COM2001 — Advanced Programming Topics

Exercise Sheet 4: Using ADTs (old exam question)

Spring Semester

Consider an ADT Q defined using a type parameter a, where $Q\ a$ represents a queue whose entries are of type a, whose syntax and semantics are defined as follows:

Sorts

$Q\ a$	queues whose entries are of type a
$Bool$	Boolean values
$MsgQ\ a$	messages associated with queues of type $Q\ a$
$MsgE\ a$	messages associated with entries of type a

Syntax

$msgNoQueue$	::	$MsgQ\ a$
$msgNoEntry$	::	$MsgE\ a$
$emptyQ$	::	$Q\ a$
$addEntry$	::	$a \to Q\ a \to Q\ a$
$removeEntry$	::	$Q\ a \to (Q\ a \cup MsgQ\ a)$
$frontEntry$	::	$Q\ a \to (a \cup MsgE\ a)$
$isEmpty$	::	$Q\ a \to Bool$

Semantics

$isEmpty\ emptyQ$	==	$True$	– (prop.1)
$removeEntry\ emptyQ$	==	$msgNoQueue$	– (prop.2)
$frontEntry\ emptyQ$	==	$msgNoEntry$	– (prop.3)
$isEmpty\ (addEntry\ x\ q)$	==	$False$	– (prop.4)
$removeEntry\ (addEntry\ x\ q)$	==	if $(isEmpty\ q)$	
		then q	– (prop.5)
		else $addEntry\ x\ (removeEntry\ q)$	– (prop.6)
$frontEntry\ (addEntry\ x\ q)$	==	if $(isEmpty\ q)$	
		then x	– (prop.7)
		else $frontEntry\ q$	– (prop.8)

Problem 1. Explain briefly what an Abstract Data Type (ADT) is, and why ADTs are useful in Software Engineering. [20]

Problem 2.

(i) Write down an expression using the operations $addEntry$, $removeEntry$ and $emptyQ$ to represent the queue that results from performing the following actions: Starting with an empty queue, the items a and b are added in that order, and then an item is removed. Next, item c is added, and a second item is removed. Finally, item d is added. [10]

(ii) Show in detail how the axioms of the ADT Q can be used to simplify the expression you gave for (i), and hence determine which item finishes up at the front of the queue. [20]

Problem 3.

(i) Write a Haskell module that implements the ADT Q by using lists of type `[a]` as the basic container for holding the queue's elements. Your code should include the following declaration, together with implementations of all the relevant functions:

```
data Q a = Q [a]
```

[25]

(ii) Prove that your implementation satisfies the semantic rules (prop.7) and (prop.8) of the Q ADT. [25]