TD 2. Premières Spécifications - Corrections

Jean-Louis BOULANGER

Décembre 2000

Résumé

Le but de ce premier TD est de manipuler les substitutions généralisées.

Mots-Clés : Substitution Généralisée.

Table des matières

1 Manipulation de substitution

```
[xx := 1](xx > yy)                                  == (1 > yy)
[xx, yy := xx+1, xx] (xx > yy)                      == (xx+1 > xx)
[xx, yy, zz := 1, 2, zz-xx] (xx > yy & zz : NAT)  == (1> 2 & zz-xx : NAT)
```

Pour la réduction de l'expression suivante :

```
[@z.(P(z) => x:= z)] Q(x)
```

on obtient

$$\forall z P(z) => Q(z)$$

2 Exercice 2

```
[PRE xx > 0 THEN yy := yy/xx END ] (yy >1)
==
( xx>0 & yy/xx > 1 )
```

qui se réduit à $xx > 0 \wedge yy > xx$

```
[CHOICE ss := ss \/{bb} || bb := TRUE or bb := FALSE END] (xx : ss)
==
( xx:ss\/{xx} & xx: ss)
```

qui se réduit à $xx \in ss$

```
[ANY vv WHERE vv > xx & vv : NAT THE xx := vv END] (xx : ss)
==
(! vv.vv>xx & vv:NAT => vv : ss)
```

3 Exercice 3

3.1 Enoncé

Soit l'extrait de la machine **RESERVATION**

```
MACHINE
     RESERVATION (max_siege)

VARIABLES
     siege

INVARIANT
     siege : 0 .. max_siege

OPERATIONS

reserver(x) =
  ``A définir''

annuler (nn) =
  `` A définir''

END
```

Questions :

- Hormis le corps des opérations, cette machine est-elle correcte ? Si non corrigez-la !
- Compléter les deux opérations *reserver* et *annuler*.

3.2 Version corrigée

Il faut ajouter :

- une clause `CONSTRAINTS` permettant de typer le paramètre formel **max_siege**,
- une clause `INITIALSATION` permettant de définir les valeurs initiales des variables du modèle.

Remarque : Les identificateurs d'un modèle B doivent toujours comporter au moins deux caractères.

```
MACHINE
     RESERVATION (max_siege)

CONSTRAINTS
     max_siege : NAT

VARIABLES
     siege

INVARIANT
```

```
     siege : 0 .. max_siege

INITIALISATION
     siege := max_siege

OPERATIONS

reserver(xx) =
  ‘‘A définir’’

annuler (nn) =
  ‘‘ A définir’’

END
```

3.3 Codage des opérations

On parle ici de programmation offensive, car chaque opération est gardée par une *substitution* du type *pré-condition*, qui permet de filtrer les appels.

3.3.1 Opération *reserver*

```
reserver(xx) =
PRE
      xx : NAT
&     siege > xx
THEN
      siege := siege - xx
END;
```

3.3.2 Opération *annuler*

```
annuler (nn) =
PRE
      nn : NAT
&     ((siege + nn) <= max_siege)
THEN
      siege := siege + nn
END
```

4 Exercice 4

4.1 Enoncé

Construire une machine qui permet de définir un **TABLEAU** avec au moins deux opérations :

- *put* qui permet de valuer la case *xx* avec la valeur *value*,
- *get* qui permet de récupérer la valeur se trouvant à la case *xx*.

4.2 Codage

Le listing suivant donne une solution.

```
MACHINE
     ARRAY (INDEX, VALEUR)

VARIABLES
     array

INVARIANT
     array : INDEX --> VALEUR

INITIALISATION
     array :(array : INDEX --> VALEUR)

OPERATIONS
     put (xx, valeur) =
     PRE
        xx : INDEX
     &  valeur : VALEUR
     THEN
        array := array <+ { xx |-> valeur}
     END;

     valeur <-- get(xx) =
     PRE
        xx : INDEX
     THEN
        valeur := array(xx)
     END

END
```

Remarques :

- la surharge d'une fonction se note
  `array := array <+ { xx |-> valeur}`
  ou `array(xx):= valeur`,
- la substitution `array :(array : INDEX --> VALEUR)` signifie que la variable *array* est un ensemble de valeurs conformes à l'invariant.