Spécification formelle en LOTOS et LotosNT (LNT)
Cahier d'exercices : N° 1

Christian Attiogbé, Meriem Ouederni — Février 2012

Christian.Attiogbe@univ-nantes.fr, meriem.ouederni@univ-nantes.fr

www.sciences.univ-nantes.fr/info/perso/permanents/attiogbe/

Exercice 1

Soit à **modéliser comme un processus**, un ascenceur (ou monte-charge) qui peut effectuer les actions : monter ou descendre.
Les hypothèses de travail sont les suivantes : les deux actions de l'ascenceur se font pas à pas ; par exemple dans un immeuble ce sera étage par étage ; l'ascenceur ne peut pas monter au delà d'un niveau (M), il ne peut pas descendre non plus en dessous d'un niveau (0). Initialement l'ascenceur est au niveau le plus bas(0), et ne peut alors que monter.

1. Donnez $\mathcal{A}$, l'alphabet du processus.
2. Esquissez sous forme graphique (à la manière d'un graphe ou d'un automate), le comportement de l'ascenceur.
3. Transcrivez le processus en LotosNT.

Exercice 2

Comparez les arbres des comportements suivants :

```
select a ; b ; stop [] a ; c ; stop end select
```

et

```
a ; select ( b ; stop [] c ; stop ) end select
```

Exercice 3

Soit un processus dont la trace à chaque exécution donne de façon indéterministe une des séquences de lettres suivantes :

```
dot deco debut debat debout deboat debora eco echo echu euro eure
```

Chaque lettre correspond au nom de l'action effectuée par le processus (comme si il imprimait le nom de chaque action effectuée).

1. Donnez l'arbre de comportement ou arbre de synchronisation décrivant le comportement du processus.
2. Donnez la spécification LOTOSNT correspondante ; on utilisera la lettre correspondant à chaque action comme son abstraction.
3. Donnez l'alphabet d'actions du processus.

Exercice 4

Donnez l'arbre de comportement associé aux procesus décrits par :

```
     a ; b ; c ; stop
|||  d ; b ; stop
```

et par

```
par b in
   a ; b ; c ; stop
|| d ; b ; stop
end par
```

Exercice 5

On rappelle que :
en présence d'un choix, les comportements qui amènent à un ***deadlock*** *ne sont pas pris en compte.*
Donnez alors le comportement équivalent à chacun des processus décrits ci après :

1. `a ; b ; stop || select a ; c ; exit [] c ; b ; stop end select`
2.
```
par  a, b  in
     select  a ; b ; stop [] c ; d ; stop   end select
||   select  a ; b ; stop [] d ; f ; stop   end select
end par
```

Exercice 6

Soit un système composé d'un utilisateur et d'un gestionnaire de cartes d'accès.
Dans ce système l'utilisateur interagit avec le gestionnaire de façon à obtenir ou non l'accès à une salle.
C'est l'utilisateur qui demande l'accès en glissant sa carte dans le lecteur de carte associé au gestionnaire.
Le gestionnaire demande le code d'accès de l'utilisateur puis, soit il autorise l'accès soit il le refuse.
Lorsque l'utilisateur a obtenu l'autorisation d'accès, il va occuper la salle pendant un moment puis en ressort par une porte de sortie en appuyant simplement sur un bouton poussoir.
Lorsque l'utilisateur n'a pas obtenu l'autorisation d'accès, il peut soit recommencer la demande d'accès soit abandonner.

1. Quels sont les processus que vous avez identifié ?
2. Listez les actions effectuées par chaque processus.
3. Esquissez un schéma du système (à la CCS de Milner).
4. Spécifiez le comportement global du système en LOTOS.
5. Complétez le cahier de charges à votre guise et donnez la spécification correspondante.

Exercice 7

Donnez une expression de comportement observationnellement équivalente à

```
hide a in
       i;a;B[a]
    || a;C[a]
end hide
```

on n'utilisera pas l'action interne `i`.

Exercice 8

1. Donnez l'alphabet du processus dont le comportement est le suivant :

```
hide vers in
  par vers in
    bas ; vers ; exit || vers ; haut ; exit
  end par
end  hide
```

2. Donnez une expression de comportement qui lui est observationnellement équivalente.

Exercice 9

Spécifiez le comportement d'un ascenceur qui effectue les actions `monter` et `descendre` mais où on ne veut observer que les actions `monter` ; par exemple pour compter le nombre d'actions `monter` effectuées.

Exercice 10

Quelle est la valeur transmise au processus `VA` dans la spécification suivante :

```
let x : Nat_Sort = 11 in
    let prm : Nat_Sort = x + x in
        let x : Nat_Sort = 99 in
  VA(prm)
```

Exercice 11

Réécrivez en utilisant la structure de choix, le comportement suivant :

```
   ecrire ; MEMOIRE(0)
[] ecrire ; MEMOIRE(1)
[] lire ; MEMOIRE(0)
[] lire ; MEMOIRE(1)
```

Exercice 12

On veut spécifier en LOTOSNT un système de régulation de température dans une pièce.
Le système lit la température ambiente de la pièce puis rechauffe, refroidit ou ne fait rien.
Si la température lue est inférieure à un seuil donné comme paramètre au système, on lance un sous-système de chauffage. Si la température lue est supérieure au seuil, le système lance plutôt un sous-système de refroidissement.

1. Donnez une première spécification où on ne se soucie pas du fait qu'il doit y avoir exclusivité entre les sous-systèmes de chauffage et refroidissement.
2. Considérez maintenant le cas où le système (de chauffage ou de refroidissement) éventuellement en marche doit être arrêté avant de lancer l'autre.

Exercice 13

Ecrivez la spécification LOTOSLNT d'un processus qui accepte sur une porte `pythag` trois entiers naturels `a, b, c` si ils vérifient l'équation de PYTHAGORE ($a^2 + b^2 = c^2$).

Exercice 14

Donnez en LOTOSNT la spécification d'un processus qui modélise le jeu du mot caché.
Le processus a un mot en paramètre. Ce sera le mot caché. Ensuite, le processus accepte sur une porte dite `pjeu` un mot donné par un utilisateur.
Le but est de trouver le mot caché. Le processus sort (termine avec succès) quand le mot est trouvé sinon il boucle en allant relire un autre mot, et ainsi de suite.
On suppose la donnée de la sorte `Mot_Sort` avec les opérations de comparaisons de mots.

Exercice 15

Soit un système composé de deux processus dits *ouvrier* et des ressources *maillet* et *marteau*.
Les processus doivent usiner des pièces en se servant soit du *marteau* soit du *maillet*.
Il n'y a qu'un *maillet* et qu'un *marteau* ; les *ouvriers* s'en servent donc de façon exclusive.
Le comportement informel d'un *ouvrier* est comme suit :
– soit il demande le *marteau* pour usiner sa pièce puis il libère le *marteau* ;
– soit il demande le *maillet*, usine sa pièce puis dépose le *maillet*.
Ce comportement est indéterministe.
On vous suggère de représenter le *marteau* et le *maillet* chacun comme un processus de type ressource. Le comportement d'un tel processus revient à se synchroniser avec le demandeur de la ressource, puis la libération de la ressource puis, la ressource devient de nouveau disponible pour une autre demande.

1. Donnez en LOTOSNT (mais sans les données) les comportements du
   – processus *ouvrier*
   – *maillet*
   – *marteau*
   – du système composé de deux *ouvriers* et des ressources *maillet* et *marteau*.
2. On considère maintenant que le choix par l'*ouvrier* d'une ressource ou une autre dépend de la nature de la pièce à usiner. L'*ouvrier* a donc en entrée une pièce p ; selon la nature de p (difficile, facile) il se servira alors soit du *marteau* (pièce difficile) soit du *maillet* ou du *marteau* (pièce facile).

   En tenant compte de l'utilisation des données, donnez la spécification en LOTOSNT des processus et du système.