Examen et projet de réalisation en B

Spécifications de l'énoncé

On considère un système automatique de distibuteurs des boissons avec les spécificités suivantes:

- Il y a 2 sortes de boissons : le Coca et le Schweppes.
- Une canette de Coca coûte 60 centimes.
- Une canette de Sweppes coûte 80 centimes.
- Un utilisateur dispose de pièces de 50 et 10 centimes.
- La machine distributrice doit signaler à l'utilisateur si le produit est disponible, et si elle peut rendre la monnaie
- La machine utilisatrice doit indiquer à l'exploitant quand le stock est inquétant.
- L'exploitant peut remettre les produits à leur quantité maximum
- L'exploitant peut vider complètement les bacs à pièce de monnaie.

Tâches

Le travail rendu comporte plusieurs aspects

- Compléter les spécifications de la machine, en justifiant vos choix
- Donner les machines abstraites, leur raffinement et leur implémentaiton pour réaliser une telle machine.
- L'implémentation devra permettre une démonstration.
- Idéalement toutes les obligations de preuves devront être déchargées automatiquement
- Si tel n'est pas le cas, donner une (brève) analyse des obligations de preuve non prouvées et du stade d'aboutissement.
- Les problèmes éventuels au niveau de la génération du code C peuvent rester sans commentaires
- Le dossier devra comprendre une explication du rôle de chaque machine, et de la structuration globale.
- Le projet est à rendre avant la fin des partiels.
- En plus du dossier, envoyer les machines, raffinements et implémentations.

Spécifications détaillées

Variables et Constantes

- Distributeur
  - Le distibuteur a une capacité maximale de canettes.
    - Lors de l'implémentation, on la fixe à 10 cannettes par type
  - Le distibuteur est rempli initialement
  - Le distibuteur a un seuil d'alerte de stock faible dans un des types de boisson
    - Lors de l'implémentation, on le fixe à 3 cannettes
  - Le distributeur doit pouvoir parfois rendre la monnaie. On a donc besoin d'un bac par type de

pièce
    - On ne définit pas de capacité maximale par bac de pièce, elle est considérée grande devant les recettes de la machine.
    - On considère les bacs de pièces initialement vides
    - Utilisateur
    - On considérera un seul utilisateur *Utilisateur*, qui sera directement en interaction avec la machine (on pourrait par la suite imaginer deux utilisateurs jean.utilisateur et pierre.utilisateur interagissant avec la machine au sein d'un composant *pièce*)
    - Un utilisateur dispose d'un certain nombre de pièces de chaque type au sein d'un porte monaie qui se chargera de déterminer avec quelles pièces payer
    - Lors de l'implémentation, on le fixe à 3x50cts et 10x10cts

Opérations

- Distibuteur
    - Le distibuteur doit signaler à l'utilisateur si le produit est disponible,
    - Le distibuteur doit signaler à l'utilisateur si elle peut rendre la monnaie. Le distibuteur peut rendre la monnaie à partir du moment où elle a 4 pièces de 10 cts
- Exploitant
    - L'exploitant peut remettre les produits à leur quantité maximum
    - L'exploitant peut vider complètement les bacs à pièce de monnaie.
- Processus d'achat
    - l'utilisateur et la machine sont tels que
    - ou bien une des conditions d'achat n'est pas validée
    - ou bien l'utilisateur perd la somme équivalente au prix, la machine gagne la somme équivalente au prix et la machine perd un produit On pourra rafiner la machine abstraite de la manière suivante
    - L'utilisateur verifie la disponibilité du produit
    - L'utilisateur vérifie qu'il a assez pour acheter
    - L'utilisateur verifie la possibilité de rendu de monnaie
    - Achat :
    - L'utilisateur donne ses pieces
    - Le distributeur reçoit les pieces
    - Le distributeur donne le produit
    - La machine indique à l'exploitant si le stock devient inquétant

Machines abstraites et Implémentation

voir fichier .mch et .imp joints

On utilise 3 machines distinctes :

- Le Distributeur, qui gère le stock de boissons, le stock de pièces, le paiement, le remplissage des boissons et la vidange du stock de pièces.
- Le PorteMonnaie, qui assure le stock de pièces de l'utilisateur, qui varie lors d'un paiement.

- L'utilisateur, qui assure la liaison entre le distributeur et le porte monnaie lors de l'achat, et qui peut vérifier la disponibilité des boissons ou de la monnaie.

L'exploitant a été intégré au distributeur. On considère que l'exploitant correspond aux opérations *DistributeurRempli* et *BacsMonnaieVides*. En effet, la composition en B ne permet pas à un exploitant de composer un distributeur étant le même que l'utilisateur, à moins de recourir à des structures exotiques (par ex l'exploitant qui importe un utilisateur qui importe la machine, et donner à l'utiliseur les fonctions de remplissage et de vidage)

Preuves

Les machines abstraites sont entièrement prouvées automatiquement. L'implémentation du *PorteMonnaie* est entièrement prouvée automatiquement.

- Concernant l'implémentation du *Distributeur*, 25 des 27 obligations de preuves sont déchargées automatiquement. Le cas qui n'est pas prouvé (PO10 de *DistributeurRecoitArgentPourX*) correspond au cas où le nombre de canette est supérieur au seuil d'alerte. Il n'y a donc pas de changement de valeur de l'alerte. On remarquera néanmoins que l'alerte peut être activée, conséquence du manque de cannette de l'autre denrée. Le cas P05, correspond au cas opposé, est prouvé correctement.
- Concernant l'implémentation de l'utilisateur, l'initialisation pose problème. En effet, il semblerait que le moteur de preuve ne soit pas capable de prouver par exemple que 10=10. Nous n'avons visiblement pas compris certains aspect du prouveur. De plus, l'obligation de preuve PO03 pose aussi problème; cela correspond au cas vu précédemment où le stock de la denrée à acheter est supérieur au seuil d'alerte.

Génération du code C

La génération en C des implémentations ne réussi pas.

- Pour le distributeur : car le checkB0 renvoie "extensive set is not a term" pour la définition de *EtatsAlerte = {0,1}*. Nous ne savons pas comment exprimer l'ensemble différemment.
- Pour le PorteMonnaie : car le checkB0 renvoie "expression in parenthesis is not a simple term" dans l'évaluation de *bool(prix<=(userNbPieces5050+userNbPieces10*10)). Il serait nécessaire de recourir à une variable intermédiaire.

Le reste est fonctionnel