PDF Problems

Exercise 4:

Formation Rules

Formation: array formation

$$\frac{\tau \text{ is a type}}{LIST(\tau) \text{ is a type}} \text{(LISTFORMATION)}$$

Explanation: A formation rule answers the following question: what types can I make. A list type is formed by supplying the type of its elements. So if τ is a type, List(τ) is also a type. This is very similar to the Array Formation rule.

Introduction Rules

Introduction: empty, cons

$$\frac{}{\Gamma_\xi, \Gamma_\sigma, \Gamma_\rho \vdash EMPTY(\tau) : LIST(\tau)} \text{(EMPTY)}$$

$$\frac{\Gamma_\xi, \Gamma_\sigma, \Gamma_\rho \vdash e_1 : \tau \quad \Gamma_\xi, \Gamma_\sigma, \Gamma_\rho \vdash e_2 : LIST(\tau)}{\Gamma_\xi, \Gamma_\sigma, \Gamma_\rho \vdash CONS(e1, e2) : LIST(\tau)} \text{(CONS)}$$

Explanation: Our two introduction rules are for an empty list and a list formed by cons.

Empty's only argument is some type τ. Since we are just initializing an empty list, we don't need any premises. Our conclusion is that we return a list of that type τ.

Cons has two arguments: the first expression is the element we are adding to the list and the second argument is the list itself. We have to make sure that the expression that we are adding to the list has the same type as all other expressions in the list. Thus, we need to have two premises that make sure that e_1 is of type τ and that e_2 is a list of that same type τ. Finally, we know that cons returns a list of type τ, so our type rule concludes that CONS(e_1, e_2) returns type $LIST(\tau)$.

Elimination Rules

Elimination: null, car, cdr

$$\frac{\Gamma_\xi, \Gamma_\sigma, \Gamma_\rho \vdash e_1 : LIST(\tau)}{\Gamma_\xi, \Gamma_\sigma, \Gamma_\rho \vdash NULL(e1) : \mathit{bool}} \text{ (NULL)}$$

$$\frac{\Gamma_\xi, \Gamma_\sigma, \Gamma_\rho \vdash e_1 : LIST(\tau)}{\Gamma_\xi, \Gamma_\sigma, \Gamma_\rho \vdash CAR(e1) : \tau} \text{ (CAR)}$$

$$\frac{\Gamma_\xi, \Gamma_\sigma, \Gamma_\rho \vdash e_1 : LIST(\tau)}{\Gamma_\xi, \Gamma_\sigma, \Gamma_\rho \vdash CDR(e1) : LIST(\tau)} \text{ (CDR)}$$

Explanation: We have three elimination rules that we must define: null, car, and cdr.

Null has one argument: e_1, which represents a list. Thus, we need to have one premise that make sure that e_1 is a list of some type τ. Finally, we know that null returns true if the list is empty and returns false otherwise. Therefore, our type rule concludes that NULL(e_1) returns type bool.

Car has one argument: e_1, which represents a list. Thus, we need to have one premise that make sure that e_1 is a list of some type τ. Finally, we know that car returns the first element or the head of the list. Therefore, our type rule concludes that CAR(e_1) returns type τ.

Cdr has one argument: e_1, which represents a list. Thus, we need to have one premise that make sure that e_1 is a list of some type τ. Finally, we know that cdr returns a list of every element except the first, or the tail of the list. Therefore, our type rule concludes that CDR(e_1) returns type $LIST(\tau)$.