Méthodes Formelles

Documents autorisés.

Exercice 1

Le code SVM suivant vise à modéliser un sas de décompression simplifié. Le modèle inclut la pression extérieure, la pression du sas et la pression intérieure. Il modélise également les commandes d'ouverture des portes intérieure et extérieure, ainsi que le système de régulation de la pression du sas.

```
MODULE main
VAR
  pression_ext : { haute, normal, basse } ;
  pression_ext_futur : { haute, normal, basse } ;
  pression_sas : { haute, normal, basse } ;
  pression_int : { haute, normal, basse } ;

  requete_ouverture_ext : { 0, 1 };
  requete_ouverture_int : { 0, 1 };

  porte_ext : { ouverte, fermee };
  porte_int : { ouverte, fermee };

  regulation : { aucune, externe, interne };

ASSIGN
  init(pression_ext_futur) := { haute, normal, basse } ;
  next(pression_ext_futur) := { haute, normal, basse } ;
  init(pression_ext) := { haute, normal, basse } ;
  next(pression_ext) := pression_ext_futur ;

  init(requete_ouverture_ext) := { 0, 1 };
  next(requete_ouverture_ext) := { 0, 1 };
  init(requete_ouverture_int) := { 0, 1 };
  next(requete_ouverture_int) := { 0, 1 };

  init(pression_sas) := normal ;
  init(pression_int) := normal ;
  init(porte_ext) := fermee ;
  init(porte_int) := fermee ;
  init(regulation) := aucune ;

  next(regulation) :=
    case
      requete_ouverture_int = 1 : interne ;
      requete_ouverture_ext = 1 & !(regulation = interne) : externe ;
      1 : aucune ;
    esac ;

  next(pression_sas) :=
    case
      regulation = externe | porte_ext = ouverte : pression_ext_futur ;
      regulation = interne | porte_int = ouverte : pression_int ;
      1 : pression_sas ;
    esac;

  next(pression_int) :=
    case
      porte_int = fermee : pression_int ;
      porte_int = ouverte & porte_ext = ouverte : pression_ext_futur ;
      porte_int = ouverte : pression_sas ;
    esac;

  next(porte_ext) :=
    case
      requete_ouverture_ext = 1 &
      regulation = externe &
      requete_ouverture_int = 0 : ouverte ;
```

```
      1 : fermee ;
    esac;

next(porte_int) :=
  case
    requete_ouverture_int = 1 & regulation = interne : ouverte ;
    1 : fermee ;
  esac;
```

Question 1 : Expliquez les lignes de codes suivantes :

```
init(requete_ouverture_ext) : { 0, 1 };
next(requete_ouverture_ext) : { 0, 1 };
```

Question 2 : Quel est l'état initial du système ?

Question 3 : A quoi sert la variable `pression_ext_futur` ?

Question 4 : Que veulent dire les spécifications suivantes ?

```
SPEC AG (porte_ext = ouverte -> pression_sas = pression_ext)
SPEC AG (porte_ext = ouverte -> AX AX pression_sas = pression_ext)
```

Sont-elles satisfaites par le système ?

Question 5 : Donnez des spécifications SMV pour les propriétés suivantes. Vous indiquerez également en justifiant si le modèle satisfait ces spécifications.

a) *La pression intérieure reste toujours à l'état « normal ».*
b) *Si la requête de la porte intérieure est activée pendant au moins 3 unités de temps, alors la porte intérieure s'ouvre.*
c) *Tant qu'aucune requête d'ouverture de la porte extérieure n'est émise, la pression du sas reste à l'état « normal », et éventuellement, aucune telle requête ne sera jamais émise.*

Question 6 : La spécification suivante : « *Si la requête de la porte extérieure est activée pendant au moins 3 unités de temps, alors la porte extérieure s'ouvre* » est-elle satisfaite ? Vous justifierez votre réponse.

Exercice 2

On considère le code escjava suivant. Certaine partie du code ont été occultées (par ####).

```
public class Tab {

    //@ ensures \result >= a;
    //@ ensures \result >= b;
    public static int max(int a, int b) {
        if (a>b)
            return a;
        else
            return b;
    }

    //@ requires t1 != null;
    //@ requires res != null;
    //@ requires t1.length <= t2.length;

    //@ ensures (\forall int i; 0 <= i && i <t1.length ==>
                                    (res[i] >= #### && res[i] >= ####));
    public static void f(int[] t1, int [] t2, int[] res) {
        int i;
        for(i=0; i<t1.length; i++)
            res[i] = max(t1[i], t2[i]);
    }
}
```

Question 1 : Que fait la fonction `f` ?

Question 2 : Complétez le code manquant (replacé par #####).

Question 3 : Le code suivant est un extrait du rapport de ESC/JAVA

```
...
Tab.java:24: Warning: Possible null dereference (Null)
            res[i] = max(t1[i], t2[i]);
...
 ------------------------------------------------------------------------
Tab.java:24: Warning: Array index possibly too large (IndexTooBig)
            res[i] = max(t1[i], t2[i]);
...
```

De quoi s'agit-il ? Que proposez-vous pour remédier à cela ?

Question 4 : La spécification de la fonction f n'est pas complète, que proposez vous pour la compléter ?