Licence Info **Corrigé**

LI5a : Développement de programmes

(A. Slissenko)

Examen

Le 13 May 2005, 13h–16h

1 [3 points]. Considérons la spécification de programme suivante :
If $S = \emptyset \vee \exists i\, |S| = 2i + 1$ **Then Return** $false$
If $S \neq \emptyset \wedge \exists i\, |S| = 2i$ **Then Return** $true$
Dans cette spécification S est un ensemble fini qui est une entrée, $|S|$ est le nombre d'éléments de S. Les **If-Then** sont exécutés en parallèle une seule fois. Quels requis correspondent à cette spécification ?

Trouvez un jeu de tests qui couvre ce programme.

Décrivez très brièvement le but d'une spécification de programme et le but d'une spécification de requis.

Réponse.

Requis : le programme reconnaît les ensembles non vides avec le nombre pair d'éléments.

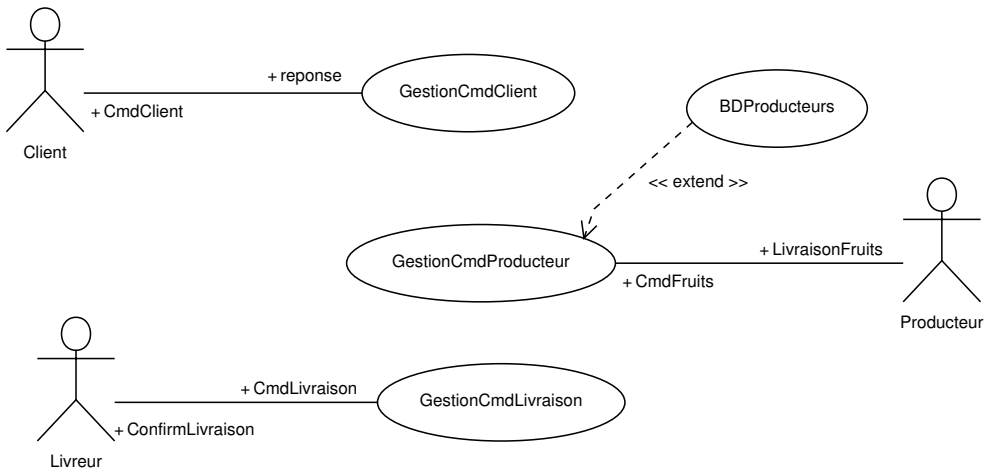
Jeu de tests : ($\emptyset$, $\{0\}$, $\{0, 1\}$).

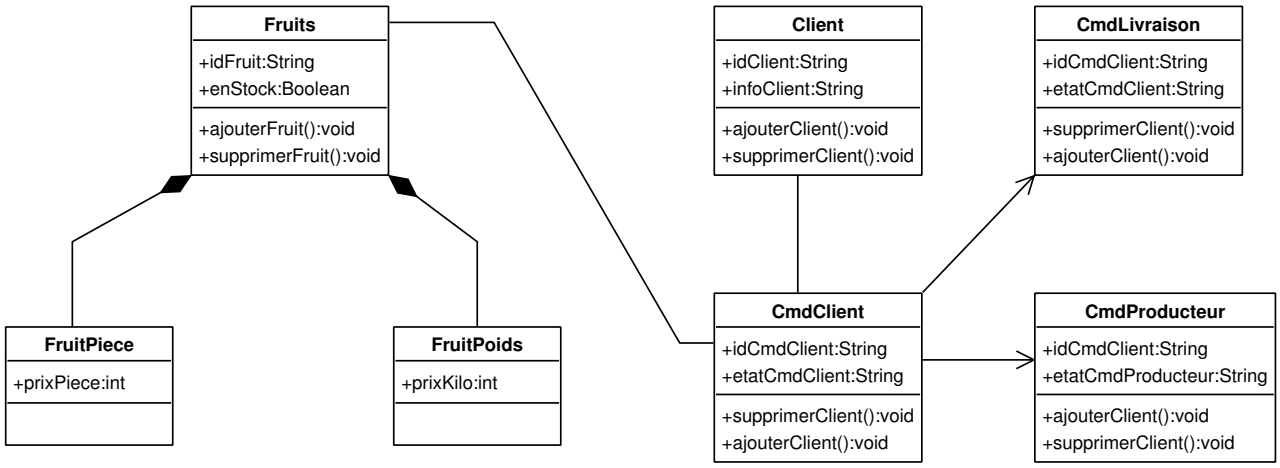
Une spécification de requis déclare les propriété à satisfaire sans référence à calcul qui pouvait atteindre ce but. Une spécification de programme décrit quel calcul à effectuer sans référence aux propriétés à satisfaire.

2 [6 points]. Décrivez un diagramme de cas d'utilisation et un diagramme de classes pour un programme qui gère les ventes en gros des fruits. Les fruits sont fournis par des producteurs. Il y a 2 types de fruits traités différemment : vendus à la pièces et vendus au poids. Les clients font leurs commandes, et en fonction des commandes des clients le programme fait des commandes de livraison et des commandes aux producteurs de fruits. Votre diagramme de classes doit avoir 6-7 classes et pour chaque classe qui n'est pas une sous-classe d'une autre — 2 attributs et 2 opérations, et 1 attribut (pas d'opérations) pour les sous-classes s'il y en a. Choisissez les classes, attributs et opérations les plus importants de votre point de vue. Pour le diagramme de cas d'utilisation choisissez 3-4 cas d'utilisation.

Attention : des diagrammes qui ne respectent pas les contraintes sur le nombre de classes, de cas d'utilisations etc. seront considérés comme incorrects.

Réponse. Les diagrammes possibles sont les suivants :





3 [8 points]. Décrivez une ASM (en utilisant seulement des **If-Then**-opérateurs exécutés en parallèle) qui contrôle un lancement de tâches sur N processeurs. Votre ASM doit gérer N processeurs dont chacun peut être soit occupé soit libre. Cet état est géré par la machine à l'aide d'une fonction $free$ (initialement tous les processeurs sont libres). La machine reçoit une tâche à lancer via une entrée $task$, la valeur de cette entrée est soit nil soit un objet à traiter (par exemple un objet peut être une instance d'un problème d'optimisation ; dans notre cas le type d'objets est abstrait) ; nil veut dire qu'il n'y a pas de tâche. Ayant reçu une tâche, la machine l'envoie à un processeur qui est libre si un tel processeur existe, sinon elle ne fait rien. Pour lancer une tâche, la machine affecte la valeur $vrai$ à la fonction $launch$ qui dépend du processeur qu'on veut utiliser et de cette tâche. Lorsqu'un processeur termine la tâche, il retourne la valeur $vrai$ pour la fonction $done$ qui peut être utilisée par l'ASM comme une entrée. (N'oubliez pas de donner un vocabulaire et des conditions initiales.)

Décrivez un diagramme Statechart (États-Transitions) pour 2 processeurs.

Réponse.

Sortes : $\mathcal{P}$ (processeurs), $Tâches$ (tâches plus nil)
Fonctions :

$task :\rightarrow Tâches$ (entrée)
$done : \mathcal{P} \rightarrow Bool$ (entrée)
$free : \mathcal{P} \rightarrow Bool$ (sortie)
$launch : \mathcal{P} \times Tâches \rightarrow Bool$ (sortie)
$flag :\rightarrow Bool$ (interne, sert pour ne pas envoyer une même tâche sur plusieurs processeurs)
$\tau : \mathcal{P} \rightarrow Tâches$ (interne, pour mémoriser la tâche courante exécutée par un processeur)

Remarque. Le texte définit clairement qu'une tâche n'est pas destinée pour un processeur spécifié. Donc le type de la fonction $task$ est comme donné ci-dessus.

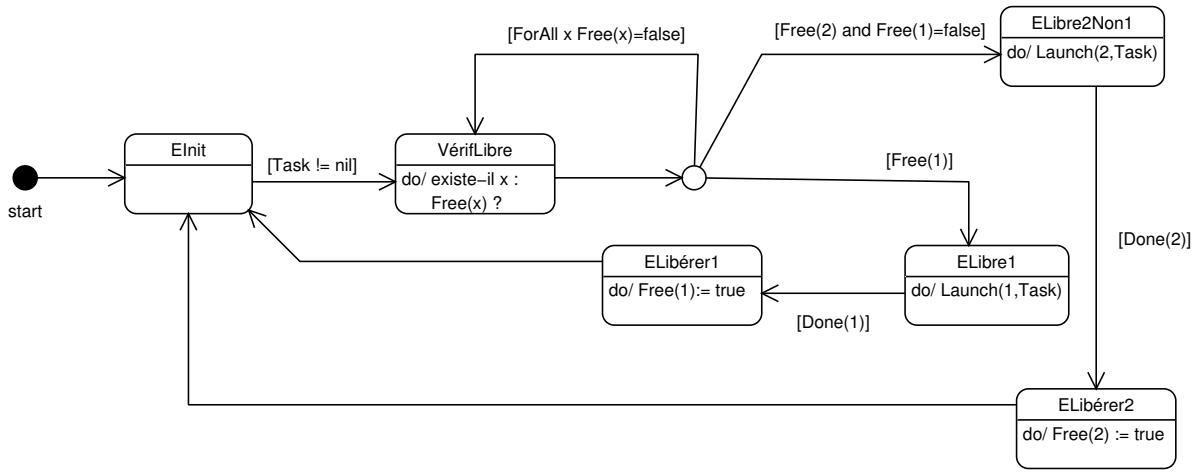
Initialisation : $task = nil$, $flag = true$, $\tau(x) = nil$, $done(x) = false$, $launch(x, T) = false$ pour tout $x \in \mathcal{P}$ et pour tout $T \in Tâches$.

Programme, où on utilise la notation $\mu =_{df} \min\{y : free(y)\}$:

ForAll $x \in \mathcal{P}$ **Do**
If $task \neq nil \wedge flag \wedge \exists\, y \in \mathcal{P}\,\big(free(y)\big)$
Then $\big[\, launch(\mu, task) := true,\ free(x) := false, \tau(\mu) := task,\ flag := false \,\big]$
If $task = nil \wedge \neg\, flag$ **Then** $flag := true$
If $\neg\, free(x) \wedge done(x)$ **Then** $\big[\, free(x) := true,\ launch(x, \tau(x)) := false \,\big]$

Remarque. La spécification de décrit pas le fonctionnement exact de la fonction $launch$. Donc on suppose que $launch(x, T)$ doit rester vrai pendant une exécution de la tâche T sur le processeur x.

Diagramme d'États pour 2 processeurs :



4 [6 points]. Décrivez une ASM (en utilisant seulement des **If-Then**-opérateurs exécutés en parallèle) qui gère le degré d'ouverture de N robinets. La fonction à gérer est Δ (le degré d'ouverture) dont les valeurs pour un robinet donné sont $\{0, 1, \ldots, \nu - 1\}$. La valeur 0 de Δ pour un robinet x signifie que le robinet x est fermé, et la valeur $(\nu - 1)$ signifie que le robinet x est complètement ouvert. La fonction d'entrée φ a 3 valeurs $\{ouvrir, fermer, nil\}$ et a pour argument un robinet. Si à un moment donné on a $\varphi(x) \neq nil$, alors la valeur suivante doit être nil. La machine doit réagir à cette entrée de la façon suivante. Pour un robinet x, si $\varphi(x)$ devient $ouvrir$ (après avoir été nil) et le degré d'ouverture de x est maximal, alors on ne fait rien, sinon on augmente le degré de 1 une fois et on ne change plus le degré tant que $\varphi(x)$ ne change pas. Pareil pour $fermer$: si le degré de fermeture est 0 alors on ne fait rien, sinon on diminue le degré d'ouverture de x de 1 une fois et on ne change plus le degré tant que $\varphi(x)$ ne change pas. Utilisez un drapeau pour changer le degré une seule fois. (N'oubliez pas de donner un vocabulaire et des conditions initiales.)

Réponse.

Sortes : $\mathcal{R}$ (robinets), $\mathcal{D} = \{0, 1, \ldots, \nu - 1\} \subset \mathbb{N}$, $\{ouvrir, fermer, nil\}$
Fonctions :
$0, 1, \nu - 1 :\rightarrow \mathcal{D}$ (constants du type $\mathcal{D}$)
$\varphi : \mathcal{R} \rightarrow \{ouvrir, fermer, nil\}$ (entrée)
$\Delta : \mathcal{R} \rightarrow \mathcal{D}$ (sortie)
$flag : \mathcal{R} \rightarrow Bool$

Initialisation : $\varphi(x) = nil$, $De(x)$ arbitraire, $flag(x) = true$ pour tout $x \in \mathcal{R}$.

ForAll $x \in \mathcal{R}$ **Do**
If $\varphi(x) = ouvrir \wedge \varphi(x) < \nu - 1 \wedge flag(x)$ **Then** $[\ \Delta(x) := \Delta(x) + 1,\ flag(x) := false\ ]$
If $\varphi(x) = fermer \wedge \varphi(x) > 0 \wedge flag(x)$ **Then** $[\ \Delta(x) := \Delta(x) - 1,\ flag(x) := false\ ]$
If $\varphi(x) = nil \wedge \neg\, flag(x)$ **Then** $flag(x) := true$