Systèmes de transitions - Modélisation TLA$^+$

Durée 1h30 - Documents autorisés

5 avril 2019

1 Questions de cours (2 points)

Soit deux variables x, un entier, et t un tableau inclus dans $[Nat \to Nat]$.

1. Donner un prédicat TLA$^+$ qui exprime que x est plus grand que toute valeur de t.
2. Donner une action TLA$^+$ qui incrémente la x-ième case du tableau t, à condition que x soit dans le domaine de définition de t.

2 Exercice (6 points)

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

MODULE *examen18_test*

EXTENDS *Naturals*
VARIABLES x, y

$$TypeInvariant \triangleq x \in Nat \land y \in \text{BOOLEAN}$$

$$DecX \triangleq x' = x - 2 \land x' \geq 0 \land y' = (y \land (x' \neq 0))$$
$$IncX \triangleq x < 3 \land x' = x + 1 \land \text{UNCHANGED } y$$
$$FreeX \triangleq y \land x = 1 \land x' \in \{0, 1, 2, 3\} \land \text{UNCHANGED } y$$
$$SetY \triangleq x = 3 \land y' = \text{TRUE} \land \text{UNCHANGED } x$$
$$Fairness \triangleq \text{WF}_{\langle x, y\rangle}(IncX) \land \text{SF}_{\langle x, y\rangle}(SetY)$$
$$Init \triangleq x = 0 \land y = \text{FALSE}$$
$$Spec \triangleq Init \land \Box[DecX \lor IncX \lor FreeX \lor SetY]_{\langle x, y\rangle} \land Fairness$$

1. Dessiner le graphe d'exécution du système de transition.
2. Indiquer si les propriétés suivantes, exprimées en logique LTL ou CTL, sont vérifiées. Justifier les réponses (argumentaire ou contre-exemple).

   (a) $\Box\neg(x = 0 \land y)$
   (b) $\Box\Diamond y$
   (c) $\Box\Diamond(x \geq 2 \lor y)$
   (d) $y \rightsquigarrow x = 1$
   (e) $\exists\Diamond(x = 2 \land y)$
   (f) $\exists\Box(x \neq 3)$
   (g) $(x = 2)\ \exists\mathcal{U}\ (x = 1)$
   (h) $\forall\Box(y \Rightarrow \exists\Diamond(x = 1))$

3 Problème : validation à deux phases (12 points [1])

On considère le problème de la validation à deux phases avec un coordinateur. Un ensemble d'agents se coordonne pour prendre une décision. Si au moins l'un d'entre eux souhaite abandonner (*abort*), alors tous doivent décider d'abandonner ; si tous souhaitent valider (*commit*), alors tous doivent valider. Pour réaliser cela, chaque agent envoie sa proposition à un coordinateur. Si le coordinateur reçoit un abort, il diffuse à tous la décision d'abandon ; s'il reçoit autant de proposition de commit qu'il y a d'agents, il diffuse à tous la décision de valider.

Un squelette de module TLA$^+$ `TwoPhaseCommit.tla` est fourni à la fin du sujet.

3.1 Module complet

1. Compléter l'action `CoordDecideCommit` pour que le coordinateur décide `committed` et envoie à tous les agents cette décision, à condition que tous les agents aient proposé `commit`.
2. Définir le prédicat de transition `NextAgent(i)` qui représente les transitions possibles pour un agent i.
3. Définir le prédicat de transition `NextCoord` qui représente les transitions possibles pour le coordinateur.
4. Définir le prédicat de transition `Next` qui représente toutes les transitions possibles du système.
5. Définir la propriété `Spec` qui décrit le système de transitions.

3.2 Spécification

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

6. `FinalAgreement` : deux agents ne peuvent pas décider l'un `committed` et l'autre `aborted`.
7. `CommitOnAgreement` : si le coordinateur décide `committed`, c'est que tous les agents sont prêts à valider ou ont validé.
8. `AbortOnVeto` : le coordinateur décide `aborted` si au moins un agent abandonne.
9. `Irrevocability` : la décision finale d'un agent (`committed` ou `aborted`) est irrévocable, il n'en change plus une fois prise.
10. `DecisionReached` : tout agent finit par décider `committed` ou `aborted`.
11. `MessagesAllReceived` : tous les messages sont finalement consommés.
12. `CanCommit` : il est possible que tous les agents valident. (Note : c'est une propriété CTL).

3.3 Équité

13. Énoncer l'équité minimale nécessaire pour que la propriété `DecisionReached` soit vérifiée.
14. La propriété `MessagesAllReceived` est-elle vérifiée avec cette équité ?

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

3.4 Vérification

15. Dessiner le graphe de transitions pour le cas particulier suivant : $N = 2$, l'agent 1 propose commit *puis* l'agent 2 propose abort (13 états).
16. Comment vérifie-t-on la propriété `FinalAgreement` ?
17. Comment vérifie-t-on la propriété `DecisionReached` ?
18. Comment vérifie-t-on la propriété CTL `CanCommit` ?

3.5 Défaillances

19. Ajouter une action pour perdre arbitrairement un message envoyé au coordinateur.
20. Parmi les propriétés `FinalAgreement`, `Irrevocability`, `DecisionReached`, `CanCommit`, lesquelles sont toujours vérifiées par le modèle ?

3.6 Module fourni : `TwoPhaseCommit.tla`

MODULE *TwoPhaseCommit*

EXTENDS *FiniteSets*, *Bags*, *Naturals*

CONSTANT N number of agents
$Agent \triangleq 1 \,..\, N$

messages used
$propose \triangleq \{\text{“commit”}, \text{“abort”}\}$
$order \triangleq \{\text{“committed”}, \text{“aborted”}\}$

VARIABLES
$amsgs$, messages for agent i
$cmsgs$, messages for the coordinator
$astate$, state of agent i
$cstate$, state of the coordinator
$commits$ number of agents that agree to commit

$vars \triangleq \langle amsgs, cmsgs, astate, cstate, commits \rangle$

$TypeInvariant \triangleq$
$\land\ amsgs \in [Agent \to \text{SUBSET } order]$
$\land\ IsABag(cmsgs)$
$\land\ BagToSet(cmsgs) \in \text{SUBSET } propose$
$\land\ commits \in Nat$
$\land\ cstate \in \{\text{“undecided”}, \text{“aborted”}, \text{“committed”}\}$
$\land\ astate \in [Agent \to \{\text{“undecided”}, \text{“readyAbort”}, \text{“readyCommit”}, \text{“aborted”}, \text{“committed”}\}]$

$Init \triangleq\ \land\ amsgs = [n \in Agent \mapsto \{\}]$
$\land\ cmsgs = EmptyBag$
$\land\ astate = [n \in Agent \mapsto \text{“undecided”}]$
$\land\ cstate = \text{“undecided”}$
$\land\ commits = 0$

$AgentProposeCommit(i) \triangleq$
  $\land\ astate[i] = \text{“undecided”}$
  $\land\ astate' = [astate \text{ EXCEPT } ![i] = \text{“readyCommit”}]$
  $\land\ cmsgs' = cmsgs \oplus SetToBag(\{\text{“commit”}\})$
  $\land\ \text{UNCHANGED } \langle cstate,\ amsgs,\ commits\rangle$

$AgentProposeAbort(i) \triangleq$
  $\land\ astate[i] = \text{“undecided”}$
  $\land\ astate' = [astate \text{ EXCEPT } ![i] = \text{“readyAbort”}]$
  $\land\ cmsgs' = cmsgs \oplus SetToBag(\{\text{“abort”}\})$
  $\land\ \text{UNCHANGED } \langle cstate,\ amsgs,\ commits\rangle$

$AgentReceiveDecision(i) \triangleq$
  $\land\ \exists\, msg \in amsgs[i] :$
    $\land\ astate' = [astate \text{ EXCEPT } ![i] = msg]$
    $\land\ amsgs' = [amsgs \text{ EXCEPT } ![i] = amsgs[i] \setminus \{msg\}]$
    $\land\ \text{UNCHANGED } \langle cstate,\ cmsgs,\ commits\rangle$

$CoordReceiveCommit \triangleq$
  $\land\ cstate = \text{“undecided”}$
  $\land\ \exists\, msg \in BagToSet(cmsgs) :$
    $\land\ msg = \text{“commit”}$
    $\land\ commits' = commits + 1$
    $\land\ cmsgs' = cmsgs \ominus SetToBag(\{msg\})$
    $\land\ \text{UNCHANGED } \langle astate,\ cstate,\ amsgs\rangle$

$CoordReceiveAbort \triangleq$
  $\land\ cstate = \text{“undecided”}$
  $\land\ \exists\, msg \in BagToSet(cmsgs) :$
    $\land\ msg = \text{“abort”}$
    $\land\ cmsgs' = cmsgs \ominus SetToBag(\{msg\})$
    $\land\ cstate' = \text{“aborted”}$
    $\land\ amsgs' = [i \in Agent \mapsto amsgs[i] \cup \{cstate'\}]$
    $\land\ \text{UNCHANGED } \langle astate,\ commits\rangle$

$CoordDecideCommit \triangleq$
  $\land\ cstate = \text{“undecided”}$
  $\land\ \ldots$

$NextAgent(i) \triangleq \text{TRUE}$
$NextCoord \triangleq \text{TRUE}$
$Next \triangleq \ldots$
$Spec \triangleq \ldots$