751 Assignment 1: akna890 / 307887883

1. Formal requirements specification for an Online Shopping System:

Prelude notes

- Yellow: Schemas/concepts in the system
- Blue: All parameters and state variables and constants
- Green: Functional requirements and schema operations while all functional requirements and operations of a schema
- The **Z** specification is a **subset** of the functionality of this specified **B** schema

Online Shopping System:

- Clock
    - TIme: Represents the current time (only increasing)
    - Tick: Increase the current time by one
- User
    - Balance: Each user has an amount of money that they hold.
    - Work: A user can work which adds a static positive amount to their balance, This is the money that they earn while they work
- Item
    - Costs: An item costs a certain amount of a users balance
- Shop
    - Users: Tracks which users have created an account with the shop
    - Float: A shop has an amount of money that it earns from people buying items at the store
    - Items: The shop tracks which items that the user has on hand after they have been delivered so that it can track item returns
    - Stocks: A shop keeps track of the amount of each item that it is holding
    - Float: A shop a balance much like a user does which is called a float
    - shippingToUser: Items take a day to arrive to the user, when an item is purchased by the user they are added to the shippingToUser set for the user, after the next day the item is delivered to the user (added to the users item set)
    - shippingToStore: When the user wishes to return an item that they have purchased they need to ship the item back, a returned item also needs to be shipped, after the next day the item is delivered back to the store and added to the stores stocks
    - Creating Account: Users can create an account which is added to the shops tracked users
    - AddItemForSale: A shop can list an item for sale which assigns it a cost and sets the stock for the item to zero
    - AddItemToCart: Adds an item to the users cart, a user can add multiple items to their cart and the items can be duplicates
    - RemoveItemFromCart: Removes the first occurrence of an item from the users cart. The item must be part of the users cart for it to be removed
    - EmptyCart: A quick functionality for resetting a users cart to be empty
    - CartValue: Outputs the value of the users cart by adding together the cost of each item (determined by the shop) within the users cart

- CanFufilCart: The shop must have the stock on hand to cover the items that the user is purchasing, this function checks that there is enough stock on hand to send to the user as a boolean
- CheckoutCart: Transfers the funds required to purchase the items in the users carts to the shop and ship the items to the user. Then clear the users cart.
- ReturnItem: A user can return that they have purchased from the shop. The user gains the amount of money equal to the cost of the item that they are returning. The item is then shipped back to the shop.
- Restock: The shop may restock items that the shop sells. Shop buys items for the same price that it sells them. The shop must have enough float (money) in order to buy the items it selects and funds are removed from the shop to cover the cost of the stock it buys
- NextDay: This function increases the day counter and completes all shipping functionality, including shipping items that the user purchased to each user and also shipping item returns back to the store and adding them to the store's stock count.

2. Z Formal Specification

- The model has been created under the filename ***OnlineShop.tex***, load this into your model checker to simulate it

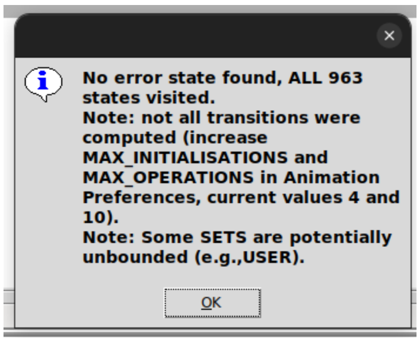
3. Extended B Specification

- The primary model file is under the filename ***OnlineShop.mch***, it is supported by the library files ***Clock.mch***, ***LibrarySequences.mch***, and ***User.mch***

4. B Model Specification and Verification Report

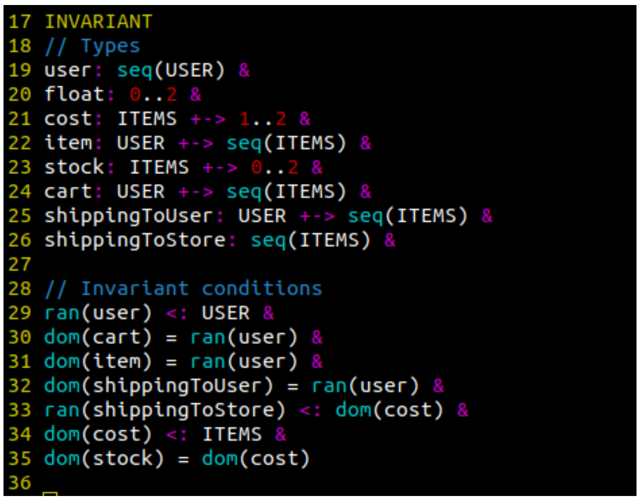
4.1 **Verification of the B Model**

Analysis of the B model (***OnlineShop.mch***) is **unbounded**, due to it using sets such as **NATURAL** which have infinite size, this means that there are infinite possible states that the model can be in. This fact is acknowledged in ProB's documentation. However, partial verification of the system can be made via significantly limiting the model to smaller constraints. The model ***OnlineShop.mch*** has these arbitrary constraints put in place in order to complete verification


```
No error state found, ALL 963
states visited.
Note: not all transitions were
computed (increase
MAX_INITIALISATIONS and
MAX_OPERATIONS in Animation
Preferences, current values 4 and
10).
Note: Some SETS are potentially
unbounded (e.g.,USER).
```

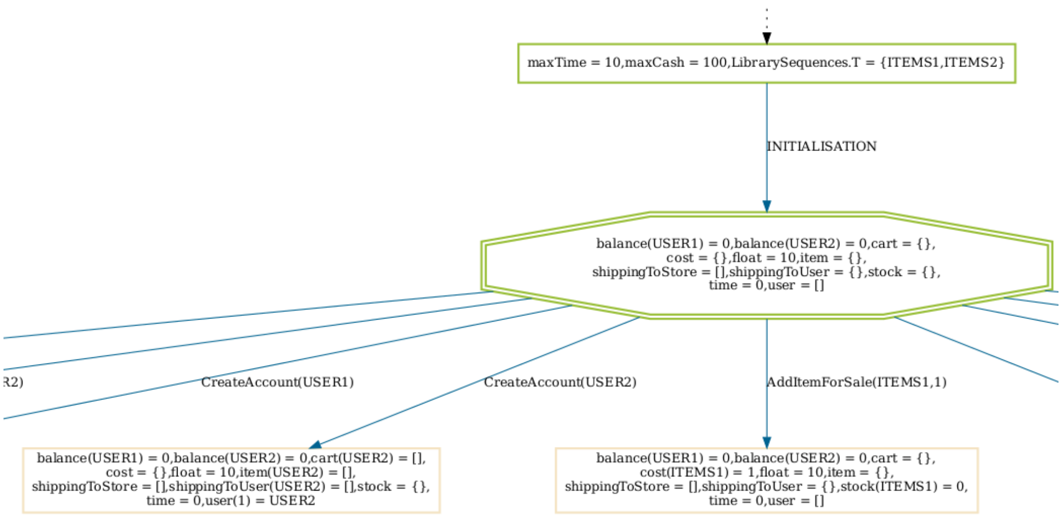
4.2 Rationale of Invariants

This section will rationalise model invariants have have been established that are within ***OnlineShop.mch***



```
17 INVARIANT
18 // Types
19 user: seq(USER) &
20 float: 0..2 &
21 cost: ITEMS +-> 1..2 &
22 item: USER +-> seq(ITEMS) &
23 stock: ITEMS +-> 0..2 &
24 cart: USER +-> seq(ITEMS) &
25 shippingToUser: USER +-> seq(ITEMS) &
26 shippingToStore: seq(ITEMS) &
27 
28 // Invariant conditions
29 ran(user) <: USER &
30 dom(cart) = ran(user) &
31 dom(item) = ran(user) &
32 dom(shippingToUser) = ran(user) &
33 ran(shippingToStore) <: dom(cost) &
34 dom(cost) <: ITEMS &
35 dom(stock) = dom(cost)
36 
```

1. **Invariant 1** (line 29). The users in the system subset of the USER enumerated set
2. **Invariant 2** (line30). When a user is added to the system they also have a cart
3. **Invariant 3** (line 31). When a user is added to the system, the system will also track which items the user has
4. **Invariant 4** (line 32). Items can only be shipped to users which exist in the system
5. **Invariant 5** (line 33). Items can only be returned to the store from users that are registered within the system
6. **Invariant 6** (line 34). The store can only add items to the store that are within the items set
7. **Invariant 7** (line 35). The store must track the stock of all of the items that it sells

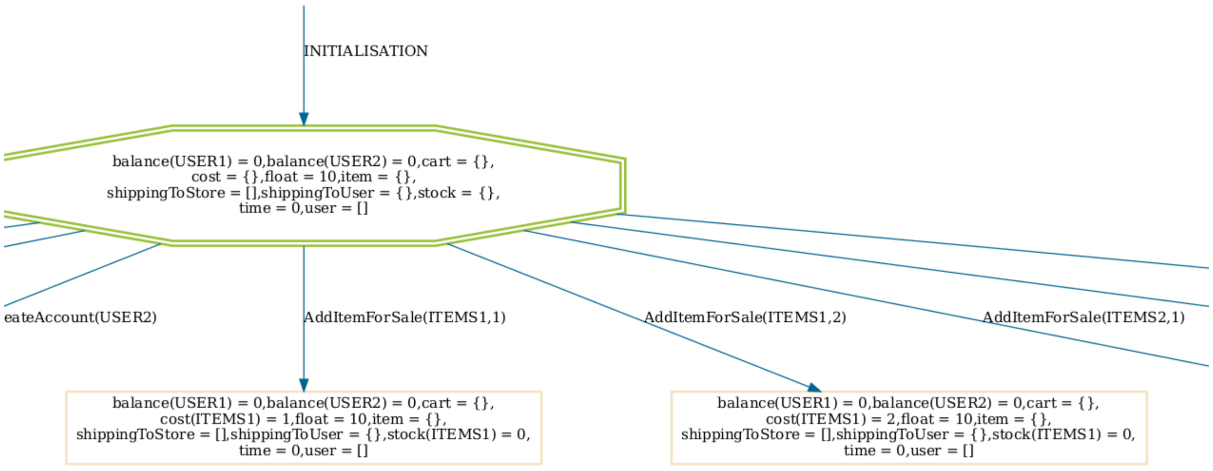

maxTime = 10,maxCash = 100,LibrarySequences.T = {ITEMS1,ITEMS2}
INITIALISATION
balance(USER1) = 0,balance(USER2) = 0,cart = {}, cost = {},float = 10,item = {}, shippingToStore = [],shippingToUser = {},stock = {}, time = 0,user = []
R2)
CreateAccount(USER1)
CreateAccount(USER2)
AddItemForSale(ITEMS1,1)
balance(USER1) = 0,balance(USER2) = 0,cart(USER2) = [], cost = {},float = 10,item(USER2) = [], shippingToStore = [],shippingToUser(USER2) = [],stock = {}, time = 0,user(1) = USER2
balance(USER1) = 0,balance(USER2) = 0,cart = {}, cost(ITEMS1) = 1,float = 10,item = {}, shippingToStore = [],shippingToUser = {},stock(ITEMS1) = 0, time = 0,user = []

When analysing the state space this proves **Invariant 1** when adding a user to the system it uses the USER enumeration defined by the machine, each user can only be added once to the system meaning that it is a subset of the USER enumeration set.

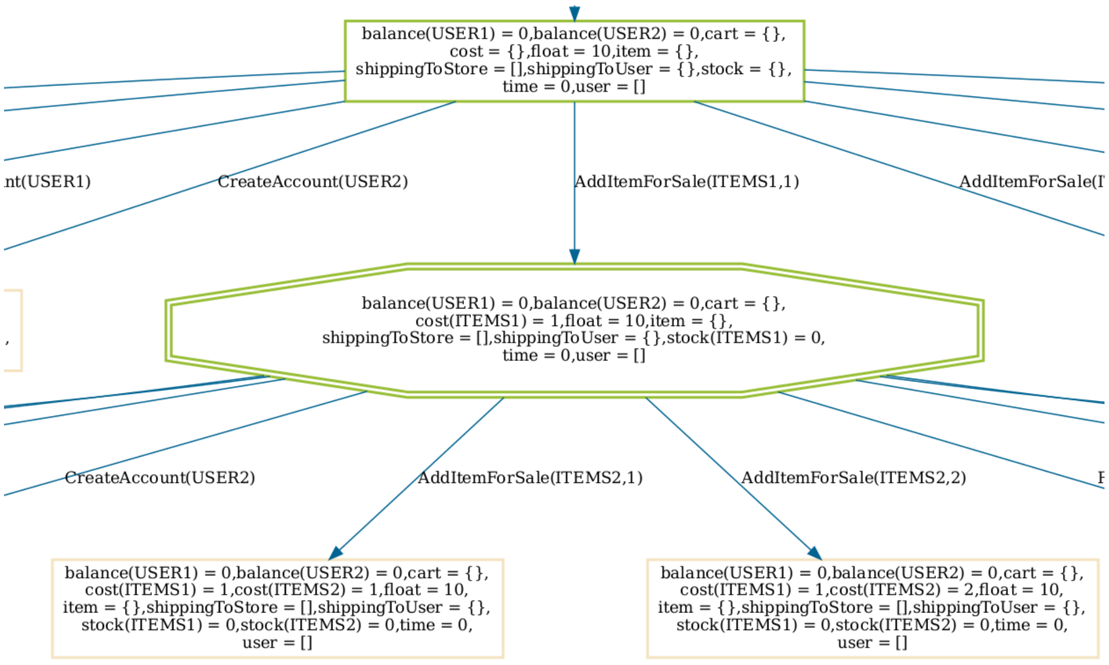
Invariant 2, **Invariant 3, Invariant 4, Invariant 5** are all also proved here, when a user is added to the system, the balance, cart, items, the items shipping to the user, and the items the user is shipping to the store are all set to a starting value (usually the empty set/sequence). This proves that the domains of the cart, items, shippingToUsers and shippingToStore maintain that they are equal to the user set whenever a user is added to the system (the invariant is maintained).

The store starts with no items for sale. Items are added for sale to the store via the AddItemForSale function and an item can only be added once to the store via the cost sequence structure

Before adding ITEM1 for sale, both items (ITEM1, ITEM2) available to be added for sale to the store:


INITIALISATION
balance(USER1) = 0,balance(USER2) = 0,cart = {}, cost = {},float = 10,item = {}, shippingToStore = [],shippingToUser = {},stock = {}, time = 0,user = []
eateAccount(USER2)
AddItemForSale(ITEMS1,1)
AddItemForSale(ITEMS1,2)
AddItemForSale(ITEMS2,1)
balance(USER1) = 0,balance(USER2) = 0,cart = {}, cost(ITEMS1) = 1,float = 10,item = {}, shippingToStore = [],shippingToUser = {},stock(ITEMS1) = 0, time = 0,user = []
balance(USER1) = 0,balance(USER2) = 0,cart = {}, cost(ITEMS1) = 2,float = 10,item = {}, shippingToStore = [],shippingToUser = {},stock(ITEMS1) = 0, time = 0,user = []

After adding ITEM1 for sale, only ITEM2 is available to be added to the store via AddItemForSale:


balance(USER1) = 0,balance(USER2) = 0,cart = {}, cost = {},float = 10,item = {}, shippingToStore = [],shippingToUser = {},stock = {}, time = 0,user = []
nt(USER1)
CreateAccount(USER2)
AddItemForSale(ITEMS1,1)
AddItemForSale(I
balance(USER1) = 0,balance(USER2) = 0,cart = {}, cost(ITEMS1) = 1,float = 10,item = {}, shippingToStore = [],shippingToUser = {},stock(ITEMS1) = 0, time = 0,user = []
CreateAccount(USER2)
AddItemForSale(ITEMS2,1)
AddItemForSale(ITEMS2,2)
balance(USER1) = 0,balance(USER2) = 0,cart = {}, cost(ITEMS1) = 1,cost(ITEMS2) = 1,float = 10, item = {},shippingToStore = [],shippingToUser = {}, stock(ITEMS1) = 0,stock(ITEMS2) = 0,time = 0, user = []
balance(USER1) = 0,balance(USER2) = 0,cart = {}, cost(ITEMS1) = 1,cost(ITEMS2) = 2,float = 10, item = {},shippingToStore = [],shippingToUser = {}, stock(ITEMS1) = 0,stock(ITEMS2) = 0,time = 0, user = []

This proves **Invariant 6** and **Invariant 7**, as with **Invariant 6** items tracked by the system (by the **costs** variable) are always a subset of the ITEM enumeration as items are only ever added to the store once. With **Invariant 7** whenever an item is added to the store via **cost,** the **stock** is set to the default value of **zero (0)** for the item. This means that whenever cost changes stock changes which means that the invariant is true.