Re^2: A Type System for Refinements and Resources

Tristan Knoth [1] Di Wang [2] Nadia Polikarpova [1] Jan Hoffmann [2]

[1]University of California, San Diego

[2]Carnegie Mellon University

Refinements: Functional Specification

Dependent Types

- Martin-Löf's Type Theory (underlying NuPRL)
- Calculus of Inductive Constructions (underlying Coq)

Some Restricted Forms of Dependent Types

	Features
[FP91]	Regular-tree based refinements for datatypes.
[HPS96]	Sized types. Only support "primitive" recursion.
[XP99]	Dependent ML. Indexed types with refinement sorts.
[CW00]	Indexed types with inductive kinds and type-level computation.
[VH04]	Sized types. Support general recursion.
[RKJ08]	Liquid types. Predicate-abstraction refinements for base types.
[WWC17]	TiML. Indexed types with refinement kinds. Proved in Coq.

Refinements: Functional Specification

Dependent Types

- Martin-Löf's Type Theory (underlying NuPRL)
- Calculus of Inductive Constructions (underlying Coq)

Some Restricted Forms of Dependent Types

	Features
[FP91]	Regular-tree based refinements for datatypes.
[HPS96]	Sized types. Only support "primitive" recursion.
[XP99]	Dependent ML. Indexed types with refinement sorts.
[CW00]	Indexed types with inductive kinds and type-level computation.
[VH04]	Sized types. Support general recursion.
[RKJ08]	Liquid types. Predicate-abstraction refinements for base types.
[WWC17]	TiML. Indexed types with refinement kinds. Proved in Coq.

Resources: Complexity Specification

Automatic Amortized Resource Analysis (AARA)

- Introduced by Hofmann and Jost in 2003 [HJ03].
- Extended to OCaml by Hoffmann et al. in 2017 [HDW17].

Some Restricted Forms of Dependent Types

	Features
[FP91]	Regular-tree based refinements for datatypes.
[HPS96]	Sized types. Only support "primitive" recursion.
[XP99]	Dependent ML. Indexed types with refinement sorts.
[CW00]	Indexed types with inductive kinds and type-level computation.
[VH04]	Sized types. Support general recursion.
[RKJ08]	Liquid types. Predicate-abstraction refinements for base types.
[WWC17]	TiML. Indexed types with refinement kinds. Proved in Coq.

Resources: Complexity Specification

Automatic Amortized Resource Analysis (AARA)

- Introduced by Hofmann and Jost in 2003 [HJ03].
- Extended to OCaml by Hoffmann et al. in 2017 [HDW17].

Some Restricted Forms of Dependent Types

	Features
[FP91]	Regular-tree based refinements for datatypes.
[HPS96]	Sized types. Only support "primitive" recursion.
[XP99]	Dependent ML. Indexed types with refinement sorts.
[CW00]	Indexed types with inductive kinds and type-level computation.
[VH04]	Sized types. Support general recursion.
[RKJ08]	Liquid types. Predicate-abstraction refinements for base types.
[WWC17]	TiML. Indexed types with refinement kinds. Proved in Coq.

Re^2: Liquid Types + AARA

Features

- Polymorphic refinement types over logical qualifiers.
- Affine types with linear potential annotations.
- Potentials are expressed in the same refinement language.

Limitations

- Limited by the capability of liquid types and AARA.
- Liquid types: Rely on decidable refinement logic.
- AARA: Currently limited to polynomial (and exponential) complexity.

Re^2: Liquid Types + AARA

Features

- Polymorphic refinement types over logical qualifiers.
- Affine types with linear potential annotations.
- Potentials are expressed in the same refinement language.

Limitations

- Limited by the capability of liquid types and AARA.
- Liquid types: Rely on decidable refinement logic.
- AARA: Currently limited to polynomial (and exponential) complexity.

A Running Example: List Append

$$\begin{aligned}
append &:: \forall\alpha.L(\alpha) \to L(\alpha) \to L(\alpha)\\
append\ \ell_1\ \ell_2 &= \mathtt{\textbf{match}}\ \ell_1\ \mathtt{\textbf{with}}\\
&\quad \mid [] \to \ell_2\\
&\quad \mid x :: xs \to \mathtt{\textbf{let}}\ ys = append\ xs\ \ell_2\ \mathtt{\textbf{in}}\ (x :: ys)
\end{aligned}$$

- Functionality: size of $append(\ell_1)(\ell_2)$ is the sum of sizes of ℓ_1 and ℓ_2
- Complexity: $append(\ell_1)(\ell_2)$ makes $2 \cdot |\ell_1|$ function calls

Review of Liquid Types

$B ::=$	bool	base type of Booleans
	$L(T)$	base type of lists
	α	type variable
$T ::=$	$\{\nu : B \mid \psi\}$	refinement type
	$x : T_x \rightarrow T$	dependent arrow type
$S ::=$	T	monomorphic type
	$\forall \alpha . S$	polymorphic type
$\psi ::=$	$\star \leq \nu \mid \nu < \star \mid \nu < \mathrm{size}(\star) \mid \cdots$	logical qualifier
	$\psi_1 \wedge \psi_2$	conjunction

Review of Liquid Types

$$append :: \forall \alpha . \ell_1 : L(\alpha) \to \ell_2 : L(\alpha) \to \{v : L(\alpha) \mid \mathsf{size}(v) = \mathsf{size}(\ell_1) + \mathsf{size}(\ell_2)\}$$

$append\ \ell_1\ \ell_2 = \mathbf{match}\ \ell_1\ \mathbf{with}$

$\mid [\,] \to$

$\{\ell_2 : L(\alpha); \mathsf{size}(\ell_1) = 0\}$

ℓ_2

$\{v : L(\alpha) \mid \mathsf{size}(v) = \mathsf{size}(\ell_2)\} <: \{v : L(\alpha) \mid \mathsf{size}(v) = \mathsf{size}(\ell_1) + \mathsf{size}(\ell_2)\}$

$\mid x :: xs \to$

$\{\ell_2 : L(\alpha), x : \alpha, xs : L(\alpha); \mathsf{size}(\ell_1) = \mathsf{size}(xs) + 1\}$

$\mathbf{let}\ ys = append\ xs\ \ell_2\ \mathbf{in}$

$\{x : \alpha, ys : \{v : L(\alpha) \mid \mathsf{size}(v) = \mathsf{size}(xs) + \mathsf{size}(\ell_2)\}; \mathsf{size}(\ell_1) = \mathsf{size}(xs) + 1\}$

$(x :: ys)$

$\{v : L(\alpha) \mid \mathsf{size}(v) = \mathsf{size}(ys) + 1\}$

$<: \{v : L(\alpha) \mid \mathsf{size}(v) = \mathsf{size}(xs) + \mathsf{size}(\ell_2) + 1\}$

$<: \{v : L(\alpha) \mid \mathsf{size}(v) = \mathsf{size}(\ell_1) + \mathsf{size}(\ell_2)\}$

Review of AARA

$B ::= \text{bool}$	base type of Booleans
$\quad L(R)$	base type of lists
$T ::= B$	base type
$\quad R_1 \to R_2$	arrow type
$R ::= T^q$	resource-annotated type

Review of AARA

$$append :: L(\text{bool}^2) \to L(\text{bool}^0) \to L(\text{bool}^0)$$

$$append\ \ell_1\ \ell_2 = \texttt{match}\ \ell_1\ \texttt{with}$$

$$\mid [\,] \to$$

$$\{\ell_2 : L(\text{bool}^0); 0\}$$

$$\ell_2$$

$$L(\text{bool}^0)$$

$$\mid x :: xs \to$$

$$\{\ell_2 : L(\text{bool}^0), x : \text{bool}, xs : L(\text{bool}^2); 2\}$$

$$\texttt{let}\ ys = append\ xs\ \ell_2\ \texttt{in}$$

$$\{x : \text{bool}, ys : L(\text{bool}^0); 0\}$$

$$(x :: ys)$$

$$L(\text{bool}^0)$$

Liquid Types + AARA

Liquid Types		AARA	
B ::=	bool	B ::=	bool
	$L(T)$		$L(R)$
	α		
T ::=	$\{v : B \mid \psi\}$	T ::=	B
	$x : T_x \longrightarrow T$		$R_1 \longrightarrow R_2$
		R ::=	T^q
S ::=	T		
	$\forall \alpha . S$		
ψ ::=	$\cdots$		
	$\psi_1 \wedge \psi_2$		

Re^2: Liquid Types + AARA

$B ::=$	bool	base type of Booleans
	$L(R)$	base type of lists
	α	type variable
$T ::=$	$\{v : B \mid \psi\}$	refinement type
	$x : R_x \to R$	dependent arrow type
$R ::=$	T^{ϕ}	resource-annotated type
$S ::=$	R	monomorphic type
	$\forall \alpha . S$	polymorphic type
$\psi ::=$	$\star \le v \mid v < \star \mid v < \mathsf{size}(\star) \mid \cdots$	logical qualifier
	$\psi_1 \wedge \psi_2$	conjunction
$\phi ::=$	$v \mid \star \mid \mathsf{size}(\star) \mid \cdots$	numeric qualifier
	$\phi_1 + \phi_2$	addition

Re^2: Liquid Types + AARA

$$append :: \forall\alpha.\ell_1 : L(\alpha^2) \to \ell_2 : L(\alpha^0) \to \{v : L(\alpha^0) \mid \text{size}(v) = \text{size}(\ell_1) + \text{size}(\ell_2)\}$$

$append\ \ell_1\ \ell_2 =$ **match** ℓ_1 **with**

$\mid [] \to$

$\{\ell_2 : L(\alpha^0); \text{size}(\ell_1) = 0; 0\}$

ℓ_2

$\{v : L(\alpha^0) \mid \text{size}(v) = \text{size}(\ell_2)\} <: \{v : L(\alpha^0) \mid \text{size}(v) = \text{size}(\ell_1) + \text{size}(\ell_2)\}$

$\mid x :: xs \to$

$\{\ell_2 : L(\alpha^0), x : \alpha, xs : L(\alpha^2); \text{size}(\ell_1) = \text{size}(xs) + 1; 2\}$

let $ys = append\ xs\ \ell_2$ **in**

$\{x : \alpha, ys : \{v : L(\alpha^0) \mid \text{size}(v) = \text{size}(xs) + \text{size}(\ell_2)\}; \text{size}(\ell_1) = \text{size}(xs) + 1; 0\}$

$(x :: ys)$

$\{v : L(\alpha^0) \mid \text{size}(v) = \text{size}(ys) + 1\}$

$<: \{v : L(\alpha^0) \mid \text{size}(v) = \text{size}(xs) + \text{size}(\ell_2) + 1\}$

$<: \{v : L(\alpha^0) \mid \text{size}(v) = \text{size}(\ell_1) + \text{size}(\ell_2)\}$

Re^2: Liquid Types + AARA

$$append :: \forall\alpha.\ell_1 : L(\alpha^2) \to \ell_2 : L(\alpha^0) \to \{v : L(\alpha^0) \mid \mathsf{size}(v) = \mathsf{size}(\ell_1) + \mathsf{size}(\ell_2)\}$$

$$append :: \forall\alpha.\ell_1 : L(\alpha)^{2\cdot\mathsf{size}(v)} \to \ell_2 : L(\alpha) \to \{v : L(\alpha) \mid \mathsf{size}(v) = \mathsf{size}(\ell_1) + \mathsf{size}(\ell_2)\}$$

$$append :: \forall\alpha.\ell_1 : L(\alpha) \to \ell_2 : L(\alpha)^{2\cdot\mathsf{size}(\ell_1)} \to \{v : L(\alpha) \mid \mathsf{size}(v) = \mathsf{size}(\ell_1) + \mathsf{size}(\ell_2)\}$$

Dynamic Semantics: Resource-Aware, Small-Step

$$\langle e, p \rangle \mapsto \langle e', p' \rangle$$

(E:Tick)

$$\frac{p \geq 0 \qquad p - c \geq 0}{\langle \texttt{tick}\ c\ \texttt{in}\ e, p \rangle \mapsto \langle e, p - c \rangle}$$

Dynamic Semantics: Resource-Aware, Small-Step

$$\langle e, p \rangle \mapsto \langle e', p' \rangle$$

(E:Tick)

$$\frac{p \geq 0 \qquad p - c \geq 0}{\langle \texttt{tick}\ c\ \texttt{in}\ e, p \rangle \mapsto \langle e, p - c \rangle}$$

Static Semantics

Language Design

Expressions in Re[2] are in A-Normal-Form, i.e., syntactic forms in non-tail positions allow only variables and values.

$$\Gamma; \Psi; \Phi \vdash e : S$$

(T:True)

$$\frac{}{\Gamma; \Psi; \Phi \vdash \texttt{true} : \{v : \text{bool} \mid v = \top\}}$$

(T:Nil)

$$\frac{\Gamma \vdash R \text{ type}}{\Gamma; \Psi; \Phi \vdash \texttt{nil} : \{v : L(R) \mid \text{size}(v) = 0\}}$$

Static Semantics

Language Design

Expressions in Re^2 are in A-Normal-Form, i.e., syntactic forms in non-tail positions allow only variables and values.

$$\Gamma; \Psi; \Phi \vdash e : S$$

(T:True)

$$\frac{}{\Gamma; \Psi; \Phi \vdash \texttt{true} : \{v : \text{bool} \mid v = \top\}}$$

(T:Nil)

$$\frac{\Gamma \vdash R \text{ type}}{\Gamma; \Psi; \Phi \vdash \texttt{nil} : \{v : L(R) \mid \text{size}(v) = 0\}}$$

Static Semantics

Language Design

Expressions in Re[2] are in A-Normal-Form, i.e., syntactic forms in non-tail positions allow only variables and values.

$$\Gamma; \Psi; \Phi \vdash e : S$$

(T:True)

$$\frac{}{\Gamma; \Psi; \Phi \vdash \mathbf{true} : \{v : \text{bool} \mid v = \top\}}$$

(T:Nil)

$$\frac{\Gamma \vdash R \text{ type}}{\Gamma; \Psi; \Phi \vdash \mathbf{nil} : \{v : L(R) \mid \text{size}(v) = 0\}}$$

Static Semantics

(T:Cond)

$$\frac{\Gamma(x) = \text{bool} \qquad \Gamma; \Psi \wedge x; \Phi \vdash e_1 : R \qquad \Gamma; \Psi \wedge \neg x; \Phi \vdash e_2 : R}{\Gamma; \Psi; \Phi \vdash \textbf{if } x \textbf{ then } e_1 \textbf{ else } e_2 : R}$$

(T:AppFO)

$$\frac{\Gamma(x_1) = x : \{\nu : B \mid \psi\}^{\phi} \to R \qquad \Gamma(x_2) = \{\nu : B \mid \psi\}}{\Gamma; \top; [x_2/\nu]\phi \vdash x_1(x_2) : R}$$

(T:MatL)

$$\frac{\begin{array}{c}\Gamma(x) = L(T^{\phi}) \qquad \Gamma; \Psi \wedge \text{size}(x) = 0; \Phi \vdash e_1 : R' \\ \Gamma, x_1 : T, x_2 : L(T^{\phi}); \Psi \wedge \text{size}(x) = \text{size}(x_2) + 1; \Phi + [x_1/\nu]\phi \vdash e_2 : R'\end{array}}{\Gamma; \Psi; \Phi \vdash \textbf{match } x \textbf{ with } \{[] \hookrightarrow e_1 \mid x_1 :: x_2 \hookrightarrow e_2\} : R'}$$

Static Semantics

(T:Cond)

$$\frac{\Gamma(x) = \mathsf{bool} \qquad \Gamma; \Psi \wedge x; \Phi \vdash e_1 : R \qquad \Gamma; \Psi \wedge \neg x; \Phi \vdash e_2 : R}{\Gamma; \Psi; \Phi \vdash \mathbf{if}\ x\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2 : R}$$

(T:AppFO)

$$\frac{\Gamma(x_1) = x : \{v : B \mid \psi\}^{\phi} \longrightarrow R \qquad \Gamma(x_2) = \{v : B \mid \psi\}}{\Gamma; \top; [x_2/v]\phi \vdash x_1(x_2) : R}$$

(T:MatL)

$$\frac{\Gamma(x) = L(T^{\phi}) \qquad \Gamma; \Psi \wedge \mathsf{size}(x) = 0; \Phi \vdash e_1 : R' \qquad \Gamma, x_1 : T, x_2 : L(T^{\phi}); \Psi \wedge \mathsf{size}(x) = \mathsf{size}(x_2) + 1; \Phi + [x_1/v]\phi \vdash e_2 : R'}{\Gamma; \Psi; \Phi \vdash \mathbf{match}\ x\ \mathbf{with}\ \{[] \hookrightarrow e_1 \mid x_1 :: x_2 \hookrightarrow e_2\} : R'}$$

Static Semantics

(T:Cond)

$$\frac{\Gamma(x) = \text{bool} \qquad \Gamma; \Psi \wedge x; \Phi \vdash e_1 : R \qquad \Gamma; \Psi \wedge \neg x; \Phi \vdash e_2 : R}{\Gamma; \Psi; \Phi \vdash \textbf{if } x \textbf{ then } e_1 \textbf{ else } e_2 : R}$$

(T:AppFO)

$$\frac{\Gamma(x_1) = x : \{v : B \mid \psi\}^{\phi} \longrightarrow R \qquad \Gamma(x_2) = \{v : B \mid \psi\}}{\Gamma; \top; [x_2/v]\phi \vdash x_1(x_2) : R}$$

(T:MatL)

$$\frac{\begin{array}{c}\Gamma(x) = L(T^{\phi}) \qquad \Gamma; \Psi \wedge \text{size}(x) = 0; \Phi \vdash e_1 : R' \\ \Gamma, x_1 : T, x_2 : L(T^{\phi}); \Psi \wedge \text{size}(x) = \text{size}(x_2) + 1; \Phi + [x_1/v]\phi \vdash e_2 : R'\end{array}}{\Gamma; \Psi; \Phi \vdash \textbf{match } x \textbf{ with } \{[] \hookrightarrow e_1 \mid x_1 :: x_2 \hookrightarrow e_2\} : R'}$$

Meta Theory

Progress

If $\cdot;\cdot;q \vdash e : S$ and $p \geq q$, then either e is a value or there exist e' and p' such that $\langle e, p\rangle \mapsto \langle e', p'\rangle$.

Preservation

If $\cdot;\cdot;q \vdash e : S$, $p \geq q$, and $\langle e, p\rangle \mapsto \langle e', p'\rangle$, then $\cdot;\cdot;p' \vdash e' : S$.

Consistency

If $\cdot;\cdot;q \vdash e : S$ and e is a value, then e satisfies the conditions indicated by S and q is greater than or equal to the potential stored in v with respect to S.

Meta Theory

Progress

If $\cdot;\cdot;q \vdash e : S$ and $p \geq q$, then either e is a value or there exist e' and p' such that $\langle e, p\rangle \mapsto \langle e', p'\rangle$.

Preservation

If $\cdot;\cdot;q \vdash e : S$, $p \geq q$, and $\langle e, p\rangle \mapsto \langle e', p'\rangle$, then $\cdot;\cdot;p' \vdash e' : S$.

Consistency

If $\cdot;\cdot;q \vdash e : S$ and e is a value, then e satisfies the conditions indicated by S and q is greater than or equal to the potential stored in v with respect to S.

Meta Theory

Progress

If $\cdot;\cdot;q \vdash e : S$ and $p \geq q$, then either e is a value or there exist e' and p' such that $\langle e, p\rangle \mapsto \langle e', p'\rangle$.

Preservation

If $\cdot;\cdot;q \vdash e : S$, $p \geq q$, and $\langle e, p\rangle \mapsto \langle e', p'\rangle$, then $\cdot;\cdot;p' \vdash e' : S$.

Consistency

If $\cdot;\cdot;q \vdash e : S$ and e is a value, then e satisfies the conditions indicated by S and q is greater than or equal to the potential stored in v with respect to S.

Interpretation into Refinement Logic

Ideas

- Reflect interpretable values in the refinement logic.
- Booleans are interpreted as $\{\top, \bot\}$. Lists are interpreted as sizes.
- Develop a denotational semantics for the refinement and resource annotations.

Interpretation into Refinement Logic

$$\mathcal{I}(\texttt{true}) = \top \qquad \mathcal{I}(\texttt{nil}) = 0$$

$$\mathcal{I}(\texttt{false}) = \bot \qquad \mathcal{I}(\texttt{cons}(v_1, v_2)) = \mathcal{I}(v_2) + 1$$

- $\vdash b : \{v : \text{bool} \mid \psi\}$ indicates that $\models [\mathcal{I}(b)/v]\psi$.
- $\vdash [b_1, \cdots, b_n] : \{v : L(\{v : \text{bool} \mid \psi'\}) \mid \psi\}$ indicates that $\models [n/\text{size}(v)]\psi \wedge \bigwedge_{i=1}^{n} [\mathcal{I}(b_i)/v]\psi'$.

Interpretation into Refinement Logic

$$\mathcal{I}(\texttt{true}) = \top \qquad \mathcal{I}(\texttt{nil}) = 0$$

$$\mathcal{I}(\texttt{false}) = \bot \qquad \mathcal{I}(\texttt{cons}(v_1, v_2)) = \mathcal{I}(v_2) + 1$$

- $\vdash b : \{v : \text{bool} \mid \psi\}$ indicates that $\models [\mathcal{I}(b)/v]\psi$.
- $\vdash [b_1, \cdots, b_n] : \{v : L(\{v : \text{bool} \mid \psi'\}) \mid \psi\}$ indicates that $\models [n/\text{size}(v)]\psi \wedge \bigwedge_{i=1}^{n} [\mathcal{I}(b_i)/v]\psi'$.

Interpretation into Refinement Logic

Consistency: Intuition

If $\cdot;\cdot;q \vdash e : S$ and e is a value, then v satisfies the conditions indicated by S and q is greater than or equal to the potential stored in v with respect to S.

Consistency: Formalization

If $\cdot;\cdot;q \vdash e : S$ and e is a value, the logical refinement of S is ψ, and the resource annotation of S is ϕ, then $\models [\mathcal{I}(e)/v]\psi$ and also $\models q \geq [\mathcal{I}(e)/v]\phi$.

Interpretation into Refinement Logic

Consistency: Intuition

If $\cdot;\cdot;q \vdash e : S$ and e is a value, then v satisfies the conditions indicated by S and q is greater than or equal to the potential stored in v with respect to S.

Consistency: Formalization

If $\cdot;\cdot;q \vdash e : S$ and e is a value, the logical refinement of S is ψ, and the resource annotation of S is ϕ, then $\models [\mathcal{I}(e)/v]\psi$ and also $\models q \geq [\mathcal{I}(e)/v]\phi$.