Computer Science 15-212, Spring 2009
Recitation 6

Straight recursion versus Tail recursion

Type	$reverse : \ 'a\ list \ \rightarrow \ 'a\ list$
Description	returns the reverse of a list given as argument.
Example	$reverse\ ([1,2,3,4,5]) \ \Longrightarrow \ [5,4,3,2,1]$

Straight Recursion

$$\text{Definition} \begin{cases} reverse(nil) & = \ nil \\ reverse(n :: l) & = \ reverse(l)@[e] \end{cases}$$

Tail Recursion

$$\text{Definition} \quad reverse'(l) = rvrs(l, nil) \text{ and } \begin{cases} rvrs(nil, acc) & = \ acc \\ rvrs(e :: l, acc) & = \ rvrs(l, e :: acc) \end{cases}$$

Proving program equivalence using structural induction

First attempt

Property: $\forall\ l :'\ a\ list\ reverse(l) \ = \ rvrs(l, nil)$

By structural induction on l,

Base Case: $l = nil$

$$\begin{array}{lrcl} \text{by definition of } reverse & reverse(nil) & = & nil \\ \text{by definition of } rvrs & rvrs(nil, nil) & = & nil \\ \text{and so} & reverse(nil) & = & rvrs(nil, nil) \end{array}$$

Induction Step: $l = e :: l'$

We want to prove that: if $reverse(e :: l') = rvrs(e :: l', nil)$

Induction Hypothesis: $reverse(l') = rvrs(l', nil)$

$$\begin{array}{lrcl} \text{by definition of } reverse & reverse(e :: l') & = & reverse(l')@[e] \\ \text{by IH} & & = & rvrs(l', nil)@[e] \end{array}$$

At this point we are stuck, we cannot go further. So, let us try to work on the right-hand side.

$$\text{by definition of } rvrs \quad rvrs(e :: l', nil) \ = \ rvrs(l', e :: nil)$$

At this point, we are stuck again! We cannot apply induction hypothesis because the second argument of $rvrs$ is not nil but some other list that may be arbitrary. This proof attempt has failed.

Second attempt

We see that we would need a weaker property to be able to apply the IH in case the second argument of $rvrs$ is any kind of list acc. In this case, the property becomes:

Property: $\forall\ l :'\ a\ list\ reverse(l)@acc\ =\ rvrs(l, acc)$

By structural induction on l,

Base Case: $l = nil$

by definition of $reverse$	$reverse(nil)@acc$	$=$	$nil@acc$
by definition of $reverse$	$reverse(nil)@acc$	$=$	acc
by definition of $rvrs$	$rvrs(nil, acc)$	$=$	acc
and so	$reverse(nil)@acc$	$=$	$rvrs(nil, acc)$

Induction Step: $l = e :: l'$

We want to prove that: if $reverse(e :: l')@acc = rvrs(e :: l', acc)$

Induction Hypothesis: $reverse(l')@acc = rvrs(l', acc)$

by definition of $reverse$	$reverse(e :: l')@acc$	$=$	$(reverse(l')@[e])@acc$
by associativity of @		$=$	$reverse(l')@([e]@acc)$
by definition of @		$=$	$reverse(l')@(e :: acc)$
by IH		$=$	$rvrs(l', e :: acc)$
by definition of $rvrs$		$=$	$rvrs(e :: l', acc)$
QED			

Notice that here we would need to prove the associativity lemma for @, i.e. $(l1@l2)@l3\ =\ l1@(l2@l3)$. I leave it to you an exercice.