Systèmes de transitions - Modélisation TLA$^+$

Durée 1h30 - Documents autorisés

15 avril 2022

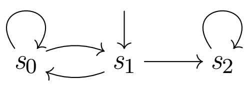
1 Questions de cours (4 points)

Soit trois variables w, x, y. w est un entier, x est une fonction dans $[Nat \rightarrow Nat]$, et y est un ensemble d'entiers.

1. Donner une propriété temporelle qui dit que y est toujours un ensemble d'entiers.
   $\Box(y \in \text{SUBSET } Nat)$ *ou* $\Box(y \subseteq Nat)$ *ou moins élégant* $\Box(\forall i \in y : i \in Nat)$
2. Donner une action qui ajoute w à l'ensemble y.
   $y' = y \cup \{w\}$ $(\land \text{ UNCHANGED } \langle w, x \rangle)$
3. Donner une action qui change la valeur de x en w pour prendre la valeur 3.
   $x' = [x \text{ EXCEPT } ![w] = 3]$ $(\land \text{ UNCHANGED } \langle y, w \rangle)$
4. Donner un prédicat qui dit que toutes les valeurs de x sont supérieures ou égales à 2.
   $\forall i \in \text{DOMAIN } x : x[i] \geq 2$
   ($\forall i \in Nat : x[i] \geq 2$ est approximatif : une fonction dans $[X \rightarrow Y]$ a un domaine inclus dans X mais pas nécessairement tout X, et pareil pour le codomaine)

2 Exercice (4 points)

Soit le système S :



Indiquer si les propriétés suivantes, exprimées en logique LTL ou CTL, sont vérifiées. Justifier les réponses (argumentaire ou contre-exemple).

	sans équité	$WF(s_0, s_1) \land WF(s_1, s_2)$	$WF(s_0, s_1) \land SF(s_1, s_2)$
$\Diamond s_0$			
$\Diamond s_2$			
$s_0 \leadsto s_2$			
$\Box\Diamond\neg s_0$			
$\exists\Box(s_0 \lor s_1)$			
$\exists\Diamond\forall\Box s_2$			

Notation : 1/4 point par bonne réponse ; -1/4 par ligne si absence d'explication ou explication erronée (sans jamais que la ligne compte négativement).

	sans équité	$WF(s_0, s_1) \wedge WF(s_1, s_2)$	$WF(s_0, s_1) \wedge SF(s_1, s_2)$
$\Diamond s_0$	*non* $s_1 \to {s_2}^\omega$	*non (idem)*	*non (idem)*
$\Diamond s_2$	*non* $(s_1 \to s_0)^\omega$	*non (idem)*	*oui*
$s_0 \leadsto s_2$	*non* $s_1 \to {s_0}^\omega$	*non* $(s_1 \to s_0)^\omega$	*oui*
$\Box\Diamond\neg s_0$	*non* $s_1 \to {s_0}^\omega$	*oui*	*oui*
$\exists\Box(s_0 \vee s_1)$	*oui*	*oui*	*non*
$\exists\Diamond\forall\Box s_2$	*oui*	$\diagdown$	$\diagdown$

1. *On a les familles d'exécution* $\langle s_1 \to ({s_0}^+ \to s_1) \to {s_0}^\omega\rangle, \langle(s_1 \to {s_0}^+)^\omega\rangle, \langle s_1 \to ({s_0}^+ \to s_1)^* \to {s_2}^\omega\rangle$.
2. *L'équité* $WF(s_0, s_1)$ *interdit de boucler sur* s_0 *et élimine donc les exécutions* $\cdots \to {s_0}^\omega$ *; la partie* $WF(s_1, s_2)$ *n'élimine aucune exécution car la transition* (s_1, s_2) *n'est jamais continûment faisable.*
3. *L'équité* $WF(s_0, s_1) \wedge SF(s_1, s_2)$ *interdit de boucler sur* s_0 *(partie* $WF(s_0, s_1)$*) et de boucler sur* $(s_1 \to {s_0}^+)$ *(partie* $SF(s_1, s_2)$*) car la transition* (s_1, s_2) *est alors infiniment souvent faisable. Les seules exécutions possibles sont celles* $\cdots \to {s_2}^\omega$.
4. *Pour* $\exists\Box(s_0 \vee s_1)$*, il suffit d'une exécution qui reste dans* $\{s_0, s_1\}$*,* $(s_1 \to s_0)^\omega$ *fait l'affaire. Noter que la négation de la formule est* $\forall\Diamond s_2$ *(CTL) ou* $\Diamond s_2$ *(LTL).*
5. *Pour* $\exists\Diamond\forall\Box s_2$*, il faut un préfixe (*$\exists\Diamond$*) qui conduit à un point où l'on reste définitivement en* s_2 *(*$\forall\Box s_2$*). Comme une fois en* s_2*, on ne peut que boucler, un préfixe* $s_1 \to s_2$ *fait l'affaire.*

3 Problème de l'ensemble indépendant maximal (12 points [1])

Dans un graphe non orienté, un ensemble *stable* ou *indépendant* (en anglais *independent set*) est un sous-ensemble de nœuds du graphe dont aucun n'est adjacent à un autre. Formellement, pour un graphe (V, E) où V sont les nœuds (*vertices*) et E les arêtes (*Edges*), un sous-ensemble S de V est un indépendant si pour tout couple de sommets de S, il n'existe pas d'arête entre eux : $\forall u, v \in S : (u, v) \notin E$. Un ensemble indépendant est *maximal* (*maximal independent set* ou *MIS*) s'il ne peut pas être étendu en ajoutant d'autres nœuds.

Par exemple, pour le graphe $2 \begin{array}{c} \diagup\; 1 \;\diagdown \\ \diagdown\; 3 \;\diagup \end{array} 4 - 5$, nous avons les ensembles indépendants maximaux suivants : $\{1, 3, 5\}, \{2, 4\}, \{2, 5\}$. Les ensembles $\{1\}$ ou $\{1, 3\}$ ou $\{1, 5\}$ sont indépendants mais pas maximaux. Noter qu'un indépendant maximal n'est pas nécessairement optimal (par exemple $\{2, 4\}$).

3.1 Algorithme séquentiel

On considère l'algorithme suivant. Il consiste à prendre un nœud au hasard dans le graphe, à l'ajouter à l'ensemble en construction et à éliminer ce nœud et tous ses voisins. Ceci est répété tant qu'il reste des nœuds.

1. Toutes les questions valent autant.

$$
\begin{array}{l}
\mathit{Nodes} = V \\
\mathit{MIS} = \emptyset \\
\text{repeat} \\
\qquad \text{pick } v \in \mathit{Nodes} \\
\qquad \mathit{MIS} := \mathit{MIS} \cup \{v\} \\
\qquad \mathit{Nodes} := (\mathit{Nodes} \setminus \{v\}) \setminus \mathit{neighbors}(v) \\
\text{until } \mathit{Nodes} = \emptyset
\end{array}
$$

Une modélisation TLA$^+$ de cet algorithme est fourni en fin de sujet (annexe A). On fixe le nombre de nœuds (N). Une arête (élément de E) est représentée par un ensemble à deux éléments, les deux nœuds que l'arête relie. Chaque exécution débute avec un graphe quelconque (appartenant à *AllGraphs* qui est l'ensemble de tous les graphes possibles avec N nœuds). Noter que E (l'ensemble des arêtes) est une constante tout du long d'une exécution.

3.1.1 Spécification

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

1. `MISisIndependent` : à tout instant, l'ensemble *MIS* est indépendant.
   $\mathit{MISisIndependent} \triangleq \Box(\mathit{Independent}(\mathit{MIS}))$
   $\mathit{MISisIndependent} \triangleq \Box(\forall u, v \in \mathit{MIS} : (u, v) \notin E)$
2. `IndependentMaximal` : toute exécution atteint un état où *MIS* est un ensemble indépendant maximal.
   $\mathit{IndependentMaximal} \triangleq \Diamond(\mathit{Independent}(\mathit{MIS}) \wedge \mathit{Maximal}(\mathit{MIS}))$
   (ou $\Diamond\Box$ mais ce n'est pas demandé)
3. `AbsenceDeDiscrimination` : tout nœud peut potentiellement être présent dans l'ensemble *MIS*.
   $\mathit{AbsenceDeDiscrimination} \triangleq \forall v \in V : \exists\Diamond(v \in \mathit{MIS})$
4. `Croissance` : l'ensemble *MIS* ne peut que grandir.
   $\mathit{Croissance} \triangleq \Box(\mathit{MIS} \subseteq \mathit{MIS}')$
   $\mathit{Croissance} \triangleq \forall s \in \text{SUBSET } V : \Box(s \in \mathit{MIS} \Rightarrow \bigcirc(s \in \mathit{MIS}))$
   ($\Box(\mathit{Cardinality}(\mathit{MIS}) \leq \mathit{Cardinality}(\mathit{MIS}'))$ est toléré vu l'ambiguïté de la question mais c'est trop faible, MIS ne garderait pas nécessairement les éléments déjà présents)

3.1.2 Équité

5. Avec la spécification TLA$^+$ fournie, indiquer si les propriétés `MISisIndependent` et `IndependentMaximal` sont vérifiées, en justifiant votre réponse.
   MISisIndependent *: oui : c'est un invariant; il est vrai dans l'état initial et l'action Next en maintient la véracité vu que Nodes ne contient que les nœuds non voisins des nœuds déjà dans MIS.*
   IndependentMaximal *: non, pas d'équité, on peut bégayer dans l'état initial.*
6. Énoncer l'équité minimale nécessaire pour que ces deux propriétés soient vérifiées.
   $\mathit{WF}_{\mathit{vars}}(\mathit{Next})$ *ou* $\forall i \in V : \mathit{WF}_{\mathit{vars}}(\mathit{Action}(v))$

3.2 Algorithme distribué

Dans la version distribué, chaque nœud échange des messages avec ses voisins. Un message est un triplet $\langle type, from, to\rangle$ où *from* et *to* sont les nœuds respectivement émetteur et destinataire, et *type* peut prendre deux valeurs "*join*" ou "*notjoin*". Pour éviter que deux voisins ne tentent simultanément d'intégrer l'ensemble indépendant, on ordonne les nœuds par leur identité. Chaque nœud conserve l'ensemble de ses voisins qu'il considère comme candidats, initialisé avec ses voisins plus prioritaires. Un nœud peut intégrer l'ensemble indépendant uniquement s'il n'a plus de voisin considéré candidat. Quand un nœud intègre l'ensemble indépendant, il envoie "*join*" à tous ses voisins et note qu'il a fini (en étant dans la composante). Quand un site reçoit "*join*", il envoie "*notjoin*" à tous ses voisins et note qu'il a fini (en étant hors de la composante). Quand un site reçoit "*notjoin*", il enlève l'émetteur de l'ensemble de ses voisins candidats. Un squelette incomplet est fourni dans l'annexe B.

3.2.1 Modélisation

7. Compléter le prédicat de transition *Next*.

$$\begin{aligned} Next \triangleq & \lor \exists v \in V : EmptyCandidate(v) \\ & \lor \exists v, w \in V : ReceiveJoin(v, w) \lor ReceiveNotJoin(v, w) \end{aligned}$$

8. Compléter l'action *ReceiveNotJoin*.

$$\begin{aligned} & ReceiveNotJoin(from, to) \triangleq \\ & \quad \land \langle \text{"}notjoin\text{"}, from, to\rangle \in network \\ & \quad \land network' = network \setminus \{\langle \text{"}notjoin\text{"}, from, to\rangle\} \\ & \quad \land A' = [A \text{ EXCEPT } ![to] = A[to] \setminus \{from\}] \\ & \quad \land \text{UNCHANGED } \langle Edge, Done, MIS\rangle \end{aligned}$$

3.2.2 Spécification

Exprimer en LTL ou CTL les propriétés suivantes :

9. `Terminaison` : l'algorithme termine, c'est-à-dire que l'on finit par atteindre un état où tous les sites ont `Done` à vrai et le réseau est vide.

$$Terminaison \triangleq \Diamond(\forall i \in V : Done[i] \land network = \emptyset)$$

10. `PlusieursMessages` : il est possible d'avoir plusieurs messages en transit à destination d'un site v.

$$PlusieursMessages(v) \triangleq \exists\Diamond(Cardinality(\{m \in network : m[2] = v\}) \geq 2)$$
$$PlusieursMessages \triangleq \forall v \in V : \exists\Diamond(Cardinality(\{m \in network : m[2] = v\}) \geq 2)$$
$$PlusieursMessages \triangleq \forall v \in V : \exists\Diamond(\exists s_1, s_2, f_1, f_2 : \langle s_1, f_1, v\rangle \in network \land \langle s_2, f_2, v\rangle \in network \land (f_1 \neq f_2 \lor s_1 \neq s_2))$$

3.3 Vérification

Indifféremment sur `maxindepsetseq` ou `maxindepsetdist`,

11. Expliquer informellement comment vérifier la propriété `MISisIndependent`.
    C'est un invariant : construire l'ensemble des états accessibles et vérifier le prédicat dans chaque état.

12. Expliquer informellement comment vérifier la propriété `AbsenceDeDiscrimination`.
C'est une propriété d'accessibilité : construire l'ensemble des états accessibles et vérifier qu'il existe, pour chaque nœud v, au moins un état qui vérifie le prédicat $v \in MIS$. Ou appliquer l'algorithme de marquage du cours pour une propriété CTL $\exists\Diamond$.

A Version séquentielle : `maxindepsetseq.tla`

MODULE *maxindepsetseq*

EXTENDS *Naturals*, *FiniteSets*
CONSTANT N size of the graphs
$V \triangleq 1 \,..\, N$ node identifiers

The set of all possible graphs with V nodes.
An edge is a set of two nodes (a node cannot have an edge to itself)
A graph is a set of edges, *i.e.* a set of sets of nodes.
$AllGraphs \triangleq \text{SUBSET } \{e \in \text{SUBSET } V : Cardinality(e) = 2\}$

VARIABLES E, $Nodes$, MIS
$vars \triangleq \langle E, Nodes, MIS \rangle$

$neighbors(v) \triangleq (\text{UNION } \{e \in E : v \in e\}) \setminus \{v\}$

$Init \triangleq$
$\land\ E \in AllGraphs$ E (the graph) is a constant in an execution
$\land\ Nodes = V$
$\land\ MIS = \{\}$

$Action(v) \triangleq$
$\land\ MIS' = MIS \cup \{v\}$
$\land\ Nodes' = (Nodes \setminus \{v\}) \setminus neighbors(v)$
$\land$ UNCHANGED E

$Next \triangleq \exists\, v \in Nodes : Action(v)$
$Spec \triangleq Init \land \Box[Next]_{vars}$

$TypeOk \triangleq \land\ E \in \text{SUBSET SUBSET } V$
$\land\ Nodes \in \text{SUBSET } V$
$\land\ MIS \in \text{SUBSET } V$

$Independent(s) \triangleq$ no two neighbors in the set
$\forall\, v \in s : neighbors(v) \cap s = \{\}$

$Maximal(s) \triangleq$ the set cannot be expanded without losing its independence.
$\forall\, v \in Node : v \in s \lor \neg Independent(s \cup \{v\})$

B Version distribuée : `maxindepsetdist.tla`

MODULE *examen21_maxindepsetdist*

EXTENDS *Naturals*, *FiniteSets*
CONSTANT N
$Node \triangleq 1 \,..\, N$
$AnyGraph \triangleq \text{SUBSET } \{e \in \text{SUBSET } Node : Cardinality(e) = 2\}$

VARIABLES $Edge$, $Done$, A, MIS, $network$
$vars \triangleq \langle Edge, Done, A, MIS, network \rangle$

$neighbors(v) \triangleq (\text{UNION } \{e \in Edge : v \in e\}) \setminus \{v\}$

$Init \triangleq$

$\land\ Edge \in AnyGraph$
$\land\ Done = [v \in Node \mapsto \text{FALSE}]$
$\land\ A = [v \in Node \mapsto \{w \in neighbors(v) : w > v\}]$
$\land\ MIS = \{\}$
$\land\ network = \{\}$

$EmptyCandidate(v) \triangleq$

$\land\ \neg Done[v]$
$\land\ A[v] = \{\}$ no candidate neighbors of higher priority
$\land\ MIS' = MIS \cup \{v\}$
$\land\ network' = network \cup \{\langle \text{"join"}, v, w \rangle : w \in neighbors(v)\}$
$\land\ Done' = [Done \text{ EXCEPT } ![v] = \text{TRUE}]$
$\land\ \text{UNCHANGED } \langle Edge, A \rangle$

$ReceiveJoin(from, to) \triangleq$

$\land\ \langle \text{"join"}, from, to \rangle \in network$
$\land\ network' = (network \setminus \{\langle \text{"join"}, from, to \rangle\}) \cup \{\langle \text{"notjoin"}, to, y \rangle : y \in neighbors(to)\}$
$\land\ Done' = [Done \text{ EXCEPT } ![to] = \text{TRUE}]$
$\land\ \text{UNCHANGED } \langle Edge, A, MIS \rangle$

$ReceiveNotJoin(from, to) \triangleq TODO$

$Next \triangleq TODO$

$Spec \triangleq Init \land \Box[Next]_{vars} \land \text{WF}_{vars}(Next)$