TD 1. Premières Spécifications - Corrections

Jean-Louis BOULANGER

Décembre 2000

Résumé

Le but de ce premier TD est de manipuler les notions liées à la théorie des ensembles et aux prédicats.

Keywords : Exercices, Théorie des ensembles, Prédicats.

Table des matières

1 Introduction

Les exercices de cette section sont inspirés du §7 du [Sommerville].

Les exercices de ce TD consistent à traduire un texte en français constitué d'affirmation sous la forme de Prédicat.

2 Exercice 1

énoncé : Dans un tableau d'entiers, il y a au moins une valeur négative.

solution :

Pour cette première solution, nous utilisons un sous-ensemble des **INT**.

```
MACHINE
       EX1
VARIABLES
       TABLEAU
INVARIANT
        TABLEAU <: INT
&       #ii.(ii:TABLEAU & (ii <0))
INITIALISATION
TABLEAU :(    TABLEAU <: INT
           &  #ii.(ii:TABLEAU & (ii <0))
         )
END
```

Une autre façon de décrire la propriété serait

$$card(\{x|x : TABLEAU \wedge x < 0\} \ > \ 0)$$

Dans le cadre de cette seconde solution, on considère que le tableau est modélisé sous forme d'une fonction totale.

```
MACHINE
        EX1_bis
SETS
        INTERVAL_ENTIER
VARIABLES
        TABLEAU
INVARIANT
        TABLEAU : INTERVAL_ENTIER --> INT
&       #ii.(ii:INTERVAL_ENTIER & TABLEAU(ii)<0)
INITIALISATION
        TABLEAU :(     TABLEAU : INTERVAL_ENTIER --> INT
                   &   #ii.(ii:INTERVAL_ENTIER & TABLEAU(ii)<0)
                 )
END
```

A partir de la première solution, il est possible de montrer qu'une version modifiée de la seconde solution est un raffinement possible.

```
REFINEMENT
      EX1_1

REFINES
       EX1

SETS
      INTERVAL_ENTIER

VARIABLES
      TABLEAU1

INVARIANT
      TABLEAU1 : INTERVAL_ENTIER --> INT
&     TABLEAU1[INTERVAL_ENTIER] = TABLEAU
&     #ii.(ii:TABLEAU1[INTERVAL_ENTIER] & (ii < 0))

INITIALISATION
      TABLEAU1 :(   TABLEAU1 : INTERVAL_ENTIER --> INT
                  & #ii.(ii:TABLEAU1[INTERVAL_ENTIER] & (ii<0))
                )
END
```

Si on avait mis l'invariant suivant, la preuve ne serait pas automatique.

$$\#ii.(ii : INTERVAL\_ENTIER \wedge TABLEAU1(ii) < 0)$$

Bilan de la preuve :

```
+----------------+----+-----+-----+-----+-----+-----+-----+
| COMPONENT      | TC | POG | Obv | nPO | nUn | %Pr | BOC |
+----------------+----+-----+-----+-----+-----+-----+-----+
| EX1            | OK | OK  |   1 |   1 |   0 | 100 | OK  |
| EX1_1          | OK | OK  |   3 |   2 |   0 | 100 | OK  |
| EX1_2          | OK | OK  |   3 |   3 |   1 |  66 | OK  |
| EX1_bis        | OK | OK  |   2 |   1 |   0 | 100 | OK  |
+----------------+----+-----+-----+-----+-----+-----+-----+
```

3 Exercice 2

énoncé : Soit un ensemble de nombres naturels, le plus petit vaut 20 et le plus grand est supérieur à 250.

solution :

Une première version qui utilise les opérateurs B .

```
MACHINE
        EX2

VARIABLES
        TABLEAU

INVARIANT
        TABLEAU <: INT
&       min (TABLEAU) = 20
&       max (TABLEAU) > 250

INITIALISATION

TABLEAU :(   TABLEAU <: INT
           & min (TABLEAU) = 20
           & max (TABLEAU) > 250
         )

END
```

On peut construire une seconde version en utilisant la construction d'ensemble par compréhension. La notion de **le plus petit vaut 20** est décomposé en deux : aucun élément inféieur à 20 et un seul élément égal à 20.

$$card(\{ii|ii : TABLEAU \wedge ii < 20\}) = 0$$
$$\wedge$$
$$card(\{ii|ii : TABLEAU \wedge ii = 20\}) = 1$$

De même pour la notion de **le plus grand est supérieur à 250**, elle se note

$$card(\{ii|ii : TABLEAU \wedge ii > 250\}) = 1$$

4 Exercice 3

énoncé : Étant donné une série de processus auxquels on associe un attribut appelé DELAI, et qui peuvent être dans l'état **en_cours**, **en_attente** ou **stoppé**, il n'y a pas de processus qui soit en attente et dont la valeur du délai soit supérieure à 2.

solution L'ensemble de tous les processus est noté **PROCESSUS**. L'ensemble des processus actifs est noté **processus** avec une relation évidente d'inclusion.

L'état d'un processus est décrit par un type énuméré noté **STATE** qui reprend les valeurs de l'énoncé.

La notion de *DELAI* est liée à un entier. On lie la notion de *DELAI* à la notion de processus au travers d'une fonction.

```
MACHINE
        EX3

DEFINITIONS
        MAX_DELAI       == 5
;       DELAI           == 1 .. MAX_DELAI

SETS
        PROCESSUS
;       STATE           = { en_cours, en_attente, stoppe }

VARIABLES
        processus
,       delai
,       state

INVARIANT
        processus       <:       PROCESSUS
&       delai            :       processus --> DELAI
&       state            :       processus --> STATE
&       card ( { pp |   pp : processus
                      & state(pp) = en_attente
                      & delai(pp) > 2 } ) = 1

INITIALISATION
   processus
,  delai
,  state  :(
               processus <:  PROCESSUS
             & delai      :  processus --> DELAI
             & state      :  processus --> STATE
             & card ( { pp | pp : processus &
                             state(pp) = en_attente &
                             delai(pp) > 2 } ) = 1
           )

END
```

5 Exercice 4

énoncé : Il existe un tableau de 100 capteurs auxquels sont associées des soupapes de contrôle. Les capteurs peuvent prendre les valeurs haut ou bas et les soupapes de contrôle peuvent être ouvertes ou fermées. Si un capteur est à l'état haut, la soupape associée est fermée.

solution : Le tableau des capteurs est modélisé par la fonction **capteur_state** qui a pour domaine **CAPTEUR** et codomaine **CAPTEUR_STATE**.

Afin d'être plus abstrait, le tableau est donc codé comme une fonction.

L'ensemble **CAPTEUR** représentant l'ensemble de tous les capteurs est soumis à la propriété $card(CAPTEUR) = MAX\_CAPTEUR$.

```
MACHINE
        EX4

DEFINITIONS
        MAX_CAPTEUR == 100

SETS
        CAPTEUR
;       CAPTEUR_STATE   = { haut, bas}
;       SOUPAPE
;       SOUPAPE_STATE   = { ouverte, fermee }

PROPERTIES
        card(CAPTEUR) = MAX_CAPTEUR

VARIABLES
        capteur_state
,       lien_capteur_soupape
,       soupape_state

INVARIANT
   capteur_state        : CAPTEUR --> CAPTEUR_STATE
&  soupape_state        : SOUPAPE --> SOUPAPE_STATE
&  lien_capteur_soupape : CAPTEUR --> SOUPAPE
&  (!xx.((xx : CAPTEUR)
          => ( (capteur_state(xx) = haut)
               => (soupape_state(lien_capteur_soupape(xx))= fermee)
        )    )
   )

INITIALISATION
  capteur_state
, lien_capteur_soupape
, soupape_state
  :(
       capteur_state        : CAPTEUR --> CAPTEUR_STATE
     & soupape_state        : SOUPAPE --> SOUPAPE_STATE
     & lien_capteur_soupape : CAPTEUR --> SOUPAPE
     & (!xx.((xx : CAPTEUR)
             => ( (capteur_state(xx) = haut)
                  => (soupape_state(lien_capteur_soupape(xx))= fermee))
       )    )
  )
END
```

6 Nombres de fibonacci

Les *nombres de fibonacci* sont définis par les relations de récurrence suivantes :

$$fib(0) = 1$$
$$fib(1) = 1$$
$$pour\ n\ >\ 0\ on\ a\ fib(n+1) = fib(n) + fib(n-1)$$

```
MACHINE
Fibonnacci

CONSTANTS
    fibonnacci

PROPERTIES
        fibonacci : NAT --> NAT
&       fibonacci (0) = 1
&       fibonacci (1) = 1
&       !xx.( (xx:NAT1)
=> (fibonacci(xx+1) = fibonacci(xx)+fibonacci(xx-1))
)

END
```

7 Factorielle

Les *factorielles* sont définies par les relations de récurrence suivantes :

$$fact(0) = 1$$
$$pour\ n\ >\ 0\ on\ a\ fact(n) = n * fact(n-1)$$

```
MACHINE
    fact

CONSTANTS
    facto

PROPERTIES
    facto : NAT --> NAT
&   facto(0) = 1
&   (!(nn). (nn : NAT & nn>0  => facto(nn) = nn * facto(nn-1)))

END
```

Références

[Sommerville] Ian Sommerville, *Le génie logiciel*,
Addison-Wesley,
1992