

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

- 1 Introduction to FJEU
- 2 Extending FJEU to RAJA
- 3 Algorithmic RAJA Typing
- 4 Conclusions

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

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

- \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 $P = \textit{annotated}$ FJEU program.
 - ① Set of *views* \mathcal{V} .
 - ② For each class C and view r we have an annotated version C^r .
 - ③ Potential:
 - $\diamond(C^r) : \text{Class} \times \text{View} \rightarrow \mathbb{Q}^+$
 - ④ (Get- and set-) views for attributes:
 - $A^{\text{get}}(C^r, a) : \text{Class} \times \text{View} \times \text{Field} \rightarrow \text{View}$ (*get-view*)
 - $A^{\text{set}}(C^r, a) : \text{Class} \times \text{View} \times \text{Field} \rightarrow \text{View}$ (*set-view*)
 - ⑤ RAJA types for methods:
 - $M(C^r, m) : \text{Class} \times \text{View} \times \text{Method} \rightarrow$
 $\mathcal{P}(\text{Views of Arguments} \rightarrow \text{Effect} \times \text{View of Result})$
 $(r_1, \dots, r_j \xrightarrow{m/m'} r_0)$
 - Effect: pair of numbers m, m' representing potential consumed and released by the method.

Sketch of RAJA System

- RAJA program $P = \textit{annotated}$ FJEU program.
 - ① Set of *views* \mathcal{V} .
 - ② For each class C and view r we have an annotated version C^r .
 - ③ Potential:
 - $\diamond(C^r) : \text{Class} \times \text{View} \rightarrow \mathbb{Q}^+$
 - ④ (Get- and set-) views for attributes:
 - $A^{\text{get}}(C^r, a) : \text{Class} \times \text{View} \times \text{Field} \rightarrow \text{View}$ (*get-view*)
 - $A^{\text{set}}(C^r, a) : \text{Class} \times \text{View} \times \text{Field} \rightarrow \text{View}$ (*set-view*)
 - ⑤ RAJA types for methods:
 - $M(C^r, m) : \text{Class} \times \text{View} \times \text{Method} \rightarrow$
 $\mathcal{P}(\text{Views of Arguments} \rightarrow \text{Effect} \times \text{View of Result})$
 $(r_1, \dots, r_j \xrightarrow{m/m'} r_0)$
 - Effect: pair of numbers m, m' representing potential consumed and released by the method.

Sketch of RAJA System

- RAJA program $P = \textit{annotated}$ 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:
 - $\diamond(C^{\mathbf{r}}) : \text{Class} \times \text{View} \rightarrow \mathbb{Q}^+$
 - ④ (Get- and set-) views for attributes:
 - $A^{\text{get}}(C^{\mathbf{r}}, \mathbf{a}) : \text{Class} \times \text{View} \times \text{Field} \rightarrow \text{View}$ (**get-view**)
 - $A^{\text{set}}(C^{\mathbf{r}}, \mathbf{a}) : \text{Class} \times \text{View} \times \text{Field} \rightarrow \text{View}$ (**set-view**)
 - ⑤ RAJA types for methods:
 - $M(C^{\mathbf{r}}, m) : \text{Class} \times \text{View} \times \text{Method} \rightarrow$
 $\mathcal{P}(\text{Views of Arguments} \rightarrow \text{Effect} \times \text{View of Result})$
 $(\mathbf{r}_1, \dots, \mathbf{r}_j \xrightarrow{m/m'} \mathbf{r}_0)$
 - Effect: pair of numbers m, m' representing potential consumed and released by the method.

Sketch of RAJA System

- RAJA program $P = \textit{annotated}$ 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:
 - $\diamond(C^{\mathbf{r}}) : \text{Class} \times \text{View} \rightarrow \mathbb{Q}^+$
 - ④ (Get- and set-) views for attributes:
 - $A^{\text{get}}(C^{\mathbf{r}}, \mathbf{a}) : \text{Class} \times \text{View} \times \text{Field} \rightarrow \text{View}$ (*get-view*)
 - $A^{\text{set}}(C^{\mathbf{r}}, \mathbf{a}) : \text{Class} \times \text{View} \times \text{Field} \rightarrow \text{View}$ (*set-view*)
 - ⑤ RAJA types for methods:
 - $M(C^{\mathbf{r}}, m) : \text{Class} \times \text{View} \times \text{Method} \rightarrow$
 $\mathcal{P}(\text{Views of Arguments} \rightarrow \text{Effect} \times \text{View of Result})$
 $(\mathbf{r}_1, \dots, \mathbf{r}_j \xrightarrow{m/m'} \mathbf{r}_0)$
 - 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;
  }
}

```

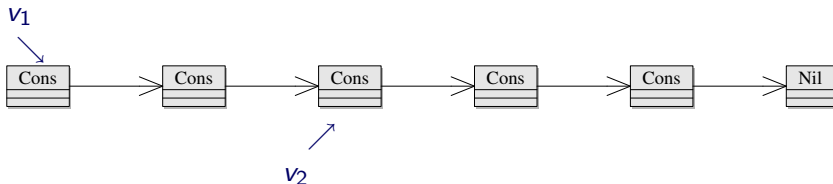
$\diamond(\cdot)$	rich	poor
List	0	0
Nil	0	0
Cons	1	0

	Cons ^{rich}	Cons ^{poor}
Aget(\cdot , next)	rich	poor
Aset(\cdot , next)	rich	poor

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

Potential

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



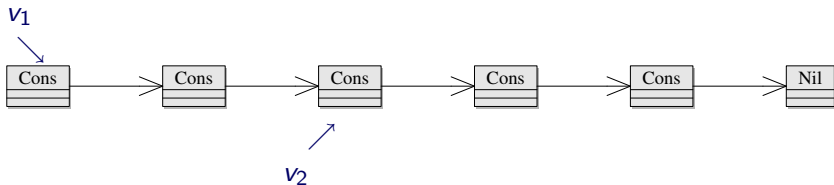
$$\Phi_{\sigma}(v_1 : \mathbf{r}) = \begin{cases} 5 & \mathbf{r} = \text{rich} \\ 0 & \mathbf{r} = \text{poor} \end{cases}$$

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

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

Potential

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



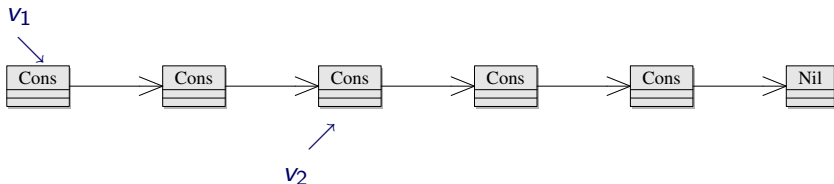
$$\Phi_{\sigma}(v_1 : \mathbf{r}) = \begin{cases} 5 & \mathbf{r} = \text{rich} \\ 0 & \mathbf{r} = \text{poor} \end{cases}$$

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

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

Potential

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



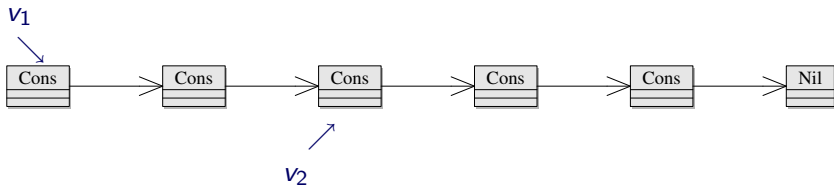
$$\Phi_{\sigma}(v_1 : \mathbf{r}) = \begin{cases} 5 & \mathbf{r} = \text{rich} \\ 0 & \mathbf{r} = \text{poor} \end{cases}$$

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

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

Potential

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



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

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

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

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 $\Gamma \frac{n}{n'} e : C^r$ so that:
 - if $\Gamma \frac{n}{n'} e : C^r$ 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

$$\frac{}{\emptyset \vdash \text{new } C : C} (\diamond \text{New})$$

$$\frac{C.a = D}{x : C \vdash x.a : D} (\diamond \text{Access})$$

$$\frac{}{x : C \vdash \text{free}(x) : E} (\diamond \text{Free})$$

$$\frac{C.a = D}{x : C, y : D \vdash x.a \leftarrow y : C} (\diamond \text{Update})$$

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

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

$$\frac{x \in \Gamma \quad \Gamma \vdash e_1 : C \quad \Gamma \vdash e_2 : C}{\text{if } x \text{ instance of } E \text{ then } e_1 \text{ else } e_2 : C} (\diamond \text{Conditional})$$

RAJA typing rules

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

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

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

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

$$\frac{x \in \Gamma \quad \Gamma \vdash \frac{n}{n'} e_1 : C^r \quad \Gamma \vdash \frac{n}{n'} e_2 : C^r}{\Gamma \vdash \frac{n}{n'} \text{ if } x \text{ instance of } E \text{ then } e_1 \text{ else } e_2 : C^r} (\diamond\text{Conditional})$$

Subtyping and weakening

- There is a weakening rule:

$$\frac{n \geq u \quad n - n' \geq u - u' \quad \Theta \vdash_{u'}^u e : D^s \quad \Gamma <: \Theta \quad D^s <: C^r}{\Gamma \vdash_{n'}^n e : C^r} \quad (\diamond \text{Waste})$$

- 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 $E <: F$:

$$\diamond(E^r) \geq \diamond(F^s) \quad (2.1)$$

$$\forall a \in A(F) . F.a^{A^{set}(E;a)} <: F.a^{A^{set}(F;a)} \quad (2.2)$$

$$\forall a \in A(F) . F.a^{A^{set}(F;a)} <: F.a^{A^{set}(E;a)} \quad (2.3)$$

$$\dots \quad (2.4)$$

Subtyping and weakening

- There is a weakening rule:

$$\frac{n \geq u \quad n - n' \geq u - u' \quad \Theta \Vdash_{u'} e : D^s \quad \Gamma <: \Theta \quad D^s <: C^r}{\Gamma \Vdash_{n'} e : C^r} \quad (\diamond \text{Waste})$$

- 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 $E <: F$:

$$\diamond(E^r) \geq \diamond(F^s) \quad (2.1)$$

$$\forall a \in A(F) . F.a^{A^{get}(E;a)} <: F.a^{A^{get}(F;a)} \quad (2.2)$$

$$\forall a \in A(F) . F.a^{A^{set}(F;a)} <: F.a^{A^{set}(E;a)} \quad (2.3)$$

$$\dots \quad (2.4)$$

Sharing relation

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

$$\frac{\Gamma, y_1 : D^{q_1}, \dots, y_n : D^{q_n} \vdash_{n'}^n e : C^r \quad \forall (D^s \mid D^{q_1}, \dots, D^{q_n})}{\Gamma, x : D^s \vdash_{n'}^n e[x/y_1, \dots, x/y_n] : C^r} (\diamond\text{Share})$$

Definition (Sharing Relation)

If $\forall (C^r \mid D^{s_1}, \dots, D^{s_n})$ then for all $E <: C, F <: D$ with $E <: F$:

$$\diamond(E^r) \geq \sum_i \diamond(F^{s_i}) \quad (2.5)$$

$$\forall i. E^r <: F^{s_i} \dots \quad (2.6)$$

- $C^r <: D^s \iff \forall (C^r \mid D^s)$

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

- is not allowed because $\forall (\text{List}^{\text{rich}} \mid \text{List}^{\text{rich}}, \text{List}^{\text{rich}})$ does not hold.

Sharing relation

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

$$\frac{\Gamma, y_1 : D^{q_1}, \dots, y_n : D^{q_n} \vdash_{n'}^n e : C^r \quad \forall (D^s \mid D^{q_1}, \dots, D^{q_n})}{\Gamma, x : D^s \vdash_{n'}^n e[x/y_1, \dots, x/y_n] : C^r} (\diamond\text{Share})$$

Definition (Sharing Relation)

If $\forall (C^r \mid D^{s_1}, \dots, D^{s_n})$ then for all $E <: C$, $F <: D$ with $E <: F$:

$$\diamond(E^r) \geq \sum_i \diamond(F^{s_i}) \quad (2.5)$$

$$\forall i. E^r <: F^{s_i} \dots \quad (2.6)$$

- $C^r <: D^s \iff \forall (C^r \mid D^s)$

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

- is not allowed because $\forall (\text{List}^{\text{rich}} \mid \text{List}^{\text{rich}}, \text{List}^{\text{rich}})$ does not hold.

Sharing relation

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

$$\frac{\Gamma, y_1 : D^{q_1}, \dots, y_n : D^{q_n} \vdash_{n'}^n e : C^r \quad \forall (D^s | D^{q_1}, \dots, D^{q_n})}{\Gamma, x : D^s \vdash_{n'}^n e[x/y_1, \dots, x/y_n] : C^r} (\diamond\text{Share})$$

Definition (Sharing Relation)

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

$$\diamond(E^r) \geq \sum_i \diamond(F^{s_i}) \quad (2.5)$$

$$\forall i. E^r <: F^{s_i} \dots \quad (2.6)$$

- $C^r <: D^s \iff \forall (C^r | D^s)$

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

- is not allowed because $\forall (\text{List}^{\text{rich}} | \text{List}^{\text{rich}}, \text{List}^{\text{rich}})$ 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.
- $\forall (C^r | C^{s_1}, C^{s_2}) \iff C^r <: C^{s_1+s_2}$

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

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.
- $\forall (C^r | C^{s_1}, C^{s_2}) \iff C^r <: C^{s_1+s_2}$

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

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: $\Delta^{\Psi} \frac{n}{n'} e^{\circ} \Leftarrow C^{\gamma}$
- 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.

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

$$((\) \xrightarrow{0/0} \text{List}^{\text{poor}}) \in M(\text{Cons}^{\text{rich}}, m)$$

$$\text{this: Cons}^{\text{rich}} \frac{0}{0} \text{this}^{\text{rich}}.\text{copy}() \Leftarrow \text{List}^{\text{poor}}$$

$$\text{Cons} <: \text{List}$$

$$\text{this: Cons}^{\text{poor}}, \text{res: List}^{\emptyset} \frac{0}{0} \text{this}^{\text{poor}} \Leftarrow \text{List}^{\text{poor}}$$

$$\text{this: Cons}^{\text{rich}+\text{poor}} \frac{0}{0} \text{let List res} = \text{this}^{\text{rich}}.\text{copy}() \text{ in } \text{this}^{\text{poor}} \Leftarrow \text{List}^{\text{poor}}$$

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: $\Delta^\Psi \frac{n}{n'} e^\circ \Leftarrow C^\gamma$
- 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.

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

$$\frac{\frac{((\rightarrow^{0/0} \text{List}^{\text{poor}}) \in M(\text{Cons}^{\text{rich}}, m))}{\text{this: } \text{Cons}^{\text{rich}} \frac{0}{0} \text{this}^{\text{rich}}.\text{copy}() \Leftarrow \text{List}^{\text{poor}}} \quad \frac{\text{Cons} <: \text{List}}{\text{this: } \text{Cons}^{\text{poor}}, \text{res: } \text{List}^\emptyset \frac{0}{0} \text{this}^{\text{poor}} \Leftarrow \text{List}^{\text{poor}}}}{\text{this: } \text{Cons}^{\text{rich}+\text{poor}} \frac{0}{0} \text{let List res} = \text{this}^{\text{rich}}.\text{copy}() \text{ in } \text{this}^{\text{poor}} \Leftarrow \text{List}^{\text{poor}}}$$

Algorithmic RAJA typing rules

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

$$\frac{\Delta^{\Psi'} \vdash_{n'}^n e_1^o \Leftarrow D\gamma_1 \quad \Delta^{\Psi''}, x: D\gamma_1 \vdash_{m'}^m e_2^o \Leftarrow C\gamma_2}{\Delta^{\Psi'+\Psi''} \vdash_{\max(m', m'+n'-m)}^{\max(n, n+m-n')} \text{let } Dx = e_1^o \text{ in } e_2^o \Leftarrow C\gamma_2} \quad (\vdash \text{Let})$$

$$\frac{x \in \Delta \quad \Delta^{\Psi'} \vdash_{n'}^n e_1^o \Leftarrow C\gamma \quad \Delta^{\Psi''} \vdash_{m'}^m e_2^o \Leftarrow C\gamma \quad u = \max(m, n)}{\Delta^{\Psi' \wedge \Psi''} \vdash_{\min(n'+u-n, m'+u-m)}^u \text{if } x \text{ instance of } E \text{ then } e_1^o \text{ else } e_2^o \Leftarrow C\gamma} \quad (\vdash \text{Cond.})$$

$$\frac{p/p' = \min\{(E_1^{q_1}, \dots, E_j^{q_j} \xrightarrow{p/p'} E_0^{q_0}) \in M(G^r, m) \mid \forall i. F_i^{t_i} <: E_i^{q_i}, E_0^{q_0} <: C\gamma\}}{\Delta^{\Psi' \wedge \Psi''}, x: G^r, + y_1: F_1^{t_1}, + \dots, + y_j: F_j^{t_j} \vdash_{p'}^p x^r.m(y_1^{t_1}, \dots, y_j^{t_j}) \Leftarrow C\gamma}$$

Algorithmic RAJA typing rules

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

$$\frac{\Delta^{\Psi'} \vdash_{n'}^n e_1^o \Leftarrow D\gamma_1 \quad \Delta^{\Psi''}, x: D\gamma_1 \vdash_{m'}^m e_2^o \Leftarrow C\gamma_2}{\Delta^{\Psi'+\Psi''} \vdash_{\max(m', m'+n'-m)}^{\max(n, n+m-n')} \text{let } Dx = e_1^o \text{ in } e_2^o \Leftarrow C\gamma_2} \quad (\vdash \text{Let})$$

$$\frac{x \in \Delta \quad \Delta^{\Psi'} \vdash_{n'}^n e_1^o \Leftarrow C\gamma \quad \Delta^{\Psi''} \vdash_{m'}^m e_2^o \Leftarrow C\gamma \quad u = \max(m, n)}{\Delta^{\Psi' \wedge \Psi''} \vdash_{\min(n'+u-n, m'+u-m)}^u \text{if } x \text{ instance of } E \text{ then } e_1^o \text{ else } e_2^o \Leftarrow C\gamma} \quad (\vdash \text{Cond.})$$

$$\frac{p/p' = \min\{(E_1^{q_1}, \dots, E_j^{q_j} \xrightarrow{p/p'} E_0^{q_0}) \in M(G^r, m) \mid \forall i. F_i^{t_i} <: E_i^{q_i}, E_0^{q_0} <: C\gamma\}}{\Delta^{\Psi'+\Psi''}, x: G^r, + y_1: F_1^{t_1}, + \dots, + y_j: F_j^{t_j} \vdash_{p'}^p x^r.m(y_1^{t_1}, \dots, y_j^{t_j}) \Leftarrow C\gamma}$$

Algorithmic RAJA typing rules

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

$$\frac{\Delta^{\Psi'} \vdash_{n'}^n e_1^o \Leftarrow D\gamma_1 \quad \Delta^{\Psi''}, x: D\gamma_1 \vdash_{m'}^m e_2^o \Leftarrow C\gamma_2}{\Delta^{\Psi'+\Psi''} \vdash_{\max(m', m'+n'-m)}^{\max(n, n+m-n')} \text{let } Dx = e_1^o \text{ in } e_2^o \Leftarrow C\gamma_2} \quad (\vdash \text{Let})$$

$$\frac{x \in \Delta \quad \Delta^{\Psi'} \vdash_{n'}^n e_1^o \Leftarrow C\gamma \quad \Delta^{\Psi''} \vdash_{m'}^m e_2^o \Leftarrow C\gamma \quad u = \max(m, n)}{\Delta^{\Psi' \wedge \Psi''} \vdash_{\min(n'+u-n, m'+u-m)}^u \text{if } x \text{ instance of } E \text{ then } e_1^o \text{ else } e_2^o \Leftarrow C\gamma} \quad (\vdash \text{Cond.})$$

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

Algorithmic RAJA typing rules

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

$$\frac{\Delta^{\Psi'} \vdash_{n'}^n e_1^o \Leftarrow D\gamma_1 \quad \Delta^{\Psi''}, x: D\gamma_1 \vdash_{m'}^m e_2^o \Leftarrow C\gamma_2}{\Delta^{\Psi'+\Psi''} \vdash_{\max(m', m'+n'-m)}^{\max(n, n+m-n')} \text{let } Dx = e_1^o \text{ in } e_2^o \Leftarrow C\gamma_2} \quad (\vdash \text{Let})$$

$$\frac{x \in \Delta \quad \Delta^{\Psi'} \vdash_{n'}^n e_1^o \Leftarrow C\gamma \quad \Delta^{\Psi''} \vdash_{m'}^m e_2^o \Leftarrow C\gamma \quad u = \max(m, n)}{\Delta^{\Psi' \wedge \Psi''} \vdash_{\min(n'+u-n, m'+u-m)}^u \text{if } x \text{ instance of } E \text{ then } e_1^o \text{ else } e_2^o \Leftarrow C\gamma} \quad (\vdash \text{Cond.})$$

$$\frac{p/p' = \min\{(E_1^{q_1}, \dots, E_j^{q_j} \xrightarrow{p/p'} E_0^{q_0}) \in M(G^r, m) \mid \forall i. F_i^{t_i} <: E_i^{q_i}, E_0^{q_0} <: C\gamma\}}{\Delta^{\Psi' \wedge \Psi''}, x: G^r, + y_1: F_1^{t_1}, + \dots, + y_j: F_j^{t_j} \vdash_{p'}^p x^r.m(y_1^{t_1}, \dots, y_j^{t_j}) \Leftarrow C\gamma} \quad (\vdash \text{Inv.})$$

Soundness proof

Lemma (Soundness of algorithmic RAJA typing)

If $\Delta^{\Psi} \vdash_{n'}^n e \Leftarrow C^{\gamma}$ 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 $\Gamma \vdash_{n/n'}^r e : C^r$ then there is an annotated version e° of the expression e with $|\Gamma|^\Psi \vdash_{u/u'}^r e^\circ \Leftarrow C^r$ 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).