Contrôle continu – Modélisation et spécification
Master Informatique

7 novembre 2016

Durée : 1h50.
Documents autorisés : Une feuille A4 manuscrite recto-verso.

Exercice 1 : (5 points) *Modélisation d'algorithmes*

On considère l'algorithme suivant pour trois processus `P1`, `P2` et `P3` avec variables partagées permettant de garantir l'exclusion mutuelle pour `P1` et `P2`.

```
int req = 0; // variables partagées
int resp = 1;
```

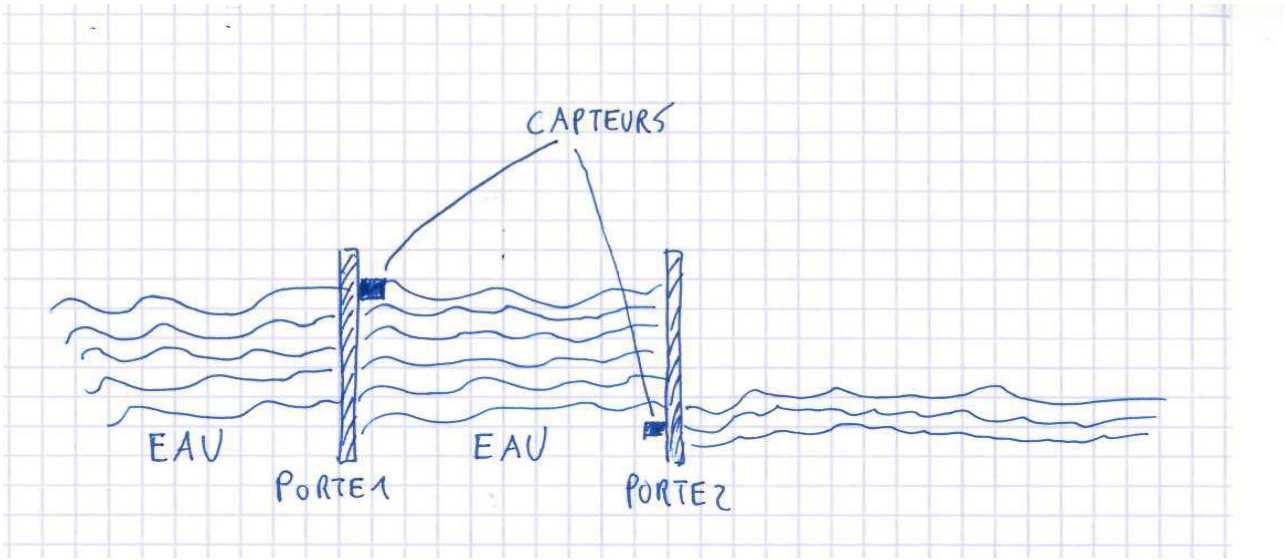
```
-- Processus P1
while(true){
  req = 1;
  while(resp != 1){}
  /*section critique*/
  req = 0;
  resp = 0;
}
```

```
-- Processus P2
int v2=0;
while(true){
  req = 2;
  while(resp!=2){}
  /*section critique*/
  req = 0;
  resp = 0;
}
```

```
-- Processus P3
while(true){
  while(req==0){}
  resp = req;
}
```

1. Donnez une caractérisation des processus P1, P2 et P3 par trois graphes de programmes $PG1$, $PG2$ et $PG3$. Quelles locations de $PG1$ et $PG2$ correspondent aux sections critiques ?
2. On considère maintenant le système de transitions étiqueté $TS(PG1|||PG2|||PG3)$ associé au graphe de programmes $(PG1|||PG2|||PG3)$. Quelle est la forme des états de ce système. Dessinez une partie de ce système en représentant au moins 10 états.
3. Donnez la formule LTL qui permet d'exprimer le fait qu'à aucun moment les deux processus sont en section critique en même temps. Est-ce-que $TS(PG1|||PG2|||PG3)$ satisfait cette formule ? Justifiez votre réponse.
4. Que se passe-t-il si on initialise la variable resp à 0 ? Dans ce cas, donnez la formule LTL qui permet de dire que P1 et P2 finissent toujours pas accéder à la section critique, c'est à dire qu'il n'y a pas de famine.
5. Est-ce-que $TS(PG1||||PG2|||PG3)$ avec la variable resp initialisée à 0 satisfait la formule donnée à la question précédente ? Justifiez votre réponse.

Exercice 2 : (5 points) *Modélisation d'une écluse*



On souhaite modéliser le comportement d'une écluse pour réguler le niveau d'eau sur un canal. L'écluse est composée

de deux portes (P1 et P2) et d'un système pour évacuer l'eau et mesurer le niveau des eaux. Les portes sont soit ouvertes, soit fermées et on peut en commander l'ouverture ou la fermeture. Au début la porte 1 est ouverte et la porte 2 est fermée. En ce qui concerne, l'écluse on peut soit la remplir d'eau ou la vider et on peut tester grâce à deux capteurs, si le niveau d'eau est en bas (quand l'écluse a été vide) ou en haut (quand l'écluse est pleine). L'écluse peut donc être dans quatre états possibles, soit elle est pleine, soit elle est vide, soit elle est en train de se vider, soit elle est en train de se remplir et pour passer de l'état en train de se vider à vide il s'agit d'une action interne de l'écluse, de même pour passer de l'état en train de se remplir à pleine. On a de plus un contrôleur qui en boucle fait les actions suivantes : ferme la porte 1, commande que l'écluse se vide, attend que l'écluse se soit vidée, ouvre la porte 2, ferme la porte 2, commande que l'écluse se remplisse, attend que l'écluse se soit remplie, ouvre la porte 1.

Nous allons modéliser le système global de l'écluse par un STE. Pour cela il vous est demandé de choisir vous même les actions et les propositions atomiques qui font sens.

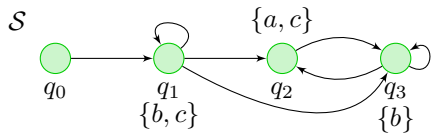
1. Proposez une modélisation par Système de Transitions Étiqueté (STE) des deux portes par deux STE.
2. Proposez une modélisation par STE de l'écluse (sans les portes).
3. Proposez une modélisation par STE du contrôleur.
4. Donnez la table de synchronisation du système global comprenant l'écluse, les deux portes et le contrôleur.
5. Donnez la formule LTL qui permet de vérifier que la porte 1 est ouverte si et seulement si le niveau de l'eau est haut et que la porte 2 est ouverte si et seulement si le niveau de l'eau est bas.

Rappel : Syntaxe et sémantique de LTL. La syntaxe des formules de LTL est définie par la grammaire suivante : $\phi ::= P \mid \phi \vee \psi \mid \phi \wedge \psi \mid \neg\phi \mid \mathbf{X}\phi \mid \phi\,\mathbf{U}\,\psi \mid \mathbf{X}^{-1}\phi \mid \phi\,\mathbf{S}\,\psi$ où P appartient à un ensemble de propositions atomiques AP. On utilise aussi les abréviations $\mathbf{F}$, $\mathbf{G}$, $\mathbf{F}^{-1}$ et $\mathbf{G}^{-1}$. Ces formules s'interprètent à une position $i \geq 0$ le long d'une exécution ρ d'un STE : $\rho(i)$ est le i-ème état et $\ell(\rho(i))$ est l'ensemble des propositions vraies en $\rho(i)$. La sémantique est définie par :

— $\rho, i \models P$ ssi $P \in \ell(\rho(i))$
— $\rho, i \models \mathbf{X}\phi$ ssi $\rho, i+1 \models \phi$
— $\rho, i \models \phi\,\mathbf{U}\,\psi$ ssi $\exists j \geq i$ t.q. $(\rho, j \models \psi) \wedge (\forall k \in [i, j[,$ on a : $\rho, k \models \phi\ )$
— $\rho, i \models \mathbf{X}^{-1}\phi$ ssi $i > 0$ et $\rho, i-1 \models \phi$
— $\rho, i \models \phi\,\mathbf{S}\,\psi$ ssi $\exists j \leq i$ t.q.$(\rho, j \models \psi) \wedge (\forall k \in]j, i],$ on a : $\rho, k \models \phi\ )$

Exercice 3 : (4 points) *Interpréter des formules LTL*

On considère le STE $\mathcal{S}$ de la figure ci-dessous.



1. Décrire l'ensemble des exécutions de $\mathcal{S}$ depuis l'état q_0. Soit $\mathcal{E}$ cet ensemble.
2. Pour chacune des formules suivantes, dire si la formule est vraie pour $\mathcal{S}$ (*ie* pour toutes les exécutions de $\mathcal{E}$). Justifier les réponses.

$$a\,\mathbf{U}\,b \qquad \mathbf{X}(a\,\mathbf{U}\,b) \qquad \mathbf{G}(c \Rightarrow (a\,\mathbf{U}\,b)) \qquad \mathbf{GF}a \Rightarrow \mathbf{GF}b \qquad \mathbf{GF}a \wedge \mathbf{GF}b \qquad \mathbf{G}(b \Rightarrow \mathbf{X}c) \qquad \mathbf{G}(c \Rightarrow \mathbf{X}b)$$

Exercice 4 : (3 points) *Modèles de formules LTL*

Pour chacune des formules suivantes, donner un modèle où la formule est vraie et un où elle est fausse :

$$\mathbf{GF}a \wedge \mathbf{GF}b \wedge \neg\mathbf{G}a \wedge \neg\mathbf{G}b \wedge \mathbf{F}(a \wedge b) \qquad \mathbf{G}(a \Rightarrow \mathbf{X}b) \wedge \mathbf{F}a \qquad \neg(a\,\mathbf{U}\,b)$$

Exercice 5 : (3 points) *Ecrire des formules LTL*

Ecrire en logique temporelle, les propriétés suivantes qui concernent un protocole de communication où des messages sont émis (on utilisera mes propositions `envoi`, `reception`, `perdu`, `ok`) :

1. Si un message est reçu, un message a été envoyé.
2. Si un message est envoyé, un message sera reçu.
3. Si un message est envoyé, un message sera reçu ou perdu.
4. Si un message est envoyé infiniment souvent, un message sera reçu infiniment souvent.
5. Le système finit par vérifier `ok` pour toujours.