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

<table>
<thead>
<tr>
<th>Reference</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>[FP91]</td>
<td>Regular-tree based refinements for datatypes.</td>
</tr>
<tr>
<td>[HPS96]</td>
<td>Sized types. Only support “primitive” recursion.</td>
</tr>
<tr>
<td>[XP99]</td>
<td>Dependent ML. Indexed types with refinement sorts.</td>
</tr>
<tr>
<td>[CW00]</td>
<td>Indexed types with inductive kinds and type-level computation.</td>
</tr>
<tr>
<td>[RKJ08]</td>
<td>Liquid types. Predicate-abstraction refinements for base types.</td>
</tr>
</tbody>
</table>
Refinements: Functional Specification

Dependent Types

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

Some Restricted Forms of Dependent Types

<table>
<thead>
<tr>
<th>Features</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular-tree based refinements for datatypes.</td>
<td>[FP91]</td>
</tr>
<tr>
<td>Sized types. Only support “primitive” recursion.</td>
<td>[HPS96]</td>
</tr>
<tr>
<td>Dependent ML. Indexed types with refinement sorts.</td>
<td>[XP99]</td>
</tr>
<tr>
<td>Indexed types with inductive kinds and type-level computation.</td>
<td>[CW00]</td>
</tr>
<tr>
<td>Sized types. Support general recursion.</td>
<td>[VH04]</td>
</tr>
<tr>
<td>Liquid types. Predicate-abstraction refinements for base types.</td>
<td>[RKJ08]</td>
</tr>
<tr>
<td>TiML. Indexed types with refinement kinds. Proved in Coq.</td>
<td>[WWC17]</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Reference</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>[FP91]</td>
<td>Regular-tree based refinements for datatypes.</td>
</tr>
<tr>
<td>[HPS96]</td>
<td>Sized types. Only support “primitive” recursion.</td>
</tr>
<tr>
<td>[XP99]</td>
<td>Dependent ML. Indexed types with refinement sorts.</td>
</tr>
<tr>
<td>[CW00]</td>
<td>Indexed types with inductive kinds and type-level computation.</td>
</tr>
<tr>
<td>[RKJ08]</td>
<td>Liquid types. Predicate-abstraction refinements for base types.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Reference</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>[FP91]</td>
<td>Regular-tree based refinements for datatypes.</td>
</tr>
<tr>
<td>[HPS96]</td>
<td>Sized types. Only support “primitive” recursion.</td>
</tr>
<tr>
<td>[XP99]</td>
<td>Dependent ML. Indexed types with refinement sorts.</td>
</tr>
<tr>
<td>[CW00]</td>
<td>Indexed types with inductive kinds and type-level computation.</td>
</tr>
<tr>
<td>[RKJ08]</td>
<td>Liquid types. Predicate-abstraction refinements for base types.</td>
</tr>
</tbody>
</table>
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.
Re²: 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

\[
\text{append} \equiv \forall \alpha. L(\alpha) \to L(\alpha) \to L(\alpha)
\]

\[
\text{append } \ell_1 \ell_2 = \text{match } \ell_1 \text{ with }
\]
\[
| [] \rightarrow \ell_2 \\
| x :: xs \rightarrow \text{let } ys = \text{append } xs \ell_2 \text{ in } (x :: ys)
\]

- Functionality: size of \(\text{append}(\ell_1)(\ell_2)\) is the sum of sizes of \(\ell_1\) and \(\ell_2\)
- Complexity: \(\text{append}(\ell_1)(\ell_2)\) makes \(2 \cdot |\ell_1|\) function calls
Review of Liquid Types

\[
B ::= \text{bool} \quad \text{base type of Booleans} \\
L(T) \quad \text{base type of lists} \\
\alpha \quad \text{type variable} \\
T ::= \{v : B \mid \psi\} \quad \text{refinement type} \\
x : T_x \to T \quad \text{dependent arrow type} \\
S ::= T \quad \text{monomorphic type} \\
\forall \alpha. S \quad \text{polymorphic type} \\
\psi ::= \star \leq v \mid v < \star \mid v < \text{size}(\star) \mid \cdots \\
\psi_1 \land \psi_2 \quad \text{conjunction}
\]
Review of Liquid Types

\[
append :: \forall \alpha. \ell_1 : L(\alpha) \to \ell_2 : L(\alpha) \to \{ v : L(\alpha) \mid \text{size}(v) = \text{size}(\ell_1) + \text{size}(\ell_2) \}
\]

\[
append \ \ell_1 \ \ell_2 = \text{match} \ \ell_1 \ \text{with}
\]

\[
\mid [] \to \{ \ell_2 : L(\alpha); \text{size}(\ell_1) = 0 \}
\]

\[
\ell_2
\]

\[
\{ v : L(\alpha) \mid \text{size}(v) = \text{size}(\ell_2) \} <: \{ v : L(\alpha) \mid \text{size}(v) = \text{size}(\ell_1) + \text{size}(\ell_2) \}
\]

\[
\mid x :: xs \to \{ \ell_2 : L(\alpha), x : \alpha, xs : L(\alpha); \text{size}(\ell_1) = \text{size}(xs) + 1 \}
\]

\[
\text{let} \ ys = append \ xs \ \ell_2 \ \text{in}
\]

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

\[
(x :: ys)
\]

\[
\{ v : L(\alpha) \mid \text{size}(v) = \text{size}(ys) + 1 \}
\]

\[
<: \{ v : L(\alpha) \mid \text{size}(v) = \text{size}(xs) + \text{size}(\ell_2) + 1 \}
\]

\[
<: \{ v : L(\alpha) \mid \text{size}(v) = \text{size}(\ell_1) + \text{size}(\ell_2) \}
\]
Review of AARA

\[
\begin{align*}
B & ::= \text{bool} & \text{base type of Booleans} \\
L(R) & \text{base type of lists} \\
T & ::= B & \text{base type} \\
R_1 & \rightarrow R_2 & \text{arrow type} \\
R & ::= T^q & \text{resource-annotated type}
\end{align*}
\]
Review of AARA

\[append \colon L(\text{bool}^2) \to L(\text{bool}^0) \to L(\text{bool}^0)\]

\[append \ell_1 \ell_2 = \text{match } \ell_1 \text{ with}\]

\[\begin{array}{ll}
| \; [] & \to \\
& \{\ell_2 : L(\text{bool}^0); 0\}
\end{array}\]

\[\ell_2\]

\[L(\text{bool}^0)\]

\[| \; x :: xs & \to \\
& \{\ell_2 : L(\text{bool}^0), x : \text{bool}, xs : L(\text{bool}^2); 2\}
\]

\[\text{let } ys = append xs \ell_2 \text{ in}\]

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

\[(x :: ys)\]

\[L(\text{bool}^0)\]
## Liquid Types + AARA

<table>
<thead>
<tr>
<th>Liquid Types</th>
<th>AARA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B ::= \text{bool}$</td>
<td>$B ::= \text{bool}$</td>
</tr>
<tr>
<td>$L(T)$</td>
<td>$L(R)$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td></td>
</tr>
<tr>
<td>$T ::= {v : B \mid \psi}$</td>
<td>$T ::= B$</td>
</tr>
<tr>
<td>$x : T_x \rightarrow T$</td>
<td>$R_1 \rightarrow R_2$</td>
</tr>
<tr>
<td>$S ::= T$</td>
<td>$R ::= T^q$</td>
</tr>
<tr>
<td>$\forall \alpha. S$</td>
<td></td>
</tr>
<tr>
<td>$\psi ::= \cdots$</td>
<td></td>
</tr>
<tr>
<td>$\psi_1 \land \psi_2$</td>
<td></td>
</tr>
</tbody>
</table>
\( \mathbb{R}^2: \) Liquid Types + AARA

\[
\begin{align*}
B & ::= \text{bool} & & \text{base type of Booleans} \\
L(R) & & & \text{base type of lists} \\
\alpha & & & \text{type variable} \\
T & ::= \{v : B \mid \psi\} & & \text{refinement type} \\
x : \ R_x \to R & & & \text{dependent arrow type} \\
R & ::= T^\phi & & \text{resource-annotated type} \\
S & ::= R & & \text{monomorphic type} \\
\forall \alpha. S & & & \text{polymorphic type} \\
\psi & ::= \star \leq v \mid v < \star \mid v < \text{size}(\star) \mid \cdots \\
\psi_1 \land \psi_2 & & & \text{conjunction} \\
\phi & ::= v \mid \star \mid \text{size}(\star) \mid \cdots \\
\phi_1 + \phi_2 & & & \text{addition}
\end{align*}
\]
Re²: Liquid Types + AARA

\[\text{append} :: \forall \alpha. \ell_1 : L(\alpha^2) \rightarrow \ell_2 : L(\alpha^0) \rightarrow \{\nu : L(\alpha^0) \mid \text{size}(\nu) = \text{size}(\ell_1) + \text{size}(\ell_2)\}\]

\[\text{append } \ell_1 \ell_2 = \text{match } \ell_1 \text{ with}
\]

\[
\mid [] \rightarrow
\quad \{\ell_2 : L(\alpha^0); \text{size}(\ell_1) = 0; 0\}
\]

\[
\ell_2
\quad \{\nu : L(\alpha^0) \mid \text{size}(\nu) = \text{size}(\ell_2)\} \triangleleft \{\nu : L(\alpha^0) \mid \text{size}(\nu) = \text{size}(\ell_1) + \text{size}(\ell_2)\}
\]

\[
\mid x :: xs \rightarrow
\quad \{\ell_2 : L(\alpha^0), x : \alpha, xs : L(\alpha^2); \text{size}(\ell_1) = \text{size}(xs) + 1; 2\}
\]

\[\text{let } ys = \text{append } xs \ell_2 \text{ in}
\]

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

\[(x :: ys)
\quad \{\nu : L(\alpha^0) \mid \text{size}(\nu) = \text{size}(ys) + 1\}
\]

\[\triangleleft \{\nu : L(\alpha^0) \mid \text{size}(\nu) = \text{size}(xs) + \text{size}(\ell_2) + 1\}
\]

\[\triangleleft \{\nu : L(\alpha^0) \mid \text{size}(\nu) = \text{size}(\ell_1) + \text{size}(\ell_2)\}\]
**Re²: Liquid Types + AARA**

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

\[
\text{append} :: \forall \alpha. \ell_1 : L(\alpha)^2 \cdot \text{size}(\nu) \rightarrow \ell_2 : L(\alpha) \rightarrow \{ v : L(\alpha) \mid \text{size}(v) = \text{size}(\ell_1) + \text{size}(\ell_2) \}
\]

\[
\text{append} :: \forall \alpha. \ell_1 : L(\alpha) \rightarrow \ell_2 : L(\alpha)^2 \cdot \text{size}(\ell_1) \rightarrow \{ v : L(\alpha) \mid \text{size}(v) = \text{size}(\ell_1) + \text{size}(\ell_2) \}
\]
Dynamic Semantics: Resource-Aware, Small-Step

\[\langle e, p \rangle \rightarrow \langle e', p' \rangle\]

(E:Tick)

\[
\begin{align*}
p & \geq 0 \\
p - c & \geq 0
\end{align*}
\]

\[\langle \text{tick } c \text{ in } e, p \rangle \rightarrow \langle e, p - c \rangle\]
Dynamic Semantics: Resource-Aware, Small-Step

\[ \langle e, p \rangle \mapsto \langle e', p' \rangle \]

(E:Tick)

\[
\begin{align*}
p & \geq 0 \\
p - c & \geq 0
\end{align*}
\]

\[ \langle \text{tick } c \text{ in } e, p \rangle \mapsto \langle e, p - c \rangle \]
Expressions in $\text{Re}^2$ are in A-Normal-Form, i.e., syntactic forms in non-tail positions allow only variables and values.

\[
\Gamma; \Psi; \Phi \vdash e : S
\]

\[\begin{align*}
(\text{T:TRUE}) & \quad (\text{T:NIL}) \\
\Gamma; \Psi; \Phi \vdash \text{true} : \{ v : \text{bool} \mid v = T \} & \quad \Gamma \vdash R \text{ type} \\
& \quad \Gamma; \Psi; \Phi \vdash \text{nil} : \{ v : L(R) \mid \text{size}(v) = 0 \}
\end{align*}\]
Static Semantics

Language Design

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

\[ \Gamma; \Psi; \Phi \vdash e : S \]

\[ \frac{\text{(T:True)}}{\Gamma; \Psi; \Phi \vdash \text{true} : \{ \nu : \text{bool} | \nu = \top \}} \]

\[ \frac{\text{(T:Nil)}}{\Gamma; \Psi; \Phi \vdash \text{nil} : \{ \nu : L(R) | \text{size}(\nu) = 0 \}} \]
Expressions in $\text{Re}^2$ are in A-Normal-Form, i.e., syntactic forms in non-tail positions allow only variables and values.

\[
\Gamma; \Psi; \Phi \vdash e : S
\]

\(\text{(T:TRUE)}\)
\[
\Gamma; \Psi; \Phi \vdash \text{true} : \{v : \text{bool} \mid v = \top\}
\]

\(\text{(T:NIL)}\)
\[
\Gamma; \Psi; \Phi \vdash \text{nil} : \{v : L(R) \mid \text{size}(v) = 0\}
\]

\(\Gamma \vdash R \text{ type}\)
Static Semantics

(T:Cond)
\[
\begin{align*}
\Gamma(x) &= \text{bool} & \Gamma; \Psi \land x; \Phi \vdash e_1 : R \\
\Gamma; \Psi \land \neg x; \Phi \vdash e_2 : R \\
\hline
\Gamma; \Psi; \Phi \vdash \textbf{if } x \textbf{ then } e_1 \textbf{ else } e_2 : R
\end{align*}
\]

(T:AppFO)
\[
\begin{align*}
\Gamma(x_1) &= x : \{v : B \mid \psi\}^\phi \rightarrow R & \Gamma(x_2) &= \{v : B \mid \psi\} \\
\hline
\Gamma; \top; [x_2/v] \phi \vdash x_1(x_2) : R
\end{align*}
\]

(T:MatL)
\[
\begin{align*}
\Gamma(x) &= L(T^\phi) & \Gamma; \Psi \land \text{size}(x) = 0; \Phi \vdash e_1 : R' \\
\Gamma, x_1 : T, x_2 : L(T^\phi); \Psi \land \text{size}(x) = \text{size}(x_2) + 1; \Phi + [x_1/v] \phi \vdash e_2 : R' \\
\hline
\Gamma; \Psi; \Phi \vdash \textbf{match } x \textbf{ with } \{[] \rightarrow e_1 \mid x_1 :: x_2 \rightarrow e_2\} : R'
\end{align*}
\]
Static Semantics

(T:Cond)
\[
\begin{align*}
\Gamma(x) &= \text{bool} & \Gamma; \Psi \land x; \Phi \vdash e_1 : R \\
\Gamma; \Psi \land \neg x; \Phi \vdash e_2 : R
\end{align*}
\]
\[
\Gamma; \Psi; \Phi \vdash \text{if } x \text{ then } e_1 \text{ else } e_2 : R
\]

(T:AppFO)
\[
\begin{align*}
\Gamma(x_1) &= x : \{v : B | \psi\}^\phi \rightarrow R & \Gamma(x_2) &= \{v : B | \psi\} \\
\Gamma; \Psi \land \text{size}(x) = 0; \Phi \vdash e_1 : R' \\
\Gamma, x_1 : T, x_2 : L(T^\phi); \Psi \land \text{size}(x) = \text{size}(x_2) + 1; \Phi + [x_1/v]\phi \vdash e_2 : R'
\end{align*}
\]
\[
\Gamma; \Psi; \Phi \vdash \text{match } x \text{ with } \{ [] \mapsto e_1 \mid x_1 :: x_2 \mapsto e_2 \} : R'
\]
Static Semantics

(T:Cond)
\[ \begin{array}{llll}
\Gamma(x) = \text{bool} & \Gamma; \Psi \land x; \Phi \vdash e_1 : R & \Gamma; \Psi \land \neg x; \Phi \vdash e_2 : R \\
\hline
\Gamma; \Psi; \Phi \vdash \text{if } x \text{ then } e_1 \text{ else } e_2 : R
\end{array} \]

(T:AppFO)
\[ \begin{array}{ll}
\Gamma(x_1) = x : \{v : B | \psi\}^\phi \rightarrow R & \Gamma(x_2) = \{v : B | \psi\} \\
\hline
\Gamma; \top; [x_2/v] \phi \vdash x_1(x_2) : R
\end{array} \]

(T:MatchL)
\[ \begin{array}{llll}
\Gamma(x) = L(T^\phi) & \Gamma; \Psi \land \text{size}(x) = 0; \Phi \vdash e_1 : R' \\
\Gamma, x_1 : T, x_2 : L(T^\phi); \Psi \land \text{size}(x) = \text{size}(x_2) + 1; \Phi + [x_1/v] \phi \vdash e_2 : R' \\
\hline
\Gamma; \Psi; \Phi \vdash \text{match } x \text{ with } \{[] \leftarrow e_1 \mid x_1 :: x_2 \leftarrow e_2\} : R'
\end{array} \]
**Meta Theory**

**Progress**

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

**Consistency**

If $q \vdash e : S$ 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 $\nu$ 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 \( q \vdash e : S \) 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 \( q \vdash e : S \), \( p \geq q \), and \( \langle e, p \rangle \mapsto \langle e', p' \rangle \), then \( q \vdash e' : S \).

## Consistency

If \( q \vdash e : S \) 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.
• Boolean values are interpreted as \{\top, \bot\}. Lists are interpreted as sizes.
• Develop a denotational semantics for the refinement and resource annotations.
Interpretation into Refinement Logic

\[ \mathcal{I}(\text{true}) = \top \]
\[ \mathcal{I}(\text{false}) = \bot \]
\[ \mathcal{I}(\text{nil}) = 0 \]
\[ \mathcal{I}(\text{cons}(v_1, v_2)) = \mathcal{I}(v_2) + 1 \]

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

\[ \mathcal{I}(\text{true}) = \top \quad \mathcal{I}(\text{false}) = \bot \quad \mathcal{I}(\text{nil}) = 0 \quad \mathcal{I}(\text{cons}(v_1, v_2)) = \mathcal{I}(v_2) + 1 \]

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

Consistency: Intuition

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

Consistency: Formalization

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

Consistency: Intuition

If ·; ·; q ⊨ 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 ·; ·; q ⊨ 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 \( \models q \geq [\mathcal{I}(e)/v]\phi \).