Efficient Type-Checking for Amortised Heap-Space Analysis

Dulma Rodriguez

Department of Computer Science
Ludwig-Maximilians-University Munich

Types 2009, Aussois, France
May 15th, 2009

Introduction

- System RAJA (Hofmann and Jost, ESOP 2006)
  - Type system for Java-like programs.
  - Compile-time analysis of heap-space requirements.
  - Amortised complexity analysis.

- Algorithmic type-checking
  - Result: Typechecker for RAJA programs.
  - Challenges:
    - Algorithmic presentation of the typing rules.
    - Elimination of non-syntax-directed rules.

Outline

Amortised Analysis of Heap-Usage

- Free-list based model
  - Deallocation in C/C++ style with primitive dispose.
  - Object creation: takes memory units from free-list.
  - Object destruction: returns memory units to free-list.
- Goal:
  - Upper bound on the size of the free-list.
  - As function of the input.
- Amortised analysis
  - Types assign each heap configuration statically a potential.
  - Object creation: must pay using potential from input.
  - Potential of consumed input: upper bound on heap space consumption.

Object-Oriented Language: FJEU

$$
\begin{array}{rll}
c ::= & \texttt{class } C \ [\texttt{extends } D] \ \{A_1; \ldots; A_k; M_1 \cdots M_j \ \} & \\
A ::= & C \ a & \\
M ::= & C_0 \ m(C_1 \ x_1, \ldots, C_j \ x_j) \{\texttt{return } e;\} & \\
e ::= & x & \text{(Variable)} \\
| & \mathsf{null} & \text{(Constant)} \\
| & \mathsf{new}\ C & \text{(Construction)} \\
| & \mathsf{free}\,(x) & \text{(Destruction)} \\
| & (C)x & \text{(Cast)} \\
| & x.a_i & \text{(Access)} \\
| & x.a_i \leftarrow x & \text{(Update)} \\
| & x.m(x_1, \ldots, x_j) & \text{(Invocation)} \\
| & \mathsf{if}\ x\ \mathsf{instanceof}\ C\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 & \text{(Conditional)} \\
| & \mathsf{let}\ D\, x = e_1\ \mathsf{in}\ e_2 & \text{(Let)}
\end{array}
$$

- $\approx$ Featherweight Java (Igarashi, Pierce, Wadler, OOPSLA'99) + imperative field update.

Copy example

```
class List {
    List copy() {
        return null;
    }
}

class Nil extends List {
    List copy() {
        return this;
    }
}
```

```
class Cons extends List {
    int elem;
    List next;

    List copy() {
        let Cons res = new Cons in
        let Cons _ = res.elem ← this.elem in
        let List rnext = this.next.copy() in
        let List _ = res.next ← rnext in
        return res;
    }
}
```

- Space consumption of $l.copy() : n = \text{length}(l)$

Copy example

```
class List {
    List copy() {
        return null;
    }
}

class Nil extends List {
    List copy() {
        return this;
    }
}
```

```
class Cons extends List {
    int elem;
    List next;

    List copy() {
        let Cons res = new Cons in
        let Cons _ = res.elem ← this.elem in
        let List rnext = this.next.copy() in
        let List _ = res.next ← rnext in
        return res;
    }
}
```

- Space consumption of $l.copy() : n = \text{length}(l)$

Copy example

```
class List {
    List copy() {
        return null;
    }
}

class Nil extends List {
    List copy() {
        return this;
    }
}
```

```
class Cons extends List {
    int elem;
    List next;

    List copy() {
        let Cons res = new Cons in
        let Cons _ = res.elem ← this.elem in
        let List rnext = this.next.copy() in
        let List _ = res.next ← rnext in
        return res;
    }
}
```

- Space consumption of $l.copy() : n = \text{length}(l)$

Sketch of RAJA System

- RAJA program P = *annotated* FJEU program.
  1. Set of *views* $\mathcal{V}$.
  2. For each class C and view $\mathbf{r}$ we have an annotated version $C^{\mathbf{r}}$.
  3. Potential:
     - $\Diamond(C^{\mathbf{r}})$: Class $\times$ View $\to \mathbb{Q}^+$
  4. (Get- and set-) views for attributes:
     - $A^{\text{get}}(C^{\mathbf{r}}, a)$: Class $\times$ View $\times$ Field $\to$ View (get-view)
     - $A^{\text{set}}(C^{\mathbf{r}}, a)$: Class $\times$ View $\times$ Field $\to$ View (set-view)
  5. RAJA types for methods:
     - $M(C^{\mathbf{r}}, m)$: Class $\times$ View $\times$ Method $\to$
       $\mathcal{P}$(Views of Arguments $\to$ Effect $\times$ View of Result)
       $(\mathbf{r}_1, \ldots, \mathbf{r}_j \xrightarrow{m/m'} \mathbf{r}_0)$
     - Effect: pair of numbers m, m' representing potential consumed and released by the method.

Sketch of RAJA System

- RAJA program P = *annotated* FJEU program.
  1. Set of *views* $\mathcal{V}$.
  2. For each class C and view $\mathbf{r}$ we have an annotated version $C^{\mathbf{r}}$.
  3. Potential:
     - $\Diamond(C^{\mathbf{r}})$: Class $\times$ View $\to \mathbb{Q}^+$
  4. (Get- and set-) views for attributes:
     - $A^{\text{get}}(C^{\mathbf{r}}, a)$: Class $\times$ View $\times$ Field $\to$ View (get-view)
     - $A^{\text{set}}(C^{\mathbf{r}}, a)$: Class $\times$ View $\times$ Field $\to$ View (set-view)
  5. RAJA types for methods:
     - $M(C^{\mathbf{r}}, m)$: Class $\times$ View $\times$ Method $\to$
       $\mathcal{P}$(Views of Arguments $\to$ Effect $\times$ View of Result)
       $(\mathbf{r}_1, \ldots, \mathbf{r}_j \xrightarrow{m/m'} \mathbf{r}_0)$
     - Effect: pair of numbers m, m' representing potential consumed and released by the method.

Sketch of RAJA System

- RAJA program P = *annotated* FJEU program.
  1. Set of *views* $\mathcal{V}$.
  2. For each class C and view $\mathbf{r}$ we have an annotated version $C^{\mathbf{r}}$.
  3. Potential:
     - $\Diamond(C^{\mathbf{r}})$: Class $\times$ View $\to \mathbb{Q}^+$
  4. (Get- and set-) views for attributes:
     - $\mathsf{A}^{\mathsf{get}}(C^{\mathbf{r}}, a)$: Class $\times$ View $\times$ Field $\to$ View (get-view)
     - $\mathsf{A}^{\mathsf{set}}(C^{\mathbf{r}}, a)$: Class $\times$ View $\times$ Field $\to$ View (set-view)
  5. RAJA types for methods:
     - $\mathsf{M}(C^{\mathbf{r}}, m)$: Class $\times$ View $\times$ Method $\to$ $\mathcal{P}$(Views of Arguments $\to$ Effect $\times$ View of Result)
       $(\mathbf{r}_1, \ldots, \mathbf{r}_j \xrightarrow{m/m'} \mathbf{r}_0)$
     - Effect: pair of numbers m, m' representing potential consumed and released by the method.

Sketch of RAJA System

- RAJA program P = *annotated* FJEU program.
  1. Set of *views* $\mathcal{V}$.
  2. For each class C and view $\mathbf{r}$ we have an annotated version $C^{\mathbf{r}}$.
  3. Potential:
     - $\Diamond(C^{\mathbf{r}})$: Class $\times$ View $\to \mathbb{Q}^+$
  4. (Get- and set-) views for attributes:
     - $\mathsf{A}^{\mathsf{get}}(C^{\mathbf{r}}, a)$: Class $\times$ View $\times$ Field $\to$ View (get-view)
     - $\mathsf{A}^{\mathsf{set}}(C^{\mathbf{r}}, a)$: Class $\times$ View $\times$ Field $\to$ View (set-view)
  5. RAJA types for methods:
     - $\mathsf{M}(C^{\mathbf{r}}, m)$: Class $\times$ View $\times$ Method $\to$ $\mathcal{P}$(Views of Arguments $\to$ Effect $\times$ View of Result)

       $(\mathbf{r_1}, \ldots, \mathbf{r_j} \xrightarrow{m/m'} \mathbf{r_0})$
     - Effect: pair of numbers m, m' representing potential consumed and released by the method.

Copy example in RAJA

```
class List {
    ...
}
class Nil extends List {
    ...
}
class Cons extends List {
    rich: pot = 1;
    poor: pot = 0;
    rich: int elem;
    poor: int elem;
    rich: List<rich,rich> next;
    poor: List<poor,poor> next;

    rich: List<poor>,0 copy(0) {
        let res = new Cons in
        let _ = res.elem ← this.elem in
        let rnext = this.next.copy() in
        let _ = res.next ← rnext in
        return res;
    }
}
```

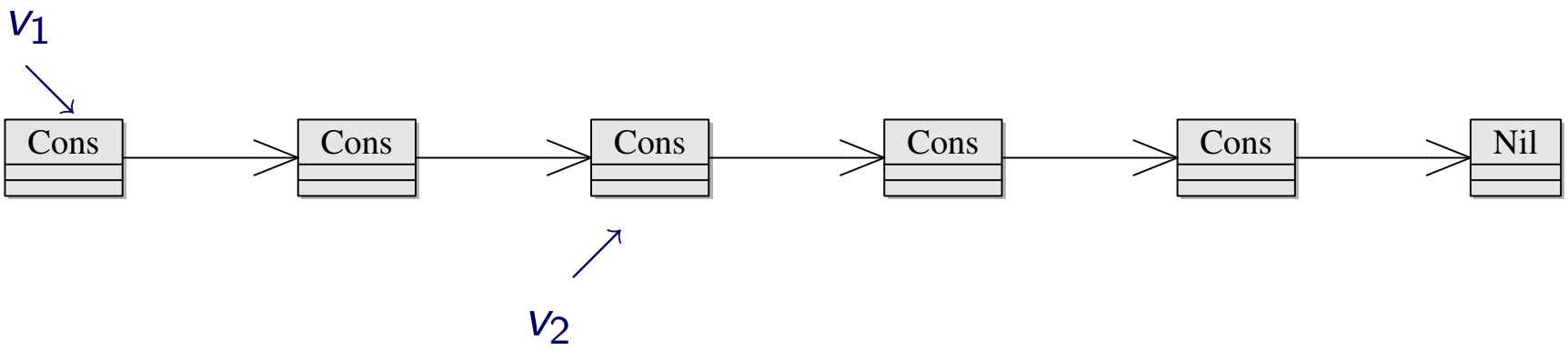
$\Diamond(\cdot)$	rich	poor
List	0	0
Nil	0	0
Cons	1	0

	Cons^{rich}	Cons^{poor}
$A^{get}(\cdot, \text{next})$	rich	poor
$A^{set}(\cdot, \text{next})$	rich	poor

	List^{rich}	Nil^{rich}	Cons^{rich}
$M(\cdot, \text{copy})$	$() \xrightarrow{0/0} \text{poor}$		

Potential

- Let σ be a heap and v_1 and v_2 the only reachable locations in the heap from the stack η:

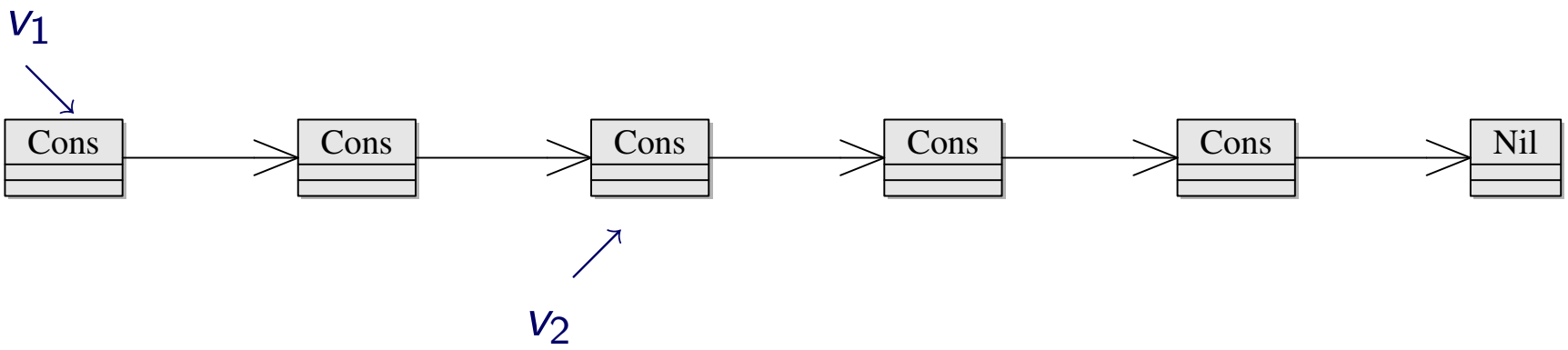


$$\Phi_\sigma(v_1 : \mathbf{r}) = \begin{cases} 5 & \mathbf{r} = \mathit{rich} \\ 0 & \mathbf{r} = \mathit{poor} \end{cases} \qquad \Phi_\sigma(v_2 : \mathbf{r}) = \begin{cases} 3 & \mathbf{r} = \mathit{rich} \\ 0 & \mathbf{r} = \mathit{poor} \end{cases}$$

$$\Phi_\sigma(\eta : \Gamma) = \Phi_\sigma(v_1 : \mathbf{r}) + \Phi_\sigma(v_2 : \mathbf{r}) = 0 \vee 3 \vee 5 \vee 8$$

Potential

- Let σ be a heap and v_1 and v_2 the only reachable locations in the heap from the stack η:

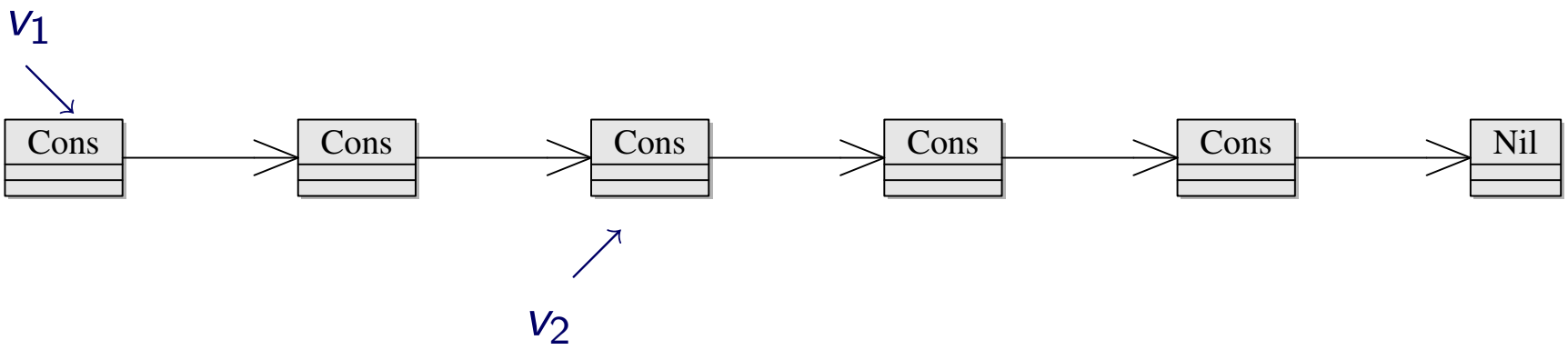


$$\Phi_\sigma(v_1 : \mathbf{r}) = \begin{cases} 5 & \mathbf{r} = \textit{rich} \\ 0 & \mathbf{r} = \textit{poor} \end{cases} \qquad \Phi_\sigma(v_2 : \mathbf{r}) = \begin{cases} 3 & \mathbf{r} = \textit{rich} \\ 0 & \mathbf{r} = \textit{poor} \end{cases}$$

$$\Phi_\sigma(\eta : \Gamma) = \Phi_\sigma(v_1 : \mathbf{r}) + \Phi_\sigma(v_2 : \mathbf{r}) = 0 \vee 3 \vee 5 \vee 8$$

Potential

- Let σ be a heap and v_1 and v_2 the only reachable locations in the heap from the stack η:

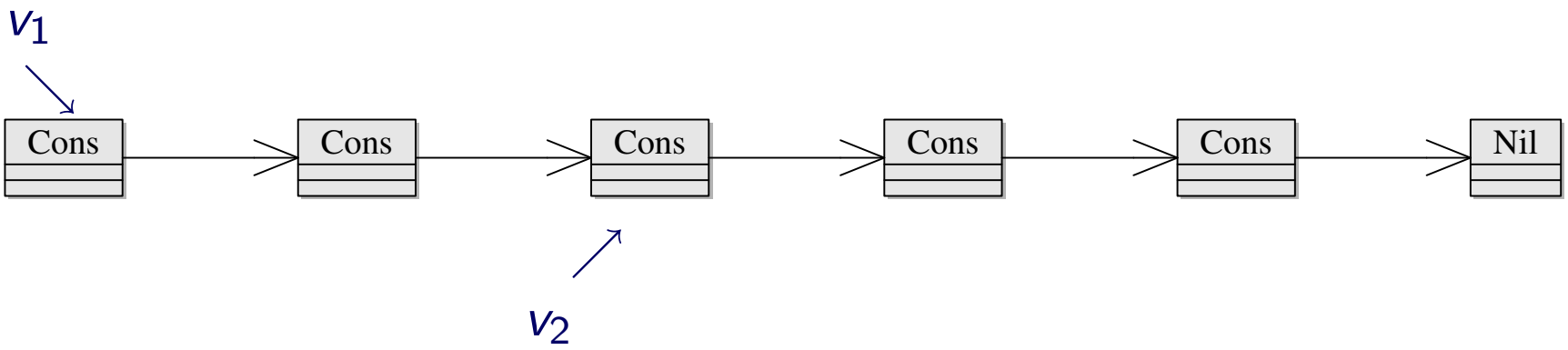


$$\Phi_\sigma(v_1 : \mathbf{r}) = \begin{cases} 5 & \mathbf{r} = \mathit{rich} \\ 0 & \mathbf{r} = \mathit{poor} \end{cases} \qquad \Phi_\sigma(v_2 : \mathbf{r}) = \begin{cases} 3 & \mathbf{r} = \mathit{rich} \\ 0 & \mathbf{r} = \mathit{poor} \end{cases}$$

$$\Phi_\sigma(\eta : \Gamma) = \Phi_\sigma(v_1 : \mathbf{r}) + \Phi_\sigma(v_2 : \mathbf{r}) = 0 \vee 3 \vee 5 \vee 8$$

Potential

- Let σ be a heap and v_1 and v_2 the only reachable locations in the heap from the stack η:



$$\Phi_\sigma(v_1 : \mathbf{r}) = \begin{cases} 5 & \mathbf{r} = \textit{rich} \\ 0 & \mathbf{r} = \textit{poor} \end{cases} \qquad \Phi_\sigma(v_2 : \mathbf{r}) = \begin{cases} 3 & \mathbf{r} = \textit{rich} \\ 0 & \mathbf{r} = \textit{poor} \end{cases}$$

$$\Phi_\sigma(\eta : \Gamma) = \Phi_\sigma(v_1 : \mathbf{r}) + \Phi_\sigma(v_2 : \mathbf{r}) = 0 \vee 3 \vee 5 \vee 8$$

Main result

- $\eta, \sigma \vdash e \rightsquigarrow v, \tau$ means:
  - expression e evaluates successfully to value v beginning with stack η and heap σ and ending with heap τ (with an unbounded freelist).

- Typing judgement $\Gamma \vdash^{n}_{n'} e : C^{\mathbf{r}}$ so that:
  - if $\Gamma \vdash^{n}_{n'} e : C^{\mathbf{r}}$ and $\eta, \sigma \vdash e \rightsquigarrow v, \tau$ then
  - e evaluates successfully with a bounded freelist of at least $n + \Phi_{\sigma}(\eta : \Gamma)$ units.

FJEU Typing rules

$$\frac{}{\emptyset \vdash \text{new } C : C}\ (\Diamond New) \qquad \frac{C.a = D}{x : C \vdash x.a : D}\ (\Diamond Access)$$

$$\frac{}{x : C \vdash \text{free}(x) : E}\ (\Diamond Free) \qquad \frac{C.a = D}{x : C, y : D \vdash x.a \leftarrow y : C}\ (\Diamond Update)$$

$$\frac{\Gamma_1 \vdash e_1 : D \qquad \Gamma_2, x : D \vdash e_2 : C}{\Gamma_1, \Gamma_2 \vdash \text{let } x = e_1 \text{ in } e_2 : C}\ (\Diamond Let)$$

$$\frac{(E_1, \ldots, E_j \rightarrow E_0) = M(C, m)}{x : C, y_1 : E_1, \ldots, y_j : E_j \vdash x.m(y_1, \ldots, y_j) : E_0}\ (\Diamond Invocation)$$

$$\frac{x \in \Gamma \qquad \Gamma \vdash e_1 : C \qquad \Gamma \vdash e_2 : C}{\text{if } x \text{ instanceof } E \text{ then } e_1 \text{ else } e_2 : C}\ (\Diamond Conditional)$$

RAJA typing rules

$$\frac{}{\emptyset \vdash^{\frac{1+\Diamond(C^r)}{0}} \text{new } C : C^{\mathbf{r}}} \ (\Diamond New) \qquad \frac{\mathsf{A}^{\mathsf{get}}(C^{\mathbf{r}}, a) = \mathbf{s} \qquad C.a = D}{x : C^{\mathbf{r}} \vdash^{\frac{0}{0}} x.a : D^{\mathbf{s}}} \ (\Diamond Access)$$

$$\frac{}{x : C^{\mathbf{r}} \vdash^{\frac{0}{1+\Diamond(C^r)}} \text{free}\,(x) : E^{\mathbf{r}}} \ (\Diamond Free) \qquad \frac{\mathsf{A}^{\mathsf{set}}(C^{\mathbf{r}}, a) = \mathbf{s} \qquad C.a = D}{x : C^{\mathbf{r}}, y : D^{\mathbf{s}} \vdash^{\frac{0}{0}} x.a \leftarrow y : C^{\mathbf{r}}} \ (\Diamond Update)$$

$$\frac{\Gamma_1 \vdash^{\frac{n}{n'}} e_1 : D^{\mathbf{s}} \qquad \Gamma_2, x : D^{\mathbf{s}} \vdash^{\frac{n'}{n''}} e_2 : C^{\mathbf{r}}}{\Gamma_1, \Gamma_2 \vdash^{\frac{n}{n''}} \text{let } x = e_1 \text{ in } e_2 : C^{\mathbf{r}}} \ (\Diamond Let)$$

$$\frac{(E_1^{\mathbf{q_1}}, \ldots, E_j^{\mathbf{q_j}} \xrightarrow{n/n'} E_0^{\mathbf{q_0}}) \in M(C^{\mathbf{r}}, m)}{x : C^{\mathbf{r}}, y_1 : E_1^{\mathbf{q_1}}, \ldots, y_j : E_j^{\mathbf{q_j}} \vdash^{\frac{n}{n'}} x.m\,(y_1, \ldots, y_j) : E_0^{\mathbf{q_0}}} \ (\Diamond Invocation)$$

$$\frac{x \in \Gamma \qquad \Gamma \vdash^{\frac{n}{n'}} e_1 : C^{\mathbf{r}} \qquad \Gamma \vdash^{\frac{n}{n'}} e_2 : C^{\mathbf{r}}}{\Gamma \vdash^{\frac{n}{n'}} \text{if } x \text{ instanceof } E \text{ then } e_1 \text{ else } e_2 : C^{\mathbf{r}}} \ (\Diamond Conditional)$$

Subtyping and weakening

- There is a weakening rule:

$$\frac{n \geq u \quad n - n' \geq u - u' \quad \Theta \vdash^{u}_{u'} e : D^{s} \quad \Gamma <: \Theta \quad D^{s} <: C^{r}}{\Gamma \vdash^{n}_{n'} e : C^{r}} \; (\Diamond Waste)$$

- where RAJA subtyping $<:$ is an extension of FJEU subtyping and is defined coinductively.

Definition (Subtyping of RAJA types)

$C^{r} <: D^{s} \iff$ for each $E <: C$, $F <: D$ with $E <: F$:

$$\Diamond(E^{r}) \geq \Diamond(F^{s}) \tag{2.1}$$

$$\forall a \in \mathsf{A}(F) \,.\, F.a^{\mathsf{A}^{get}(\mathbf{E}^{r};a)} <: F.a^{\mathsf{A}^{get}(\mathbf{F}^{s};a)} \tag{2.2}$$

$$\forall a \in \mathsf{A}(F) \,.\, F.a^{\mathsf{A}^{set}(\mathbf{F}^{s};a)} <: F.a^{\mathsf{A}^{set}(\mathbf{E}^{r};a)} \tag{2.3}$$

$$\ldots \tag{2.4}$$

Subtyping and weakening

- There is a weakening rule:

$$\frac{n \geq u \quad n - n' \geq u - u' \quad \Theta \vdash^{u}_{u'} e : D^{\mathsf{s}} \quad \Gamma <: \Theta \quad D^{\mathsf{s}} <: C^{\mathsf{r}}}{\Gamma \vdash^{n}_{n'} e : C^{\mathsf{r}}} \ (\Diamond Waste)$$

- where RAJA subtyping $<:$ is an extension of FJEU subtyping and is defined coinductively.

Definition (Subtyping of RAJA types)

$C^{\mathsf{r}} <: D^{\mathsf{s}} \iff$ for each $E <: C$, $F <: D$ with $E <: F$:

$$\Diamond(E^{\mathsf{r}}) \geq \Diamond(F^{\mathsf{s}}) \tag{2.1}$$

$$\forall a \in \mathsf{A}(F) \,.\, F.a^{\mathsf{A}^{\mathsf{get}}(\mathsf{E}^{\mathsf{r}};\mathsf{a})} <: F.a^{\mathsf{A}^{\mathsf{get}}(\mathsf{F}^{\mathsf{s}};\mathsf{a})} \tag{2.2}$$

$$\forall a \in \mathsf{A}(F) \,.\, F.a^{\mathsf{A}^{\mathsf{set}}(\mathsf{F}^{\mathsf{s}};\mathsf{a})} <: F.a^{\mathsf{A}^{\mathsf{set}}(\mathsf{E}^{\mathsf{r}};\mathsf{a})} \tag{2.3}$$

$$\ldots \tag{2.4}$$

Sharing relation

- The rule ($\Diamond Share$) enables several uses of a variable.

$$\frac{\Gamma, y_1 : D^{q_1}, \ldots, y_n : D^{q_n} \vdash^{n}_{n'} e : C^{r} \qquad \curlyvee(D^{s} \,|\, D^{q_1}, \ldots, D^{q_n})}{\Gamma, x : D^{s} \vdash^{n}_{n'} e[x/y_1, \ldots, x/y_n] : C^{r}} \; (\Diamond Share)$$

Definition (Sharing Relation)

If $\curlyvee(C^{r} \,|\, D^{s_1}, \ldots, D^{s_n})$ then for all $E <: C$, $F <: D$ with $E <: F$:

$$\Diamond(E^{r}) \geq \sum_i \Diamond(F^{s_i}) \tag{2.5}$$

$$\forall i \,.\, E^{r} <: F^{s_i} \ldots \tag{2.6}$$

- $C^{r} <: D^{s} \iff \curlyvee(C^{r} \,|\, D^{s})$

$$\Gamma, l : \mathsf{List}^{\mathsf{rich}} \vdash^{n}_{n'} \mathsf{let}\ nl = l.\mathsf{copy}()\ \mathsf{in}\ l.\mathsf{copy}() : \mathsf{List}^{\mathsf{poor}}$$

- is not allowed because $\curlyvee\left(\mathsf{List}^{\mathsf{rich}} \,\middle|\, \mathsf{List}^{\mathsf{rich}}, \mathsf{List}^{\mathsf{rich}}\right)$ does not hold.

Sharing relation

- The rule $(\Diamond Share)$ enables several uses of a variable.

$$\frac{\Gamma, y_1 : D^{q_1}, \ldots, y_n : D^{q_n} \vdash^{n}_{n'} e : C^{r} \qquad \curlyvee(D^{s} \,|\, D^{q_1}, \ldots, D^{q_n})}{\Gamma, x : D^{s} \vdash^{n}_{n'} e[x/y_1, \ldots, x/y_n] : C^{r}} \; (\Diamond Share)$$

Definition (Sharing Relation)

If $\curlyvee(C^{r} \,|\, D^{s_1}, \ldots, D^{s_n})$ then for all $E <: C$, $F <: D$ with $E <: F$:

$$\Diamond(E^{r}) \geq \sum_i \Diamond(F^{s_i}) \tag{2.5}$$

$$\forall i \,.\, E^{r} <: F^{s_i} \ldots \tag{2.6}$$

- $C^{r} <: D^{s} \iff \curlyvee(C^{r} \,|\, D^{s})$

$$\Gamma, l : \mathrm{List}^{\mathrm{rich}} \vdash^{n}_{n'} \mathrm{let}\ nl = l.\mathrm{copy}()\ \mathrm{in}\ l.\mathrm{copy}() : \mathrm{List}^{\mathrm{poor}}$$

- is not allowed because $\curlyvee\left(\mathrm{List}^{\mathrm{rich}} \,\middle|\, \mathrm{List}^{\mathrm{rich}}, \mathrm{List}^{\mathrm{rich}}\right)$ does not hold.

Sharing relation

- The rule $(\Diamond Share)$ enables several uses of a variable.

$$\frac{\Gamma, y_1 : D^{q_1}, \ldots, y_n : D^{q_n} \vdash^{n}_{n'} e : C^{r} \qquad \curlyvee(D^{s} \,|\, D^{q_1}, \ldots, D^{q_n})}{\Gamma, x : D^{s} \vdash^{n}_{n'} e[x/y_1, \ldots, x/y_n] : C^{r}} \; (\Diamond Share)$$

Definition (Sharing Relation)

If $\curlyvee(C^{r} \,|\, D^{s_1}, \ldots, D^{s_n})$ then for all $E <: C$, $F <: D$ with $E <: F$:

$$\Diamond(E^{r}) \geq \sum_i \Diamond(F^{s_i}) \tag{2.5}$$

$$\forall i \,.\, E^{r} <: F^{s_i} \ldots \tag{2.6}$$

- $C^{r} <: D^{s} \iff \curlyvee(C^{r} \,|\, D^{s})$

$$\Gamma, l : \mathsf{List}^{\mathsf{rich}} \vdash^{n}_{n'} \mathsf{let}\ nl = l.\mathsf{copy}()\ \mathsf{in}\ l.\mathsf{copy}() : \mathsf{List}^{\mathsf{poor}}$$

- is not allowed because $\curlyvee\left(\mathsf{List}^{\mathsf{rich}} \,\middle|\, \mathsf{List}^{\mathsf{rich}}, \mathsf{List}^{\mathsf{rich}}\right)$ does not hold.

Challenges and contributions

- Challenges for implementing RAJA:
  - The rules ($\Diamond Waste$) and ($\Diamond Share$) are non-syntax-directed.
  - The intermediate views in the rules ($\Diamond Share$) needs to be found out.
- Contributions
  - Algorithmic (syntax-directed) typing rules.
  - Algorithmic views.
  - Improved definitions of subtyping and sharing.

Challenges and contributions

- Challenges for implementing RAJA:
  - The rules ($\Diamond Waste$) and ($\Diamond Share$) are non-syntax-directed.
  - The intermediate views in the rules ($\Diamond Share$) needs to be found out.

- Contributions
  - Algorithmic (syntax-directed) typing rules.
  - Algorithmic views.
  - Improved definitions of subtyping and sharing.

Algorithmic views

Algorithmic views (δ, γ) provide:

- l.u.b. $(r \vee s)$ and g.l.b.$(r \wedge s)$ for RAJA types.
- A formal addition operation $(r + s)$ on views.
- $\curlyvee(C^r \,|\, C^{s_1}, C^{s_2}) \iff C^r <: C^{s_1+s_2}$

$$
\begin{aligned}
\Diamond(C^{s_1 \wedge s_2}) &= \max(\Diamond(C^{s_1}), \Diamond(C^{s_2})) \\
\Diamond(C^{s_1 \vee s_2}) &= \min(\Diamond(C^{s_1}), \Diamond(C^{s_2})) \\
\Diamond(C^{s_1+s_2}) &= \Diamond(C^{s_1}) + \Diamond(C^{s_2}) \\
&\dots
\end{aligned}
$$

Algorithmic views

Algorithmic views (δ, γ) provide:

- l.u.b. $(r \vee s)$ and g.l.b.$(r \wedge s)$ for RAJA types.
- A formal addition operation $(r + s)$ on views.
- $\curlyvee(C^r \,|\, C^{s_1}, C^{s_2}) \iff C^r <: C^{s_1+s_2}$

$$
\begin{array}{lcl}
\Diamond(C^{s_1 \wedge s_2}) & = & \max(\Diamond(C^{s_1}), \Diamond(C^{s_2})) \\
\Diamond(C^{s_1 \vee s_2}) & = & \min(\Diamond(C^{s_1}), \Diamond(C^{s_2})) \\
\Diamond(C^{s_1+s_2}) & = & \Diamond(C^{s_1}) + \Diamond(C^{s_2}) \\
 & \ldots &
\end{array}
$$

Idea of the algorithm

- Subtyping and sharing are integrated in the rules.
- Variable occurrences are annotated with a view.
- The views from the different occurrences are summed up.
- Judgement: $\Delta^{\Psi} \vdash^{n}_{n'} e^{\circ} \leftleftarrows C^{\gamma}$
- where Δ is an FJEU context and Ψ a map from variables to *algorithmic* views.
- $\Rightarrow \Delta^{\Psi}$ is a map from variables to *algorithmic* RAJA types.

$$\mathsf{typecheck}(\Delta, e^{\circ}, C^{\gamma}) = \begin{cases} (\Psi, n, n') & \text{if } \Delta^{\Psi} \vdash^{n}_{n'} e^{\circ} \leftleftarrows C^{\gamma} \\ \mathsf{fail} & \text{otherwise} \end{cases}$$

$$\frac{(() \xrightarrow{0/0} \mathsf{List}^{\mathsf{poor}}) \in M(\mathsf{Cons}^{\mathsf{rich}}, m)}{\mathsf{this}\colon \mathsf{Cons}^{\mathsf{rich}} \vdash^{0}_{0} \mathsf{this}^{\mathsf{rich}}.\mathsf{copy}() \leftleftarrows \mathit{List}^{\mathsf{poor}}} \qquad \frac{\mathsf{Cons} <: \mathsf{List}}{\mathsf{this}\colon \mathsf{Cons}^{\mathsf{poor}}, \mathsf{res}\colon \mathit{List}^{\emptyset} \vdash^{0}_{0} \mathsf{this}^{\mathsf{poor}} \leftleftarrows \mathsf{List}^{\mathsf{poor}}}$$

$$\mathsf{this}\colon \mathsf{Cons}^{\mathsf{rich}+\mathsf{poor}} \vdash^{0}_{0} \mathsf{let}\,\mathsf{List}\,\mathsf{res} = \mathsf{this}^{\mathsf{rich}}.\mathsf{copy}()\,\mathsf{in}\,\mathsf{this}^{\mathsf{poor}} \leftleftarrows \mathsf{List}^{\mathsf{poor}}$$

Idea of the algorithm

- Subtyping and sharing are integrated in the rules.
- Variable occurrences are annotated with a view.
- The views from the different occurrences are summed up.
- Judgement: $\Delta^{\Psi} \vdash^{n}_{n'} e^{\circ} \leftleftarrows C^{\gamma}$
- where Δ is an FJEU context and Ψ a map from variables to *algorithmic* views.
- $\Rightarrow \Delta^{\Psi}$ is a map from variables to *algorithmic* RAJA types.

$$\mathsf{typecheck}(\Delta, e^{\circ}, C^{\gamma}) = \begin{cases} (\Psi, n, n') & \text{if } \Delta^{\Psi} \vdash^{n}_{n'} e^{\circ} \leftleftarrows C^{\gamma} \\ \mathsf{fail} & \text{otherwise} \end{cases}$$

$$\frac{(()\xrightarrow{0/0}\mathsf{List}^{\mathsf{poor}}) \in M(\mathsf{Cons}^{\mathsf{rich}}, m) \qquad \mathsf{Cons} <: \mathsf{List}}{\mathsf{this}: \mathsf{Cons}^{\mathsf{rich}} \vdash^{0}_{0} \mathsf{this}^{\mathsf{rich}}.\mathsf{copy}() \leftleftarrows \mathit{List}^{\mathsf{poor}} \qquad \mathsf{this}: \mathsf{Cons}^{\mathsf{poor}}, \mathsf{res}: \mathit{List}^{\emptyset} \vdash^{0}_{0} \mathsf{this}^{\mathsf{poor}} \leftleftarrows \mathsf{List}^{\mathsf{poor}}}$$

$$\mathsf{this}: \mathsf{Cons}^{\mathsf{rich+poor}} \vdash^{0}_{0} \mathsf{let}\ \mathsf{List}\ \mathsf{res} = \mathsf{this}^{\mathsf{rich}}.\mathsf{copy}()\ \mathsf{in}\ \mathsf{this}^{\mathsf{poor}} \leftleftarrows \mathsf{List}^{\mathsf{poor}}$$

Algorithmic RAJA typing rules

- Notation: $\Delta^{\Psi'+\Psi''}$ means that if $\Delta_x^{\Psi'} = C^{\mathbf{r}}$ and $\Delta_x^{\Psi''} = C^{\mathbf{s}}$ then $\Delta_x^{\Psi'+\Psi''} = C^{\mathbf{r}+\mathbf{s}}$.
- $\Delta^{\Psi' \wedge \Psi''}$ is defined similarly.

$$\frac{\Delta^{\Psi'} \vdash^{n}_{n'} e_1^{\circ} \leftleftarrows D^{\gamma_1} \qquad \Delta^{\Psi''}, x : D^{\gamma_1} \vdash^{m}_{m'} e_2^{\circ} \leftleftarrows C^{\gamma_2}}{\Delta^{\Psi'+\Psi''} \vdash^{\max(n,\, n+m-n')}_{\max(m',\, m'+n'-m)} \text{let } D\, x = e_1^{\circ} \text{ in } e_2^{\circ} \leftleftarrows C^{\gamma_2}} \; (\vdash Let)$$

$$\frac{x \in \Delta \quad \Delta^{\Psi'} \vdash^{n}_{n'} e_1^{\circ} \leftleftarrows C^{\gamma} \qquad \Delta^{\Psi''} \vdash^{m}_{m'} e_2^{\circ} \leftleftarrows C^{\gamma} \quad u = \max(m, n)}{\Delta^{\Psi' \wedge \Psi''} \vdash^{u}_{\min(n'+u-n,\, m'+u-m)} \text{if } x \text{ instanceof } E \text{ then } e_1^{\circ} \text{ else } e_2^{\circ} \leftleftarrows C^{\gamma}} \; (\vdash Cond.)$$

$$\frac{p/p' = \min\{\, (E_1^{\mathbf{q}_1}, \ldots, E_j^{\mathbf{q}_j} \xrightarrow{p/p'} E_0^{\mathbf{q}_0}) \in M(G^{\mathbf{r}}, m) \mid \forall i \,.\, F_i^{\mathbf{t}_i} <: E_i^{\mathbf{q}_i}, E_0^{\mathbf{q}_0} <: C^{\gamma} \,\}}{, x : G^{\mathbf{r}} ,^{+} y_1 : F_1^{\mathbf{t}_1} ,^{+} \ldots ,^{+} y_j : F_j^{\mathbf{t}_j} \vdash^{p}_{p'} x^{\mathbf{r}}.m(y_1^{\mathbf{t}_1}, \ldots, y_j^{\mathbf{t}_j}) \leftleftarrows C^{\gamma}} \; (\vdash Inv.)$$

Algorithmic RAJA typing rules

- Notation: $\Delta^{\Psi'+\Psi''}$ means that if $\Delta_x^{\Psi'} = C^{\mathbf{r}}$ and $\Delta_x^{\Psi''} = C^{\mathbf{s}}$ then $\Delta_x^{\Psi'+\Psi''} = C^{\mathbf{r}+\mathbf{s}}$.
- $\Delta^{\Psi' \wedge \Psi''}$ is defined similarly.

$$\frac{\Delta^{\Psi'} \vdash^{\frac{n}{n'}} e_1^\circ \leftleftarrows D^{\gamma_1} \qquad \Delta^{\Psi''}, x : D^{\gamma_1} \vdash^{\frac{m}{m'}} e_2^\circ \leftleftarrows C^{\gamma_2}}{\Delta^{\Psi'+\Psi''} \vdash^{\frac{\max(\, n,\, n+m-n')}{\max(\, m',\, m'+n'-m)}} \text{let } D\, x = e_1^\circ \text{ in } e_2^\circ \leftleftarrows C^{\gamma_2}} \; (\vdash Let)$$

$$\frac{x \in \Delta \quad \Delta^{\Psi'} \vdash^{\frac{n}{n'}} e_1^\circ \leftleftarrows C^{\gamma} \qquad \Delta^{\Psi''} \vdash^{\frac{m}{m'}} e_2^\circ \leftleftarrows C^{\gamma} \quad u = \max(\, m, n)}{\Delta^{\Psi' \wedge \Psi''} \vdash^{\frac{u}{\min(\, n'+u-n,\, m'+u-m)}} \text{if } x \text{ instanceof } E \text{ then } e_1^\circ \text{ else } e_2^\circ \leftleftarrows C^{\gamma}} \; (\vdash Cond.)$$

$$\frac{p/p' = \min\{\, (E_1^{\mathbf{q}_1}, \ldots, E_j^{\mathbf{q}_j} \xrightarrow{p/p'} E_0^{\mathbf{q}_0}) \in M(G^{\mathbf{r}}, m) \mid \forall i \,.\, F_i^{\mathbf{t}_i} <: E_i^{\mathbf{q}_i}, E_0^{\mathbf{q}_0} <: C^{\gamma} \,\}}{, x : G^{\mathbf{r}}, {}^{+}\, y_1 : F_1^{\mathbf{t}_1}, {}^{+} \ldots, {}^{+}\, y_j : F_j^{\mathbf{t}_j} \vdash^{\frac{p}{p'}} x^{\mathbf{r}}.m(y_1^{\mathbf{t}_1}, \ldots, y_j^{\mathbf{t}_j}) \leftleftarrows C^{\gamma}} \; (\vdash Inv.)$$

Algorithmic RAJA typing rules

- Notation: $\Delta^{\Psi'+\Psi''}$ means that if $\Delta_x^{\Psi'} = C^{\mathbf{r}}$ and $\Delta_x^{\Psi''} = C^{\mathbf{s}}$ then $\Delta_x^{\Psi'+\Psi''} = C^{\mathbf{r}+\mathbf{s}}$.
- $\Delta^{\Psi' \wedge \Psi''}$ is defined similarly.

$$\frac{\Delta^{\Psi'} \vdash^{n}_{n'} e_1^{\circ} \leftleftarrows D^{\gamma_1} \qquad \Delta^{\Psi''}, x : D^{\gamma_1} \vdash^{m}_{m'} e_2^{\circ} \leftleftarrows C^{\gamma_2}}{\Delta^{\Psi'+\Psi''} \vdash^{\max(n,\, n+m-n')}_{\max(m',\, m'+n'-m)} \text{let } D\,x = e_1^{\circ} \text{ in } e_2^{\circ} \leftleftarrows C^{\gamma_2}} \; (\vdash \mathit{Let})$$

$$\frac{x \in \Delta \quad \Delta^{\Psi'} \vdash^{n}_{n'} e_1^{\circ} \leftleftarrows C^{\gamma} \qquad \Delta^{\Psi''} \vdash^{m}_{m'} e_2^{\circ} \leftleftarrows C^{\gamma} \quad u = \max(m,n)}{\Delta^{\Psi' \wedge \Psi''} \vdash^{u}_{\min(n'+u-n,\, m'+u-m)} \text{if } x \text{ instanceof } E \text{ then } e_1^{\circ} \text{ else } e_2^{\circ} \leftleftarrows C^{\gamma}} \; (\vdash \mathit{Cond.})$$

$$\frac{p/p' = \min\{\, (E_1^{q_1}, \ldots, E_j^{q_j} \xrightarrow{p/p'} E_0^{q_0}) \in M(G^{\mathbf{r}}, m) \mid \forall i \,.\, F_i^{t_i} <: E_i^{q_i}, E_0^{q_0} <: C^{\gamma} \,\}}{, x : G^{\mathbf{r}} ,^{+} y_1 : F_1^{t_1} ,^{+} \ldots ,^{+} y_j : F_j^{t_j} \vdash^{p}_{p'} x^{\mathbf{r}}.m(y_1^{t_1}, \ldots, y_j^{t_j}) \leftleftarrows C^{\gamma}} \; (\vdash \mathit{Inv.})$$

Algorithmic RAJA typing rules

- Notation: $\Delta^{\Psi'+\Psi''}$ means that if $\Delta_x^{\Psi'} = C^{\mathbf{r}}$ and $\Delta_x^{\Psi''} = C^{\mathbf{s}}$ then $\Delta_x^{\Psi'+\Psi''} = C^{\mathbf{r}+\mathbf{s}}$.
- $\Delta^{\Psi' \wedge \Psi''}$ is defined similarly.

$$\frac{\Delta^{\Psi'} \vdash^{\frac{n}{n'}} e_1^\circ \leftleftarrows D^{\gamma_1} \qquad \Delta^{\Psi''}, x : D^{\gamma_1} \vdash^{\frac{m}{m'}} e_2^\circ \leftleftarrows C^{\gamma_2}}{\Delta^{\Psi'+\Psi''} \vdash^{\frac{\max(\,n,\, n+m-n'\,)}{\max(\,m',\, m'+n'-m\,)}} \text{let } D\,x = e_1^\circ \text{ in } e_2^\circ \leftleftarrows C^{\gamma_2}} \; (\vdash \mathit{Let})$$

$$\frac{x \in \Delta \quad \Delta^{\Psi'} \vdash^{\frac{n}{n'}} e_1^\circ \leftleftarrows C^{\gamma} \qquad \Delta^{\Psi''} \vdash^{\frac{m}{m'}} e_2^\circ \leftleftarrows C^{\gamma} \quad u = \max(\,m,\, n\,)}{\Delta^{\Psi' \wedge \Psi''} \vdash^{\frac{u}{\min(\,n'+u-n,\, m'+u-m\,)}} \text{if } x \text{ instanceof } E \text{ then } e_1^\circ \text{ else } e_2^\circ \leftleftarrows C^{\gamma}} \; (\vdash \mathit{Cond.})$$

$$\frac{p/p' = \min\{\, (E_1^{\mathbf{q_1}}, \ldots, E_j^{\mathbf{q_j}} \xrightarrow{p/p'} E_0^{\mathbf{q_0}}) \in M(G^{\mathbf{r}}, m) \mid \forall i \,.\, F_i^{\mathbf{t_i}} <: E_i^{\mathbf{q_i}}, E_0^{\mathbf{q_0}} <: C^{\gamma} \,\}}{, x : G^{\mathbf{r}} ,^{+} y_1 : F_1^{\mathbf{t_1}} ,^{+} \ldots ,^{+} y_j : F_j^{\mathbf{t_j}} \vdash^{\frac{p}{p'}} x^{\mathbf{r}}.m\,(y_1^{\mathbf{t_1}}, \ldots, y_j^{\mathbf{t_j}}) \leftleftarrows C^{\gamma}} \; (\vdash \mathit{Inv.})$$

Soundness proof

Lemma (Soundness of algorithmic RAJA typing)

If $\Delta^{\Psi} \vdash^{n}_{n'} e \leftleftarrows C^{\gamma}$ *then* $\Delta^{\Psi} \vdash^{n}_{n'} |e| : C^{\gamma}$.

Proof.

By induction on algorithmic typing derivations, using the $(\Diamond Waste)$ rule. □

Completeness proof

Lemma (Completeness of algorithmic RAJA typing)

If $\Gamma \vdash^{n}_{n'} e : C^{\mathbf{r}}$ *then there is an annotated version* e° *of the expression* e *with* $|\Gamma|^{\Psi} \vdash^{u}_{u'} e^{\circ} \leftleftarrows C^{\mathbf{r}}$ *for some* $u \leq n$ *and* $u - u' \leq n - n'$ *so that* $\Gamma <: |\Gamma|^{\Psi}$.

Proof.

By induction on typing derivations, using admissibility lemmas for the rules ($\Diamond Share$) and ($\Diamond Waste$). □

Efficiency

- Linear number of subtyping constraints.
- Algorithmic views are created "on demand".
- $\Rightarrow$ their number is polynomial.
- Subtyping for this subset of algorithmic views can be computed in polynomial time.

Conclusions and further work

- Conclusions
  - Our type-based analysis encompasses:
    1. Objects
    2. Inheritance
    3. Downcast
    4. Imperative update
    5. Aliasing
    6. Circular data
  - There is a prototype implementation in Ocaml (raja.tcs.ifi.lmu.de).
  - Implementation allows:
    - Testing the system with bigger programs
    - Improving the system
- Further work
  - Elimination of the annotations of the variable occurrences.
  - Type inference
  - More examples: Iterators on lists, etc.
  - Making the system more expressive (maybe with type states).