Scheme Theory

COMP105 Fall 2015

James McCants

Problem 31

Prove that: (length (reverse xs)) = (length xs)
Proof by structural induction on xs.

Base case where xs is '():

(length (reverse '()))

`= {substitute actual param in reverse}`

(length (if (null? `()) '() (append (reverse (cdr `())) (list1 (car '()))))))

`= {null?=empty law}`

(length (if #t `() (append (reverse (cdr '())) (list1 (car '()))))))

`= {if-#t law}`

(length '())

Induction step, assume that xs is not nil, xs = (cons y ys):

(length (reverse xs))

`= {by assumption that xs is nut nil, xs = (cons y ys)}`

(length (reverse (cons y ys)))

`= {substitute actual params into def of reverse}`

(length (if (null? (cons y ys)) (cons y ys) (append (reverse (cdr (cons y ys))) (list1 (car (cons y ys))))))

`= {null?-cons law}`

(length (if #f (cons y ys) (append (reverse (cdr (cons y ys))) (list1 (car (cons y ys))))))

`= {if-#f law}`

(length (append (reverse (cdr (cons y ys))) (list1 (car (cons y ys)))))

`= {car-cons law}`

(length (append (reverse (cdr (cons y ys))) (list1 y)))

`= {cdr-cons law}`

(length (append (reverse ys) (list1 y)))

`= {append law}`

(length (+ (length (reverse ys)) (length (list1 y))))

`= {induction hypothesis}`

(length (+ (length ys) (length (list1 y))))

`= {list1-rule}`

(length (+ (length ys) (length y)))

`= {length-cons law}`

(length (cons y ys))

`= {assumption that (cons y ys) = xs}`

(lenth xs)