

This is an interim listing of the Supermarket model for tutorial 3.

**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  
*reorderlevel*   level at which a product needs to be re-ordered  
*reorder*     a set of all products requiring re-ordering  
*topay*       amount a customer —represented by a trolley— needs to pay  
*stock*       all product in the supermarket: on shelf and in trolleys

**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* :  $reorderlevel \in products \leftrightarrow \mathbb{N}_1$   
*inv11* :  $reorder \subseteq products$   
*inv12* :  $topay \in trolleys \rightarrow \mathbb{N}$   
*inv13* :  $stock \in products \rightarrow \mathbb{N}$   
*inv14* :  $\forall p.p \in products$   
 $\Rightarrow stock(p) = shelf(p) + ProdInTrolleys(p \mapsto customers)$

**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* :  $stock := \emptyset$

**end**

**SetPrice**  $\hat{=}$

**any**

*product*  
*aprice*

**where**

*grd1* :  $product \in PRODUCT$   
*grd2* :  $aprice \in \mathbb{N}$

**then**

*act1* :  $price(product) := aprice$   
record the price

*act2* :  $products := products \cup \{product\}$   
add the product to products

*act3* :  $stock : |stock' \in products \cup \{product\} \rightarrow \mathbb{N} \wedge$   
 $(product \notin products \Rightarrow stock' = stock \cup \{product \mapsto 0\})$   
 $\wedge (product \in products \Rightarrow stock' = stock)$

this event includes new and old products these require different actions; for stock

*act4* :  $shelf : |shelf' \in SHELF \wedge$   
 $(product \notin products \Rightarrow shelf' = shelf \cup \{product \mapsto 0\})$   
 $\wedge (product \in products \Rightarrow shelf' = shelf)$

and for shelf; both are total functions

**end**

**AddProduct**  $\hat{=}$

Add product to the supermarket shelf

**any**

*product*  
*amount*

**where**

*grd1* :  $product \in products$   
*grd3* :  $amount \in \mathbb{N}_1$

**then**

*act1* :  $shelf(product) := shelf(product) + amount$

*act2* :  $stock(product) := stock(product) + amount$

**end**

**GetTrolley**  $\hat{=}$

New customer gets a trolley

**any**

*trolley*

**where**

*grd1* :  $card(trolleys) \leq maxtrolley$   
*grd2* :  $trolley \in TROLLEY \setminus trolleys$

**then**

*act1* :  $trolleys := trolleys \cup \{trolley\}$

*act2* :  $customers(trolley) := \emptyset$

*act3* :  $topay(trolley) := 0$

**end**

**AddProductTrolley**  $\hat{=}$ 

Add some of product to trolley from shelf

**any***product**amount**trolley***where***grd1* : *product*  $\in$  *dom(shelf)**grd2* : *shelf(product)*  $\geq$  *amount**grd3* : *trolley*  $\in$  *trolleys***then***act1* : *customers* :  $|customers' \in trolleys \rightarrow BASKET \wedge$  $(product \in dom(customers(trolley)))$  $\Rightarrow customers'(trolley) = customers(trolley) \triangleleft \{product \mapsto customers(trolley)(product) + amount\}$  $\wedge (product \notin dom(customers(trolley)))$  $\Rightarrow customers'(trolley) = customers(trolley) \cup \{product \mapsto amount\}$ 

product may not be already in trolley basket

*act2* : *shelf(product)* := *shelf(product)* - *amount*

remove quantity of product from the shelf

**end****RemProductTrolley**  $\hat{=}$ 

Move product from trolley back to shelf

**any***product**amount**trolley***where***grd1* : *product*  $\in$  *products**grd2* : *trolley*  $\in$  *trolleys**grd3* : *amount*  $\leq$  *customers(trolley)(product)***then***act1* : *customers* :  $|customers' \in trolleys \rightarrow BASKET \wedge$  $(customers(trolley)(product) > amount \Rightarrow customers'(trolley)(product) =$  $customers(trolley)(product) - amount) \wedge$  $(customers(trolley)(product) = amount \Rightarrow customers'(trolley) = \{product\} \triangleleft$  $customers(trolley))$ *act2* : *shelf(product)* := *shelf(product)* + *amount***end****CheckOut**  $\hat{=}$ 

Carry out checkout processing

**any***trolley* choose one trolley wanting to checkout*product* choose all of one product in the trolley**where***grd1* : *trolley*  $\in$  *trolleys**grd2* : *product*  $\in$  *dom(customers(trolley))*

**then**

$act1 : \text{topay}(\text{trolley}) := \text{topay}(\text{trolley}) + \text{customers}(\text{trolley})(\text{product}) * \text{price}(\text{product})$

add cost of this product to the amount to pay

$act2 : \text{customers}(\text{trolley}) := \{\text{product}\} \triangleleft \text{customers}(\text{trolley})$

remove all of this product from trolley

$act3 : \text{stock}(\text{product}) := \text{stock}(\text{product}) - \text{customers}(\text{trolley})(\text{product})$

also remove from stock in supermarket

**end**

**ToPay**  $\hat{=}$

Pay: final checkout action

**any**

$\text{trolley}$  per trolley

$\text{amount}$  amount paid

**where**

$grd1 : \text{trolley} \in \text{trolleys}$

$grd2 : \text{customers}(\text{trolley}) = \emptyset$

trolley is empty

$grd3 : \text{amount} = \text{topay}(\text{trolley})$

amount paid is correct amount

**then**

$act1 : \text{trolleys} := \text{trolleys} \setminus \{\text{trolley}\}$

$act2 : \text{topay} := \{\text{trolley}\} \triangleleft \text{topay}$

$act3 : \text{customers} := \{\text{trolley}\} \triangleleft \text{customers}$

**end**

**SetReorderLevel**  $\hat{=}$

Set reorder level

**any**

$\text{product}$

$\text{level}$

**where**

$grd1 : \text{product} \in \text{products}$

$grd2 : \text{level} \in \mathbb{N}_1$

**then**

$act1 : \text{reorderlevel}(\text{product}) := \text{level}$

**end**

**StockAlert**  $\hat{=}$

Set stock reorder alert

**any**

$\text{product}$

**where**

$grd1 : \text{product} \in \text{dom}(\text{reorderlevel})$

$grd2 : \text{stock}(\text{product}) < \text{reorderlevel}(\text{product})$

$grd3 : \text{product} \notin \text{reorder}$

**then**

$act1 : \text{reorder} := \text{reorder} \cup \{\text{product}\}$

place product in reorder set

```
end
ReOrder  $\hat{=}$ 
  Act on reorder alert
  any
    product
  where
    grd1 : product  $\in$  reorder
  then
    act1 : reorder := reorder \ {product}
    remove product from alert set nothing else shown here as the state of this machine
    does not contain information on ordering
  end
end
END
```