Systèmes de transitions - Modélisation TLA

Durée 1h45 - Documents autorisés

13 avril 2012

Questions de cours (4 points) 1

Soit S un ensemble fini d'entiers naturels. Répondre aux questions suivantes en respectant la syntaxe TLA.

- 1. Donner une expression représentant le sous-ensemble de S restreint aux nombres ≥ 10 .
- 2. Donner un prédicat qui teste si S contient au moins un entier pair.
- 3. Soit x une variable. Écrire une action exécutable seulement si x est dans S, et qui dans ce cas change x pour une autre valeur de S.
- 4. Soit $f \in [S \to Nat]$. Écrire une action qui transforme f en une fonction où les valeurs d'indices pairs sont doublées, les autres restant inchangées.

2 Exercice (4 points)

Soit le module Test.tla ci-dessous définissant le système de transitions Spec. Les propriétés suivantes, exprimées en logique LTL ou CTL, sont-elles vérifiées (donner une justification informelle)?

1. $\exists \Diamond (y > 10)$

5. $\forall \Diamond (x = y)$

 $2. \ \exists \Box (y=0)$

6. $\Diamond (y > 10)$

3. $\forall \Box \exists \Diamond (y=0)$

7. $\Box(y > 0)$

4. $\exists \Box \forall \Diamond (x+y=1)$

8. $\Box \Diamond (x=0)$

Module fourni: Test.tla 2.1

— MODULE examen11_test —

EXTENDS Naturals Variables x, y

 $TypeInvariant \triangleq \land x \in Nat \land y \in Nat$

 $SwitchX \triangleq \land x' = 1 - x \land \texttt{unchanged} \ y$

 $IncrY \stackrel{\Delta}{=} \land x = 1 \land y' = y + 1 \land unchanged x$ $Decr Y \triangleq \land y > 0 \land y' = y - 1 \land UNCHANGED x$

 $\begin{array}{ll} \mathit{Fairness} \; \stackrel{\triangle}{=} \; \mathrm{WF}_{\langle x, \, y \rangle}(\mathit{Switch}X) \wedge \mathrm{WF}_{\langle x, \, y \rangle}(\mathit{Incr}Y) \wedge \mathrm{WF}_{\langle x, \, y \rangle}(\mathit{Decr}Y) \\ \mathit{Spec} \; \stackrel{\triangle}{=} \; \wedge x = 1 \wedge y = 0 \wedge \Box[\mathit{Switch}X \vee \mathit{Incr}Y \vee \mathit{Decr}Y]_{\langle x, \, y \rangle} \wedge \mathit{Fairness} \end{array}$

3 Problème des Lecteurs/Rédacteurs (12 points)

On souhaite modéliser et vérifier le problème des lecteurs/rédacteurs. Soit un ensemble fini de processus qui cherchent à accéder à une ressource commune, soit en tant que lecteur, soit en tant que rédacteur. Un processus rédacteur doit être seul à accéder à la ressource, tandis qu'au contraire plusieurs processus lecteurs peuvent se la partager. Un processus ne peut pas être à la fois lecteur et rédacteur. Un squelette de module TLA Lectred.tla est fourni en 3.6.

3.1 Module complet

- 1. Définir le prédicat de transitions Next qui représente toutes les transitions possibles.
- 2. Définir la propriété Spec qui décrit le système de transitions (sans équité).

3.2 Exclusion entre processus

- 1. Définir une propriété Exclusion exprimant les contraintes d'accès à la ressource des lecteurs/rédacteurs qui doivent être toujours vérifiées.
- 2. Le squelette proposé ne vérifie pas cette propriété. Modifier uniquement les actions afin de respecter cette propriété.
- 3. Justifier la difficulté d'implanter de façon répartie une telle solution.

3.3 Accès à la ressource

- 1. Définir une propriété Vivant qui exprime le fait qu'il y a toujours au moins un processus demandeur qui finira par accéder à la ressource.
- 2. Définir une propriété Acces d'accès garanti à la ressource pour tout processus demandeur.
- 3. Ajouter en justifiant l'équité faible minimale sur les actions, sans laquelle la propriété Vivant est trivialement fausse.

3.4 Analyse des exécutions

Malgré l'équité faible, l'accès à la ressource n'est pas garanti. On s'intéresse au graphe d'exécution du système. Pour diminuer la complexité, on considère N=3 processus dont les deux premiers ne cherchent qu'à lire, et le troisième qu'à écrire. De plus, le processus 0 fait toujours sa demande de lecture avant le processus 1. De même, le processus 0 obtient toujours l'accès avant le processus 1 et libère toujours après. Ces différentes contraintes sont résumées dans la figure 1. On admettra que les contraintes imposées aux processus lecteurs permettent tout de même de représenter le cas général.

- 1. Dessiner le graphe d'exécution du système de transitions en suivant les contraintes proposées (15 états).
- 2. Expliquer comment vérifier l'absence d'interblocage par un examen du graphe.
- 3. Expliquer comment vérifier la propriété Vivant par un examen du graphe.
- 4. Proposer une exécution cyclique de la forme : $s_0 \to \dots s_{n-1} \to (s_n \to \dots \to s_{n+p})^{\omega}$ pour laquelle un des processus demandeurs n'accède jamais à la ressource, même en supposant l'équité forte sur les actions. Justifier votre réponse.

3.5 Gestion des priorités

Pour garantir l'absence de famine pour les rédacteurs, on va imposer la propriété de "priorité aux rédacteurs", qui empêche tout lecteur supplémentaire d'accéder à la ressource si un rédacteur cherche déjà à y accéder.

- 1. Définir une propriété PrioRed qui spécifie cette contrainte.
- 2. Proposer une implantation en TLA sans nouvelle variable d'état, en modifiant les actions et/ou l'équité.

3.6 Module fourni : Lectred.tla

```
— MODULE examen11_Lectred —
EXTENDS Naturals, FiniteSets
CONSTANTS N nombre de processus
Proc \triangleq 0 \dots N-1
Et at \ \triangleq \ \{\text{``P''}, \text{``DL''}, \text{``DE''}, \text{``L''}, \text{``E''}\} \ \text{raccourcis pour ``Pense''}, \text{``DemandeLire''}, \text{``DemandeEcrire''}, \text{``Lit''}, \text{``Ecrit''}
VARIABLES etat les états possibles pour chaque processus
TypeInvariant \stackrel{\Delta}{=} \land etat \in [Proc \rightarrow Etat]
Init \stackrel{\triangle}{=} etat = [p \in \mathit{Proc} \mapsto \text{``P''}]
Demander(i) \triangleq
      \wedge etat[i] = "P"
      \land \exists e \in \{\text{"DL"}, \text{"DE"}\} : etat' = [etat \text{ except } ![i] = e]
EntrerLect(i) \triangleq
      \land etat[i] = "DL"
      \wedge etat' = [etat \ EXCEPT \ ![i] = "L"]
EntrerEcr(i) \triangleq
     \wedge \ etat[i] = "DE"
     \wedge etat' = [etat \ EXCEPT \ ![i] = "E"]
Sortir(i) \triangleq
     \land etat[i] \in \{\text{``L''}, \text{``E'}\}
     \wedge etat' = [etat \ EXCEPT \ ![i] = "P"]
```

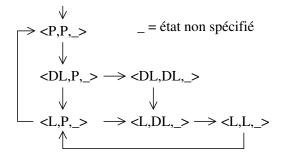


Figure 1 – Graphe d'exécution à compléter