Rendu 2

"January 31, Write a first description of the model and explain your strategy of development of Event-B models"

I. Aperçu du rendu

Nous définissons dans ce document notre stratégie de développement du modèle Event-B. La stratégie générale est celle des raffinements successifs. Dans un premier temps, nous tentons de modéliser la version la plus générale et la plus abstraite du modèle des cartes Suica. Ensuite, au fur et à mesure des raffinements, nous nous rapprochons de la réalité.

Ainsi, nous présentons ci-dessous différents états, qui sont des ensembles {Machine, Contexte}, correspondant aux états successifs de ce qui sera modélisé en Event-B. Nous présentons de manière détaillée l'état le plus général (modèle initial), et nous serons plus succincts pour les états suivants.

Cela décrit la modélisation que nous implanterons. Nous pourrons ainsi vérifier nos propriétés que nous définirons avec les invariants nécessaires.

II. Stratégie de développement des modèles Event-B

ETAT 0 (modèle initial)

MACHINE M0

Sees

C0

Variables

- cartes
- solde

Invariants

- cartes ∈ CARTES
- solde ∈ cartes → ℕ

Evénements

- Initialisation
  - cartes := ∅
  - solde := ∅
- Créer carte
  - **ANY** c **WHERE** c ∉ cartes
    THEN solde(c) := 0; cartes = cartes ∪ {c}
- Supprimer carte
  - **ANY** c **WHERE** c ∈ cartes; solde(c)=0
    THEN solde := {c} ⩤ solde; cartes = cartes \ {c}
- Recharger carte

- **ANY** c, m **WHERE** c ∈ cartes; m ∈ ℕ, m >= 1 000 and m <= 20 000, solde(c) <= 20000-m
  THEN solde(c) := solde(c) + m

- Dépenser
  - **ANY** c, m **WHERE** c ∈ cartes; m ∈ ℕ, solde(c) >= m
    THEN solde(c) := solde(c) - m

CONTEXTE C0

Sets

- CARTES

ETAT 1 (1er raffinement)

MACHINE M1

L'objectif de ce premier raffinement est de considérer un modèle plus précis dans lequel il est possible de dépenser le solde de la carte dans des commerces ou dans les réseaux de transport.

On ajoute une fonction "estdanslemetro" (`estdanslemetro`: cartes ⟶ Bool) qui vaut 1 quand la dernière fois qu'on a passé la carte c'était pour rentrer.

A ceci s'ajoute un événement d'entrée dans le métro qui stipule pour une carte qu'elle est entrée dans le métro. (estdanslemetro(c) = 1)

Après on refine dépenser pour payer quand on sort d'une gare (ie si la var vaut 1 et que le lieu est une gare) et un autre refinement pour les commerces.

CONTEXTE C1

On ajoute au contexte C0 l'ensemble des lieux de paiements (gares ou commerces). Avec des constantes gares et commerces, qui forment une partition des lieux de paiements.

ETAT 2 (2ème raffinement)

MACHINE M2

L'objectif de ce second raffinement est de rajouter l'exigence de versement de la caution lors de l'acquisition de la carte. Il se trouve que cette caution ne peut être dépensée et doit être versée dès l'acquisition. Une modélisation possible est donc d'initialiser (lors de l'instanciation) la carte avec un solde de -500¥ (montant de la caution), obligeant ainsi l'utilisateur à la charger suffisamment (ce qui correspond au versement de la caution) pour pouvoir l'utiliser d'une part, et bloquant cette somme d'autre part (une transaction n'étant permise que si le solde après cette transaction reste positif).

On change le type de solde, qui devient donc ici carte ⟶ ℤ.

On doit donc gérer également le remboursement de la caution lors de la suppression de la carte, ainsi que celui de l'argent restant moyennant une dépense de 220¥. Pour ce faire, on associe à la carte une instance d'un utilisateur d'un nouvel ensemble dans le contexte. Etant donné qu'il n'y a aucune contrainte sur l'utilisateur, ils sont indistinguables. Le caractère "non-nominatif" est donc traduit par cette modélisation. Cela permet seulement de modéliser les transactions entre la carte et un utilisateur quelconque.

Un booléen modélise la volonté de l'utilisateur de récupérer son argent. L'événement supprimer carte est alors raffiné suivant que l'utilisateur fasse ou non une demande de remboursement.

On raffine l'événement. Il prend ce booléen en paramètre et rend l'argent correspondant à la somme de la caution à laquelle s'ajoute éventuellement celle de l'argent restant moins les frais de 220 yens.

CONTEXTE C2

PERSONNES

ETAT 3 (3ème raffinement)

MACHINE M3

Dans ce troisième raffinement, le but est de payer un montant différent en fonction du trajet effectué. SUICA fonctionne avec un système de zones. Les trajets au sein d'une même zone valent le même montant. Lorsqu'un trajet nécessite de changer de zone, l'utilisateur doit de nouveau passer sa carte à un 'access point'. N'ayant pas d'informations plus précises, nous considérons que les stations faisant la jointure entre deux zones possèdent deux types de 'access point' correspondant aux 2 zones.

Nous précisons que les deux types d'access point sont définis comme deux gares différentes dans l'ensemble des lieux de paiement.

Ainsi lorsqu'un voyageur change de zone, il doit d'abord passer par un access point de la zone qu'il quitte afin de régler son trajet dans celle-ci, puis par un access point de la zone qu'il rejoint afin de signaler son entrée dans le réseau.

Nous considérons que les soucis de logistique sont réglés par une aménagement intelligent de ces stations.

Nous changeons donc notre fonction de paiement pour le métro définie au 1er raffinement afin qu'elle prenne en compte la zone d'entrée du voyageur. Etant donné que le voyageur ne peut sortir d'une zone sans payer son voyage il suffit alors d'accommoder le paiement en fonction de la zone d'entrée.

Il faut pour cela ajouter une variable prixZone: gares $\rightarrow \mathbb{N}$

CONTEXTE C2

Rien de plus.

ETAT 4 (4ème raffinement)

Ce raffinement consiste en l'ajout du principe de validité d'une carte. Pour rappel, on considère qu'une carte n'est plus active si elle est inutilisée pendant une période supérieure à 10 ans.

On a donc une fonction valide initialisée à ∅ et définie par valide : cartes ⟶ VALIDITE

Il faut ensuite raffiner les événements de création de carte en ajoutant valide(c) = inactive, et en supprimant du domaine la carte de la fonction valide.

Lorsqu'on fait un achat et que valide(c) = inactive alors valide(c) = active

Aussi, il faut ajouter des nouveaux événements en lien avec la perte de la carte ou l'écoulement de plus de 10 ans de temps. Pour ces événements, valide(c) = périmée

Afin d'exprimer les 10 ans d'inaction, un événement "tic" incrémentera le temps (que nous compterons en mois), et chaque action de la carte réinitialisera ce timer.

Un dernier événement péremption avec comme garde **WHEN** tic = 10 * 12, passera la validité de la carte à périmée (valide(c) = périmée)

Enfin, il faut raffiner tous les événements pour être pris en compte seulement si la carte est valide ou invalide.

CONTEXTE C3

On considère un ensemble VALIDITE et des constantes inactive, active et périmée (ou bloquée), formant une partition de VALIDITE.