Systèmes de transitions - Modélisation TLA$^+$

Durée 1h45 - Documents autorisés

11 avril 2016

1 Questions de cours (2 points)

Soit trois variables A, B et f. A et B contiennent des ensembles d'entiers, et $f \in [A \rightarrow Nat]$. Répondre aux questions suivantes en respectant la syntaxe TLA$^+$.

1. Donner une action qui change A en un sous-ensemble de B, à condition que A et B soient disjoints.
   $A \cap B = \emptyset \land A' \in \textsc{subset}\ B$ $(ou\ \land A' \subseteq B)$
2. Donner une expression représentant l'ensemble des x de A tels que $f[x]$ est dans B.
   $\{x \in A \mid f[x] \in B\}$ *ou* $f^{-1}(B) \cap A$
3. Donner une propriété temporelle qui dit que la fonction f ne prend que des valeurs paires.
   $\Box(\forall x \in A : f[x]\%2 = 0)$
   $\Box(\{x \in A : f[x]\%2 \neq 0\} = \emptyset)$
4. Donner une propriété temporelle qui dit que l'ensemble A ne décroît pas.
   selon l'interprétation :
   $\Box(A \subseteq A')$
   $\Box(Cardinality(A) \leq Cardinality(A'))$
   $\forall k \in Nat : \Box(Cardinality(A) = k \Rightarrow \Box(Cardinality(A) \geq k))$

2 Exercice (6 points)

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

1. Donner le graphe d'exécution correspondant.

```
-> 0,0 --1--> 1,0
            /  ^
      __2__/   |4
     /         |
-> 0,1 --1--> 1,1 --3--> 2,1
```

   (+ boucle)

2. Les propriétés suivantes, exprimées en logique LTL ou CTL, sont-elles vérifiées (donner une justification informelle) ?

(a) $\Box(y \geq x)$

(b) $\Diamond(x = 2)$

(c) $\Box\Diamond(y \neq x)$

(d) $(x \neq y) \leadsto (x = y)$

(e) $\exists\Box(x + y = 1)$

(f) $\forall\Box\forall\Diamond(y = 1)$

(g) $\forall\Box\exists\Diamond(x = 2)$

(h) $\forall\Box((y = 1) \Rightarrow \forall\Box(y \geq 1))$

(a) non, états (2,1) ou (1,0)

(b) non, sans SF(1,3)

(c) ok, (0,0 et 1,1 non stables)

(d) non, .. $\to (2,1)^\omega$ *ou* $((0,1) \to (1,0))^\omega$

(e) ok (uniquement Act2 depuis 0,1)

(f) ok (0,0 et 1,0 non stables)

(g) ok ($x = 2$ *est toujours accessible)*

(h) non ($y = 1$ *non stable)*

2.1 Module fourni : `Test.tla`

```
---- MODULE Test ----
EXTENDS Naturals
VARIABLES x,y
----------------------------------------------------------------
Init == x = 0 /\ y \in { 0, 1 }
Act1  == x = 0 /\ x' = 1 /\ UNCHANGED y
Act2  == x+y = 1 /\ x' = y /\ y' = x
Act3  == x+y = 2 /\ x' = 2 /\ UNCHANGED y
Act4  == x+y = 2 /\ y' = 0 /\ UNCHANGED x

Spec == /\ Init /\ [] [ Act1 \/ Act2 \/ Act3 \/ Act4 ]_<<x,y>>
        /\ WF_<<x,y>>(Act1) /\ WF_<<x,y>>(Act2) /\ WF_<<x,y>>(Act3)
================================================================
```

3 Problème : partitionnement (12 points)

Une partition d'un ensemble E est un ensemble de sous-ensembles, tels que l'union forme l'ensemble E, et l'intersection deux à deux des sous-ensembles est vide. Par exemple $\{\{2,4,8\},\{1,5\}\}$ est un partition de $\{1,2,4,5,8\}$. Dans la suite, la partition est réduite à *deux* sous-ensembles.

On considère deux sites qui possèdent chacun un ensemble distinct d'entiers, $S0$ et $T0$, qui forment donc une partition d'un ensemble $E = S0 \cup T0$. L'objectif est de construire une autre partition S, T de E, tel que S (respectivement T) a le même nombre d'élément que $S0$ (resp. $T0$), et tous les éléments de S sont inférieurs à tous les éléments de T. On appelle ceci une *partition parfaite*.

Pour cela, les sites vont s'échanger des éléments jusqu'à ne plus pouvoir : S envoie son plus grand élément, puis T répond avec son plus petit, puis S envoie son plus grand, et ainsi de suite. Initialement, on démarre avec un envoi de S. La terminaison est détectée par S quand il reçoit un élément plus grand que son max. Il envoie alors à T une valeur distinguée (autre que celles de $S0$ et $T0$) pour lui indiquer l'arrêt.

Un squelette du module est fourni en annexe. Cette version utilise des séquences pour représenter les canaux de communication de S vers T et de T vers S. Attention, ce module contient un petit bug, probablement invisible et pas gênant pour la compréhension, qu'il s'agira de corriger question 11.

Rappel sur les séquences : une séquence en extension est notée $\langle a, b, c \rangle$. Pour une séquence s, $Len(s)$ est sa longueur (son nombre d'éléments), $Head(s)$ son premier élément (la séquence doit être non vide), $Tail(s)$ est tout sauf le premier élément (même contrainte), $s_1 \circ s_2$ est la concaténation de deux séquences et $Append(s, e) \triangleq s \circ \langle e \rangle$.

3.1 Transitions

1. Donner le prédicat de transition `Next`, spécifiant toutes les transitions possibles du problème modélisé.
   $\exists i, j \in Values \cup \{StopValue\} : Tswap(i,j) \vee Sswap(i,j) \wedge Send(i,j) \wedge Tend(i,j)$
2. Préciser la ou les propriétés d'équité minimale nécessaires pour atteindre une solution.
   WF sur toutes les actions pour éviter bégaiement :
   $Fairness \triangleq \forall i, j \in Values \cup \{StopValue\} : WF(Tswap(i,j)) \wedge WF(Sswap(i,j)) \wedge WF(Send(i,j)) \wedge WF(Tend(i,j))$
3. Donner la spécification complète du problème.
   $Spec = Init \wedge \Box[Next]_{vars} \wedge Fairness$

3.2 Spécification

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

4. `DoNotLoseElements` : on ne perd ni ne crée de valeurs par rapport aux valeurs initialement présentes dans $S0$ et $T0$.
   $DoNotLoseElements \triangleq \Box((S \cup T \cup range(channelS) \cup range(channelT)) \setminus \{StopValue\} = S0 \cup T0)$
5. `ChannelsAreSlots` : la longueur des canaux est toujours inférieure ou égale à 1.
   $ChannelsAreSlots \triangleq \Box(Len(channelS) \leq 1 \wedge Len(channelT) \leq 1)$
6. `ChannelsAreExclusive` : à tout instant au plus un canal contient un message.
   $ChannelsAreExclusive \triangleq \Box(Len(channelS) = 0 \vee Len(channelT) = 0)$
7. `PerfectPartition` : il existe un moment où tous les éléments de S sont plus petits que tous les éléments de T.
   $PerfectPartition \triangleq \Diamond(Cardinality(S) = Cardinality(S0) \wedge Cardinality(T) = Cardinality(T0) \wedge max(S) \leq min(T))$
8. `PerfectionIsStable` : si une partition parfaite est atteinte, elle reste définitivement parfaite.
   $PerfectionIsStable \triangleq \Box(S \neq \emptyset \wedge T \neq \emptyset \wedge max(S) \leq min(T) => \Box(max(S) \leq min(T)))$

3.3 Analyse du problème

9. Donner le graphe d'exécution pour $S0 = \{1, 3, 5\}$ et $T0 = \{2, 4\}$.
   $état = S, chanS, chanT, T$
   $\{1,3\},,5,\{2,4\} \rightarrow \{1,3\},2,,\{4,5\} \rightarrow \{1,2\},,3,\{4,5\} \rightarrow \{1,2\},4,,\{3,5\} \rightarrow \{1,2,4\},,0,\{3,5\} \rightarrow \{1,2,4\},,,\{3,5\}$

10. Utiliser ce graphe pour démontrer qu'au moins l'une des propriétés temporelles est invalidée (préciser laquelle et le moyen de montrer sa fausseté).
    PerfectPartition est fausse
11. Corriger le module fourni.
    Tswap : $\land\ out = min(T \cup \{in\})$
    Sswap : $\land\ out = max(S \cup \{in\})$
    Note : renforcer Tswap avec out < in (et Sswap avec out > in) conduit à un interblocage avec le message en transit qui ne peut pas être reçu : c'est faux.
12. Pour deux ensembles $S0$ et $T0$ quelconques, quel est le nombre maximal de transitions nécessaires pour atteindre la solution ?
    |S0|+|T0| + 1 ou 2 (à calculer)

3.4 Autre modélisation

`ChannelsAreSlots` and `ChannelsAreExclusive` nous indiquent qu'il y a au plus une valeur en transit, soit de S vers T, soit de T vers S. On propose alors de remplacer les variables `chanS` et `chanT` par une variable `slot` contenant *une* valeur et une variable `dest` contenant le sens (vers S ou vers T).

13. Énoncer la nouvelle version de la propriété `TypeInvariant`.
    $TypeInvariant \triangleq S \in SUBSETValues \land T \in SUBSETValues$
    $\land\ slot \in Values \cup \{StopValue\} \land dest \in \{"S", "T"\}$
14. Donner le nouveau code des actions `Tswap`, `Sswap`, `Send` et `Tend`.
    cf setpartitioning2
15. Cette version est un raffinement de la version initiale. Donner la relation de raffinement (le *refinmentMapping*) qui lie les variables du second module aux variables du module initial.
    $chanS = (IF\ \ dest = "S"\ \ THEN\ \langle slot \rangle\ ELSE\ \langle\rangle)$
    $chanT = (IF\ \ dest = "T"\ \ THEN\ \langle slot \rangle\ ELSE\ \langle\rangle)$

On ajoute un attaquant qui peut seulement inverser le sens de la destination :

$$
\begin{aligned}
Attacker &\triangleq\ \land\ dest' = (\text{IF } dest = "S" \text{ THEN } "T" \text{ ELSE } "S")\\
&\ \ \land\ \text{UNCHANGED } \langle S, T, slot \rangle\\
Next &\triangleq\ \ldots \lor Attacker
\end{aligned}
$$

16. La propriété de sûreté `DoNotLoseElements` est-elle invalidée ? (justifier la réponse)
    non : DoNotLoseElements est une propriété sûreté qui ne porte que sur dest, même indirectement (on ne change pas la valeur de slot)
17. La propriété de vivacité `PerfectPartition` est-elle invalidée ? (justifier la réponse)
    oui : attaquant qui inverse systématiquement tout envoi, empêchant qu'il ne soit jamais reçu par le destinataire
18. Que faut-il changer au module TLA$^+$ pour qu'il vérifie ces deux propriétés, si jamais elles sont invalidées ?
    SF sur les réceptions (donc Tswap, Sswap, Send, Tend) : la réception étant infiniment souvent faisable, elle sera finalement faite et le système progresse.

MODULE *examen15_setpartitioning*

EXTENDS *FiniteSets*, *Sequences*, *Naturals*

CONSTANT N, $S0$, $T0$ Max number of values and initial sets

ASSUME $N \in Nat$

$Values \triangleq 1 \mathrel{..} N$ Les valeurs possibles contenues dans les ensembles de départ

$StopValue \triangleq 0$ Une valeur non contenue dans *Values* (sert pour la terminaison)

ASSUME $S0 \subseteq Values \land S0 \neq \{\} \land T0 \subseteq Values \land T0 \neq \{\} \land S0 \cap T0 = \{\}$

VARIABLES S, T, $chanS$, $chanT$

$vars \triangleq \langle S, T, chanS, chanT \rangle$

$min(V) \triangleq$ CHOOSE $m \in V : \forall x \in V : m \leq x$ L'élément minimal d'un ensemble non vide

$max(V) \triangleq$ CHOOSE $m \in V : \forall x \in V : x \leq m$ L'élément maximal d'un ensemble non vide

$range(seq) \triangleq \{seq[v] : v \in \text{DOMAIN } seq\}$ L'ensemble des éléments contenus dans une séquence

---

$Init \triangleq$ LET $v \triangleq max(S0)$ IN initialement, S envoie son *max* à T

$\land S = S0 \setminus \{v\}$

$\land chanT = \langle v \rangle$

$\land T = T0$

$\land chanS = \langle \rangle$

$Tswap(in, out) \triangleq$ T reçoit un élément valide et renvoie son *min*

$\land$ $Len(chanT) > 0$

$\land$ $in = Head(chanT)$

$\land$ $in \neq StopValue$

$\land$ $out = min(T)$

$\land$ $T' = (T \cup \{in\}) \setminus \{out\}$

$\land$ $chanS' = Append(chanS, out)$

$\land$ $chanT' = Tail(chanT)$

$\land$ UNCHANGED $\langle S \rangle$

$Sswap(in, out) \triangleq$ S reçoit un élément et renvoie son *max*, à condition que l'élément reçu soit strictement plus petit que son *max*.

$\land$ $Len(chanS) > 0$

$\land$ $in = Head(chanS)$

$\land$ $out = max(S)$

$\land$ $in < out$

$\land$ $S' = (S \cup \{in\}) \setminus \{out\}$

$\land$ $chanT' = Append(chanT, out)$

$\land$ $chanS' = Tail(chanS)$

$\land$ UNCHANGED $\langle T \rangle$

$Send(in, out) \triangleq$ S reçoit un élément plus grand que son *max* $\Rightarrow$ c'est fini, on envoie la *StopValue* à T

$\land Len(chanS) > 0$

$\land in = Head(chanS)$

$\land out = max(S)$

$\land in \geq out$

$\land S' = S \cup \{in\}$

$\land chanT' = Append(chanT, StopValue)$

$\land chanS' = Tail(chanS)$

$\land$ UNCHANGED $\langle T \rangle$

$Tend(in, out) \triangleq$ T reçoit la *StopValue*

$\land Len(chanT) > 0$

$\land in = Head(chanT)$

$\land in = StopValue$

$\land chanT' = Tail(chanT)$

$\land$ UNCHANGED $\langle S, T, chanS \rangle$