Examen Theorie et pratique de la concurrence

Mardi 24 mai 2011, 3h, aucun document autorisé

Questionnaire : Un questionnaire est à remplir sur `http://www.liafa.jussieu.fr/~francoisl/m1tpc.html`. Merci d'y répondre rapidement !

Ce sujet comporte deux exercices et un problème. Le barème est donné à titre indicatif.

Exercice 1 : CCS { 6 points (2/2/2)

1. Construire les systèmes de transitions étiquetés associés aux termes CCS suivants :
   - $P_0 \stackrel{\text{def}}{=} a \cdot b \cdot \emptyset \, + \, a \cdot P_0$
   - $P_1 \stackrel{\text{def}}{=} a \cdot (b \cdot \emptyset \, + \, P_1)$
   - $P_2 \stackrel{\text{def}}{=} a \cdot (b \cdot \emptyset \, + P_2) + a \cdot \emptyset$
   - $P_3 \stackrel{\text{def}}{=} \bar{b} \cdot \bar{a} \cdot P_3$
   - $P_4 \stackrel{\text{def}}{=} (P_2 \, + \, P_3)$
   - $P_5 \stackrel{\text{def}}{=} (P_0 \mid P_3) \backslash \{a\}$
2. Est-ce que les états q_1, r_1 et s_1 des systèmes de transitions de la figure 1 sont bisimilaires ? Justifier votre réponse.

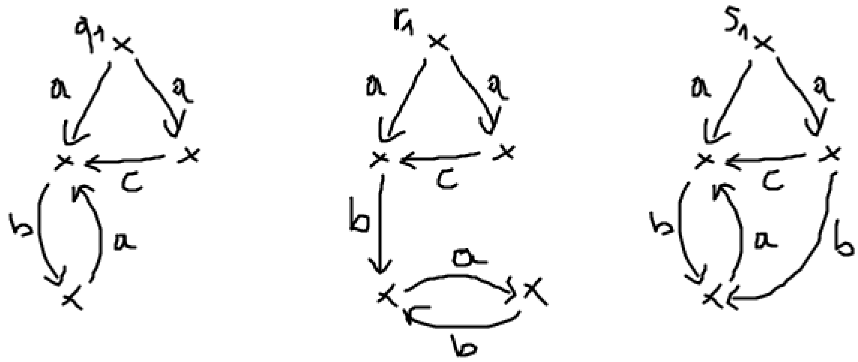

Figure 1 – Exemples de STE (exercice 1)

3. (a) Pour chaque formule de HML ci-dessous, dire si elle est vraie pour les états q_1, r_1 et s_1 de la figure 1 ? (justifier vos réponses)
   - $\langle a\rangle(\langle c\rangle\top)$
   - $\langle a\rangle(\langle b\rangle\top \, \wedge \, \langle c\rangle\top)$
   - $[a]([b]\bot)$
   - $[a](\langle b\rangle\top)$
   - $[b](\langle c\rangle\top)$

   (b) Y-a-t-il une formule de HML qui est vraie en q_1 mais fausse en s_1 ? Et une formule de HML qui est vraie en q_1 mais fausse en r_1 ?

Probleme : Exclusion mutuelle { 8 points

On considère un algorithme d'exclusion mutuelle pour n processus $P_0, \ldots, P_{n-1}$. Il utilise les objets partagés suivants :

- `turn` : une variable partagée à valeur dans $\{0, \ldots, n-1\}$, elle est initialisée avec une de ces valeurs ;
- `lock` : une variable "test-and-set" initialisée à FAUX ;
- `Waiting[0...n-1]` : un tableau de booléens partagé, initialisé avec des FAUX.

De plus, chaque processus utilise deux variables locales : `key` et `aux` (comme en cours, on notera $\texttt{key}^i$ la variable `key` du processus P_i).

Rappel : une variable "test-and-set" est une variable booléenne partagée disposant de deux opérations **atomiques** : (1) "test-and-set" : met la variable à VRAI et renvoie l'ancienne valeur, (si `lock` vaut FAUX, `test-and-set(lock)` met `lock` à VRAI et renvoie FAUX ; et si `lock` vaut VRAI, `test-and-set(lock)` met `lock` à VRAI et renvoie VRAI), et (2) "reset" : met la variable à FAUX.

Le processus P_i (avec $i = 0, \ldots, n-1$) est décrit par le code suivant :

```
Processus Pi:
variables locales:
         int: aux, key
loop forever :
p1: section non critique (SNC)
p2: Waiting[i] := VRAI
p3: key := VRAI
p4: while (Waiting[i] and key) :
p5:           key := test-and-set(lock)
p6: section critique (SC)
p7: Waiting[i] := FAUX
p8: if turn == i
p9:                 then: aux := (turn+1) % n
p10:                else: aux := turn
p11: if Waiting[aux]
p12:                then: turn := aux
p13:                      Waiting[aux] := FAUX
p14:                else:  turn := (aux+1) % n
p15:                       reset(lock)
```

1. Comment s'énoncent l'exclusion mutuelle et l'absence de famine en LTL pour cet algorithme ?
2. Expliquer comment fonctionne cet algorithme.
3. Montrer que l'algorithme vérifie l'exclusion mutuelle.

   Pour cette question, on pourra utiliser (après l'avoir montré !) l'invariant $\mathcal{I}$ défini ci-dessous basé sur les deux ensembles de (numéros de) processus $\mathcal{P}_1$ et $\mathcal{P}_2$ suivants :
   - $\mathcal{P}_1$: contient les numéros i des processus en (1) `p3`, `p4` ou `p5` avec `Waiting[i]` = FAUX, ou (2) `p4` ou `p5` avec $\texttt{key}^i$ = FAUX.
   - $\mathcal{P}_2$: contient les numéros des processus exécutant la section critique ou leur post-protocole (c'est-à-dire en `p6`, `p7`, ... ou `p15`)

L'invariant $\mathcal{I}$ est défini comme la conjonction des 4 propriétés suivantes :

(a) $\mathcal{P}_1$ contient au plus un numéro,

(b) $\mathcal{P}_2$ contient au plus un numéro,

(c) si $\mathcal{P}_1$ est non vide alors $\mathcal{P}_2$ est vide, et

(d) $\texttt{lock} = VRAI \iff \Big((\mathcal{P}_1 \text{ est non vide}) \vee (\mathcal{P}_2 \text{ est non vide})\Big)$.

Rappel : Comme dans le cours, on dit qu'un processus est en `p4` pour signifier que la prochaine instruction qu'il exécutera sera `p4`...

4. Montrer qu'il n'y a pas d'interblocage.
5. Sous hypothèse d'équité entre processus et en supposant que toute section critique termine et conduit à l'exécution du post-protocole, montrer que l'algorithme vérifie l'absence de famine.
   Et, supposons que le processus P_j exécute son pré-protocole :
   (a) Combien de fois un certain processus P_i peut accéder à la section critique avant P_j ?
   (b) Combien de fois des processus peuvent accéder à la section critique avant P_j ?

Exercice 2 : Simulation et bisimulation { 6 points (0,5/1/0,5/1/3)

Dans cet exercice, on utilise les notions suivantes :

- **simulation :** Etant donnés deux systèmes de transitions étiquetés $\mathcal{S}_1 = (S_1, \mathsf{Act}, \rightarrow_1)$ et $\mathcal{S}_2 = (S_2, \mathsf{Act}, \rightarrow_2)$, on dit qu'une relation binaire $\mathcal{R} \subseteq S_1 \times S_2$ est une **simulation** si et seulement si : pour tous sommets s et t tels que $s\mathcal{R}t$, et pour tout $\alpha \in \mathsf{Act}$, si $s \rightarrow_1 s^0$ alors il existe $t \rightarrow_2 t^0$ tel que $s^0\mathcal{R}t^0$.
  On dit que s est "simulé" par t ssi il existe une simulation $\mathcal{R}$ telle que $s\mathcal{R}t$. On le note $s \sqsubseteq t$.
- **traces :** Etant donnés un STE $\mathcal{S} = (S, \mathsf{Act}, \rightarrow)$ et un état $s \in S$, on note $\mathcal{L}(s)$ l'ensemble des mots finis (de Act) étiquetant les exécutions partant de s (ces mots sont appelés les traces de t). Formellement, on a donc :

$$\mathcal{L}(s) \stackrel{\text{def}}{=} \{w \in \mathsf{Act} \ \mid \text{si } w = a_1 \cdots a_n \text{ alors } \exists\, s \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots s_{n\ 1} \xrightarrow{a_n}\}$$

Comme dans le cours, on notera $s \sim t$ lorsque s et t sont (fortement) bisimilaires.

1. Expliquer (informellement) ce que signifie $s \sqsubseteq t$.

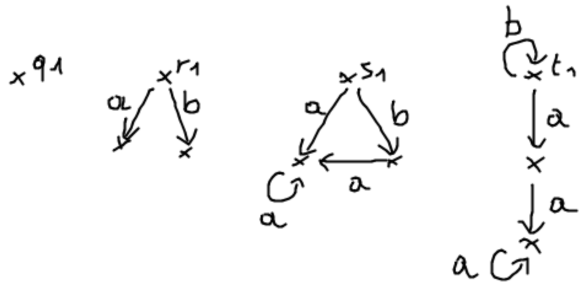

Figure 2 – Exemples de STE (exercice 2)

2. Etant donnés les systèmes de transitions de la figure 2, a-t-on les propriétés suivantes :

$$q_1 \sqsubseteq r_1? \quad r_1 \sqsubseteq s_1? \quad s_1 \sqsubseteq t_1? \quad r_1 \sqsubseteq q_1? \quad s_1 \sqsubseteq r_1? \quad t_1 \sqsubseteq s_1?$$

3. Décrire $\mathcal{L}(s_1)$.

4. Montrer que $t \sqsubseteq s$ implique $\mathcal{L}(t) \subseteq \mathcal{L}(s)$. Montrer que $\mathcal{L}(t) \subseteq \mathcal{L}(s)$ n'implique pas $t \sqsubseteq s$.
5. On dit que deux états s et t se *simulent mutuellement* (et on le note $s \bowtie t$) lorsque $s \sqsubseteq t$ et $t \sqsubseteq s$.
   (a) Montrer que $\bowtie$ est une relation d'équivalence (c'est-à-dire qu'elle est réflexive, symétrique et transitive).
   (b) Montrer que $s \sim t$ implique $s \bowtie t$. Montrer que l'inverse n'est pas vrai : il existe des états qui se simulent mutuellement mais qui ne sont pas bisimilaires.
   (c) Montrer que $s \sim t \Leftrightarrow s \bowtie t$ lorsque s et t sont des etats de systemes de transitions deterministes.
   (Un système de transition $(S, \mathsf{Act}, \rightarrow)$ est déterministe ssi pour tout état $s \in S$ et toute action $\alpha \in \mathsf{Act}$, il existe au plus une transition partant de s et étiquetée par α)