

System Modelling and Design

Ken Robinson

April 22, 2009

©Ken Robinson 2005

mailto::k.robinson@unsw.edu.au

Discussion of SuperMarket Model Ken Robinson April 22, 2009

1 Purpose of this tutorial

This tutorial will discuss the Event-B/RODIN solution previously distributed, which will be referred to as *SuperMarketSolnV0*. It is recommended that the archive for this model be installed and referenced through RODIN. The new supermarket model is *SuperMarketSolnV1* and an archive of that is also available.

This discussion will:

1. Explain some aspects of the model.
2. Correct any errors in the model that are noted.
3. Discuss undischarged proof obligations. Some of that discussion will uncover errors or omissions in the model.
4. Discuss the use of the RODIN provers.

1.1 SuperMarket_ctx: The Context

PRODUCT the set of all products of interest to the supermarket. In the original solution this was given as an enumerated set. This might make the model seem more “real”, but it is not necessary. In version 1 of the solution it will simply be a finite set. This requires a new axiom $\text{finite}(\text{PRODUCT})$, which was implicit in the case it was an enumerated set.

SHELF models a shelf of products, that is a bag of products. Originally modelled as a partial function, $\text{PRODUCT} \leftrightarrow \mathbb{N}$, but as a partial function it doesn't make a lot of sense to use \mathbb{N} for the range (quantity) so will change to \mathbb{N}_1 .

PRICE modelled as \mathbb{N} .

BASKET a bag of products associated with a customer's trolley, modelled as $\text{PRODUCT} \leftrightarrow \mathbb{N}_1$.

ProdInTrolleys a function that models a bag of products composed of the bags of products that are the baskets of the set of trolleys.

1.2 ProdInTrolleys

The *ProdInTrolleys* function in the earlier model was incomplete and also faulty.

The type of *ProdInTrolleys* is

$$ProdInTrolleys \in PRODUCT \times (TROLLEY \leftrightarrow BASKET) \rightarrow \mathbb{N}$$

where

$$BASKET = PRODUCT \leftrightarrow \mathbb{N}$$

models a set of bags of products. This will be used to model the basket associated with trolley.

The intended meaning is that $ProdInTrolleys(product \mapsto customers)$ where

$$customers \in trolleys \rightarrow (products \leftrightarrow \mathbb{N})$$

denotes the sum of the count of *product* across all the trolleys in use by customers.

Thus *ProdInTrolleys* models a *distributed bag*.

Defining the behaviour of ProdInTrolleys

The likely first mistake in defining the behaviour of *ProdInTrolleys* is to attempt to define a function that will compute the sum for some particular state of the supermarket.

This is doomed to failure and is not necessary. What we need to define is how the sum of some product *changes* in events. The process resembles an inductive proof:

Initial state: in which there are no trolleys and the sum of any product will be 0.

an event: the event may change the state of the trolleys —usually only one trolley— and we are interested in the incremental change in the sum of the products.

We will present a set of axioms that describe the various ways in which the content of the trolleys will be modified.

No customers, hence no products in trolleys

$$\begin{aligned} \text{axm8: } & \forall p, ts \cdot p \in PRODUCT \\ & \wedge ts \in TROLLEY \leftrightarrow BASKET \\ & \wedge ts = \emptyset \\ & \Rightarrow \\ & ProdInTrolleys(p \mapsto ts) = 0 \end{aligned}$$

The trolley for one basket is modified, and the modified basket contains the product:

$$\begin{aligned} \text{axm9: } & \forall p, ts, t, b \cdot p \in PRODUCT \wedge ts \in TROLLEY \leftrightarrow BASKET \\ & \wedge t \in TROLLEY \\ & \wedge b \in BASKET \\ & \wedge p \in dom(b) \\ & \Rightarrow \\ & ProdInTrolleys(p \mapsto (ts \triangleleft \{t \mapsto b\})) \\ & = ProdInTrolleys(p \mapsto (\{t\} \triangleleft ts)) + b(p) \end{aligned}$$

The trolley for one basket is modified, and the modified basket does not contain the product:

$$\begin{aligned}
 \text{axm10: } & \forall p, ts, t, b \cdot p \in \text{PRODUCT} \wedge ts \in \text{TROLLEY} \leftrightarrow \text{BASKET} \\
 & \wedge t \in \text{TROLLEY} \\
 & \wedge b \in \text{BASKET} \\
 & \wedge p \notin \text{dom}(b) \\
 \Rightarrow & \\
 & \text{ProdInTrolleys}(p \mapsto (ts \triangleleft \{t \mapsto b\})) \\
 & = \text{ProdInTrolleys}(p \mapsto (\{t\} \triangleleft ts))
 \end{aligned}$$

A trolley containing the product is removed:

$$\begin{aligned}
 \text{axm11: } & \forall p, ts, t \cdot p \in \text{PRODUCT} \wedge ts \in \text{TROLLEY} \leftrightarrow \text{BASKET} \\
 & \wedge t \in \text{dom}(ts) \\
 & \wedge p \in \text{dom}(ts(t)) \\
 \Rightarrow & \\
 & \text{ProdInTrolleys}(p \mapsto (\{t\} \triangleleft ts)) \\
 & = \text{ProdInTrolleys}(p \mapsto ts) - ts(t)(p)
 \end{aligned}$$

A trolley that does not contain the product is removed:

$$\begin{aligned}
 \text{axm12: } & \forall p, ts, t \cdot p \in \text{PRODUCT} \wedge ts \in \text{TROLLEY} \leftrightarrow \text{BASKET} \\
 & \wedge t \in \text{dom}(ts) \\
 & \wedge p \notin \text{dom}(ts(t)) \\
 \Rightarrow & \\
 & \text{ProdInTrolleys}(p \mapsto (\{t\} \triangleleft ts)) = \text{ProdInTrolleys}(p \mapsto ts)
 \end{aligned}$$

1.3 Proof

In this section we will discuss all undischarged proof obligations and show how they are discharged.

1.3.1 INIT/inv14/INV

Goal: $\text{ProdInTrolleys}(p \mapsto \emptyset) = 0$

Find universal quantification for case with $ts = \emptyset$

$$\begin{aligned}
 \forall p \in \text{PRODUCT}, ts \in \text{TROLLEY} \leftrightarrow \text{BASKET} \wedge \\
 ts = \emptyset \\
 \Rightarrow \\
 \text{ProdInTrolleys}(p \mapsto ts) = 0
 \end{aligned}$$

and instantiate ProdInTrolleys with $p = p$ and $ts = \emptyset$.

$$\begin{aligned}
 \emptyset \in \text{TROLLEY} \leftrightarrow \text{BASKET} \\
 \Rightarrow \\
 \text{ProdInTrolleys}(p \mapsto \emptyset) = 0
 \end{aligned}$$

New hypothesis

Click on \implies and select *apply modus ponens*.

goal: $\emptyset \in TROLLEY \leftrightarrow BASKET$

run auto prover PO discharged.

1.3.2 SetPrice/inv2/INV

goal: $shelf' \in products \cup product \rightarrow \mathbb{N}$

Examine the hypotheses and you will notice that it is not known whether $product \in products$. The proof can go in two directions.

Enter $product \in products$ in the proof control scratchpad and select **dc** to proceed by cases. The proof will proceed first with $product \in products$ and when the goal is discharged the proof will proceed with $product \notin products$

Select autoprover and PO will discharge and the hypotheses will display the second case.

Select autoprover again and PO will completely discharge.

1.3.3 SetPrice/inv13/INV

goal: $stock'(p) = shelf'(p) + ProdInTrolleys(p \mapsto customers)$

Again use case analysis on $product$ starting with $product \in products$

Two runs of the autoprover should discharge the PO