Systèmes de transitions - Modélisation TLA+

Durée 1h45 - Documents autorisés

7 avril 2015

1 Questions de cours (2 points)

Soit D un ensemble. Soient $S \subseteq D$ et $f \in [D \rightarrow Nat]$ deux variables. Répondre aux questions suivantes en respectant la syntaxe TLA.

1. Donner un prédicat testant si au plus un entier x de S est tel que $f[x] = 0$.
   $Cardinality(\{x \in S : f[x] = 0\}) \leq 1$
2. Donner une expression représentant l'ensemble des entiers x de S tels que $f[x]$ est l'élément maximum de l'ensemble $f(D)$.
   $\{x \in S : (\forall y \in D : f[x] \geq f[y])\}$
   $\{x \in S : f[x] = max(\{f[y] : y \in D\})\}$
3. Donner une action ajoutant à S les entiers x de D tels que $f[x] = 0$.
   $S' = S \cup \{x \in D : f[x] = 0\}$
   $S' = S \cup f^{-1}[0]$
4. Donner une propriété temporelle exprimant que l'ensemble $f(S)$ n'est jamais vide.
   $\Box(\{f[x] : x \in S\} \neq \emptyset)$ *ou* $\Box(S \neq \emptyset)$

2 Exercice (6 points)

Soit le module TLA fourni `Test.tla` définissant le système de transitions `Spec`.

1. Donner le graphe d'exécution correspondant
2. Les propriétés suivantes, exprimées en logique LTL ou CTL, sont-elles vérifiées (donner une justification informelle) ?

(a) $\Diamond \neg s_0$

(b) $\Diamond s_1 \Rightarrow \Diamond s_2$

(c) $\Box(s_1 \Rightarrow \Diamond s_2)$

(d) $\Box(s_0 \vee s_1 \vee s_3)$

(e) $\Diamond\Box\neg s_2 \Rightarrow \Box(s_0 \vee s_3)$

(f) $\exists\Box s_0$

(g) $\exists\Diamond\exists\Box s_1$

(h) $\forall\Diamond s_3 \Rightarrow \forall\Diamond s_1$

(i) $\exists\Diamond\forall\Box(s_1 \vee s_2)$

(j) $\forall\Box\exists\Diamond s_1$

```
---- MODULE Test ----
EXTENDS Naturals
VARIABLES etat
----------------------------------------------------------------
TypeInvariant == etat \in { "s0", "s1", "s2", "s3" }
----------------------------------------------------------------
Trans0  == /\ etat = "s0" /\ etat' \in { "s1", "s3" }

Trans1  == /\ etat = "s1" /\ etat' = "s2"

Trans23 == /\ etat \in { "s2", "s3" } /\ etat' = "s1"
```

```
Spec == /\ etat = "s0" /\ [] [ Trans0 \/ Trans1 \/ Trans23 ]_<<etat>>
        /\ WF_<<etat>>(Trans1)
=================================================================
```

1.
$$\begin{tikzcd} {} \arrow[r] & s_0 \arrow[r] \arrow[dr] & s_1 \arrow[r, bend left] & s2 \arrow[l, bend left] \\ & & s_3 \arrow[u] & \end{tikzcd}$$
+ bégaiement partout

2. *(a) non (bégaiement initial)*
(b) oui WF
(c) oui WF
(d) non $\cdots {s_2}^\omega$
(e) oui : $\Box\Diamond s_2 \vee \Box(s_0 \vee s_3)$
(f) oui (bégaiement initial)
(g) non WF
(h) non : $\forall\Diamond s_3$ *est faux* $(s_0 \to {s_1}^\omega)$
(i) oui $\cdots (s_1 \to s_2)^\omega$
(j) oui (s_1 *toujours accessible)*

3 Problème : détection de la terminaison (12 points [1])

On considère un calcul concurrent constitué d'un ensemble de processus. Tant qu'un processus est actif, il peut réveiller des processus passifs. Par contre, un processus passif (ou en attente), ne peut bien sûr pas réveiller d'autres processus. La question est de détecter quand le calcul est fini, c'est-à-dire que tous les processus sont passifs. Comme il n'est pas possible d'évaluer globalement et instantanément la propriété que tous les processus sont passifs, on regarde un à un chaque processus. Pour cela, on utilise un parcours cyclique (analogue à un jeton circulant) qui passe successivement sur tous les processus. Le jeton ne progresse que quand un processus est passif.

Un tour où le jeton ne voit que des processus passifs ne suffit pas : on peut observer à un instant donné qu'un processus est passif, mais il peut être réveillé plus tard par un autre processus, qui sera lui aussi vu comme passif quand on le regardera. Considérons deux processus p_i et p_j ; lors de la visite de p_i, p_i est passif et p_j actif ; avant que le jeton ne parvienne à p_j, p_j réveille p_i et devient passif ; quand le jeton visite p_j, il le trouve passif ; mais il est faux de conclure que p_i et p_j sont tous deux simultanément passifs.

Pour éviter cela, le jeton colore en blanc les processus rencontrés passifs, et un processus qui devient actif se colore en noir. La détection de la terminaison a lieu quand le jeton fait un tour complet en ne rencontrant que des processus blancs.

Note : l'état actif/passif ne concerne que le calcul dont on veut détecter la terminaison. Un processus passif peut toujours agir vis-à-vis de la détection de la terminaison (par exemple, transmettre le jeton).

Un squelette du module est fourni en 3.6

3.1 Transitions

Définir les prédicats de transitions suivants, en respectant les règles énoncées ci-dessus :

1. Compléter `Init` pour l'initialisation de la variable `passif`, tel qu'elle puisse contenir n'importe quelle configuration de processus actifs/passifs.
   $passif \in [Processus \to BOOLEAN]$
2. Compléter `ReveillerAutreProc(i)`, tel que le processus actif i réveille un autre processus.

1. Toutes les questions valent autant, sauf la question 15 qui vaut double.

$ReveillerAutreProc(i) \triangleq$
$\land \neg passif[i]$
$\land \exists j \in Processus \setminus \{i\} :$
$\quad \land passif' = [passif \ EXCEPT \ ![j] = FALSE]$
$\quad \land couleur' = [couleur \ EXCEPT \ ![j] = Noir]$
$\land UNCHANGED\langle fini, jeton\rangle$

3. Compléter `DetecterTerminaison(i)`, tel que le processus i détecte la terminaison du calcul. Le critère de détection est que le processus est blanc, qu'il a le jeton, que le jeton a visité N processus.

   $DetecterTerminaison(i) \triangleq$
   $\land jeton[i] = N$
   $\land couleur[i] = Blanc$
   $\land fini' = TRUE$
   $\land UNCHANGED\langle couleur, jeton, passif\rangle$

4. `Next` : toutes les transitions possibles du problème modélisé.

   $Next \triangleq$
   $\exists i \in Processus :$
   $\quad \lor ReveillerAutreProc(i)$
   $\quad \lor DetecterTerminaison(i)$
   $\quad \lor EnvoyerJeton(i)$
   $\quad \lor DevenirPassif(i)$

3.2 Équité

5. Pour les actions suivantes, expliquer informellement s'il est nécessaire de mettre de l'équité ou si cela est superflu :
   (a) `ReveillerAutreProc(i)`
   (b) `DevenirPassif(i)`
   (c) `DetecterTerminaison(i)`
   (d) `EnvoyerJeton(i)`

   (a) ReveillerAutreProc : non (action applicative)
   (b) DevenirPassif : oui (sinon le calcul peut ne jamais se terminer) ou non (action applicative)
   (c) DetecterTerminaison : oui (sinon on pourrait ne jamais détecter la terminaison)
   (d) EnvoyerJeton : oui (sinon le jeton ne serait pas obligé de tourner)

6. Donner l'équité minimale nécessaire pour qu'un processus ne reste pas toujours actif.
   $\forall i \in Processus : WF_{vars}(DevenirPassif(i))$

7. Donner l'équité minimale nécessaire pour que le jeton tourne et visite tous les processus.

   (a) $\forall i \in Processus : WF_{vars}(EnvoyerJeton(i))$
   (b) $\forall i \in Processus : WF_{vars}(EnvoyerJeton(i)) \land WF_{vars}(DevenirPassif(i))$
   (c) $\forall i \in Processus : SF_{vars}(EnvoyerJeton(i)) \land WF_{vars}(DevenirPassif(i))$

   Seule la 7c est bonne : dans la 7a, un processus ayant le jeton peut rester définitivement actif ; dans la 7b, deux processus font ping-pong passif/actif sans que le jeton ne bouge jamais.

3.3 Spécification

Définir les propriétés suivantes (qui ne sont pas nécessairement vérifiées par le modèle) :

8. `TerminaisonCorrecte` : si la terminaison est détectée, alors tous les processus sont passifs.
   $\Box(\mathit{fini} \Rightarrow \forall i \in \mathit{Processus} : \mathit{passif}[i])$
9. `TerminaisonDétectée` : Si les processus sont tous passifs, alors la terminaison sera finalement détectée.
   $(\forall i \in \mathit{Processus} : \mathit{passif}[i]) \leadsto \mathit{fini}$
10. `DétectionStable` : si la terminaison est détectée, alors elle reste définitivement vraie.
    $\Box(\mathit{fini} \Rightarrow \Box\mathit{fini})$
11. `CouleurCorrecte` : un processus blanc est nécessairement passif.
    $\forall i \in \mathit{Processus} : \Box((\mathit{couleur}[i] = \mathit{Blanc}) \Rightarrow \mathit{passif}[i])$
12. `PassifDevientBlanc` : tout processus définitivement passif finit par devenir blanc.
    $\forall i \in \mathit{Processus} : (\Diamond\Box\mathit{passif}[i]) \Rightarrow \Diamond(\mathit{couleur}[i] = \mathit{Blanc})$
    N'est vrai que si $\mathit{WF}(\mathit{DevenirPassif}) \wedge \mathit{SF}(\mathit{EnvoyerJeton})$
13. `TransmissionJeton` : le jeton n'est jamais transmis par un processus actif.
    $\forall i \in \mathit{Processus} : \Box((\mathit{jeton}[i]' \neq \mathit{jeton}[i] \wedge \mathit{jeton}[i]' = 0) \Rightarrow \mathit{passif}[i])$
    $\forall i \in \mathit{Processus} : \Box(\neg\mathit{passif}[i] \wedge \mathit{jeton}[i] \neq 0) \Rightarrow \text{UNCHANGED } \mathit{jeton}[i])$

13. `NoirRéinitialise` : un processus noir finit par remettre le jeton à 1.
$\forall i \in \mathit{Processus} : \mathit{couleur}[i] = \mathit{Noir} \leadsto \mathit{jeton}[\mathit{suivant}(i)] = 1$ *(* vicieux *)*
N'est vrai que si $\mathit{WF}(\mathit{DevenirPassif}) \wedge \mathit{SF}(\mathit{EnvoyerJeton})$

13. `ProcDevientPassif` : tout processus actif finit par être passif.
$\forall i \in \mathit{Processus} : \Box\Diamond\mathit{passif}[i]$
$\forall i \in \mathit{Processus} : (\neg\mathit{passif}[i]) \leadsto \mathit{passif}[i]$

13. `CalculTermine` : le calcul se termine.
Selon le sens qu'on donne à la phrase :
$\Diamond(\forall i \in \mathit{Processus} : \mathit{passif}[i])$
$\Diamond\mathit{fini}$ *ou* $\Diamond\Box\mathit{fini}$ *(avec `DétectionStable`)*

3.4 Analyse du problème – preuve

14. En s'appuyant sur le prédicat d'état initial et les quatre actions, démontrer l'invariant `CouleurCorrecte`.
    Soit $\mathit{CouleurCorrecte}(i) \triangleq ((\mathit{couleur}[i] = \mathit{Blanc}) \Rightarrow \mathit{passif}[i])$. *Alors `CouleurCorrecte`* $\triangleq \forall i : \Box\mathit{CouleurCorrecte}(i)$. *On note* $\mathit{next}(i)$ *la conjonction des actions.*
    (a) $\mathit{Init} \Rightarrow \forall i : \mathit{CouleurCorrecte}(i)$ *: car* $\mathit{couleur}[i] = \mathit{Noir}$
    (b) $\forall i, j : i \neq j \wedge \mathit{CouleurCorrecte}(i) \wedge \mathit{next}(j) \Rightarrow \mathit{CouleurCorrecte}(i)'$ *: car unchanged* $\mathit{vars}[i]$
    (c) $\forall i : \mathit{CouleurCorrecte}(i) \wedge \mathit{ReveillerAutreProc}(i) \Rightarrow \mathit{CouleurCorrecte}(i)'$ *: car le processus devient noir*
    (d) $\forall i : \mathit{CouleurCorrecte}(i) \wedge \mathit{DevenirPassif}(i) \Rightarrow \mathit{CouleurCorrecte}(i)'$ *: car* $\mathit{passif}[i]'$ *est vrai*
    (e) $\forall i : \mathit{CouleurCorrecte}(i) \wedge \mathit{DetecterTerminaison}(i) \Rightarrow \mathit{CouleurCorrecte}(i)'$ *: car unchanged* $\mathit{passif}[i]$ *et* $\mathit{couleur}[i]$
    (f) $\forall i : \mathit{CouleurCorrecte}(i) \wedge \mathit{EnvoyerJeton}(i) \Rightarrow \mathit{CouleurCorrecte}(i)'$ *: car unchanged passif et* $\mathit{passif}[i]$ *(donc* $\mathit{passif}[i]'$*) et* $\mathit{couleur}[i]' = \mathit{Blanc}$

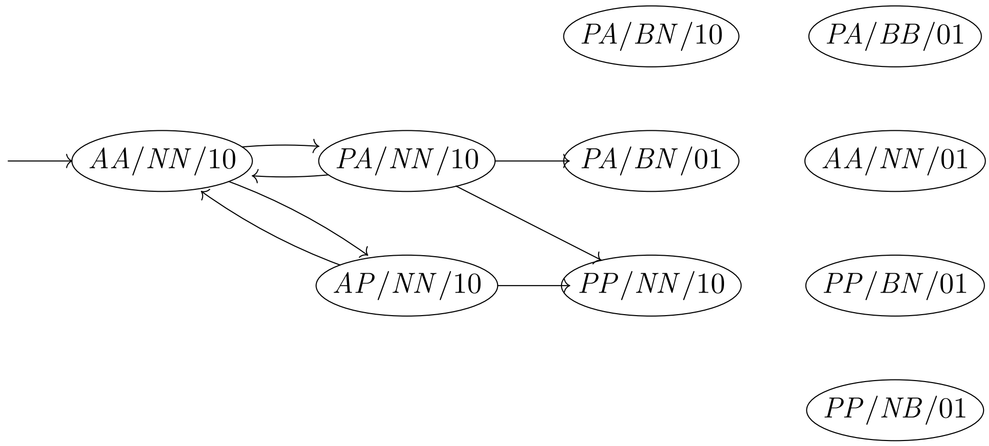
3.5 Analyse du problème – exploration de l'espace d'états

Voici une portion du graphe d'exécutions (avec à la fois des états inaccessibles et des états manquants). Un état $PA/BN/01$ correspond à $\mathit{passif}[0] \wedge \neg\mathit{passif}[1] \wedge (\mathit{couleur}[0] = \mathit{Blanc}) \wedge (\mathit{couleur}[1] = \mathit{Noir}) \wedge (\mathit{jeton}[0] = 0) \wedge (\mathit{jeton}[1] = 1)$, c'est-à-dire :

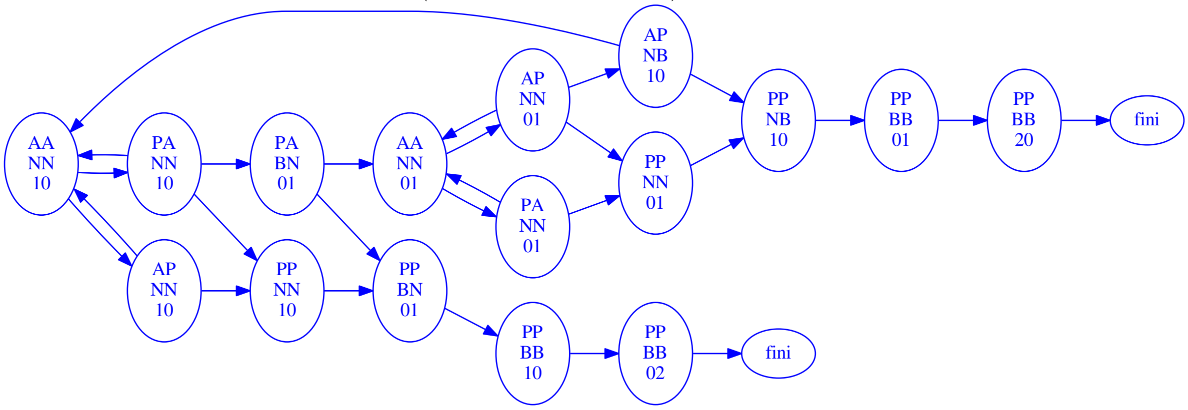
— processus 0 : Passif, Blanc, sans jeton

— processus 1 : Actif, Noir, ayant le jeton, avec compteur = 1

On notera de manière distinctive les états où le booléen *fini* est vrai.



15. Construire le graphe complet (18 états accessibles).



16. Utiliser le graphe, en expliquant comment, pour vérifier la propriété `CouleurCorrecte`.
    C'est un invariant ⇒ à vérifier pour tous les états accessibles du graphe.
    Il n'y a aucun état où un processus est Actif et Blanc ⇒ Ok.
17. Utiliser le graphe, en expliquant comment, pour vérifier la propriété `TerminaisonDétectée`.
    Depuis n'importe quel état PP, on atteint nécessairement un état où fini est vrai (hors bégaiement supprimé par l'équité faible sur EnvoyerJeton et DetecterTerminaison).

3.6 Module fourni : `terminaison_misra.tla`

```
-------------------------------- MODULE terminaison_misra --------------------------------
EXTENDS Naturals, FiniteSets
CONSTANT N        \* le nombre de processus
ASSUME N > 1      \* au moins 2 processus

Processus ≜ 0 .. N − 1
Blanc ≜ "blanc"
Noir ≜ "noir"
Couleur ≜ {Blanc, Noir}

VARIABLES
      couleur,   \* couleur de chaque processus
      passif,    \* actif/passif pour chaque processus
      jeton,     \* contient le nombre de processus blancs visités en séquence, ou 0 si le processus n'a pas le jeton
      fini       \* indique si la terminaison est détectée

vars ≜ ⟨couleur, passif, jeton, fini⟩

TypeInvariant ≜
  □( ∧ couleur ∈ [Processus → Couleur]
     ∧ passif ∈ [Processus → BOOLEAN ]
     ∧ jeton ∈ [Processus → 0 .. N]
     ∧ fini ∈ BOOLEAN
   )

possedeJeton(i) ≜ (jeton[i] ≠ 0)     \* Le processus i a-t-il le jeton ?
suivant(i) ≜ (i + 1)%N               \* Site suivant pour le jeton
------------------------------------------------------------------------------------------
Init ≜
  ∧ couleur = [i ∈ Processus ↦ Noir]
  ∧ jeton = [i ∈ Processus ↦ IF (i = 0) THEN 1 ELSE 0]   \* jeton sur le site 0
  ∧ fini = FALSE
  ∧  \* À COMPLÉTER : initialisation de passif

ReveillerAutreProc(i) ≜   \* le processus actif i réveille un autre processus
   \* À COMPLÉTER

DevenirPassif(i) ≜        \* le processus actif i devient passif
  ∧ ¬passif[i]
  ∧ passif′ = [passif EXCEPT ![i] = TRUE]
  ∧ UNCHANGED ⟨couleur, jeton, fini⟩

DetecterTerminaison(i) ≜  \* le processus i détecte la terminaison
   \* À COMPLÉTER

EnvoyerJeton(i) ≜  \* le processus i transmet le jeton au suivant
  ∧ possedeJeton(i)                          \* il a le jeton
  ∧ ¬(jeton[i] = N ∧ couleur[i] = Blanc)     \* et la terminaison n'est pas encore détectée
  ∧ passif[i]                                \* et il est passif
  ∧ couleur′ = [couleur EXCEPT ![i] = Blanc]
  ∧ jeton′ = [jeton EXCEPT ![i] = 0, ![suivant(i)] = (IF (couleur[i] = Blanc) THEN jeton[i] + 1 ELSE 1)]
  ∧ UNCHANGED ⟨passif, fini⟩

Next ≜  \* À COMPLÉTER

Fairness ≜  \* À COMPLÉTER

Spec ≜
  ∧ Init
  ∧ □[Next]_⟨vars⟩
  ∧ Fairness
==========================================================================================
```