TD n°2 Méthode B : Machines abstraites

AI20/LO22

Exercice 1

On cherche à spécifier les différents services utiles au fonctionnement d'un système de location de PRODUITS. On distingue ici la notion de PRODUIT (« un film DVD », « une œuvre littéraire particulière » …) de celle d'EXEMPLAIRE : identifiant une instance physique d'un produit (« ce DVD que j'ai loué de tel film », « mon exemplaire de tel livre »…). Dans un magasin de location on peut trouver plusieurs exemplaires d'un même produit. La machine suivante est un squelette d'une telle spécification que vous devez compléter (cf. parties en gras) ou commenter en répondant aux questions qui suivent :

1°) Expliquez la différence entre produit et PRODUIT.
2°) Expliquez la différence entre exemplaire et EXEMPLAIRE.
3°) Commentez la troisième ligne de l'invariant.
4°) Donnez une définition pour l'ensemble des exemplaires non loué.
5°) Donnez une définition pour l'ensemble des produits disponibles à la location (i.e. dont au moins un exemplaire n'est pas loué)
6°) Expliquez ce que fait l'*opérationMystère* et sous quelles conditions elle est utilisable.
7°) Donnez les obligations de preuves associées à l'*opérationMystère* et précisez celles qui sont triviales.
8°) Complétez la précondition de l'opération *supprimerExemplaire*.
9°) Complétez l'opération *supprimerProduit*.
10°) Spécifiez l'opération *louer*.
11°) Spécifiez l'opération *restituer*.
12°) On souhaite ajouter une contrainte invariante, imposant qu'aucun client (i.e. compte) ne puisse louer plus de 10 produits (ou plutôt 10 exemplaires) dont au maximum 3 DVD. Que faut-il ajouter à l'invariant pour exprimer cette contrainte ? Quelles opérations seraient impactées par cette nouvelle contrainte ?

```
MACHINE
  location
SETS
  TYPE={K7,DVD,LIVRE} ; COMPTE ; PRODUIT ; EXEMPLAIRE
VARIABLES
  produit, aPourType, estUnExemplaireDe, compte, estLouePar
INVARIANT
  produit ⊆ PRODUIT ∧
  compte ⊆ COMPTE ∧
  estUnExemplaireDe ∈ EXEMPLAIRE +->> produit ∧ /* à commenter*/
  aPourType ∈ produit --> TYPE ∧
  estLouePar ∈ exemplaire +-> compte
  /* à compléter avec contrainte de la question 12 (indépendante des autres questions) */
DEFINITIONS
  exemplaire == dom(estUnExemplaireDe) ;
  exempl(prod) == estUnExemplaireDe~[{prod}] ;
  nbExempl(prod) == card(exempl(prod)) ;
  exemplaireDisponible == /* à compléter avec l'ensemble des exemplaires non loués */
  produitDisponible == /* à compléter avec la définition de l'ensemble des produits disponibles à
la location */

INITIALISATION
  produit, compte, estUnExemplaireDe, aPourType, estLouePar := ∅,∅,∅,∅,∅

OPERATIONS

  ensExemp <--operationMystère(prod,type, nbEx)= /* à commenter */
```

```
  PRE
        prod ∈ PRODUIT-produit ∧
        type ∈ TYPE ∧
        nbEx ∈ NAT1
  THEN
        produit := produit ∪ {prod} ||
        aPourType(prod) := type ||
        ANY ens WHERE
                ens ⊆ EXEMPLAIRE - exemplaire &
                card(ens) = nbEx
        THEN
                ensExemp := ens ||
                estUnExemplaireDe := estUnExemplaireDe ∪ (ens * {prod})
        END
  END;
```

supprimerExemplaire(ex) = /* supprime un exemplaire d'un produit, cela ne doit pas être le dernier du produit, et il ne doit pas être loué */

```
  PRE
        ex ∈ exemplaire ∧ /* à compléter */
  THEN
        estUnExemplaireDe := {ex} <<| estUnExemplaireDe
  END;
```

supprimerProduit(prod) = /* retire un produit avec tous ses exemplaires (nécessite qu'aucun d'eux ne soit loué) */

```
  PRE
        prod ∈ produit ∧
        estLouePar[exempl(prod)] = ∅
  THEN
        /* à compléter */
  END;
```

ex <-- louer(prod,cmpt) = /* Demande de location d'un exemplaire d'un produit par un client ; retourne l'exemplaire loué (cette opération n'est appelable que dans le cas où au moins un exemplaire du produit est disponible)*/

```
        /* à compléter */
```

restituer(ex)= /* Retour d'un exemplaire loué */

```
        /* à compléter */

END
```

Exercice 2

1) Fournir la pré condition de l'opération affectation.

```
MACHINE
truc
SETS
ETUDIANT ; GROUPE; PROF
DEFINITIONS
 représentantsDeGroupe == ran (aPourReprésentant)
VARIABLES
```

```
 profs, étudiants, aPourGroupe, aPourReprésentant
INVARIANT
 profs <: PROF &
étudiants <: ETUDIANT & groupes <: GROUPE &
aPourGroupe : étudiants +-> groupes &
aPourReprésentant : groupes >+> étudiants & aPourReprésentant <: aPourGroupe~
INITIALISATION
profs, étudiants, groupes , aPourGroupe, aPourReprésentant,  : = {},{}, {},{}, {}
OPERATIONS
affectation (et, gp) ==
PRE

THEN
aPourGroupe := aPourGroupe \/ { et |-> gp }
END
END
```

2) Si vous considérez qu'une ou des parties de la spéc ci-après (elle fait partie de la machine truc) est/sont erronée(s) entourez la/les.

```
ajoutReprésentant (et) ==
PRE
 et : ETUDIANT & et : dom (aPourGroupe) & et /: ran (aPourReprésentant)
THEN
 aPourReprésentant := aPourReprésentant \/ { aPourGroupe (et) |-> et} ||
 représentantsDeGroupe := représentantsDeGroupe \/ {et}
END;
```

3) L'invariant est complété comme suit. Si vous considérez qu'une ou des parties de la spéc ci-après (elle fait partie de la machine truc) est/sont erronée(s) entourez la/les.
   profs /\ étudiants = {}
   card (profs) < card (étudiants)

Exercice 3

Soit la machine ci-après. Entourez les lignes qui sont erronées (car soit en contradiction avec d'autres lignes, soit ne respectant pas la notation B et les concepts sous-jacents)

```
Soit
MACHINE
compteur

VARIABLES
cpt
INVARIANT
cpt <:  0..10 /* ligne 1 */
INITIALISATION
cpt := 0     /* ligne 2 */
OPERATIONS
 incrémenter ==
  PRE
   cpt : 0..9   /* ligne 3 */
  THEN
   cpt = cpt +1    /* ligne 4 */
  END
END
```

Remplissez le tableau suivant. L'ensemble des remplacements proposés doit faire que la machine compteur soit correcte et spécifie bien ce qu'on appelle un compteur (nous ne considérons qu'une seule opération). Faites le MINIMUM de corrections possibles pour spécifier cette machine

N° ligne	Pourquoi erronée	Remplacement proposé
1		
4		

Exercice 4

Question 1

Soit

SETS

(1) AA; BB

VARIABLES
(2) R1
INVARIANT
(3) R1 : AA --> BB
INITIALISATION
(4) R1 := { }
OPERATIONS
 AjoutDansR1 (a1, b1) =
  PRE
(5) a1 /: dom (R1) & b1 : BB
  THEN
(6) R1 := R1 \/ {a1 |-> b1}
  END
END

1) Cette spécification est-elle correcte ?
2) D'après vous quelles sont les lignes erronées ?
3) Pour chaque erreur, expliquez en quoi c'est erroné
4) Qu'elle est la ligne qui a entraîné les autres erreurs