TD 1

Modélisation

Exercice 1. Scolarité

On veut spécifier un système de gestion des étudiants. Les étudiants préparant un diplôme et les étudiants diplômés appartiennent à un ensemble STUDENTS (définissant un type) défini dans un contexte Student_Def comme suit.

CONTEXT
  $Student\_Def$
SETS
  $STUDENTS$
  $DIPLOMAS$
AXIOMS
  $axm1 : partition(DIPLOMAS, \{Licence\}, \{Master\}, \{Doctorat\})$
END

L'ensemble DIPLOMAS décrit trois diplômes, la licence, le master et le doctorat.

La machine Ecole décrite ci-dessous utilise le contexte Student_Def.

MACHINE
  $Ecole$
SEES
  $Student\_Def$
VARIABLES
  $Students$,
  $Old\_Students$,
  $Diplome\_En\_Cours$,
  $Diplome\_Obtenu$
INVARIANT
  INV1 : $Students \subseteq STUDENTS$
  INV2 : $Old\_Students \subseteq STUDENTS$
  INV3 : $Diplome\_En\_Cours \in Students \longrightarrow DIPLOMAS$
  INV4 : $Diplome\_Obtenu \in Old\_Students \longrightarrow DIPLOMAS$
  ...
INITIALISATION
  $Students := \emptyset$
  $Old\_Students := \emptyset$
  $Diplome\_En\_Cours := \emptyset$
  $Diplome\_Obtenu := \emptyset$
Events
  $Inscription \triangleq \ldots$

  $Obtenir\_Diplome \triangleq \ldots$

Deux ensembles, définis dans l'état (clause *variables*) de cette machine, sont gérés par cette machine. Il s'agit de

— l'ensemble Students des étudiants qui préparent un diplôme et de

— l'ensemble Old_Students des étudiants ayant obtenu un diplôme.

Ces deux ensembles sont .

A l'initialisation Students et Old_Students sont vides.

Les fonctions Diplome_En_Cours et Diplome_Obtenu, également définies dans l'état (clause *variables*), permettent respectivement d'identifier, pour chaque étudiant, le diplôme en cours de préparation et le diplôme obtenu.

Les évènements suivants sont introduits.

— Inscription est un évènement qui crée un nouvel étudiant et lui associe un diplôme en cours de préparation.

— Obtenir_Diplome permet de conserver tous les étudiants qui ont obtenu un diplome. Ils ne sont donc plus inscrits et deviennent des éléments de Old_Students

À l'initialisation, les ensembles Diplome_En_Cours et Diplome_Obtenu sont vides.

Questions

Q1 Définir un invariant inv5 pour indiquer qu'un étudiant qui a obtenu un diplôme ne prépare plus de diplôme.

Q2 Définir un autre invariant inv6 pour indiquer que le nombre maximal d'étudiants préparant un diplôme donné est inférieur ou égal à 30

Q3 Compléter la machine abstraite Event-B Ecole en exprimant la spécification formelle des évènements Inscription et Obtenir_Diplome. Vous veillerez à respecter l'invariant dans cette spécification.

Q4 Justifier, en quelques lignes, la correction des évènements définis en question Q3 par rapport aux invariants proposés en questions Q1 et Q2.

Q5 Parmi les étudiants qui préparent un diplôme (étudiants dans l'ensemble Diplome_En_Cours) on souhaite distinguer, dans un nouveau modèle, les étudiants qui préparent une *Licence*, un *Master* ou un *Doctorat*. Un raffinement est préconisé pour obtenir ce modèle.

Décrire

— les variables caractérisant le nouvel état dans le raffinement obtenu

— les invariants de ce raffinement

— les évènements raffinés ainsi que les éventuels nouveaux évènements.

Exercice 2. Gestion de mémoire partagée

On s'intéresse à un modèle de gestion de mémoire partagée au travers de la gestion d'un parking. On considère un parking de taille constante N avec des entrées et des sorties de véhicules. A l'initialisation, le parking est vide.

Q1 Dans un premier temps, on ne s'intéressera qu'aux entrées et sorties de véhicules. Décrire une machine Event-B comprenant

— un état (variables) qui caractérise l'état du parking ;

— le ou les invariant(s) nécessaires pour caractériser les états corrects/sûrs du parking ;

— les évènements

  — `Entrer` qui permet à un véhicule de rentrer dans le parking et d'occuper une place libre quelconque ;

  — `Sortir` qui permet à un véhicule présent dans le parking de le quitter.

Q2 On souhaite rendre ce parking payant. Chaque véhicule qui quitte le parking devra s'acquitter d'une somme S (entier positif).

Décrire un raffinement de la machine Event-B obtenue précédemment qui

- introduit un nouvel évènement `Payer` qui permet d'associer une somme S à un véhicule pour indiquer que ce véhicule s'est acquitté de la somme S
- et qui ne permet à un véhicule de quitter le parking seulement si la somme de S a été acquittée pour ce véhicule.

Il faudra donc

- définir l'état de ce raffinement ainsi que les invariants nécessaires,
- raffiner les évènements `Entrer` et `Sortir` si nécessaire,
- introduire le nouvel évènement *Payer* qui provoquera la sortie.

Exercice 3. Un Thermostat

On s'intéresse à la modélisation d'un thermostat pour régler une température dans une pièce. Cette température doit être contrôlée par des actions qui modifient la valeur de la température.

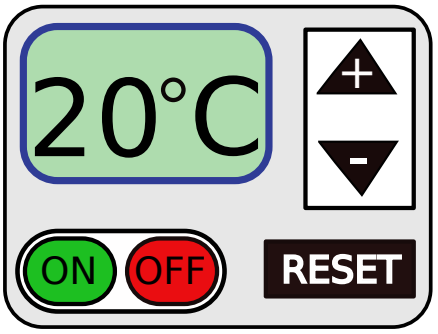

Figure 1 – IHM du thermostat

Les exigences associées à ce système sont définies ci-dessous.

- Une interface utilisateur permet d'allumer ou d'éteindre le thermostat. (FUN).
- Une interface utilisateur permet de fixer la température en utilisant les touches + (plus, up) et - (moins, down). (FUN)
- Le thermostat peut être re-initialisé à tout moment à la température par défaut quand le thermostat est allumé "ON". (FUN)
- La température par défaut du thermostat est 18C
- La température minimale du thermostat est 7C
- La température maximale du thermostat est 35C.

Question.

Q1 Ecrire un modèle Event-B décrivant ce système.

Question susidiaire.

On souhaite doter le thermostat précédent d'un mécanisme de contrôle de la température ambiante. Pour cela, on introduit un capteur de température qui est capable de donner la température de la pièce. Le contrôle consiste donc à déclencher des actions du thermostat pour maintenir ou atteindre une température souhaitée.

Q2 Proposer un modèle Event-B offrant cette nouvelle fonctionnalité