Licence Info **Corrigé**

5SINF200 : Développement de programmes

(A. Slissenko)

Examen

Le 21 décembre 2006, 13h30–15h30
Utilisation des notes, des livres, des docs, etc. autorisée.

Ce corrigé contient des réponses aux questions et des commentaires. En examen il faut donner seulement des réponses courtes et claires.

1. Trouvez un test qui couvre le programme
if $x^2+3xy-1>0 \vee x+y=0$ **then return** 0 ;
if $x^2+3xy-1=0 \wedge x<0$ **then return** 1 ;
if $x^2+3xy-1<0$ **then return** 2

L'exécution des opérateurs **if-then** est séquentielle. Les variables x et y sont du type réel. Trouvez un test petit et simple.

Réponse.

(x,y)	Formule qui est faite vraie	Ligne avec la garde vraie
$(2,0)$	$x^2+3xy-1>0$	1
$(0,0)$	$x+y=0$	1
$(-1,0)$	$x^2+3xy-1=0 \wedge x<0$	2
$(0,-1)$	$x^2+3xy-1<0$	3
$(1,0)$	aucune	aucune

Remarquons que $x+y=0$ implique $x^2+3xy-1<0$ (car si $x+y=0$, alors $x=-y$ et donc $x^2+3xy-1=y^2-3y^2-1=-2y^2-1<0$) et nous ne pouvons pas faire un test qui déclenche $x+y=0$ de la première garde et ne déclenche pas la garde du troisième opérateur. Heureusement, les opérateurs **if-then** sont séquentiels. Le dernier test dans le tableau est aussi important car l'entrée $(1,0)$ n'est pas exclue.

2a. Décrivez une ASM de base (en utilisant seulement des **if-then**-opérateurs exécutés en parallèle) qui contrôle une nutrition d'animaux exotiques décrite ci-dessous.

Les animaux se nourrissent exclusivement de boissons. Les animaux sont de types différents. On peut avoir plusieurs animaux d'un même type. Les animaux constituent une sorte *Animal*. Un élément de cette sorte est en fait le nom de l'animal, non son type. Nous n'avons pas besoin d'introduire une sorte des types d'animaux.

Les boissons sont aussi de types différents. Ces types constituent une sorte *Boisson*. Chaque animal peut consommer un seul type de boisson. Ce type est donné par une fonction externe statique (i.e. par une fonction qui ne change pas avec le temps) $anmlBssn$.

Les boissons se trouvent dans des réservoirs. Les réservoirs constituent une sorte *Réservoir*. Ils sont numérotés, donc on peut parler du réservoir avec le numéro minimal qui satisfait une condition donnée. Un réservoir donné contient toujours la boisson du type donné par la fonction externe statique $rsrvBssn$. On peut avoir plusieurs réservoirs contenant une boisson d'un type donné. À l'aide des deux 2 fonctions $anmlBssn$ et $rsrvBssn$ on peut exprimer, par exemple, que le type de boisson du réservoir r convient à l'animal x.

Pour les éléments des sortes *Animal* et *Réservoir* vous devez utiliser les variables (logiques) suivantes : x, y pour *Animal* et r, r', r'' pour *Réservoir*. Vous n'avez pas besoin de variables pour la sorte

Boisson. Cette dernière sorte apparaît dans le type des fonctions et, implicitement, dans les formules, par exemple, dans une formule qui dit qu'un réservoir r contient une boisson pour un animal x.

L'entrée *nourrir* a comme argument un animal. La fonction de sortie *ouvrir* a comme argument un réservoir et un animal. Ces deux fonctions sont à valeurs booléennes. La condition $ouvrir(r, x) = true$ signifie que le réservoir r est utilisé pour nourrir l'animal x. À un moment donné un réservoir ne peut être utilisé que pour un seul animal.

La condition $nourrir(x) = true$ implique qu'il faut trouver un réservoir avec la boisson convenant à l'animal x et qui n'est pas utilisé pour nourrir un autre animal à ce moment-là. Deux situations sont possibles.

- Un tel réservoir existe. Alors il faut donner la valeur *true* à *ouvrir* pour ce réservoir et cet animal. C'est le début d'une séance de nutrition pour cet animal. La fin de séance de nutrition d'un animal x est commandée par la condition $nourrir(x) = false$. Si cette condition devient vraie, alors le contrôleur doit immédiatement faire $ouvrir(r, x)$ égal à $false$, où r est le réservoir qui nourrit x à ce moment-là.
- Un tel réservoir n'existe pas. Cela se produit si au moment où $nourrir(x)$ devient vrai, tous les réservoirs avec la boisson du type $anmlBssn(x)$ sont utilisés pour d'autres animaux. Dans ce cas le contrôleur doit déclencher une alarme qui est représentée par la fonction de sortie *alarme* dont l'argument est un animal et dont la valeur est booléenne (*true* signifie que l'alarme est déclenchée). Le signal $alarme(x)$ doit être arrêté par le contrôleur si au moins une de 2 conditions suivantes est vraie : $nourrir(x)$ devient $false$ ou un réservoir convenant à x devient disponible.

On suppose qu'initialement les signaux *nourrir*, *alarme*, *ouvrir* sont faux pour tous ses arguments.

Pour rendre votre programme lisible introduisez une notation pour une relation qui caractérise les paires (r, x) telles que " un réservoir r contient une boisson convenable pour un animal x et n'est pas utilisé pour un autre animal ". Assurez-vous que les gardes de votre ASM sont vraies à des instants isolés. Pour ce faire utilisez les fonctions de votre vocabulaire comme des drapeaux. Vous pouvez utiliser les quantificateurs sur vos sortes dans les gardes pour dire, par exemple, qu'il n'y a pas de réservoir qui nourrit un animal donné. Pour choisir un réservoir parmi plusieurs réservoirs disponibles utilisez des fonctions mathématiques, par exemple, max, min.

N'oubliez pas qu'une ASM consiste en un vocabulaire, une initialisation et un programme.

Réponse.

Vocabulaire.

Sortes : *Boisson*, *Animal* (variables : x, y), *Réservoir* (variables : r, r', r'')

Fonctions externes :

- $nourrir$: $Animal \to Bool$ (dynamique, $nourrir(x)$ signale qu'il faut nourrir l'animal x)
- $rsrvBssn$: $Réservoir \to Boisson$ (statique, $rsrvBssn(r)$ retourne le type de la boisson qui se trouve dans le réservoir r)
- $anmlBssn$: $Animal \to Boisson$ (statique, $anmlBssn(x)$ retourne le type de la boisson convenable pour l'animal x)

Fonctions internes :

- $ouvrir$: $Réservoir \times Animal \to Bool$ (fonction de sortie, $ouvrir(r, x) = true$ permet que l'animal x puisse boire du réservoir r)
- $alarm$: $Animal \to Bool$ (fonction de sortie, $alarme(x) = true$ déclenche une alarme pour x, $alarme(x) = false$ arrête l'alarme déclenchée)

Initialisation : $nourrir(x) = alarme(x) = ouvrir(r, x) = false$

Description du programme ci-dessous.

Notation : $\Phi(r, x) =_{df} \big(rsrvBssn(r) = anmlBssn(x) \wedge \forall y \neg\, ouvrir(r, y)\big)$

La formule $\Phi(r, x)$ dit que le réservoir r contient une boisson convenable pour l'animal x et ce réservoir n'est utilisé pour aucun animal, l'animal x inclus.

Complément de la spécification. La spécification ne dit pas si pour plusieurs animaux x_i qui consomme une boisson d'un même type le signal $nourrir(x_i)$ peut être déclenché en même instant (on parle du premier instant quand le signal devient vrai). Cette possibilité fortement complique la programmation du contrôleur. Nous profitons du fait que la spécification ne dit rien sur ce sujet et imposons la contrainte : pour tout animaux x et y, $x \neq y$, tels que $anmlBssn(x) = anmlBssn(y)$ le signal *nourrir* se déclenche à des moments différents (bien sûr il peut rester vrai pour plusieurs animaux simultanément après le déclenchement).

Commentaires sur le programme.

Le programme est paramétré par l'animal x et par le réservoir r. Le paramètre r intervient seulement dans la ligne 4. Les animaux sont traités indépendamment, donc on peut penser seulement à l'animal x quand on lit le programme.

La ligne 1 traite le cas où le signal *nourrir* un animal x est déclenché, il y a un réservoir convenable pour x qui est libre et l'animal x n'est nourri d'aucun réservoir. Dans le cas le contrôleur choisit le premier réservoir (dans l'ordre de la numérotation de réservoirs) qui est convenable et libre. Remarquons que après l'update de cette ligne sa garde devient fausse. Il peut s'avérer que $nourrir(x)$ est vrai depuis quelque temps mais il n'y avait pas de réservoir disponible.

La ligne 2 concerne le cas où le signal *nourrir* l'animal x est déclenché, mais il n'y a pas de réservoir disponible et le signal d'alarme n'est pas encore déclenché. Dans ce cas le contrôleur déclenche le signal d'alarme pour x, et ce signal sert comme drapeau pour cet opérateur.

La ligne 3 arrête le signal d'alarme. C'est fait dans les conditions suivantes. Une alarme pour x est en marche. Mais soit le signal nourrir s'arrête soit un réservoir convenable devient disponible. Encore, la même *alarme* sert comme drapeau pour cet opérateur.

La ligne 4 arrête la nutrition de x. Le signal *ouvrir* sert aussi comme drapeau. Dans la garde de cet opérateur on voit le paramètre r. Formellement dit, cet opérateur ferme tous les réservoirs ouverts pour x. Mais en fait, il n'y a qu'un.

Remarquons que les variables r' et r'' dans les formules du programmes sont liées. Par conséquent on ne peut pas faire référence sur r' qui se trouve dans le scope d'un quantificateur hors du scope de ce quantificateur. Toute variable liée peut être renommée, par exemple, la variable r'' de min peut être remplacée par r' ou par r. Pour faciliter la lecture du programme la variable r joue seulement le rôle de paramètre du programme, la variable r' est utilisée dans les scopes de quantificateurs de gardes du programme et r'' est utilisée dans le scope de min.

forall $x \in Animal$, $r \in Réservoir$ **do**
1 : **if** $nourrir(x) \wedge \exists r'\, \Phi(r', x) \wedge \forall r' \neg ouvrir(r', y)$
then $ouvrir(\min\{r'' : \Phi(r'', x)\}, x) := true$
2 : **if** $nourrir(x) \wedge \neg \exists r'\, \Phi(r', x) \wedge \forall r' \neg ouvrir(r', y) \wedge \neg alarme(x)$
then $alarme(x) := true$
3 : **if** $alarme(x) \wedge (\neg nourrir(x) \vee \exists r'\, ouvrir(r', x))$
then $alarme(x) := false$
4 : **if** $\neg nourrir(x) \wedge ouvrir(r, x)$
then $ouvrir(r, x) := false$

2b. Décrivez des requis pour la spécification ci-dessus. Quelle propriété décrit la sûreté et laquelle la vivacité ?

Réponse. Voilà quelques propriétés (la liste n'est pas complète, i. e. elle ne représente pas toutes les propriétés à assurer) :

Sûreté : Deux animaux ne boivent pas d'un même réservoir en même temps.

Pour préciser cette et d'autres phrases (ce qui n'est pas demandé par ce sujet) on utilise t et τ comme

variable pour le temps. Rappelons que f° est une version temporisée de la fonction f, c'est-à-dire si f est du type $\mathcal{X} \to \mathcal{Z}$, alors f° est du type $Temps \times \mathcal{X} \to \mathcal{Z}$

$\forall\, t\, x\, y\, r\, \big((x \neq y \wedge ouvrir^\circ(t,r,x)) \to \neg\, ouvrir^\circ(t,r,y)\big)$

Sûreté : Un animal ne boit jamais une boisson non convenable.

$\forall\, t\, x\, r\, \big(anmlBssn(x) \neq rsvrBssn(r) \to \neg\, ouvrir^\circ(t,r,x)\big)$

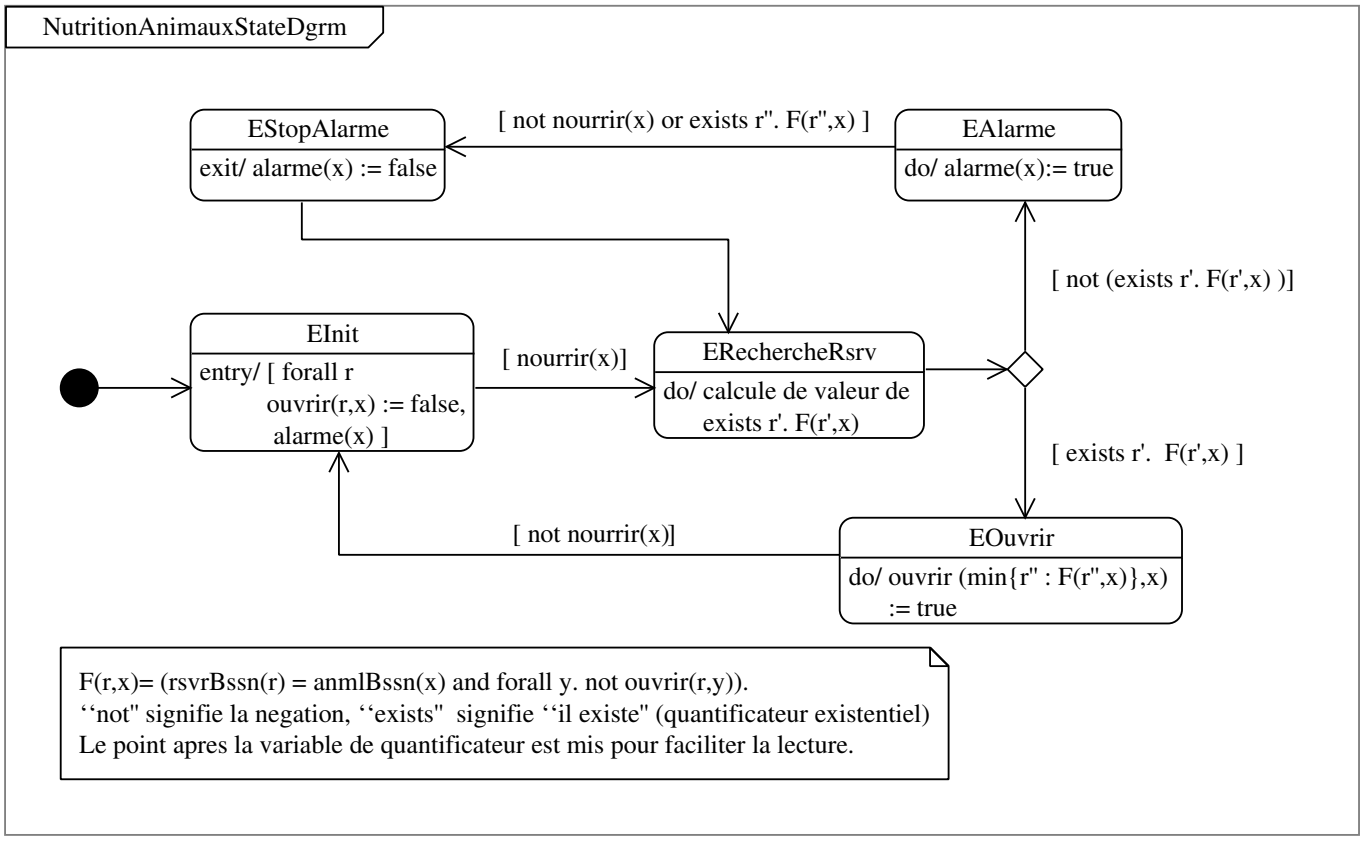
Vivacité : Si le signal nourrir un animal est activée et s'il y a un réservoir disponible avec la boisson convenable, l'animal obtient sa boisson.

$\forall\, t\, x\, \big((nourrir^\circ(t,x) \wedge \exists\, r'\, \Phi^\circ(t,r',x)) \to \exists\, r \exists\, \varepsilon > 0\, \forall\, \tau \in (t, t+\varepsilon)\, ouvrir^\circ(\tau,r,x)\big)$,
où ε est une variable pour les nombres réels.

On peut ajouter des propriétés qui spécifient le déclenchement d'alarme.

2c. Dessinez un diagramme de machine à états (statechart) pour le contrôleur qui gère l'animal x (le paramètre de votre diagramme). Votre diagramme doit avoir 5–7 états (sans compter le pseudo-état *start*).

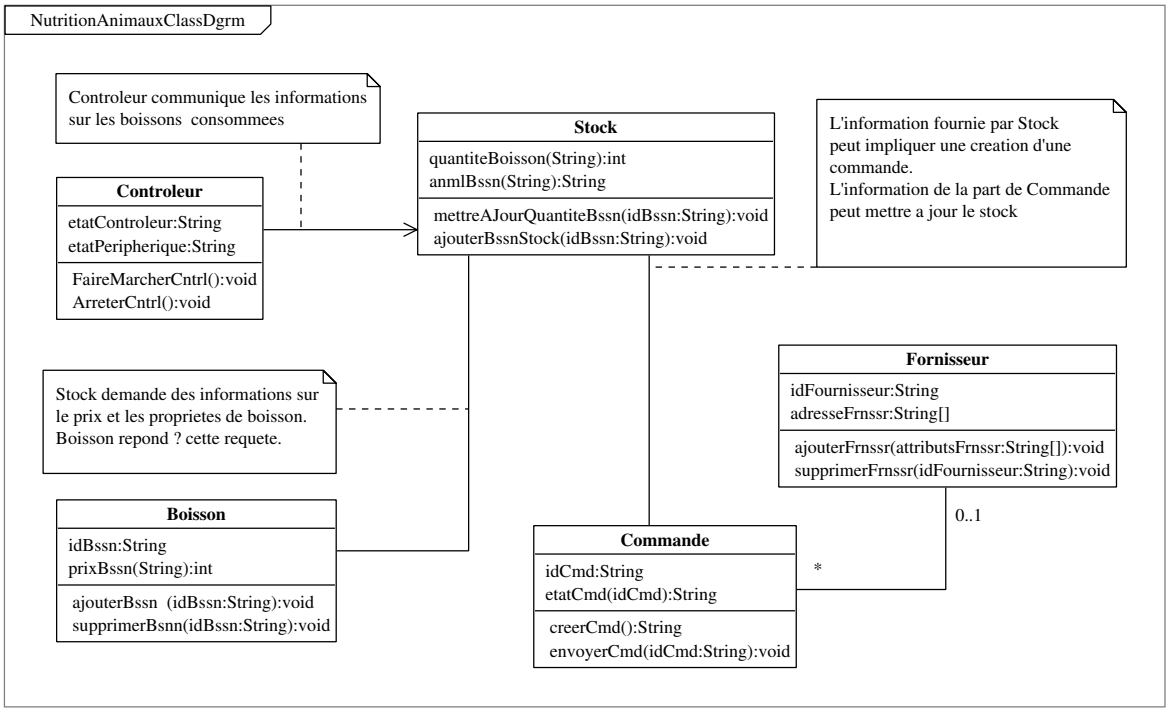
Réponse. Un diagramme possible :



3. Supposons qu'il faille concevoir un logiciel qui interagit avec le contrôleur de l'exercice 2 et de plus gère les commandes de boissons pour les animaux.

Un concepteur de logiciel a commencé à faire des diagrammes UML. Il a distingué les classes suivantes : *Contrôleur*, *Stock*, *Boisson*, *Commande*, *Fournisseur*. Donnez pour chaque classe 2 attributs et 2 opérations. Dessinez des associations entre ces classes. Pour les associations qui impliquent *Stock*, *Boisson*, *Commande* faites un commentaire court sur les informations échangées entre ces classes.

Réponse. Un diagramme de classes possible est ci-dessous. Les commentaires donnent une réponse à la question sur l'échange d'informations entre les classes. Le logiciel que j'utilise ne compile pas correctement en postscript les lettres avec accents, donc il n'y a pas d'accents.



Dessinez un diagramme de cas d'utilisation pour les acteurs : administrateur de nutrition (qui contrôle le signal *nourrir*), fournisseur de boissons, et pour les cas d'utilisation : contrôleur, gestionnaire de commandes de boissons, base de données de fournisseur, gestionnaire de stock de boissons.

Un diagramme de cas d'utilisation possible :