Licence 3 Info **Corrigé**

5SINF200 : Développement de programmes

(A. Slissenko)

Examen

Le 05 janvier 2006, 13h30–15h30
Utilisation des notes, des livres, des docs, etc. autorisée.

1. Trouvez un test qui couvre le programme
If $x^2 + x - 2 > 0 \vee x = 0$ **Then Return** 0 ;
If $x^2 + x - 2 = 0 \wedge x < 0$ **Then Return** 1 ;
If $x^2 + x - 2 < 0$ **Then Return** 2

L'exécution des opérateurs **If-Then** est séquentielle, x est du type entier.

Essayez de trouver un test petit et simple.

Réponse. Valeurs de x qui constituent un test couvrant :

2 (le 1er atome de la 1ère garde est vrai, le 2ème atome de la 1ère garde est faux ; la 1ère garde est vraie),
0 (le 1er atome de la 1ère garde est faux, le 2ème atome de la 1ère garde est vrai ; la 1ère garde est vraie),
-2 (la 1ère garde est fausse, la 2ème garde est vraie),
-1 (les 2 premières gardes sont fausses, la 3ème garde est vraie).

2. Décrivez une ASM (en utilisant seulement des **If-Then**-opérateurs exécutés en parallèle) qui contrôle plusieurs chaînes de montage. Le contrôleur s'occupe seulement des vis des objets qui se trouvent sur les chaînes : si toutes les vis sont mises alors il met la fonction $fait$ à vrai, sinon il met la fonction $refaire$ à vrai et indique la vis manquante via la fonction $ajouter$. Les fonctions $fait$ et $refaire$ ne doivent pas être vraies en même temps (pour la même chaîne). On suppose que le système de contrôle détecte une seule vis s'il y a des vis manquantes.

Les chaînes constituent un ensemble $Chaînes$, les vis (plus précisément, les types de vis) un ensemble Vis.

Les capteurs qui fournissent l'information au contrôleur sont les suivants : le capteur *objet* à valeurs booléennes dit s'il y a un objet à contrôler ; le capteur $visManque$ à valeurs dans $(Vis \cup \{nil\})$ dit s'il y a une vis manquante ou non et indique le type de vis manquante si c'est le cas. La valeur nil signifie qu'il n'y a pas de vis manquantes.

La fonction $ajouter$ a comme valeur un élément de Vis. On suppose que la fonction *objet* devient $false$ pour une certaine durée après le traitement de chaque objet. Il est clair que toutes les fonctions mentionnées dependent d'une chaîne.

N'oubliez pas de réinitialiser les sorties après les updates si nécessaire.

N'oubliez pas de donner un vocabulaire et des conditions initiales.

Décrivez un diagramme Statechart (Etats-Transitions) pour un ensemble de 2 chaînes.

Réponse.

ASM.

Vocabulaire.

Sortes : $Chaînes$, Vis. On suppose, comme toujours, que les sortes sont disjointes (on ne parle pas de l'élément spécial nil).

La spécification ne mentionne pas la sorte d'objets. Ce fait et des autres indications disent que cette sorte est inutile. La fonction $objet$ dit qu'il y a un objet à traiter sur la chaîne qui est son argument.

Fonctions :
$objet : Chaînes \to Bool$ (entrée)
$visseManque : Chaînes \to (Visses \cup \{nil\})$ (entrée)
$fait : Chaînes \to Bool$ (sortie)
$refaire : Chaînes \to Bool$ (sortie)
$ajouter : Chaînes \to Visses$ (sortie)
$flag : Chaînes \to Bool$ (interne) [Voir ci-dessous une version sans $flag$.]

La spécification dit explicitement que toutes les fonctions dépendent d'une chaîne, donc leurs types ont la forme $Chaînes \cdots \to \ldots$. De plus la spécification donne pour chaque fonction le type de sa valeur. Ces indications définissent les types des fonctions de la spécification uniquement. Le drapeau $flag$ est introduit pour faire le gardes 'ponctuelles', i. e. vraies à des instants isolés. On peut utiliser comme drapeau les fonctions $fait$ et $refaire$ (voir ci-dessous).

Initialisation : $objet(x) = fait(x) = refaire(x) = flag(x) = false$, $visManque(x) = nil$, $ajouter(x)$ est arbitraire pour tout $x \in Chaînes$ (strictement dit, il nous faut une constante du type Vis pour donner une valeur initiale de $ajouter(x)$).

Programme. Une solution possible :

ForAll $x \in Chaînes$
If $objet(x) \wedge visManque(x) = nil \wedge \neg flag(x)$ **Then** $\big[fait(x) := true,\ flag(x) := true\big]$
If $objet(x) \wedge visManque(x) \neq nil \wedge \neg flag(x)$
Then $\big[refaire(x) := true,\ ajouter(x) := visManque(x),\ flag(x) := true\big]$
If $\neg objet(x) \wedge flag(x)$ **Then** $\big[fait(x) := false,\ refaire(x) := false,\ flag(x) := false\big]$

Commentaires. Les chaînes sont indépendantes, donc le nom de chaîne, qui est un argument de toute fonction, joue un rôle très simple : modulo cet argument la machine est la même que pour une seule chaîne.

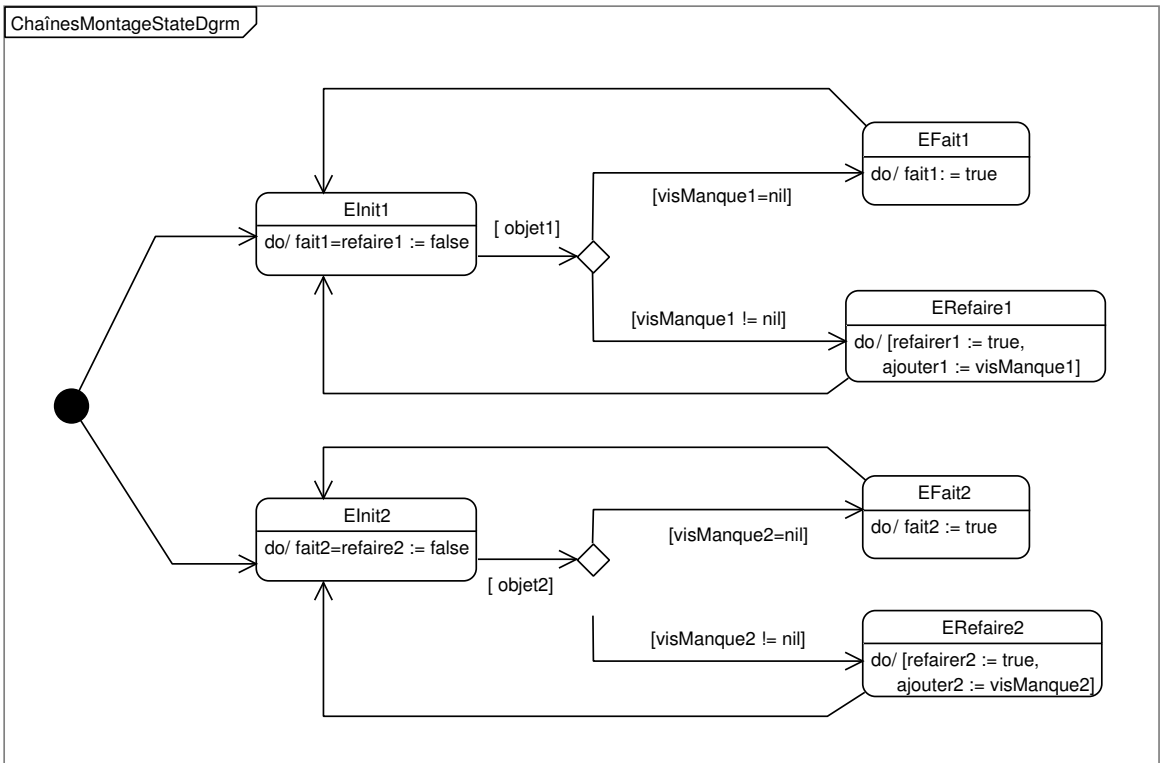
Si on ne utilise pas la fonction $flag$, ou un autre moyen équivalent, alors les signaux de contrôle $fait(x)$ et $refaire(x)$ $(= true)$ sont produits en continu pendant la durée où la fonction $objet(x)$ reste vraie. Un tel programme demande une interprétation spéciale ; à priori il n'est pas claire comment interpréter la commande $refaire(x)$ envoyée en continu. Normalement les gardes doivent être vraies à des instants de temps isolés, autrement dit elles doivent être vraies ponctuellement. Pour notre contrôleur il faut envoyer les signaux de contrôle une seule fois pendant que la fonction $objet(x)$ reste vraie. Si $\neg objet(x)$ alors il faut avoir $fait(x) = refaire(x) = false$.

Programme. Une autre solution possible :

ForAll $x \in Chaînes$
If $objet(x) \wedge visManque(x) = nil \wedge \neg fait(x) \wedge \neg refaire(x)$
Then $\big[fait(x) := true\big]$
If $objet(x) \wedge visManque(x) \neq nil \wedge \neg fait(x) \wedge \neg refaire(x)$
Then $\big[refaire(x) := true,\ ajouter(x) := visManque(x)\big]$
If $\neg objet(x) \wedge \big(fait(x) \vee refaire(x)\big)$ **Then** $\big[fait(x) := false,\ refaire(x) := false\big]$

Le deuxième programme est plus économique mais moins robuste s'il faut le modifier.

Un diagramme d'états possible est ci-dessous.



Le diagramme se compose de 2 parties pareilles : l'une pour la chaîne 1, l'autre pour la chaîne 2. Ici on n'utilise pas les drapeaux car le niveau d'abstraction de cette spécification est plus élevé que celui de l'ASM. Cependant on peut suivre l'ASM exactement, en particulier, on peut détailler le diagramme et y ajouter les drapeaux.