Contrôle continu - Session 2 – Modélisation et spécification
Master Informatique

8 décembre 2011

Durée : 1h30. Tous les documents (sauf livre) sont autorisés.

Chaque exercice est sur **5 points.**

Exercice 1 : *Les philosophes*

Cinq philosophes sont assis autour d'une table ronde. Chaque philosophe passe sa vie entre penser et manger. Au centre de la table est mis un plat toujours rempli de spaghetti. Pour manger des spaghetti, chaque philosophe doit utiliser deux fourchettes. Malheureusement, il n'y a que cinq fourchettes, chaque fourchette étant placée entre deux philosophes. Un philosophe ne peut utiliser que les fourchettes qui sont à sa gauche et à sa droite immédiate.

On souhaite donner un modèle pour ce système en utilisant les STE (et leurs opérateurs de composition).

1. Modéliser le comportement d'un philosophe, puis celui d'une fourchette. Préciser l'alphabet des actions utilisé.
2. Montrer comment on peut obtenir les 5 philosophes et les 5 fourchettes en partant des STE de l'exercice précédent.
3. Montrer comment les 5 philosophes et les 5 fourchettes obtenus précédemment peuvent être composés en parallèle afin d'obtenir un modèle de notre problème.
4. Comment peut-on obtenir, en partant du modèle du point précédent, le modèle D_1 du dîner dans lequel que les actions des philosophes "penser" et "manger" sont observables.
5. En partant des modèles d'un philosophe et d'une fourchette écrits au premier point, modéliser le dîner D_2 dans lequel chaque philosophe a ses propres fourchettes. Donner le STE résultat de la composition d'un philosophe avec ses deux fourchettes. Puis, écrire l'expression permettant d'obtenir le modèle du dîner D_2 dans lequel uniquement les actions des philosophes "penser" et "manger" sont observables.
6. Préciser la relation (inclusion ou équivalence de traces, simulation, bisimulation) entre les modèles des deux dîners (D_1 et D_2) considérés. Justifier votre réponse de façon informelle.

Exercice 2 : *Algorithme d'exculsion mutuelle*

```
int wantP = 0, wantQ = 0; // variables partagées
```

```
-- Processus P
while(true){
    section non-critique;
    if (wantQ == -1)
        wantP = -1;
    else
        wantP = 1;
    while(wantP == wantQ);
    section critique;
    wantP = 0;
}
```

```
-- Processus Q
while(true){
    section non-critique;
    if (wantP == -1)
        wantQ = 1;
    else
        wantQ = -1;
    while (wantP == -wantQ);
    section critique;
    wantQ= 0;
}
```

Proposez un réseau de Petri donnant un modèle complet de l'algorithme d'exclusion mutuelle proposé ci-dessus.

Exercice 3 : *Simulation et bisimulation*

1. On considère les deux STE dessinées sur la Figure 1 dont les états initiaux sont 0 :
   (a) Quelle est la relation entre les traces de ces deux STE ?
   (b) Est-ce-que le STE de gauche simule le STE de droite ? Justifiez votre réponse.
   (c) Est-ce-que le STE de droite simule le STE de gauche ? Justifiez votre réponse.
   (d) Est-ce-que ces deux STE sont bisimilaires ? Justifiez votre réponse.
2. On considère les deux STE dessinées sur la Figure 2 dont les états initiaux sont 0 :
   (a) Quelle est la relation entre les traces de ces deux STE ?
   (b) Est-ce-que le STE de gauche simule le STE de droite ? Justifiez votre réponse.

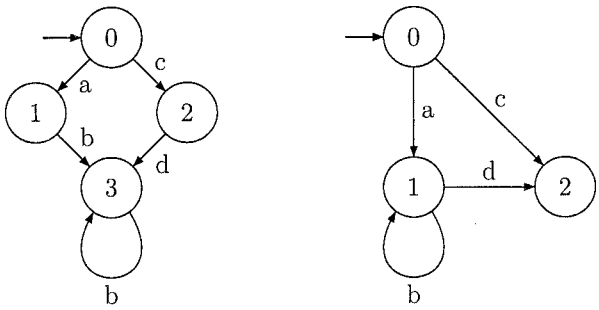

FIGURE 1 – STE

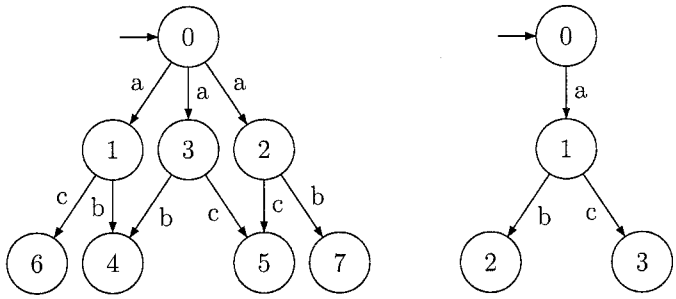

FIGURE 2 – STE

(c) Est-ce-que le STE de droite simule le STE de gauche ? Justifiez votre réponse.

(d) Est-ce-que ces deux STE sont bisimilaires ? Justifiez votre réponse.

Exercice 4 : *Buffers*

n cours, vous avez modélisé et spécifié l'exemple d'un tampon à une place qui reçoit des données par un *port* étiqueté par `put` et qui envoie ces données par un port étiqueté `get`. Dans cet exercice on vous demande de modéliser un tampon à une place qui, pour chaque donnée reçue, il envoie un acquittement à l'émetteur. Plus précisément, en plus des ports `get` et `put`, ce tampon comporte :

- un port `ackput` sur lequel il reçoit l'acquittement et
- un port `ackget` sur lequel il envoie l'acquittement reçu.

Une implémentation possible de ce type de tampon reçoit et envoie la donnée, puis reçoit et envoie l'acquittement.

1. Donner une modélisation de cette implementation.
2. Comparer son comportement avec celui du tampon à une place simple en utilisant les différentes relations étudiées en cours : inclusion de traces, simulation, etc.
3. Refaire le point précédent si les ports `ackput` et `ackget` sont renommés vers l'action interne τ.
4. On compose en parallèle n copies du modèle obtenu tel que les ports `get` et `ackput` communiquent avec les ports `put` respectivement `ackget` du voisin de gauche. Est-ce-que ce système caractérise un tampon à n places ? Justifiez votre réponse.
5. Dans le cas d'une réponse négative à la question précédente, modifiez l'implémentation pour obtenir un modèle pour un tampon à n places en mettant n copies du modèle en parallèle.