A Relational Framework for Higher-Order Shape Analysis

Meta-theory

Calculus λ_R

x, y, z, ν	$\in$	*variables*	
C	$::=$	$\mathsf{Cons} \mid \mathsf{Nil}$	*constructors*
v	$::=$	$\lambda(x:\tau).\, e \mid v:\tau \mid C\,\overline{v}$	*value*
e	$::=$	$x \mid e\,x \mid \texttt{let}\ x = e\ \texttt{in}\ e$ $\mid \texttt{match}\ x\ \texttt{with}\ C\,\overline{x} \Rightarrow e\,\texttt{else}\ e$ $\mid e:\tau$	*expression*
T	$::=$	$\mathsf{unit} \mid \mathsf{int} \mid \mathsf{intlist}$	*datatypes*
τ	$::=$	$\{\nu : T \mid \phi\} \mid x:\tau \to \tau$	*dependent types*

Specification Language

R	$\in$	*relation names*	
ϕ	$::=$	$r = r \mid r \subset r \mid \phi \wedge \phi \mid \phi \vee \phi \mid \top$	*type refinement*
r	$::=$	$R(x) \mid r \cup r \mid r \times r$	*relational expression*
Σ_R	$::=$	$\langle R, \tau_R, \overline{C\,\overline{x} \Rightarrow r}\rangle$ $\mid \langle R, \tau_R, R^*\rangle$	*relation definition*
θ	$::=$	$T \mid T * \theta$	*tuple sort*
τ_R	$::=$	$\mathsf{intlist} :\to \{\theta\}$	*relation sort*

Figure 1: Language

1. Core Language Metatheory

To simplify meta-theory, we use a calculus with propositional atoms as our intermediate language. We refer to the language as λ_ϕ . The language is described in Figure 3.

For a λ_R type environment Γ, we define its λ_ϕ projection (denoted Γ^F) as following:

$$\begin{array}{lcl} (\cdot)^F & = & \cdot \\ (\Gamma,\, x : \{\nu : T \mid \phi\})^F & = & \Gamma^F,\, x : \mathcal{F}(T) \\ (\Gamma,\, R :: \tau_R)^F & = & \Gamma^F,\, R : [\![\tau_R]\!]_L \end{array}$$

LEMMA 1.1. *If* $\Gamma^F \vdash \nu_1^F : \tau^F$ *and* $\Gamma^F \vdash \nu_2^F : \tau^F$*, then* $\gamma_\sqcup(\nu_1^F, \odot, \nu_2^F)$ *is a* λ_ϕ *value* ν^F *such that* $\Gamma^F \vdash \nu^F : \tau^F$ *, where* $\odot \in \{\vee, \Rightarrow, \Leftrightarrow\}$

Proof By structural induction over τ^F.

- Case bool : By inversion on type derivation of ν_1^F and ν_2^F, we know that ν_1^F, and ν_2^F are λ_ϕ propositions ϕ_1^F and ϕ_2^F respectively. From the definition, $\gamma_\sqcup(\phi_1^F, \odot, \phi_2^F) = \phi^F \odot \phi^F$, where $\odot \in \{\vee, \Rightarrow, \Leftrightarrow\}$. Proof follows from the type rules of λ_ϕ for propositions.
- Case $T^F \to \tau_1^F$: By inversion on type derivations of ν_1^F and ν_2^F, we know that

$$\nu_1^F = \lambda(k : T^F).\, e_1, \text{ and } \nu_2^F = \lambda(k : T^F).\, e_2,$$

for some k, e_1, e_2, such that

$$\Gamma^F,\, k : T^F \vdash e_1 : \tau_1^F \text{ and } \Gamma^F,\, k : T^F \vdash e_2 : \tau_1^F.$$

Now, by structural induction on ν_1^F and ν_2^F, and eliminating inconsistent cases, we have the following inductive hypothesis (IH):

$$\gamma_\sqcup(e_1, \odot, e_2) \text{ is a value } \nu_k^F \text{, and } \quad \Gamma^F,\, k : T^F \vdash \nu_k^F : \tau_1^F$$

From first conjunct of IH: $\gamma_\sqcup(\nu_1^F, \odot, \nu_2^F) = \lambda(k : T^F).\nu_k^F$ is a value.

From the second conjunct of IH: Observing that this is the premise of type rules for functions in λ_ϕ , we conclude that:

$$\Gamma^F \vdash \lambda(k : T^F).\gamma_\sqcup(e_1, \odot, e_2) : T^F \to \tau_1^F$$

Definition ***(JoinType)*** $JoinType$ of τ_1^F and τ_2^F is defined by structural recursion on the type structure:

- $JoinType$(bool,bool) = bool
- $JoinType$(bool,$T^F \to \tau^F$) = $T^F \to JoinType$(bool,τ^F)
- $JoinType$($T^F \to \tau_1^F$,τ_2^F) = $T^F \to JoinType(\tau_1^F, \tau_2^F)$

LEMMA 1.2. *If* $\Gamma^F \vdash \nu_1^F : \tau_1^F$ *and* $\Gamma^F \vdash \nu_2^F : \tau_2^F$*, then* $\gamma_\bowtie(\nu_1^F, \wedge, \nu_2^F)$ *is a* λ_ϕ *value* ν^F*, such that* $\Gamma^F \vdash \nu^F : JoinType(\tau_1^F, \tau_2^F)$

Proof By induction on the structure of ν_1^F and ν_2^F, followed by inversion on their typing derivations. ∎

LEMMA 1.3. *if* $\Gamma \vdash R :: \mathsf{intlist} :\to \{\theta\}$ *then,* $\Gamma^F \vdash [\![R]\!]_L : A_1 \to [\![\theta]\!]_L \to$ *bool*

Proof $[\![\mathsf{intlist} :\to \{\theta\}]\!]_L \stackrel{def}{=} [\![\mathsf{intlist}]\!]_L \to [\![\theta]\!]_L \to bool$.
Since $\mathcal{F}$(intlist) = A_1, we have that:
$[\![R]\!]_L = [\![R :: \mathsf{intlist} :\to \{\theta\}]\!]_L$
$= \eta_{wrap}(R,\ [\![\mathsf{intlist} :\to \{\theta\}]\!]_L)$
$= \eta_{wrap}(R,\ A_1 \to [\![\{\theta\}]\!]_L \to bool)$.
But, when $\Gamma^F \vdash \phi^F : \tau^F$, then $\Gamma^F \vdash \eta_{wrap}(\phi^F, \tau^F) : \tau^F$.
Therefore, $\Gamma^F \vdash [\![R]\!]_L : A_1 \to [\![\theta]\!]_L \to bool$ ∎

LEMMA 1.4. *If* $\Gamma \vdash r : \theta$*, then* $\Gamma^F \vdash [\![r]\!]_L : [\![\theta]\!]_L \to$ *bool*

Proof By induction on the sort derivation. Cases:

- $r = R(x)$, where

$$\Gamma \vdash R :: \mathsf{intlist} :\to \{\theta\} \qquad \Gamma \vdash x : \mathsf{intlist}.$$

From Lemma 1.3, we know that

Sort Checking Specification Language $\boxed{\Gamma \vdash r :: \{\theta\}}$

S-REL
$$\frac{(R :: \tau_R) \in \Gamma}{\Gamma \vdash R :: \tau_R}$$

S-APP
$$\frac{\Gamma \vdash R :: T :\rightarrow \{\theta\} \quad \Gamma \vdash x : T}{\Gamma \vdash R(x) :: \{\theta\}}$$

S-UNION
$$\frac{\Gamma \vdash r_1 :: \{\theta\} \quad \Gamma \vdash r_2 :: \{\theta\}}{\Gamma \vdash r_1 \cup r_2 :: \{\theta\}}$$

S-CROSS
$$\frac{\Gamma \vdash r_1 :: \{\theta_1\} \quad \Gamma \vdash r_2 :: \{\theta_2\}}{\Gamma \vdash r_1 \times r_2 :: \{\theta_1 * \theta_2\}}$$

Well-Formedness $\boxed{\Gamma \vdash r, \quad \Gamma \vdash \phi, \quad \Gamma \vdash \tau}$

WF-RPRED
$$\frac{\odot \in \{=, \subset\} \qquad \Gamma \vdash r_1 :: \theta \qquad \Gamma \vdash r_2 :: \theta}{\Gamma \vdash r_1 \odot r_2}$$

WF-REF
$$\frac{\odot \in \{\wedge, \vee, \Rightarrow\} \qquad \Gamma \vdash \phi_1 \qquad \Gamma \vdash \phi_2}{\Gamma \vdash \phi_1 \odot \phi_2}$$

WF-BASE
$$\frac{\Gamma, \nu : T \vdash \phi}{\Gamma \vdash \{\nu : T \,|\, \phi\}}$$

WF-FUN
$$\frac{\Gamma \vdash \tau_1 \quad \Gamma, x : \tau_1 \vdash \tau_2}{\Gamma \vdash x : \tau_1 \rightarrow \tau_2}$$

Figure 2: Static semantics of λ_R specfication language

Typed Calculus with Propositions (λ_ϕ)

$$\begin{array}{lcl} x & \in & \lambda_R \; variable \qquad R \;\in\; \lambda_R \; relation \\ j,k & \in & bound \; variable \end{array}$$

$$\begin{array}{lcll} v & ::= & x \mid k \mid j \mid R & variable \\ \phi^F & ::= & v \mid v = v \mid \phi^F \; \phi^F \mid \phi^F \Leftrightarrow \phi^F \mid v : \tau^F & atom \\ & \mid & \phi^F \Rightarrow \phi^F \mid \phi^F \vee \phi^F \mid \phi^F \wedge \phi^F & \\ \nu^F & ::= & \lambda(k : T^F).\, \nu^F \mid \phi^F & value \\ e & ::= & \nu^F \mid e \; e & expression \\ T^F & ::= & A \mid bool & base \; type \\ \tau^F & ::= & bool \mid T^F \rightarrow \tau^F & type \end{array}$$

MSFOL

$$\begin{array}{lcll} \phi^L & ::= & \forall (k : T^F).\, \phi^L \mid \phi^F \mid \phi^L \wedge \phi^L & proposition \\ & \mid & \phi^L \vee \phi^L \mid \phi^L \Rightarrow \phi^L & \end{array}$$

Auxiliary Functions

$$\begin{array}{lcl} Eval & : & e \rightarrow \nu^F \\ \mathcal{F} & : & T \rightarrow A \\ \eta_{wrap} & : & \{\phi^F\} \times \{\tau^F\} \rightarrow \{\nu^F \,|\, \nu^F \equiv_\eta \phi^F\} \\ \eta_{wrap}(\phi^F, \tau_1^F \rightarrow \tau_2^F) & = & \lambda(k : \tau_1^F).\, \eta_{wrap}(\phi^F \; k, \tau_2^F) \\ \eta_{wrap}(\phi^F, T^F) & = & \phi^F \end{array}$$

Semantics of Relational Expressions $\boxed{[\![r]\!]_L}$

$$\begin{array}{lcl} [\![T]\!]_L & \stackrel{def}{=} & \mathcal{F}(T) \\ [\![T * \theta]\!]_L & \stackrel{def}{=} & [\![T]\!]_L \rightarrow [\![\theta]\!]_L \\ [\![T :\rightarrow \{\theta\}]\!]_L & \stackrel{def}{=} & [\![T]\!]_L \rightarrow [\![\theta]\!]_L \rightarrow bool \\ [\![RId\}]\!]_L & \stackrel{def}{=} & \lambda(j : [\![\,\texttt{int}\,]\!]_L). \\ & & \quad \lambda(k : [\![\,\texttt{int}\,]\!]_L).\, j = k \\ [\![R :: \tau_R]\!]_L & \stackrel{def}{=} & \eta_{wrap}(R, [\![\tau_R]\!]_L) \end{array}$$

$$\begin{array}{lcl} [\![R(x)]\!]_L & \stackrel{def}{=} & Eval([\![R]\!]_L \; x) \\ [\![r_1 \cup r_2]\!]_L & \stackrel{def}{=} & \gamma_\sqcup([\![r_1]\!]_L, \vee, [\![r_2]\!]_L) \\ [\![r_1 \times r_2]\!]_L & \stackrel{def}{=} & \gamma_\bowtie([\![r_1]\!]_L, \wedge, [\![r_2]\!]_L) \\ \gamma_\sqcup(\lambda(k : T^F).\, e_1, \odot, \lambda(k : T^F).\, e_2) & \hookrightarrow & \lambda(k : T^F).\, \gamma_\sqcup(e_1, \odot, e_2) \\ \gamma_\sqcup(\phi_1^F, \odot, \phi_2^F) & \hookrightarrow & \phi_1^F \odot \phi_2^F \\ \gamma_\bowtie(\lambda j : T_j^F.\, \phi_1^F, \odot, \overline{\lambda k : T_k^F}.\, \phi_1^F) & \hookrightarrow & \lambda(j : T_j^F).\, \overline{\lambda(k : T_k^F)}.\, \phi_1^F \odot \phi_2^F \end{array}$$

Semantics of Type Refinements $\boxed{[\![\phi]\!]_L}$

$$\begin{array}{lcl} [\![r_1 = r_2]\!]_L & \stackrel{def}{=} & \gamma_\sqcup([\![r_1]\!]_L, \Leftrightarrow, [\![r_2]\!]_L) \\ [\![r_1 \subset r_2]\!]_L & \stackrel{def}{=} & \gamma_\sqcup([\![r_1]\!]_L, \Rightarrow, [\![r_2]\!]_L) \\ [\![\phi_1 \wedge \phi_2]\!]_L & \stackrel{def}{=} & \gamma_\bowtie([\![\phi_1]\!]_L, \wedge, [\![\phi_2]\!]_L) \end{array}$$

$$\begin{array}{lcl} [\![\phi_1 \vee \phi_2]\!]_L & \stackrel{def}{=} & \gamma_\sqcup([\![\phi_1]\!]_L, \vee, [\![\phi_2]\!]_L) \\ \gamma_\forall(\lambda(k : T^F).\, \nu^F) & \hookrightarrow & \forall (k : T^F).\, \gamma_\forall(\nu^F) \\ \gamma_\forall(\phi^F) & \hookrightarrow & \phi^F \end{array}$$

Figure 3: Semantics of Specification Language

$$\Gamma^F \vdash [\![R]\!]_L : A_1 \rightarrow [\![\theta]\!]_L \rightarrow bool \quad \Gamma^F \vdash x : A_1$$

From function application type rule of λ_ϕ , we have:

$$\Gamma^F \vdash Eval([\![R]\!]_L \; x) : [\![\theta]\!]_L \rightarrow bool$$

- $r = r_1 \cup r_2$, where

$$\Gamma \vdash r_1 :: \theta, \text{and} \quad \Gamma \vdash r_1 :: \theta$$

Inductive hypothesis (IH):

$$\Gamma^F \vdash [\![r_1]\!]_L : \tau^F, \text{and} \quad \Gamma^F \vdash [\![r_2]\!]_L : \tau^F.$$

Where, $\tau^F = [\![\theta]\!]_L \rightarrow bool$.

We know that $[\![r_1 \cup r_2]\!]_L \stackrel{def}{=} \gamma_\sqcup([\![r_1]\!]_L, \vee, [\![r_2]\!]_L)$. From lemma 1.1, proof for this case follows.

- $r = r_1 \times r_2$, where

$$\Gamma \vdash r_1 :: \theta_1, \quad \Gamma \vdash r_1 :: \theta_2, \text{and } \theta = \theta_1 * \theta_2$$

Inductive hypothesis (IH):

$$\Gamma^F \vdash [\![r_1]\!]_L : \tau_1^F, \text{and} \quad \Gamma^F \vdash [\![r_2]\!]_L : \tau_2^F.$$

Where, $\tau_1^F = [\![\theta_1]\!]_L \rightarrow bool$, and $\tau_2^F = [\![\theta_2]\!]_L \rightarrow bool$.

We know that $[\![r_1 \times r_2]\!]_L \stackrel{def}{=} \gamma_\sqcup([\![r_1]\!]_L, \wedge, [\![r_2]\!]_L)$. From Lemma 1.2, we have:

$$\Gamma^F \vdash [\![r]\!]_L : JoinType(\tau_1^F, \tau_2^F) \Rightarrow$$
$$\Gamma^F \vdash [\![r]\!]_L : [\![\theta_1]\!]_L \rightarrow [\![\theta_2]\!]_L \rightarrow bool \Rightarrow$$

It remains to prove that $[\![\theta_1 * \theta_2]\!]_L = [\![\theta_1]\!]_L \rightarrow [\![\theta_2]\!]_L$, which is trivial to establish by structural induction on θ_1. ∎

LEMMA 1.5. *If $\Gamma \vdash \phi$, then $[\![\phi]\!]_L$ is an λ_ϕ value.*

Calculus $\lambda_{\forall R}$

$$
\begin{array}{lcl}
t & \in & tuple-sort\ variables \qquad x, y, k \ \in \ variables \\
\mathtt{'a}, \mathtt{'b} & \in & type\ variables
\end{array}
$$

$$
\begin{array}{lcll}
\nu & ::= & C\,\overline{T}\,\overline{\theta\,\mathcal{R}}\,\overline{\nu} \mid \Lambda a.\, e & value \\
 & \mid & \mathring{\Lambda} t.\Lambda(R :: a :\rightarrow t).\, e \mid \ \ldots & \\
e & ::= & e\,T \mid e\,\theta\,\mathcal{R} \mid \mathring{\Lambda} t.\Lambda(R :: a :\rightarrow t).\, e & expression \\
 & \mid & \mathtt{match}\ x\ \mathtt{with}\ C\,\overline{T}\,\overline{\theta\,\mathcal{R}}\,\overline{x} \ \Rightarrow\ e & \\
 & & \quad \mathtt{else}\ e \mid \ \ldots & \\
Tt & ::= & T \mid t & \\
T & ::= & \mathtt{'a} \mid \mathtt{'a\,list} \mid \ \ldots & datatypes \\
\tau & ::= & \{\nu : T \,|\, \Phi\} \mid \ \ldots & dependent\ type \\
\delta & ::= & \forall t.\forall(R :: \mathtt{'a} :\rightarrow t).\, \delta \mid \tau & parametric\ type \\
\sigma & ::= & \forall \mathtt{'a}.\, \sigma \mid \delta & type\ scheme
\end{array}
$$

Specification Language

$$
\begin{array}{lcll}
\Phi & ::= & \rho = \rho \mid \rho \subset \rho \mid \Phi \wedge \Phi & type\ refinement \\
 & \mid & \Phi \vee \Phi \mid \top & \\
\rho & ::= & \mathcal{R}(x) \mid \rho \cup \rho \mid \rho \times \rho & rel.\ expression \\
\mathcal{R} & ::= & \mathcal{R}\,T \mid \mathcal{R}\,\theta\,\mathcal{R} \mid R & instantiation \\
\theta & ::= & t \mid t * \theta \mid \ \ldots & tuple\ sort \\
\tau_R & ::= & \forall t.\, (\mathtt{'a} :\rightarrow t) :\rightarrow (\mathtt{'a\,list} :\rightarrow \theta) & relation\ sort \\
 & \mid & \ldots & \\
\sigma_R & ::= & \forall a.\, \tau_R \mid \tau_R & sort\ scheme \\
\Sigma_R & ::= & \langle R,\, R_p,\, \sigma_R,\, \overline{C\,\overline{x} \Rightarrow r} \rangle & rel.\ definition \\
 & \mid & \langle R,\, R_p,\, \sigma_R,\, \mathcal{R}^* \rangle \mid \ \ldots &
\end{array}
$$

Figure 4: $\lambda_{\forall R}$ - Complete calculus with parametric relations

$$
\begin{array}{lcll}
r & ::= & R(x) \mid r \times r & \\
F_R & ::= & \lambda(\overline{x : T}).\, r & transformer \\
e_b & ::= & \mathtt{bind}\,(R(x), F_R) & bind\ expression \\
E_b & ::= & \lambda(x : T).\ \mathtt{bind}\,(R(x), F_R) & bind\ abstraction \\
\psi & ::= & R = E_b & bind\ equation \\
\Sigma_R^b & ::= & \lambda R.\, E_b & bind\ definition
\end{array}
$$

Figure 6: Bind Syntax

Proof By induction on the derivation of well-formedness judgement. Cases can be proved by straightforward application of Lemmas 1.4, 1.1 and 1.2.

LEMMA 1.6. *If* $\Gamma \vdash \phi$*, then* $\gamma_\forall([\![\phi]\!]_L)$ *is an MSFOL formula.*

Proof From lemma 1.5, we know that $[\![\phi]\!]_L$ is an λ_ϕ value ν^F. Next, by structural induction on ν^F, we prove that $\gamma_\forall(\nu^F)$ is an MSFOL formula.

LEMMA 1.7. (Completeness of semantics) *For every type refinement* ϕ*, if* $\Gamma \vdash \phi$*, then* $\mathtt{compile}\,(\Gamma, \phi)$ *terminates and produces an MSFOL formula.*

Proof From lemma 1.6

THEOREM 1.8. (**Decidability**) *Type checking in* λ_R *is decidable.*

Proof Follows from Lemma 1.7 and decidability proof of EPR logic. ∎

2. Parametric Language Meta-theory

2.1 Decidability of Type Checking

The full syntax of calculus ($\lambda_{\forall R}$) for parametric language is shown in Figure 4. The rules to rewrite $\lambda_{\forall R}$ type refinements to a conjunction of bind equations and non-parametric (λ_R) type refinements are shown in Figure 5.

THEOREM 2.1. (**Decidability**) *Type checking in* $\lambda_{\forall R}$ *is decidable.*

Proof Follows from 1. Completeness of our rewrite rules for well-formed refinements, 2. Completeness of encoding bind equations in MSFOL, 3. Decidability proof of EPR logic, to which bind equations are compiled to, and decidability result (Theorem 1.8) for λ_R. ∎

Kind Rules $\boxed{\Gamma \vdash r :: \delta_R}$

K-PARAM-REL

$$\frac{\Sigma^{\pi}_{R}(R) = E^{\pi}_{R} \quad \Gamma \vdash E^{\pi}_{R} :: \delta_R}{\Gamma \vdash R :: \delta_R}$$

K-BIND

$$\frac{\overline{T_2} = \theta_1 \qquad \Gamma, x : T_1 \vdash R(x) :: \{\theta_1\} \quad \Gamma, \overline{k : T_2} \vdash r :: \{\theta_2\}}{\Gamma \vdash \lambda(x : T_1).\ \texttt{bind}\,(R_1(x), \lambda(\overline{k : T_2}).\, r) :: T_1 :\to \theta_2}$$

K-INST

$$\frac{\Gamma \vdash \mathcal{R}_2 :: \tau_R \qquad \Gamma \vdash \mathcal{R}_1 :: \tau_R \to \tau_R}{\Gamma \vdash \mathcal{R}_1\, \mathcal{R}_2 :: \tau_R}$$

Rewrite Rules for Type Refinements $\boxed{\Phi \hookrightarrow \psi \wedge \phi}$

$$\psi \quad ::= \quad true \mid (R = E_b) \wedge \psi \quad \textit{bind equations}$$

RW-REF

$$\frac{\Phi_1 \hookrightarrow \psi_1 \wedge \phi_1 \quad \Phi_2 \hookrightarrow \psi_2 \wedge \phi_2 \quad \odot \in \{\wedge, \vee\}}{\Phi_1 \odot \Phi_2 \hookrightarrow (\psi_1 \wedge \psi_2) \wedge (\phi_1 \odot \phi_2)}$$

RW-REF-ATOM

$$\frac{\rho_1 \hookrightarrow \psi_1 \wedge r_1 \quad \rho_2 \hookrightarrow \psi_2 \wedge r_2 \quad \odot \in \{=, \subset\}}{\rho_1 \odot \rho_2 \hookrightarrow (\psi_1 \wedge \psi_2) \wedge (r_1 \odot r_2)}$$

RW-PARAM-REL

$$\frac{\Sigma^{\pi}_{R}(R) = E^{\pi}_{R}}{R \hookrightarrow E^{\pi}_{R}}$$

RW-KINST

$$\frac{\mathcal{R} \hookrightarrow \Lambda t.\, E^{\pi}_{R}}{\mathcal{R}\,\theta \hookrightarrow [\theta/t]\, E^{\pi}_{R}}$$

RW-RINST

$$\frac{\mathcal{R}_1 \hookrightarrow \lambda(R_1 :: \tau_R).\, E^{\pi}_{R} \quad \mathcal{R}_2 \hookrightarrow R_2}{\mathcal{R}_1\, \mathcal{R}_2 \hookrightarrow [R_2/R_1]\, E^{\pi}_{R}}$$

RW-RAPP

$$\frac{\mathcal{R} \hookrightarrow E_b \quad freshVar(R)}{E_b(x) \hookrightarrow (R = E_b) \wedge R(x)}$$

Semantics of Bind Equations $\boxed{[\![\psi]\!]_L}$

$$\begin{array}{rcl}
[\![R_2 = \lambda(x : T_1).\ \texttt{bind}\,(R_1(x), \lambda(\overline{k : T_2}).\, r)]\!]_L & \stackrel{def}{=} & \forall(x : [\![T_1]\!]_L).\ \gamma_{\Rightarrow}([\![R_1(x)]\!]_L,\ \forall(\overline{(k : [\![T_2]\!]_L)}[\![r]\!]_L,\ [\![R_2(x)]\!]_L) \\
 & \wedge & \forall(x : [\![T_1]\!]_L).\ \gamma_{\Leftarrow}([\![R_1(x)]\!]_L,\ \forall(\overline{k : [\![T_2]\!]_L})[\![r]\!]_L,\ [\![R_2(x)]\!]_L) \\
\gamma_{\Rightarrow}(\forall(\overline{k : T^F_1}).\phi^F_1,\ \forall(\overline{k : T^F_1}).\forall(\overline{j : T^F_2}).\phi^F_2,\ \nu^F) & \hookrightarrow & \forall(\overline{k : T^F_1}).\forall(\overline{j : T^F_2}).\ \phi^F_1 \wedge \phi^F_2 \Rightarrow \nu^F\,\overline{j} \\
\gamma_{\Leftarrow}(\forall(\overline{k : T^F_1}).\phi^F_1,\ \forall(\overline{k : T^F_1}).\forall(\overline{j : T^F_2}).\phi^F_2,\ \nu^F) & \hookrightarrow & \forall(\overline{j : T^F_2}).\exists(\overline{k : T^F_1}).\ \nu^F\,\overline{j} \Rightarrow \phi^F_1 \wedge \phi^F_2
\end{array}$$

Figure 5: Semantics of bind equations for parametric relations in $\lambda_{\forall R}$