Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Licenciatura en Ciencias de la Computación

Trabajo Práctico Final sobre Verificación de Software

Ingeniería de Software I y II

Joaquín Arroyo

17 de febrero de 2024

Índice

1. Introducción

Este trabajo tiene como objetivo verificar un problema de software a través de un proceso que inicia con su especificación formal, la formulación de propiedades/teoremas y sus respectivas demostraciones, seguido de simulaciones y concluyendo con la creación de casos de prueba.

Las herramientas/tecnologías utilizadas para la realización del trabajo son las siguientes:

- Z
- $\{log\}$
- $Z/EVES$
- $FASTEST$

1.1. Enunciado del problema

Sistema de Gestión de Stock de Productos. Se busca desarrollar un sistema de gestión de stock para una tienda en línea que administre la disponibilidad, compra, venta y reposición de productos. Este sistema tiene como objetivo principal mantener actualizados los niveles de inventario y asegurar la disponibilidad de productos para su venta.

El sistema debe ser capaz de:

1. Definir y mantener registros de los diferentes tipos de datos como Producto, Stock, Pedido, etc.
2. Realizar operaciones como añadir un producto al stock, realizar una venta, actualizar la cantidad de productos disponibles, etc.

Además el sistema debe:

1. Mantener invariantes en tiempo de ejecución, como la ausencia de un stock negativo, la correcta actualización del stock tras una venta, etc.
2. Cumplir con propiedades deseables, como la consistencia de la base de datos de stock, la integridad de la gestión de pedidos y reposiciones, la actualización correcta de los niveles de stock después de las ventas, etc.

2. Especificación

2.1. Designaciones

- p es un producto $\approx p \in PRODUCT$
- pid es un identificador de producto $\approx pid \in PID$
- m es el mensaje que devuelve una operación $\approx m \in Msg$
- Se agrega un nuevo producto p con un identificador $pid \approx AddNewProduct(pid, p)$
- Se le añade stock c a un producto con id $pid \approx AddStock(pid, c)$
- Se elimina stock c a un producto con id $pid \approx RemoveStock(pid, c)$
- Se realiza una venta del producto pid con cantidad $c \approx NewSell(pid, c)$

2.2. Especificación en Z

Se introducen los tipos utilizados en la especificación.

$[PID, PRODUCT]$

$MSG ::= error \mid ok$

Definimos las variables de estado con su estado inicial.

Store

$$stock : PID \leftrightarrow \mathbb{N}$$
$$products : PID \leftrightarrow PRODUCT$$
$$sells : PID \leftrightarrow \operatorname{seq} \mathbb{N}$$

StoreInit

Store

$$stock = \emptyset$$
$$products = \emptyset$$
$$sells = \emptyset$$

Invariantes de normalización de variables de estado.

PfunStockInv

Store

$$stock \in PID \nrightarrow \mathbb{N}$$

PfunProductsInv

Store

$$products \in PID \nrightarrow PRODUCT$$

PfunSellsInv

Store

$$sells \in PID \nrightarrow \operatorname{seq} \mathbb{N}$$

Invariante que establece que todo producto existente tiene un stock asociado.

ProductsStockInv

Store

$$\operatorname{dom}\ products = \operatorname{dom}\ stock$$

Invariante que establece que el stock de los productos es mayor o igual a cero.

StockQuantityInv

Store

$$\operatorname{ran}\ stock \subseteq \mathbb{N}$$

Invariante que establece que todo producto existente tiene 0 o más ventas.

ProductsSellsInv

Store

$$\operatorname{dom}\ products = \operatorname{dom}\ sells$$

Las operaciones modeladas permiten agregar un nuevo producto al sistema, agregar/remover stock a un producto existente, y persistir la venta de un producto existente.

AddNewProduct. La siguiente operación permite agregar un nuevo producto al sistema. Es necesario verificar que el identificador del producto no esté ya ingresado.

AddNewProductOk

$\Delta Store$
$pid? : PID$
$p? : PRODUCT$
$m! : MSG$

$pid? \notin \text{dom } products$
$products' = products \cup \{pid? \mapsto p?\}$
$stock' = stock \cup \{pid? \mapsto 0\}$
$sells' = sells \cup \{pid? \mapsto \langle\rangle\}$
$m! = ok$

ExistingPidError

$\Xi Store$
$pid? : PID$
$m! : MSG$

$pid? \in \text{dom } products$
$m! = error$

$AddNewProduct == AddNewProductOk \lor ExistingPidError$

AddStock. La siguiente operación permite agregar stock a un producto. Es necesario verificar que el identificador del producto esté ya ingresado, es decir, que el producto exista, y que además la cantidad de stock sea positiva.

AddStockOk

$\Delta Store$
$pid? : PID$
$c? : \mathbb{N}$
$m! : MSG$

$c? > 0$
$pid? \in \text{dom } products$
$stock' = stock \oplus \{pid? \mapsto stock\ pid? + c?\}$
$products' = products$
$sells' = sells$
$m! = ok$

StockQuantityError

$\Xi Store$
$c? : \mathbb{N}$
$m! : MSG$

$c? \leq 0$
$m! = error$

UnexistingPidError

$\Xi Store$
$pid? : PID$
$m! : MSG$

$pid? \notin \text{dom } products$
$m! = error$

$AddStock == AddStockOk \lor UnexistingPidError \lor StockQuantityError$

RemoveStock. La siguiente operación permite remover stock de un producto. Es necesario verificar que el identificador del producto esté ya ingresado, y además que la cantidad a remover exista.

RemoveStockOk

$\Delta Store$
$pid? : PID$
$c? : \mathbb{N}$
$m! : MSG$

$c? > 0$
$pid? \in \text{dom } products$
$stock\ pid? - c? \geq 0$
$stock' = stock \oplus \{pid? \mapsto stock\ pid? - c?\}$
$sells' = sells$
$products' = products$
$m! = ok$

NotEnoughStockError

$\Xi Store$
$pid? : PID$
$c? : \mathbb{N}$
$m! : MSG$

$pid? \in \text{dom } products$
$stock(pid?) - c? < 0$
$m! = error$

$RemoveStock == RemoveStockOk \lor StockQuantityError \lor UnexistingPidError \lor NotEnoughStockError$

NewSell. La siguiente operación permite persistir la venta de un producto con una cantidad asociada. Es necesario verificar que el producto exista, además la cantidad a vender debe ser positiva y debe haber stock del producto. Esta operación no solo persiste la venta, si no también actualiza el stock del producto vendido.

NewSellOk

$\Delta Store$
$pid? : PID$
$c? : \mathbb{N}$
$m! : MSG$

$c? > 0$
$pid? \in \text{dom } products$
$stock\ pid? - c? \geq 0$
$sells' = sells \oplus \{pid? \mapsto sells\ pid? \frown \langle c? \rangle\}$
$stock' = stock \oplus \{pid? \mapsto stock\ pid? - c?\}$
$products' = products$
$m! = ok$

$$NewSell == NewSellOk \lor StockQuantityError \lor UnexistingPidError \lor NotEnoughStockError$$

2.3. Simulaciones en {*log*}

Notar que para la traducción de la especificación Z a $\{log\}$ fue necesario utilizar funciones sobre listas definidas en la librería *setlogliblist.slog* mencionada en la página oficial de setlog.

Al necesitar estas funciones tipadas, y por problemas con el *VCG*, el cuál no reconocía las funciones importadas, dichas funciones fueron copiadas y tipadas en el código de la traducción de la especificación Z (En la siguiente sección se profundiza más sobre estos problemas).

Simulación 1. En esta simulación, se agrega el producto (1, p1) al sistema, luego se le da un stock de 10 unidades, y por último se venden 5 unidades de dicho producto en un primer momento, y luego otras 5.

```
storeInit(Stock, Products, Sells)                                              &
addNewProduct(Stock, Products, Sells, 1, 'p1', ok, Stock1, Products1, Sells1) &
addStock(Stock1, Products1, Sells1, 1, 10, ok, Stock2, Products2, Sells2)      &
newSell(Stock2, Products2, Sells2, 1, 5, ok, Stock3, Products3, Sells3)        &
newSell(Stock3, Products3, Sells3, 1, 5, ok, Stock_, Products_, Sells_).

Resultado:

Stock     = {},
Products  = {},
Sells     = {},
Stock1    = {[1,0]},
Products1 = {[1,p1]},
Sells1    = {[1,{}]},
Stock2    = {[1,10]},
Products2 = {[1,p1]},
Sells2    = {[1,{}]},
Stock3    = {[1,5]},
Products3 = {[1,p1]},
Sells3    = {[1,{[1,5]}]},
Stock_    = {[1,0]},
Products_ = {[1,p1]},
Sells_    = {[1,{[2,5],[1,5]}]}
```

Simulación 2. En la segunda simulación, agregamos el producto $(2, p2)$ al sistema, luego le damos un stock de 5 unidades, intentamos vender 6 unidades lo cuál deriva en un error, continuamos agregando una unidad al stock del producto $p2$, y nuevamente intentamos vender 6 unidades de dicho producto.

```
storeInit(Stock, Products, Sells)                                              &
addNewProduct(Stock, Products, Sells, 2, 'p2', ok, Stock1, Products1, Sells1) &
addStock(Stock1, Products1, Sells1, 2, 5, ok, Stock2, Products2, Sells2)      &
newSell(Stock2, Products2, Sells2, 2, 6, error, Stock3, Products3, Sells3)    &
addStock(Stock3, Products3, Sells3, 2, 1, ok, Stock4, Products4, Sells4)      &
newSell(Stock4, Products4, Sells4, 2, 6, ok, Stock_, Products_, Sells_).

Resultado:

Stock     = {},
Products  = {},
Sells     = {},
Stock1    = {[2,0]},
Products1 = {[2,p2]},
Sells1    = {[2,{}]},
```

```
Stock2    = {[2,5]},
Products2 = {[2,p2]},
Sells2    = {[2,{}]},
Stock3    = {[2,5]},
Products3 = {[2,p2]},
Sells3    = {[2,{}]},
Stock4    = {[2,6]},
Products4 = {[2,p2]},
Sells4    = {[2,{}]},
Stock_    = {[2,0]},
Products_ = {[2,p2]},
Sells_    = {[2,{[1,6]}]}
```

2.4. VCG en {*log*}

En la primera interacción con el VCG, tuve problemas debido a la utilización de la librería oficial de listas de *setlog*, ya que las funciones utilizadas de dicha librería no eran reconocidas por el VCG.

En primera instancia, intenté agregar la cláusula *consult_lib.* en el código del VCG, lo cuál no funcionó. Luego, como solución final, no solo por este problema, sino también por lo mencionado en la sección anterior acerca del tipado, decidí incluir las dos funciones utilizadas en el archivo de mi especificación, en el cuál además las tipé.

Finalmente, pude ejecutar el VCG y llamar al método generado por este para verificar las condiciones de verificación. Luego de esto, fallaron la condiciones *addNewProduct_pi_pfunStockInv* y *addNewProduct_pi_pfunSellsInv* por lo que pasé a utilizar el comando **findh** el cuál me devolvió las hipótesis *productStockInv* y *productsSellsInv* para agregar sobre dichas condiciones respectivamente. Luego de agregar dichas hipótesis, volví a ejecutar las condiciones de verificación obteniendo un resultado exitoso.

3. Demostración sobre *Z/EVES*

El lema de invariancia elegido para demostrar es el siguiente:

theorem AddStockPI
$StockQuantityInv \land AddStock \Rightarrow StockQuantityInv'$

proof[*AddUserRightPI*]
invoke AddStock;
split AddStockOk;
cases;
simplify;
reduce;
next;
split UnexistingPidError;
cases;
simplify;
reduce;
next;
split StockQuantityError;
cases;
simplify;
reduce;
next;
simplify;
■

4. Casos de prueba

Los casos de prueba fueron generados a partir de la especificación Z con FASTEST.

Se generaron casos de prueba para la operación *AddStock*. Para ello, se utilizaron los siguientes comandos:

```
loadspec spec.tex
selop AddStock
genalltt
addtactic AddStock_DNF_1 SP \in pid? \in \dom products
addtactic AddStock_DNF_2 SP \notin pid? \notin \dom products
genalltt
addtactic AddStock_DNF_1 NR c? \langle 1, 100000 \rangle
genalltt
addtactic AddStock_DNF_1 SP + stock(pid?) + c?
genalltt
prunett
genalltca
```

En primer lugar, se aplica la táctica *DNF*, que separa a la clase *VIS* en las clases $AddStock\_DNF_1$, $AddGroup\_DNF_2$ y $AddGroup\_DNF_3$ que representan los casos en donde se aplican las operaciones *AddStockOk*, *UnexistingPidError* y *StockQuantityError* respectivamente.

Aplicamos las particiones de $\in$ y $\notin$ a las dos primeras subclases ($AddStock\_DNF_1$ y $AddStock\_DNF_2$) debido a que estas dos son las que presentan las condiciones con $\in$ y $\notin$ respectivamente.

Posteriormente, se utiliza la táctica NR en $AddStock\_DNF_1$ con el rango $\langle 1, 100000 \rangle$ debido a que esta clase es la única que presenta la condición donde el stock es positivo, así podemos obtener casos de prueba que abarquen desde una cantidad mínima hasta una considerable cantidad de stock.

Por último utilizamos nuevamente la táctica SP con el operador + sobre la subclase $AddStock\_DNF_1$ debido a que esta es la que presenta la condición sobre la variable de estado *stock* en la cuál se agrega stock al stock anterior del producto recibido.

Luego, el árbol de prueba generado es el siguiente:

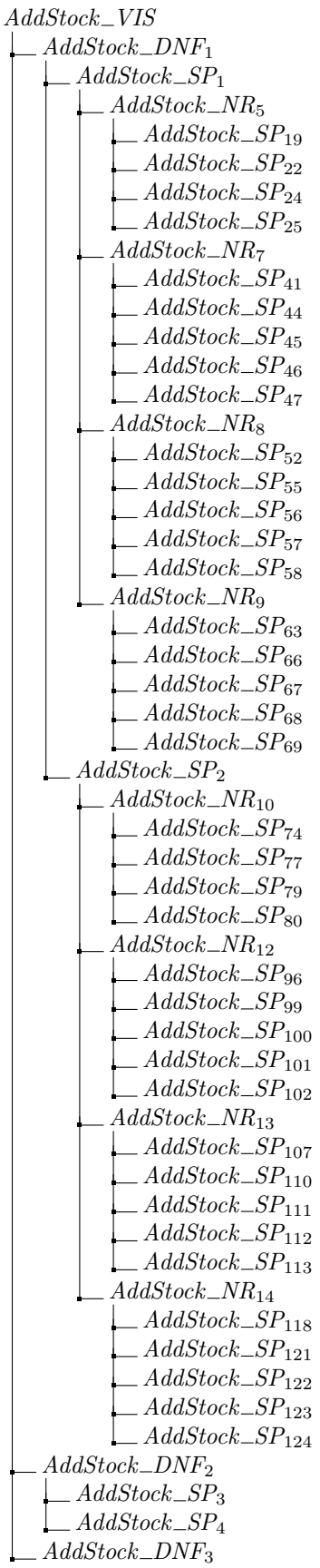

Figura 1: Árbol de prueba generado.

Y los casos de prueba abstractos generados son los siguientes:

AddStock_SP_19_TCASE

AddStock_SP_19

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto -4294967296)\}$
$c? = 1$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_45_TCASE

AddStock_SP_45

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 1)\}$
$c? = 2$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_22_TCASE

AddStock_SP_22

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 0)\}$
$c? = 1$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_46_TCASE

AddStock_SP_46

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 2)\}$
$c? = 2$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_24_TCASE

AddStock_SP_24

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 1)\}$
$c? = 1$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_47_TCASE

AddStock_SP_47

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 3)\}$
$c? = 2$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_25_TCASE

AddStock_SP_25

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 2)\}$
$c? = 1$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_52_TCASE

AddStock_SP_52

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto -4294967296)\}$
$c? = 100000$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_41_TCASE

AddStock_SP_41

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto -4294967296)\}$
$c? = 2$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_55_TCASE

AddStock_SP_55

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 0)\}$
$c? = 100000$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_44_TCASE

AddStock_SP_44

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 0)\}$
$c? = 2$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_56_TCASE

AddStock_SP_56

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 1)\}$
$c? = 100000$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_57_TCASE

AddStock_SP_57

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 100000)\}$
$c? = 100000$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_58_TCASE

AddStock_SP_58

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 100001)\}$
$c? = 100000$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_63_TCASE

AddStock_SP_63

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto -4294967296)\}$
$c? = 100001$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_66_TCASE

AddStock_SP_66

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 0)\}$
$c? = 100001$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_67_TCASE

AddStock_SP_67

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 1)\}$
$c? = 100001$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_68_TCASE

AddStock_SP_68

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 100001)\}$
$c? = 100001$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_69_TCASE

AddStock_SP_69

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 100002)\}$
$c? = 100001$
$products = \{(pID1 \mapsto pRODUCT2)\}$

AddStock_SP_74_TCASE

AddStock_SP_74

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto -4294967296)\}$
$c? = 1$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_77_TCASE

AddStock_SP_77

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 0)\}$
$c? = 1$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_79_TCASE

AddStock_SP_79

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 1)\}$
$c? = 1$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_80_TCASE

AddStock_SP_80

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 2)\}$
$c? = 1$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_96_TCASE

AddStock_SP_96

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto -4294967296)\}$
$c? = 2$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_107_TCASE

AddStock_SP_107

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto -4294967296)\}$
$c? = 100000$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_99_TCASE

AddStock_SP_99

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 0)\}$
$c? = 2$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_110_TCASE

AddStock_SP_110

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 0)\}$
$c? = 100000$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_100_TCASE

AddStock_SP_100

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 1)\}$
$c? = 2$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_111_TCASE

AddStock_SP_111

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 1)\}$
$c? = 100000$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_101_TCASE

AddStock_SP_101

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 2)\}$
$c? = 2$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_112_TCASE

AddStock_SP_112

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 100000)\}$
$c? = 100000$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_102_TCASE

AddStock_SP_102

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 3)\}$
$c? = 2$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_113_TCASE

AddStock_SP_113

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 100001)\}$
$c? = 100000$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_118_TCASE

AddStock_SP_118

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto -4294967296)\}$
$c? = 100001$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_124_TCASE

AddStock_SP_124

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 100002)\}$
$c? = 100001$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_121_TCASE

AddStock_SP_121

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 0)\}$
$c? = 100001$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_3_TCASE

AddStock_SP_3

$sells = \emptyset$
$pid? = pID1$
$stock = \emptyset$
$c? = -4294967296$
$products = \emptyset$

AddStock_SP_122_TCASE

AddStock_SP_122

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 1)\}$
$c? = 100001$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_SP_4_TCASE

AddStock_SP_4

$sells = \emptyset$
$pid? = pID1$
$stock = \emptyset$
$c? = -4294967296$
$products = \{(pID2 \mapsto pRODUCT1)\}$

AddStock_SP_123_TCASE

AddStock_SP_123

$sells = \emptyset$
$pid? = pID1$
$stock = \{(pID1 \mapsto 100001)\}$
$c? = 100001$
$products = \{(pID1 \mapsto pRODUCT2),$
$(pID3 \mapsto pRODUCT2)\}$

AddStock_DNF_3_TCASE

AddStock_DNF_3

$sells = \emptyset$
$pid? = pID1$
$stock = \emptyset$
$c? = -4294967296$
$products = \emptyset$

Se pudieron generar casos de prueba sobre todas las hojas. Paso a analizar los casos donde aparece el valor -4294967296:

- En los casos*AddStock_SP_19*, *AddStock_SP_41*, *AddStock_SP_52*, *AddStock_SP_63*, *AddStock_SP_74*, *AddStock_SP_96*,*AddStock_SP_107* y *AddStock_SP_118* podemos ver que el producto con $pID1$ tiene asociado un stock de -4294967296 lo cuál es correcto debido a que tenemos la condición de que *stock pid?* debe ser negativo.
- En los casos de *AddStock_SP_3*, *AddStock_SP_4* y *AddStock_DNF_3* no hay problema con que la variable de entrada $c?$ tenga el valor -4294967296 ya que en el primer y segundo caso no existe una condición sobre dicha variable, y en el tercer caso $c?$ debe ser 0 o negativo, lo cuál se cumple.