Examen de MI047 - CPS

Frédéric Peschanski & Romain Demangeon

Mai 2015

Durée : 2 heures **Barème** donné à titre indicatif.

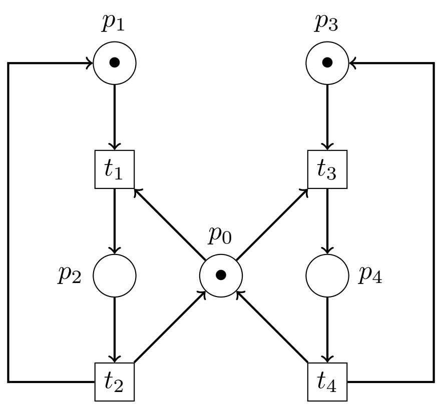
Documents autorisés : Notes de cours

ATTENTION : les 3 exercices de l'énoncé devront être rendus sur des feuilles séparées.

Exercice 1 : Spécifications et tests ($\approx$ 7 points)

L'objectif de cet exercice est de définir une interprétation des *Réseaux de Petri* (RdP) dans le langage de spécification de CPS. Il existe une multitude de variantes des RdP et nous allons considérer une version simple ici.

Un exemple de RdP est proposé ci-dessous :



Ce RdP est composé :

— des *places* $p0, p1, p2, p3, p4$ contenant des *jetons* (*tokens*) représentés par le symbole $\bullet$,

— des *transitions* $t1, t2, t3, t4$,

— d'un ensemble d'*arcs* dirigés reliant chacun soit une place p_i à une transition t_j, soit une transition t_k à une place t_l. Pour un arc (p_i, t_j) on dit que p_i est une *place entrante* de la transition t_j et pour tout arc (t_k, p_l) on dit que p_l est une place sortante de t_k.

La relation entre l'ensemble des places et les jetons qu'elles contiennent se nomme le *marquage* (*marking*) du RdP. C'est en quelque sorte son état.

Le principe de fonctionnement des RdP consiste en un déplacement de jeton(s) de place(s) en place(s) correspondant au franchissement d'une transition t clairement identifiée.

Pour que le franchissement d'une transition t soit possible, il faut que toute place p entrante de t contienne au moins un jeton. Si c'est le cas alors :

— exactement un jeton sera consommé dans chaque place p entrante t

— exactement un jeton sera produit dans chaque place p' sortante de t.

Question 1.1. Donner le RdP correspondant au franchissement de la transition t_1 sur l'exemple ci-dessus. Si ce franchissement n'est pas possible alors justifier. Quelles sont les séquences $t_i, t_j, t_k, \ldots$ possibles ? Selon vous que modélise ce RdP ? Quels sont les rôles des places p_0, p_2 et p_4 ?

Question 1.2. Définir dans le langage de spécification du cours un service de nom Petri permettant de modéliser un RdP en général.

Conseils de modélisation :

On supposera l'existence d'un type Id correspondant aux identifiants de places $p_0, p_1, \ldots$ et de transitions $t_1, t_2, \ldots$.

Puisque l'on souhaite uniquement modéliser la dynamique des RdP, dans la section des constructeurs ont utilisera en paramètre la description structurelle du réseau : les places, les transitions et les arcs.

Dans la section des observateurs, on ajoutera en complément des observateurs nécessaires au modèle :

- un observateur permettant de compter le nombre total de jetons (*tokens*) présents dans le réseau.
- un observateur permettant de tester si le RdP est en interblocage (*deadlocked*) ou non.
- un observateur permettant de tester si une transition t donnée est franchissable (*fireable*) ou non

Deux opérateurs (au moins) seront proposée :

- un opérateur permettant de franchir une transition t donnée (si elle est franchissable)
- un opérateur permettant le passage de l'état courant du Rdp à un état suivant possible.

Remarque : on s'attachera à la concision et la lisibilité de la spécification. Ainsi on pourra introduire des observateurs permettant de simplifier les écritures, et on fera attention à l'étape importante de minimisation.

Exercice 2 : Promela/Spin ($\approx$ 5 points)

Question 2.1. Donner un modèle *Promela* du RdP de l'exercice précédent (dans son état initial).

Le modèle ne devra comporter qu'une unique définition de processus actif. Pour chaque transition t_i du RdP franchie, on affichera (par `printf`) le nom de cette transition.

Voici une trace d'exécution possible du modèle :

```
$ spin -u30 petri.pml
      t1
      t2
      t3
      t4
      t3
-------------
depth-limit (-u30 steps) reached
...
```

Question 2.2. On peut montrer que le RdP considéré est *1-safe*, ce qui signifie qu'il y a au plus un jeton par place. Expliquer les modifications nécessaires au modèle pour vérifier cette propriété.

Question 2.3. Exprimer une propriété qui vous semble importante concernant le couple de places $(p2, p4)$ sur ce modèle, et qui n'est plus vérifiée si la place p_0 contient 2 jetons. Expliquer les modifications nécessaires au modèle pour vérifier cette propriété et donner un contre-exemple dans le cas ou p_0 contient 2 jetons.

Question 2.4. Le RdP proposé n'est clairement pas équitable, expliquer pourquoi. Proposer une modification du modèle Promela (sans modification du RdP proprement dit) permettant de retrouver cette propriété d'équité.

Exercice 3 : Logique de Hoare ($\approx$ 10pts)

Un polynôme P peut s'écrire $\sum_{i=0}^{n} a_i.X^i$, (ou, informellement : $a_0 + a_1.X + \cdots + a_{n-1}.X^{n-1} + a_n.X^n$). Les a_i sont les *coefficients* de P. Si `T` est un tableau de nombres de longueur $\geq (n+1)$, on note par $\mathbf{Pol}_n(\mathtt{T})$ le polynôme *représenté par* `T`, formellement défini par :

$$\sum_{i=0}^{n} \mathtt{T}[i].X^i.$$

On note par $\mathbf{Pol}(\mathtt{T})$ le polynôme $\mathbf{Pol}_{\mathtt{T.length}-1}(\mathtt{T})$ c'est à dire :

$$\sum_{i=0}^{\mathtt{T.length}-1} \mathtt{T}[i].X^i.$$

On note 0 le polynôme nul et 1 le polynôme unité.

Par exemple, si $\mathtt{T} = [3; 2; 0; -6]$ on a $\mathbf{Pol}(\mathtt{T}) = 3 + 2.X - 6.X^3$.

Question 3.1. : Degré d'un polynôme (≈ 2pts) Le degré d'un polynôme est son plus grand coefficient non nul. Soit le code suivant, calculant le degré d'un polynôme :

```
DEGRE(T):
    j := 0;
    r := 0;
    while j != T.length
        {
        if T[j] = 0
            {
            r = r;
            }
        else
            {
            r = j;
            }
        j := j + 1;
        }
```

On considère comme invariant pour la boucle : «le degré de $\mathbf{Pol}_{(j-1)}(\mathtt{T})$ est r».

Montrer **formellement** [1] :

$$\{\}\mathtt{DEGRE(T)}\{\text{le degré de } \mathbf{Pol}(\mathtt{T}) \text{ est } \mathtt{r}\}$$

Question 3.2. : Addition de deux polynômes (≈ 3pts) Soit le code suivant, réalisant l'addition de deux polynômes représentés par des tableaux :

```
ADD(T1,T2):
    j := 0;
    while j != T1.length
        {
        R[j] = T1[j] + T2[j];
        j := j + 1;
        }
```

On supposera que T1, T2, R ont même longueur.

1. Trouver un invariant et un variant de boucle.
2. Montrer **formellement** :

$$\{\}\mathtt{ADD(T1,T2)}\{\mathbf{Pol}(\mathtt{R}) = \mathbf{Pol}(\mathtt{T1}) + \mathbf{Pol}(\mathtt{T2})\}$$

Question 3.3. : Multiplication de deux polynômes (≈ 4pts) Soit le code suivant, réalisant la multiplication de deux polynômes représentés par des tableaux :

```
MULT(T1,T2):
    j := 0;
    while j != T1.length
        {
        k := 0;
        while k != T1.length
            {
            R[j+k] = T1[j].T2[k];
            k := k + 1;
```

1. En détaillant les étapes et en expliquant, à chaque étape, quelle règle de la logique est utilisée.

```
        }
    j := j + 1;
    }
```

On supposera que T1 et T2 ont la même longueur et que $\texttt{R.length} = 2 \times \texttt{T1.length}$

1. En supposant qu'initialement $\mathbf{Pol}(\texttt{R}) = 0$, montrer que «$\mathbf{Pol}(\texttt{R}) = \mathbf{Pol}_{j-1}(\texttt{T1}).\mathbf{Pol}(\texttt{T2}) + (\texttt{T[j]}.X^j).\mathbf{Pol}_{k-1}(\texttt{T2})$» est un invariant de la boucle interne. Trouver un variant de la boucle interne.
2. En supposant qu'initialement $\mathbf{Pol}(\texttt{R}) =)0$, montrer que «$\mathbf{Pol}(\texttt{R}) = \mathbf{Pol}_{j-1}(\texttt{T1}).\mathbf{Pol}(\texttt{T2})$» est un invariant de la boucle externe. Trouver un variant de la boucle externe.
3. Montrer **formellement** :

$$\{\mathbf{Pol}(\texttt{R}) = 0\}\texttt{MULT}(\texttt{T1}, \texttt{T2})\{\mathbf{Pol}(\texttt{R}) = \mathbf{Pol}(\texttt{T1}).\mathbf{Pol}(\texttt{T2})\}$$

.

Question 3.4. : Developpement (bonus) On ajoute à la logique de Hoare une règle pour l'appel de procédure :

$$(\mathbf{Call})\ \frac{\{P\}\texttt{PROC}(\texttt{x1}, ..., \texttt{xn})\{Q\}}{\{P[\texttt{v1}/\texttt{x1}, ..., \texttt{vp}/\texttt{xp}]\}\texttt{PROC}[\texttt{v1}, ..., \texttt{vn}]\{Q[\texttt{v1}/\texttt{x1}, ..., \texttt{vp}/\texttt{xp}]\}}$$

PROC est une procédure (un programme) définie préalablement de paramètres $\texttt{x1}, ..., \texttt{xn}$ et $\texttt{PROC}[\texttt{v1}, ..., \texttt{vn}]$ un appel à cette procédure avec les arguments $\texttt{v1}, ..., \texttt{vn}$. La formule $P[\texttt{v}/\texttt{x}]$ est la formule P dans laquelle on a remplacé toutes les occurences de x par v.

Soit le code suivant, développant un produit de plusieurs polynomes stockés dans L (qui est un tableau de tableau).

```
DEVELOP(L):
    t := 0;
    while t != L.length
        {
        ZEROS[R];
        MULT[D,L[t]];
        D := R;
        t := t + 1;
        }
```

On suppose que tous les tableaux $\texttt{L[i]}$ ont la même longueur et que $\texttt{D.length} = \texttt{L[1].length} \times \texttt{L.length}$.

On suppose que $\{\}\texttt{ZEROS(T)}\{\mathbf{Pol}(\texttt{T}) = 0\}$, c'est-à-dire que la procédure ZEROS[2] (ré)initialise à 0 le tableau passé en paramètre.

1. En supposant qu'initialement $\mathbf{Pol}_n(\texttt{D}) = 1$ (c'est à dire qu'au début du programme D représente le polynome unité 1), trouver un invariant à cette boucle.
2. En utilisant la règle (**Call**) et le résultat de la question précédente, montrer que :

$$\{\mathbf{Pol}(\texttt{D}) = 1\}\texttt{DEVELOP(L)}\{\mathbf{Pol}(\texttt{D}) = \prod_{t=1}^{\texttt{L.length}} \mathbf{Pol}(\texttt{L[t]})\}$$

2. Son code n'est pas donné ni demandé.