

Solutions to Exercises 2.5, 2.12 and 2.14 of [1]

Aad Mathijssen

20th December 2004

We present solutions to exercises 2.5, 2.12 and 2.14 of [1]. In these solutions, we have chosen different names for the list operations than the ones that are proposed in [1], because these names were too operational. Additionally, the symmetry between pairs of operations is also expressed by their names.

Exercise 2.5

In the following, an algebraic specification of a list on an arbitrary non-empty domain D is given. Lists are constructed using the operations empty list ($[]$) and cons (\triangleright), i.e. an element at the head of a list. Also we provide the operations snoc (\triangleleft), i.e. an element at the tail of a list, left head ($lhead$), right head ($rhead$), left tail ($ltail$), right tail ($rtail$), the empty predicate ($isempty$), list length ($\#$), and list concatenation ($++$). First, we define the sorts and operations.

```
sort   List, D
func   []           :                → List
       _ ▷ _       : D × List → List
map    _ ◁ _       : List × D  → List
       lhead, rhead : List      → D
       ltail, rtail : List      → List
       isempty     : List      → Bool
       #-          : List      → Nat
       _ ++ _      : List × List → List
```

Next, we give properties for the non-constructor operations by defining equations. These equations are defined inductively on the structure of sort $List$, i.e. operations that have a list as an argument are defined in terms of the constructor operations.

```
var   d, e : D
      s : List
rew   [] ◁ e           = e ▷ []
      (d ▷ s) ◁ e     = d ▷ (s ◁ e)
      lhead(d ▷ s)    = d
      rhead(d ▷ [])   = d
      rhead(d ▷ (e ▷ s)) = rhead(e ▷ s)
      ltail(d ▷ s)    = s
      rtail(d ▷ [])   = []
      rtail(d ▷ (e ▷ s)) = d ▷ rtail(e ▷ s)
```

```

var    $d : D$ 
         $s, t : List$ 
rew    $isempty([]) = T$ 
         $isempty(d \triangleright s) = F$ 
         $\#[[]] = 0$ 
         $\#(d \triangleright s) = S(\#s)$ 
         $[] ++ t = t$ 
         $(d \triangleright s) ++ t = d \triangleright (s ++ t)$ 

```

Note that the head and tail operations are not defined on the empty list.

Exercise 2.12

A concatenation of a list s , from which the last element has been removed, with a list t , in which the last element of s is inserted, is expressed by the following term:

$$rtail(s) ++ (rhead(s) \triangleright t)$$

We need to prove:

$$\text{if } isempty(s) = F \text{ then } rtail(s) ++ (rhead(s) \triangleright t) = s ++ t, \quad (1)$$

where s, t are variables of sort *List*.

From the condition and the equations for *isempty*, terms that are substituted for s have to be of the form $d \triangleright u$, where d and u are terms of sort D and *List*, respectively. Then proof obligation (1) can be rephrased as

$$rtail(d \triangleright u) ++ (rhead(d \triangleright u) \triangleright t) = (d \triangleright u) ++ t, \quad (2)$$

where d is a variable of sort D and t, u are variables of sort *List*.

In order to prove (2), we need to do induction on a variable of sort *List*. The inference rule for induction on *List* is as follows. For all equations $\phi(s)$, where s is a variable of sort *List* occurring in ϕ :

$$(\text{induction on } List) \quad \frac{\phi([]) \quad \phi(d \triangleright s) \text{ where } \phi(s)}{\phi(s)}$$

Here, d is a variable of sort D , not occurring in ϕ . Also, the extra assumption $\phi(s)$ is called the *induction hypothesis*.

From the inductive structure of the equations for operations *rtail*, *rhead* and *++*, we can see that it is only profitable to do induction on variable u in order to prove (2). Then, by (induction on *List*), this follows from:

$$rtail(d \triangleright []) ++ (rhead(d \triangleright []) \triangleright t) = (d \triangleright []) ++ t \quad (3)$$

and

$$rtail(d \triangleright (e \triangleright u)) ++ (rhead(d \triangleright (e \triangleright u)) \triangleright t) = (d \triangleright (e \triangleright u)) ++ t, \quad (4)$$

where we may assume

$$rtail(d \triangleright u) ++ (rhead(d \triangleright u) \triangleright t) = (d \triangleright u) ++ t \quad (5)$$

as the induction hypothesis.

We give proofs of (3) and (4) in a *calculational style*. Also, we implicitly use congruence, substitution and symmetry. In order to prove (3), we derive the right hand side from the left hand side as follows:

$$\begin{aligned}
& rtail(d \triangleright []) ++ (rhead(d \triangleright []) \triangleright t) \\
= & \quad \{ \text{base case of } rtail \text{ and } rhead \} \\
& [] ++ (d \triangleright t) \\
= & \quad \{ \text{base case of } ++ \} \\
& d \triangleright t \\
= & \quad \{ \text{base case of } ++ \} \\
& d \triangleright ([] ++ t) \\
= & \quad \{ \text{inductive case of } ++ \} \\
& (d \triangleright []) ++ t
\end{aligned}$$

Then by transitivity, we have proven (3). Assuming (5), we derive for (4):

$$\begin{aligned}
& rtail(d \triangleright (e \triangleright u)) ++ (rhead(d \triangleright (e \triangleright u)) \triangleright t) \\
= & \quad \{ \text{inductive case of } rtail \text{ and } rhead \} \\
& (d \triangleright rtail(e \triangleright u)) ++ (rhead(e \triangleright u) \triangleright t) \\
= & \quad \{ \text{inductive case of } ++ \} \\
& d \triangleright (rtail(e \triangleright u) ++ (rhead(e \triangleright u) \triangleright t)) \\
= & \quad \{ \text{induction hypothesis (5)} \} \\
& d \triangleright ((e \triangleright u) ++ t) \\
= & \quad \{ \text{inductive case of } ++ \} \\
& (d \triangleright (e \triangleright u)) ++ t
\end{aligned}$$

Then by transitivity, we have proven (4). Hence, we proven (2) and (1).

Exercise 2.14

We need to prove that the empty list and a nonempty list are always different, i.e. we need to prove that

$$[] = d \triangleright s \tag{6}$$

does *not* hold. We do this by contradiction. Suppose (6) does hold. From the congruence rule, also

$$isempty([]) = isempty(d \triangleright s)$$

holds. Then from the equations of *isempty*, also

$$T = F$$

holds. But according to axiom *Bool1*, this does not hold. So have arrived at a contradiction. Hence, (6), i.e. the assumption, does not hold.

References

- [1] W.J. Fokkink, J.F. Groote, M.A. Reniers. *Modelling Distributed Systems*. Unpublished.