réseau de Petri