

The $:$ and $::$ Relations

$$\frac{\text{triv } x}{x : \text{Triv}} \quad (1)$$

$$\frac{\text{triv } x}{x :: \text{Box}(\text{Triv})} \quad (2)$$

$$\frac{\text{int } x}{x : \text{Int}} \quad (3)$$

$$\frac{\text{int } x}{x :: \text{Box}(\text{Int})} \quad (4)$$

$$\frac{x :: T}{\text{immut } x :: \text{Immut}(T)} \quad (5)$$

$$\frac{x :: T}{\text{ref } x :: \text{Box}(\text{Ref}(T))} \quad (6)$$

$$\frac{x :: T}{\& x : \text{Ref}(T)} \quad (7)$$

$$\frac{x : \text{Ref}(T)}{x @ : T} \quad (8)$$

$$\frac{x : \text{Ref}(T)}{x @ :: T} \quad (9)$$

$$\frac{x :: T \quad y :: \mathcal{U} \quad T <: \mathcal{U} \quad x : \mathcal{V}}{x := y : \mathcal{V}} \quad (10)$$

$x : T$ is read as “expression x is of type T and is in an *r-context*.”

$x :: T$ is read as “variable x is of type T and is in an *l-context*.”

The operators could also be referred to as the “r-type of” and “l-type of” operators.

l-context denotes everything that is *assignable* (indicated as a storable memory). r-context, on the other hand, denotes everything that is *expressible* (can be produced by an expression).

There is no r-value (e.g. expression) of the type $\text{Box}(T)$.

We omit rules for *Con* types as they only operate on r-values.

We omit rules for *Fun* types as they only accept r-values. Any variable and/or primitive type has both r-value and l-value (when it comes to primitive types, only r-value). In all cases, the r-value part of the actual parameter is passed when the function is being called.

The $<:$ Relation

$$\overline{\mathcal{Box}(\mathit{Triv}) <: \mathcal{Box}(\mathit{Triv})} \quad (11)$$

$$\overline{\mathcal{Box}(\mathit{Int}) <: \mathcal{Box}(\mathit{Int})} \quad (12)$$

$$\frac{T <: \mathcal{Box}(\mathcal{U})}{T <: \mathit{Immut}(\mathcal{Box}(\mathcal{U}))} \quad (13)$$

$$\frac{\mathcal{Box}(T) <: \mathit{Immut}(\mathcal{U})}{\mathit{Immut}(\mathcal{Box}(T)) <: \mathit{Immut}(\mathcal{U})} \quad (14)$$

$$\frac{T <: \mathcal{U} \quad \mathcal{Box}?(T)}{\mathit{Ref } T <: \mathit{Ref } \mathcal{U}} \quad (15)$$

$\mathcal{Box}?$ statement checks for both mutable and immutable containers. In other words, $\mathcal{Box}(\mathit{Int})$ and $\mathit{Immut}(\mathcal{Box}(\mathit{Int}))$ would both satisfy the $\mathcal{Box}?$ condition.