Méthodes Formelles

1h – documents autorisés

Exercice (ESC/JAVA)

On considère le code source Java suivant, annoté par une spécification Esc/Java. Six passages de code sont cachés: ils ont été remplacés par les expressions `**EXPR1**`, `**EXPR2**`, `**EXPR3**`, `**EXPR4**`, `**EXPR5**`, `**EXPR6**`.

Question 1. Que font les fonctions g(int [] t, int n) et h(int [] t) ?

Question 2. Que proposez vous pour les expressions `**EXPR1**`, `**EXPR2**`, `**EXPR3**`, `**EXPR4**`, `**EXPR5**`, `**EXPR6**` (de sorte que esc/java accepte le code source) ?

Question 3. Ces fonctions sont-elles sous-spécifiées ? Si oui, expliquez pourquoi.

```
public class A {

    static int idx;

    /*****************************************************************/
    /*@ requires t != null;                                          */
    /*@ requires **EXPR1**                                           */
    /*                                                               */
    /*@ ensures (\forall int j; j>=0 && j<n ==> **EXPR2**);          */
    /*@ ensures \result == t[idx];                                   */
    /*@ ensures idx >=0 && idx < n                                   */
    /*****************************************************************/
    public static int g (int [] t, int n) {
        int i = 1;
        int m = t[0];
        idx=0;

        /*@ loop_invariant i>=0;                                     */
        /*@ loop_invariant i <= n;                                   */
        /*@ loop_invariant **EXPR3**                                 */
        /*@ loop_invariant m == t[**EXPR4**];                        */
        /*@ loop_invariant idx >=0 && idx < n                        */
        while (i<n) {
            if (t[i] > m) {
                m = t[i];
                idx = i;
            }
            i++;
        }
        return m;
    }


    /*****************************************************************************/
    /*@ requires t != null;                                                      */
    /*@ requires t.length >= 2;                                                  */
    /*                                                                           */
    /*@ ensures (\forall int j; j>=0 && j < t.length-1 ==> t[j] <= t[j+1]);      */
    /*****************************************************************************/
    public static void h (int [] t) {
        int i,m;

        /*@ loop_invariant i>=0                                                  */
        /*@ loop_invariant i < t.length                                          */
        /*@ loop_invariant (\forall int j; **EXPR5** ==> t[j] <= t[j+1]);        */
        /*@ loop_invariant
          i==t.length-1 || (\forall int j; 0 <= j && j < i+1 ==> t[j] <= t[**EXPR6**]);
         */
        for (i=t.length-1; i>0; i--) {
            m = g(t,i+1);
            t[idx] = t[i];
            t[i] = m;
        }
    }
}
```

Exercice (SMV)

La modélisation SMV jointe décrit un mécanisme de sas de décompression. Le sas de décompression possède une porte donnant sur l'extérieur et une porte donnant sur une cabine intérieure. La pression extérieure est trop basse, et la pression dans la cabine doit être maintenue normale. Il y a un mécanisme de pressurisation à l'intérieur du sas. La modélisation SMV vise à vérifier que la pression de la cabine restera normale quoiqu'il arrive.

Question 1. Expliquez la ligne

```
next(etat) := { 0, 1 };
```

du module `bouton`.

Question 2. Dans le module `porte`, expliquez la spécification suivante :

```
SPEC AG (fermeture.Actif & !ouverture.Actif -> AX Fermee)
```

Est-elle satisfaite ?

Question 3. Rédigez une spécification pour le module `porte` exprimant le fait qu'il existe une trajectoire du système pour laquelle la porte reste fermée en permanence.

Question 4. Rédigez une spécification pour le module `main` exprimant le fait que si la porte intérieure est ouverte et si la pression du sas est basse, alors la pression de la cabine est basse au temps suivant. Cette propriété est-elle satisfaite ?

Question 5. Dans le module `main`, la spécification

```
SPEC AG pression_int = normale
```

est-elle satisfaite ? (vous justifierez votre réponse)

```
--------------------------------------------------------------------------
MODULE bouton
--------------------------------------------------------------------------
VAR
  etat : { 0, 1 };
ASSIGN
  init(etat) := { 0, 1 };
  next(etat) := { 0, 1 };
DEFINE
  Actif := etat=1;

--------------------------------------------------------------------------
MODULE porte(condition)
--------------------------------------------------------------------------
VAR
  etat : { ouverte, fermee, verrouillee, requete_deverrouillage };
  ouverture : bouton;
  fermeture : bouton;
  verrouillage : bouton;
  deverrouillage : bouton;
ASSIGN
  init(etat) := verrouillee;
  next(etat) :=
    case
      etat=ouverte :
      case
        fermeture.Actif : fermee;
        1 : ouverte;
      esac;

      etat=fermee :
      case
        ouverture.Actif : ouverte;
        verrouillage.Actif : verrouillee;
        1 : fermee;
      esac;

      etat=verrouillee :
      case
        !condition : verrouillee;
        deverrouillage.Actif : requete_deverrouillage;
        1 : verrouillee;
      esac;

      etat=requete_deverrouillage :
      case
        !condition : verrouillee;
        1 : fermee;
      esac;
    esac;
DEFINE
  Ouverte := etat = ouverte;
  Fermee := etat = fermee | etat = verrouillee | etat=requete_deverrouillage;
  Verrouillee := etat = verrouillee ;
--------------------------------------------------------------------------
SPEC AG ((EF Ouverte) & (EF Fermee))
SPEC AG (fermeture.Actif & !ouverture.Actif -> AX Fermee)
--------------------------------------------------------------------------

--------------------------------------------------------------------------
MODULE main
--------------------------------------------------------------------------
VAR
  porte_int : porte(porte_ext.Verrouillee);
  porte_ext : porte(porte_int.Verrouillee);

  pression_sas : { normale, basse };
  pression_int : { normale, basse };

  pressurisation : { desactivee, activee };

ASSIGN
  init(pression_sas) := normale;
  init(pression_int) := normale;

  next(pressurisation) :=
    case
```

```
      !porte_ext.Verrouillee : desactivee;
      porte_ext.Verrouillee : activee ;
    esac;

  next(pression_sas) :=
    case
      porte_ext.Fermee :
      case
        pressurisation=activee : normale;
        pressurisation=desactivee : pression_sas;
      esac;
      porte_ext.Ouverte : basse;
    esac;

  next(pression_int) :=
    case
      porte_int.Ouverte : pression_sas;
      porte_int.Fermee : pression_int;
    esac;
--------------------------------------------------------------------------
SPEC AG (porte_ext.Ouverte -> AX pression_sas=basse)
SPEC AG pression_int=normale
--------------------------------------------------------------------------
```