CS412/932: Formal systems development

Assignment 1

Due in (via Tabula) 12 noon Monday 13th November 2023

This is the first assignment for CS412. It's worth 20% of the overall mark. There are 3 questions and each carries equal marks. You should answer all 3 questions. This assignment should ideally be carried out in pairs. However, if you wish to do it on your own you may.

Pairs should work together through all the questions to arrive at a joint solution. The assignment is intended to help you understand the work and this will also enable you to do better in the final exam. The work should be that of your pair only and you should not share work between different pairs. Please complete and sign the cover sheet from the module website. The expectation is that you will have contributed equally. If for some reason this does not happen please contact me to discuss further.

I suggest you make a single directory for your work for this assignment (eg, *ass*1*dir*). If you wish to handwrite answers to any of the questions you can always scan them and include them in the directory too (obvious names please). Each person should then submit two files on Tabula:

- a scan of the completed group coversheet
- a zipped version of the directory with your solutions.

NB: for any of you unfamiliar with department machines: you can create a tar file of the whole directory with, eg:

```
        tar -cvf ass1dir.tar ass1dir
```

and gzip with:

```
        gzip ass1dir.tar ass1dir.tar.gz
```

Please do not convert B files - if I can't run them in the B tool I won't be able to mark them and they will receive a mark of 0. You should also ensure that you and your partner both submit the same thing (I'll only look at the first one I come across). You both need to submit to stop Tabula complaining. Where applicable, please use the machine names and identifier names suggested in the question to help me find and mark your answers. Obviously, you can choose additional names as required.

You should provide comments within your machines to explain what they are doing, detail any design decisions made, any problems encountered etc. If different approaches or interpretations are possible, comment your machine to explain what you have done and why. Additional comments (for example, justification for any unproved conditions for that machine) are welcome. Although formal proof of consistency is not required for the machines you write, examination of proof conditions and investigation of any unproved ones is a good way of ensuring you have a consistent machine. You may ignore B0 warnings.

Question 1

(a) You are dealt a hand of cards about which one of the following statements is true and the other is false:

- if there is a king in the hand, then there is an ace in the hand;
- it there isn't a king in the hand, then there is an ace in hand.

By formalising this using logical statements, show which (if any) of the statements below necessarily follow.

1. There is a king in the hand.
2. There is not a king in the hand.
3. There is an ace in the hand.
4. There is not an ace in the hand.

(b) Suppose $ss \in \operatorname{seq}\mathbb{N}$ and $nn \in \mathbb{N}$ and $ii \in 1\,..\,size(ss)$. Calculate and simplify:

$$[nn, ii := nn + ss(ii), ii + 1](nn = \Sigma xx \bullet (xx \in 1\,..\,ii - 1 \mid ss(xx)))$$

What do you notice about this example and where might this sort of situation arise?

(c)
```
MACHINE            Marks(mmax)
CONSTRAINTS        mmax: NAT1
VARIABLES          record
INVARIANT          record : 0..mmax --> NAT
INITIALISATION     record := {}
OPERATIONS
   addmark(mm) = PRE mm:NAT THEN record(mm) := record(mm)+1 END
END
```

The `Marks` machine is used to record marks obtained on a test for statistical purposes, with `mmax` representing the maximum obtainable mark. For example, if a student scores 60 then calling `addmark(60)` will increment the count for the number obtaining 60. (NB: information on *who* obtained the mark is not kept.) Write down the correctness requirements for initialisation and for an abstract machine operation. Use them to demonstrate the errors in this machine. Suggest corrections and show how the proofs succeed once your alterations are made.

(d) Define an operation which, for two sequences of similar type, decides whether or not the second is a contiguous subsequence of the first. Eg: if $ss = \langle 1, 5, 3, 2, 9, 8 \rangle$ and $tt = \langle 3, 2, 9 \rangle$ then tt is a subsequence of ss.

Define an operation which, for two sequences of similar type, decides whether or not the second is a general subsequence of the first. Eg: if $ss = \langle 1, 5, 9, 2, 9, 8 \rangle$ and $tt = \langle 5, 9, 9 \rangle$ then tt is a subsequence of ss. In this case, elements of tt occur in the correct order in ss but may be interspersed by other values.

Question 2

This question asks you to develop a B abstract machine from the stated requirements. Using proB will help you to investigate the behaviour of your machine. You should generate proof conditions and examine them in the prover to obtain a better understanding and to spot mistakes. However, for this question you don't have to fully prove all obligations, but should examine any conditions that are not readily provable to check that they really are true. Work in the prover to break down the goals if you can and check that they come down to steps which are clearly true. Otherwise, something is wrong and you need to modify your machine.

You are required to write a specification to help a company keep track of orders and stock control. The company receives orders to supply goods. Each order consists of a list of items required and the amount of each to be supplied, for example: "5 widgets and 3 thingummies". Orders are stored when received and will be dealt with on an "earliest received" basis if possible, however, there will be a limit on the backlog of orders that can be stored. The company holds a stock of the items it sells and from time to time will receive a delivery to replenish its stores and this may contain different amounts of various items. An order will only be serviced if it can be supplied in full, so if there are 10 widgets but no thingummies currently in stock, the example order given above could not be supplied at present. When an order is serviced it is removed from the wait list and the stock record is decremented as appropriate.

Write an abstract machine to describe this system and to provide the successful cases of the following operations (that is, you don't have to worry about error conditions).

- `receiveorder` to receive an order and add it to the wait list;
- `receivestock` to receive a delivery of stock (of possibly mixed items) and add it to the existing stock;
- `howmany` which, for a given item, *ii*, will output the amount of that item currently in stock;
- `whereisitem` which, for a given item, *ii*, will provide an output which identifies all waiting orders which include a request for *ii*;
- `totordered` which, for a given item, *ii*, will output the total amount of *ii* required by all orders currently waiting to be serviced;
- `lowstock` which, given a number, nn, will output all items for which the stock level has fallen below that amount;
- `oneoflowest` to output an item which currently has (or shares) the lowest stock level;
- `serviceorder` to service (if possible) the order which has been waiting longest.

It may be the case that the longest-waiting order cannot be serviced because there is insufficient stock, but another order can. Write a different version of a servicing operation, `complicatedserviceorder` which finds (if one exists) the longest-waiting order which can be supplied in full, and which services this order.

Question 3

A bag (or multiset) is a structure which records the number of times each element in the bag occurs. There is no ordering, but unlike a simple set, elements may occur in a bag more than once. The usual approach to modelling a bag is to regard it as a partial function mapping each item in the bag to the number of times it occurs in the bag. For example, if a shopping bag contains 4 tins of beans and a cucumber we could model it as:

$$\{beans \mapsto 4, cucumber \mapsto 1\}$$

If an item is not in the bag, it does not appear in the domain (that is, 0 is not in the range). Support for bags is not provided in the B toolkit. For this question you are asked to provide a machine to represent bags.

The abstract machine $Bag(ELEM, cap)$ is to provide the specification for a bag containing elements of type $ELEM$; cap is a number representing the maximum amount of any one element that can be stored in the bag. It should maintain the variable *bag* with a suitable type (consider the example above!). The *Bag* machine should provide the following operations:

name	**inputs**	**description**	**outputs**
addbag	*ee*	adds an $ee : ELEM$ to the bag	
rembag	*ee*	removes one $ee : ELEM$ from the bag	
countelm	*ee*	returns the number of $ee : ELEM$ in the bag (0 if none)	*oo*
totelms		returns the total number of elements in the bag (eg: 5 for the shopping bag example above)	*oo*
isin	*ee*	returns a Boolean to indicate whether or not $ee : ELEM$ is in the bag	*bb*
bagunion	*bb*	the input is itself a bag all of which is to be added to *bag*	
outputseq		the output is a sequence containing exactly the members of the bag (with correct multiplicity)	ss
addinseq	ss	ss is a sequence of ELEMs, all of which should be added to the bag	
totalcost		if $cost : ELEM \rightarrow NAT$ is added as another machine variable, *totalcost* outputs the total value of everything in the bag	cc

Write the specification for the abstract machine, *Bag*.

Once again, you should investigate your machine using the means at your disposal to ensure that the behaviour is as expected and that it is consistent.