

Cours - Systèmes de Transition

Louis Thevenet

Table des matières

1. Mise en pratique : La factorielle	2
2. Homme-Loup-Mouton-Chou	2
3. Problème Lecteurs/Rédacteurs	2
3.1. Preuve axiomatique de ExclusionLR	2
3.2. Raffinement	2
4. L'algorithme de Peterson	2
5. Token Ring	2
5.1. Contre-exemple 1	2
5.2. Contre-exemple 2	2
5.3. Raffinage : Tableau de jetons	2
5.4. Raffinage : Modéliser les canaux de communication	2

1. Mise en pratique : La factorielle

```
1 ----- MODULE Fact0 -----
2
3 EXTENDS Naturals
4 CONSTANT N
5 VARIABLE res
6
7 Init == res = Fact[N]
8 Next == UNCHANGED res (*ou FALSE*)
9 Spec == Init /\land [Next]_res
10 =====
```

Liste 1. – 0 transition

```
1 ----- MODULE Fact1 -----
2
3 EXTENDS Naturals
4 CONSTANT N
5 ASSUME N \in Nat
6 VARIABLES res, i
7
8 Init ==
9   /\ res = 1
10  /\ i = 1
11
12 Mult ==
13   /\ i <= N
14   /\ res' = res * i
15   /\ i' = i + 1
16
17 Next == Mult
18
19 Spec == Init /\land [Next]_res,i
20 =====
```

Liste 2. – Avec transitions

```
1 ----- MODULE Fact1 -----
2
3 EXTENDS Naturals
4 CONSTANT N
5 ASSUME N \in Nat
6 VARIABLES res, factors
7
8 Init ==
9   /\ res = 1
10  /\ factors = 1..N
11
12 Mult(i) ==
13   /\ res' = res * i
14   /\ factors' = factors \ {i}
15
16 Next == \E i \in factors : Mult (i)
17
18 Spec == Init /\land [Next]_res,factors
19 =====
```

Liste 3. – Sans ordre particulier

```
1 ----- MODULE Fact1 -----
2
3 EXTENDS Naturals
4 CONSTANT N
5 ASSUME N \in Nat
6 VARIABLES res, factors
7
8 Init ==
9   /\ res = 1
10  /\ factors = 1..N
11
12 Mult(I) ==
13   /\ res' = (*on multiplie les éléments de I à res*)
14   /\ factors' = 1..N
15
16 Next == \E I \in SUBSET factors : Mult (I)
17
18 Spec == Init /\land [Next]_res,factors
19 =====
```

Liste 4. – Sans ordre particulier

2. Homme-Loup-Mouton-Chou

On doit les faire passer d'une rive à l'autre d'une rivière.

- Il faut un homme pour ramer
- Sans la surveillance de l'homme
 - le mouton mange le chou
 - le loup mange le mouton

```
1 ----- MODULE hlmc -----
2
3 VARIABLES h, m, c, l
4 RIVES == {"G", "D"}
5
6 Inv(r) ==
7   IF r = "G"
8   THEN "D"
9   ELSE "G"
10
11 TypeInvariant == {h, l, m, c} \subseq RIVES
12
13 Init ==
14   /\ h = "G"
15   /\ l = "G"
16   /\ m = "G"
17   /\ c = "G"
18   (*\ PasMiam*)
19
20 PasMiam ==
21   /\ (l = m => h = m)
22   /\ (c = m => h = m)
23
24 MoveH ==
25   /\ h' = Inv(h)
26   /\ UNCHANGED <<l, m, c>>
27   /\ PasMiam'
28
29 MoveHL ==
30   /\ h' = Inv(h)
31   /\ l' = Inv(l)
32   /\ h = l
33   /\ UNCHANGED <<m, c>>
34   /\ PasMiam'
35
36 MoveHM ==
37   /\ h' = Inv(h)
38   /\ m' = Inv(m)
39   /\ h = m
40   /\ UNCHANGED <<l, c>>
41   /\ PasMiam'
42
43 MoveHC ==
44   /\ h' = Inv(h)
45   /\ c' = Inv(c)
46   /\ h = c
47   /\ UNCHANGED <<l, m>>
48   /\ PasMiam'
49
50 Next ==
51   \/\ MoveH
52   \/\ MoveHL
53   \/\ MoveHM
54   \/\ MoveHC
55
56 Spec ==
57   /\ Init
58   /\ [Next]_<<h,l,m,c>>
59
60 But == [](- {h,l,m,c} = {"D"})
61 =====
```

Liste 5. – Sans ordre particulier

3. Problème Lecteurs/Rédacteurs

```
1 MODULE LR0
2 EXTENDS Naturals
3 VARIABLES nl, nr
4
5 TypeInvariant ==
6   /\ nl \in Nat
7   /\ nr \in 0..1
8
9 Initial ==
10  /\ nl = 0
11  /\ nr = 0
12
13 EntrerL ==
14  /\ nr = 0
15  /\ nl' = nl+1
16  /\ UNCHANGED <<nr>>
17
18 SortirL ==
19  /\ nl > 0
20  /\ nl' = nl - 1
21  /\ UNCHANGED <<nr>>
22
23 EntrerR ==
24  /\ nl = 0
25  /\ nr = 0
26  /\ UNCHANGED <<nl>>
27  /\ nr' = 1
28
29 SortirR ==
30  /\ nr = 1
31  /\ UNCHANGED <<nl>>
32  /\ nr' = 0
33
34 Next ==
35   \/\ EntrerL
36   \/\ SortirL
37   \/\ EntrerR
38   \/\ SortirR
39
40 Spec ==
41   /\ Initial
42   /\ [Next]_nl, nr
43   /\ WF_nl, nr(SortirL)
44   /\ WF_nl, nr(SortirR)
45
46 ExclusionLR ==
47   [](nl = 0 /\ nr = 0)
48
49 (*EclusionR ==
50   [](nr \in 0..1)
51   (* déjà dans invariant de type*)
52 *)
```

Liste 6. – Lecteurs/Rédacteurs 0

3.1. Preuve axiomatique de ExclusionLR

- A l'état initial

$$\text{Initial} \Rightarrow \text{nl} = 0 \vee \text{nr} = 0 \vee$$

- A chaque transition

$$(\text{nl} = 0 \vee \text{nr} = 0) \wedge [\text{Next}]_{\text{nl}, \text{nr}} \Rightarrow \text{nl}' = 0 \vee \text{nr}' = 0$$

- on étudie chaque transition séparément
 - dégalement
$$(\text{nl} = 0 \vee \text{nr} = 0) \wedge \text{nl}' = \text{nl} \wedge \text{nr}' = \text{nr} \Rightarrow \text{nl}' = 0 \vee \text{nr}' = 0$$
 - EntrerL ✓
$$(\text{nl}=0 \vee \text{nr} = 0) \wedge \text{nr} = 0 \wedge \text{nl}' = \text{nl} + 1 \wedge \text{nr}' = \text{nr} + 1 \Rightarrow \text{nl}' = 0 \vee \text{nr}' = 0$$
 - SortirL ✓
$$(\text{nl}=0 \vee \text{nr} = 0) \wedge \text{nl} > 0 \wedge \text{nl}' = \text{nl} - 1 \wedge \text{nr}' = \text{nr} + 1 \Rightarrow \text{nl}' = 0 \vee \text{nr}' = 0$$
 - EntrerR ✓
 - SortirR ✓

3.2. Raffinement

```
1 MODULE LR1
2 EXTENDS Naturals
3 VARIABLES nl, nr, ndemr (*nombre demande rédacteurs*)
4
5 TypeInvariant ==
6   /\ nl \in Nat
7   /\ nr \in 0..1
8   /\ ndemr \in Nat
9
10 Initial ==
11  /\ nl = 0
12  /\ nr = 0
13  /\ ndemr = 0
14
15 EntrerL ==
16  /\ nr = 0
17  /\ nl' = nl+1
18  /\ UNCHANGED <<nr>>
19  /\ UNCHANGED <<ndemr>>
20
21 SortirL ==
22  /\ nl > 0
23  /\ nl' = nl - 1
24  /\ UNCHANGED <<nr>>
25  /\ UNCHANGED <<ndemr>>
26
27 EntrerR ==
28  /\ nl = 0
29  /\ nr = 0
30  /\ UNCHANGED <<nl>>
31  /\ nr' = 1
32  /\ ndemr > 0
33  /\ ndemr' = ndemr - 1
34
35 SortirR ==
36  /\ nr = 1
37  /\ UNCHANGED <<nl>>
38  /\ nr' = 0
39  /\ UNCHANGED <<ndemr>>
40
41 DemanderR ==
42  /\ ndemr' = ndemr + 1
43  /\ UNCHANGED <<nr, nl>>
44
45 Next ==
46   \/\ EntrerL
47   \/\ SortirL
48   \/\ EntrerR
49   \/\ SortirR
50   \/\ DemanderR
51
52 Spec ==
53   /\ Initial
54   /\ [Next]_nl, nr
55   /\ WF_nl, nr(SortirL)
56   /\ WF_nl, nr(SortirR)
57   /\ WF_nl, nr(EntrerR)
58
59 ExclusionLR ==
60   [](nl = 0 /\ nr = 0)
61
62 (*EclusionR ==
63   [](nr \in 0..1)
64   (* déjà dans invariant de type*)
65 *)
```

Liste 7. – Lecteurs/Rédacteurs 1

LR1 est-il un raffinement de LR0 ? Oui car les variables sont les mêmes et les actions sont aussi les mêmes (« raffinement de déterminisme ») ⇒ exclusion est préservée adms LR1

4. L'algorithme de Peterson

Il s'agit d'exclusion mutuellement entre deux processus.

```
1 bool demande[2];
2 int tour;
3
4 // Pour le processus i dans {0, 1}
5 demande[i] = true;
6 tour = 1 - i;
7 while (demande[1-i] && tour == 1-i) {
8   // attendre
9 }
10 demande[i] = false;
11 // section critique
```

Liste 8. – Algorithme de Peterson

Le tableau demande est « auxiliaire », sa valeur est déterminée par l'endroit du programme où on se trouve.

En général, ce type d'algorithme se décompose de la façon suivante :



```
1 MODULE Peterson
2 EXTENDS Naturals, FiniteSets
3
4 VARIABLES demande, tour, etat
5
6 TypeInvariant ==
7   /\ demande \in {0..1 -> BOOLEAN}
8   /\ tour \in 0..1
9   /\ etat \in {"T", "H", "E"}
10
11 Initial ==
12  /\ demande = [i \in 0..1 |> FALSE]
13  /\ tour = 0
14  /\ etat = [ i \in 0..1 |> "T"]
15
16 Demander(i) ==
17  /\ etat[i] = "T"
18  /\ etat' = [etat EXCEPT ![i] = "H"]
19  /\ demande' = [demande EXCEPT ![i] = TRUE]
20  /\ tour' = 1 - i
21
22 Sortir(i) ==
23  /\ etat[i] = "E"
24  /\ etat' = [etat EXCEPT ![i] = "T"]
25  /\ demande' = [demande EXCEPT ![i] = FALSE]
26  /\ tour' = tour
27
28 Entrer(i) ==
29  /\ etat[i] = "H"
30  /\ etat' = [etat EXCEPT ![i] = "E"]
31  /\ \neg demande[1-i] /\ tour = i
32
33 Next == \E i \in 0..1 : Entrer(i) /\ Demander(i) /\ Sortir(i)
34
35 Spec ==
36   /\ Initial
37   /\ [Next]_demande, tour, etat
38   /\ forall i \in 0..1 : WF_{demande, tour, etat}(Sortir(i))
39   /\ WF_{demande, tour, etat}(Demander(i))
40
41 ExclusionMutuelle ==
42   [](~ (etat[0] = "E" /\ etat[1] = "E"))
43   (* ou *)
44   [](\forallall i, j \in 0..1 : (etat[i] = "E" /\ etat[j] = "E" => i = j))
45   (* ou *)
46   []((Cardinality{(i \in 0..1 : etat[i] = "E")} <= 1)
47
48 AbsenceDeFamine ==
49   \forallall i \in 0..1 : etat[i] = "H" -> etat[i] = "E"
50
51 DemandeMaintenue ==
52   \forallall i \in 0..1 : [](etat[i] = "H" => etat[i]' \in {"H", "E"})
53
54 AbsenceDeDeadlock ==
55   (i \in 0..1 | etat[i] = "H") != \emptyset -> [] \in 0..1 | etat[i] = "E" !
56
57 \emptysetset
58   (* ou *)
59   \forallall i \in 0..1 : etat[i] = "H" => \E j \in 0..1 : etat[j] = "E"
```

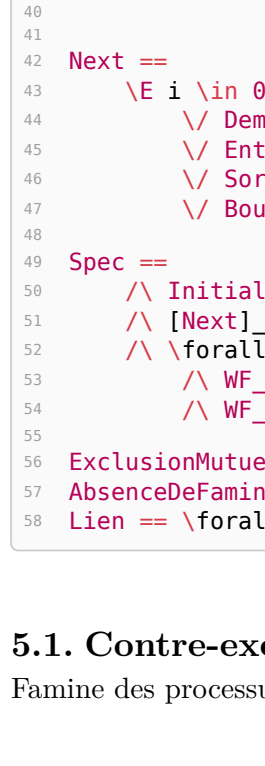
Liste 9. – Algorithme de Peterson en TLA+

5. Token Ring



Le token se déplace de processus en processus, c'est un problème de type exclusion mutuelle.

Chaque processus a 3 états : Thinking, Hungry et Eating



```
1 MODULE TokenRing0
2 EXTENDS Naturals, FiniteSets
3 CONSTANT N
4 ASSUME N \in Nat
5
6 ETAT == {"T", "H", "E"}
7
8 VARIABLES == etat, token
9
10 TypeInvariant ==
11   /\ etat \in {0..N-1 -> ETAT}
12   /\ token \in 0..N-1
13
14 Initial ==
15   /\ etat = [i \in 0..N-1 |> "T"]
16   /\ token \in 0..N-1 (* ou juste 0 *)
17
18 Demander(i) ==
19   /\ etat[i] = "T"
20   /\ etat' = [etat EXCEPT ![i] = "H"]
21   /\ UNCHANGED token
22
23 Sortir(i) ==
24   /\ etat[i] = "E"
25   /\ etat' = [etat EXCEPT ![i] = "T"]
26   /\ UNCHANGED token
27
28 Entrer(i) ==
29   /\ etat[i] = "H"
30   /\ etat' = [etat = etat EXCEPT ![i] = "E"]
31   /\ token = i
32   /\ UNCHANGED token (* important *)
33
34 Bouger(i) ==
35   /\ token = i
36   /\ token' = (i+1) % N
37   /\ UNCHANGED etat
38   /\ etat[i] != "E"
39   (* /\ etat[i] != "E" (*on vire ça*) *)
40
41 Next ==
42   \E i \in 0..N-1:
43     \/\ Demander(i)
44     \/\ Entrer(i)
45     \/\ Sortir(i)
46     \/\ Bouger(i)
47
48 Spec ==
49   /\ Initial
50   /\ [Next]_etat, token
51   /\ forall i \in 0..N-1:
52     /\ WF_{etat, token}(Sortir(i))
53     /\ WF_{etat, token}(Entrer(i))
54
55 ExclusionMutuelle == [(Cardinality{(i \in 0..N-1: etat[i] = "E"})} <= 1)]
56 AbsenceDeFamine == \forallall i \in 0..N-1: etat[i] = "H" -> etat[i] = "E"
57 Lien == \forallall i, j \in 0..N-1: [etat[i] = "E" => etat[j] = "E"
58 TokenUnique == \forallall i, j \in 0..N-1: token[i] /\ token[j] => i = j
59
60 ]
```

Liste 10. – Token Ring 0

5.1. Contre-exemple 1

Famine des processus autre que 0.

$$\text{token} = 0$$
$$(\text{Demander}(0) \rightarrow \text{Entrer}(0) \rightarrow \text{Sortir}(0))^\omega$$

On décide de pousser le token dehors.

```
1 Sortir(i) ==
2   /\ etat[i] = "E"
3   /\ etat' = [etat EXCEPT ![i] = "T"]
4   (* /\ UNCHANGED token (*on vire ça*) *)
```

Liste 11. – Nouveau Sortir(i)

5.2. Contre-exemple 2

Le jeton bouge très vite et les processus ne peuvent pas accéder à leur ressource.

$$\text{jeton} = 0$$

$$(\text{Demander}(0) \rightarrow \text{Bouger}(0) \rightarrow \text{Bouger}(1) \rightarrow \dots \rightarrow \text{Bouger}(N-1))^\omega$$

On ajoute une condition à Bouger

```
1 Bouger(i) ==
2   /\ token = i
3   /\ token' = (i+1) % N
4   /\ UNCHANGED etat
5   /\ etat[i] = "T"
6   (* /\ etat[i] != "E" (*on vire ça*) *)
```

Liste 12. – Nouveau Bouger(i)

Au lieu de modifier Sortir et Bouger, on aurait pu ajouter de l'équité plus forte que faible pour forcer l'exécution quand c'est nécessaire.

5.3. Raffinage : Tableau de jetons

```
1 MODULE TokenRing1
2 EXTENDS Naturals, FiniteSets
3 CONSTANT N
4 ASSUME N \in Nat
5
6 ETAT == {"T", "H", "E"}
7
8 VARIABLES == etat, token
9
10 TypeInvariant ==
11   /\ etat \in {0..N-1 -> ETAT}
12   /\ token \in {0..N-1 -> BOOLEAN}
13
14 Initial ==
15   /\ etat = [i \in 0..N-1 |> "T"]
16   /\ \E i \in 0..N-1: token = [j \in 0..N-1 |> i=j]
17
18 Demander(i) ==
19   /\ etat[i] = "T"
20   /\ etat' = [etat EXCEPT ![i] = "H"]
21   /\ UNCHANGED token
22
23 Sortir(i) ==
24   /\ etat[i] = "E"
25   /\ etat' = [etat EXCEPT ![i] = "T"]
26   /\ UNCHANGED token
27
28 Entrer(i) ==
29   /\ etat[i] = "H"
30   /\ etat' = [etat = etat EXCEPT ![i] = "E"]
31   /\ token[i]
32   /\ UNCHANGED token (* important *)
33
34 Bouger(i) ==
35   /\ token[i]
36   /\ token' = [token EXCEPT ![i]=FALSE, ![(i+1)%N] = TRUE]
37   /\ UNCHANGED etat
38   /\ etat[i] != "E"
39
40 Next ==
41   \E i \in 0..N-1:
42     \/\ Demander(i)
43     \/\ Entrer(i)
44     \/\ Sortir(i)
45     \/\ Bouger(i)
46
47 Spec ==
48   /\ Initial
49   /\ [Next]_etat, token
50   /\ forall i \in 0..N-1:
51     /\ WF_{etat, token}(Sortir(i))
52     /\ WF_{etat, token}(Entrer(i))
53
54 ExclusionMutuelle == [(Cardinality{(i \in 0..N-1: etat[i] = "E"})} <= 1)]
55 AbsenceDeFamine == \forallall i \in 0..N-1: etat[i] = "H" -> etat[i] = "E"
56 Lien == \forallall i, j \in 0..N-1: [etat[i] = "E" => etat[j] = "E"
57 TokenUnique == \forallall i, j \in 0..N-1: token[i] /\ token[j] => i = j
58
59 ]
```

Liste 13. – Token Ring 1

Pour prouver le raffinement entre TokenRing0 et TokenRing1, il faut au moins exhiber une fonction de mapping entre les variables de TokenRing0 et sur celles de TokenRing1.

$$\text{etat1} = \text{etat0}$$
$$\text{token1} = \text{CHOOSE } i \in 0..N-1 : \text{token}[i]$$

5.4. Raffinage : Modéliser les canaux de communication

On l'a pas fait.