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

Higher-Order Contracts

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

Assertion ($Even \rightarrow Even$)

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

Higher-Order Contracts

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

Assertion ($Even \rightarrow Even$)

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

- $(add2Even\ 2) \longrightarrow^{*} 4$ ✓

Higher-Order Contracts

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

Assertion $(Even \rightarrow Even)$

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

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

Higher-Order Contracts

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

Assertion ($Even \rightarrow Even$)

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

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

Assertion ($Odd \rightarrow Odd$)

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

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

Assertion $(\mathit{Even} \rightarrow \mathit{Even})$

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

- $(\mathit{add2Even}\ 2) \longrightarrow^* 4$ ✓
- $(\mathit{add2Even}\ 1) \longrightarrow^*$ ✗ blame context $\square$ 1

Assertion $(\mathit{Odd} \rightarrow \mathit{Odd})$

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

- $(\mathit{add2Odd}\ 1) \longrightarrow^* 3$ ✓

Higher-Order Contracts

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

Assertion ($Even \rightarrow Even$)

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

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

Assertion ($Odd \rightarrow Odd$)

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

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

Combination of Contracts

Observation

- $\lambda x.x + 2$ works for *even* and *odd* arguments

Combination of Contracts

Observation

- $\lambda x.x + 2$ works for *even* and *odd* arguments
- $\lambda x.x + 2$ fulfills $\mathit{Odd} \rightarrow \mathit{Odd}$ **and** $\mathit{Even} \rightarrow \mathit{Even}$

Combination of Contracts

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?

Observation

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

Intersection Contract!

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

This Work

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

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

Let $add2 = ((\lambda x.x + 2)\ @\ (Even \rightarrow Even) \cap (Odd \rightarrow Odd))$

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

No blame because of the intersection contract!

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

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

Intersection Contract

Assertion

Let $add1 = ((\lambda x.x + 1) \mathbin{@} (Even \to Even) \cap (Pos \to Pos))$

Definition

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

Intersection Contract

Assertion

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

- $(add1\ 3) \quad \longrightarrow^{*} 4$ ✓

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 $\mathcal{D}$)

Intersection Contract

Assertion

Let $add1 = ((\lambda x.x + 1) \ @ \ ({\color{red}Even} \to Even) \cap ({\color{red}Pos} \to Pos))$

- $(add1\ 3) \longrightarrow^* 4$ ✓
- $(add1\ -1) \longrightarrow^*$ ✗ blame context $\square$ -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 $\mathcal{D}$)

Intersection Contract

Assertion

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

- $(add1\ 3) \longrightarrow^* 4$ ✓
- $(add1\ -1) \longrightarrow^*$ ✗ blame context $\Box\ -1$
- $(add1\ 2) \longrightarrow^*$ ✗ 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 $\mathcal{D}$)

Contract Assertion

Example

- $((\lambda x.x + 1) \mathbin{@} (\mathit{Even} \rightarrow \mathit{Even}) \cap (\mathit{Pos} \rightarrow \mathit{Pos}))\ 3 \longrightarrow^* 4$

- 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

Contract Assertion

Example

- $((\lambda x.x + 1) \mathbin{@} (\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos}))\ 3 \longrightarrow^* 4$

- 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

Contract Assertion

Example

- $((\lambda x.x + 1) \mathbin{@} (\textit{Even} \rightarrow \textit{Even}) \cap (\textit{Pos} \rightarrow \textit{Pos}))\ 3 \longrightarrow^{*} 4$

- 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

Contract Assertion

Example

- $((\lambda x.x + 1) \ @ \ (\textit{Even} \rightarrow \textit{Even}) \cap (\textit{Pos} \rightarrow \textit{Pos}))\ 3 \longrightarrow^* 4$ ✓

- 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

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

Flat Contract

Evaluation Rule

Flat

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

Interpretation

Interpretation of a constraint list

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

- 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 $\flat.\mathit{context}$

Flat Contract (cont'd)

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

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

Blame Calculation

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.

Function Contract

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

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

Intersection Contract

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

$$\frac{\mu(\flat.subject) \sqsupseteq \mu(\flat_1.subject \wedge \flat_2.subject) \quad \mu(\flat.context) \sqsupseteq \mu(\flat_1.context \vee \flat_2.context)}{\mu \models \flat \blacktriangleleft \flat_1 \cap \flat_2}$$

Union Contract

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

In the Paper

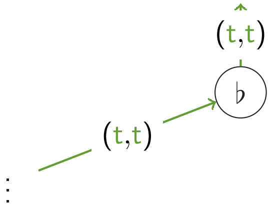
Technical Results

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

Conclusion

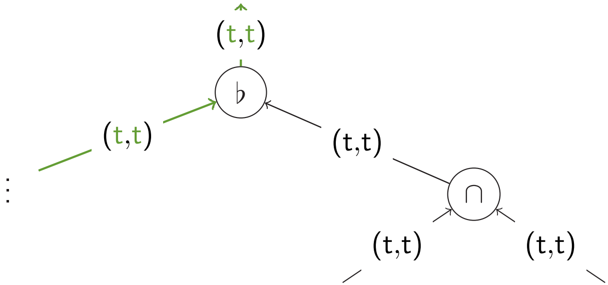
- Intersection and union contracts provide dynamic guarantees equivalent to their type-theoretic counterparts
- Constraint-based blame calculation enables higher-order contracts with unrestricted intersection and union
- Formal basis of *TreatJS*, a language embedded, higher-order contract system implemented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

Constraint Graph



Reduction

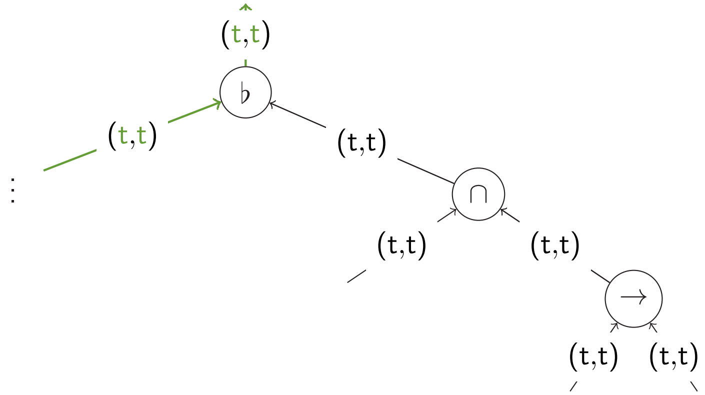
- ς,
  $((\lambda x.x + 1)\ @^\flat\ ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos})))\ 0$



Reduction

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

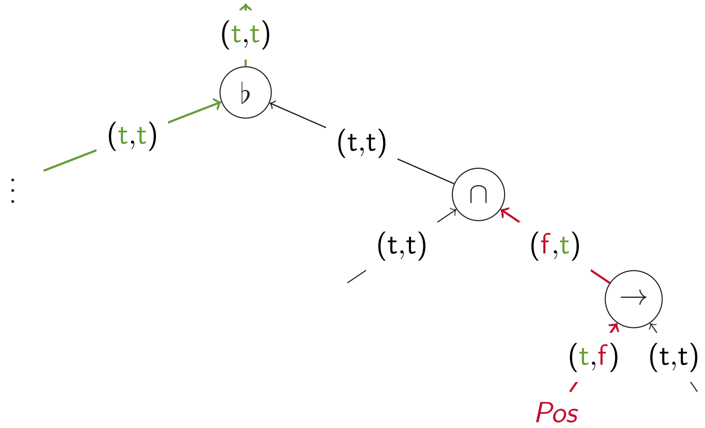
Constraint Graph



Reduction

- $\longrightarrow \flat_2 \blacktriangleleft (\flat_3 \to \flat_4) : \cdots,$
  $(((\lambda x.x + 1) @^{\flat_1} (\mathit{Even} \to \mathit{Even})) (0 @^{\flat_3} \mathit{Pos})) @^{\flat_4} \mathit{Pos}$

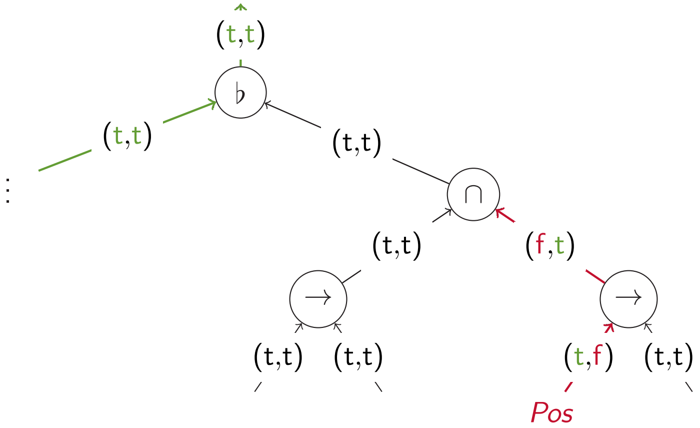
Constraint Graph



Reduction

- $\longrightarrow \flat_3 \blacktriangleleft (\mathit{false}) : \cdots,$
  $(((\lambda x.x + 1)\ @^{\flat_1}\ (\mathit{Even} \to \mathit{Even}))\ 0)\ @^{\flat_4}\ \mathit{Pos}$

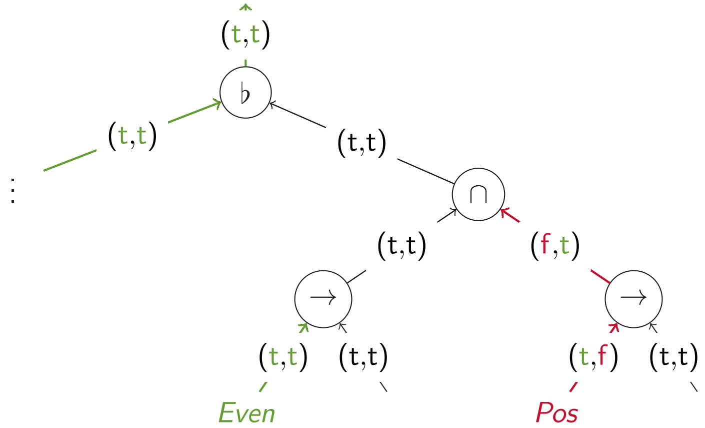
Constraint Graph



Reduction

- $\longrightarrow \flat_1 \blacktriangleleft (\flat_5 \to \flat_6) : \cdots,$
  $(((\lambda x.x + 1)\ (0\ @^{\flat_5}\ \mathit{Even}))\ @^{\flat_6}\ \mathit{Even})\ @^{\flat_4}\ \mathit{Pos}$

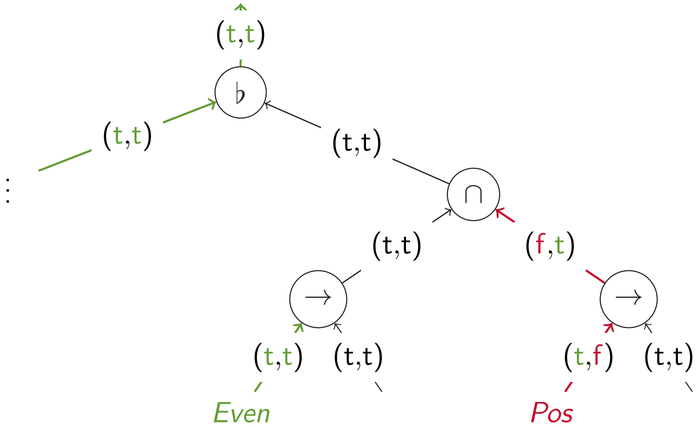
Constraint Graph



Reduction

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

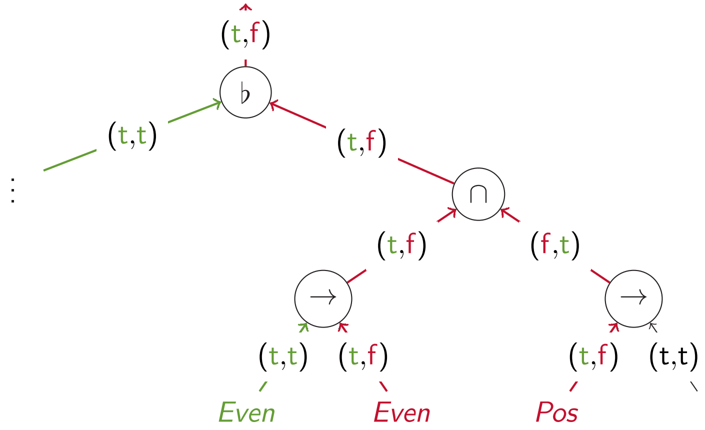
Constraint Graph



Reduction

- $\longrightarrow \cdots,$
  $(1 \ @^{\flat 6} \ \mathit{Even}) \ @^{\flat 4} \ \mathit{Pos}$

Constraint Graph



Reduction

- $\longrightarrow \flat_6 \blacktriangleleft (\mathit{false}) : \cdots,$
  $\mathit{blame}^{\flat}$

Intersection and Union Types

Intersection Type

- $\lambda x.x + 2 : \mathit{Even} \to \mathit{Even}$
- $\lambda x.x + 2 : \mathit{Odd} \to \mathit{Odd}$
- $\lambda x.x + 2 : \mathit{Even} \to \mathit{Even} \cap \mathit{Odd} \to \mathit{Odd}$

Union Type

- $\lambda x.x - 2 : \mathit{Even} \to \mathit{Even}$
- $\lambda x.x - 2 : \mathit{Even} \to \mathit{Even} \cup \mathit{Pos} \to \mathit{Pos}$

Flat Contract [Findler,Felleisen'02]

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

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

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: $(M\ V) \longrightarrow^* false$

Higher-Order Contract [Findler,Felleisen'02]

- $Even \rightarrow Even$

Higher-Order Contract [Findler,Felleisen'02]

- $Even \rightarrow Even$

Assertion

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

Definition

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

Higher-Order Contract [Findler,Felleisen'02]

- $\mathit{Even} \rightarrow \mathit{Even}$

Assertion

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

Flat Contract

Examples

- $\mathit{Odd} \cup \mathit{Even}$

Flat Contract

Examples

- *Odd* $\cup$ *Even*

Flat Contract

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

Union Contract

Assertion

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

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 $\mathcal{D}$)

Union Contract

Assertion

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

- $(mod3\ 4) \longrightarrow^{*} 1$ ✓

Definition

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

Union Contract

Assertion

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

- $(mod3\ 4) \longrightarrow^* 1$ ✓
- $(mod3\ 1) \longrightarrow^*$ ✗ blame context $\Box$ 1

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 $\mathcal{D}$)

Union Contract

Assertion

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 $\mathcal{D}$)

Contract Assertion

Evaluation Rule

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

Constraint Satisfaction

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

Constraint List

Constraint Satisfaction

CS-Empty

$$\mu \models \cdot$$

CS-Cons

$$\frac{\mu \models \kappa \qquad \mu \models \varsigma}{\mu \models \kappa : \varsigma}$$

Union Contract

Evaluation Rule

Union

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

Constraint Satisfaction

C-Union

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

Blame Calculation

Definition

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

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

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

$$\frac{\varsigma \text{ is blame state for } \flat}{\varsigma, M \longmapsto \varsigma, \mathit{blame}^{\flat}}$$

Example Reduction

Reduction

- $\cdot,$
  $((\lambda x.x + 1) \;@^{\flat}\; ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos}))) \; 0$

Example Reduction

Reduction

- $\cdot,$
  $((\lambda x.x + 1) \; @^{\flat} \; ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos}))) \; 0$
- $\longrightarrow \flat\blacktriangleleft(\flat_0) : \cdot,$
  $((\lambda x.x + 1) \; @^{\flat_0} \; ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos}))) \; 0$

Example Reduction

Reduction

- $\cdot,$
  $((\lambda x.x+1) \text{ @}^{\flat} ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos}))) \; 0$
- $\longrightarrow \flat \blacktriangleleft (\flat_0) : \cdot,$
  $((\lambda x.x+1) \text{ @}^{\flat_0} ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos}))) \; 0$
- $\longrightarrow \flat_0 \blacktriangleleft (\flat_1 \cap \flat_2) : \cdots,$
  $(((\lambda x.x+1) \text{ @}^{\flat_1} (\mathit{Even} \to \mathit{Even})) \text{ @}^{\flat_2} (\mathit{Pos} \to \mathit{Pos})) \; 0$

Example Reduction

Reduction

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

Example Reduction

Reduction

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

Example Reduction

Reduction

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

Example Reduction

Reduction

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

Example Reduction

Reduction

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

Example Reduction

Reduction

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

Example Reduction

Reduction

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

Example Reduction

Reduction

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

Example Reduction

Reduction

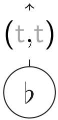
- $\longrightarrow \cdots,$
  $(1 \mathbin{@^{\flat_6}} \mathit{Even}) \mathbin{@^{\flat_4}} \mathit{Pos}$
- $\longrightarrow \flat_6 \blacktriangleleft (\mathit{false}) : \cdots,$
  $\mathit{blame}^{\flat}$

Example Reduction

Reduction

- $\longrightarrow \flat_6 \blacktriangleleft(\mathit{false}) : \cdots,$
  $\mathit{blame}^{\flat}$

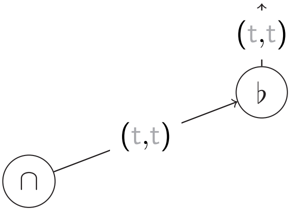
Constraint Graph



Example

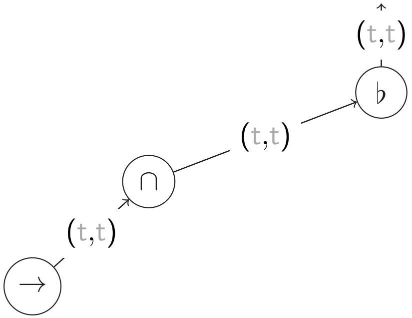
- $((\lambda x.x+1)\ @^{\flat}\ ((\mathit{Even} \rightarrow \mathit{Even}) \cap (\mathit{Pos} \rightarrow \mathit{Pos})))\ 1 \longrightarrow^{*} 2$ ✓

Constraint Graph



Example

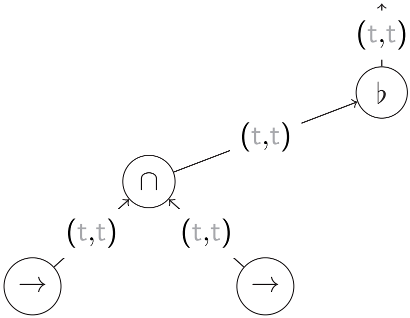
- $((\lambda x.x + 1)\ @^{\flat}\ ((Even \rightarrow Even) \cap (Pos \rightarrow Pos)))\ 1 \longrightarrow^{*} 2$ ✓



Example

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

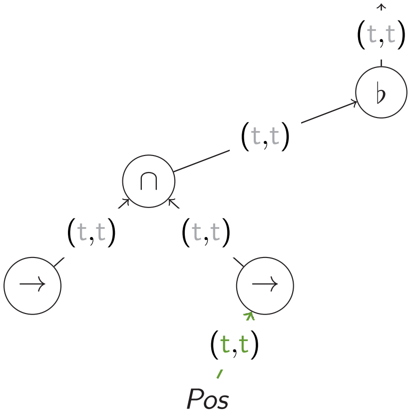
Constraint Graph



Example

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

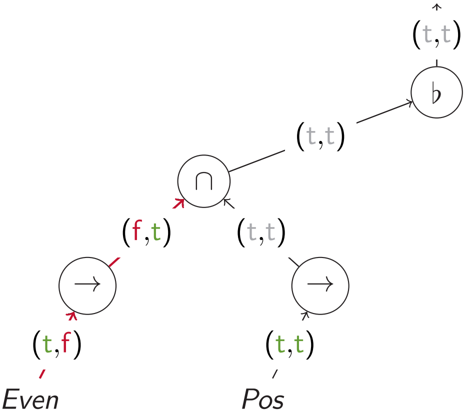
Constraint Graph



Example

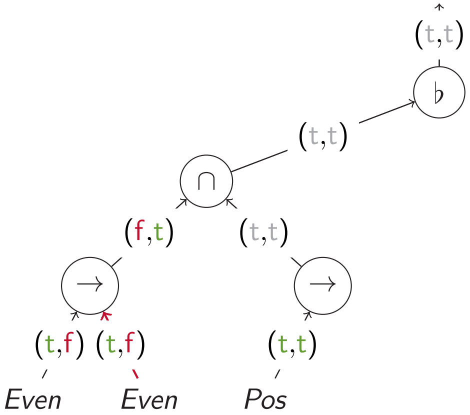
- $((\lambda x.x+1)\ @^{\flat}\ ((Even \to Even) \cap (Pos \to Pos)))\ 1 \longrightarrow^{*} 2$ ✓

Constraint Graph



Example

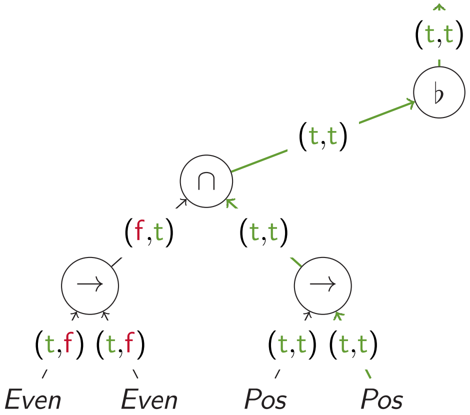
- $((\lambda x.x + 1)\ @^{\flat}\ ((\mathit{Even} \to \mathit{Even}) \cap (\mathit{Pos} \to \mathit{Pos})))\ 1 \longrightarrow^{*} 2$ ✓



Example

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

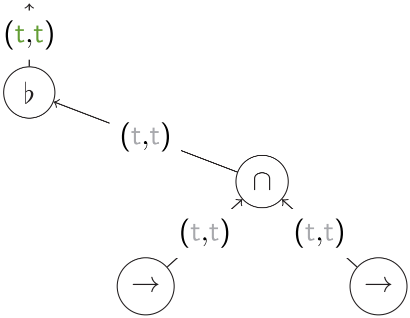
Constraint Graph



Example

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

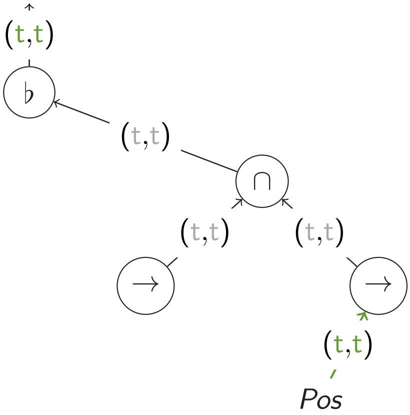
Constraint Graph



Example

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

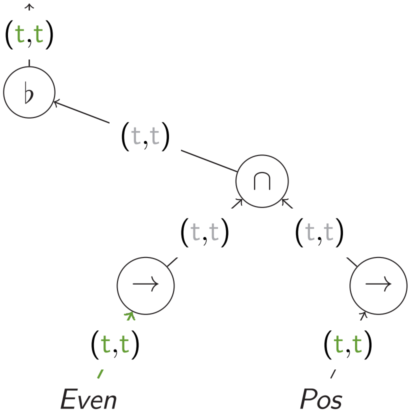
Constraint Graph



Example

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

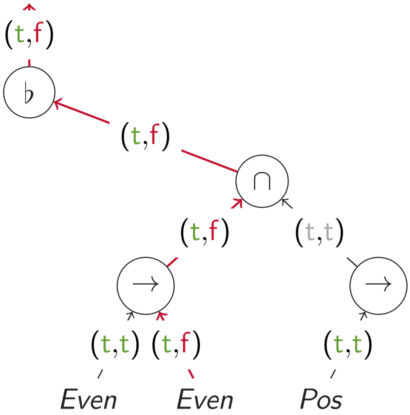
Constraint Graph



Example

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

Constraint Graph



Example

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

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 $\mathcal{C}$.

Technical Results (cont'd)

Theorem (Contract soundness for expressions)

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

Theorem (Contract soundness for contexts)

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

Technical Results (cont'd)

Theorem (Subject blame soundness)

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

Theorem (Context blame soundness)

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