CSci 2041 - Advanced Programming Principles

Spring 2017

Homework 3

Name: Tiannan Zhou

Email: zhou0745@umn.edu ID:5232494

Question 1:

We would like to show that

$$\forall n: power\ n\ x = x^n$$

The principle of induction for natural numbers is

$$\forall n, P(0)\ and\ P(n) \Longrightarrow P(n+1)$$

In this case, our property P is defined as

$$P(n)\ \text{ is }\ power\ n\ x = x^n$$

Our induction proof would have two cases:

- $P(0)$: show that $power\ 0\ x = x^0 = 1$

  $power\ 0\ x$
  $= 1.0$ by definition of $power$

- $P(n+1)$: show that $power\ (n+1)\ x = x^{n+1}$ where $power\ n\ x = x^n$ holds by the inductive hypothesis.

  $power\ (n+1)\ x$
  $= x \times power\ (n+1-1)\ x$ by definition of $power$
  $= x \times power\ n\ x$ by property of addition and subtraction
  $= x \times x^n$ by inductive hypothesis
  $= x^{n+1}$ by property of multiplication and power

$P(n)$ has been proven.

Question 2:

We would like to show that

$$\forall n \in nat, power\ n\ x = x^{toInt(n)}$$

The principle of induction for this type nat is

$$\forall n \in nat, P(Zero)\ and\ P(n) \Longrightarrow P(Succ\ n)$$

In this case, our property P is defined as

$$P(n)\ \text{ is }\ power\ n\ x = x^{toInt(n)}$$

Our induction proof would have two cases:

- $P(Zero)$: show that $power\ Zero\ x = x^{toInt(Zero)}$

  $power\ Zero\ x$
  $= 1$ by definition of $power$
  $= x^0$ by definition of power
  $= x^{toInt\ (Zero)}$ by definition of $toInt$

- $P(Succ\ n)$: show that $power\ (Succ\ n)\ x = x^{toInt(Succ\ n)}$ when $power\ n\ x = x^{toInt(n)}$ holds by the inductive hypothesis.

$$\begin{aligned} & power\ (Succ\ n)\ x \\ &= x \times power\ n\ x \text{ by the definition of } power \\ &= x \times x^{toInt(n)} \text{ by induction hypothesis} \\ &= x^{toInt(n)+1} \text{ by the definition of multiplication and power} \\ &= x^{toInt(Succ\ n)} \text{ by the definition of } toInt \end{aligned}$$

$P(n)$ has been proven.

Question 3:

We would like to show that

$$length\ (l\ @\ r) = length\ l + length\ r$$

The principle of induction for list is

$$\forall l \in 'a\ list, P([\,]) \text{ and } P(l) \Longrightarrow P(v :: l)$$

In this case, our property $P(l)$ is defined as

$$\forall r \in 'a\ list, length\ (l\ @\ r) = length\ l + length\ r$$

Our induction proof would have two cases:

- $P([\,]): \forall r \in 'a\ list, length([\,]\ @\ r) = length\ [\,] + length\ r$

$$\begin{aligned} & \forall r \in 'a\ list, (\\ & \quad length([\,]\ @\ r) \\ & = length\ r \text{ by properties of lists} \\ & = 0 + length\ r \text{ by properties of addition} \\ & = length\ [\,] + length\ r \text{ by definition of } length \\ &) \end{aligned}$$

- $P(x :: xs): \forall r \in 'a\ list, length\ (x :: xs\ @\ r) = length(x :: xs) + length\ r$

$$\begin{aligned} & \forall r \in 'a\ list, (\\ & \quad length(x :: xs\ @\ r) \\ & = 1 + length\ (xs\ @\ r) \text{ by definition of } length \\ & = 1 + length\ xs + length\ r \text{ by induction hypothesis} \\ & = length(x :: xs) + length\ r \text{ by definition of } length \\ &) \end{aligned}$$

$P(l)$ has been proven.

Question 4:

We would like to show that

$$length(reverse\ l) = length\ l$$

The principle of induction for list is

$$\forall l \in 'a\ list, P([\,]) \text{ and } P(l) \Longrightarrow P(v :: l)$$

In this case, our property $P(l)$ is defined as

$$P(l)\ is\ length(\ reverse\ l\ ) = length\ l$$

Our induction proof would have two cases:

- $P([\,]): length(\ reverse\ [\,]) = length\ [\,]$

$$\begin{aligned} & length(reverse\ [\,]) \\ &= length\ [\,] \text{ by definition of } reverse \end{aligned}$$

- $P(x :: xs)$: $length(\ reverse\ x :: xs) = length(x :: xs)$

  $length(\ reverse\ x :: xs\ )$

  $= length(\ reverse\ xs\ @\ [x]\ )$ by the definition of $reverse$

  $= length(reverse\ xs) + length\ [x]$ by the property of $length$ proven in Question 3

  $= length\ xs + length\ [x]$ by induction hypothesis

  $= length\ xs + 1$ by definition of $length$

  $= 1 + length\ xs$ by property of addition

  $= length(x :: xs)$ by definition of $length$

$P(l)$ has been proven.

Question 5:

The principle of induction for list is

$$\forall l \in 'a\ list\ , P([\,]) \ and\ P(l) \Longrightarrow P(v :: l)$$

We would like to prove $P'(l)$ first while

$P'(l)$ is $\forall r \in 'a\ list, append\ l\ r = l\ @\ r$

Our induction proof for $P'(l)$ would have two cases:

- $P'([\,])$: $\forall r \in 'a\ list, append\ [\,]\ r = [\,]\ @\ r$

  $\forall r \in 'a\ list$, (

  $append\ [\,]\ r$

  $= r$ by definition of $append$

  $= [\,]\ @\ r$ by properties of lists

)

- $P'(x :: xs)$: $\forall r \in 'a\ list, append\ (x :: xs)\ r = (x :: xs)\ @\ r$

  $\forall r \in 'a\ list$, (

  $append\ (x :: xs)\ r$

  $= x :: (append\ xs\ r)$ by definition of $append$

  $= x :: (xs\ @\ r)$ by induction hypothesis

  $= (x :: xs)\ @\ r$ by properties of lists

)

$P'(l)$ has been proven.

Now we would like to prove $P(l1)$ while

$P(l1)$ is $\forall l2 \in 'a\ list, reverse(append\ l1\ l2) = append\ (reverse\ l2)\ (reverse\ l1)$

Our induction proof for $P(l1)$ would have two cases:

- $P([\,])$: $\forall l2 \in 'a\ list, reverse(append\ [\,]\ l2) = append\ (reverse\ l2)\ (reverse\ [\,])$

  $\forall l2 \in 'a\ list$, (

  $reverse(append\ [\,]\ l2)$

  $= reverse([\,]\ @\ l2)$ by property of $append$ proven by $P'(l)$

  $= reverse(l2)$ by properties of lists

  $= reverse(l2)\ @\ [\,]$ by properties of lists

  $= reverse(l2)\ @\ reverse([\,])$ by definition of $reverse$

  $= append\ (reverse\ l2)\ reverse([\,])$ by property of $append$ proven by $P'(l)$

)

- $P(x :: xs)$: $\forall l2 \in 'a\ list, reverse(append\ (x :: xs)\ l2) = append\ (reverse\ l2)\ (reverse\ (x :: xs))$

  $\forall l2 \in 'a\ list$, (

  $reverse(append\ (x :: xs)\ l2)$

$= reverse((x :: xs)\ @\ l2)$ by property of $append$ proven by $P'(l)$

$= reverse\big(x :: (xs\ @\ l2)\big)$ by properties of lists

$= reverse(xs\ @\ l2)\ @\ [x]$ by definition of $reverse$

$= reverse(append\ xs\ l2)\ @\ [x]$ by property of $append$ proven by $P'(l)$

$= append\ (reverse\ l2)\ (reverse\ xs)\ @\ [x]$ by induction hypothesis

$= (reverse\ l2)\ @\ (reverse\ xs)\ @\ [x]$ by property of $append$ proven by $P'(l)$

$= (reverse\ l2)\ @\ (reverse\ xs\ @\ [x])$ by property of lists

$= (reverse\ l2)\ @\ \big(reverse\ (x :: xs)\big)$ by definition of $reverse$

$= append\ (reverse\ l2)\ \big(reverse\ (x :: xs)\big)$ by property proven by $P'(l)$

)

$P(l1)$ has been proved.

Question 6:

We would like to show

$$sorted\ l => sorted\ (place\ e\ l)$$

The principle of induction for list is

$$\forall l \in' a\ list\ , P([\,])\ and\ P([x])\ and\ P(l) \Longrightarrow P(v :: l)$$

In this case, our property $P(l)$ is defined as

$$P(l)\ is\ that\ sorted\ l \Longrightarrow sorted\ (place\ e\ l)\ is\ true$$

We just need to prove the situation that B is also true when A is true in order to prove that $A \Longrightarrow B$ is true (by property of implication). Our induction proof lists below:

Base case:

$P([\,])$: to prove $sorted\ [\,] \Longrightarrow sorted(place\ e\ [\,])\ is\ true$

$sorted\ [\,]$

$= true$ by definition of $sorted$

$= sorted\ e :: [\,]$ by definition of sorted

$= sorted\ [e]$ by property of lists

$= sorted(place\ e\ [\,])$ by definition of $place$

Thus, $sorted\ [\,] \Longrightarrow sorted(place\ e\ [\,])$ is true, P([ ]) has been proven.

$P([x])$: to prove $sorted\ [x] \Longrightarrow sorted(place\ e\ [x])$ is $true$

$sorted\ [x]$

$= true$ by definition of $sorted$

Case 1: $e < x$

$sorted(place\ e\ [x])$

$= sorted(e :: x)$ by definition of $place$

$= e \leq x \wedge sorted([x])$ by definition of $sorted$

$= true \wedge sorted([x])$ by $e < x$

$= true \wedge true$ by definition of $sorted$

$= true$ by property of Boolean expression

Case 2: $e \geq x$

$sorted(place\ e\ [x])$

$= sorted\big(x :: (place\ e\ [\,])\big)$ by definition of $place$

$= sorted(x :: [e])$ by definition of $place$
$= sorted(x :: e :: [\,])$ by property of lists
$= x \leq e \wedge sorted\ [e]$ by definition of $sorted$
$= \text{true} \wedge \text{sorted [e]}$ by $e \geq x$
$= true \wedge true$ by definition of $sorted$
$= true$ by property of Boolean expressions

Thus, $sorted\ [x] \Longrightarrow sorted(place\ e\ [x])$ is true, P([x]) has been proven.g

Inductive case:

$P(x :: v :: xs)$: to prove the statement $sorted\ x :: v :: xs \Longrightarrow sorted\ \big(place\ e\ (x :: v :: xs)\big)\ is\ true$

We assume $x \leq v$ to make sure $sorted\ x :: v :: xs$ can be true. (We don't care the situation when $sorted\ x :: v :: xs$ is false because the implication statement would always be true if the left-hand side is false (by property of Propositional Logic)) Thus, my proof below would just show the cases under condition that the left-hand is always true.

Case 1: $e < x$

Left hand: $sorted\ x :: v :: xs$
$= true \wedge sorted(x :: v :: xs)$ by property of Boolean expression
$= e \leq x \wedge sorted(x :: v :: xs)$ by $e < x$
$= sorted(e :: x :: v :: xs)$ by definition of $sorted$
$= sorted\big(place\ e\ (x :: v :: xs)\big)$ by definition of $place$

Thus, $sorted\ x :: v :: xs \Longrightarrow sorted\ \big(place\ e\ (x :: v :: xs)\big)$ is true under the condition of $e < x$

Case 2: $x \leq e$

Right hand: $sorted\big(place\ e\ (x :: v :: xs)\big)$

$= sorted\ \Big(x :: place\big(e\ (v :: xs)\big)\Big)$ by definition of place

$= x \leq \min(e, v) \wedge sorted\ \Big(place\big(e\ (v :: xs)\big)\Big)$ by definition of $sorted$ and assumption

$= true \wedge sorted\ \Big(place\big(e\ (v :: xs)\big)\Big)$ by $x \leq e\ and\ x \leq v$

$= true \wedge true$ by inductive hypothesis and assumption
$= true$ by property of Boolean expressions

Thus, $P(l)$ has been proven.

Question 7:

1. Since the function $place$ would place the element e into the list l where it found the first element currently in list l which is greater than x. Thus, in the result list from function $place\ e\ l$, all the elements on the left of new inserted element e are smaller than e. Therefore, function $is\_elem\ e\ (place\ e\ l)$'s assumption does not matter in this scenario due to the premise mentioned above made by the function $place$. Function is_elem would skip all the elements on the left of the target element e (since they are smaller than the element we're looking for), then find e and return true.

2. It doesn't need. We can directly assume $sorted\ l$ is true when we prove the statement. Because of the

property of implication statement that when the left-hand is false, the full single-direction implication statement is always true, we just need to prove the situation that the right-hand side is true as well while the left-hand side is true to make sure the full statement is true. The situation of left-hand side being false does not matter.