Université de Sherbrooke, Département d'informatique

IGL501: Méthodes formelles en génie logiciel

Examen périodique, lundi 15 octobre de 13h30 à 16h20, local D7-2020

Toute documentation permise

1. (16 pts) Traduisez les énoncés suivants en formules selon le langage Tarski. Indentez vos formules pour augmenter la lisibilité.

   (a) Tous les cubes sont plus gros que tous les tétraèdres.
   Solution:
   $$\forall x \forall y\ ((cube(x) \wedge tet(y)) \Rightarrow larger(x, y))$$

   (b) L'objet le plus gros est un cube (cet objet doit exister).
   Solution:
   $$\exists x\ (cube(x) \wedge \forall y\ (x \neq y \Rightarrow larger(x, y)))$$

   (c) Les objets du monde sont triés de gauche à droite et de l'avant vers l'arrière, c'est-à-dire que si un objet est situé devant un autre, il doit être inférieur ou égal en taille; de même, si un objet est situé à gauche d'un autre, il doit être inférieur ou égal en taille
   Solution:
   $$\begin{array}{l} \forall x \forall y(\\ \quad (leftOf(x, y) \vee frontOf(x, y)) \\ \quad \Rightarrow \\ \quad \neg larger(x, y)) \end{array}$$

   (d) Si un cube large existe, alors tous les tétraèdres sont petits, sauf exactement un tétraèdre qui est large et situé à droite de ce gros cube.
   Solution:
   $$\begin{array}{l} \exists x\ (cube(x) \wedge large(x)) \\ \Rightarrow \\ \exists y\ (tet(y) \wedge large(y) \wedge \\ \quad (\exists x\ (cube(x) \wedge large(x) \wedge rightOf(x, y))) \\ \quad \wedge \\ \quad \forall z\ (\\ \quad\quad (tet(z) \wedge z \neq y) \\ \quad\quad \Rightarrow \\ \quad\quad small(z))) \end{array}$$

2. (20 pts) Donnez une preuve des formules propositionnelles suivantes, en utilisant seulement les règles de la déduction naturelle présentées dans le devoir 2.

   (a) $(\neg A \vee B) \Rightarrow (A \Rightarrow B)$
   Solution:
   $$\dfrac{\dfrac{\dfrac{\dfrac{\lceil \neg A \vee B \rceil^{[1]} \qquad \dfrac{\lceil \neg A \rceil^{[2]} \qquad \lceil A \rceil^{[3]}}{\textbf{false}}\,[\mathrm{E}_{\neg}] \qquad \dfrac{\lceil \neg B \rceil^{[4]} \qquad \lceil B \rceil^{[2]}}{\textbf{false}}\,[\mathrm{E}_{\neg}]}{\textbf{false}}\,[\mathrm{E}_{\vee}]^{[2]}}{B}\,[\mathrm{E}_{\mathbf{false}}]^{[4]}}{A \Rightarrow B}\,[\mathrm{I}_{\Rightarrow}]^{[3]}}{\neg A \vee B \Rightarrow (A \Rightarrow B)}\,[\mathrm{I}_{\Rightarrow}]^{[1]}$$

(b) $(A \land \neg B) \Rightarrow \neg(A \Rightarrow B)$

Solution:

$$
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\lceil A \land \neg B \rceil^{[1]}}{A}\,[\mathrm{E}_{\land 1}] \quad \lceil A \Rightarrow B \rceil^{[2]}}{B}\,[\mathrm{E}_{\Rightarrow}] \quad \dfrac{\lceil A \land \neg B \rceil^{[1]}}{\neg B}\,[\mathrm{E}_{\land 1}]}{\textbf{false}}\,[\mathrm{E}_{\neg}]}{\neg(A \Rightarrow B)}\,[\mathrm{E}_{\mathrm{false}}]^{[2]}}{A \land \neg B \Rightarrow \neg(A \Rightarrow B)}\,[\mathrm{I}_{\Rightarrow}]^{[1]}
$$

3. (8 pts) Soit les définitions suivantes.

$$
\begin{aligned}
A &= \{0, 1\}, \\
B &= \{a, b\}, \\
D &= \{0 \mapsto a, 1 \mapsto b\}
\end{aligned}
$$

Déterminez si les énoncés suivants sont vrais et justifiez votre réponse par un commentaire ou un argument mathématique.

(a) $\{0 \mapsto a\} \in A \times B$ **Solution:** Faux. Les éléments de $A \times B$ sont des couples, pas des ensembles de couples.

(b) $\{0 \mapsto a\} \subseteq A \leftrightarrow B$ **Solution:** Faux. $A \leftrightarrow B$ est un ensemble de relations; donc on devrait avoir à gauche un ensemble de relations, et non pas seulement une relation.

(c) $D \subseteq A \times B$ **Solution:** Vrai. D est un ensemble de paires.

(d) $D \in A \rightarrowtail B$ **Solution:** Vrai. La fonction est totale et associe des éléments distincts.

4. (6 pts) Évaluez les expressions suivantes, en utilisant les définitions données au numéro précédent.

(a) $A \rightarrowtail\!\!\!\!\twoheadrightarrow B$ **Solution:** $\{\{0 \mapsto a, 1 \mapsto b\}, \{0 \mapsto b, 1 \mapsto a\}\}$

(b) $\{0\} ⩤ D$ **Solution:** $\{1 \mapsto b\}$

(c) $(A \times B) \mathbin{<\!\!\!+} \{1 \mapsto a\}$ **Solution:** $\{0 \mapsto a, 0 \mapsto b, 1 \mapsto a\}$

5. (5 pts) Déterminez la précondition la plus faible de l'opération suivante

$$op(x) = PRE\ \ldots\ THEN\ y := y - x\ END$$

afin qu'elle préserve l'invariant

$$y \geq 0$$

Justifiez votre réponse. **Solution:** $[y := y - x]y \geq 0 \equiv y - x \geq 0 \Leftrightarrow y \geq x$

6. (5 pts) Écrivez une opération qui retourne l'ensemble des nombres premiers. Un nombre n est premier ssi ses seuls facteurs sont 1 et n.

Solution: $s \leftarrow premier = s := \{n \mid n \in \mathbb{N} \land \forall x, y \cdot x \in \mathbb{N} \land y \in \mathbb{N} \land n = x * y \Rightarrow \{x, y\} = \{1, n\}\}$

7. (40 pts) Écrivez une spécification B pour le système suivant. Le système doit permettre de gérer des équipes de hockey. Une équipe a les attributs suivants: idEquipe et nomEquipe. Un joueur a les attributs suivants: idJoueur et nomJoueur. Une équipe embauche un joueur en signant un contrat qui spécifie seulement le salaire. Au cours de sa carrière, un joueur peut

faire partie de plusieurs équipes; entre autres, il peut avoir joué pour une équipe à des époques différentes suite aux échanges. À chaque fois qu'un joueur est embauché ou échangé, on crée un contrat. On doit être capable d'afficher en ordre *chronologique* les équipes successives avec lesquelles un joueur a évolué et le salaire pour chacun de ces contrats. Pour identifier les variables d'état, considérez seulement les opérations suivantes.

(a) `ajoJoueur(idJoueur,nomJoueur)`
ajoute le joueur dans le système.

(b) `supJoueur(idJoueur)`
supprime le joueur du système; son historique de contrats est aussi supprimé.

(c) `ajoÉquipe(idEquipe,nomEquipe)`
ajoute l'équipe dans le système.

(d) `engager(idEquipe,idJoueur,salaire)`
engagement d'un joueur par une équipe.

(e) `sortie` ← `affCarriere <idJoueur>`
affiche en ordre *chronologique* les contrats d'un joueur.

Ne spécifiez que les opérations `ajoJoueur`, `supJoueur`, `engager`, par soucis de concision.

À titre illustratif, voici une séquence d'appels effectuée à partir de l'état initial de la machine. Le dernier appel illustre le résultat de l'opération `affCarriere`.

```
ajoJoueur(1,"Joe")
ajoJoueur(2,"Jack")
ajoEquipe(1,"Les poches")
ajoEquipe(2,"Les etoiles")
engager(1,1,1000)
engager(2,2,1000000)
engager(2,1,1500000)
engager(1,1,500)
affCarriere(1) retourne la séquence
  [ "Les poches" ↦ 1000, "Les etoiles" ↦ 1500000, "Les poches" ↦ 500 ]
```

```
MACHINE
    hockey
  SETS
    JOUEUR;
    EQUIPE
  VARIABLES
    joueur, nomJoueur, contrats,
    equipe, nomEquipe
  INVARIANT
    joueur ⊆ JOUEUR ∧
    nomJoueur ∈ joueur → STRING ∧
    equipe ⊆ EQUIPE ∧
    nomEquipe ∈ equipe → STRING ∧
    contrats ∈ joueur → seq(equipe × NAT)
  INITIALISATION
    joueur, nomJoueur, contrats,
    equipe, nomEquipe
    :=
     ∅ , ∅ , ∅ , ∅ , ∅
  OPERATIONS
  ajoJoueur(idJoueur,pNomJoueur) =
    PRE
      idJoueur ∈ JOUEUR - joueur ∧
      pNomJoueur ∈ STRING
    THEN
      joueur := joueur ∪ {idJoueur} ||
      nomJoueur(idJoueur) := pNomJoueur ||
      contrats(idJoueur) := []
    END;
  supJoueur(idJoueur) =
    PRE
      idJoueur ∈ joueur
    THEN
      joueur := joueur - {idJoueur} ||
      nomJoueur := {idJoueur} ⩤ nomJoueur ||
      contrats := {idJoueur} ⩤ contrats
    END;
  ajoEquipe(idEquipe,pNomEquipe) =
    PRE
      pNomEquipe ∈ STRING ∧
      idEquipe ∈ EQUIPE - equipe
    THEN
      equipe := equipe ∪ {idEquipe} ||
      nomEquipe(idEquipe) := pNomEquipe
    END;
  engager(idEquipe,idJoueur,salaire) =
    PRE
      idEquipe ∈ equipe ∧
```

$idJoueur \in joueur \wedge$
$salaire \in \mathbf{NAT}$
THEN
$\mathbf{contrats}(idJoueur) := contrats(idJoueur) \leftarrow (idEquipe \mapsto salaire)$
END;
$sortie \leftarrow \mathbf{affCarriere}(idJoueur) =$
PRE
$idJoueur \in joueur$
THEN
$sortie := contrats(idJoueur)$
END
END