Master Ingénierie Informatique

Université Paris Diderot (Paris 7) - UFR Informatique

Modélisation et Spécification : Examen

14 Décembre 2011

Durée : 2 heures
NB : Documents manuscrits autorisés. Toute réponse doit être justifiée.

Exercice 1

Question 1 : Exprimer en LTL les propriétés suivantes :

1. Les deux propriétés P et Q seront vraies (chacune) un nombre infini de fois.
2. Il existe un nombre infini de points où P et Q seront vraies (en même temps).
3. A chaque fois que P est vraie, il est inévitable que Q soit vraie dans le futur en passant par au moins un point où H est vraie.
4. Pour tout instant où Q est vraie, il existe un instant antérieur à cet instant (c'est-à-dire dans le passé) où P était vraie.

Question 2 Exprimer en CTL les propriétés suivantes :

1. A chaque fois que P est vraie, il est possible que Q soit vraie à un moment dans le futur.
2. A chaque fois que P est vraie, il est possible que Q soit continuellement vraie dans le futur.
3. Il est possible que P soit vraie à un moment dans le futur, et qu'à partir de ce moment là, Q devienne vraie infiniment souvent sur tous les chemins d'exécution.

Question 3 Les équivalences ci-dessous (entre formules LTL) sont-elles vraies :

1. $(\Diamond P) \wedge (\Diamond Q) \equiv \Diamond(P \wedge \Diamond Q)$
2. $(\Diamond P) \wedge (\Diamond Q) \equiv \Diamond(P \wedge Q)$
3. $(\Diamond P) \vee (\Diamond Q) \equiv \Diamond(P \vee Q)$
4. $(\Diamond P) \wedge (\Box\Diamond Q) \equiv \Diamond(P \wedge \Box\Diamond Q)$
5. $(\bigcirc P) \wedge (\Diamond P) \equiv \bigcirc\Diamond P$
6. $(\bigcirc P) \wedge (\Diamond P) \equiv \Diamond P$
7. $(\bigcirc P) \wedge (\Diamond P) \equiv \Diamond\bigcirc P$
8. $(\bigcirc P) \wedge (\Diamond P) \equiv \bigcirc P$
9. $(\Box P) \vee (\Box Q) \equiv \Box(P \vee Q)$
10. $(\Box P) \wedge (\Box Q) \equiv \Box(P \wedge Q)$
11. $(\Box\Diamond P) \wedge (\Box\Diamond Q) \equiv \Box\Diamond(P \wedge Q)$
12. $(\Diamond\Box P) \wedge (\Diamond\Box Q) \equiv \Diamond\Box(P \wedge Q)$

Exercice 2

On considère un système de saisie de prix dans une caisse de supermarché. Le système permet de scanner les codes barre d'objets et d'imprimer leurs prix. Le système est composé de trois composantes fonctionnant en parallèle :

1. Un processus de lecture de code. C'est un processus itératif ayant deux actions : Scanner_code qui permet de lire un code barre, puis Emission_code qui émet le code lu.
2. Un processus de conversion de code en prix. C'est un processus itératif ayant deux actions : Recevoir_code lui permettant de recevoir un code, puis Emission_prix qui émet le prix correspondant au code reçu.
3. Un processus d'impression de prix. C'est un processus itératif ayant deux actions : *Recevoir_prix* lui permettant de recevoir un prix, puis Imprimer_prix permettant d'imprimer le prix reçu.

On fait abstraction des valeurs des codes et des prix (c'est-à-dire que l'on ne distinguera pas entre les actions selon les valeurs de leurs paramètres).

Question 1 : Donner des systèmes de transition modélisant chacun des processus décrits ci-dessus.

Question 2 : Donner un modèle pour le système de saisie des prix basé sur une composition parallèle des trois processus ci-dessus. (Préciser les actions de synchronisation.)

Question 3 : Donner le système de transition correspondant à ce modèle.

On considère que seules les actions Scanner_code et Imprimer_prix sont visibles (les autres sont considérées comme internes au système).

Question 4 : Donner un système de transition modéllisant le comportement visible du système de saisie. Combien d'objets peuvent être scannés avant qu'un prix soit imprimé ?

Question 5 : Est-il possible de montrer sur le modèle considéré la propriété suivante : Les prix des objets sont affichés dans le même ordre que celui dans lequel ils ont été scannés.

Exercice 3

On considère deux processus (identiques) qui utilisent durant leurs exécutions une ressource partagée *R*. Cette utilisation doit se faire en exclusion mutuelle. Chaque processus peut être à tout moment dans l'un des trois états : *Préparation*, *Attente*, *Utilisation*. Un processus dans l'état *Préparation* peut soit faire une action qui le remet dans cet état, soit passer dans l'état *Attente* en demandant la ressource au gestionnaire de la ressource (scheduler). Puis, lorsque le scheduler lui accorde l'accès à la ressource, le processus passe dans l'état *Utilisation*. Lorsqu'il a terminé son utilisation de la ressource, il signale au scheduler qu'il libère la ressource et retourne à l'état *Préparation*. On suppose que le temps d'utilisation de la ressource par chaque processus est borné.

On considère dans un premier temps qu'en cas de conflit entre les processus (les deux processus demande l'accès à la ressource), le scheduler accorde la ressource de manière non-déterministe à l'un ou à l'autre.

Question 1 : Modéliser le système par un réseau de Petri : modéliser chacun des processus ainsi que le scheduler, et les mettre en parallèle de manière à assurer l'exclusion mutuelle.

Question 2 : Exprimer en LTL la propriété : Chaque processus qui a demandé l'accès à la ressource obtiendra certainement accès à elle dans le futur. (Utiliser des propositions atomiques correspondant au fait d'être dans chacun des états possibles.) Montrer que cette propriété n'est pas satisfaite par le système de la question 1.

On considère maintenant un scheduler qui attribue la ressource en alternance aux deux processus : d'abord au processus 1 puis au processus 2, ensuite au processus 1, puis au processus 2, et ainsi de suite, dans cet ordre.

Question 3 : Modéliser le système avec ce nouveau scheduler par un réseau de Petri. Montrer que ce système ne satisfait pas la propriété de la question 2. Proposer un nouveau scheduler permettant d'obtenir un système qui satisfait la propriété de la question 2, et donner une modélisation (par réseau de Petri) de ce système.

On suppose maintenant que les processus utilisent deux ressources différentes $R1$ et $R2$ qu'ils doivent acquérir dans un ordre *quelconque*, c'est-à-dire, soit en obtenant d'abord R_1, passant dans un nouvel état d'attente (*Attente'*), puis ensuite R_2, soit en faisant l'inverse en obtenant R_2 puis R_1. Pour cela, chacun des processus demande au scheduler l'une ou l'autre des ressources, et lorsqu'il l'obtient, il demande au scheduler la ressource qui lui manque.

Question 4 : Modéliser à l'aide d'un réseau de Petri le nouveau système. Montrer que ce système peut se bloquer, c'est-à-dire atteindre un état à partir duquel aucune action n'est possible.

Question 5 : Proposer une modélisation du système par un réseau de Petri dans laquelle il n'y a pas de blocage.