

Ensuring Acyclicity of the Type Hierarchy in Core Fortress

Eric Allen Joe Hallett Victor Luchangco Sukyoung Ryu
 Sam Tobin-Hochstadt

March 23, 2007

This document gives restrictions on Core Fortress programs to prevent cycles in the type hierarchy created by the program. The semantics of Core Fortress is defined in a companion report [?]. The restrictions are permissive enough to allow the definition of the covariant and contravariant types. Note that we use these restrictions only to guarantee the acyclicity of the type hierarchy; they can be replaced by any other set of restrictions that makes this guarantee. After introducing the restrictions we prove that they are sufficient.

Suppose we have the following trait definition in a program p :

$$\text{trait } T[\overrightarrow{\alpha \text{ extends } K}] \text{ extends } \{\overrightarrow{N}\} \text{ where } \{\overrightarrow{\beta \text{ extends } H}\} \text{ -- end}$$

We say that T *directly depends on* $S \neq T$ in p if $N_i = S[\overrightarrow{\tau}]$ for some i and $\overrightarrow{\tau}$, and that T *depends on* S in p if T directly depends on S or T depends on some R that directly depends on S ; the depends relation is the transitive closure of the directly-depends relation. Note that T never directly depends on itself. We say that T is *self-extending* in p if $N_i = T[\overrightarrow{\tau}]$ for some i and $\overrightarrow{\tau}$; we say that N_i is a *self-supertype* of T . We often omit mentioning the program p when it is obvious from context. Note that objects cannot be self-extending.

We impose the following three restrictions on the trait definition:

- (R1) T must not depend on itself (i.e., there are no cycles except through self-extension).
- (R2) T has at most one self-supertype.
- (R3) If T is self-extending with self-supertype $T[\overrightarrow{\tau}]$, then for each type parameter α_i , one of the following holds:
 - (a) $\tau_i = \alpha_i$;
 - (b) $\tau_i = K_i$;
 - (c) $\tau_i = \beta_j$ and $\alpha_i = H_j$ for some j ; or
 - (d) $\tau_i = \alpha_j$ and $\alpha_i = K_j$ for some $j \neq i$.

We now show that these restrictions guarantee the acyclicity of the type hierarchy.

Lemma 1. *Consider a well-typed program p with the following trait definition that meets the restrictions above:*

$$\text{trait } T[\overrightarrow{\alpha \text{ extends } K}] \text{ extends } \{\overrightarrow{N}\} \text{ where } \{\overrightarrow{\beta \text{ extends } H}\} \text{ -- end}$$

where $T[\overrightarrow{\tau}] \in \{\overrightarrow{N}\}$. For all $\overrightarrow{\tau}^j$ and $\overrightarrow{\tau}^i$ such that

$$p; \emptyset \vdash T[\overrightarrow{\tau}^j] <: T[\overrightarrow{\tau}^i],$$

and for any i ,

1. if $\tau_i = \alpha_i$ then $\tau_i'' = \tau_i'$;
2. if $\tau_i = K_i$ then $p; \emptyset \vdash \tau_i' <: \tau_i''$;
3. if $\tau_i = \beta_j$ and $\alpha_i = H_j$ for some j then $p; \emptyset \vdash \tau_i'' <: \tau_i'$; and

4. if $\tau_i = \alpha_j$ and $\alpha_i = K_j$ for some $j \neq i$
then $p; \emptyset \vdash \tau_i'' <: \tau_i'$.

Proof. Fix $\vec{\tau}'$. Note that $\vec{\tau}'$ cannot have any free type variables because we are using an empty bound environment. Suppose, for contradiction, that there exists $\vec{\tau}''$ and i such that the lemma does not hold. Choose the $\vec{\tau}''$ with the minimum-length derivation for $p; \emptyset \vdash T[\vec{\tau}'] <: T[\vec{\tau}'']$, and consider the last rule applied in this derivation. It cannot be [S-TRANS] by the minimality assumption. It cannot be [S-REFL] because in that case $\vec{\tau}' = \vec{\tau}''$, so the lemma would hold. Therefore, it must be [S-EXT]. The next rule in the derivation cannot be [E-VAR] because it does not produce a judgement of the correct form. Therefore, it must be [E-EXT], with $T[\vec{\tau}''] = [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]N_l$ for some l and $\vec{\tau}^\beta$ such that $p; \emptyset \vdash \vec{\tau}' <: [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\vec{K}$ and $p; \emptyset \vdash \vec{\tau}^\beta <: [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\vec{H}$. Because T has at most one self-supertype, $N_l = T[\vec{\tau}']$. Therefore $\vec{\tau}'' = [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\vec{\tau}$.

case: $\tau_i = \alpha_i$

$$\tau_i'' = [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\alpha_i = [\vec{\tau}^\beta / \vec{\beta}]\tau_i'. \text{ Because } \tau_i' \text{ has no free type variables, we have } \tau_i'' = \tau_i'.$$

case: $\tau_i = K_i$

$$\tau_i'' = [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]K_i. \text{ Because}$$

$$p; \emptyset \vdash \vec{\tau}' <: [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\vec{K},$$

we have $p; \emptyset \vdash \tau_i'' <: \tau_i'$.

case: $\tau_i = \beta_j$ and $\alpha_i = H_j$ for some j

$$\tau_i'' = [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\beta_j = \tau_j^\beta. \text{ Because}$$

$$p; \emptyset \vdash \vec{\tau}^\beta <: [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\vec{H},$$

we have:

$$p; \emptyset \vdash \tau_i'' = \tau_j^\beta <: [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]H_j = [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\alpha_i = \tau_i'.$$

case: $\tau_i = \alpha_j$ and $\alpha_i = K_j$ for some $j \neq i$

$$\tau_i'' = [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\alpha_j = [\vec{\tau}^\beta / \vec{\beta}]\tau_j' = \tau_j'. \text{ Because } p; \emptyset \vdash \vec{\tau}' <: [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\vec{K}, \text{ we have:}$$

$$p; \emptyset \vdash \tau_j' <: [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]K_j = [\vec{\tau}^\beta / \vec{\beta}][\vec{\tau}' / \vec{\alpha}]\alpha_i = \tau_i'.$$

Thus, the lemma holds (contrary to the assumption that it doesn't). \square

Theorem 1. *If every trait definition in a well-typed program p satisfies the restrictions above, then $p; \emptyset \vdash \tau <: \tau' <: \tau$ implies that $\tau = \tau'$.*

Proof. Suppose, for contradiction, that the theorem does not hold. Fix p and choose τ and $\tau' \neq \tau$ so that $p; \emptyset \vdash \tau <: \tau'$ and $p; \emptyset \vdash \tau' <: \tau$, and $p; \emptyset \vdash \tau' <: \tau$ has the minimum-length derivation. Note that τ cannot be Object because we have $p; \emptyset \vdash \text{Object} <: \tau'$ only if $\tau' = \text{Object}$. So τ has the form $T[\vec{\tau}']$ for some T and $\vec{\tau}'$. Consider the last rule applied to the derivation of $p; \emptyset \vdash \tau' <: \tau$.

It cannot be [S-REFL] because $\tau' \neq \tau$. It cannot be [S-TRANS] because of the minimality assumption. Thus, the last rule applied to derive $p; \emptyset \vdash \tau' <: \tau$ must be [S-EXT]. Now consider the next rule in the derivation. It cannot be [E-VAR] because $\text{dom}(\emptyset) = \emptyset$. So, it must be [E-EXT].

In this case, $\tau' = C[\vec{\tau}']$ for some C and $\vec{\tau}'$, where the definition of C has the form:

$$- C[\overline{\alpha \text{ extends } K}] \text{ extends } \{\vec{N}\} \text{ where } \{\overline{\beta \text{ extends } H}\} - \text{end}$$

Because of the restriction (R1), $C = T$, which is self-extending. Thus, because $\tau = T[\vec{\tau}]$ and $\tau' = T[\vec{\tau}']$, we have:

$$\begin{aligned} p; \emptyset \vdash T[\vec{\tau}'] &<: T[\vec{\tau}] \quad \text{and} \\ p; \emptyset \vdash T[\vec{\tau}] &<: T[\vec{\tau}']. \end{aligned}$$

By Lemma 1 and the restriction (R3), for each i , either $\tau_i = \tau'_i$ or $p; \emptyset \vdash \tau_i <: \tau'_i$ and $p; \emptyset \vdash \tau'_i <: \tau_i$. In the latter case, we also get $\tau_i = \tau'_i$ by the minimality assumption. Thus, $\vec{\tau} = \vec{\tau}'$, and $\tau = T[\vec{\tau}] = T[\vec{\tau}'] = \tau'$, contradicting the assumption that $\tau \neq \tau'$. \square