# Examen Théorie et pratique de la concurrence

Mardi 18 mai 2010
Notes de cours et polycopié autorisés

**Questionnaire** : Un questionnaire est à remplir sur `http://www.liafa.jussieu.fr/~francois1/m1tpc.html`. Merci d'y répondre rapidement !

## Problème : Exclusion mutuelle – 8 points

On considère un algorithme d'exclusion mutuelle pour $n$ processus $P_1, \ldots, P_n$ dont le fonctionnement est assuré par un autre processus, appelé serveur, $S$. Le protocole utilise trois variables partagées par tous les processus et le serveur : `req` (entier), `autorisation` (entier) et `fin` (booléen).

1. On considère d'abord la version de l'algorithme décrite à la figure 1 ci-dessous.

```
int req := 0 , autorisation :=0     // variables partagées
boolean fin := true

-- Processus Pi (i=1...n)
loop forever :
p1: section non critique (SNC)
p2: while (autorisation != i) :
p3:    req := i
p4: section critique (SC)
p5: fin := true
p6: req := 0

-- Processus S
loop forever :
s1: await (req != 0)
s2: fin := false
s3: autorisation := req
s4: await (fin)
s5: autorisation := 0
```

FIG. 1 – Algorithme 1

Montrer que cet algorithme ne garantit pas l'exclusion mutuelle dès qu'il y a au moins deux processus (en plus du serveur $S$).

2. On considère à présent la version de l'algorithme décrite à la figure 2 :

```
int req := 0 , autorisation :=0      // variables partagées
boolean fin := true

-- Processus Pi (i=1...n)
loop forever :
p1: section non critique (SNC)
p2: while (autorisation != i) :
p3:    req := i
p4: section critique (SC)
p5: fin := true
p6: req := 0
p7: await(autorisation == 0)

-- Processus S
loop forever :
s1: await (req != 0)
s2: fin := false
s3: autorisation := req
s4: await (fin)
s5: autorisation := 0
```

FIG. 2 – Algorithme 2

(a) Comment s'énonce l'exclusion mutuelle et l'absence de famine en LTL pour cet algorithme ?

(b) Montrer la propriété suivante :

**Prop. 1** $\Box\Big((\texttt{autorisation} == 0) \;\Leftrightarrow\; (\texttt{s1} \vee \texttt{s2} \vee \texttt{s3})\Big)$.

**Rappel :** Comme dans le cours, on utilise les propositions $\texttt{s1}, \texttt{s2} \ldots \texttt{s5}$ pour signifier que la prochaine instruction du serveur $S$ qui sera exécutée est $\texttt{s1}, \ldots$ ou $\texttt{s5}$. Et pour le processus $P_i$, on utilisera $\texttt{p1}^i, \ldots, \texttt{p7}^i$ pour désigner la prochaine instruction de $P_i$ qui sera exécutée.
Ainsi la configuration de départ de l'algorithme vérifie $\texttt{p1}^i \wedge \ldots \texttt{p1}^n \wedge \texttt{s1} \wedge (\texttt{req} == 0) \wedge (\texttt{autorisation} == 0) \wedge (\texttt{fin} == \texttt{true})$.

(c) Montrer que chaque exécution de l'instruction s2 change la valeur de fin (de true à false), c'est-à-dire :

**Prop. 2** $\Box\Big(\texttt{s2} \Rightarrow (\texttt{fin} == \texttt{true})\Big)$

(d) [*] Montrer que lorsque $P_i$ exécute sa section critique, alors fin vaut false, c'est-à-dire :

**Prop. 3** $\Box\Big(\texttt{p4}^i \Rightarrow (\texttt{fin} == \texttt{false})\Big)$

(e) Montrer que cet algorithme garantit l'exclusion mutuelle.

(f) Est-ce que cet algorithme garantit l'absence de famine ?

(g) Ecrire cet algorithme en CCS pour $n = 2$.

## Exercice 1 : CCS – 7 points (3/1/1/2)

1. Construire les systèmes de transitions étiquetées associés aux termes CCS suivants :
   - $P_1 \stackrel{\text{def}}{=} a \cdot a \cdot (b \cdot \emptyset + b \cdot P_1)$
   - $P_2 \stackrel{\text{def}}{=} \bar{a} \cdot P_2$
   - $P_3 \stackrel{\text{def}}{=} (P_1 \mid P_2)$
   - $P_4 \stackrel{\text{def}}{=} P_3 \backslash \{a\}$
2. Est-ce que les états $q_1$, $t_1$ et $s_1$ des systèmes de transitions de la figure 3 sont bisimilaires ? Justifier votre réponse.

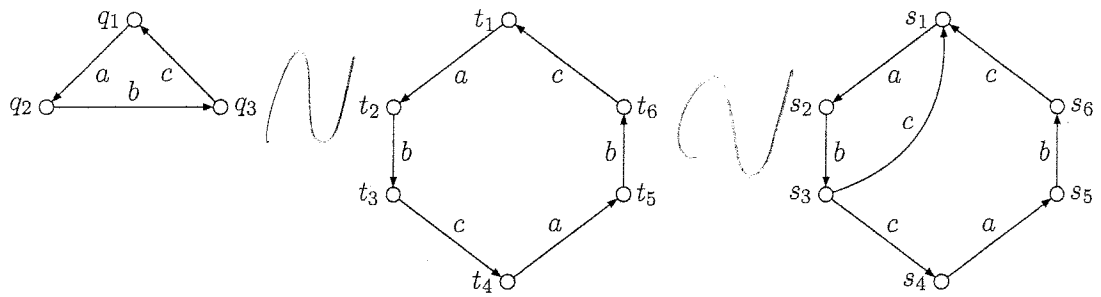


FIG. 3 – Systèmes de transitions de la question 2

3. Est-ce que les états $q_1$, $t_1$ et $s_1$ des systèmes de transitions de la figure 4 sont bisimilaires ? Justifier votre réponse.

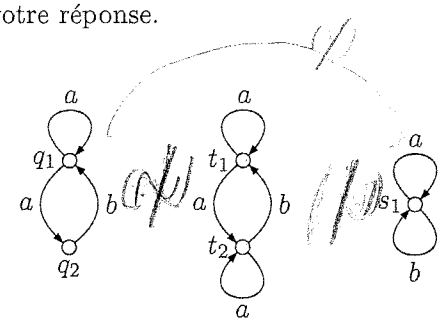


FIG. 4 – Systèmes de transitions des questions 3 et 4 (b)

4. (a) Que signifient les formules HML suivantes :
   - $\langle a \rangle \langle b \rangle \top$
   - $\langle a \rangle ([a] \bot \vee [b] \bot)$
   - $\langle a \rangle (\langle a \rangle \top \wedge \langle b \rangle \top)$
   - $[a] (\langle a \rangle \top \vee \langle b \rangle \top)$
   - $[b] \langle a \rangle \langle c \rangle \top$

   (b) Pour chacune d'elle, expliquer si elle est vraie pour les états $q_1$, $t_1$ et $s_1$ de la figure 4.

   (c) Donner une formule de HML qui est vraie en $s_1$ mais fausse en $t_1$ et en $q_1$.

**Exercice 2 : Simulation et bisimulation – 5 points**

On dit qu'une relation binaire $\mathcal{R}$ sur un ensemble d'états d'un système de transitions étiqueté $(S, \mathsf{Act}, \rightarrow)$ 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 $s \xrightarrow{\alpha} s'$ alors il existe $t \xrightarrow{\alpha} t'$ tel que $s'\mathcal{R}t'$.

On dit que $s$ est "simulé" par $t$ ssi il existe une simulation $\mathcal{R}$ telle que $s\mathcal{R}t$. On le note $s \sqsubseteq t$.

1. Etant donnés les systèmes de transitions de la figure 5, montrer que l'on a les propriétés suivantes : $q_1 \sqsubseteq t_1$ et $s_1 \sqsubseteq r_1$.

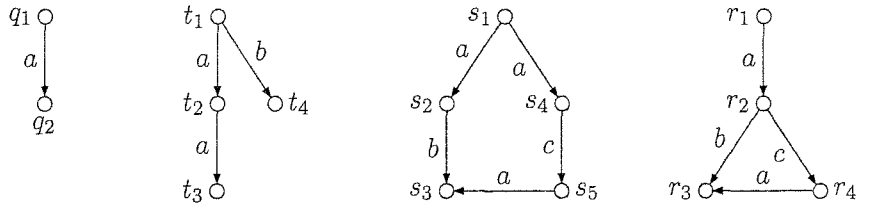


FIG. 5 – Systèmes de transitions de exercice 2

Est-ce que l'inverse ( $t_1 \sqsubseteq q_1$ et $r_1 \sqsubseteq s_1$) est vrai ? Expliquer.

2. Adapter à la simulation le jeu défini en cours pour caractériser la bisimulation.
3. On dit que deux états $s$ et $t$ se *simulent mutuellement* (et on le note $s \bowtie t$) lorsque $s \sqsubseteq t$ et $t \sqsubseteq s$.
   - Montrer que $\bowtie$ est une relation d'équivalence (c'est-à-dire qu'elle est réflexive, symétrique et transitive).
   - Montrer que pour tout état, $s \sim t$ (NB : $\sim$ désigne la bisimulation) implique $s \bowtie t$, mais que l'inverse n'est pas vrai : il existe des états qui se simulent mutuellement mais qui ne sont pas bisimilaires.