2IA05 Functional Programming

Authors
Etienne van Delden

March 12, 2012

Introduction

This is a summary for the course 2IA05: Functional Programming on the University of Technology Eindhoven. It was taught during semester 2A 2012.

1

General comments and notation

1.1 Proof by Induction

The Induction Hypothesis is assumed during a proof by Induction and does not necessarily have to be written down.

1.2 Finite lists

The set $\mathcal{L}_n(A)$ are all (finite) lists, with element type A, of length $n, n \geq 0$. Examples/definitions:

$$
\begin{array}{lcl}
\mathcal{L}_0(A) & = & \{[]\} \\
\mathcal{L}_{n+1}(A) & = & \{a \triangleright s \mid a \in A \wedge s \in \mathcal{L}_n(A)\} \\
\mathcal{L}_*(A) & = & (\biguplus n : n \geq 0 : \mathcal{L}_n(A))
\end{array}
$$

2

Functions

2.1 ``Map'': $\bullet$

The function ``map'', denoted as $\bullet$, is the function that applies a function f on each element of a list, with a function $f : A \to B$. It is specified as:

$$(f\bullet) : \mathcal{L}_*(A) \to \mathcal{L}_*(B)$$

with

$$(\forall i : 0 \leq i < n : (f \bullet s) \cdot i = f \cdot (s \cdot i))$$

for $s \in \mathcal{L}_*(A)$, with $\#s = n$ and $\#(f \bullet s) = \#s$. Let's now derive the $\bullet$-function.

TODO: put the following on the right spot

$$f \bullet [] = []$$

The definition follows from the type of the $\bullet$-function.

Base

For $a \in A$, $s \in \mathcal{L}_n(A)$, we have the following base case $i = 0$;

$$\begin{array}{lll} & (f \bullet (a \triangleright s)) \cdot 0 & \text{specification}\bullet \\ = & f \cdot ((a \triangleright s) \cdot 0) & \text{property} \triangleright \\ = & f \cdot a & \triangleright\text{-trick} \\ = & (f \cdot a \triangleright \mathbf{?}) \cdot 0 & \end{array}$$

Induction Step

For $i : 0 \leq i < n$;

$$\begin{array}{ll} & (f \bullet (a \rhd s)) \cdot (i+1) \qquad \text{specification}\bullet \\ = & f \cdot ((a \rhd s) \cdot (i+1)) \qquad \text{property} \rhd \\ = & f \cdot (s \cdot i) \qquad \text{specification} \bullet\text{, I.H} \\ = & (f \bullet s) \cdot i \qquad \rhd\text{-trick} \\ = & (\mathbf{?} \rhd f \bullet s) \cdot (i+1) \end{array}$$

Result

To get the final result, we can combine the two derived results and fill in the ``Don't cares'' (**?**):

$$\begin{array}{ll} & \text{specification } f\bullet \qquad \text{Combination} \\ \Leftarrow & (\forall i : 0 \leq i < n+1 : \\ & \quad (f \bullet (a \rhd s)) \cdot i = \\ & \quad (f \cdot a \rhd f \bullet s) \cdot i) \qquad \text{Leibniz (extensionality)} \\ \Leftarrow & f \bullet (a \rhd s) = (f \cdot a) \rhd f \bullet s \end{array}$$

Thus, the definition of ``$\bullet$'' is:

$$\begin{array}{lcl} f \bullet [] & = & [] \\ f \bullet (a \rhd s) & = & f \cdot a \rhd f \bullet s \end{array}$$

Lemma's Used

$$\begin{array}{lcl} (a \rhd s) \cdot 0 & = & a \\ (a \rhd s) \cdot (i+1) & = & s \cdot i, i \geq 0 \end{array}$$

2.2 ``Reverse'': rev

The function rev reverses the order of all elements in a list. The type of rev is

$$\mathcal{L}_n(A) \rightarrow \mathcal{L}_n(a) \qquad n \geq 0$$

with specification

$$(\forall i : 0 \leq i \leq n : rev \cdot s \cdot i = s \cdot (n - i))$$

for all $s \in \mathcal{L}_{n+1}(A)$.

We can now derive the definition of rev, by using Induction. From the type of rev it follows that:

$$rev \cdot [] = []$$

Base

$$\begin{array}{lll} & rev \cdot (a \triangleright s) \cdot n & \text{specification } rev \\ = & (a \triangleright s) \cdot 0 & \text{property } \triangleright \\ = & a & \triangleright\text{-trick} \\ = & [a] \cdot 0 & \text{property } +\!\!+ \text{ ; assume} \#t = n \\ = & (t\#[a]) \cdot n & \end{array}$$

As we start counting at 0, this is correct.

Step

For $s \in \mathcal{L}_N(A)$ and $a \in A$.

$$\begin{array}{lll} & rev \cdot (a \triangleright s) \cdot i & \text{specification } rev \\ = & (a \triangleright s) \cdot (n - i) & \text{assume } i < n, \text{ property } \triangleright \\ = & s \cdot (n - 1 - i) & \text{Ind. Hypothesis, specification } rev \\ = & rev \cdot s \cdot i & textrmproperty +\!\!+ \\ = & (rev \cdot s +\!\!+ \; ?) \cdot i & \end{array}$$

Result

We can now combine the results from the Complete Induction derivation to form the definition of rev :

$$\begin{array}{lcl} rev \cdot [] & = & [] \\ rev \cdot (a \triangleright s) & = & rev \cdot s +\!\!+ [a] \\ rev \cdot s & = & s \cdot n \triangleright rev \cdot (s \lceil n) \end{array}$$

This solution runs in $O(n^2)$ time complexity. This can be improved by using a general function that has an accumalator list in which the answer is built. See Section

3

Generalisation by Abstraction

3.1 ``Reverse" $grev$

The function $grev$ is a generalized version of function rev (as defined in Section). It uses an extra parameter $t \in \mathcal{L}_n(A)$ in which the result list is built. We specify $grev$ as follows:

$$grev \cdot t \cdot s \quad = \quad rev \cdot \mathbin{+\!\!+} \; t$$

Use:

$$rev \cdot s \quad = \quad grev \cdot [] \cdot s$$

Base case

$$\begin{array}{lll} & grev \cdot t \cdot [] & \text{specification } grev \\ = & rev \cdot [] \mathbin{+\!\!+} t & \text{property } rev \\ = & [] \mathbin{+\!\!+} t & \text{property } \mathbin{+\!\!+} \\ = & t & \end{array}$$

Induction step

$$\begin{array}{lll} & grev \cdot t \cdot (a \triangleright s) & \text{specification } grev \\ = & rev \cdot (a \triangleright s) \mathbin{+\!\!+} t & \text{property } rev \\ = & & \end{array}$$

4

Divide and Conquer