Concepts et Model Checking – TP 3

Pour ce TP, nous allons travailler avec SPIN, un outil qui sait vérifier la logique de LTL sur des modèles spécifiés dans PROMELA. Pour commencer :

- Télécharger l'archive ZIP qui contient Spin dans la page web du cours et extraire dans un nouveau dossier (disons `spincmc`).
- Dans ce dossier `spincmc`, faire `make` pour compiler. Vous devriez obtenir un exécutable avec le nom de `spin`.
- Il y a deux scripts fournis pour travailler avec Spin, ils s'appellent `spinLTL` et `spinFairLTL`, et un modèle Promela qui s'appelle `dekker.pml`. Usage (exemple) :

  ```
  ./spinLTL dekker.pml '[]!(crit0 && crit1)'
  ```

  Le second paramètre est une formule de LTL, avec `[]` pour **G**, `<>` pour **F** et `&&`, `||`, `!` pour conjonction, disjonction, négation. Si la formule tient dans toutes les exécutions, vous aurez un message du genre `no errors found`. Sinon, vous aurez une exécution dans laquelle la formule ne tient pas. La version `spinFairLTL` ne considère que des exécutions où tout processus progresse infiniment souvent.

Question 1 – Algorithme de Dekker

Le fichier `dekker.pml` contient une variante de l'algorithme de Dekker qui assure l'exclusion mutuelle entre deux processus (voir Wikipédia pour une explication détaillé de l'algorithme). Pourtant, une erreur s'est glissée dans cette réalisation.

1. Utiliser Spin pour tester s'il est possible pour les deux processus d'atteindre leur section critique en même temps. Si c'est le cas, trouver l'erreur en suivant l'exécution fautive et corriger le modèle.
2. Utiliser Spin pour tester si le processus `p0` parvient à entrer dans sa zone critique s'il l'essaye (une fois que `flag0` est vrai, `crit0` doit finalement l'être aussi). Remarquez-vous une différence entre `spinLTL` et `spinFairLTL` ?

Question 2 – Algorithme de Peterson

L'algorithme de Peterson est une alternative à l'algo de Dekker qui a été présenté dans le cours.

1. Créer un nouveau modèle Promela qui réalise l'algorithme de Peterson.
2. Tester si votre modèle satisfait bien les deux propriétés exigées de l'algo de Dekker.