Génie Logiciel Avancé

Année 2022-2023

Prof. Burkhart Wolff
wolff@lri.fr

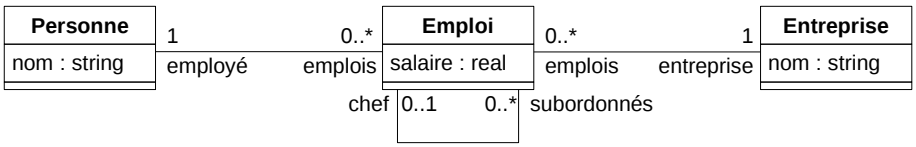
Adrien Durier
adrien.durier@universite-paris-saclay.fr
Jeremy Marrez
jeremy.marrez@universite-paris-saclay.fr

TD 2 - Spécification des opérations

Semaine du 26 septembre 2017

Exercice 1

On considère le diagramme de classes suivant, qui modélise les différents emplois occupés par des personnes dans des entreprises. Un emploi est caractérisé par son salaire et associe de manière unique une personne et une entreprise. Une personne ne peut pas occuper plusieurs emplois dans la même entreprise mais peut par contre travailler dans plusieurs entreprises. Il peut exister une relation de hiérarchie entre les employés d'une même entreprise, modélisée par l'association réflexive sur la classe Emploi. Une personne reçoit toujours un salaire strictement inférieur à son supérieur. Pour simplifier, les personnes et les entreprises sont caractérisées de manière unique par leur nom.



On associe à ce diagramme de classes les invariants suivants :

(a) $\forall p1, p2 \in \mathsf{Personne}.\ p1 \neq p2 \Rightarrow p1.\mathsf{nom} \neq p2.\mathsf{nom}$

(b) $\forall p \in \mathsf{Personne}.\ |p.\mathsf{emplois}| = |p.\mathsf{emplois.entreprise}|$

(c) $\forall e1, e2 \in \mathsf{Entreprise}.\ e1 \neq e2 \Rightarrow e1.\mathsf{nom} \neq e2.\mathsf{nom}$

(d) $\forall e \in \mathsf{Emploi}.\ e.\mathsf{salaire} > 0$

(e) $\forall e \in \mathsf{Emploi}.\ e.\mathsf{chef} \neq NULL \Rightarrow e.\mathsf{salaire} < e.\mathsf{chef.salaire}$

(f) $\forall e \in \mathsf{Emploi}.\ e.\mathsf{chef} \neq NULL \Rightarrow e.\mathsf{entreprise} = e.\mathsf{chef.entreprise}$

1. Donner une spécification formelle en termes de pré et post-conditions pour les opérations suivantes.
   (a) Pour une instance $emp \in \mathsf{Emploi}$, $emp.\mathsf{augmenter}(montant : int) : int$ permet d'augmenter le salaire de cet emploi du montant passé en argument, et renvoie le nouveau salaire. Il faut que les invariants liés à la hiérarchie dans l'entreprise soient toujours vérifiés après l'augmentation.
   (b) Pour une instance $emp \in \mathsf{Emploi}$, $emp.\mathsf{diriger}(emp2 : \mathsf{Emploi})$ permet à emp de devenir le supérieur de l'emploi passé en argument, si cette personne n'a pas déjà un supérieur et que la contrainte sur les salaires est respectée.

(c) On veut modifier l'opération diriger pour qu'elle puisse être appelée sans condition. Elle renverra alors un booléen : *true* si les conditions étaient remplies et *emp* est devenu le supérieur de $emp2$; *false* dans le cas contraire. Dans ce dernier cas, l'état n'est pas modifié.

(d) Pour une instance $ent \in$ Entreprise, ent.embaucher($nom : string, salaire : int$) : Emploi permet d'embaucher une personne avec le salaire donné, strictement positif. Il faut que cette personne existe et n'appartienne pas déjà aux employés de l'entreprise. L'emploi créé lors de l'opération est renvoyé.

(e) Pour une instance $ent \in$ Entreprise, ent.licencier($nom : string$) permet de licencier une personne. Il faut que cette personne soit un employé de l'entreprise. Tous ces liens avec l'entreprise et ses collègues doivent être supprimés.

(f) Pour une instance $pers \in$ Personne, $pers$.collegues(ent : Entreprise) : Set(Personne) construit l'ensemble des personnes travaillant dans la même entreprise que la personne. Il faut que cette personne ait un emploi dans l'entreprise passée en argument. Une personne ne fait pas partie de l'ensemble de ses collègues.

(g) Pour une instance $emp \in$ Emploi, emp.superieurs() : Set(Emploi) construit l'ensemble des emplois placés hiérarchiquement au-dessus de l'emploi concerné.

2. Construire un scénario symbolique sous la forme d'un diagramme de séquence qui illustre qu'un employé ne peut devenir le supérieur d'un autre employé que si la contrainte sur les salaires est respectée. On montrera l'embauche de deux personnes dans la même entreprise, puis la nécessité d'augmenter un employé pour qu'il puisse devenir le supérieur de l'autre. On utilisera la deuxième version de l'opération diriger.
   Exprimer formellement les contraintes qui caractérisent ce scénario.
3. Construire un scénario symbolique sous la forme d'un diagramme de séquence qui illustre que le licenciement supprime tous les liens de hiérarchie entre l'employé et ses collègues. On utilisera trois personnes reliées par une hiérarchie bien choisie, on licenciera une de ces personnes, puis on montrera qu'une nouvelle hiérarchie est possible.
   Exprimer formellement les contraintes qui caractérisent ce scénario.

Solutions

1. (a) $emp.\textsf{augmenter}(montant : int) : int$
   pre : $montant > 0$
   pre : $emp.\textsf{chef} \neq NULL \Rightarrow emp.\textsf{salaire} + montant < emp.\textsf{chef.salaire}$
   post : $emp.\textsf{salaire} = old(emp.\textsf{salaire}) + montant$
   post : $result = emp.\textsf{salaire}$
   post : $modifiesOnly(emp)$

   (b) *Première version de* **diriger** *avec pré-condition*
   $emp.\textsf{diriger}(emp2 : \textsf{Emploi})$
   pre : $emp2.\textsf{entreprise} = emp.\textsf{entreprise} \wedge emp2.\textsf{chef} = NULL \wedge emp2.\textsf{salaire} < emp.\textsf{salaire}$
   post : $emp2.\textsf{chef} = emp \wedge emp2 \in emp.\textsf{subordonnes}$
   post : $modifiesOnly(\{emp, emp2\})$

   (c) *Deuxième version de* **diriger** *sans pré-condition*
   $emp.\textsf{diriger}(emp2 : \textsf{Emploi}) : boolean$
   pre : $true$
   post : $(old(emp2.\textsf{entreprise}) = old(emp.\textsf{entreprise}) \wedge old(emp2.\textsf{chef}) = NULL \wedge old(emp2.\textsf{salaire}) < old(emp.\textsf{salaire})) \longrightarrow emp2.\textsf{chef} = emp \wedge result = true \wedge modifiesOnly(\{emp, emp2\})$
   post : $(old(emp2.\textsf{entreprise}) \neq old(emp.\textsf{entreprise}) \vee old(emp2.\textsf{chef}) \neq NULL \wedge old(emp2.\textsf{salaire}) \geq old(emp.\textsf{salaire})) \longrightarrow result = false \wedge modifiesOnly(\emptyset)$

   (d) $ent.\textsf{embaucher}(nom : string, salaire : int) : \textsf{Emploi}$
   pre : $\exists p \in \textsf{Personne}.\ p.\textsf{nom} = nom \wedge p \notin ent.\textsf{emplois.employe} \wedge salaire > 0$
   post : $isNew(result) \wedge result.\textsf{salaire} = salaire \wedge (result.\textsf{employe.nom} = nom) \wedge (result \in result.\textsf{employe.nom.emplois}) \wedge (result.\textsf{entreprise} = ent) \wedge (result \in ent.\textsf{emplois}) \wedge (result.\textsf{chef} = NULL) \wedge (result.\textsf{subordonnes} = NULL)$
   post : $modifiesOnly(\{result.\textsf{employe}\} \cup \{result\} \cup \{ent\})$

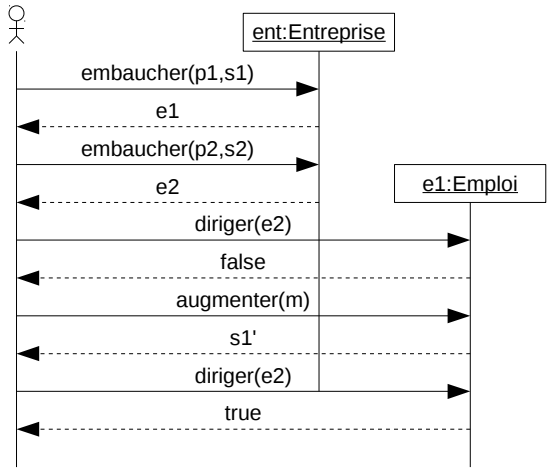
   (e) $ent.\textsf{licencier}(nom : string)$
   pre : $\exists p \in \textsf{Personne}.\ p.\textsf{nom} = nom, p \in ent.\textsf{emplois.employe}$
   post : $\exists p \in \textsf{Personne}.\ p.\textsf{nom} = nom, p \notin ent.\textsf{emplois.employe}$
   post : $modifiesOnly(\{e \in old(ent.\textsf{emplois}) \mid old(e.\textsf{employe.nom}) = nom\}.\textsf{chef} \cup \{e \in old(ent.\textsf{emplois}) \mid old(e.\textsf{employe.nom}) = nom\}.\textsf{subordonnes} \cup \{ent\} \cup \{e \in old(ent.\textsf{emplois}) | old(e.\textsf{employe.nom}) = nom\} \cup \{p \in \textsf{Personne} | p.\textsf{nom} = nom\})$

   (f) $pers.\textsf{collegues}(ent : \textsf{Entreprise}) : \textsf{Set}(\textsf{Personne})$
   pre : $ent \in pers.\textsf{emplois.entreprise}$
   post : $result = \{p \in \textsf{Personne} \mid ent \in p.\textsf{emplois.entreprise} \wedge p \neq pers\}$
   post : $modifiesOnly(\emptyset)$

   (g) $emp.\textsf{superieurs}() : \textsf{Set}(\textsf{Emploi})$
   pre : $true$

$\textbf{post} : emp.\mathsf{chef} = NULL \Rightarrow result = \emptyset$
$\textbf{post} : emp.\mathsf{chef} \neq NULL \Rightarrow result = emp.\mathsf{chef.superieurs} \cup emp.\mathsf{chef}$
$\textbf{post} : modifiesOnly(\emptyset)$

2.

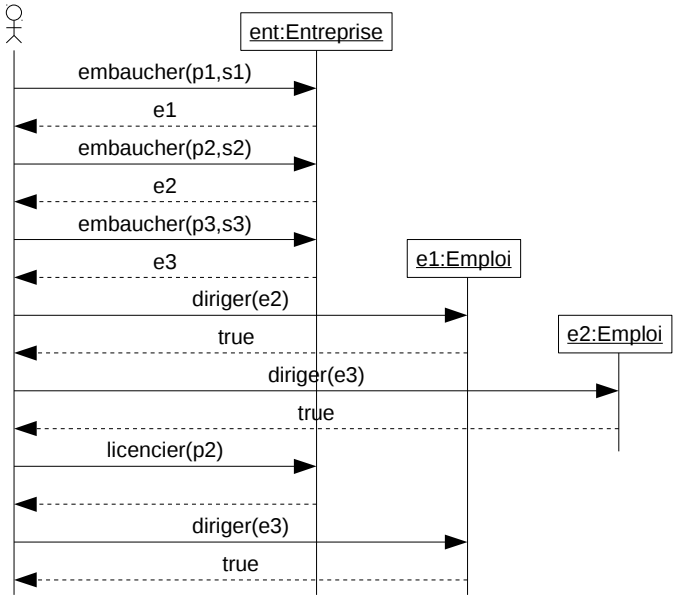


Contrainte exprimée dans l'état final :

$$s1 > 0 \wedge p2 \neq p1 \wedge s2 > 0 \wedge s1 \leq s2 \wedge m > 0 \wedge s1 + m = s1' \wedge s1' > s2$$

Remarque : ça vaut le coup de faire les diagrammes d'objets pour chaque étape, pour montrer comment l'état du système évolue au fur et à mesure.

3.



Contrainte exprimée dans l'état final :

$$s1 > 0 \wedge p2 \neq p1 \wedge s2 > 0 \wedge p3 \neq p2 \wedge p3 \neq p1 \wedge s3 > 0 \wedge s1 > s2 > s3$$