Devoir "Introduction à la vérification"

Master 1 Informatique, 2018–2019

Date de remise : Vendredi 3 mai 2019

Exercice 1 On se place sur un alphabet $\Sigma = 2^{\mathrm{AP}}$, où AP est un ensemble fini de propositions atomiques. Une formule φ de LTL sur un alphabet Σ définit l'ensemble suivant de mots infinis :

$$L(\varphi) = \{u \in \Sigma^\omega \mid u, 0 \models \varphi\}$$

Pour chaque formule LTL φ suivante,

- Construire un automate de Bûchai (non généralisé) $\mathcal{A}_\varphi$ reconnaissant $L(\varphi)$,
- Justifier que l'automate est correct, c'est-à-dire, prouver que $L(\mathcal{A}_\varphi) = L(\varphi)$.

Note. Il est déconseillé d'utiliser l'algorithme prouvé en cours en raison du nombre d'états importants qu'il génère. Il vaut mieux faire une construction et un raisonnement au cas par cas.

1. $\alpha = (p \,\mathsf{U}\, q) \,\mathsf{U}\, r$,
2. $\beta = p \,\mathsf{U}\, (q \,\mathsf{U}\, r)$,
3. $\gamma = \mathsf{F}((p \wedge \mathsf{XX}\neg p) \vee (\neg p \wedge \mathsf{XX}p))$,
4. $\delta = \neg\gamma$,

Exercice 2 On considère l'alphabet $\Sigma = 2^{\mathrm{AP}}$ avec $\mathrm{AP} = \{p, q\}$. On note $a = \emptyset$, $b = \{p\}$, $c = \{q\}$ et $d = \{p, q\}$ et on considère l'ensemble suivant de mots infinis :

$$L = \{w \in \Sigma^\omega \mid \text{entre deux } a \text{ de } w, \text{ soit il y a un autre } a, \text{ soit le nombre de } b \text{ est au plus } 1\}.$$

Écrire une formule LTL définissant le langage L.

Exercice 3 (Modélisation d'un protocole d'exclusion mutuelle en Promela) On considère le protocole d'exclusion mutuelle suivant pour `N` processus. Il utilise une variable partagée `tour`, dont l'objectif est de coder l'identité d'un processus voulant entrer en section critique : lorsque sa valeur est l'identité d'un processus `P`, cela signifie que le processus `P` essaie d'entrer en section critique. La valeur -1, qui ne représente l'identité d'aucun processus, signifie qu'aucun processus n'essaie actuellement d'entrer en section critique. Chaque processus exécute un programme modélisé par le code Promela suivant :

```
1  #define NB_PROC 3
2  int tour = -1;
3  active [NB_PROC] proctype P()
4  {
5    do
6      :: tour == -1;
7         tour = _pid;
8         tour == _pid;
9         cs();          /* Section critique */
10        tour = -1
11   od
12 }
```

1. Rappeler la semantique de l'opérateur == utilisé aux lignes 6 et 8.
2. Montrer que l'exclusion mutuelle n'est pas garantie, en donnant une trace d'erreur la plus courte possible qui ne la respecte pas.
3. Remplacer l'appel `cs();` ligne 9 par du code Promela pour détecter l'erreur (exclusion mutuelle non garantie) avec Spin. Quelle est la trace d'erreur fournie par Spin en mode vérification ? Utiliser iSpin pour la visualiser.

On suppose maintenant que pour chaque processus, le délai pour exécuter la ligne 7 après la ligne 6 est inférieur à celui pour exécuter la ligne 8 après la ligne 7. De façon informelle, l'affectation de la ligne 7 est plus rapide que le test fait ligne 8. L'objectif des questions suivantes est :

- de simuler deux minuteries permettant de modéliser cette hypothèse, et
- de vérifier si le protocole devient correct sous cette hypothèse.

4. Ajouter un tableau pour modéliser un chronomètre par processus, utilisé pour assurer que l'exécution de la ligne 7 ne peut être faite qu'après `D1` tics d'horloge après la ligne 6, et que l'exécution de la ligne 8 ne peut arriver que `D2` tics d'horloge après la ligne 7. Choisir de petites valeurs pour `D1`, `D2` avec `D2` $>$ `D1`, par exemple `D2` $= 3$ et `D1` $= 2$.
5. Écrire un processus `Decrementer_Chronos()` pour décrémenter tous les chronomètres qui ne sont pas encore à 0, de façon atomique (en utilisant un bloc `d_step{}`).
6. Remplacer chacune des instructions aux lignes 6, 7, 8, et 9–10 par des blocs atomiques réalisant ces instructions et gérant en plus les chronomètres (c'est-à-dire, les chronomètres doivent être redémarrés, et/ou testés).
7. Vérifier le nouveau modèle en utilisant Spin, pour un nombre de processus allant de 3 à 9 (pour ces valeurs, on doit pouvoir observer l'explosion du nombre d'états. Pour cette raison, vérifier le protocole au delà de 10 processus est difficile).