A CORRECTNESS

The following theorems clarify key correctness properties of the compilation rules and the target language type system and operational semantics.

Theorem 5 (Soundness of evaluation). If $\emptyset;\emptyset \vdash t : T$ then either

1. $\langle t, \emptyset, \bullet, \emptyset \rangle$ diverges,
2. $\langle t, \emptyset, \bullet, \emptyset \rangle \rightarrow^{*} (\langle \text{Fail} \rangle u, \rho, C, \sigma)$, or
3. $\langle t, \emptyset, \bullet, \emptyset \rangle \rightarrow^{*} (v, \bullet, \sigma)$ and $\exists \Sigma$ such that $\emptyset;\Sigma \vdash v : T$ and $\emptyset;\Sigma \vdash \sigma$.

Proof. Via standard subject reduction and progress lemmas in the style of Wright and Felleisen [24]. □

Theorem 6 (Well-typed compilation). For all $E$, $e$, and $T$, the following statements are equivalent:

1. $E \vdash e : T$
2. $\exists t$ such that $E \vdash e \hookrightarrow t : T$ and $E;\emptyset \vdash t : T$

Proof. Inductions on the compilation and source language typing derivations. □

Theorem 7 (Size of coercions). For any $e$, $c$ such that

1. $\emptyset \vdash e \hookrightarrow t : T$ and
2. $\langle t, \emptyset, \bullet, \emptyset \rangle \rightarrow^{*} M$ and
3. $M$ contains $c$,

$\exists S \in e$ such that $\text{size}(c) \leq 5(2^{\text{height}(S)} - 1)$.

Proof. Induction on the length of the reduction sequence, using Lemma 2; the base case is by induction on the compilation derivation, using Lemma 3. □

Proof (of Lemma 2). Induction on $c$. Assume $c = c_1; \ldots; c_n$, where each $c_i$ is not a sequential composition. Suppose some $c_i = \text{Ref } d_1 d_2$. So $\emptyset \vdash c_i : \text{Ref } S \hookrightarrow \text{Ref } T$. Hence $c_i$ can be preceded only by $\text{Ref } !$, which must be the first coercion in the sequence, and similarly can be followed only by $\text{Ref } !$, which must be the last coercion in the sequence.

Thus in the worst case $c = \text{Ref } ?; \text{Ref } d_1 d_2; \text{Ref } !$, and $\text{size}(c) = 5 + \text{size}(d_1) + \text{size}(d_2)$. Applying the induction hypothesis to the sizes of $d_1$ and $d_2$ yields:

$$\text{size}(c) \leq 5 + 2(5(2^{\text{height}(S)} - 1)) = 5(2^{\text{height}(c)} - 1)$$

The case for $\text{Fun } d_1 d_2$ is similar. The coercions $I$ and $\text{Fail}$ can only appear alone. Finally, coercions of the form $D?; D!$ are valid. However, composition of a coercion $c$ matching this pattern with one of the other valid coercions is either ill-typed or triggers a normalization that yields a coercion identical to $c$. □

Proof (of Theorem 4). During evaluation, the reduction rules prevent the nesting of adjacent pairs of nested coercions in any value or term in the configuration; similarly, no evaluation context may contain adjacent nested coercions. Thus the number of coercions in any component of the configuration is proportional to the size of that component. By Theorem 7 the size of each coercion is in $O(2^{\text{height}(S)})$ for the largest $S$ in $e$. □
Figure 4: Target Language Syntax, Semantics, and Type Rules

Terms: \( s, t ::= k \mid a \mid x \mid \lambda x:T. t \mid t \mid \mathbf{ref} \ t \mid \!\!t \mid t := t \mid (c) t \)

Stores: \( \sigma ::= \emptyset \mid \sigma[a := v] \)

Typing environments: \( E ::= \emptyset \mid E, x:T \)

Store typings: \( \Sigma ::= \emptyset \mid \Sigma, a : T \)

Configurations: \( M ::= \langle t, \rho, C, \sigma \rangle \mid \langle v, C, \sigma \rangle \)

Values: \( u ::= x \mid f \mid k \mid a \)

Uncoered values: \( u ::= x \mid f \mid k \mid a \)

Function closures: \( f ::= \langle \lambda x:T. t, \rho \rangle \)

Environments: \( \rho ::= \emptyset \mid \rho[x := v] \)

Frames: \( F ::= \langle \bullet \rangle \mid \langle v \bullet \rangle \mid \mathbf{ref} \bullet \mid \bullet \mid \bullet ::= t \mid v ::= \bullet \)

Coercion frames: \( G ::= \langle c \rangle \bullet \)

Contexts: \( C ::= \bullet \mid C : (F, \rho) \mid D : (G, \rho) \)

Coercion Contexts: \( D ::= \bullet \mid \bullet : C : (F, \rho) \)

Type rules:

\[
\begin{align*}
\text{T-Var} & : & E \Sigma \vdash t : T & & E \Sigma \vdash \sigma
\end{align*}
\]

\[
\begin{align*}
\text{T-Fun} & : & \langle s, c \rangle \in E & & E \Sigma \vdash \lambda x:T. s : T & & E \Sigma \vdash (c : T) : T
\end{align*}
\]

\[
\begin{align*}
\text{T-App} & : & E \Sigma \vdash t_1 : (s \to T) & & E \Sigma \vdash t_2 : S & & E \Sigma \vdash t_1 t_2 : T
\end{align*}
\]

\[
\begin{align*}
\text{T-Ref} & : & E \Sigma \vdash t : T & & E \Sigma \vdash \text{ref } t : \text{Ref } T & & E \Sigma \vdash \sigma
\end{align*}
\]

\[
\begin{align*}
\text{T-Deref} & : & E \Sigma \vdash t : T & & E \Sigma \vdash \text{ref } t : \text{Ref } T & & E \Sigma \vdash t : T
\end{align*}
\]

\[
\begin{align*}
\text{T-Assign} & : & E \Sigma \vdash t_1 : T & & E \Sigma \vdash t_2 : T & & E \Sigma \vdash t_1 = t_2 : T
\end{align*}
\]

\[
\begin{align*}
\text{T-Cast} & : & E \Sigma \vdash S : \to T & & E \Sigma \vdash s : S & & E \Sigma \vdash t : T
\end{align*}
\]

\[
\begin{align*}
\text{T-Const} & : & E \Sigma \vdash k : \nu(k
\end{align*}
\]

\[
\begin{align*}
\text{T-Addr} & : & (a : T) \in \Sigma & & \forall (a : \sigma(a) : \Sigma(a) \vdash \sigma)
\end{align*}
\]

\[
\begin{align*}
\text{T-Store} & : & dom(\sigma) = dom(\Sigma)
\end{align*}
\]