1 Purpose of this assignment

This assignment extends assignment 3a to produce a refinement of the MessageBankS machine produced in 3a. In this assignment a new machine, MessageBankL, will be produced that should have the same functionality as MessageBankS, but the modelling mimics a list implementation such as might be done in a language such as C.

The assignment uses functions to simulate pointers and you will need to look carefully at B’s relational composition operations: iterate and closure.

It is important to appreciate that a principal purpose of this exercise is modelling so your model should be as easy to understand as possible.

2 An Overview

The model produced in MessageBankS used sequences and since a sequence is a monolithic structure the integrity of each customer’s queue of messages is ensured. However, if we were to implement MessageBankS, it could be argued that monolithic sequences are not efficient. This is seen especially in the DeleteMsg operation. A more efficient implementation strategy is to use linked lists, in which a successor to an item in the list is linked using a pointer. B does not have pointers, but we can model pointers using maplets and linked structures using relations/functions.

The scheme is represented by the following picture.

```
  firstmsg  nextmsg  nextmsg  ...  lastmsg
  o -->   * -->   * -->    ...    * -->   o

  :  firstmsg  nextmsg  nextmsg  lastmsg
  o -->   * -->   * -->    * -->   o

  :  firstmsg  nextmsg  nextmsg  lastmsg
  o -->   * -->   * -->    * -->   o

  :  firstmsg  nextmsg  nextmsg  lastmsg
  o -->   * -->   * -->    * -->   o

  o = customer  * = message
```
2.1 The ItemPool machine

An ItemPool machine that manages “items” is provided. This machine assigns a token to an item and has an operation \texttt{ItemVal} that “dereferences” a token and returns the value of the item.

Please notice that the \texttt{NewItem} operation is different to the normal form of this type of operation. Normally this operation would return a result that is the new token assigned to item. \texttt{NewItem} expects the caller to allocate the new token and pass it as an argument. The reason for this unconventional formulation is the lack of sequential composition at the specification level.

3 What you have to do

This assignment must continue to use the same development as used for the development of MessageBankS (ass3a).

The only role for MessageBankS in ass3b is in the refinement MessageBankSR. The primary role there will be in the writing of the "refinement relation" in the invariant of the machine.

The real basic challenge in ass3b is the writing of MessageBankL and in particular in the invariant of that machine.

If you do need any comments on your version of MessageBankS then please ask your tutor.

MessageBankS This remains as for ass3a.

MessageBankL the list version of the MessageBank. Part of this machine is sketched out in figure 3. The variables are shown with primitive invariants. The invariants are written here with relations; that is simply to indicate that we are talking about sets of pairs. A general relation will describe a graph and that will be too general for our purpose. \textit{You will have to decide} what sort of relation is appropriate in each case. Notice that msgcount is redundant, but it will be convenient to have a variable that contains the count of messages for each customer.

You need to:

- fill in the precondition and body of each operation;
- add extra constraints to the invariant that represent the “healthiness conditions” for the lists. An example of such a condition is:

\[
/*" the first message of a list is not the next message of any message "*/
!cid.(cid : customers & cid : dom(firstmsg) => (firstmsg(cid) /: ran(nextmsg)))
\]

You should add annotations, like that shown above, to explain to the reader informally what you are saying about your model.

Just remember that the text between /* and */ is LaTeX text and so some characters will have a special meaning. Any mathematical text should be inside $...$, but it probably would be best to avoid significant mathematical text as that follows in the formal expression.

Make sure that you use comments in /* ... */; which will be marked up, not source only comments /* ... */.

Notice that in the above I am using a device for obtaining indented annotations as follows:
If a line ends with an operation ( => in the above ) followed by (, then the remainder
will be indented until a line whose first character is ) matching the opening ), ie

some stuff operator (  
more stuff  
)  

If you can’t follow the above, please note that otherwise, line breaks in an expression
must occur at an operator.
These annotations will play an important part in the marking of the assignment, as
they will tell us what is intended by the formal invariant, even if you get it wrong.
• you may wish to add assertions. Assertions are properties that are implied by the
invariant. Assertions are expressed the same as the invariants. If you do not want
assertions, simply omit the assertions clause.
• produce an animation script for MessageBankL. This script should cover the same
scenario as for MessageBankS.

MessageBankSR. This is a refinement of MessageBankS and the exercise is to demonstrate that
MessageBankL is a refinement of MessageBankS, that is it takes MessageBankS towards
implementation. This is done as follows:

• MessageBankSR (the refinement of MessageBankS) is defined as an extension of Mes-
sageBankL as shown in figure 4. This part of the exercise is trivial.
• then a refinement relation that relates the state of MessageBankL to MessageBankS
is given in the invariant. One such relation is shown, but others are required. The
refinement relation describes the way in which MessageBankL simulates MessageBankS.
The given refinement relation:

\[
\text{messages}(cid)(pos) = \text{items}(\text{nextmsg}^{pos-1}(\text{firstmsg}(cid)))
\]
says that for each customer the message at position pos in the sequence model of the
queue of messages is the same as the message obtained by iterating nextmsg pos – 1
times and applying this to the first message in the list model.
Notice that since MessageBankSR has no state of its own there is no invariant required
to constrain that state.
• there are two parts to the demonstration of refinement:
  1. the completion of the refinement relation;
  2. the discharge of proof obligations. In this case the POs are concerned with showing
that MessageBankSR is a refinement of MessageBankS. This part of the exercise
is difficult and bonus marks will be awarded for significant progress towards its
completion. Bonus marks will also be awarded for significant attempts to discharge
the POs for MessageBankL.

Important: The discharge of POs for this assignment is hard and will account for
only 20% of the non-bonus marks for the assignment.
The important thing is getting the invariant and operations correct and describing
what you’re doing.
MACHINE ItemPool_CTX
SETS POOL
END

Figure 1: ItemPool_CTX.mch

MACHINE ItemPool ( ITEM , maxpool )
CONSTRAINTS maxpool ∈ \mathbb{N}_1
SEES ItemPool_CTX
VARIABLES items , pool
INVARIANT
    pool ⊆ POOL ∧
    items ∈ pool → ITEM
INITIALISATION pool , items := \{\} , \{}

OPERATIONS
   NewItem ( itemid , item ) ≡
    pre  item ∈ ITEM ∧ itemid ∈ POOL ∧ itemid ∈ POOL − pool
    then pool := pool ∪ \{ itemid \} ∥ items ( itemid ) := item
    end ;
   DeleteItem ( itemid ) ≡
    pre  itemid ∈ POOL ∧ itemid ∈ pool then
    pool := pool − \{ itemid \} ∥ items := \{ itemid \} ⊲ items
    end ;
   item ←− ItemVal ( itemid ) ≡
    pre  itemid ∈ POOL ∧ itemid ∈ pool then
    item := items ( itemid )
    end
END

Figure 2: ItemPool.mch
MACHINE MessageBankL ( MSG , maxcustomer , maxmessage )
CONSTRAINTS maxcustomer ∈ N₁ ∧ maxmessage ∈ N₁
SEES CUSTOMER_CTX , ItemPool_CTX
INCLUDES ItemPool ( MSG , maxcustomer × maxmessage )
VARIABLES customers ,
firstmsg ,
nextmsg ,
currentmsg ,
lastmsg ,
msgcount
INVARIANT customers ⊆ CUSTOMER ∧
firstmsg ∈ customers ↔ pool ∧
nextmsg ∈ pool ↔ pool ∧
currentmsg ∈ customers ↔ pool ∧
lastmsg ∈ customers ↔ pool ∧
msgcount ∈ customers → 0 .. maxmessage ∧

List healthiness conditions

ASSERTIONS

INITIALISATION

OPERATIONS

custid ←− NewCustomer ;
RemoveCustomer ( custid ) ;
NewMsg ( custid , msg ) ;
msg ←− FirstMsg ( custid ) ;
msg ←− CurrentMsg ( custid ) ;
msg ←− NextMsg ( custid ) ;
DeleteMsg ( custid )

END

Figure 3: MessageBankL.mch
REFINEMENT MessageBankSR
REFINES MessageBankS
SEES CUSTOMER_CTX, ItemPool_CTX
EXTENDS MessageBankL (MSG, maxcustomer, maxmessage)

INARIANT

Every msg at non-zero position pos in the sequence messages is at position pos in the list of messages

∀ (cid, pos) . (cid ∈ customers ∧ pos ∈ dom(messages(cid)) ∧ pos ≠ 0 ⇒

messages(cid)(pos) = items(nextmsg_{pos - 1}(firstmsg(cid))) ∧

Other refinement relations

END

Figure 4: MessageBankSR.ref