Blame Assignment for Higher-Order Contracts with Intersection and Union

Roman Matthias Keil, Peter Thiemann
University of Freiburg, Germany

September 2, 2015, The 20th ACM SIGPLAN International Conference on Functional Programming, ICFP 2015
Vancouver, British Columbia, Canada

UNI FREIBURG

Notizen

Higher-Order Contracts

UNI FREIBURG

- $Even = flat(\lambda x.x\%2 = 0)$
- $Odd = flat(\lambda x.x\%2 = 1)$

Assertion $(Even \rightarrow Even)$

Let $add2Even = ((\lambda x.x + 2)$ @ $(Even \rightarrow Even))$ Let $add2Even = ((\lambda x.x + 2)$ @ $(Even \rightarrow Even))$ Let $add2Even = ((\lambda x.x + 2)$ @ $(Even \rightarrow Even))$

- $(add2Even\ 2) \longrightarrow^* 4$ ✓
- $(add2Even\ 1) \longrightarrow^*$ ✗ blame context ☐ 1

Assertion $(Odd \rightarrow Odd)$

Let $add2Odd = ((\lambda x.x + 2)$ @ $(Odd \rightarrow Odd))$ Let $add2Odd = ((\lambda x.x + 2)$ @ $(Odd \rightarrow Odd))$ Let $add2Odd = ((\lambda x.x + 2)$ @ $(Odd \rightarrow Odd))$

- $(add2Odd\ 1) \longrightarrow^* 3$ ✓
- $(add2Odd\ 2) \longrightarrow^*$ ✗ blame context ☐ 2

Notizen

Combination of Contracts

UNI FREIBURG

Observation

- $\lambda x.x + 2$ works for *even* and *odd* arguments
- $\lambda x.x + 2$ fulfills $Odd \rightarrow Odd$ **and** $Even \rightarrow Even$
- How can we express that with a single contract?

Intersection Contract!

Notizen

Inspiration

Intersection Type

- $V : S \cap T$
- Models overloading
- Models multiple inheritances

Union Type

- $V : S \cup T$
- Dual of intersection type
- Domain of overloaded functions

Notizen

This Work

- Extend higher-order contracts with intersection and union
- Specification based on the type theoretic construction

Assertion $(Even \to Even) \cap (Odd \to Odd)$

Let $add2 = ((\lambda x.x + 2) \mathbin{@} (Even \to Even) \cap (Odd \to Odd))$

- $(add2\ 2) \longrightarrow^* 4$ ✓
- $(add2\ 1) \longrightarrow^* 3$ ✓

No blame because of the intersection contract!

Notizen

Flat Contract

- $Even = flat(\lambda x.x\%2 = 0)$
- $Odd = flat(\lambda x.x\%2 = 1)$
- $Pos = flat(\lambda x.x > 0)$

Examples

- $Pos \cap Even$

Flat Contract

- $flat(\lambda x.P) \cap flat(\lambda x.Q) \equiv flat(\lambda x.P \wedge Q)$

Notizen

Intersection Contract

Assertion

Let $add1 = ((\lambda x.x + 1)$ @ $(Even \rightarrow Even) \cap (Pos \rightarrow Pos))$ Let $add1 = ((\lambda x.x + 1)$ @ $(Even \rightarrow Even) \cap (Pos \rightarrow Pos))$ Let $add1 = ((\lambda x.x + 1)$ @ $(Even \rightarrow Even) \cap (Pos \rightarrow Pos))$ Let $add1 = ((\lambda x.x + 1)$ @ $(Even \rightarrow Even) \cap (Pos \rightarrow Pos))$

- $(add1\ 3) \longrightarrow^* 4$ ✓
- $(add1\ {-1}) \longrightarrow^*$ ✗ blame context $\square\ {-1}$
- $(add1\ 2) \longrightarrow^*$ ✗ blame subject $(\lambda x.x + 1)$

Definition

- Context gets *blamed* for $\mathcal{C} \cap \mathcal{D}$ iff:
  (Context gets *blamed* for $\mathcal{C}$) $\wedge$ (Context gets *blamed* for $\mathcal{D}$)
- Subject M gets *blamed* for $\mathcal{C} \cap \mathcal{D}$ iff:
  (M gets *blamed* for $\mathcal{C}$) $\vee$ (M gets *blamed* for $\mathcal{D}$)

Notizen

Contract Assertion

Example

- $((\lambda x.x + 1)$ @ $(EvenEven \rightarrow Even) \cap (PosPos \rightarrow PosPos))\ 3 \longrightarrow^* 4$ ✓

- A failing contract must not signal a violation immediately
- Violation depends on combinations of failures in different sub-contracts
- Contract assertion must connect each contract with the enclosing operations

Notizen

Operational Semantics

Reduction Relation

$$\varsigma, M \longrightarrow \varsigma', N$$

- M, N expressions
- ς list of constraints
- One constraint for each contract operator $\rightarrow$, $\cap$, $\cup$
- One constraint for each flat contract
- Blame calculation from a list of constraints

Notizen

Flat Contract

UNI FREIBURG

Evaluation Rule

$$\frac{\textsc{Flat}\quad M\ V \longrightarrow^* W \qquad \varsigma' = \flat \blacktriangleleft (W) : \varsigma}{\varsigma, E[V \mathbin{@^\flat} \mathit{flat}(M)] \longrightarrow \varsigma', E[V]}$$

Notizen

Interpretation

UNI FREIBURG

Interpretation of a constraint list

$$\mu \in (\llparenthesis \flat \rrparenthesis \times \{\mathit{subject}, \mathit{context}\}) \to \mathbb{B}$$

- An interpretation μ is a mapping from blame label $\flat$ to records of elements of $\mathbb{B} = \{\mathsf{t}, \mathsf{f}\}$, order $\mathsf{t} \sqsubset \mathsf{f}$
- Ordering reflects gathering of information with each execution step
- Each blame label $\flat$ is associated with two truth values, $\flat.\mathit{subject}$ and $\flat.\mathit{context}$

Notizen

Flat Contract (cont'd)

UNI FREIBURG

Evaluation Rule

$$\frac{\textsc{Flat}\quad M\ V \longrightarrow^* W \qquad \varsigma' = \flat \blacktriangleleft (W) : \varsigma}{\varsigma, E[V \mathbin{@^\flat} \mathit{flat}(M)] \longrightarrow \varsigma', E[V]}$$

Constraint Satisfaction

$$\frac{\textsc{C-Flat}\quad \mu(\flat.\mathit{subject}) \sqsupseteq W \qquad \mu(\flat.\mathit{context}) \sqsupseteq \mathsf{t}}{\mu \models \flat \blacktriangleleft W}$$

Notizen

Blame Calculation

UNI FREIBURG

Definition

ς is a *blame state* if there exists a top-level blame label such that

$$\mu(\flat.\mathit{subject}) \sqsupseteq \mathsf{f} \;\vee\; \mu(\flat.\mathit{context}) \sqsupseteq \mathsf{f}$$

- Evaluation stops if a blame state is reached.

Notizen

Function Contract

UNI FREIBURG

Evaluation Rule

Function

$$\frac{\flat_1, \flat_2 \notin \varsigma \qquad \varsigma' = \flat \blacktriangleleft (\flat_1 \rightarrow \flat_2) : \varsigma}{\varsigma, E[(V \mathbin{@^{\flat}} (\mathcal{C} \rightarrow \mathcal{D}))\; W] \longrightarrow \varsigma', E[(V\; (W \mathbin{@^{\flat_1}} \mathcal{C})) \mathbin{@^{\flat_2}} \mathcal{D}]}$$

Constraint Satisfaction

C-Function

$$\frac{\begin{array}{c}\mu(\flat.\mathit{subject}) \sqsupseteq \mu(\flat_1.\mathit{context} \wedge (\flat_1.\mathit{subject} \Rightarrow \flat_2.\mathit{subject})) \\ \mu(\flat.\mathit{context}) \sqsupseteq \mu(\flat_1.\mathit{subject} \wedge \flat_2.\mathit{context})\end{array}}{\mu \models \flat \blacktriangleleft \flat_1 \rightarrow \flat_2}$$

Notizen

Intersection Contract

UNI FREIBURG

Evaluation Rule

Intersection

$$\frac{\flat_1, \flat_2 \notin \varsigma \qquad \varsigma' = \flat \blacktriangleleft (\flat_1 \cap \flat_2) : \varsigma}{\varsigma, E[(V \mathbin{@^{\flat}} (Q \cap R))\; W] \longrightarrow \varsigma', E[((V \mathbin{@^{\flat_1}} Q) \mathbin{@^{\flat_2}} R)\; W]}$$

Constraint Satisfaction

C-Intersection

$$\frac{\begin{array}{c}\mu(\flat.\mathit{subject}) \sqsupseteq \mu(\flat_1.\mathit{subject} \wedge \flat_2.\mathit{subject}) \\ \mu(\flat.\mathit{context}) \sqsupseteq \mu(\flat_1.\mathit{context} \vee \flat_2.\mathit{context})\end{array}}{\mu \models \flat \blacktriangleleft \flat_1 \cap \flat_2}$$

Notizen

Union Contract

- Dual of intersection contract
- Exchange $\wedge$ and $\vee$ in the blame calculation
- *Delayed* evaluation changes to an *immediate* evaluation

Notizen

In the Paper

Technical Results

- Contract Rewriting
- Deterministic and nondeterministic specification of contract monitoring
- Denotational specification of the semantics of contracts
- Theorems for contract and blame soundness

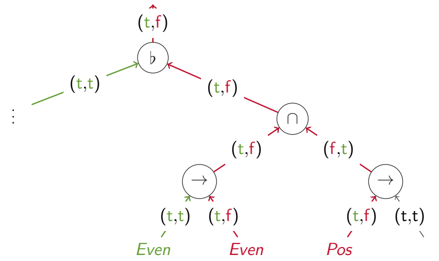
Notizen

Conclusion

- Intersection and union contracts provide dynamic guarantees equivalent to their type-theoretic counterparts
- Constraint-based blame calculation enables higher-order contracts with unrestricted intersection and union
- Formal basis of *TreatJS*, a language embedded, higher-order contract system implemented for JavaScript

Notizen

Constraint Graph



Reduction

- ς,
  $((\lambda x.x+1)\ @^{\flat}\ ((Even \rightarrow Even) \cap (Pos \rightarrow Pos)))\ 0$
- $\longrightarrow \flat \blacktriangleleft (\flat_1 \cap \flat_2) : \cdots,$
  $(((\lambda x.x+1)\ @^{\flat_1}\ (Even \rightarrow Even))\ @^{\flat_2}\ (Pos \rightarrow Pos))\ 0$

- $\longrightarrow \flat_2 \blacktriangleleft (\flat_3 \rightarrow \flat_4) : \cdots,$
  $(((\lambda x.x+1)\ @^{\flat_1}\ (Even \rightarrow Even))\ (0\ @^{\flat_3}\ Pos))\ @^{\flat_4}\ Pos$
- $\longrightarrow \flat_3 \blacktriangleleft (false) : \cdots,$
  $(((\lambda x.x+1)\ @^{\flat_1}\ (Even \rightarrow Even))\ 0)\ @^{\flat_4}\ Pos$
- $\longrightarrow \flat_1 \blacktriangleleft (\flat_5 \rightarrow \flat_6) : \cdots,$
  $(((\lambda x.x+1)\ (0\ @^{\flat_5}\ Even))\ @^{\flat_6}\ Even)\ @^{\flat_4}\ Pos$
- $\longrightarrow \flat_5 \blacktriangleleft (true) : \cdots,$
  $(((\lambda x.x+1)\ 0)\ @^{\flat_6}\ Even)\ @^{\flat_4}\ Pos$

Notizen

Intersection and Union Types

Intersection Type

- $\lambda x.x+2 : Even \rightarrow Even$
- $\lambda x.x+2 : Odd \rightarrow Odd$
- $\lambda x.x+2 : Even \rightarrow Even \cap Odd \rightarrow Odd$

Union Type

- $\lambda x.x-2 : Even \rightarrow Even$
- $\lambda x.x-2 : Even \rightarrow Even \cup Pos \rightarrow Pos$

Notizen

Flat Contract [Findler,Felleisen'02]

- $Pos = flat(\lambda x.x > 0)$
- $Even = flat(\lambda x.x\%2 = 0)$

Assertion

- $1@Pos \longrightarrow 1$ ✓
- $0@Pos \longrightarrow$ ✗ blame subject 0

Definition

- Subject V gets blamed for *Flat Contract* $flat(M)$ iff:
  $(M\ V) \longrightarrow^* false$

Notizen

Higher-Order Contract [Findler,Felleisen'02]

UNI FREIBURG

- $Even \rightarrow Even$

Assertion

- $((\lambda x.x+1)@Even \rightarrow Even)\ 1 \longrightarrow^{*}$ ✗ blame context $\square\ 1$
- $((\lambda x.x+1)@Even \rightarrow Even)\ 2 \longrightarrow^{*}$ ✗ blame subject

Definition

- Context gets *blamed* for $\mathcal{C} \rightarrow \mathcal{D}$ iff:
  Argument x gets *blamed* for $\mathcal{C}$ (as subject)
- Subject M gets *blamed* for $\mathcal{C} \rightarrow \mathcal{D}$ at $\square\ V$ iff:
  $\neg$ (Context gets *blamed* $\mathcal{C}$) $\wedge$ ($M\ V$ gets *blamed* $\mathcal{D}$)

Notizen

Flat Contract

UNI FREIBURG

Examples

- $Odd \cup Even$

Flat Contract

- $flat(\lambda x.P) \cup flat(\lambda x.Q) \equiv flat(\lambda x.P \vee Q)$

Notizen

Union Contract

UNI FREIBURG

Assertion

Let $mod3 = ((\lambda x.x\%3)\ @\ (Even \rightarrow Even) \cup (Pos \rightarrow Pos))$ Let $mod3 = ((\lambda x.x\%3)\ @\ (Even \rightarrow Even) \cup (Pos \rightarrow \underline{Pos}))$ Let $mod3 = ((\lambda x.x\%3)\ @\ (Even \rightarrow Even) \cup (Pos \rightarrow \underline{Pos}))$ Let $mod3 = ((\lambda x.x\%3)\ @\ (Even \rightarrow Even) \cup (Pos \rightarrow \underline{Pos}))$

- $(mod3\ 4) \longrightarrow^{*} 1$ ✓
- $(mod3\ 1) \longrightarrow^{*}$ ✗ blame context $\square\ 1$
- $(mod3\ 6) \longrightarrow^{*}$ ✗ blame subject $(\lambda x.x\%3)$

Definition

- Context gets *blamed* for $\mathcal{C} \cup \mathcal{D}$ iff:
  (Context gets *blamed* for $\mathcal{C}$) $\vee$ (Context gets *blamed* for $\mathcal{D}$)
- Subject M gets *blamed* for $\mathcal{C} \cup \mathcal{D}$ iff:
  (M gets *blamed* for $\mathcal{C}$) $\wedge$ (M gets *blamed* for $\mathcal{D}$)

Notizen

Contract Assertion

Evaluation Rule

$$\textsc{Assert}\quad \frac{\flat \notin \varsigma \qquad \varsigma' = \flat\blacktriangleleft(\flat) : \varsigma}{\varsigma, E[V \mathbin{@^{\flat}} \mathcal{C}] \longrightarrow^{*} \varsigma', E[V \mathbin{@^{\flat}} \mathcal{C}]}$$

Constraint Satisfaction

$$\textsc{C-Assert}\quad \frac{\mu(\flat.\mathit{subject}) \sqsupseteq \mu(\flat_1.\mathit{subject}) \qquad \mu(\flat.\mathit{context}) \sqsupseteq \mu(\flat_1.\mathit{context})}{\mu \models \flat\blacktriangleleft(\flat_1)}$$

Notizen

Constraint List

Constraint Satisfaction

$$\textsc{CS-Empty}\quad \mu \models \cdot \qquad\qquad \textsc{CS-Cons}\quad \frac{\mu \models \kappa \qquad \mu \models \varsigma}{\mu \models \kappa : \varsigma}$$

Notizen

Union Contract

Evaluation Rule

$$\textsc{Union}\quad \frac{\flat_1, \flat_2 \notin \varsigma \qquad \varsigma' = \flat\blacktriangleleft(\flat_1 \cup \flat_2) : \varsigma}{\varsigma, E[V \mathbin{@^{\flat}} (\mathcal{C} \cup \mathcal{D})] \longrightarrow \varsigma', E[(V \mathbin{@^{\flat_1}} \mathcal{C}) \mathbin{@^{\flat_2}} \mathcal{D}]}$$

Constraint Satisfaction

$$\textsc{C-Union}\quad \frac{\begin{array}{c}\mu(\flat.\mathit{subject}) \sqsupseteq \mu(\flat_1.\mathit{subject} \vee \flat_2.\mathit{subject}) \\ \mu(\flat.\mathit{context}) \sqsupseteq \mu(\flat_1.\mathit{context} \wedge \flat_2.\mathit{context})\end{array}}{\mu \models \flat \blacktriangleleft \flat_1 \cup \flat_2}$$

Notizen

Blame Calculation

Definition

ς is a *blame state* if there exists a top-level blame identifier such that

$$\mu(\flat.\mathit{subject}) \sqsupseteq \mathsf{f} \ \vee\ \mu(\flat.\mathit{context}) \sqsupseteq \mathsf{f}$$

$$\frac{\varsigma, M \longrightarrow^{*} \varsigma', N \qquad \varsigma \text{ is not a blame state}}{\varsigma, M \longmapsto \varsigma', N} \qquad \frac{\varsigma \text{ is blame state for } \flat}{\varsigma, M \longmapsto \varsigma, \mathit{blame}^{\flat}}$$

Notizen

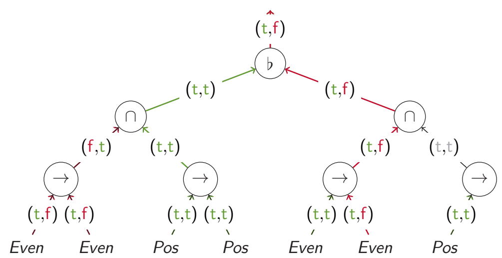
Example Reduction

Reduction

- $\cdot,$
  $((\lambda x.x + 1)\ @^{\flat}\ ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos})))\ 0$
- $\longrightarrow \flat \blacktriangleleft (\flat_0) : \cdot,$
  $((\lambda x.x + 1)\ @^{\flat_0}\ ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos})))\ 0$
- $\longrightarrow \flat_0 \blacktriangleleft (\flat_1 \cap \flat_2) : \cdots,$
  $(((\lambda x.x + 1)\ @^{\flat_1}\ (\mathit{Even} \to \mathit{Even}))\ @^{\flat_2}\ (\mathit{Pos} \to \mathit{Pos}))\ 0$
- $\longrightarrow \flat_2 \blacktriangleleft (\flat_3 \to \flat_4) : \cdots,$
  $(((\lambda x.x + 1)\ @^{\flat_1}\ (\mathit{Even} \to \mathit{Even}))\ (0\ @^{\flat_3}\ \mathit{Pos}))\ @^{\flat_4}\ \mathit{Pos}$
- $\longrightarrow \flat_3 \blacktriangleleft (\mathit{false}) : \cdots,$
  $(((\lambda x.x + 1)\ @^{\flat_1}\ (\mathit{Even} \to \mathit{Even}))\ 0)\ @^{\flat_4}\ \mathit{Pos}$
- $\longrightarrow \flat_1 \blacktriangleleft (\flat_5 \to \flat_6) : \cdots,$
  $(((\lambda x.x + 1)\ (0\ @^{\flat_5}\ \mathit{Even}))\ @^{\flat_6}\ \mathit{Even})\ @^{\flat_4}\ \mathit{Pos}$
- $\longrightarrow \flat_5 \blacktriangleleft (\mathit{true}) : \cdots,$
  $(((\lambda x.x + 1)\ 0)\ @\ \mathit{Even})\ @\ \mathit{Pos}$
- $\longrightarrow \cdots,$
  $(1\ @^{\flat_6}\ \mathit{Even})\ @^{\flat_4}\ \mathit{Pos}$
- $\longrightarrow \flat_6 \blacktriangleleft (\mathit{false}) : \cdots,$
  $\mathit{blame}^{\flat}$

Notizen

Constraint Graph



Example

- $((\lambda x.x + 1)\ @^{\flat}\ ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos})))\ 1 \longrightarrow^{*} 2$ ✓
- $((\lambda x.x + 1)\ @^{\flat}\ ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos})))\ 2 \longrightarrow^{*}$ ✗

Notizen

Technical Results

Definition (Contract Satisfaction)

The semantics of a contract $\mathcal{C}$ defines

1. a set $[\![\mathcal{C}]\!]^+$ of *closed terms* (subjects) that *satisfy* $\mathcal{C}$
2. a set $[\![\mathcal{C}]\!]^-$ of *closed contexts* that *respect* $\mathcal{C}$

The definition is mutually inductive on the structure of $\mathcal{C}$.

Notizen

Technical Results (cont'd)

Theorem (Contract soundness for expressions)

For all $M, \mathcal{C}, \flat.\ M @^\flat \mathcal{C} \in [\![\mathcal{C}]\!]^+$

Theorem (Contract soundness for contexts)

For all $\mathcal{L}, \mathcal{C}, \flat.\ \mathcal{L}[\square @^\flat \mathcal{C}] \in [\![\mathcal{C}]\!]^-$

Notizen

Technical Results (cont'd)

Theorem (Subject blame soundness)

Suppose that $M \in [\![\mathcal{C}]\!]^+$.
If $\varsigma, E[M @^\flat \mathcal{C}] \longmapsto^* \varsigma', N$ *then* $[\![\varsigma']\!](\flat, \mathit{subject}) \sqsubseteq \mathsf{t}$.

Theorem (Context blame soundness)

Suppose that $\mathcal{L} \in [\![\mathcal{C}]\!]^-$.
If $\varsigma, \mathcal{L}[M @^\flat \mathcal{C}] \longmapsto^* \varsigma', N$, *then* $[\![\varsigma']\!](\flat, \mathit{context}) \sqsubseteq \mathsf{t}$.

Notizen