

Toward Type-Preserving Compilation of Coq

William J. Bowman



TYPES in target languages

If $\Gamma \vdash e : t$ then $\Gamma^+ \vdash e^+ : t^+$ (where $^+$ denotes compilation)

- Correct compilation with linking
- Secure compilation
- Certifying compilation / Proof-carrying code
- Type-directed optimization

Calculus of Constructions, Closure Converted

$$\frac{x_i : t_i \dots \vdash e : t' \quad \Gamma \vdash \Pi(x_i : t_i \dots). t' : U}{\Gamma \vdash \lambda(x_i : t_i \dots). e : \Pi(x_i : t_i \dots). t'}$$

A specification for closure conversion:
functions must be closed to be well-typed

STANDARD CLOSURE CONVERSION FAILS

$$(\lambda x : t. e)^+ = \text{pack } \langle t_{env}, \langle env, \lambda(x_{env} : t_{env}, x : t^+). e \rangle \rangle \quad \text{where } e = \text{let } x_i = \pi_i x_{env} \text{ in } e^+$$

$$(\Pi x : t. t')^+ = \exists \alpha : \text{Type}_i. (\alpha \times \Pi(x_{env} : \alpha, x : t^+). t'^+)$$

$$env = \langle x_i \dots \rangle = \text{fv}(e^+)$$

Type translation does not know the environment
expects $e : t'^+$

but term translation puts environment in the type (due to dependency)
creates $e : t'^+[\pi_i x_{env}/x_i]$

$$\frac{\frac{\frac{x_{env} : t_{env}, x : t^+ \vdash e : t'^+}{\Gamma \vdash \lambda(x_{env} : t_{env}, x : t^+). e : \Pi(x_{env} : t_{env}, x : t^+). t'^+}}{\vdots}}{\Gamma \vdash \text{pack } \langle t_{env}, \langle env, \lambda(x_{env} : t_{env}, x : t^+). e \rangle \rangle : \exists \alpha : t''. (\alpha \times \Pi(x_{env} : \alpha, x : t^+). t'^+)}$$

Intuitively, x_{env} should only ever be env
so $x_i \equiv \pi_i x_{env}$

Problem: Type translation must refer to the environment, which doesn't exist until the type is "used" to type check a closure

OUR TRANSLATION WORKS

$$(\lambda x : t. e)^+ = \text{pack } \langle t_{env}, env, \lambda(x_{env} : t_{env}, x : t). e \rangle \quad \text{where } e = \text{let } x_i = \pi_i x_{env} \text{ in } e^+$$

$$(\Pi x : t. t')^+ = \exists \alpha : \text{Type}_i, x_{te} : \alpha. (x_{te} \Rightarrow \Pi x : t^+. t'^+)$$

$$env = \langle x_i \dots \rangle = \text{fv}(e^+)$$

Solution:

1. Use **translucent function types** $e' \Rightarrow t$ to unify environment with free type variables (Morrisett *et al.* 1998)
A function that knows which term it will be applied to, yielding additional type equalities.

$$\frac{\Gamma \vdash e : \Pi x : t'. t_1 \quad t_1[e'/x] \equiv t \quad \Gamma \vdash e' : t' \quad \Gamma \vdash e : e' \Rightarrow t}{\Gamma \vdash e : e' \Rightarrow t} \quad \Gamma \vdash e e' : t$$

Now, our intuition is formalized within the closure: $t'^+[\pi_i x_{env}/x_i][env/x_{env}] \equiv t'^+$

2. Use **dependent existential types** to quantify over the environment.

The environment exists, but only as a variable, until we use the translated type to type-check a closure.

$$\frac{\frac{\frac{\vdots}{\Gamma \vdash \lambda(x_{env} : t_{env}, x : t^+). e : \Pi(x_{env} : t_{env}, x : t^+). t'^+[\pi_i x_{env}/x_i]}{\text{trivial, since } env = \langle x_i \dots \rangle} \quad t'^+[\pi_i x_{env}/x_i][env/x_{env}] \equiv t'^+}{\Gamma \vdash \lambda(x_{env} : t_{env}, x : t^+). e : env \Rightarrow \Pi x : t^+. t'^+}}{\Gamma \vdash \text{pack } \langle t_{env}, env, \lambda(x_{env} : t_{env}, x : t^+). e \rangle : \exists \alpha : \text{Type}_i, x_{te} : \alpha. (x_{te} \Rightarrow \Pi x : t^+. t'^+)}$$

OTHER interesting aspects

Type-preservation proof

Lemma (Preservation of Equivalence)

If $\Gamma \vdash e \equiv e'$, then $\Gamma^+ \vdash e^+ \equiv e'^+$

Lemma (Preservation of Subtyping)

If $\Gamma \vdash t_1 \preceq t_2$, then $\Gamma^+ \vdash t_1^+ \preceq t_2^+$

Lemma (Type Preservation)

1. If $\vdash \Gamma$ then $\vdash \Gamma^+$
2. If $\Gamma \vdash e : t$ then $\Gamma^+ \vdash e^+ : t^+$

Compiler correctness "for free"

Lemma (Simulation of Reduction Relation)

If $\Delta; \Gamma \vdash e \triangleright_x e'$ then $\Gamma^+ \vdash e^+ \triangleright_x^* e'^+$ and $e \equiv e'^+$

Because the type system
includes the contextual closure of
the small-step reduction relation.

An eta principle / normal form for closures

$$\frac{\Gamma \vdash e \triangleright^* \text{pack } \langle \Sigma_{env}, env, \lambda(x_{env} : \Sigma_{env}, x : t_1). e_1 \rangle \quad \Gamma \vdash e' \triangleright^* e_2}{\Gamma, x : t_1[env/x_{env}] \vdash e_1[env/x_{env}] \equiv \text{unpack } \langle \alpha, x_{te}, f \rangle = e_2 \text{ in } f x_{te} x} \quad \Gamma \vdash e \equiv e'$$

Necessary to reason syntactically about closures in the type system

Soon:
Inductive types &
guarded recursion.

