UNIVERSITÉ PARIS 13

Institut Galilée, Université Paris 13 — 2017–2018
Master 2 PLS

SITH

TP 1

Étienne André

Version du sujet : 23 février 2018

Ce TP s'effectuera individuellement. Vous rédigerez un compte-rendu qui sera envoyé à l'enseignant le jour du TP avant 23h59. Ce compte-rendu prendra la forme d'un document mis en forme avec LibreOffice et exporté en PDF, ainsi que le fichier `cafe.imi`.
Nom du fichier : `SITH-TP1-nom-prenom.pdf`
Objet du courriel : `[M2 PLS] SITH TP n°1`
Adresse électronique : `SITH(arobase)lipn13.fr`

IMITATOR et les fichiers de modèles sont téléchargeables sur
`https://www.lipn.fr/~andre/enseignement/SITH/`

Exercice 1 : Prise en main de IMITATOR

IMITATOR est un outil de vérification pour les automates temporisés paramétrés.

Considérer le fichier `fischer.imi` (à ouvrir avec un éditeur de texte), qui est un modèle du protocole d'exclusion mutuelle de Fischer.

Question 1 : En regardant la syntaxe d'IMITATOR, reconstruire graphiquement à la main le premier automate (« `proc1` »).

Exercice 2 : Le protocole d'exclusion mutuelle de Fischer

On s'intéresse dans tout cet exercice au modèle décrit dans le fichier `fischer.imi` (à ouvrir dans un éditeur de texte). Ce fichier modélise le protocole d'exclusion mutuelle de Fischer, un protocole qui garantit que seul un processus à la fois peut accéder à une ressource, appelée section critique.

Question 1 : Quelle est la combinaison des états de contrôle des trois automates où l'exclusion mutuelle est violée ?

Question 2 : Calculer le graphe des états symboliques :

```
$ ./imitator fischer.imi -mode statespace -output-trace-set-verbose
```

Que constate-t-on ? Est-ce étonnant ? Pourquoi ?

(Note : pour arrêter un programme qui boucle, on peut utiliser `contrôle + C`.)

Question 3 : L'optimisation algorithmique consistant à tester l'inclusion d'un état dans un autre, au lieu de l'égalité, s'utilise avec l'option `-incl`. Relancer la génération du graphe des états symboliques. Que constate-t-on quant à la terminaison ?

L'analyse a généré le fichier `fischer-statetspace.jpg` qui montre une représentation graphique de l'espace d'états, avec les contraintes associées.

Question 4 : À partir de l'espace d'états obtenu à la question précédente, appliquer manuellement l'algorithme EF-synthesis. En déduire une contrainte nécessaire et suffisante garantissant l'exclusion mutuelle (c'est-à-dire garantissant que l'exclusion mutuelle n'est jamais violée). Expliquer la méthode dans le compte-rendu.

Question 5 : Décommenter et modifier la ligne suivante en bas de votre fichier :

```
(* property := unreachable loc[proc1] = TODO & loc[proc2] = TODO; *)
```

de façon à ce que les états à éviter soient ceux où l'exclusion mutuelle est violée.

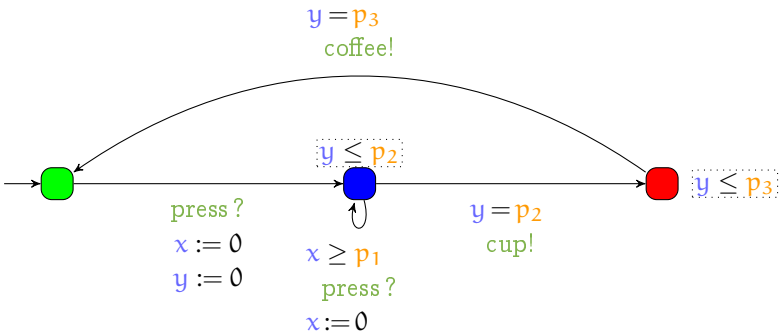
Question 6 : Appliquer EF-synthesis avec IMITATOR pour déterminer les valeurs de paramètres pour lesquelles l'exclusion mutuelle est garantie :

```
$ ./imitator fischer.imi -mode EF -output-trace-set-verbose -incl
```

Ce résultat est-il cohérent avec votre résultat calculé manuellement plus haut ?

Exercice 3 : La machine à café

Question 1 : Rédiger un modèle `cafe.imi` au format IMITATOR qui correspond à l'automate ci-dessous :



Question 2 : On considère l'instance des paramètres suivante :

$$p_1 = 2 \quad p_2 = 5 \quad p_3 = 8$$

Nous allons étudier le comportement de la machine à café avec ces paramètres en utilisant IMITATOR.

Pour cela :

1. instancier les paramètres dans le modèle : dans l'en-tête du modèle, remplacer `p1` par `p1=2`, `p2` par `p2=5`, et `p3` par `p3=8`.

2. générer le graphe des états symboliques.

Pour ces paramètres, combien de sucres peut-on commander au minimum ? Au maximum ?

(à la suite de cette question, annuler les modifications faites au modèle)

Question 3 : Grâce à IMITATOR, nous allons synthétiser d'autres valeurs des paramètres pour lesquelles le comportement du système est le même que pour les valeurs ci-dessus.

Effectuer une synthèse de paramètres par méthode inverse.

```
$ ./imitator cafe.imi cafe.pi0 -output-trace-set-verbose
```

Quelle est la contrainte résultat ? Que garantit-elle ? Comment peut-on le vérifier ?

Question 4 : Ajouter au modèle un autre automate, qui va jouer le rôle d'un *observateur*. Cet observateur doit accéder à un état erreur si et seulement si la propriété suivante est violée : « il est impossible de commander un café avec 3 sucres ou plus »

Autrement dit, l'état erreur est accessible si et seulement s'il est possible de commander un café à 3 sucres ou plus.

Note : pendant la construction de votre modèle, vous pouvez vérifier la syntaxe avec la commande :

```
$ ./imitator votremodele.imi -PTA2JPG
```

Question 5 : Synthétiser les valeurs de paramètres pour lesquelles il est impossible de commander un café avec 3 sucres ou plus.

Exercice 4 (complémentaire) : Fischer, suite

Question 1 : Ouvrir le fichier `fischer.pi0`. Quelle est l'instance des deux paramètres définie dans ce fichier ? Satisfait-elle la contrainte que vous avez générée dans les questions précédentes ?

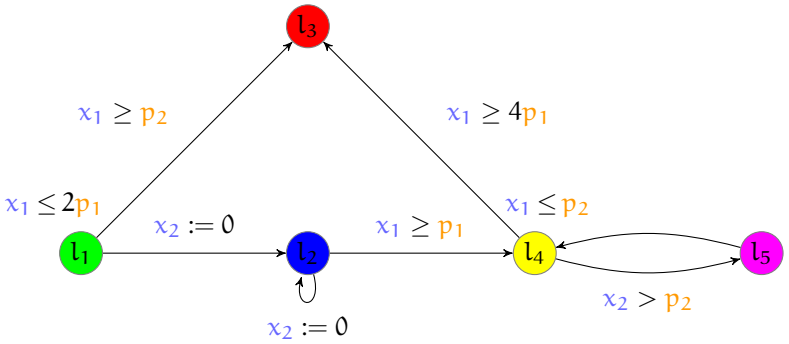
Question 2 : À l'aide du fichier d'instance des deux paramètres `fischer.pi0`, effectuer une synthèse de paramètres par méthode inverse :

```
$ ./imitator fischer.imi fischer.pi0 -incl -output-trace-set-verbose
```

Quelle est la contrainte obtenue ? Garantit-elle bien l'exclusion mutuelle ? Comment le sait-on ?

Exercice 5 (complémentaire) : Synthèse de paramètres pour un exemple simple

Question 1 : Rédiger un modèle `exemple.imi` au format IMITATOR qui correspond à l'automate ci-dessous :



Question 2 : Générer le graphe des états symboliques.

Question 3 : À partir de l'espace d'états obtenu à la question précédente, appliquer manuellement l'algorithme EF-synthesis afin de synthétiser la contrainte garantissant que l'état de contrôle l_3 n'est pas accessible.

Question 4 : Appliquer automatiquement EF-synthesis avec IMITATOR afin de déterminer les valeurs de paramètres permettant d'accéder à l'état de contrôle l_5.

Que signifie ce résultat ? Comment pouvez-vous l'expliquer ?

Question 5 : On considère l'instance des paramètres suivante :

$$p_1 = 1 \quad p_2 = 3$$

On pourrait montrer (par exemple avec Uppaal) que, pour cette instance des paramètres, l'état de contrôle l_3 n'est pas accessible. Nous allons le vérifier directement avec IMITATOR.

Pour cela :

1. instancier les paramètres dans le modèle : dans l'en-tête du modèle, remplacer `p1` par `p1=1`, et `p2` par `p2=3`.
2. générer le graphe des états symboliques.

L'état de contrôle l_3 est-il accessible ? Comment peut-on le vérifier ?

(à la suite de cette question, annuler les modifications faites au modèle)

Question 6 : Grâce à IMITATOR, nous allons synthétiser d'autres valeurs des paramètres pour lesquelles l'état de contrôle l_3 n'est pas accessible.

Effectuer une synthèse de paramètres par méthode inverse. Quelle est la contrainte résultat ? Garantit-elle bien que l_3 n'est pas accessible ? Comment peut-on le vérifier ?