Université de Sherbrooke, Département d'informatique

IGL501 : Méthodes formelles en génie logiciel, Examen périodique
Professeur : Marc Frappier, Vendredi 11 octobre 2013, 8h30 à 11h20, local D4-2021

Documentation permise. 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 :

1	30 pts	2	10 pts	3	20 pts	4	40 pts	Total	100 pts

1. (30 pts) Prouvez les formules suivantes

a) A ∧ (¬ A ∨ B) ⇒ A ∧ B

$$\dfrac{\dfrac{\dfrac{\lceil A \wedge (\neg A \vee B)\rceil^{[1]}}{\neg A \vee B}\,[\mathrm{E}_{\wedge 2}] \qquad \dfrac{\dfrac{\dfrac{\lceil A \wedge (\neg A \vee B)\rceil^{[1]}}{A}\,[\mathrm{E}_{\wedge 1}] \qquad \lceil \neg A)\rceil^{[2]}}{\mathbf{false}}\,[\mathrm{E}_{\neg}]}{A \wedge B}\,[\mathrm{E}_{\mathbf{false}}] \qquad \dfrac{\dfrac{\lceil A \wedge (\neg A \vee B)\rceil^{[1]}}{A}\,[\mathrm{E}_{\wedge 1}] \qquad \lceil B\rceil^{[2]}}{A \wedge B}\,[\mathrm{E}_{\vee}]^{[2]}}{A \wedge B}}{A \wedge (\neg A \vee B) \Rightarrow A \wedge B}\,[\mathrm{I}_{\Rightarrow}]^{[1]}$$

b) A ∨ (¬ A ∧ B) ⇒ A ∨ B

$$\dfrac{\dfrac{\lceil A \vee (\neg A \wedge B)\rceil^{[1]} \qquad \dfrac{\lceil A\rceil^{[2]}}{A \vee B}\,[\mathrm{I}_{\vee 1}] \qquad \dfrac{\dfrac{\lceil \neg A \wedge B\rceil^{[2]}}{B}\,[\mathrm{E}_{\wedge 2}]}{A \vee B}\,[\mathrm{I}_{\vee 2}]}{A \vee B}\,[\mathrm{E}_{\vee}]^{[2]}}{A \vee (\neg A \wedge B) \Rightarrow A \vee B}\,[\mathrm{I}_{\Rightarrow}]^{[1]}$$

2. (10 pts) Soit **SETS** $A=\{a1,a2\}$. Indiquez si les expressions suivantes sont vraies ou fausses, Justifiez votre réponse.

1. a1↦a2 ⊆ A↔A; **réponse: faux; mal typé**
2. a1↦a2 ∈ A × A; **réponse: vrai**
3. { a1↦a2 } ⊆ A↔A; **réponse: faux; mal typé**
4. { a1↦a2 } ∈ A ⇸ A; ; **réponse: vrai**
5. A↠B ⊆ A↔A ; **réponse: vrai**;

3. (20 pts) Pour chaque opération ainsi que pour l'initialisation de la machine suivante, indiquez si l'invariant est préservé.

MACHINE *Q3*

CONSTANTS *AA*

PROPERTIES
$AA = 0..1$

VARIABLES xx, zz

INVARIANT
$xx \in 1..3$
$\wedge \quad zz \in AA \rightarrowtail AA$

INITIALISATION
$xx := 2$
|| **ANY** $zz\_$ **WHERE** $zz\_ \in AA \rightarrowtail AA$ **THEN** $zz := zz\_$ **END**
; **réponse: vrai**

OPERATIONS

op1(yy) = **PRE** $yy \in 0..1 \wedge xx = 2$ **THEN CHOICE** $xx := xx + yy$ **OR** $xx := xx - yy$ **END END**
; **réponse: vrai**
op2(yy) =
PRE $yy \in -1..1$ **THEN**
SELECT $yy < 0$ **THEN** $xx := xx - yy$
WHEN $yy \geq 0$ **THEN** $xx := xx + yy$
END
END
; **réponse: faux**
op3 =
ANY yy **WHERE** $yy \in 0..1$ **THEN**
SELECT $xx = 2$ **THEN** $xx := xx + yy$
WHEN $xx < 2$ **THEN** $xx := xx - yy$
WHEN $xx > 2$ **THEN** $xx := xx + yy$
END
END
; **réponse: faux**
op4 =
ANY aa, bb **WHERE** $aa \in AA \wedge bb \in AA \wedge aa \neq bb$ **THEN**
$zz := zz \Leftarrow \{aa \mapsto zz(bb), bb \mapsto zz(aa)\}$
END
réponse: vrai
END

4. (40 pts) Écrivez une spec B pour le système suivant. Le système gère des trains sur une voie. Une voie est divisée en segments. Chaque segment comporte un stationnement. Pour simplifier, les segments sont numérotés de 0 à max_segment. Dans l'état initial, il n'y a aucun train. L'opération ajouter_train(tt,ss) ajoute un train tt dans un segment ss. L'opération avancer(tt) fait passer un train tt au segment suivant. L'opération garer(tt) gare un train sur le stationnement d'un segment, afin de laisser passer un autre train. L'opération repartir(tt) déplace le train tt du stationnement du segment et le remet sur le segment. Il ne peut y avoir plus d'un train sur un stationnement. Il ne peut y avoir plus d'un train sur un segment. Spécifiez tous les invariants pour assurer la sécurité des trains.

Solution

MACHINE *Q4*

SETS *TRAIN*

CONSTANTS *max_segment*, *SEGMENT*

PROPERTIES
$max\_segment \in 0..\mathbf{MAXINT}$
$\wedge \quad SEGMENT = 0..max\_segment$

VARIABLES
train, *position*, *stationnement*

INVARIANT
$train \subseteq TRAIN$
$\wedge \quad position \in train \rightarrowtail\!\!\!\!\!\!\rightarrow SEGMENT$
$\wedge \quad stationnement \in train \rightarrowtail\!\!\!\!\!\!\rightarrow SEGMENT$
$\wedge \quad train = \mathbf{dom}(position) \cup \mathbf{dom}(stationnement)$
$\wedge \quad \mathbf{dom}(position) \cap \mathbf{dom}(stationnement) = \varnothing$

INITIALISATION
$train := \varnothing$
$\| \quad position := \varnothing$
$\| \quad stationnement := \varnothing$

OPERATIONS

ajouter_train(*tt*,*ss*) =
PRE
$tt \in TRAIN$
$\wedge \quad tt \notin train$
$\wedge \quad ss \in SEGMENT$
$\wedge \quad ss \notin \mathbf{ran}(position)$
THEN
$train := train \cup \{tt\}$
$\| \quad \mathbf{position}(tt) := ss$
END

;
avancer(*tt*) =
PRE
tt ∈ *TRAIN*
∧ *tt* ∈ *train*
∧ *tt* ∈ **dom**(*position*)
∧ *position*(*tt*)+1 ∉ **ran**(*position*)
∧ *position*(*tt*) ≠ *max_segment*
THEN
position(*tt*) := *position*(*tt*)+1
END
;
garer(*tt*) =
PRE
tt ∈ *TRAIN*
∧ *tt* ∈ *train*
∧ *tt* ∈ **dom**(*position*)
∧ *position*(*tt*) ∉ **ran**(*stationnement*)
THEN
position := {*tt*}⩤*position*
|| **stationnement**(*tt*) := *position*(*tt*)
END
;
repartir(*tt*) =
PRE
tt ∈ *TRAIN*
∧ *tt* ∈ *train*
∧ *tt* ∈ **dom**(*stationnement*)
∧ *stationnement*(*tt*) ∉ **ran**(*position*)
THEN
stationnement := {*tt*}⩤*stationnement*
|| **position**(*tt*) := *stationnement*(*tt*)
END
END