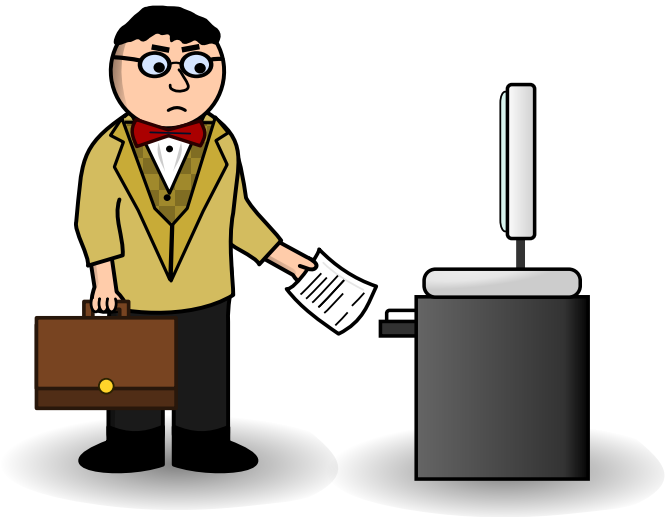
Contracts Made Manifest

Michael Hansen

Indiana University

October 27, 2010



Contributions

- Extension of Gronski and Flanagan 2007
- Strengthen results by adding translation ψ ($\lambda_H \to \lambda_C$)
- Add dependent versions of λ_C and λ_H
  - Highlight differences between picky and lax λ_C
- Extend ψ and ϕ to dependent languages

Contracts: Latent vs. Manifest

- Latent (λ_C)
  - Purely dynamic
  - Transparent to type system
  - Two blame labels
    - ★ l - "My fault"
    - ★ l' - "Someone else's fault"
- Manifest (λ_H)
  - Refinement types with casts
  - Records most recent check in type
  - Single blame label (l - "Invalid cast")

Contracts: Latent vs. Manifest

- Latent (λ_C)
  - Purely dynamic
  - Transparent to type system
  - Two blame labels
    - l - "My fault"
    - l' - "Someone else's fault"
- Manifest (λ_H)
  - Refinement types with casts
  - Records most recent check in type
  - Single blame label (l - "Invalid cast")

Non-dependent λ_C

- Simply-typed Lambda Calculus with latent contracts
- Contracts only over base types
- $\langle\{x : B \mid t\}\rangle^{l,l'}$
  - t may only have x free
- $\phi : \lambda_C \to \lambda_H$
  - Preserves behavior
  - Homomorphism
    - Base type contracts $\to$ base type casts
    - Function contracts $\to$ function casts

Non-dependent λ_C

- Simply-typed Lambda Calculus with latent contracts
- Contracts only over base types
- $\langle\{x : B \mid t\}\rangle^{l,l'}$
  - t may only have x free
- $\phi : \lambda_C \to \lambda_H$
  - Preserves behavior
  - Homomorphism
    - ★ Base type contracts $\to$ base type casts
    - ★ Function contracts $\to$ function casts

Non-dependent λ_C Syntax and Evaluation

Syntax

$$\begin{array}{rcl} B & ::= & \mathsf{Bool} \mid \ldots \\ k & ::= & \mathsf{true} \mid \mathsf{false} \mid \ldots \end{array}$$

Types and contracts

$$\begin{array}{rcl} T & ::= & B \mid T_1 \to T_2 \\ c & ::= & \{x{:}B \mid t\} \mid c_1 \mapsto c_2 \end{array}$$

Terms, values, results, and evaluation contexts

$$\begin{array}{rcl} t & ::= & x \mid k \mid \lambda x{:}T_1.\ t_2 \mid t_1\ t_2 \mid \\ & & \Uparrow l \mid \langle c \rangle^{l,l'} \mid \langle \{x{:}B \mid t_1\}, t_2, k \rangle^{l} \\ v & ::= & k \mid \lambda x{:}T_1.\ t_2 \mid \langle c \rangle^{l,l'} \mid \langle c_1 \mapsto c_2 \rangle^{l,l'}\ v \\ r & ::= & v \mid \Uparrow l \\ E & ::= & [\,]\ t \mid v\ [\,] \mid \langle \{x{:}B \mid t\}, [\,], k \rangle^{l} \end{array}$$

$$\frac{}{k\ v \longrightarrow_c [\![k]\!](v)}\ \text{E\_Const}$$

$$\frac{}{(\lambda x{:}T_1.\ t_2)\ v \longrightarrow_c t_2\{x := v\}}\ \text{E\_Beta}$$

$$\frac{}{\langle \{x{:}B \mid t\} \rangle^{l,l'}\ k \longrightarrow_c \langle \{x{:}B \mid t\}, t\{x := k\}, k \rangle^{l}}\ \text{E\_CCheck}$$

$$\frac{}{\langle \{x{:}B \mid t\}, \mathsf{true}, k \rangle^{l} \longrightarrow_c k}\ \text{E\_OK}$$

$$\frac{}{\langle \{x{:}B \mid t\}, \mathsf{false}, k \rangle^{l} \longrightarrow_c \Uparrow l}\ \text{E\_Fail}$$

$$\frac{}{(\langle c_1 \mapsto c_2 \rangle^{l,l'}\ v)\ v' \longrightarrow_c \langle c_2 \rangle^{l,l'}\ (v\ (\langle c_1 \rangle^{l',l}\ v'))}\ \text{E\_CDecomp}$$

$$\frac{t_1 \longrightarrow_c t_2}{E\,[t_1] \longrightarrow_c E\,[t_2]}\ \text{E\_Compat}$$

$$\frac{}{E\,[\Uparrow l] \longrightarrow_c \Uparrow l}\ \text{E\_Blame}$$

Non-dependent λ_C Type Rules

- Checks occur in empty context ($\emptyset$)

$$\boxed{\Gamma \vdash t : T} \qquad \frac{x{:}T \in \Gamma}{\Gamma \vdash x : T} \quad \text{T\_Var}$$

$$\frac{}{\Gamma \vdash k : \mathrm{ty_c}(k)} \quad \text{T\_Const}$$

$$\frac{\Gamma, x{:}T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x{:}T_1.\ t_2 : T_1 \to T_2} \quad \text{T\_Lam}$$

$$\frac{\Gamma \vdash t_1 : T_1 \to T_2 \qquad \Gamma \vdash t_2 : T_1}{\Gamma \vdash t_1\ t_2 : T_2} \quad \text{T\_App}$$

$$\frac{\vdash_c c : T}{\Gamma \vdash \langle c \rangle^{l,l'} : T \to T} \quad \text{T\_Contract}$$

$$\frac{}{\Gamma \vdash \Uparrow l : T} \quad \text{T\_Blame}$$

$$\frac{\emptyset \vdash k : B \qquad \emptyset \vdash t_2 : \mathsf{Bool} \qquad \vdash_c \{x{:}B \mid t_1\} : B \qquad \vdash t_2 \supset t_1\{x := k\}}{\emptyset \vdash \langle \{x{:}B \mid t_1\}, t_2, k \rangle^{l} : B} \quad \text{T\_Checking}$$

$$\boxed{\vdash_c c : T} \qquad \frac{x{:}B \vdash t : \mathsf{Bool}}{\vdash_c \{x{:}B \mid t\} : B} \quad \text{T\_BaseC}$$

$$\frac{\vdash_c c_1 : T_1 \qquad \vdash_c c_2 : T_2}{\vdash_c c_1 \mapsto c_2 : T_1 \to T_2} \quad \text{T\_FunC}$$

$$\boxed{\vdash t_2 \supset t_1} \qquad \frac{t_1 \longrightarrow_c^* \mathsf{true} \ \text{ implies } \ t_2 \longrightarrow_c^* \mathsf{true}}{\vdash t_1 \supset t_2} \quad \text{T\_Imp}$$

Non-dependent λ_H

- Simply-typed Lambda Calculus with refinement types and casts
- Only base types can be refined
- Raw types
  - $x : B \to \{x : B \mid \text{true}\}$
- $\langle S_1 \Rightarrow S_2 \rangle^l$
  - Cast from type S_1 to S_2
  - Only one blame label
- $\psi : \lambda_H \to \lambda_C$
  - Preserves behavior
- $ty_c(k)$ and $ty_h(k)$ must agree on "type skeleton"
  - if $ty_c(k) = B_1 \to B_2$ then $ty_h(k) = \{x : B_1 \mid s_1\} \to \{x : B_2 \mid s_2\}$

Non-dependent λ_H

- Simply-typed Lambda Calculus with refinement types and casts
- Only base types can be refined
- Raw types
  - $x : B \to \{x : B \mid \text{true}\}$
- $\langle S_1 \Rightarrow S_2 \rangle^l$
  - Cast from type S_1 to S_2
  - Only one blame label
- $\psi : \lambda_H \to \lambda_C$
  - Preserves behavior
- $ty_c(k)$ and $ty_h(k)$ must agree on "type skeleton"
  - if $ty_c(k) = B_1 \to B_2$ then $ty_h(k) = \{x : B_1 \mid s_1\} \to \{x : B_2 \mid s_2\}$

Non-dependent λ_H Syntax and Evaluation

Types

$$S \quad ::= \quad \{x{:}B \mid s_1\} \mid S_1 \to S_2$$

Terms, values, results, and evaluation contexts

$$\begin{array}{lcl} s & ::= & x \mid k \mid \lambda x{:}S_1.\ s_2 \mid s_1\ s_2 \mid \\ & & \Uparrow l \mid \langle S_1 \Rightarrow S_2 \rangle^l \mid \langle \{x{:}B \mid s_1\}, s_2, k \rangle^l \\ w & ::= & k \mid \lambda x{:}S_1.\ s_2 \mid \langle S_1 \Rightarrow S_2 \rangle^l \mid \\ & & \langle S_{11} \to S_{12} \Rightarrow S_{21} \to S_{22} \rangle^l\ w \\ q & ::= & w \mid \Uparrow l \\ F & ::= & []\ s \mid w\ [] \mid \langle \{x{:}B \mid s\}, [], k \rangle^l \end{array}$$

Erase

$$\begin{array}{lcl} \lfloor \{x{:}B \mid s\} \rfloor & = & B \\ \lfloor S_1 \to S_2 \rfloor & = & \lfloor S_1 \rfloor \to \lfloor S_2 \rfloor \end{array}$$

Raw

$$\begin{array}{lcl} \lceil \{x{:}B \mid s\} \rceil & = & \{x{:}B \mid \mathsf{true}\} \\ \lceil S_1 \to S_2 \rceil & = & \lceil S_1 \rceil \to \lceil S_2 \rceil \end{array}$$

$$\frac{}{k\ w \longrightarrow_h [\![k]\!](w)}\ \text{F\_CONST}$$

$$\frac{}{(\lambda x{:}S_1.\ s_2)\ w_2 \longrightarrow_h s_2\{x := w_2\}}\ \text{F\_BETA}$$

$$\frac{}{\langle \{x{:}B \mid s_1\} \Rightarrow \{x{:}B \mid s_2\} \rangle^l\ k \longrightarrow_h \langle \{x{:}B \mid s_2\}, s_2\{x := k\}, k \rangle^l}\ \text{F\_CCHECK}$$

$$\frac{}{\langle \{x{:}B \mid s\}, \mathsf{true}, k \rangle^l \longrightarrow_h k}\ \text{F\_OK}$$

$$\frac{}{\langle \{x{:}B \mid s\}, \mathsf{false}, k \rangle^l \longrightarrow_h \Uparrow l}\ \text{F\_FAIL}$$

$$\frac{}{(\langle S_{11} \to S_{12} \Rightarrow S_{21} \to S_{22} \rangle^l\ w)\ w' \longrightarrow_h \langle S_{12} \Rightarrow S_{22} \rangle^l\ (w\ (\langle S_{21} \Rightarrow S_{11} \rangle^l\ w'))}\ \text{F\_CDECOMP}$$

$$\frac{s_1 \longrightarrow_h s_2}{F\,[s_1] \longrightarrow_h F\,[s_2]}\ \text{F\_COMPAT}$$

$$\frac{}{F\,[\Uparrow l] \longrightarrow_h \Uparrow l}\ \text{F\_BLAME}$$

Non-dependent λ_H Type Rules

- Again, checks occur in empty context ($\emptyset$)

$\boxed{\Delta \vdash s : S}$

$$\frac{x{:}S \in \Delta}{\Delta \vdash x : S} \quad \text{S\_Var}$$

$$\frac{}{\Delta \vdash k : \mathrm{ty_h}(k)} \quad \text{S\_Const}$$

$$\frac{\vdash S_1 \qquad \Delta, x{:}S_1 \vdash s_2 : S_2}{\Delta \vdash \lambda x{:}S_1.\ s_2 : S_1 \to S_2} \quad \text{S\_Lam}$$

$$\frac{\Delta \vdash s_1 : S_1 \to S_2 \qquad \Delta \vdash s_2 : S_1}{\Delta \vdash s_1\ s_2 : S_2} \quad \text{S\_App}$$

$$\frac{\vdash S_1 \qquad \vdash S_2 \qquad \lfloor S_1 \rfloor = \lfloor S_2 \rfloor}{\Delta \vdash \langle S_1 \Rightarrow S_2 \rangle^l : S_1 \to S_2} \quad \text{S\_Cast}$$

$$\frac{\vdash S}{\Delta \vdash \Uparrow l : S} \quad \text{S\_Blame}$$

$$\frac{\Delta \vdash s : S_1 \qquad \vdash S_2 \qquad \vdash S_1 <: S_2}{\Delta \vdash s : S_2} \quad \text{S\_Sub}$$

$$\frac{\begin{array}{c}\emptyset \vdash k : \{x{:}B \mid \mathsf{true}\} \qquad \emptyset \vdash s_2 : \{x{:}\mathsf{Bool} \mid \mathsf{true}\} \\ \vdash \{x{:}B \mid s_1\} \qquad \vdash s_2 \supset s_1\{x := k\}\end{array}}{\emptyset \vdash \langle \{x{:}B \mid s_1\}, s_2, k \rangle^l : \{x{:}B \mid s_1\}} \quad \text{S\_Checking}$$

$\boxed{\vdash S_1 <: S_2}$

$$\frac{\forall k \in \mathcal{K}_B.\ \vdash s_1\{x := k\} \supset s_2\{x := k\}}{\vdash \{x{:}B \mid s_1\} <: \{x{:}B \mid s_2\}} \quad \text{SSub\_Refine}$$

$$\frac{\vdash S_{21} <: S_{11} \qquad \vdash S_{12} <: S_{22}}{\vdash S_{11} \to S_{12} <: S_{21} \to S_{22}} \quad \text{SSub\_Fun}$$

$\boxed{\vdash s_1 \supset s_2}$

$$\frac{s_1 \longrightarrow_h^* \mathsf{true} \ \text{ implies } \ s_2 \longrightarrow_h^* \mathsf{true}}{\vdash s_1 \supset s_2} \quad \text{S\_Imp}$$

$\boxed{\vdash S}$

$$\frac{}{\vdash \{x{:}B \mid \mathsf{true}\}} \quad \text{SWF\_Raw}$$

$$\frac{x{:}\{x{:}B \mid \mathsf{true}\} \vdash s : \{x{:}\mathsf{Bool} \mid \mathsf{true}\}}{\vdash \{x{:}B \mid s\}} \quad \text{SWF\_Refine}$$

$$\frac{\vdash S_1 \qquad \vdash S_2}{\vdash S_1 \to S_2} \quad \text{SWF\_Fun}$$

Non-dependent Translations

- Interesting parts deal with casts and contracts
- Everything else is translated homomorphically
- $\phi : \lambda_C \to \lambda_H$
  - Mimic two blame labels - $\phi(\langle c \rangle^{l,l'}) = \lambda x : \lceil c \rceil . \langle \phi(c) \Rightarrow \lceil c \rceil \rangle^{l'} (\langle \lceil c \rceil \Rightarrow \phi(c) \rangle^{l})$
  - Descend into contract predicates - $\phi(\{x : B \mid t\}) = \{x : B \mid \phi(t)\}$
  - Split function contracts - $\phi(c_1 \longmapsto c_2) = \phi(c_1) \to \phi(c_2)$
- $\psi : \lambda_H \to \lambda_C$
  - Duplication of blame label - $\psi(\langle S_1 \Rightarrow S_2 \rangle^{l}) = \langle \psi(S_1, S_2) \rangle^{l,l}$
  - Use cast target predicate - $\psi(\{x : B \mid s_1\}, \{x : B \mid s_2\}) = \{x : B \mid \psi(s_2)\}$
  - Translate domain contravariantly - $\psi(S_{11} \to S_{12}, S_{21} \to S_{22}) = \psi(S_{21}, S_{11}) \longmapsto \psi(S_{12}, S_{22})$
- Both ϕ and ψ preserve behavior in a strong sense

Non-dependent Translations

- Interesting parts deal with casts and contracts
- Everything else is translated homomorphically
- $\phi : \lambda_C \to \lambda_H$
  - Mimic two blame labels - $\phi(\langle c \rangle^{l,l'}) = \lambda x : \lceil c \rceil . \langle \phi(c) \Rightarrow \lceil c \rceil \rangle^{l'}(\langle \lceil c \rceil \Rightarrow \phi(c) \rangle^{l})$
  - Descend into contract predicates - $\phi(\{x : B \mid t\}) = \{x : B \mid \phi(t)\}$
  - Split function contracts - $\phi(c_1 \longmapsto c_2) = \phi(c_1) \to \phi(c_2)$
- $\psi : \lambda_H \to \lambda_C$
  - Duplication of blame label - $\psi(\langle S_1 \Rightarrow S_2 \rangle^{l}) = \langle \psi(S_1, S_2) \rangle^{l,l}$
  - Use cast target predicate - $\psi(\{x : B \mid s_1\}, \{x : B \mid s_2\}) = \{x : B \mid \psi(s_2)\}$
  - Translate domain contravariantly - $\psi(S_{11} \to S_{12}, S_{21} \to S_{22}) = \psi(S_{21}, S_{11}) \longmapsto \psi(S_{12}, S_{22})$
- Both ϕ and ψ preserve behavior in a strong sense

Non-dependent Translations

- Interesting parts deal with casts and contracts
- Everything else is translated homomorphically
- $\phi : \lambda_C \to \lambda_H$
  - Mimic two blame labels - $\phi(\langle c \rangle^{l,l'}) = \lambda x : \lceil c \rceil . \langle \phi(c) \Rightarrow \lceil c \rceil \rangle^{l'}(\langle \lceil c \rceil \Rightarrow \phi(c) \rangle^{l})$
  - Descend into contract predicates - $\phi(\{x : B \mid t\}) = \{x : B \mid \phi(t)\}$
  - Split function contracts - $\phi(c_1 \longmapsto c_2) = \phi(c_1) \to \phi(c_2)$
- $\psi : \lambda_H \to \lambda_C$
  - Duplication of blame label - $\psi(\langle S_1 \Rightarrow S_2 \rangle^{l}) = \langle \psi(S_1, S_2) \rangle^{l,l}$
  - Use cast target predicate - $\psi(\{x : B \mid s_1\}, \{x : B \mid s_2\}) = \{x : B \mid \psi(s_2)\}$
  - Translate domain contravariantly - $\psi(S_{11} \to S_{12}, S_{21} \to S_{22}) = \psi(S_{21}, S_{11}) \longmapsto \psi(S_{12}, S_{22})$
- Both ϕ and ψ preserve behavior in a strong sense

Non-dependent Translations

- Interesting parts deal with casts and contracts
- Everything else is translated homomorphically
- $\phi : \lambda_C \to \lambda_H$
  - Mimic two blame labels - $\phi(\langle c \rangle^{l,l'}) = \lambda x : \lceil c \rceil . \langle \phi(c) \Rightarrow \lceil c \rceil \rangle^{l'}(\langle \lceil c \rceil \Rightarrow \phi(c) \rangle^{l})$
  - Descend into contract predicates - $\phi(\{x : B \mid t\}) = \{x : B \mid \phi(t)\}$
  - Split function contracts - $\phi(c_1 \longmapsto c_2) = \phi(c_1) \to \phi(c_2)$
- $\psi : \lambda_H \to \lambda_C$
  - Duplication of blame label - $\psi(\langle S_1 \Rightarrow S_2 \rangle^{l}) = \langle \psi(S_1, S_2) \rangle^{l,l}$
  - Use cast target predicate - $\psi(\{x : B \mid s_1\}, \{x : B \mid s_2\}) = \{x : B \mid \psi(s_2)\}$
  - Translate domain contravariantly - $\psi(S_{11} \to S_{12}, S_{21} \to S_{22}) = \psi(S_{21}, S_{11}) \longmapsto \psi(S_{12}, S_{22})$
- Both ϕ and ψ preserve behavior in a strong sense

All Is Well?

Being Picky

- Problem: picky λ_C can blame more than lax λ_C

Example (Abusive Contracts)

$$I = \{x : \text{Int} \mid \text{true}\} \qquad N = \{x : \text{Int} \mid \text{nonzero } x\}$$

$$c_1 \longmapsto c_2 = f : (N \longmapsto I) \longmapsto \{z : \text{Int} \mid f\ 0 = 0\}$$

- Using Lax λ_C

$$((\langle c_1 \longmapsto c_2 \rangle^{l,l'} \lambda f : \text{Int} \to \text{Int} \,.\, 0)\ \lambda x : \text{Int} \,.\, x)\ 5 \to_c^* 5$$

- Translation to λ_H - $\phi(f)$
  - Domain contract of f is effectively rechecked (due to cast)
  - Results in blame

Being Picky

- Problem: picky λ_C can blame more than lax λ_C

Example (Abusive Contracts)

$$I = \{x : \text{Int} \mid \text{true}\} \qquad N = \{x : \text{Int} \mid \text{nonzero } x\}$$

$$c_1 \longmapsto c_2 = f : (N \longmapsto I) \longmapsto \{z : \text{Int} \mid f\ 0 = 0\}$$

- Using Lax λ_C

$$(((\langle c_1 \longmapsto c_2 \rangle^{l,l'} \lambda f : \text{Int} \to \text{Int} \,.\, 0)\ \lambda x : \text{Int} \,.\, x)\ 5 \to_c^* 5$$

- Translation to λ_H - $\phi(f)$
  - Domain contract of f is effectively rechecked (due to cast)
  - Results in blame

Being Picky

- Problem: picky λ_C can blame more than lax λ_C

Example (Abusive Contracts)

$$I = \{x : \text{Int} \mid \text{true}\} \qquad N = \{x : \text{Int} \mid \text{nonzero } x\}$$

$$c_1 \longmapsto c_2 = f : (N \longmapsto I) \longmapsto \{z : \text{Int} \mid f\ 0 = 0\}$$

- Using Lax λ_C

$$((\langle c_1 \longmapsto c_2 \rangle^{l,l'} \lambda f : \text{Int} \to \text{Int} \,.\, 0)\ \lambda x : \text{Int} \,.\, x)\ 5 \to_c^*\ 5$$

- Translation to λ_H - $\phi(f)$
  - Domain contract of f is effectively rechecked (due to cast)
  - Results in blame

Being Picky

- Problem: picky λ_C can blame more than lax λ_C

Example (Abusive Contracts)

$$I = \{x : \text{Int} \mid \text{true}\} \qquad N = \{x : \text{Int} \mid \text{nonzero } x\}$$

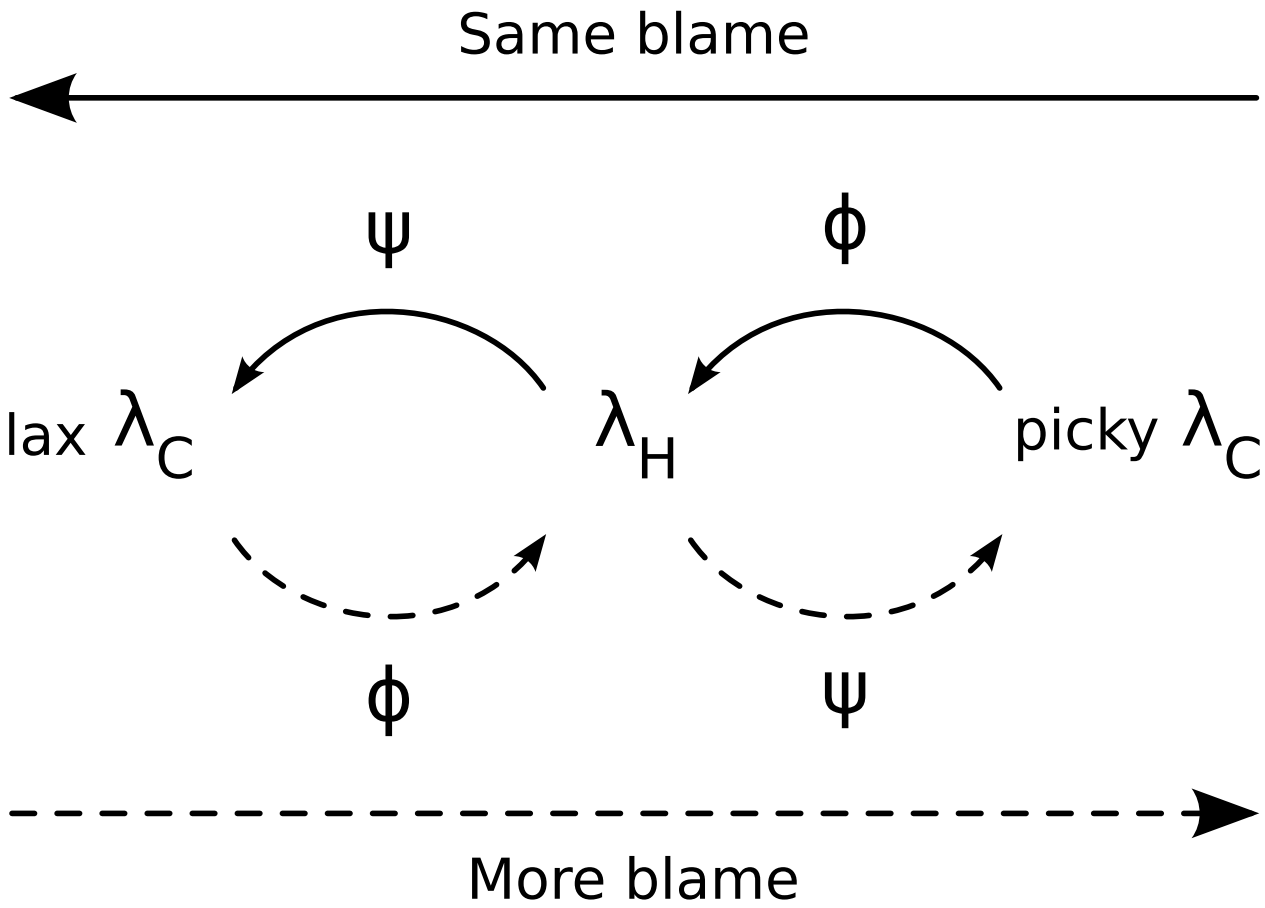
$$c_1 \longmapsto c_2 = f : (N \longmapsto I) \longmapsto \{z : \text{Int} \mid f\ 0 = 0\}$$

- Using Lax λ_C

$$((\langle c_1 \longmapsto c_2 \rangle^{l,l'} \lambda f : \text{Int} \to \text{Int} \,.\, 0)\ \lambda x : \text{Int} \,.\, x)\ 5 \to_c^*\ 5$$

- Translation to λ_H - $\phi(f)$
  - Domain contract of f is effectively rechecked (due to cast)
  - Results in blame

The Axis of Blame



Dependent λ_C Changes

- Not much – contracts and types barely interact
- Track blame labels throughout contract well-formedness check
- Choose picky or lax decomposition

Contracts and contexts

$$c ::= \{x{:}B \mid t\} \mid x{:}c_1 \mapsto c_2$$
$$\Gamma ::= \emptyset \mid \Gamma, x{:}T \mid \Gamma, x{:}c^{l,l'}$$

Operational Semantics

$$\frac{}{(\langle x{:}c_1 \mapsto c_2\rangle^{l,l'}\ v)\ v' \longrightarrow_c \langle c_2\{x := v'\}\rangle^{l,l'}\ (v\ (\langle c_1\rangle^{l',l}\ v'))}\ \text{E\_CDecompLax}$$

$$\frac{}{(\langle x{:}c_1 \mapsto c_2\rangle^{l,l'}\ v)\ v' \longrightarrow_c \langle c_2\{x := \langle c_1\rangle^{l',l}\ v'\}\rangle^{l,l'}\ (v\ (\langle c_1\rangle^{l',l}\ v'))}\ \text{E\_CDecompPicky}$$

Typing rules

$$\frac{x{:}T \in \Gamma}{\Gamma \vdash x : T}\ \text{T\_VarT} \qquad \frac{x{:}c^{l,l'} \in \Gamma}{\Gamma \vdash x : \lfloor c \rfloor}\ \text{T\_VarC}$$

$$\frac{\Gamma \vdash_c^{l,l'} c : T}{\Gamma \vdash \langle c\rangle^{l,l'} : T \to T}\ \text{T\_Contract}$$

$$\frac{\Gamma, x{:}B \vdash t : \mathsf{Bool}}{\Gamma \vdash_c^{l,l'} \{x{:}B \mid t\} : B}\ \text{T\_RefineC}$$

$$\frac{\Gamma \vdash_c^{l',l} c_1 : T_1 \qquad \Gamma, x{:}c_1^{\,l',l} \vdash_c^{l,l'} c_2 : T_2}{\Gamma \vdash_c^{l,l'} x{:}c_1 \mapsto c_2 : T_1 \to T_2}\ \text{T\_FunC}$$

Dependent λ_H Changes

Types

$$S \quad ::= \quad \{x{:}B \mid s\} \mid x{:}S_1 \to S_2$$

Operational semantics

$$\frac{}{\begin{array}{l}(\langle x{:}S_{11} \to S_{12} \Rightarrow x{:}S_{21} \to S_{22}\rangle^l\ w)\ w' \longrightarrow_h \\ \langle S_{12}\{x := \langle S_{21} \Rightarrow S_{11}\rangle^l\ w'\} \Rightarrow S_{22}\{x := w'\}\rangle^l \\ \quad (w\ (\langle S_{21} \Rightarrow S_{11}\rangle^l\ w'))\end{array}}\ \text{F\_CDecomp}$$

Typing rules

$$\frac{\Delta \vdash s_1 : (x{:}S_1 \to S_2) \qquad \Delta \vdash s_2 : S_1}{\Delta \vdash s_1\ s_2 : S_2\{x := s_2\}}\ \text{S\_App}$$

$$\frac{\Delta, x{:}\{x{:}B \mid \mathsf{true}\} \vdash s : \{x{:}\mathsf{Bool} \mid \mathsf{true}\}}{\Delta \vdash \{x{:}B \mid s\}}\ \text{SWF\_Refine}$$

$$\frac{\Delta \vdash S_1 \qquad \Delta, x{:}S_1 \vdash S_2}{\Delta \vdash x{:}S_1 \to S_2}\ \text{SWF\_Fun}$$

$$\frac{\Delta, x{:}\{x{:}B \mid \mathsf{true}\} \vdash s_1 \supset s_2}{\Delta \vdash \{x{:}B \mid s_1\} <: \{x{:}B \mid s_2\}}\ \text{SSub\_Refine}$$

$$\frac{\Delta \vdash S_{21} <: S_{11} \qquad \Delta, x{:}S_{21} \vdash S_{12} <: S_{22}}{\Delta \vdash x{:}S_{11} \to S_{12} <: x{:}S_{21} \to S_{22}}\ \text{SSub\_Fun}$$

$$\frac{\forall \sigma.\ (\Delta \models \sigma \wedge \sigma(s_1) \longrightarrow_h^* \mathsf{true}) \ \text{ implies } \ \sigma(s_2) \longrightarrow_h^* \mathsf{true}}{\Delta \vdash s_1 \supset s_2}\ \text{S\_Imp}$$

Denotations of types and kinds

$$\begin{array}{rcl}
s \in [\![\{x{:}B \mid s_0\}]\!] & \iff & s \longrightarrow_h^* \Uparrow l \\
& & \vee \quad (\exists k \in \mathcal{K}_B.\ s \longrightarrow_h^* k \\
& & \quad \wedge\ s_0\{x := k\} \longrightarrow_h^* \mathsf{true}) \\
s \in [\![x{:}S_1 \to S_2]\!] & \iff & \forall q \in [\![S_1]\!].\ s\ q \in [\![S_2\{x := q\}]\!] \\
\{x{:}B \mid s\} \in [\![\star]\!] & \iff & \forall k \in \mathcal{K}_B. \\
& & \quad s\{x := k\} \in [\![\{x{:}\mathsf{Bool} \mid \mathsf{true}\}]\!] \\
x{:}S_1 \to S_2 \in [\![\star]\!] & \iff & S_1 \in [\![\star]\!] \\
& & \wedge\ \forall q \in [\![S_1]\!].\ S_2\{x := q\} \in [\![\star]\!]
\end{array}$$

Semantic judgments

$$\begin{array}{rcl}
 & & \forall \sigma \text{ s.t. } \Delta \models \sigma : \\
\Delta \models S_1 <: S_2 & \iff & [\![\sigma(S_1)]\!] \subseteq [\![\sigma(S_2)]\!] \\
\Delta \models s : S & \iff & \sigma(s) \in [\![\sigma(S)]\!] \\
\Delta \models S & \iff & \sigma(S) \in [\![\star]\!]
\end{array}$$

Closing substitutions

$$\frac{}{\emptyset \models \emptyset}\ \text{SCS\_Empty} \qquad \frac{s \in [\![S]\!] \qquad \Delta\{x := s\} \models \sigma}{x{:}S, \Delta \models \sigma\{x := s\}}\ \text{SCS\_Ext}$$

Example 1 - lax $\lambda_C \to \lambda_H$ (1/2)

Example

$$c_{11} \longmapsto c_{12} = f : (x : \{x : \mathrm{Int} \mid \textbf{true}\} \longmapsto \{y : \mathrm{Int} \mid \mathrm{nonzero}\ y\})$$

$$c_2 = \{z : \mathrm{Int} \mid f\,0 = 0\}$$

- $(\langle (f : c_{11} \longmapsto c_{12}) \longmapsto c_2 \rangle^{l,l'} (\lambda f.0))\ (\lambda x.0)$
- $\langle c_2\{f := \lambda x.0\}\rangle^{l,l'} ((\lambda f.0)\ (\langle c_{11} \longmapsto c_{12}\rangle^{l',l} (\lambda x.0)))$
- $\langle c_2\{f := \lambda x.0\}\rangle^{l,l'} 0$
- $(\lambda x.0)\ 0 = 0$

Example 1 - lax $\lambda_C \to \lambda_H$ (2/2)

Example

$$c_{11} \longmapsto c_{12} = f : (x : \{x : \mathrm{Int} \mid \textbf{true}\} \longmapsto \{y : \mathrm{Int} \mid \mathrm{nonzero}\ y\})$$

$$c_2 = \{z : \mathrm{Int} \mid f\,0 = 0\}$$

- $\phi(f : (c_{11} \longmapsto c_{12}) \longmapsto c_2) = f : \phi(x : c_{11} \longmapsto c_{12}) \to \phi(c_2)$
- $S_1 = x : \{x : \mathrm{Int} \mid \textbf{true}\} \to \{y : \mathrm{Int} \mid \mathrm{nonzero}\ y\}$
- $\lceil S_1 \rceil = \lceil \mathrm{Int} \rceil \to \lceil \mathrm{Int} \rceil$
- $f : S_1 \to (\lambda z : \lceil Int \rceil.((\underline{\langle S_1 \Rightarrow \lceil S_1 \rceil \rangle})\langle \lceil S_1 \rceil \Rightarrow S_1 \rangle\ f))\ 0 = 0$
- When $f = \lambda x.0$, the co-domain cast to nonzero fails

Example 2 - lax $\lambda_C \to \lambda_H$ (1/2)

Example

$$c_{11} \longmapsto c_{12} = f : (x : \{x : \mathrm{Int} \mid \mathrm{nonzero}\ x\} \longmapsto \{y : \mathrm{Int} \mid \mathbf{true}\})$$

$$c_2 = \{z : \mathrm{Int} \mid f\,0 = 0\}$$

- $(\langle (f : c_{11} \longmapsto c_{12}) \longmapsto c_2 \rangle^{l,l'} (\lambda f.0))\ (\lambda x.0)$
- $\langle c_2\{f := \lambda x.0\} \rangle^{l,l'} ((\lambda f.0)\ (\langle c_{11} \longmapsto c_{12} \rangle^{l',l} (\lambda x.0)))$
- $\langle c_2\{f := \lambda x.0\} \rangle^{l,l'} 0$
- $(\lambda x.0)\ 0 = 0$

Example 2 - lax $\lambda_C \to \lambda_H$ (2/2)

Example

$$c_{11} \longmapsto c_{12} = f : (x : \{x : \mathrm{Int} \mid \mathrm{nonzero}\ x\} \longmapsto \{y : \mathrm{Int} \mid \textbf{true}\})$$

$$c_2 = \{z : \mathrm{Int} \mid f\,0 = 0\}$$

- $\phi(f : (c_{11} \longmapsto c_{12}) \longmapsto c_2) = f : \phi(x : c_{11} \longmapsto c_{12}) \to \phi(c_2)$
- $S_1 = x : \{x : \mathrm{Int} \mid \mathrm{nonzero}\ x\} \to \{y : \mathrm{Int} \mid \textbf{true}\}$
- $\lceil S_1 \rceil = \lceil \mathrm{Int} \rceil \to \lceil \mathrm{Int} \rceil$
- $f : S_1 \to (\lambda z : \lceil Int \rceil.((\langle S_1 \Rightarrow \lceil S_1 \rceil \rangle)\underline{\langle \lceil S_1 \rceil \Rightarrow S_1 \rangle}\ f))\ 0 = 0$
- When $f = \lambda x.0$, the domain cast to nonzero fails

Example 3 - $\lambda_H \to$ picky λ_C (1/2)

Example

$$S_1 = f : (x : \lceil \text{Int} \rceil \to \{y : \text{Int} \mid \text{nonzero } y\}) \to \lceil \text{Int} \rceil$$

$$S_2 = f : (x : \lceil \text{Int} \rceil \to \lceil \text{Int} \rceil) \to \{z : \text{Int} \mid f\ 0 = 0\}$$

- $S_{11} = \lceil \text{Int} \rceil \to \{y : \text{Int} \mid \text{nonzero } y\}$ $\quad S_{12} = \lceil \text{Int} \rceil$
- $w = (\lambda f : S_{11} \to S_{12}\ .\ 0)$ $\quad w' = (\lambda x : \lceil \text{Int} \rceil\ .\ 0)$
- $S_{21} = \lceil \text{Int} \rceil \to \lceil \text{Int} \rceil$ $\quad S_{22} = \{z : \text{Int} \mid f\ 0 = 0\}$
- $(\langle S_1 \Rightarrow S_2 \rangle w)\ w' = \langle S_{12} \Rightarrow S_{22} \rangle\ (w\ (\underline{\langle S_{21} \Rightarrow S_{11} \rangle}\ w'))$
  - Co-domain of S_{11} not enforced!
- $S_{22}\{f := w'\} \to_h^* ((w'\ 0) = 0) \to_h^*$ **true**

Example 3 - $\lambda_H \to$ picky λ_C (2/2)

Example

$$S_1 = f : (x : \lceil \text{Int} \rceil \to \{y : \text{Int} \mid \text{nonzero } y\}) \to \lceil \text{Int} \rceil$$

$$S_2 = f : (x : \lceil \text{Int} \rceil \to \lceil \text{Int} \rceil) \to \{z : \text{Int} \mid f\ 0 = 0\}$$

- $S_{11} = \lceil \text{Int} \rceil \to \{y : \text{Int} \mid \text{nonzero } y\}$ $\quad$ $S_{22} = \{z : \text{Int} \mid f\ 0 = 0\}$
- $c = \psi(S_1, S_2) = \psi(S_{11} \to S_{12}, S_{21} \to S_{22})$
- $f : \psi(S_{21}, \underline{S_{11}}) \longmapsto \psi(S_{12}\{f := \langle S_{21} \Rightarrow S_{11} \rangle\ f\}, \underline{S_{22}})$
- $c_1 = \psi(S_{11})$ $\quad$ $c_2 = \psi(S_{22})$
- $(\langle c_1 \longmapsto c_2 \rangle (\lambda f : \text{Int} \,.\, 0))\ (\lambda x : \text{Int} \,.\, 0)$
- E_DECOMP_PICKY: $c_2\{f := \langle c_1 \rangle v'\}$ where $v' = (\lambda x : \text{Int} \,.\, 0)$

Big Picture

- ϕ and ψ correspond exactly in the non-dependent case
- First-order dependence does not change this
  - Contracts can't be abusive
- Higher-order dependence
  - ϕ from lax λ_C can blame more
  - ψ to picky λ_C can blame more
- Caveats
  - No recursion
  - No side-effects (other than blame)
  - No contracts or refinements over functions
  - Constraints on types
    - $ty_h(k) <: \lceil ty_c(k) \rceil$
    - `sqrt` must be subtype of $\lceil \mathrm{Float} \to \mathrm{Float} \rceil$
    - Domain must be defined for all $\mathcal{K}_{\mathrm{float}}$!

Dependent Translation into λ_H (1/2)

ϕ from dependent λ_C to dependent λ_H

Terms

$$
\begin{array}{rcl}
\phi(\Gamma_1, x{:}T\,,\,\Gamma_2 \vdash x : T) & = & x \\
\phi(\Gamma_1, x{:}c^{l,l'}\,,\,\Gamma_2 \vdash x : \lfloor c \rfloor) & = & \langle \phi(\Gamma_1 \vdash_c^{l,l'} c : \lfloor c \rfloor) \Rightarrow \lceil c \rceil \rangle^{l'}\, x \\
\phi(\Gamma \vdash k : T) & = & k \\
\phi(\Gamma \vdash \lambda x{:}T_1.\, t_2 : T_1 \to T_2) & = & \lambda x{:}\lceil T_1 \rceil.\, \phi(\Gamma, x{:}T_1 \vdash t_2 : T_2) \\
\phi(\Gamma \vdash t_1\, t_2 : T_2) & = & \phi(\Gamma \vdash t_1 : T_1 \to T_2)\, \phi(\Gamma \vdash t_2 : T_1) \\
\phi(\Gamma \vdash \Uparrow l : T) & = & \Uparrow l \\
\phi(\emptyset \vdash \langle c, t, k \rangle^l : B) & = & \langle \phi(\emptyset \vdash_c^{l,l'} c : B), \phi(\emptyset \vdash t : \mathsf{Bool}), k \rangle^l \\
\phi(\Gamma \vdash \langle c \rangle^{l,l'} : T) & = & \lambda x{:}\lceil c \rceil.\, \langle \phi(\Gamma \vdash_c^{l',l} c : T) \Rightarrow \lceil c \rceil \rangle^{l'} \left(\langle \lceil c \rceil \Rightarrow \phi(\Gamma \vdash_c^{l,l'} c : T) \rangle^{l}\, x \right) \quad \text{where } x \text{ is fresh}
\end{array}
$$

Types

$$
\begin{array}{rcl}
\phi(\Gamma \vdash_c^{l,l'} \{x{:}B \mid t\} : B) & = & \{x{:}B \mid \phi(\Gamma, x{:}B \vdash t : \mathsf{Bool})\} \\
\phi(\Gamma \vdash_c^{l,l'} x{:}c_1 \mapsto c_2 : T_1 \to T_2) & = & x{:}\phi(\Gamma \vdash_c^{l',l} c_1 : T_1) \to \phi(\Gamma, x{:}{c_1}^{l',l} \vdash_c^{l,l'} c_2 : T_2)
\end{array}
$$

Contexts

$$
\begin{array}{rcl}
\phi(\vdash \emptyset) & = & \emptyset \\
\phi(\vdash \Gamma, x{:}T) & = & \phi(\vdash \Gamma), x{:}\lceil T \rceil \\
\phi(\vdash \Gamma, x{:}c^{l,l'}) & = & \phi(\vdash \Gamma), x{:}\phi(\Gamma \vdash_c^{l,l'} c : \lfloor c \rfloor)
\end{array}
$$

Result correspondence

$$k \approx k : B \iff k \in \mathcal{K}_B$$

$$v \approx w : T_1 \to T_2 \iff \forall t \sim s : T_1.\, v\, t \sim w\, s : T_2$$

$$\Uparrow l \approx \Uparrow l : T$$

Term correspondence

$$t \sim s : T \iff t \longrightarrow_c^* r \wedge s \longrightarrow_h^* q \wedge r \approx q : T$$

Dependent Translation into λ_H (2/2)

ϕ from lax λ_C

Value correspondence

$$k \approx_\succ k : B \iff k \in \mathcal{K}_B$$

$$v \approx_\succ w : T_1 \to T_2 \iff \forall t \sim_\succ s : T_1.\, v\, t \sim_\succ w\, s : T_2$$

Term correspondence

$$t \sim_\succ s : T \iff$$
$$s \longrightarrow_h^* \Uparrow l \vee (t \longrightarrow_c^* v \wedge s \longrightarrow_h^* w \wedge v \approx_\succ w : T)$$

Contract / type correspondence

$$\{x{:}B \mid t\} \sim_\succ \{x{:}B \mid s\} : B \iff$$
$$\forall k \in \mathcal{K}_B.\, t\{x := k\} \sim_\succ s\{x := k\} : \mathsf{Bool}$$

$$x{:}c_1 \mapsto c_2 \sim_\succ x{:}S_1 \to S_2 : T_1 \to T_2 \iff$$
$$c_1 \sim_\succ S_1 : T_1 \wedge$$
$$\forall t \sim_\succ s : T_1.\, c_2\{x := t\} \sim_\succ S_2\{x := s\} : T_2$$

Dual closing substitutions

$$\Gamma \models_\succ \delta \iff \forall x \in \mathrm{dom}(\Gamma).\, \delta_1(x) \sim_\succ \delta_2(x) : \lfloor \Gamma(x) \rfloor$$

ϕ from picky λ_C

Contract / type correspondence

$$\{x{:}B \mid t\} \sim^{l,l'} \{x{:}B \mid s\} : B \iff$$
$$\forall k \in \mathcal{K}_B.\, t\{x := k\} \sim s\{x := k\} : \mathsf{Bool}$$

$$x{:}c_1 \mapsto c_2 \sim^{l,l'} x{:}S_1 \to S_2 : T_1 \to T_2 \iff$$
$$c_1 \sim^{l',l} S_1 : T_1 \wedge$$
$$\forall t \sim s : T_1.$$
$$c_2\{x := \langle c_1 \rangle^{l',l}\, t\} \sim^{l,l'} S_2\{x := \langle \lceil S_1 \rceil \Rightarrow S_1 \rangle^{l'}\, s\} : T_2$$

Dual closing substitutions

$$\Gamma \models \delta \iff \begin{cases} \forall x{:}T \in \Gamma.\, \delta_1(x) \sim \delta_2(x) : T \\ \forall x{:}c^{l,l'} \in \Gamma.\, \delta_1(x) = \langle \delta_1(c) \rangle^{l,l'}\, t \\ \qquad \delta_2(x) = \langle \lceil c \rceil \Rightarrow \delta_2(S) \rangle^{l}\, s \\ \qquad \text{s.t. } S = \phi(\Gamma \vdash_c^{l,l'} c : \lfloor c \rfloor) \\ \qquad \wedge\ t \sim s : \lfloor c \rfloor \end{cases}$$

Dependent Translation into λ_C (ψ)

ψ mapping λ_H to λ_C

Term translation

$$\begin{array}{rcl@{\qquad}rcl}
\psi(x) & = & x & \psi(k) & = & k \\
\psi(\Uparrow l) & = & \Uparrow l & \psi(\langle S_1 \Rightarrow S_2 \rangle^l) & = & \langle \psi^l(S_1, S_2) \rangle^{l,l} \\
\multicolumn{3}{r@{\qquad}}{\psi(\lambda x{:}S.\ s) = \lambda x{:}\lfloor S \rfloor.\ \psi(s)} & & & \\
\multicolumn{3}{r@{\qquad}}{\psi(s_1\ s_2) = \psi(s_1)\,\psi(s_2)} & & & \\
\multicolumn{3}{r@{\qquad}}{\psi(\langle \{x{:}B \mid s_1\}, s_2, k \rangle^l) = \langle \{x{:}B \mid \psi(s_1)\}, \psi(s_2), k \rangle^l} & & &
\end{array}$$

Type-to-contract translation

$$\begin{array}{rcl}
\psi^l(\{x{:}B \mid s_1\}, \{x{:}B \mid s_2\}) & = & \{x{:}B \mid \psi(s_2)\} \\
\psi^l(x{:}S_{11} \to S_{12}, x{:}S_{21} \to S_{22}) & = & \\
\multicolumn{3}{c}{x{:}\psi^l(S_{21}, S_{11}) \mapsto \psi^l(S_{12}\{x := \langle S_{21} \Rightarrow S_{11} \rangle^l\, x\}, S_{22})}
\end{array}$$

ψ into lax λ_C

Contract / type correspondence

$$\{x{:}B \mid t\} \sim \{x{:}B \mid s_1\} \Rightarrow^l \{x{:}B \mid s_2\} : B \iff \forall k \in \mathcal{K}_B.\ t\{x := k\} \sim s_2\{x := k\} : \mathsf{Bool}$$

$$\begin{array}{c}
x{:}c_1 \mapsto c_2 \sim x{:}S_{11} \to S_{12} \Rightarrow^l S_{21} \to S_{22} : T_1 \to T_2 \iff \\
c_1 \sim S_{21} \Rightarrow^l S_{11} : T_1 \quad \wedge \\
\forall t \sim s : T_1. \\
c_2\{x := t\} \sim S_{12}\{x := \langle S_{21} \Rightarrow S_{11} \rangle^l\, s\} \Rightarrow^l S_{22}\{x := s\} : T_2
\end{array}$$

ψ into picky λ_C

Contract / type correspondence

$$\{x{:}B \mid t\} \sim_\prec \{x{:}B \mid s_1\} \Rightarrow \{x{:}B \mid s_2\} : B \iff \forall k \in \mathcal{K}_B.\ t\{x := k\} \sim_\prec s_2\{x := k\} : \mathsf{Bool}$$

$$\begin{array}{c}
x{:}c_1 \mapsto c_2 \sim_\prec x{:}S_{11} \to S_{12} \Rightarrow x{:}S_{21} \to S_{22} : T_1 \to T_2 \\
\iff \\
c_1 \sim_\prec S_{21} \Rightarrow S_{11} : T_1 \wedge \\
\forall l. \forall t \sim_\prec s : T_1. \\
c_2\{x := t\} \sim_\prec S_{12}\{x := \langle S_{21} \Rightarrow S_{11} \rangle^l\, s\} \Rightarrow S_{22}\{x := s\} : T_2
\end{array}$$

Dual closing substitutions

$$\Gamma \models \delta \iff \forall x \in \mathrm{dom}(\Gamma).\ \delta_1(x) \sim_\prec \delta_2(x) : \lfloor \Gamma(x) \rfloor$$

Questions?