Examen de Développement Formel de Systèmes

Documents autorisés, 1h30

Les réponses doivent être justifiées et pertinentes. Des preuves formelles ne sont pas demandées, mais l'argumentaire se doit d'être rigoureux.

Partie I : Questions de cours

Exercice 1 Répondez aux questions suivantes en quelques lignes (max. 3) :

1. Expliquez les différences entre *contextes* et *machines*, et leurs utilités respectives.
2. Qu'est-ce que l'*interblocage* (*deadlock*) ? Qu'est-ce qui le caractérise dans un modèle Event-B ?
3. Expliquez la notion de *bonne définition* (*well-definedness*) d'une formule. On donnera un exemple pertinent, qui permet de distinguer cette notion de celles de correction syntaxique et de typage.
4. Comment appelle-t-on la propriété qui relie les variables d'une machine concrète et d'une machine abstraite dans un raffinement ? Quelle est son utilité ?

Partie II : Étude de cas

Note : il n'est pas demandé de recopier les machines dans leur intégralité, mais seulement les éléments à changer, le cas échéant.

On s'intéresse à un système pour réaliser des achats, dont le fonctionnement est similaire, par exemple, à certaines plate-formes de vente en ligne.

```
CONTEXT Achats0_Ctx
SETS STATUT
CONSTANTS EnCours, Paiement, Payee
AXIOMS
 axm1: partition(STATUT, {EnCours}, {Paiement}, {Payee})
END
```

FIGURE 1 – Contexte pour la machine abstraite

Exercice 2 (Modèle abstrait) Considérons le modèle abstrait donné à la Figure 2, qui implémente un système simple permettant à un utilisateur de gérer une commande.

Le contexte `Achats0_Ctx` définit les états possibles d'une commande (ensemble *STATUT*) : en cours de composition par l'utilisateur (*EnCours*), en cours de paiement (*Paiement*) et payée (*Payee*). Ces états sont mutuellement exclusifs.

La machine `Achats0` décrit l'évolution d'une commande. La variable *statut* représente l'état de la commande. Cet état est initialement *EnCours*. Il passe à *Paiement* via l'événement `Finaliser` qui modélise l'utilisateur achevant sa sélection, puis de *Paiement* à *Payee* via l'événement `Payer` qui modélise l'utilisateur réglant le montant de sa commande.

La variable *paye* garde trace du montant payé par l'utilisateur, et représente d'une certaine façon le transfert d'argent qui advient au moment du paiement.


```
MACHINE Achats0
SEES Achats_Ctx0
VARIABLES
 statut
 paye
INVARIANTS
 inv1: statut ∈ STATUT
 inv2: paye ∈ ℤ
EVENTS
 INITIALISATION
 THEN
  act1: statut := EnCours
  act2: paye :∈ ℤ
 END
```

```
Finaliser
WHERE
 grd1: statut = EnCours
THEN
 act1: statut := Paiement
END

Payer
ANY prix
WHERE
 grd1: statut = Paiement
 grd2: prix ∈ ℤ
THEN
 act1: statut := Payee
 act2: paye := prix
END
```

FIGURE 2 – Machine abstraite représentant le système

2.1 Quel est le sens du symbole :∈ de l'action `act2` de l'initialisation ? Quel est l'intérêt d'utiliser un tel opérateur ? Donnez une écriture équivalente de cette action en utilisant le prédicat avant-après (Before-After Predicate).

2.2 Quelle est la nature formelle de *prix* dans l'événement `Payer` ? Quelle est son utilité ?

2.3 Lorsque la commande est payée, le montant *paye* doit être positif strictement. Complétez la machine avec un invariant `inv3` qui capture cette exigence.

2.4 La machine `Achats0` est-elle correcte vis-à-vis de ses invariants ? Justifiez et, si nécessaire, complétez le modèle pour qu'elle le soit.

```
CONTEXT Achats1_Ctx EXTENDS Achats0_Ctx
SETS ITEMS
END
```

```
MACHINE Achats1 REFINES Achats0
SEES Achats1_Ctx
VARIABLES
 statut
 paye
 Items
 total
INVARIANTS
 inv1: Items ⊆ ITEMS
 inv2: total...
```

```
INITIALISATION REFINES INITIALISATION...
Finaliser REFINES Finaliser...
Payer REFINES Payer...

Ajouter
ANY item, prix
WHERE
 grd1: item ∈ ITEMS
 grd2: prix ∈ ℤ
THEN
 act1: Items...
 act2: total...
END
```

FIGURE 3 – Contexte et machine du premier raffinement

Exercice 3 (Premier raffinement) On désire préciser le fonctionnement du modèle abstrait afin de prendre en compte un mécanisme rudimentaire de *panier d'achat*. Pour cela, on raffine le premier modèle afin d'obtenir le modèle décrit à la Figure 3.

Le contexte `Achats1_Ctx` étend le contexte `Achats0_Ctx` et définit un ensemble de travail pour les articles que l'on peut acheter (*ITEMS*). La machine `Achats1` est un raffinement de la machine `Achats0` qui implémente le système de vente incluant un panier. Le panier est modélisé par la variable *Items* qui est un ensemble d'articles, et par la variable *total* qui garde trace du prix cumulé du panier. L'action d'ajouter un élément (*item*) d'un prix donné (*prix*) dans le panier se fait via l'événement `Ajouter`.

3.1 Complétez l'événement d'initialisation pour les variables *Items* et *total*.

3.2 Complétez les actions (`act1` et `act2`) de l'événement `Ajouter`.

3.3 On ne peut ajouter un article au panier que si la commande est toujours *en cours*. Complétez les gardes de `Ajouter` afin de prendre en compte cette exigence.
3.4 Le prix d'un article est nécessairement strictement positif. Complétez l'événement `Ajouter` de manière à satisfaire cette exigence.
Notons que, si le panier n'est pas vide, alors le total est nécessairement non-nul. Compléter le modèle de façon à prendre en compte cette exigence.
3.5 On se penche maintenant sur l'événement `Payer`. Concrètement, quel prix l'utilisateur doit-il payer pour le panier qu'il a constitué ? Indiquez *précisément* les modifications à apporter à cet événement afin de refléter cet aspect.
3.6 `Ajouter` raffine-t-il un événement de la machine abstraite ? Si oui, lequel ? Le raffinement proposé ici est-il correct ? Sinon, indiquez comment faire en sorte qu'il le soit.
3.7 La machine `Achats1` est-elle correcte ? Justifiez et, si nécessaire, complétez le modèle pour que ça soit le cas.

Exercice 4 (Autres raffinements) Dans cet exercice, on s'intéresse à divers raffinements que l'on pourrait réaliser sur l'exemple précédent.
4.1 On désire prendre en compte la notion de *quantités* de chaque article dans le panier. Concrètement, chaque article du panier est associé à une quantité. Lorsque l'on ajoute un article, si l'article était déjà dans le panier, alors on incrémente sa quantité. Sinon, on l'ajoute dans le panier.

Proposez un raffinement `Achats2` de `Achats1` afin d'implémenter cette fonctionnalité. Pour cela, il est conseillé d'exploiter une variable `quantite` qui, à chaque article dans le panier, associe un nombre entier qui est sa quantité. On raffinera par ailleurs l'événement `Ajouter` en deux événements, suivant que l'article est déjà dans le panier ou non.

On veillera à correctement identifier d'éventuelles *propriétés de sûreté* pour ce nouveau système, et on justifiera sa correction, ainsi que la correction du raffinement avec `Achats1`.
4.2 On aimerait maintenant affiner davantage le mécanisme de paiement : au moment de payer, l'utilisateur sélectionne d'abord un moyen de payement (par exemple : carte bancaire, transfert, chèque par la poste), puis il entre les informations nécessaires et le payement est réalisé.

Proposez un raffinement `Achats3` de `Achats2` afin d'implémenter cette fonctionnalité. On veillera bien à identifier les caractéristiques de ce raffinement : nouvelles données statiques éventuelles (type, constantes, ...), variables abstraites et concrètes, nouveaux invariants, nouveaux événements, liens entres les éléments abstraits et concrets, etc. Justifiez la correction du raffinement et du modèle ainsi obtenu.

Exercice 5 (Autres propriétés) Dans cet exercice, on étudie le comportement des machines obtenues aux exercices précédents.
5.1 Dans une modélisation cohérente, on aimerait que le système soit toujours en mesure d'achever sa tâche, autrement dit que l'utilisateur puisse finaliser puis payer sa commande. De quel type de propriété s'agit-il ? La machine `Achats0` respecte-t-elle cette propriété ? Comment le démontrer ?
5.2 La propriété de la question 5.1 est-elle *en général* préservée par le raffinement ? La machine `Achats1` la préserve-t-elle ? Justifiez et, le cas échéant et si cela est possible, indiquez comment faire en sorte qu'elle soit respectée.