CIRT: Calculus of Inductive Refinement Types

Realizability Semantics of Refinement Type Theory

Alessio Ferrarini[1,2] Niki Vazou[1] G. A. Kavvos[3]

[1]IMDEA Software Institute

[2]Universidad Politécnica de Madrid

[3]University of Bristol

Refinement Types 101

$$\tau \;::=\; \ldots \mid \{x\colon \tau \mid \phi\} \mid (x\colon \tau) \to \tau \mid \ldots$$

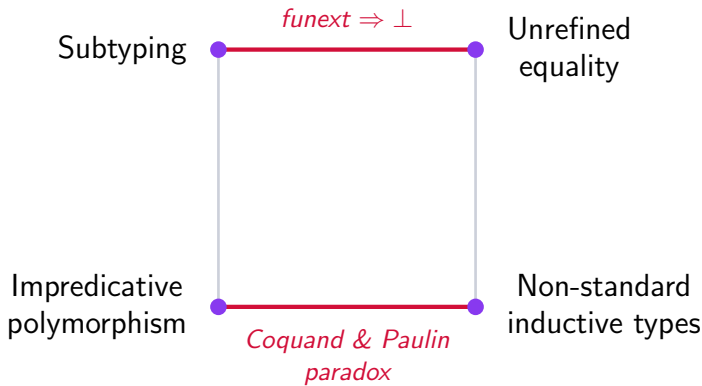
$$\phi \;::=\; \ldots \mid \quad t \equiv t \quad \mid \quad \phi \wedge \phi \quad \mid \ldots$$

```
{-@ div :: Int → {x: Int | x /= 0} → Int @-}
```

```
div 10 5 ✓     div 10 0 ✗     if x /= 0 then div 10 x else 0 ✓
```

The Four Horsemen of Refinement Types

Engineered first, proven "sound" after
(usually with syntactic methods)



Subtyping

$$\texttt{\{x: Int | x = 0\}} \preceq \texttt{\{x: Int | x >= 0\}}$$

$$\texttt{\{x: Int | x >= 0\}} \to \texttt{\{x: Int | x = 0\}} \preceq \texttt{\{x: Int | x = 0\}} \to \texttt{\{x: Int | x >= 0\}}$$

```
{-@ zero :: {x: Int | x = 0} @-}
{-@ f :: {x: Int | x >= 0} → Int @-}
f zero ✓
```

Unrefined equality

$$\texttt{\textbackslash x} \to \texttt{x + 1} \equiv \texttt{\textbackslash x} \to \texttt{1 + x} \checkmark$$

$$\texttt{\textbackslash x} \to \texttt{x} \equiv \texttt{\textbackslash x} \to \texttt{abs x} \times$$

```
-- even at {x: Int | x >= 0} → Int
```

Impredicative polymorphism

$$\frac{\Gamma, \alpha : \mathsf{Ty} \vdash \tau : \mathsf{Ty}}{\Gamma \vdash \forall \alpha : \mathsf{Ty}.\, \tau : \mathsf{Ty}}$$

Explicit and proof relevant

$1 \not\equiv 2$ $\quad$ `1, 2 :: Int` $\quad$ `Int :: Ty`

not like `SProp` (Rocq), `Prop` (Lean), nor `Prop` (Rocq)

Positive types

```
data Rou where
  Over :: Rou
  Next :: ((Rou → [Nat]) → [Nat]) → Rou
```

Recursive occurrences under an even number of arrows.

Type for breadth-first traversal (Hofmann)

Hyperfunctions (Kidney & Wu, POPL 2026)

zip fusion, breadth-first traversal, effect handlers, coroutines, CCS semantics

Stratified types

```
data Typ = TIota | TArr Typ Typ

{-@ stratified Val @-}
data Val where
  {-@ VIota :: Int → Ix Val TIota @-}
  {-@ VArr :: t:Typ → u:Typ → (Ix Val t → Ix Val u)
           → Ix Val (TArr t u) @-}
```

Free recursive occurrences as long as the index is smaller.

The counterpart of large elimination

```
Val :: Typ → Ty
Val TIota      = Int
Val (TArr t u) = Val t → Val u
```

Semantic Models 101

Interpret terms and types as mathematical objects

$$t \mapsto [\![t]\!] \qquad\qquad \tau \mapsto [\![\tau]\!]$$

Soundness

$$\vdash t : \tau \implies [\![t]\!] \in [\![\tau]\!]$$

Consistency

$$[\![\tau]\!] = \emptyset \implies \nexists t.\ \vdash t : \tau$$

PERs over a Scott domain

Terms are interpreted as elements of the following Scott domain, $[\![t]\!]\gamma \in \mathcal{D}$:

Scott domain: an ω-algebraic, bounded-complete complete partial order

$$\mathcal{D} \cong [\mathcal{D} \to \mathcal{D}] \oplus \mathbb{N}_\perp \oplus \cdots$$

$$\mathcal{D}_0 = \{\perp\} \qquad \mathcal{D}_{n+1} = [\mathcal{D}_n \to \mathcal{D}_n] \oplus \mathbb{N}_\perp \oplus \cdots$$

We have two interpretations of types:

- **Refined**: PERs over $\mathcal{D}$, $\quad [\![\tau]\!]\gamma \in \mathcal{PER}(\mathcal{D})$
  - symmetric and transitive
- **Unrefined**: CUPERs over the same domain, $\quad \lfloor\!\lfloor \tau \rfloor\!\rfloor \gamma \in \mathcal{CUPER}(\mathcal{D})$
  - pointed: $(\perp, \perp) \in S$
  - complete: closed under directed lubs
  - uniform: closed under the projections π_n
- **Contexts**: $[\![\Gamma]\!]$ is a PER over environments γ
- **Kinds**: $[\![\kappa]\!]\gamma$ is a set of PER families, $\quad [\![\mathsf{Ty}]\!]\gamma = \mathcal{PER}(\mathcal{D})$

Interpreting judgments

$$
\begin{aligned}
\Gamma \vdash t : \tau \quad &\rightsquigarrow \quad (\llbracket t \rrbracket \gamma, \llbracket t \rrbracket \gamma') \in \llbracket \tau \rrbracket \gamma \quad \text{for } (\gamma, \gamma') \in \llbracket \Gamma \rrbracket \\
\Gamma \vdash \tau : \kappa \quad &\rightsquigarrow \quad \llbracket \tau \rrbracket \gamma \in \llbracket \kappa \rrbracket \gamma \\
\Gamma \vdash \tau \preceq \sigma \quad &\rightsquigarrow \quad \llbracket \tau \rrbracket \gamma \subseteq \llbracket \sigma \rrbracket \gamma \quad \text{and} \quad \lfloor\!\lfloor \tau \rfloor\!\rfloor \gamma = \lfloor\!\lfloor \sigma \rfloor\!\rfloor \gamma \\
\Gamma \models \phi \quad &\rightsquigarrow \quad \llbracket \phi \rrbracket \gamma = tt
\end{aligned}
$$

Interpreting terms

$$
\begin{aligned}
[\![n]\!]\gamma &= i_{\mathbb{N}}(n) \\
[\![x]\!]\gamma &= \gamma(x) \\
[\![\lambda x.\, t]\!]\gamma &= i_{\to}(v \mapsto [\![t]\!](\gamma, v)) \\
[\![t\ u]\!]\gamma &= [\![t]\!]\gamma \cdot [\![u]\!]\gamma \\
[\![\Lambda \alpha.\, t]\!]\gamma &= [\![t]\!]\gamma \\
[\![t\ \tau]\!]\gamma &= [\![t]\!]\gamma
\end{aligned}
$$

Term interpretation is totally independent from types, guaranteeing that we can erase them.

Two interpretations of every type

$$[\![\mathsf{Nat}]\!]\gamma = \{(n, n) \mid n \in \mathbb{N}\} \qquad \lfloor\!\lfloor \mathsf{Nat} \rfloor\!\rfloor \gamma = [\![\mathsf{Nat}]\!]\gamma \cup \{(\bot, \bot)\}$$

$$\lfloor\!\lfloor \{x \colon \tau \mid \phi\} \rfloor\!\rfloor \gamma = \lfloor\!\lfloor \tau \rfloor\!\rfloor \gamma \qquad [\![\{x \colon \tau \mid \phi\}]\!]\gamma = \{(v, w) \in [\![\tau]\!]\gamma \mid [\![\phi]\!](\gamma, v) = tt\}$$

Refined included in unrefined

$[\![\tau]\!]\gamma \subseteq \lfloor\!\lfloor \tau \rfloor\!\rfloor \gamma$: refinements only add logical information.

Function types

Unrefined: related arguments to related results.

$$\lfloor\!\lfloor (x\colon \tau) \to \sigma \rfloor\!\rfloor \gamma = \{(f, g) \mid \forall (v, w) \in \lfloor\!\lfloor \tau \rfloor\!\rfloor \gamma.\ (f\, v, g\, w) \in \lfloor\!\lfloor \sigma \rfloor\!\rfloor \gamma\}$$

Refined: the refined condition *and* the unrefined one.

$$[\![(x\colon \tau) \to \sigma]\!] \gamma = \left\{ (f, g) \;\middle|\; \begin{array}{ll} \forall (v, w) \in [\![\tau]\!] \gamma.\ (f\, v, g\, w) \in [\![\sigma]\!] (\gamma, v) & \text{(i)} \\ \forall (v, w) \in \lfloor\!\lfloor \tau \rfloor\!\rfloor \gamma.\ (f\, v, g\, w) \in \lfloor\!\lfloor \sigma \rfloor\!\rfloor \gamma & \text{(ii)} \end{array} \right\}$$

Propositions

$$[\![t \equiv u]\!]\gamma = tt \iff ([\![t]\!]\gamma, [\![u]\!]\gamma) \in \lfloor\!\lfloor \tau \rfloor\!\rfloor \gamma$$

The unrefined semantics $\lfloor\!\lfloor \tau \rfloor\!\rfloor$ accounts for the unrefined equality

$$[\![\phi \wedge \psi]\!]\gamma = [\![\phi]\!]\gamma \wedge [\![\psi]\!]\gamma$$

Inductive types in CIRT

$$\mu K^{\langle \nu \rangle}\{\ldots\} \qquad \nu \in \{+, m\}$$

- **Positive** $(+)$: occurrences under an even number of arrows
- **Stratified** (m): recursive occurrences at smaller indices, for a metric m

$$\mathsf{Rou} \triangleq \mu \mathsf{R}^{\langle + \rangle}\{\, \mathsf{Over} : \mathsf{R},\ \mathsf{Next} : ((\mathsf{R} \to [\mathsf{Nat}]) \to [\mathsf{Nat}]) \to \mathsf{R} \,\}$$

$$\mathsf{Val} \triangleq \mu \mathsf{V}^{\langle m \rangle}\big\{\, \mathsf{VIota} : \mathsf{Nat} \to \mathsf{V}\,\mathsf{TIota},\ \mathsf{VArr} : (t\colon \mathsf{Typ}) \to (u\colon \mathsf{Typ}) \to (\mathsf{V}\,t \to \mathsf{V}\,u) \to \mathsf{V}\,(\mathsf{TArr}\,t\,u) \,\big\}$$

The eliminator

$$elim_{\mathsf{Val}}(c_{\mathsf{I}}, c_{\mathsf{A}}) : (t\colon \mathsf{Typ}) \to (v\colon \mathsf{Val}\, t) \to \rho(t, v)$$

$$c_{\mathsf{I}} : (n\colon \mathsf{Nat}) \to \mathsf{Unit} \to \rho(\mathsf{TIota}, \mathsf{VIota}\, n)$$

$$c_{\mathsf{A}} : (t\colon \mathsf{Typ}) \to (u\colon \mathsf{Typ}) \to (f\colon \mathsf{Val}\, t \to \mathsf{Val}\, u) \to \mathsf{Unit} \to \mathsf{Unit} \to ((x\colon \mathsf{Val}\, t) \to \rho(u, f\, x)) \to \rho(\mathsf{TArr}\, t\, u, \mathsf{VArr}\, t\, u\, f)$$

Induction hypotheses only for strictly positive occurrences:
the same induction principle as CIC
Captures structurally recursive programs (Abel 1999)

Typing the eliminator

$$\frac{D \triangleq \mu K^{\langle \nu \rangle} \{\, C_i : (\overline{z \colon \sigma}) \to K\, \overline{t}\,\} \qquad \Gamma, \overline{x : \tau}, y : D\, \overline{x} \vdash \rho \qquad \forall i.\ \ \Gamma, \overline{z : \sigma[D/K]}, \overline{\mathsf{El}_{K;\rho}(z, \sigma)[D/K]} \vdash c_i : \rho[\overline{t}/\overline{x}][C_i\, \overline{z}/y]}{\Gamma \vdash \mathit{elim}_D(\overline{c}) : (\overline{x \colon \tau}) \to (y \colon D\, \overline{x}) \to \rho}$$

$$\begin{aligned}
\mathsf{El}_{K;\rho}(z, K\, \overline{t}) &= \rho[\overline{t}/\overline{x}][z/y] \\
\mathsf{El}_{K;\rho}(z, (x \colon \tau) \to \sigma) &= (x \colon \tau) \to \mathsf{El}_{K;\rho}(z\, x, \sigma) \\
\mathsf{El}_{K;\rho}(z, \{x \colon \tau \mid \phi\}) &= \mathsf{El}_{K;\rho}(z, \tau) \\
\mathsf{El}_{K;\rho}(z, \forall \alpha.\, \tau) &= \forall \alpha.\, \mathsf{El}_{K;\rho}(z\, \alpha, \tau) \\
\mathsf{El}_{K;\rho}(z, \tau) &= \mathsf{Unit} \qquad \text{otherwise}
\end{aligned}$$

Denotation of introduction and elimination

$$[\![C_i]\!]\gamma = i_{\to}(\overline{v} \mapsto \langle C_i, \overline{v}\rangle)$$

$$[\![elim_D(\overline{c})]\!]\gamma = i_{\to}\Big(\mathsf{lfp}_r\ \ \overline{k}, v \mapsto \begin{cases} [\![c_i]\!]\gamma \cdot \overline{w} \cdot \overline{\mathsf{rec}_{\tau_j}(w_j, r)} & \text{if } v = \langle C_i, \overline{w}\rangle \\ \bot & \text{otherwise} \end{cases}\Big)$$

$$\begin{aligned} \mathsf{rec}_{K\,\overline{t}}(v, r) &= r\ \overline{[\![t]\!]}\ v \\ \mathsf{rec}_{(x\colon \tau)\to\sigma}(v, r) &= i_{\to}(w \mapsto \mathsf{rec}_{\sigma}(v \cdot w, r)) \\ \mathsf{rec}_{\{x\colon \tau|\phi\}}(v, r) &= \mathsf{rec}_{\forall\alpha.\,\tau}(v, r) = \mathsf{rec}_{\tau}(v, r) \\ \mathsf{rec}_{\tau}(v, r) &= \star \qquad \text{otherwise} \end{aligned}$$

Unrefined semantics of inductive types

- Positive and stratified types are interpreted the same way
- We have to account for diverging programs: $\bot$
- Standard approach for mixed-variance type operators (Amadio 1991): ultrametric space over PERs, show the operator is contractive (Banach)
- We adapt the minimal invariance argument (Pitts 1996), no metric:

$$\bigsqcup_n \pi_n = id \qquad A_n = \{(\pi_n a, \pi_n b) \mid (a, b) \in A\}$$

$$\mathcal{F}_\bot \text{ contractive:} \quad A_n = B_n \implies \mathcal{F}_\bot(A)_{n+1} = \mathcal{F}_\bot(B)_{n+1}$$

$$\lfloor\!\lfloor \mu K^{\langle \nu \rangle}\{\ldots\} \rfloor\!\rfloor = \mathsf{fix}_X\, \mathcal{F}_\bot(X)$$

Constructors are contractive, so the fixed point exists and it is unique

Refined semantics of inductive types

$$[\![\mu K^{\langle\nu\rangle}\{\ldots\}]\!] = \mathsf{lfp}_X\, \mathcal{F}(X) \qquad X \subseteq \lfloor\!|\mu K^{\langle\nu\rangle}\{\ldots\}|\!\rfloor$$

least fixed point among the candidates included in the unrefined interpretation

- **Positive**: occurrences under an even number of arrows make $\mathcal{F}$ monotone with respect to $\subseteq$, so Knaster–Tarski gives the least fixed point
- **Stratified**: the interpretation at an index only uses smaller indices, so it is defined by well-founded recursion on m; the fixed point is unique, hence least

Soundness of the eliminator

Goal, with $e \triangleq [\![elim_D(\overline{c})]\!]$:

$$(v, v') \in [\![D]\!] \implies (e\, v, e\, v') \in [\![\rho]\!]$$

- **Unrefined**: fixed-point induction on e works, because CUPERs are pointed and complete.
- **Refined**: PERs are not complete, so we do induction on the data instead.

$$P \triangleq \{(v, v') \in [\![D]\!] \mid (e\, v, e\, v') \in [\![\rho]\!]\}$$

- **Positive**: we need both
  - $\mathcal{F}$ is monotone: $\mathcal{F}(P) \subseteq \mathcal{F}([\![D]\!]) = [\![D]\!]$
  - P is closed under the constructors: the eliminator respects the motive on $\mathcal{F}(P)$

$$\mathcal{F}(P) \subseteq P \implies [\![D]\!] = \mathsf{lfp}\, \mathcal{F} \subseteq P$$

- **Stratified**: similar argument, but by induction on the index.

TL;DR: consistency

$$\Gamma \vdash t : \tau \implies (\llbracket t \rrbracket \gamma, \llbracket t \rrbracket \gamma') \in \llbracket \Gamma \vdash \tau : \mathsf{Ty} \rrbracket \gamma \quad \text{for } (\gamma, \gamma') \in \llbracket \Gamma \rrbracket$$

$$\llbracket \Gamma \vdash \forall \alpha \colon \mathsf{Ty}.\, \alpha : \mathsf{Ty} \rrbracket \gamma = \bigcap_{P \subseteq P_\perp} P = \emptyset$$

for $P \in \mathcal{PER}(\mathcal{D})$ and $P_\perp \in \mathcal{CUPER}(\mathcal{D})$

$$\nexists t.\ \cdot \vdash t : \forall \alpha \colon \mathsf{Ty}.\, \alpha$$

Summary

- First semantic model of refinement types: subtyping, unrefined equality, impredicative polymorphism and non-standard inductive types
- First proof of consistency of a realistic calculus for refinement types
- Two layers: refined PERs included in unrefined CUPERs over a Scott domain

Questions?