

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

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  Spec == Init \land [Next]_{res,factors}
18 =====
```

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} \subseque 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

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

- on étudie à chaque transition séparément

- bégaiement

$$(\text{nl} = 0 \vee \text{nr} = 0) \wedge \text{nl}' = \text{nl} \wedge \text{nr}' = \text{nr} \Rightarrow \text{nl}' = 0 \vee \text{nr}' = 0$$

- 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
6  TypeInvariant ==
7    /\ nl \in Nat
8    /\ nr \in 0..1
9    /\ ndemr \in Nat
10
11  Initial ==
12    /\ nl = 0
13    /\ nr = 0
14    /\ ndemr = 0
15
16  EntrerL ==
17    /\ nr = 0
18    /\ nl' = nl+1
19    /\ UNCHANGED <<nr>>
20    /\ UNCHANGED <<ndemr>>
21
22  SortirL ==
23    /\ nl > 0
24    /\ nl' = nl - 1
25    /\ UNCHANGED <<nr>>
26    /\ UNCHANGED <<ndemr>>
27
28  EntrerR ==
29    /\ nl = 0
30    /\ nr = 0
31    /\ UNCHANGED <<nl>>
32    /\ nr' = 1
33    /\ ndemr > 0
34    /\ ndemr' = ndemr - 1
35
36  SortirR ==
37    /\ nr = 1
38    /\ UNCHANGED <<nl>>
39    /\ nr' = 0
40    /\ UNCHANGED <<ndemr>>
41
42  DemanderR ==
43    /\ ndemr' = ndemr + 1
44    /\ UNCHANGED <<nr, nl>>
45
46  Next ==
47    \/ EntrerL
48    \/ SortirL
49    \/ EntrerR
50    \/ SortirR
51    \/ DemanderR
52
53  Spec ==
54    /\ Initial
55    /\ [Next]_{nl, nr}
56    /\ WF_{nl, nr}(SortirL)
57    /\ WF_{nl, nr}(SortirR)
58    /\ WF_{nl, nr}(EntrerR)
59
60  ExclusionLR ==
61    [] (nl = 0 /\ nr = 0)
62
63  (*EclusionR ==
64    [] (nr \in 0..1)
65    (* déjà dans invariant de type*)
66  *)
```

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 ») \Rightarrow exclusion est préservée adns LR1

4. L'algorithme de Peterson

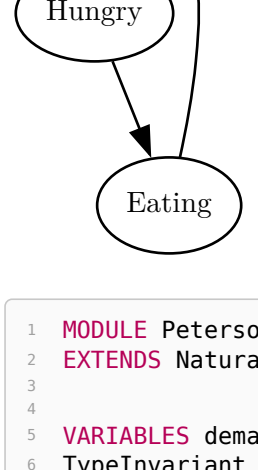
Il s'agit d'exclusion mutuellement entre deux processus.

```
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
5  VARIABLES demande, tour, etat
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    /\ etat' = [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] \vee tour = i)
32
33  Next == \E i \in 0..1 : Entrer(i) \vee Demander(i) \vee 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    [] (\forall i \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    \forall i \in 0..1 : etat[i] = "H" -> etat[i] = "E"
50
51  DemandeMaintenue ==
52    \forall i \in 0..1 : [] (etat[i] = "H" => etat[i]' \in {"H", "E"})
53
54  AbsenceDeDeadlock ==
55    {i \in {0..1} | etat[i] = "H"} != \emptyset -> {i \in 0..1 | etat[i] = "E"} != \emptyset
56    (* ou *)
57    \forall i \in 0..1 : etat[i] = "H" => \E j \in 0..1 : etat[j] = "E"
```

Liste 9. – Algorithme de Peterson en TLA+