

## EventB Exercises 3 (extended) Supermarket

The objective of this set of tutorial exercises is to develop a specification of a simple supermarket.

### 1 The machines

#### 1.1 The Supermarket.ctx machine

This context machine models the “things” that you find in a supermarket.

**CONTEXT** Supermarket.ctx

#### SETS

TROLLEY

PRODUCT

#### CONSTANTS

SHELF

PRICE

Milk

Cheese

Cereal

BASKET

ProdInTrolleys

maxtrolley

#### AXIOMS

*axm7:*  $PRICE = \mathbb{N}$

*axm1:*  $SHELF = PRODUCT \leftrightarrow \mathbb{N}$

*axm2:*  $PRODUCT = \{Milk, Cheese, Cereal\}$

*axm3:*  $Milk \neq Cheese$

*axm4:*  $Milk \neq Cereal$

*axm5:*  $Cheese \neq Cereal$

*axm6:*  $BASKET = PRODUCT \leftrightarrow \mathbb{N}_1$

*axm8:*  $ProdInTrolleys \in PRODUCT \times (TROLLEY \leftrightarrow BASKET) \leftrightarrow \mathbb{N}$

*axm9:*  $\forall p, ts, t \cdot p \in PRODUCT \wedge ts \in TROLLEY \leftrightarrow BASKET \wedge t \in dom(ts)$   
 $\Rightarrow ProdInTrolleys(p \mapsto ts) = 0$

*axm10:*  $maxtrolley \in \mathbb{N}_1$

*axm11*:  $\forall p, ts \cdot p \in PRODUCT \wedge ts \in TROLLEY \rightarrow BASKET \wedge dom(ts) = \emptyset$   
 $\Rightarrow ProdInTrolleys(p \mapsto ts) = 0$

**END**

Explain the sets and constants you see in the above machine. Give an alternative specification of *PRODUCT* using *partition*.

**1.2 The Supermarket machine**

For the supermarket we want to model the products in the supermarket, the shelf containing the products, the trolleys available for customers, the customers with trolleys and products in those trolleys.

**Important:** all products on the shelves of the supermarket and in the trolleys must have a price.

Here is part of the Supermarket machine.

**MACHINE** Supermarket

**SEES** Supermarket.ctx

**VARIABLES**

**shelf**      The supermarket shelf containing products

**trolleys**    Trolley identities

**products**    products with known prices

**price**      total function from products to a price

**customers**    customers with trolleys

**topay**      amount a customer —represented by a trolley— needs to pay

**stock**      all product in the supermarket: on shelf and in trolleys

**reorderlevel**    level at which a product needs to be re-ordered

**reorder**      a set of all products requiring re-ordering

**onorder**      record reordering of product

**INVARIANTS**

*inv1*:  $shelf \in SHELF$

*inv2:*  $trolleys \subseteq TROLLEY$   
*inv3:*  $finite(trolleys)$   
*inv4:*  $products \subseteq PRODUCT$   
*inv5:*  $finite(products)$   
*inv6:*  $price \in products \rightarrow PRICE$   
*inv7:*  $dom(shelf) = products$   
*inv8:*  $customers \in trolleys \rightarrow BASKET$   
*inv9:*  $\forall t.t \in trolleys \Rightarrow dom(customers(t)) \subseteq products$   
*inv10:*  $topay \in trolleys \rightarrow \mathbb{N}$   
*inv11:*  $stock \in products \rightarrow \mathbb{N}$   
*inv12:*  $\forall p.p \in products$   
 $\Rightarrow stock(p) = shelf(p) + ProdInTrolleys(p \mapsto customers)$   
*inv13:*  $reorderlevel \in products \rightarrow \mathbb{N}_1$   
*inv14:*  $reorder \subseteq products$   
*inv15:*  $onorder \subseteq products$

## EVENTS

### Initialisation

**begin** *act1:*  $shelf := \emptyset$   
*act2:*  $trolleys := \emptyset$   
*act3:*  $products := \emptyset$   
*act4:*  $price := \emptyset$   
*act5:*  $customers := \emptyset$   
*act6:*  $reorderlevel := \emptyset$   
*act7:*  $reorder := \emptyset$   
*act8:*  $topay := \emptyset$   
*act9:*  $stock := \emptyset$   
*act10:*  $onorder := \emptyset$   
**end**

The above machine is intended to model:

- products on the shelf of the supermarket
- products in customer trolleys
- total stock of products: note that *stock* includes all products that are still in the supermarket, either on the shelf, in customers' trolleys or perhaps in reserve somewhere else in the supermarket.
- checkout

- stock alert when stock level drops below some minimum requirement

Complete the Initialisation and add the following events:

**Setprice** set the *price* of a *product*;

**AddStock** add some *amount* of *product* to the supermarket *stock*;

**AddProductShelf** add some *amount* of *product* to the shelf of the supermarket;

**GetTrolley** get a vacant *trolley*;

**AddProductTrolley** take some *amount* of *product* on shelf and add to *trolley*;

**RemProductTrolley** take some *amount* of *product* from *trolley* and return to shelf.

**SetMinStock** set the minimum *amount* of *product* to have in stock;

**CheckOut** checkout *product* from *trolley*;

**Pay** pay for products in *trolley*;

**ReturnTrolley** return empty *trolley*;

**ReStock** indicate that stock of *product* has fallen below minimum stock level.

### 1.3 Refinement of Supermarket Machine

Refine the Supermarket machine, especially showing two methods of implementing CheckOut: one allowing multiple product items to be processed and the other processing one product items at a time.

Events that don't change can be simply inherited using the mysterious first menu on the event line in Rodin.