Cours - Systèmes de Transition

Louis Thevenet

Table des matières

1. Mise en pratique : La factorielle

```
------- MODULE Fact0 -------

EXTENDS Naturals
CONSTANT N
VARIABLE res

Init == res = Fact[N]
Next == UNCHANGED res (*ou FALSE*)
Spec == Init \land [Next]_res
=============
```

Liste 1. – 0 transition

```
------- MODULE Fact1 -------

EXTENDS Naturals
CONSTANT N
ASSUME N \in Nat
VARIABLES res, i

Init ==
    /\ res = 1
    /\ i = 1

Mult ==
    /\ i <= N
    /\ res' = res * i
    /\ i' = i + 1

Next == Mult

Spec == Init \land [Next]_<<res,i>>
=============
```

Liste 2. – Avec transitions

```
------- MODULE Fact1 -------

EXTENDS Naturals
CONSTANT N
ASSUME N \in Nat
VARIABLES res, factors

Init ==
    /\ res = 1
    /\ factors = 1..N

Mult(i) ==
    /\ res' = res * i
    /\ factors' = factors \ {i}

Next == \E i \in factors : Mult (i)

Spec == Init \land [Next]_<<res,factors>>
=============
```

Liste 3. – Sans ordre particulier

```
------- MODULE Fact1 -------

EXTENDS Naturals
CONSTANT N
ASSUME N \in Nat
VARIABLES res, factors

Init ==
    /\ res = 1
    /\ factors = 1..N

Mult(I) ==
    /\ res' = (*on multiplie les éléments de I à res*)
    /\ factors = 1..N

Next == \E I \in SUBSET factors : Mult (i)
Spec == Init \land [Next]_<<res,factors>>
=============
```

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

```
------- MODULE hlmc -------

    VARIABLES h, m, c, l
    RIVES == {"G", "D"}

    Inv(r) ==
        IF r = "G"
          THEN "D"
          ELSE "G"

    TypeInvariant == {h, l, m,c} \subseteq RIVES

    Init ==
        /\ h = "G"
        /\ l = "G"
        /\ m = "G"
        /\ c = "G"
        (*/\ PasMiam*)

    PasMiam ==
        /\ (l = m => h = m)
        /\ (c = m => h = m)

    MoveH ==
        /\ h' = Inv(h)
        /\ UNCHANGED <<l, m, c>>
        /\ PasMiam'

    MoveHL ==
        /\ h' = Inv(h)
        /\ l' = Inv(l)
        /\ h = l
        /\ UNCHANGED << m, c >>
        /\ PasMiam'

    MoveHM ==
        /\ h' = Inv(h)
        /\ m' = Inv(m)
        /\ h = m
        /\ UNCHANGED << l, c >>
        /\ PasMiam

    MoveHC ==
        /\ h' = Inv(h)
        /\ c' = Inv(c)
        /\ h = c
        /\ UNCHANGED << l, m >>
        /\ PasMiam

    Next ==
        \/ MoveH
        \/ MoveHL
        \/ MoveHM
        \/ MoveHC

    Spec ==
        /\ Init
        /\ [Next]_<<h,l,m,c>>

    But == [](~ {h,l,m,c} = {"D"})
=============================================
```

Liste 5. – Sans ordre particulier

3. Problème Lecteurs/Rédacteurs

```
MODULE LR0
EXTENDS Naturals
VARIABLES nl, nr

TypeInvariant ==
    /\ nl \in Nat
    /\ nr \in 0..1

Initial ==
    /\ nl = 0
    /\ nr = 0

EntrerL ==
    /\ nr = 0
    /\ nl' = nl+1
    /\ UNCHANGED <<nr>>

SortirL ==
    /\ nl > 0
    /\ nl' = nl -1
    /\ UNCHANGED <<nr>>

EntrerR ==
    /\ nl = 0
    /\ nr = 0
    /\ UNCHANGED <<nl>>
    /\ nr' = 1

SortirR ==
    /\ nr = 1
    /\ UNCHANGED <<nl>>
    /\ nr' = 0

Next ==
    \/ EntrerL
    \/ SortirL
    \/ EntrerR
    \/ SortirR

Spec ==
    /\ Initial
    /\ [Next]_(nl, nr)
    /\ WF_(nl, nr)(SortirL)
    /\ WF_(nl, nr)(SortirR)

ExclusionLR ==
    [](nl = 0 /\ nr = 0)

(*ExclusionR ==
    [](nr \in 0..1)
    (* déjà dans invariant de type*)
*)
```

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

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

  - 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

```
MODULE LR1
EXTENDS Naturals
VARIABLES nl, nr, ndemr (*nombre demande rédacteurs*)

TypeInvariant ==
    /\ nl \in Nat
    /\ nr \in 0..1
    /\ ndemr \in Nat

Initial ==
    /\ nl = 0
    /\ nr = 0
    /\ ndemr = 0

EntrerL ==
    /\ nr = 0
    /\ nl' = nl+1
    /\ UNCHANGED <<nr>>
    /\ UNCHANGED <<ndemr>>

SortirL ==
    /\ nl > 0
    /\ nl' = nl -1
    /\ UNCHANGED <<nr>>
    /\ UNCHANGED <<ndemr>>

EntrerR ==
    /\ nl = 0
    /\ nr = 0
    /\ UNCHANGED <<nl>>
    /\ nr' = 1
    /\ ndemr > 0
    /\ ndemr' = ndemr - 1

SortirR ==
    /\ nr = 1
    /\ UNCHANGED <<nl>>
    /\ nr' = 0
    /\ UNCHANGED <<ndemr>>

DemanderR ==
    /\ ndemr' = ndemr + 1
    /\ UNCHANGED <<nr, nl>>

Next ==
    \/ EntrerL
    \/ SortirL
    \/ EntrerR
    \/ SortirR
    \/ DemanderR

Spec ==
    /\ Initial
    /\ [Next]_(nl, nr)
    /\ WF_(nl, nr)(SortirL)
    /\ WF_(nl, nr)(SortirR)
    /\ WF_(nl, nr) (EntrerR)

ExclusionLR ==
    [](nl = 0 /\ nr = 0)

(*ExclusionR ==
    [](nr \in 0..1)
    (* déjà dans invariant de type*)
*)
```

Liste 7. – Lecteurs/Rédacteurs 1

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