Wir sehen $l_1 @ l_2$ = append l_1 l_2 nach Def.; Somit gilt Lemma 2 auch für äquivalente @-Aufrufe

1) Induktion über n:= length l und eine beliebe Anzahl von Elementen für a.

1. Basis: n = 0, also l = [], dann gilt

reverse (fold-left (fun a x → f x :: a) [a] []) $\overset{\text{Def. fold}}{=}$ reverse [a]

$\overset{\text{Lemma 1}}{=}$ reverse [a] @ [] $\overset{\text{Def. map}}{=}$ reverse [a] @ (map f [])

1. Schritt: Sei $n \in \mathbb{N}_0$ beliebig fixiert und l eine beliebige Liste mit length l = n und x ein Element, sodass length (x::l) = n+1

1. I.H.: Die Aussage gelte für l =: xs

1. Behauptung: Die Aussage gilt dann auch für (x::l)

Beweis: reverse (fold_left (fun a x → f x :: a) [a] (x::xs))

$\overset{\text{Def. fold}}{=}$ reverse (—— " —— [f x; a] xs)

$\overset{\text{I.H.}}{=}$ reverse [f x; a] @ map f xs

$\overset{\text{Def rev}}{=}$ (reverse [a] @ [f x]) @ map f xs

$\overset{\text{Lemma 2}}{=}$ reverse [a] @ ([f x] @ map f xs)

$\overset{\text{Def @}}{=}$ —— " —— (f x :: [] @ map f xs)

$\overset{\text{Def @}}{=}$ —— " —— (f x :: map f xs)

$\overset{\text{Def map}}{=}$ reverse [a] @ map f (x::xs) □

2) reverse (fold-left (fun a x → f x :: a) [] l)

$\overset{\text{Aufgabe 1}}{=}$ reverse [] @ map f l $\overset{\text{Def. rev}}{=}$ [] @ map f l $\overset{\text{Def @}}{=}$ map f l □

3) Wir führen Induktion über n:= length l und beliebige $i \in \mathbb{Z}$. Der Rest erfolgt analog zu
Wir zeigen nur den Beweis des 1. Schritts.
allen anderen Induktionsbeweisen (übersetzt: Kevin ist zu faul für Formalismus;)

<Formalismus>

⋮

Beweis: fold_left (fun a x → x+a) i (x::xs)

$\overset{\text{Def. fold}}{=}$ —— " —— (x+i) xs

$\overset{\text{I.H.}}{=}$ sum (x+i) xs $\overset{\text{Def. sum}}{=}$ sum i (x::xs)

Beachte: Einsetzen der I.H. nur möglich, da i beliebig im Beweis. Hätte man etwa fold_left (fun a x → x+a) 0 l = sum 0 l zeigen sollen, hätte man zuerst Aufgabe 3 beweisen müssen.