Proof Problems

Exercise 22:

Laws for Append:

append-empty law: (append '() ys) == ys

append-cons law: (append (cons z zs) ys) == (cons z (append zs ys))

Claim: (append (append xs ys) zs) == (append xs (append ys zs))

Proof by Equational Reasoning:

Case 1: append-empty

Given xs = '()
(append xs (append ys zs)) ;; RHS
= {by assumption, xs = '()}
(append '() (append ys zs))
= {append-empty law}
(append ys zs)
= {reverse append-empty law}
(append (append '() ys) zs)
= { by assumption, xs = '()}
(append (append xs ys) zs) ;; LHS
#t

Case 2: append-cons

Given xs = (cons n ns)
(append xs (append ys zs)) ;; RHS
= {by assumption, xs = (cons n ns)}
(append (cons n ns) (append ys zs))
= {append-cons law}
(cons n (append ns (append ys zs)))
= {apply inductive hypothesis}

(cons n (append (append ns ys) zs))
= {reverse append-cons law}
(append (cons n (append ns ys)) zs)
= {reverse append-cons law}
(append (append (cons n ns) ys) zs)
= {by assumption, xs = (cons n ns)}
(append (append xs ys) zs) ;; LHS
#t

Exercise A:

a)

Proof: In order to prove the (cdr (cons x xs)) == xs, we can construct a formal derivation tree.

Derivation Tree:

$$\dfrac{\begin{array}{c}\langle \text{ e}, \rho, \sigma_1\rangle \Downarrow \langle \text{ PRIMITIVE(cons)}, \sigma_2\rangle \\ \dfrac{x \in \text{dom } \rho \quad \rho(x) \in \text{dom } \sigma}{\langle VAR(x), \rho, \sigma_2\rangle \Downarrow \langle \sigma_2(\rho(x)), \sigma_2\rangle}\text{ VAR} \\ \dfrac{xs \in \text{dom } \rho \quad \rho(xs) \in \text{dom } \sigma}{\langle VAR(xs), \rho, \sigma_2\rangle \Downarrow \langle \sigma_2(\rho(xs)), \sigma_2\rangle}\text{ VAR} \\ l_1 \notin \text{dom } \sigma_3 \quad l_2 \notin \text{dom } \sigma_3 \quad l_1 \neq l_2\end{array}}{\langle AP(e, VAR(x), VAR(xs)), \rho, \sigma_0\rangle \Downarrow \langle PR\langle l_1, l_2\rangle, \sigma_2\{l_1 \mapsto \sigma_2(\rho(x)), l_2 \mapsto \sigma_2(\rho(xs))\}\rangle}\text{ CONS}$$

$$\dfrac{\begin{array}{c}\langle \text{ e}, \rho, \sigma_0 \rangle \Downarrow \langle \text{ PRIMITIVE(cdr)}, \sigma_1\rangle \\ \langle \text{(cons VAR(x) VAR(xs))}, \rho, \sigma_1\rangle \Downarrow \langle PR\langle l_1, l_2\rangle, \sigma_2\rangle\end{array}}{\langle AP(\text{e, (cons x xs)}), \rho, \sigma_0\rangle \Downarrow \langle xs, \sigma_2\rangle}\text{ CDR}$$

Key: AP = APPLY, PR = PAIR

Explanation: Since we needed to prove that (cdr (cons x xs)) == xs, the basis for my derivation tree were the rules for CONS and CDR.

By order of operations, cons x xs should be evaluated first. Thus, I stacked the CONS rule on top of the CDR rule. For my cons derivation, I replaced e_1 and e_2 with VAR(x) and VAR(xs) respectively and derived their values with the VAR rule. Finally, the evaluation of the expression APPLY(e, VAR(x), VAR(xs)) produced the value PAIR $\langle l_1, l_2\rangle$ and the store $\sigma_2\{l_1 \mapsto \sigma_2(\rho(x)), l_2 \mapsto \sigma_2(\rho(xs))\}\rangle$. Also it should be noted that I had the evaluation of e in σ_1 instead of σ_0 because σ_0 will be solely for cdr.

Finally, using the CDR rule, I evaluated APPLY(cdr (cons x xs)) to produce the value xs with the store σ_2.

b)

The purpose of this part is to show that when cons is applied to general expressions, the two sides aren't always equal.

Imagine that we defined some global variable x that is equal to 4 (val x 4).

Now, imagine that we have two terminating expressions e_1 = (set x (+ x 1)) and e_2 = x. It is very clear that the evaluation of (cdr (cons e_1 e_2)) terminates as well.

Before (cdr (cons e_1 e_2)) fully evaluates, e_2 originally evaluates to 4. However, once both e_1 and e_2 are evaluated, the global variable x is incremented by 1. Thus, evaluating (cdr (cons e_1 e_2)) produces the value 5 instead of 4 (e_2).

Therefore, we have provided a case in which the two sides are not equal.

Extra Credit Problems

Exercise TDP:

My laws for take and drop:

take-empty: (take n '()) == '()

take-none: (take 0 (cons x xs)) == '()

take-cons: (take n (cons x xs)) == (cons x (take (- n 1) xs)) where n != 0

drop-empty: (drop n '()) == '()

drop-none: (drop 0 (cons x xs)) == xs

drop-cons: (drop n (cons x xs)) == (drop (- n 1) xs)

Claim: (append (take n xs) (drop n xs)) == xs

Proof by Equational Reasoning:

Case 1: take-empty/drop-empty

Given xs = '()
(append (take n xs) (drop n xs)) ;; LHS
= {by assumption, xs = '()}
(append (take n '()) (drop n '()))
= {take-empty and drop-empty laws}
(append '() '())
= {by appending rules}
'()
= {by assumption, xs = '()}
xs ;; RHS
#t

Case 2: take-none/drop-none

Given n = 0
(append (take n xs) (drop n xs)) ;; LHS
= {by assumption, n = 0}

(append (take 0 xs) (drop 0 xs))
= {take-none and drop-none laws}
(append '() xs)
= {by appending rules}
xs ;; RHS
#t

Case 3: take-cons/drop-cons

Given xs = (cons y ys)
(append (take n xs) (drop n xs)) ;; LHS
= {by assumption, xs = (cons y ys)}
(append (take n (cons y ys)) (drop n (cons y ys)))
= {take-cons and drop-cons law}
(append (cons y (take (- n 1) ys)) (drop (- n 1) ys))
= {append cons law from exercise 22}
(cons y (append (take (- n 1) ys) (drop (- n 1) ys)))
= {apply inductive hypothesis}
(cons y ys)
= {by assumption, xs = (cons y ys)}
xs ;; RHS
#t