Systèmes de transitions - Modélisation TLA$^+$

Durée 1h45 - Documents autorisés

31 mars 2017

1 Questions de cours (2 points)

Soit quatre variables x, y, S, f. x et y sont des entiers, S est un ensemble d'entiers et $f \in [S \to Nat]$.

Répondre aux questions suivantes en respectant la syntaxe TLA$^+$.

1. Donner une expression qui représente le sous-ensemble des éléments de S qui sont plus grands que x et plus petits que y.
   $\{a \in S : x \leq a \land a \leq y\}$ *ou* $S \cap x..y$
2. Donner une action qui ajoute x à S à condition que y soit déjà dans S.
   $y \in S \land S' = S \cup \{x\} \land$ UNCHANGED $\langle x, y, f \rangle$
3. Donner une propriété temporelle qui exprime que S est toujours un ensemble d'entiers.
   $\Box(S \in$ SUBSET $Nat)$ *ou* $\Box(S \subseteq Nat)$
4. Donner une propriété temporelle qui exprime qu'une valeur de la fonction f sera un jour supérieure à x.
   $\Diamond(\exists i \in S : f[i] > x)$

2 Exercice (5 points)

Soit le module TLA$^+$ ci-dessous définissant le système de transitions `Spec`.

MODULE *examen15b_test*

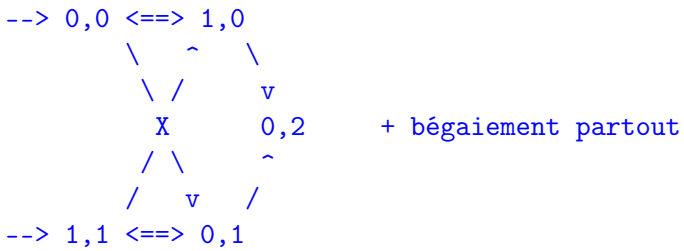
EXTENDS *Naturals*, *TLC*
VARIABLES x, y

$$Init \triangleq x \in \{0, 1\} \land y = x$$

$$action1 \triangleq x = y \land x' = 0 \land y' = 1$$
$$action2 \triangleq x = y \land x' = 1 \land y' = 0$$
$$action3 \triangleq x + y = 1 \land x' = y \land \text{UNCHANGED } y$$
$$action4 \triangleq x \neq y \land x' = 0 \land y' = 2$$

$$Next \triangleq action1 \lor action2 \lor action3 \lor action4$$
$$Fairness \triangleq \text{WF}_{\langle x, y \rangle}(action1) \land \text{SF}_{\langle x, y \rangle}(action4)$$
$$Spec \triangleq Init \land \Box[Next]_{\langle x, y \rangle} \land Fairness$$

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) $\Box(x + y \leq 2)$
(b) $\Box\Diamond(x = 1)$
(c) $\Diamond\Box(y = 2)$
(d) $\exists\Box(x = 0)$
(e) $\forall\Diamond(y = 2)$
(f) $\forall\Box(x \neq y \Rightarrow \forall\Diamond x = y)$

(a) OK d'après le graphe
(b) KO exécution $0,0 \to 0,1 \to 0,2 \to \ldots$
(c) OK équité faible sur 1 $\Rightarrow$ *on va infiniment souvent en* $0,1$ *ou* $1,0$ *; équité forte sur 4* $\Rightarrow$ *on finit en* $0,2$
(d) OK trace $0,0 \to 0,1 \to 0,2 \to \ldots$
(e) OK équité $\Rightarrow$ *on finit en* $0,2$
(f) KO dans l'état $0,2$

3 Problème : algorithme de sélection du maximum (13 points [1])

On souhaite modéliser et vérifier un algorithme qui détermine la valeur maximale possédée par N sites. Initialement, chaque site possède une valeur quelconque, et l'algorithme finit par atteindre un état où tous les sites sont d'accord sur la valeur maximale. Note : les sites n'ont pas nécessairement une valeur distincte, et plusieurs sites peuvent posséder la valeur maximale.

Nous allons étudier plusieurs descriptions de l'algorithme, de plus en plus déterministes.

Un squelette de module TLA$^+$ `maximum1.tla` est fourni en 3.6.

3.1 `maximum1.tla` Module simple non déterministe

1. Compléter le prédicat `Init` définissant les états initiaux possibles, de la façon la plus générale. Les valeurs initiales possédées par les sites devront être quelconques dans $0..V-1$.
Init $\triangleq$ *valinit* $\in [Site \to Valeur]$
2. Donner le prédicat de transition `Next`, spécifiant toutes les transitions possibles du problème modélisé.
Next $\triangleq \exists i, j \in Site : changer(i, j)$
3. Préciser la ou les propriétés d'équité minimale nécessaires pour atteindre une solution.
Fairness $\triangleq \forall i, j \in Site : WF_{vars}(changer(i, j))$
L'équité faible est au moins nécessaire pour éviter le bégaiement. On verra que ça suffit (ceci est bien flou)

1. Toutes les questions valent autant.

3.2 Spécification

Exprimer en TLA$^+$ les propriétés suivantes (qui ne sont pas nécessairement vérifiées par le modèle) :

4. `ValinitEstConstant` : le tableau `valinit` est constant (son contenu ne change pas).
   $ValinitEstConstant \triangleq \Box(valinit' = valinit)$
   $ValinitEstConstant \triangleq \forall p \in Site : \forall k \in Valeur : \Box(valinit[p] = k => \Box(valinit[p] = k))$
   $ValinitEstConstant \triangleq \forall t \in [Site \to Valeur] : \Box(valinit = t => \Box(valinit = t))$
   $ValinitEstConstant \triangleq \exists t \in [Site \to Valeur] : \Box(valinit = t)$
5. `ValeursParmiValeursInitiales` : les valeurs actuelles connues sont un sous ensemble des valeurs initales.
   $\Box(range(valeur) \subseteq range(valinit))$
6. `ValeurEstCroissant` : pour chaque site, sa valeur courant est croissante.
   $ValeurEstCroissant \triangleq \forall p \in Site : \forall k \in Valeur : \Box(valeur[p] = k => \Box(valeur[p] >= k))$
   $ValeurEstCroissant \triangleq \forall p \in Site :: \Box(valeur[p]' >= valeur[p])$
7. `NonNul` : un site qui démarre avec une valeur nulle prendra une valeur non nulle.
   $\forall p \in Site : (valinit[p] = 0) \rightsquigarrow (valeur[p] \neq 0)$
   Non vérifiée si le max est 0 !
8. `ÇaTermine` : on atteint un point où il n'y a plus qu'une seule valeur commune à tous les sites.
   $CaTermine \triangleq \Diamond(Cardinality(\{valeur[i] : i \in Site\}) = 1)$
   $CaTermine \triangleq \Diamond\Box(Cardinality(\{valeur[i] : i \in Site\}) = 1)$
   $CaTermine \triangleq \Diamond(\forall i, j \in Site : valeur[i] = valeur[j])$
   $CaTermine \triangleq \exists k \in Valeur : \Diamond(\forall i \in Site : valeur[i] = k)$
9. `ÇaTermineAvecMax` : on atteint un point où il n'y a plus qu'une seule valeur commune à tous les sites, et cette valeur est la valeur maximale des valeurs initiales.
   $CaTermineAvecMax \triangleq \Diamond(range(valeur) = \{max(range(valinit))\})$
   $CaTermineAvecMax \triangleq \Diamond\Box(range(valeur) = \{max(range(valinit))\})$
   $CaTermineAvecMax \triangleq \Diamond\Box(\forall i \in Site : valeur[i] = max(range(valinit)))$

3.3 `maximum2.tla` : Choix non arbitraire

Au lieu de se comparer avec n'importe quel autre site, un site ne se compare qu'avec son prédécesseur (i.e. le site i se compare avec $prec(i)$).

10. Modifier le modèle TLA$^+$ fourni pour cela.
    $Next \triangleq \exists i \in Site : changer(i, prec(i))$
    ou renforcer $changer$ *avec* $j = prec(i)$
11. Donner la propriété à vérifier pour montrer que `maximum2` est un raffinage de `maximum1`.
    $maximum2!spec => maximum1!spec$

3.4 Vérification

12. Dessiner le graphe de transitions pour `maximum2` en supposant qu'il y a 4 sites qui ont pour valeurs initiales respectives 0/0/3/1 (9 états).

```
0031 ---> 0033 -----------> 3033 ----------> 3333
     \--> 1031 --> 1033 -/                /
                \--> 1131 --> 1133 --> 3133
 + bégaiement partout
```

13. Comment vérifie-t-on la propriété `ValeursParmiValeursInitiales` en examinant le graphe des transitions ?
    en regardant tous les états

14. Comment vérifie-t-on la propriété `ÇaTermine` en examinant le graphe des transitions ?
    absence de boucles autres que le bégaiement, et vu l'équité, ça progresse ⇒ toute exécution atteint un état sans successeur (hors lui-même).

15. La propriété `ÇaTermineAvecMax` est-elle toujours vérifiée par ce modèle, quelque soit les valeurs initiales des sites ?
    oui ! (ceci n'est pas une réponse valide)

3.5 `maximum3.tla` : Choix ordonné

La version 2 est plus déterministe que la version 1 mais n'évite pas des transitions superflues où un site se compare à multiples reprises avec le site précédent. Nous allons imposer un ordre obtenu par un jeton circulant. Seul le site qui a le jeton peut se comparer avec son prédécesseur. On montrera qu'il suffit de faire 2 tours pour garantir que tous les sites possèdent le maximum global. Noter que le jeton ne transporte pas de valeur, seulement l'autorisation de se comparer.

On ajoute une variable `jeton` qui contient le site où est le jeton, et une variable `tour` dans 0..2. Initialement, le jeton est sur le site 0 et tour vaut 0.

16. Modifier l'action `changer(i)` pour que la mise à jour de `valeur` n'ait lieu que si le site a le jeton et que celui-ci a fait moins de 2 tours. Cette action `changer` doit aussi transmettre le jeton au site suivant et incrémenter si nécessaire `tour`.

```
changer(i) ==
  /\ jeton = i
  /\ tour = 0 \/ tour = 1
  /\ valeur[i] < valeur[prec(i)]
  /\ valeur' = [ valeur EXCEPT ![i] = valeur[prec(i)] ]
  /\ jeton' = (i+1)%N
  /\ tour' = IF jeton = N-1 THEN tour+1 ELSE tour
  /\ UNCHANGED valinit
```

17. Ajouter une action `passer(i)` pour transmettre aussi le jeton (et incrémenter tour si nécessaire) quand la comparaison des valeurs échoue (= quand le site a déjà une valeur plus grande que son prédécesseur).

```
passer(i) ==
   /\ jeton = i
   /\ tour = 0 \/ tour = 1
   /\ valeur[i] >= valeur[prec(i)]
   /\ jeton' = (i+1)%N
   /\ tour' = IF jeton = N-1 THEN tour+1 ELSE tour
   /\ UNCHANGED <<valeur,valinit>>
```

18. Donner une propriété exprimant que si tour vaut 2, alors tous les sites ont la même valeur, laquelle est le maximum des valeurs initiales.
    $\Box((tour = 2) => (range(valeur) = \{max(range(valinit))\}))$
19. Donner une propriété spécifiant que tour est croissant.
    Ambigu : $\Box(tour' >= tour)$ *ou* $(\forall k : tour = k \rightsquigarrow tour > k)$ *(qui est faux)*
20. Montrer qu'il suffit effectivement de 2 tours pour parvenir à la solution.
    Une fois le jeton au site suivant le site possédant le max, N-1 transitions pour propager le max à tout le monde = 1 tour. Dans le pire des cas, le jeton démarre sur le suivant du suivant du max => N-1 transitions inutiles pour amener le jeton au site suivant le max (1 tour). Cqfd.

3.6 Module fourni : `maximum1.tla`

MODULE *maximum*1

Calcule le maximum des valeurs situés sur les sites, en comparant sa *valeur* avec la *valeur* d'un autre site (quelconque)

EXTENDS *Naturals*, *FiniteSets*

CONSTANTS N, nbre de sites
V *valeur* maximale possédée

$Site \triangleq 0 \,..\, N-1$ les sites
$Valeur \triangleq 0 \,..\, V-1$ les valeurs possédées

VARIABLES
valinit, *valeur* initialement possédée par chaque site (constante)
valeur *valeur* maximale actuellement connue par le site

$vars \triangleq \langle valinit, valeur \rangle$

$max(S) \triangleq$ CHOOSE $x \in S : \forall\, y \in S : y \leq x$ maximum d'un ensemble
$range(f) \triangleq \{f[x] : x \in$ DOMAIN $f\}$ codomaine d'une fonction
$prec(i) \triangleq (i + N - 1)\%N$ site précédent

$TypeInvariant \triangleq \Box(\, \wedge valinit \in [Site \to Valeur]$
$\wedge valeur \in [Site \to Valeur])$

$Init \triangleq \wedge \ldots$ *INITIALISATION valinit* À *COMPLÉTER*
$\wedge valeur = valinit$

$changer(i, j) \triangleq$ le site i récupère la *valeur* du site j si elle est plus grande
$\wedge\ valeur[i] < valeur[j]$
$\wedge\ valeur' = [valeur$ EXCEPT $![i] = valeur[j]]$
$\wedge$ UNCHANGED *valinit*

$Next \triangleq \ldots$ À *DÉFINIR*

$Fairness \triangleq \ldots$ À *DEFINIR*

$Spec \triangleq \wedge Init$
$\wedge \Box[Next]_{vars}$
$\wedge Fairness$