Travaux Pratiques de *Model-checking* n°1

Master d'informatique

———

Introduction à SPIN

Dans tous les exercices, « montrez » signifie « montrez en utilisant l'outil de *model-checking* ».

▸ **Exercice 1.** *Modélisez et vérifiez (ça n'est peut être pas vrai !) l'exclusion mutuelle de la section critique de deux processus $p_i, i \in \{0, 1\}$ exécutant l'algorithme suivant :*

```
while true do
    Section non critique ;
    b_i ← true ;
    while k ≠ i do
        while b_{(i+1)%2} do
            ;
        end
        k ← i ;
    end
    Section critique ;
    b_i ← false
end
```

Au départ, la variable globale k vaut 1.

Vous pourrez utiliser plusieurs méthodes, par exemple une assertion puis une formule de LTL. Si l'exclusion mutuelle n'est pas garantie, vous exhiberez une trace d'exécution le montrant.

▸ **Exercice 2.** *(tiré du SpinLab.) Les* réseaux de Petri *sont une représentation formelle du comportement de processus concurrents. On considère celui de la Figure 1. Les cercles sont les* places*, qui peuvent contenir un ou plusieurs jetons, représentés par un point dans la place. Sur la figure qui est l'état initial du système, on trouve deux jetons : un dans la place 1, l'autre dans la place 4. Les t_i sont des* transitions*. La transition t_2 ne peut être utilisée que si un processus est dans p_2, et un autre dans p_4. L'effet de l'utilisation de la transition est de supprimer un jeton de p_2, un autre de p_4, et de mettre un jeton dans p_3. La transition t_3 ne peut être utilisée que si un jeton se trouve*

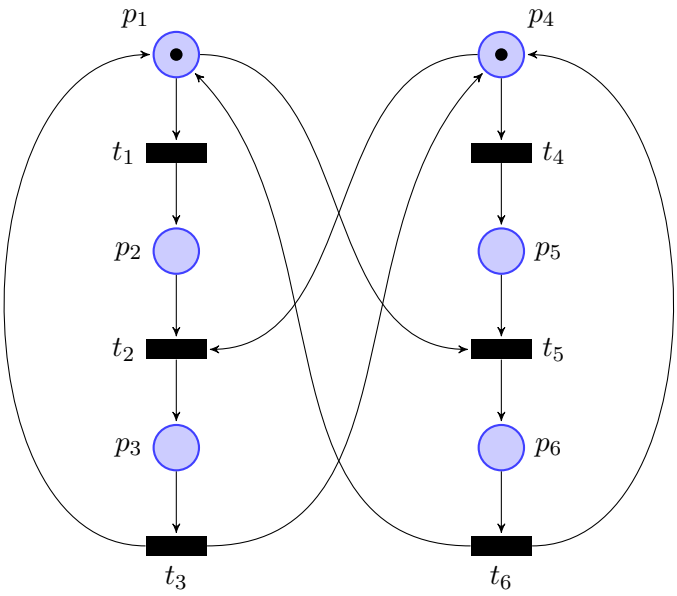

FIGURE 1 – Un réseau de Petri

dans p_3, et son effet est de supprimer un jeton de p_3 et d'ajouter un jeton dans p_1 et un autre dans p_4. On limitera le nombre de jetons à 255 par place.

Modéliser le réseau de Petri. Vous utiliserez un tableau de places, chaque élément étant un octet représentant le nombre de processus dans cette place. Le système est constitué d'un seul processus, dont le rôle est, dans une boucle infinie, de réaliser une transition.

Répondez aux questions suivantes en utilisant SPIN. Existe t-il un parcours pour lequel le système bloque ? Montrez que dans la configuration initiale de la Figure 1, le nombre de jetons ne sera jamais supérieur à 2.

▸ **Exercice 3.** *Le problème des philosophes, quand ils sont 4, peut être modelisé par le réseau de Petri de la Figure 3. Représentez ce réseau de Petri en Promela (comme dans l'exercice précédent, votre modèle ne sera exécuté que par un seul processus Promela). Demandez à SPIN la réponse aux questions suivantes :*

- *peut-il y avoir interblocage ?*
- *deux philosophes peuvent-ils manger simultanément ?*
- *un philosophe qui souhaite manger peut-il mourir de faim (hors cas d'interblocage) ?*

▸ **Exercice 4.** *Un processus émet des messages à destination d'un récepteur à travers un canal parfait. Les messages sont $0, 1, \ldots, N-1, 0, 1, \ldots$ (un octet modulo une constante N). Chaque transmission a un* numéro de séquence, *qui est un bit : 0 ou 1. Le comportement de l'émetteur est le suivant : il émet son $i$ème message dans une trame de type MSG avec le numéro de séquence s. Tant qu'il n'a pas reçu un acquittement*

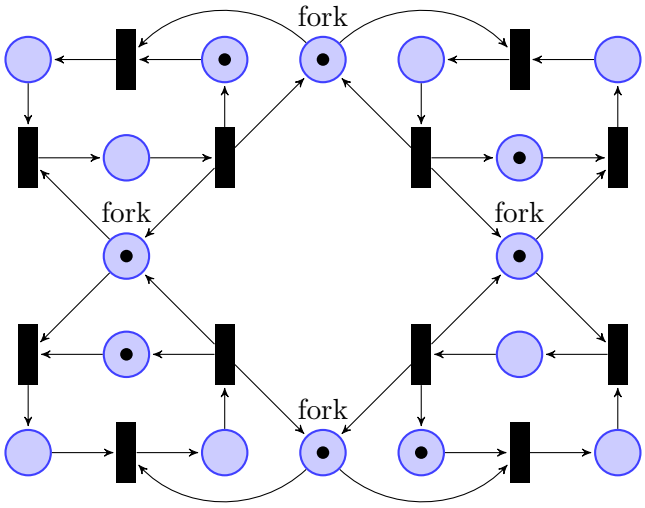

FIGURE 2 – Un réseau de Petri modélisant le dîner de 4 philosophes. Chaque fourchette est représentée par un jeton. Chaque philosophe a 3 états possibles, chacun représenté par une place. Au départ, chaque philosophe pense (premier état), puis il acquiert une fourchette, et souhaite manger. Pour celà, il lui faut une autre fourchette (second état). Il mange (troisième état), puis rend simultanément les deux fourchettes. Le philosophe est dans l'état quand le jeton est dans la place correspondante.

(type de message ACK, numéro de séquence s) pour cette trame, il peut à chaque instant ré-emettre la trame, ou bien de ne rien faire. De son coté, le récepteur acquitte chaque trame reçue. Tant qu'il ne reçoit pas le message suivant, il continue d'acquitter le précédent ou ne fait rien.

Montrer que tout paquet est acquitté avant que l'émetteur ne transmette le suivant.

On suppose maintenant le canal bruité : il peut supprimer des messages. Modifiez votre modèle pour introduire du bruit dans le canal, remontrez la propriété.

▸ **Exercice 5.** *On souhaite modéliser le comportement d'un ascenceur dans un immeuble de 3 étages. Le système est composé :*

- *d'un ascenceur, modélisé par un processus, qui se déplace d'étage en étage. À chaque unité de temps, il a le choix entre*
  - *se déplacer vers l'étage supérieur (s'il n'est pas déjà tout en haut) ;*
  - *se déplacer vers l'étage inférieur (s'il n'est pas déjà tout en bas) ;*
  - *s'arrêter à l'étage auquel il se trouve et demander l'ouverture de la porte. Une fois la porte ouverte, l'ascenceur ne peut plus se déplacer jusqu'à ce que la porte se ferme.*
- *trois portes palières p_1, p_2, p_3, une par étage, chacune modélisée par un processus. Chaque porte s'ouvre à la demande de l'ascenceur. Quand la porte est ouverte, le processus correspondant affiche un message, puis la porte se ferme.*

Implantez un modèle en SPIN. Vous ferez communiquer portes et ascenceur par un

canal. Montrez les propriétés suivantes :

- *la porte ne peut être ouverte qu'à un seul étage à la fois ;*
- *si la porte est ouverte, alors l'ascenceur est à l'étage ;*
- *chaque étage sera toujours servi, c'est-à-dire que l'ascenceur passe par tous les étages (sans nécessairement que la porte n'y soit ouverte).*

***Partie optionnelle** : modifiez vos processus représentant les portes pour modéliser la présence d'un bouton « appel » sur chacune des portes et la présence de trois boutons (un par étage) dans l'ascenceur, avec la sémantique habituelle pour ces boutons. Modifiez le comportement de l'ascenceur pour*

- *qu'il ne demande l'ouverture de la porte à l'étage où il se trouve que si c'est nécessaire ;*
- *tout demandeur soit desservi.*

et celui des portes pour simuler un appui aléatoire sur son bouton « appel ». Quand ce bouton utilisé et que l'ascenceur arrive, un des boutons se trouvant dans l'ascenceur est utilisé (pour spécifier l'étage de destination) avant que l'ascenceur ne reparte.

Prouvez que votre système se comporte bien :

- *les propriétés précédentes doivent toutes être vraies ;*
- *la porte p_i ne s'ouvre que si nécessaire ;*
- *toute requête est correctement servie.*

▸ **Exercice 6.** *Les enfants au front sale.*

Le model-checking peut aussi être utilisé pour vérifier des idées de preuve. C'est ce qui est proposé dans cet exercice.

$N \geq 2$ enfants futés jouent dans la boue. Le père arrive, mécontent du spectacle. Il pose alors la question : « un de vous au moins a le front sale. Levez la main si vous savez si vous avez le front sale ». Chaque enfant n'a d'autre moyen que de regarder le front des autres, et de réfléchir ! Le père répète sa question jusqu'à ce que tous les enfants aient répondu. En au plus $N + 1$ tours, tous les enfants ont répondu.

Essayez avec $N = 3$. Devinez un algorithme et modélisez-le en SPIN. Vérifiez la correction de l'algorithme pour une constante donnée N. Bien sûr, on ne peut pas en déduire que l'algorithme est vrai pour tout N.