Université de Sherbrooke
Département d'informatique

IGL501-IGL710 : Méthodes formelles en génie logiciel

Examen périodique

Professeur : Marc Frappier

Lundi 26 otobre 2015, 13h30 à 16h20
Salles : D3-2038

Notes importantes :

- Toute documentation permise, sauf un appareil électronique.
- Répondez dans le cahier d'examen fourni.
- La correction est, entre autres, basée sur le fait que chacune de vos réponses soit :
  - claire, c'est-à-dire lisible et compréhensible pour le lecteur;
  - précise, c'est-à-dire exacte et sans erreur;
  - concise, c'est-à-dire qu'il n'y ait pas d'élément superflu;
  - complète, c'est-à-dire que tous les éléments requis sont présents.

Pondération :

Question	Point
1	12
2	12
3	26
4	50
total	100

1. (**12 pt**) Traduisez les énoncés suivants avec le langage de Tarski.

   (a) Si les cubes sont tous sur la même ligne, alors les tétraèdres sont tous sur une même colonne.

   (b) Il existe un cube grand ssi tous les tétraèdres sont petits.

   (c) Le cube a est le plus petit des cubes. Vous ne pouvez pas utiliser un quantificateur $\exists$ pour cette question. Utilisez le prédicat Smaller(x, y) qui retourne vrai ssi x est plus petit que y.

   (d) Le cube a est le plus petit des cubes. Vous ne pouvez pas utiliser un quantificateur $\forall$ pour cette question. Utilisez le prédicat Smaller(x, y).

   (e) L'existence d'un tétraèdre petit est une condition suffisante pour que tous les cubes soient gros.

   (f) L'existence d'un tétraèdre petit est une condition nécessaire pour que tous les cubes soient gros.

Solution:

1. (∀x∀y((Cube(x) ∧ Cube(y)) → SameRow(x, y))) → (∀x∀y((Tet(x) ∧ Tet(y)) → SameCol(x, y)))
2. (∃x (Cube(x) ∧ Large(x))) ↔ (∀x (Tet(x) → Small(x)))
3. Cube(a) ∧ ∀x ((Cube(x) ∧ x ≠ a) → Smaller(a, x))
4. Cube(a) ∧ ¬∃x ¬((Cube(x) ∧ x ≠ a) → Smaller(a, x))
5. (∃x (Tet(x) ∧ Small(x))) → (∀x (Cube(x) → Large(x)))
6. (∀x (Cube(x) → Large(x))) → (∃x (Tet(x) ∧ Small(x)))

2. (**12 pt**) Pour chaque opération suivante, indiquez si elle préserve l'invariant. Si elle le préserve, justifiez votre réponse (un texte suffit; vous pouvez aussi donnez une preuve si vous préférez). Si elle ne le préserve pas, donnez un contre-exemple et trouvez la précondition la plus faible (la moins restrictive) qui permet de préserver l'invariant

   (a) Opération : `A(x) = PRE x : NAT THEN y := y+x END`
   Invariant : `y : 0..k`
   Solution: Non. Contre-exemple: $x = 1$ et $y = k$. Précondition: $x + y \leq k$

   (b) Opération : `B(x) = PRE x : NAT THEN CHOICE y:=y+x OR y:=y+x+1 END END`
   Invariant : `y : 0..k`
   Solution: Non. Contre-exemple: $x = 1$ et $y = k - 1$. Précondition: $x + 1 + y \leq k$

   (c) Opération : `C(x,y) = PRE x : NAT & y : NAT THEN f(x) := y END`
   Invariant : `f : NAT >-> NAT`
   Solution: Non. Contre-exemple: $x = 1$ et $y = 1$ et $f(2) = 1$.
   Précondition: $f(x) = y$

   (d) Opération : `D = ANY x,y WHERE x : NAT & y : NAT THEN f(x) := y END`
   Invariant : `f : NAT +-> NAT`
   Solution: Oui. Seule l'image de x est modifiée. Si $x \notin \mathrm{dom}(f)$, x a une seule image, y. Si $x \in \mathrm{dom}(f)$, alors on remplace l'image de x par y et x a toujours une seule image.

3. (**26 pt**) Traduisez le schéma UML de la figure 1 en invariant en B, en représentant toutes les contraintes. La relation r1 est une classe associative. La relation r3 est une relation ternaire. De plus, nous exigeons que chaque instance de la classe A et chaque instance de la classe D participe au moins une fois dans l'association (i.e., pour chaque instance de A, il existe au moins un triplet de r3 qui y fait référence; idem pour la classe D). L'attribut k1 est unique et non nul dans la classe A, c'est-à-dire qu'il a toujours une valeur et qu'il n'y a pas deux instances de A qui ont la même valeur pour k1. Voici les déclarations des SETS que vous devez utiliser pour le typage des variables:

```
SETS A;B;C;D
```

Solution:

```
a <:  A &
b <:  B &
d <:  D &
k1 :  a >-> NAT &
r1 :  a → b &
r2 :  (r1) ↣↠ d &
r3 <:  (a*b)*d &
dom(dom(r3)) = a &
ran(r3) = d
```

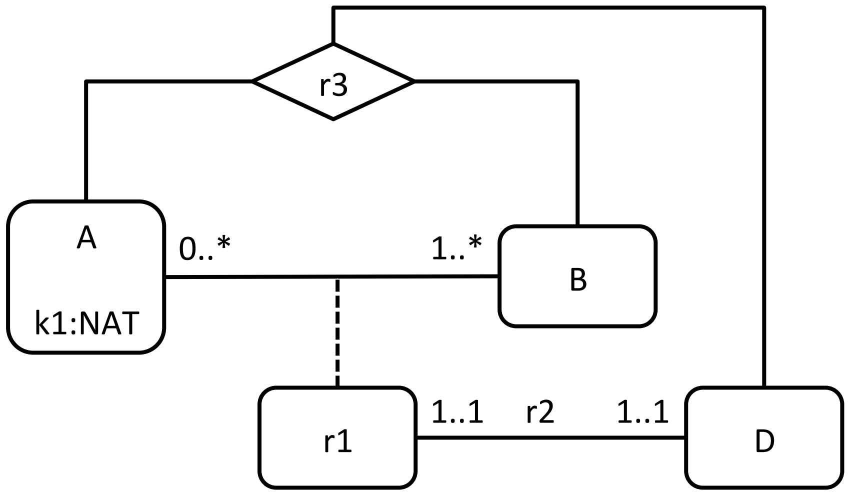

Figure 1: Diagramme UML de la question 2

4. (**50 pt**) Considérez la description suivante d'un système de gestion de tâches. La figure 2 contient le diagramme de classe UML de ce système. Nous identifions chaque élément des exigences, afin que vous puissiez les identifier dans vos invariants.

   (a) Lors de la création d'une tâche, ses dates de début et de fin ne sont pas connues.

   (b) Lorsqu'on démarre une tâche, on lui affecte une date de début. Lorsqu'elle est terminée, on lui affecte une date de fin, qui est supérieure à la date de début, bien sûr.

   (c) Une tâche peut être affectée à plusieurs personnes en même temps, et une personne peut travailler sur au plus k tâches en même temps; cette constante k est globale pour le système.

   (d) Le système conserve les compétences de chaque personne et les compétences requises pour une tâche.

   (e) Pour réaliser une tâche, une personne doit disposer des compétences requises par la tâche.

   (f) Une tâche requiert aussi des ressources pour être réalisée.

   (g) On suppose que les ressources ne peuvent être partagées: si une tâche t a besoin d'une ressource r, alors aucune autre tâche ne peut utiliser cette ressource r entre la date de début et de fin de t.

   (h) La relation affectée ne contient que les tâches sur lesquelles une personne travaille présentement; lorsque qu'une tâche est terminée, elle est désaffectée des personnes qui y travaillaient.

   (i) La relation utiliséePar ne contient que les ressources utilisées par une tâche. Lorsqu'une tâche est terminée, ses ressources sont libérées afin qu'elles puissent être utilisées par une autre tâche.

   (j) Une ressource ne peut être utilisée que si elle est requise.

   (k) On ne peut affecter une personne ou une ressource à une tâche terminée.

Produisez une spécification en B pour ce système, en ne donnant que les informations suivantes

(a) Clause SETS.

(b) Clause CONSTANTS.

(c) Clause PROPERTIES.

(d) Clause VARIABLES : voici les variables que vous devez utiliser:
personne, tache, ressource, competence, affectee, dispose, necessite, utiliseePar, requiert, dateDebut, dateFin.

(e) Clause INVARIANT : donnez chaque invariant et donnez un commentaire pour l'expliquer au besoin (i.e., pour ceux qui ne sont pas que du typage).

(f) Clause INITIALISATION : pas nécessaire de donner cette clause (toutes les variables sont initialisées à vide).

(g) Clause OPERATIONS spécifiez seulement les opérations suivantes:

  i. `AffecterPersonne(p,t)` : cette opération affecte la personne `p` à la tâche `t`. `t` doit être démarrée.

  ii. `AffecterRessource(r,t)` : cette opération affecte la ressource `r` à la tâche `t`. `t` doit être démarrée.

  iii. `TerminerTache(t,d)` : cette opération termine la tâche `t` à la date `d`. Toutes les personnes et les ressources de la tâche sont désaffectées.

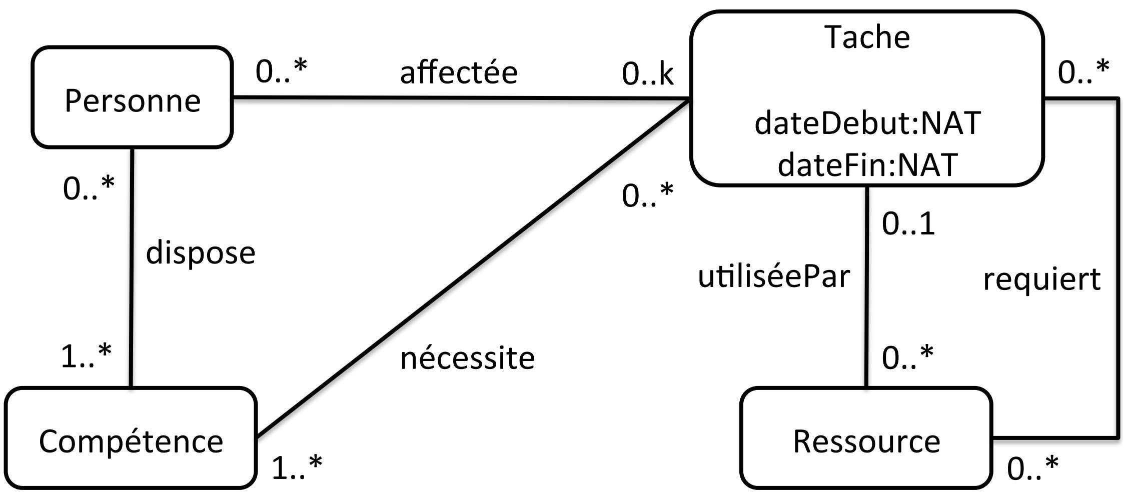

Figure 2: Diagramme UML de la question 4