CubeX with Declaration-site Variance

Ross Tate

November 26, 2013

Adding declaration-site variance to Cubex requires 3 changes:

1. Allowing the "out" annotation on class/interface type parameters
2. Checking that covariant type parameters are only used at covariant locations in class/interface signatures
3. Updating type-checking algorithms, namely subtyping, joining, and type inference

1 Lexing and Parsing

1.1 Core and Full Language Extensions

$$
\begin{array}{rrcl}
\text{variance} & v & ::= & + \mid \pm \\
\text{variance declaration} & \Theta_\pm & ::= & v\nu_p, \ldots, v\nu_p \\
\text{interface} & i & ::= & \textbf{interface } \nu_c\langle\Theta_\pm\rangle \textbf{ extends } \tau \ \{\textbf{fun } \nu_v\sigma; \ldots \textbf{fun } \nu_v\sigma; \ \textbf{fun } \nu_v\sigma \ s \ldots \textbf{fun } \nu_v\sigma \ s\} \\
\text{class} & c & ::= & \textbf{class } \nu_c\langle\Theta_\pm\rangle(\Gamma) \textbf{ extends } \tau \ \{s \ldots s \ \textbf{super}(e, \ldots, e); \ \textbf{fun } \nu_v\sigma \ s \ldots \textbf{fun } \nu_v\sigma \ s\} \\
\text{class context} & \Psi & ::= & \varnothing \mid \Psi, \textbf{interface } \nu_c\langle\Theta_\pm\rangle \textbf{ extends } \tau \ \{\Delta; \nu_v, \ldots, \nu_v\} \mid \Psi, \textbf{class } \nu_c\langle\Theta_\pm\rangle \textbf{ extends } \tau \ \{\Delta\}
\end{array}
$$

Figure 1: Changes to the Cubex Core Language Grammar

The extension to the full language differs from the core language in a few ways:

- The new keyword `out` is used to represent $+$.
- The absence of `out` represents $\pm$.
- $+$ and $\pm$ are not used in the full language.

2 Validating

The following are the changed typing judgements and rules. Whenever a kind context Θ or $\nu, \ldots, \nu$ is used where a variance declaration $\Theta_\pm$ or $v\nu, \ldots, v\nu$ is available, then one should use the kind context that results from removing the variance annotations. In the other direction, given a kind context Θ, we use $\pm\Theta$ to denote the variance annotation resulting from giving all type variables in Θ the $\pm$ variance.

Judgement	Meaning	Figure
$\Psi \mid \Theta \vdash \tau <: \tau'$	τ is a subtype of τ'	2
$\Psi \mid \Theta \vdash_v \tau <: \tau'$	τ is more precise than τ' at variance v	2
$\Psi \vdash \nu\langle\Theta_\pm\rangle$	class/interface ν has variance declaration $\Theta_\pm$	2
$\Psi \mid \Theta_\pm \vdash_v \tau$	τ has variance v with respect to variance declaration $\Theta_\pm$	3
$\Psi \mid \Theta_\pm \vdash_\pm \Gamma$	Γ only uses invariant type variables in $\vdash_\pm$	3
$\Psi \mid \Theta_\pm \vdash_+ \sigma$	σ is covariant with respect to $\Theta_\pm$	3
$\Psi \mid \Delta \mid \Gamma \vdash i : \Psi'$	i is a valid interface with signature Ψ'	4
$\Psi \mid \Delta \mid \Gamma \vdash c : \Psi' \mid \Delta'$	c is a valid class with signature Ψ' and constructor Δ'	4
$\vdash p$	p is a valid program in the initial environment	5

3 Evaluation

Same as for PA3 except no tests will be on just stages 1 through 4. Instead, 90% of tests will be on stages 5 through 8, and 10% will be on stage 9.

$\boxed{\Psi \mid \Theta \vdash \tau <: \tau \qquad \Psi \mid \Theta \vdash_v \tau <: \tau \qquad \Psi \vdash \nu\langle\Theta_\pm\rangle}$

$$\frac{\text{for all } i, \quad \Psi \mid \Theta \vdash \tau_i <: \tau_i' \quad \text{and} \quad \Psi \mid \Theta \vdash \tau_i' <: \tau_i}{\Psi \mid \Theta \vdash \nu\langle\tau_1, \ldots, \tau_n\rangle <: \nu\langle\tau_1', \ldots, \tau_n'\rangle} \text{ becomes } \frac{\Psi \vdash \nu\langle v_1\nu_1, \ldots, v_n\nu_n\rangle \quad \text{for all } i, \quad \Psi \mid \Theta \vdash_{v_i} \tau_i <: \tau_i'}{\Psi \mid \Theta \vdash \nu\langle\tau_1, \ldots, \tau_n\rangle <: \nu\langle\tau_1', \ldots, \tau_n'\rangle}$$

$$\frac{\Psi \mid \Theta \vdash \tau <: \tau'}{\Psi \mid \Theta \vdash \texttt{Iterable}\langle\tau\rangle <: \texttt{Iterable}\langle\tau'\rangle} \text{ is removed since it now arises as a special case}$$

$$\frac{\Psi \mid \Theta \vdash \tau <: \tau'}{\Psi \mid \Theta \vdash_+ \tau <: \tau'} \qquad \frac{\Psi \mid \Theta \vdash \tau <: \tau' \quad \Psi \mid \Theta \vdash \tau' <: \tau}{\Psi \mid \Theta \vdash_\pm \tau <: \tau'}$$

$$\frac{\textbf{interface } \nu\langle\Theta_\pm\rangle \textbf{ extends } \tau \{\ldots\} \text{ in } \Psi}{\Psi \vdash \nu\langle\Theta_\pm\rangle} \qquad \frac{\textbf{class } \nu\langle\Theta_\pm\rangle \textbf{ extends } \tau \{\ldots\} \text{ in } \Psi}{\Psi \vdash \nu\langle\Theta_\pm\rangle}$$

Figure 2: Subtyping

$\boxed{\Psi \mid \Theta_\pm \vdash_v \tau \qquad \Psi \mid \Theta_\pm \vdash_\pm \Gamma \qquad \Psi \mid \Theta_\pm \vdash_+ \sigma}$

$$\frac{\Psi \mid \Theta_\pm \vdash_\pm \tau}{\Psi \mid \Theta_\pm \vdash_+ \tau} \qquad \frac{}{\Psi \mid \Theta_\pm \vdash_v \top} \qquad \frac{}{\Psi \mid \Theta_\pm \vdash_v \bot} \qquad \frac{v\nu \text{ in } \Theta_\pm}{\Psi \mid \Theta \vdash_v \nu}$$

$$\frac{\Psi \vdash \nu\langle v_1\nu_1, \ldots, v_n\nu_n\rangle \quad \text{for all } i, \quad \Psi \mid \Theta_\pm \vdash_{v_i} \tau_i}{\Psi \mid \Theta_\pm \vdash_+ \nu\langle\tau_1, \ldots, \tau_n\rangle} \qquad \frac{\text{for all } i, \quad \Psi \mid \Theta_\pm \vdash_\pm \tau_i}{\Psi \mid \Theta_\pm \vdash_\pm \nu\langle\tau_1, \ldots, \tau_n\rangle} \qquad \frac{\Psi \mid \Theta_\pm \vdash_v \tau \quad \Psi \mid \Theta_\pm \vdash_v \tau'}{\Psi \mid \Theta_\pm \vdash_v \tau \cap \tau'}$$

$$\frac{\text{for all } i, \quad \Psi \mid \Theta_\pm \vdash_\pm \tau_i}{\Psi \mid \Theta_\pm \vdash_\pm \nu_1 : \tau_1, \ldots, \nu_n : \tau_n} \qquad \frac{\Psi \mid \Theta_\pm, \pm\hat{\Theta} \vdash_\pm \Gamma \quad \Psi \mid \Theta_\pm, \pm\hat{\Theta} \vdash_+ \tau}{\Psi \mid \Theta_\pm \vdash_+ \langle\hat{\Theta}\rangle(\Gamma) : \tau}$$

Figure 3: Variance Checking

$\boxed{\Psi \mid \Delta \mid \Gamma \vdash i : \Psi \qquad \Psi \mid \Delta \mid \Gamma \vdash c : \Psi \mid \Delta}$

$$\frac{\begin{array}{c}\Psi \mid \Theta \vdash_\top \tau \qquad \Psi \mid \Theta_\pm \vdash_+ \tau \\ \Psi' = \textbf{interface } \nu\langle\Theta_\pm\rangle \textbf{ extends } \tau \{\nu_1\sigma_1, \ldots, \nu_n\sigma_n, \hat{\nu}_1\hat{\sigma}_1, \ldots, \hat{\nu}_{\hat{n}}\hat{\sigma}_{\hat{n}}; \hat{\nu}_1, \ldots, \hat{\nu}_{\hat{n}}\} \\ \text{for all } i, \quad \Psi, \Psi' \mid \Theta \vdash \sigma_i \quad \Psi, \Psi' \mid \Theta_\pm \vdash_+ \sigma_i \quad \text{for all } i, \quad \Psi, \Psi' \mid \Theta \vdash \hat{\sigma}_i \quad \Psi, \Psi' \mid \Theta_\pm \vdash_+ \hat{\sigma}_i \\ \Psi, \Psi' \mid \Theta \vdash \nu\langle\Theta\rangle : \Delta' \\ \text{for all } \bar{\nu}\langle\bar{\Theta}\rangle(\bar{\Gamma}) : \bar{\tau} \text{ in } \Delta', \quad \text{for all } \bar{\nu}' \text{ in } \bar{\Theta}, \quad \bar{\nu}' \text{ not in } \Theta \\ \text{for all } i, \quad \hat{\sigma}_i = \langle\Theta_i\rangle(\Gamma_i) : \tau_i \quad \Psi, \Psi' \mid \Theta, \Theta_i \mid \Delta, \Delta' \mid \Gamma \mid \Gamma_i \vdash^{\textbf{true}}_{\tau_i} \hat{s}_i : \Gamma_i'\end{array}}{\Psi \mid \Delta \mid \Gamma \vdash \textbf{interface } \nu\langle\Theta_\pm\rangle \textbf{ extends } \tau \{\textbf{fun } \nu_1\sigma_1; \ldots; \textbf{fun } \nu_n\sigma_n; \textbf{ fun } \hat{\nu}_1\hat{\sigma}_1 = \hat{s}_1 \ldots \textbf{fun } \hat{\nu}_{\hat{n}}\sigma_{\hat{n}} = \hat{s}_{\hat{n}}\} : \Psi'}$$

$$\frac{\begin{array}{c}\Psi \mid \Theta \vdash_{\hat{\tau}} \tau \qquad \Psi \mid \Theta_\pm \vdash_+ \tau \\ \Psi' = \textbf{class } \nu\langle\Theta_\pm\rangle \textbf{ extends } \tau \{\nu_1\sigma_1; \ldots; \nu_n\sigma_n;\} \qquad \Delta' = \nu\langle\Theta\rangle(\hat{\Gamma}) : \nu\langle\Theta\rangle \\ \text{for all } i, \quad \Psi, \Psi' \mid \Theta \vdash \sigma_i \qquad \Psi, \Psi' \mid \Theta_\pm \vdash_+ \sigma_i \\ \Psi, \Psi' \mid \Theta \vdash \hat{\Gamma} \quad \hat{\Gamma}_0 = \hat{\Gamma} \quad \text{for all } i, \quad \Psi, \Psi' \mid \Theta \mid \Delta, \Delta' \mid \Gamma \mid \hat{\Gamma}_{i-1} \vdash s_i : \hat{\Gamma}_i \\ \hat{\tau}(e_1, \ldots, e_k) = \top() \quad \text{or} \quad \Psi, \Psi' \mid \Theta \mid \Delta, \Delta' \mid \Gamma, \hat{\Gamma}_m \vdash \hat{\tau}(e_1, \ldots, e_k) : \hat{\tau} \\ \Psi, \Psi' \mid \Theta \vdash \nu\langle\Theta\rangle : \Delta'' \\ \text{for all } \bar{\nu}\langle\bar{\Theta}\rangle(\bar{\Gamma}) : \bar{\tau} \text{ in } \Delta'', \quad \text{for all } \bar{\nu}' \text{ in } \bar{\Theta}, \quad \bar{\nu}' \text{ not in } \Theta \\ \text{for all } i, \quad \sigma_i = \langle\Theta_i\rangle(\Gamma_i) : \tau_i \quad \Psi, \Psi' \mid \Theta, \Theta_i \mid \Delta, \Delta', \Delta'' \mid \Gamma, \hat{\Gamma}_m \mid \Gamma_i \vdash^{\textbf{true}}_{\tau_i} \hat{s}_i : \Gamma_i' \\ \text{for all } \hat{\nu}\sigma \text{ in } \Delta'', \quad \Psi, \Psi' \mid \Theta \vdash \nu\langle\Theta\rangle : \hat{\nu}\end{array}}{\Psi \mid \Delta \mid \Gamma \vdash \textbf{class } \nu\langle\Theta_\pm\rangle(\hat{\Gamma}) \textbf{ extends } \tau \{s_1 \ldots s_m \textbf{ super}(e_1, \ldots, e_k); \textbf{ fun } \nu_1\sigma_1 \ \hat{s}_1 \ \ldots \ \textbf{fun } \nu_n\sigma_n \ \hat{s}_n\} : \Psi' \mid \Delta'}$$

Figure 4: Class and Interface Checking

$\boxed{\vdash p}$

$$
\begin{array}{ll}
\Psi_0 = & \textbf{class } \texttt{Iterable}\langle +\texttt{E}\rangle \textbf{ extends } \top \{\ \}, \\
& \textbf{class } \texttt{Boolean}\langle\rangle \textbf{ extends } \top \{ \\
& \quad \texttt{negate}\langle\rangle() : \texttt{Boolean}\langle\rangle; \\
& \quad \texttt{and}\langle\rangle(\texttt{that} : \texttt{Boolean}\langle\rangle) : \texttt{Boolean}\langle\rangle; \\
& \quad \texttt{or}\langle\rangle(\texttt{that} : \texttt{Boolean}\langle\rangle) : \texttt{Boolean}\langle\rangle; \\
& \quad \texttt{through}\langle\rangle(\texttt{upper} : \texttt{Boolean}\langle\rangle, \texttt{includeLower} : \texttt{Boolean}\langle\rangle, \texttt{includeUpper} : \texttt{Boolean}\langle\rangle) : \texttt{Iterable}\langle\texttt{Boolean}\langle\rangle\rangle; \\
& \quad \texttt{onwards}\langle\rangle(\texttt{inclusive} : \texttt{Boolean}\langle\rangle) : \texttt{Iterable}\langle\texttt{Boolean}\langle\rangle\rangle; \\
& \quad \texttt{lessThan}\langle\rangle(\texttt{that} : \texttt{Boolean}\langle\rangle, \texttt{strict} : \texttt{Boolean}\langle\rangle) : \texttt{Boolean}\langle\rangle; \\
& \quad \texttt{equals}\langle\rangle(\texttt{that} : \texttt{Boolean}\langle\rangle) : \texttt{Boolean}\langle\rangle; \\
& \}, \\
& \textbf{class } \texttt{Integer}\langle\rangle \textbf{ extends } \top \{ \\
& \quad \texttt{negative}\langle\rangle() : \texttt{Integer}\langle\rangle; \\
& \quad \texttt{times}\langle\rangle(\texttt{factor} : \texttt{Integer}\langle\rangle) : \texttt{Integer}\langle\rangle; \\
& \quad \texttt{divide}\langle\rangle(\texttt{divisor} : \texttt{Integer}\langle\rangle) : \texttt{Iterable}\langle\texttt{Integer}\langle\rangle\rangle; \\
& \quad \texttt{modulo}\langle\rangle(\texttt{modulus} : \texttt{Integer}\langle\rangle) : \texttt{Iterable}\langle\texttt{Integer}\langle\rangle\rangle; \\
& \quad \texttt{plus}\langle\rangle(\texttt{summand} : \texttt{Integer}\langle\rangle) : \texttt{Integer}\langle\rangle; \\
& \quad \texttt{minus}\langle\rangle(\texttt{subtrahend} : \texttt{Integer}\langle\rangle) : \texttt{Integer}\langle\rangle; \\
& \quad \texttt{through}\langle\rangle(\texttt{upper} : \texttt{Integer}\langle\rangle, \texttt{includeLower} : \texttt{Boolean}\langle\rangle, \texttt{includeUpper} : \texttt{Boolean}\langle\rangle) : \texttt{Iterable}\langle\texttt{Integer}\langle\rangle\rangle; \\
& \quad \texttt{onwards}\langle\rangle(\texttt{inclusive} : \texttt{Boolean}\langle\rangle) : \texttt{Iterable}\langle\texttt{Integer}\langle\rangle\rangle; \\
& \quad \texttt{lessThan}\langle\rangle(\texttt{that} : \texttt{Integer}\langle\rangle, \texttt{strict} : \texttt{Boolean}\langle\rangle) : \texttt{Boolean}\langle\rangle; \\
& \quad \texttt{equals}\langle\rangle(\texttt{that} : \texttt{Integer}\langle\rangle) : \texttt{Boolean}\langle\rangle; \\
& \}, \\
& \textbf{class } \texttt{Character}\langle\rangle \textbf{ extends } \top \{ \\
& \quad \texttt{unicode}\langle\rangle() : \texttt{Integer}\langle\rangle; \\
& \quad \texttt{equals}\langle\rangle(\texttt{that} : \texttt{Character}\langle\rangle) : \texttt{Boolean}\langle\rangle; \\
& \}, \\
& \textbf{class } \texttt{String}\langle\rangle \textbf{ extends } \texttt{Iterable}\langle\texttt{Character}\langle\rangle\rangle \{\ \texttt{equals}\langle\rangle(\texttt{that} : \texttt{String}\langle\rangle) : \texttt{Boolean}\langle\rangle;\ \} \\
\Delta_0 = & \texttt{character}\langle\rangle(\texttt{unicode} : \texttt{Integer}\langle\rangle) : \texttt{Character}\langle\rangle, \texttt{string}\langle\rangle(\texttt{characters} : \texttt{Iterable}\langle\texttt{Character}\rangle) : \texttt{String}\langle\rangle \\
\Gamma_0 = & \texttt{input} : \texttt{Iterable}\langle\texttt{String}\langle\rangle\rangle
\end{array}
$$

$$\frac{\Psi_0 \mid \Delta_0 \mid \Gamma_0 \vdash p}{\vdash p}$$

Figure 5: Program Checking