# Trace-Based Control-Flow Analysis

Benoît Montagu, Thomas Jensen

Inria

Cambium Seminar — June 14th 2021

# Salto project

**What:** static analysis for OCaml programs

**Where:** Celtique research team, Inria Rennes

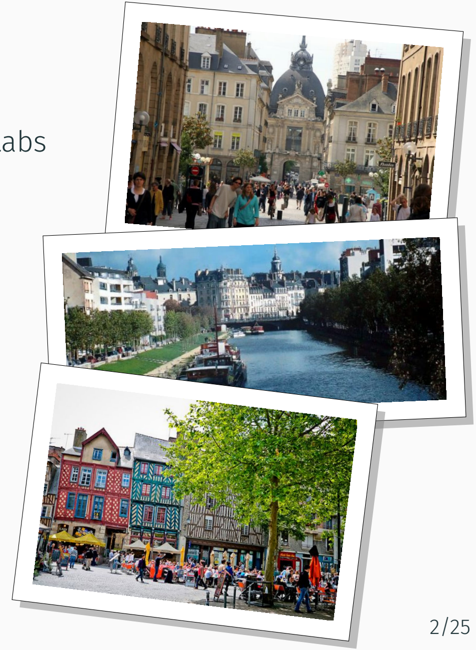
**Who:** B. Montagu + T. Genet + T. Jensen + Nomadic Labs

**When:** from October 2021

We are *hiring* a research engineer
on a 3 year contract

Contact me!

benoit.montagu@inria.fr



# Control-Flow Analysis

# Control-Flow Analysis

**A classic static analysis for the $\lambda$-calculus** (Jones and Mycroft 1986; Shivers 1991)

For a whole program, CFA answers the questions:

- Which values can be produced by a program?
- Which values can be produced, at any program point of a program?
- To which values can bound variables be assigned?
- 👉 Useful for compilers and for program verification

$$\left(\left(\left(\lambda x.\ (x\ \ x\ )\ \right)\ \ (\lambda y.\ y\ )\ \right)\ \ (\lambda z.\ z\ )\ \right) \longrightarrow^{\star} \lambda z.\ z$$

# Control-Flow Analysis

**A classic static analysis for the $\lambda$-calculus** (Jones and Mycroft 1986; Shivers 1991)

For a whole program, CFA answers the questions:

- Which values can be produced by a program?
- Which values can be produced, at any program point of a program?
- To which values can bound variables be assigned?
- ☞ Useful for compilers and for program verification

$$\left(\left(\left(\lambda x.\ (x^0\, x^1)^2\right)^3\ (\lambda y.\, y^4)^5\right)^6\ (\lambda z.\, z^7)^8\right)^9 \longrightarrow^\star \lambda z.\, z^7$$

# Control-Flow Analysis

**A classic static analysis for the $\lambda$-calculus** (Jones and Mycroft 1986; Shivers 1991)

For a whole program, CFA answers the questions:

- Which values can be produced by a program?
- Which values can be produced, at any program point of a program?
- To which values can bound variables be assigned?
- Useful for compilers and for program verification

$$\Big(\big(\big(\lambda x.\, (x^0\, x^1)^2\big)^3\; (\lambda y.\, y^4)^5\big)^6\; (\lambda z.\, z^7)^8\Big)^9 \longrightarrow^\star \lambda z.\, z^7$$

**For each program point:**

the "cache" $\hat{C}$
$$\begin{cases} 2 \mapsto \{\lambda y.\, y^4\} \\ 6 \mapsto \{\lambda y.\, y^4\} \\ 9 \mapsto \{\lambda z.\, z^7\} \end{cases}$$
...

# Control-Flow Analysis

**A classic static analysis for the $\lambda$-calculus** (Jones and Mycroft 1986; Shivers 1991)

For a whole program, CFA answers the questions:

- Which values can be produced by a program?
- Which values can be produced, at any program point of a program?
- To which values can bound variables be assigned?
- Useful for compilers and for program verification

$$\left(\left(\left(\lambda x.\ (x^0\, x^1)^2\right)^3\ (\lambda y.\, y^4)^5\right)^6\ (\lambda z.\, z^7)^8\right)^9 \longrightarrow^\star \lambda z.\, z^7$$

**For each program point:**

the "cache" $\hat{C}$ $\begin{cases} 2 \mapsto \{\lambda y.\, y^4\} \\ 6 \mapsto \{\lambda y.\, y^4\} \\ 9 \mapsto \{\lambda z.\, z^7\} \end{cases}$

...

**For each variable:**

the "environment" $\hat{\rho}$ $\begin{cases} x \mapsto \{\lambda y.\, y^4\} \\ y \mapsto \{\lambda y.\, y^4, \lambda z.\, z^7\} \\ z \mapsto \{\} \end{cases}$

# The importance of context sensitivity in CFA

For greater precision, one must distinguish *instances* of variables

$$\left(\left(\left(\lambda x.\, (x^0\, x^1)^2\right)^3\, (\lambda y.\, y^4)^5\right)^6\, (\lambda z.\, z^7)^8\right)^9 \longrightarrow^\star \lambda z.\, z^7$$

# The importance of context sensitivity in CFA

For greater precision, one must distinguish *instances* of variables

$$\left(\left(\left(\lambda x.\ (x^0\ x^1)^2\right)^3\ (\lambda y.\ y^4)^5\right)^6\ (\lambda z.\ z^7)^8\right)^9 \longrightarrow^\star \lambda z.\ z^7$$

**No context sensitivity:** $\hat{\rho}(y) = \{\lambda y.\ y^4, \lambda z.\ z^7\}$ has a snowball effect!

$\hat{C}(6) = \{\lambda y.\ y^4, \lambda z.\ z^7\}$ $\hat{C}(9) = \{\lambda y.\ y^4, \lambda z.\ z^7\}$

☞ Final result approximated: $\{\lambda y.\ y^4, \lambda z.\ z^7\}$ instead of $\{\lambda z.\ z^7\}$

# The importance of context sensitivity in CFA

For greater precision, one must distinguish *instances* of variables

$$\left(\left(\left(\lambda x.\ (x^0\ x^1)^2\right)^3\ (\lambda y.\ y^4)^5\right)^6\ (\lambda z.\ z^7)^8\right)^9 \longrightarrow^\star \lambda z.\ z^7$$

**No context sensitivity:** $\hat{\rho}(y) = \{\lambda y.\ y^4, \lambda z.\ z^7\}$ has a snowball effect!

$\hat{C}(6) = \{\lambda y.\ y^4, \lambda z.\ z^7\}$ $\hat{C}(9) = \{\lambda y.\ y^4, \lambda z.\ z^7\}$

☞ Final result approximated: $\{\lambda y.\ y^4, \lambda z.\ z^7\}$ instead of $\{\lambda z.\ z^7\}$

**With context sensitivity:** distinguish instances of variables using calling contexts

In calling context 2: $\hat{\rho}(y) = \{\lambda y.\ y^4\}$ In calling context 9: $\hat{\rho}(y) = \{\lambda z.\ z^7\}$

☞ Final result becomes more precise: $\{\lambda z.\ z^7\}$

# The importance of context sensitivity in CFA

For greater precision, one must distinguish *instances* of variables

$$\left(\left(\left(\lambda x.\ (x^0\ x^1)^2\right)^3\ (\lambda y.\ y^4)^5\right)^6\ (\lambda z.\ z^7)^8\right)^9 \longrightarrow^\star \lambda z.\ z^7$$

**No context sensitivity:** $\hat{\rho}(y) = \{\lambda y.\ y^4, \lambda z.\ z^7\}$ has a snowball effect!

$\hat{C}(6) = \{\lambda y.\ y^4, \lambda z.\ z^7\}$ $\hat{C}(9) = \{\lambda y.\ y^4, \lambda z.\ z^7\}$

☞ Final result approximated: $\{\lambda y.\ y^4, \lambda z.\ z^7\}$ instead of $\{\lambda z.\ z^7\}$

**With context sensitivity:** distinguish instances of variables using calling contexts

In calling context 2: $\hat{\rho}(y) = \{\lambda y.\ y^4\}$ In calling context 9: $\hat{\rho}(y) = \{\lambda z.\ z^7\}$

☞ Final result becomes more precise: $\{\lambda z.\ z^7\}$

Call strings = sequence of locations of function calls ⚠ Infinite number!

☞ A standard strategy: $k$-CFA remembers only the $k$ most recent calls

# The issue with closures in CFA

Evaluation can produce closures of unbounded heights:

```
let id x = x in
let compose f g x = f (g x) in
let rec iter n h =
  if n <= 0 then id
  else compose h (iter (n−1) h)
in
iter 42 id
```

# The issue with closures in CFA

Evaluation can produce closures of unbounded heights:

```
let id x = x in
let compose f g x = f (g x) in
let rec iter n h =
  if n <= 0 then id
  else compose h (iter (n−1) h)
in
iter 42 id
```

$$(\lambda x.\, f\,(g\,x)) \left[ \begin{array}{l} f \mapsto (\lambda x.\, x), \\ g \mapsto (\lambda x.\, f\,(g\,x)) \left[ \begin{array}{l} f \mapsto (\lambda x.\, x), \\ g \mapsto \ldots \end{array} \right] \end{array} \right]$$

Sets of such closures must be represented in a finite way!

# The issue with closures in CFA

Evaluation can produce closures of unbounded heights:

```
let id x = x in
let compose f g x = f (g x) in
let rec iter n h =
  if n <= 0 then id
  else compose h (iter (n-1) h)
in
iter 42 id
```

$$(\lambda x.\, f\,(g\,x)) \left[ \begin{array}{l} f \mapsto (\lambda x.\, x), \\ g \mapsto (\lambda x.\, f\,(g\,x)) \left[ \begin{array}{l} f \mapsto (\lambda x.\, x), \\ g \mapsto \dots \end{array} \right] \end{array} \right]$$

Sets of such closures must be represented in a finite way!

CFA introduces indirections through the environment $\rho$:

$$\{(\lambda x.\, f\,(g\,x))[f \mapsto f, g \mapsto g]\} \text{ where } \begin{cases} \rho(f) = \{(\lambda x.\, x)[]\} \\ \rho(g) = \{(\lambda x.\, x), (\lambda x.\, f\,(g\,x))[f \mapsto f, g \mapsto g]\} \end{cases}$$

# The issue with closures in CFA

**Evaluation can produce closures of unbounded heights:**

```
let id x = x in
let compose f g x = f (g x) in
let rec iter n h =
  if n <= 0 then id
  else compose h (iter (n−1) h)
in
iter 42 id
```

$$(\lambda x.\, f\,(g\,x))\left[\begin{array}{l} f \mapsto (\lambda x.\, x), \\ g \mapsto (\lambda x.\, f\,(g\,x))\left[\begin{array}{l} f \mapsto (\lambda x.\, x), \\ g \mapsto \ldots \end{array}\right] \end{array}\right]$$

Sets of such closures must be represented in a finite way!

CFA introduces indirections through the environment $\rho$:

$$\{(\lambda x.\, f\,(g\,x))[f \mapsto f, g \mapsto g]\} \text{ where } \begin{cases} \rho(f) = \{(\lambda x.\, x)[\,]\} \\ \rho(g) = \{(\lambda x.\, x), (\lambda x.\, f\,(g\,x))[f \mapsto f, g \mapsto g]\} \end{cases}$$

"$\rho$ is a store where shallow closures are allocated"

Addresses in the store ≈ variable names + call strings

# What is this talk about?

**Remark 1:** environment $\rho$ is a heap abstraction (Horn and Might 2010)

The environment $\rho$ is semantically justified using an imperative semantics to $\lambda$-terms: slices of values are allocated and stored in a global heap.

☞ Part 1 of this talk: a more natural justification, based on execution traces

# What is this talk about?

**Remark 1:** environment $\rho$ is a heap abstraction (Horn and Might 2010)

The environment $\rho$ is semantically justified using an imperative semantics to $\lambda$-terms: slices of values are allocated and stored in a global heap.

☞ Part 1 of this talk: a more natural justification, based on execution traces

☞ Part 2 of this talk: a semantic justification for constraint-based CFA

# What is this talk about?

**Remark 1:** environment $\rho$ is a heap abstraction (Horn and Might 2010)

The environment $\rho$ is semantically justified using an imperative semantics to $\lambda$-terms: slices of values are allocated and stored in a global heap.

☞ Part 1 of this talk: a more natural justification, based on execution traces

☞ Part 2 of this talk: a semantic justification for constraint-based CFA

**Remark 2:** two sources of divergence for a CFA analyser

1. An infinity of calling contexts (call strings)
2. An infinity of possible closures (infinity of addresses in $\rho$)

Standard solution in CFA: use a domain with finite height

- ⊞ Makes the search space finite, which ensures convergence
- ⊟ Expressive domains have unbounded heights (e.g., intervals of integers)

☞ Part 3 of this talk: CFA with unbounded domains, based on widening

# Traces for Control-Flow

# Traces for Control Flow

- Language of study: untyped $\lambda$-calculus
- Textbook reduction semantics for $\lambda$-calculus + a trace:

$$t \xrightarrow{\mathrm{tr}}{}^{*} u$$

- **One event for function calls**: which $\beta$-reduction happened?

$$\beta(\pi, \lambda^{\ell}\mathrm{x}.\, t, v) \qquad \text{(context, called function, argument)}$$

- **One event for returning results**: which value was produced?

$$\mathrm{Ret}(\pi, a, v) \qquad \text{(context, program point, produced value)}$$

- … and labels to tell **in which contexts** the events were produced

$$\frac{t \xrightarrow{\mathrm{tr}}{}^{*} u}{[t]^{a} \xrightarrow{a \cdot \mathrm{tr}}{}^{*} [u]^{a}} \qquad \text{(details in the paper)}$$

# A Labelled $\lambda$-Calculus that Traces Control-Flow Events

$$
\begin{array}{llll}
t \in \mathcal{T} & ::= & \mathsf{x} & \text{(Variables)} \\
 & \mid & \lambda^{\ell}\mathsf{x}.\, t & \text{(Abstraction)} \\
 & \mid & t\, t & \text{(Application)} \\
 & \mid & [t]^{a} & \text{(Annotation)} \\
\mathrm{tr} \in \mathbb{T} & ::= & \varepsilon \quad \mid \quad \mathrm{e}, \mathrm{tr} & \text{(Traces)}
\end{array}
$$

$$
\begin{array}{llll}
a & ::= & & \text{(Annotations)} \\
 & \mid & p & \text{(program points)} \\
 & \mid & \mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, t, v) & \text{(reduction points)} \\
\mathrm{e} & ::= & \mathsf{Ret}(\pi, a, v) & \text{(Events)} \\
 & \mid & \beta(\pi, \lambda^{\ell}\mathsf{x}.\, t, v) &
\end{array}
$$

# A Labelled $\lambda$-Calculus that Traces Control-Flow Events

$$
\begin{array}{lcll}
t \in \mathcal{T} & ::= & \mathsf{x} & \text{(Variables)} \\
 & \mid & \lambda^{\ell}\mathsf{x}.\, t & \text{(Abstraction)} \\
 & \mid & t\ t & \text{(Application)} \\
 & \mid & [t]^{a} & \text{(Annotation)} \\
\mathrm{tr} \in \mathbb{T} & ::= & \varepsilon \quad \mid \quad \mathrm{e}, \mathrm{tr} & \text{(Traces)}
\end{array}
$$

$$
\begin{array}{lcll}
a & ::= & & \text{(Annotations)} \\
 & \mid & p & \text{(program points)} \\
 & \mid & \mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, t, v) & \text{(reduction points)} \\
\mathrm{e} & ::= & \mathsf{Ret}(\pi, a, v) & \text{(Events)} \\
 & \mid & \beta(\pi, \lambda^{\ell}\mathsf{x}.\, t, v) &
\end{array}
$$

$\lambda^{\ell}\mathsf{x}.\, t$ was called on argument $v$

Context for next reductions

$$(\lambda^{\ell}\mathsf{x}.\, t)\ v \xrightarrow{\beta(\varepsilon, \lambda^{\ell}\mathsf{x}.\, t, v)} [t[\mathsf{x} \leftarrow v]]^{\mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, t, v)}$$

# A Labelled $\lambda$-Calculus that Traces Control-Flow Events

$$
\begin{array}{rcll}
t \in \mathcal{T} & ::= & \mathsf{x} & \text{(Variables)} \\
 & | & \lambda^\ell \mathsf{x}.\, t & \text{(Abstraction)} \\
 & | & t\, t & \text{(Application)} \\
 & | & [t]^a & \text{(Annotation)} \\
\mathrm{tr} \in \mathbb{T} & ::= & \varepsilon \quad | \quad \mathrm{e}, \mathrm{tr} & \text{(Traces)}
\end{array}
$$

$$
\begin{array}{rcll}
a & ::= & & \text{(Annotations)} \\
 & | & p & \text{(program points)} \\
 & | & \mathsf{Call}(\lambda^\ell \mathsf{x}.\, t, v) & \text{(reduction points)} \\
\mathrm{e} & ::= & \mathsf{Ret}(\pi, a, v) & \text{(Events)} \\
 & | & \beta(\pi, \lambda^\ell \mathsf{x}.\, t, v) &
\end{array}
$$

$\lambda^\ell \mathsf{x}.\, t$ was called on argument $v$

Context for next reductions

$v$ was returned at annotation $a$

$$(\lambda^\ell \mathsf{x}.\, t)\ v \xrightarrow{\beta(\varepsilon, \lambda^\ell \mathsf{x}.\, t, v)} [t[\mathsf{x} \leftarrow v]]^{\mathsf{Call}(\lambda^\ell \mathsf{x}.\, t, v)}$$

$$[v]^a \xrightarrow{\mathsf{Ret}(\varepsilon, a, v)} v$$

# A Labelled $\lambda$-Calculus that Traces Control-Flow Events

$$
\begin{array}{llll}
t \in \mathcal{T} & ::= & \mathsf{x} & \text{(Variables)} \\
& | & \lambda^\ell \mathsf{x}.\, t & \text{(Abstraction)} \\
& | & t\, t & \text{(Application)} \\
& | & [t]^a & \text{(Annotation)} \\
\mathrm{tr} \in \mathbb{T} & ::= & \varepsilon \quad | \quad \mathrm{e}, \mathrm{tr} & \text{(Traces)}
\end{array}
$$

$$
\begin{array}{llll}
a & ::= & & \text{(Annotations)} \\
& | & p & \text{(program points)} \\
& | & \mathsf{Call}(\lambda^\ell \mathsf{x}.\, t, v) & \text{(reduction points)} \\
\mathrm{e} & ::= & \mathsf{Ret}(\pi, a, v) & \text{(Events)} \\
& | & \beta(\pi, \lambda^\ell \mathsf{x}.\, t, v) &
\end{array}
$$

$\lambda^\ell \mathsf{x}.\, t$ was called on argument $v$

Context for next reductions

$v$ was returned at annotation $a$

$$(\lambda^\ell \mathsf{x}.\, t)\, v \xrightarrow{\beta(\varepsilon, \lambda^\ell \mathsf{x}.\, t, v)} [t[\mathsf{x} \leftarrow v]]^{\mathsf{Call}(\lambda^\ell \mathsf{x}.\, t, v)} \qquad [v]^a \xrightarrow{\mathsf{Ret}(\varepsilon, a, v)} v$$

$$\frac{t \xrightarrow{\mathrm{e}} t'}{t\, u \xrightarrow{\mathrm{e}} t'\, u} \qquad \frac{t \xrightarrow{\mathrm{e}} t'}{v\, t \xrightarrow{\mathrm{e}} v\, t'} \qquad \frac{t \xrightarrow{\mathrm{e}} u}{[t]^a \xrightarrow{a \cdot \mathrm{e}} [u]^a}$$

Contexts are recorded in the events

# Example

$$\left[\left[\left[\lambda^{\mathsf{a}}\mathsf{x}.\ \left[[\mathsf{x}]^0\ [\mathsf{x}]^1\right]^2\right]^3\ \left[\lambda^{\mathsf{b}}\mathsf{y}.\ [\mathsf{y}]^4\right]^5\right]^6\ \left[\lambda^{\mathsf{c}}\mathsf{z}.\ [\mathsf{z}]^7\right]^8\right]^9 \longrightarrow^{\star} \lambda^{\mathsf{c}}\mathsf{z}.\ [\mathsf{z}]^7$$

# Example

$$\left[\left[\left[\lambda^{a}x.\ \left[[x]^{0}\ [x]^{1}\right]^{2}\right]^{3}\ \left[\lambda^{b}y.\ [y]^{4}\right]^{5}\right]^{6}\ \left[\lambda^{c}z.\ [z]^{7}\right]^{8}\right]^{9} \longrightarrow^{\star} \lambda^{c}z.\ [z]^{7}$$

# Example

$$\left[\left[\left[\lambda^{a}x.\ \left[[x]^{0}\ [x]^{1}\right]^{2}\right]^{3}\left[\lambda^{b}y.\ [y]^{4}\right]^{5}\right]^{6}\left[\lambda^{c}z.\ [z]^{7}\right]^{8}\right]^{9} \longrightarrow^{\star} \lambda^{c}z.\ [z]^{7}$$

Reduct: $\lambda^{a}x.\ \left[[x]^{0}\ [x]^{1}\right]^{2}$

Event: $\mathrm{Ret}(9 \cdot 6, 3, \lambda^{a}x.\ \left[[x]^{0}\ [x]^{1}\right]^{2})$

# Example

$$\left[\left[\left(\lambda^{a}x.\ \left[[x]^0\ [x]^1\right]^2\right)\ \left[\lambda^{b}y.\ [y]^4\right]^5\right]^6\ \left[\lambda^{c}z.\ [z]^7\right]^8\right]^9 \longrightarrow^\star \lambda^{c}z.\ [z]^7$$

Reduct: $\lambda^{a}x.\ \left[[x]^0\ [x]^1\right]^2$

Event: $\mathrm{Ret}(9 \cdot 6, 3, \lambda^{a}x.\ \left[[x]^0\ [x]^1\right]^2)$

# Example

$$\Big[\Big[\Big(\lambda^{a}x.\ \big[[x]^{0}\ [x]^{1}\big]^{2}\Big)\ \big[\lambda^{b}y.\ [y]^{4}\big]^{5}\Big]^{6}\ \big[\lambda^{c}z.\ [z]^{7}\big]^{8}\Big]^{9} \longrightarrow^{\star} \lambda^{c}z.\ [z]^{7}$$

# Example

$$\left[\left[\left(\lambda^{a} x.\ \left[[x]^{0}\ [x]^{1}\right]^{2}\right)\ \left[\lambda^{b} y.\ [y]^{4}\right]^{5}\right]^{6}\ \left[\lambda^{c} z.\ [z]^{7}\right]^{8}\right]^{9} \longrightarrow^{\star} \lambda^{c} z.\ [z]^{7}$$

Reduct: $\lambda^{b} y.\ [y]^{4}$

Event: $\mathrm{Ret}(9 \cdot 6, 5, \lambda^{b} y.\ [y]^{4})$

# Example

$$\left[\left[\left(\lambda^{\mathsf{a}}\mathsf{x}.\ \left[[\mathsf{x}]^{0}\ [\mathsf{x}]^{1}\right]^{2}\right)\left(\lambda^{\mathsf{b}}\mathsf{y}.\ [\mathsf{y}]^{4}\right)\right]^{6}\left[\lambda^{\mathsf{c}}\mathsf{z}.\ [\mathsf{z}]^{7}\right]^{8}\right]^{9} \longrightarrow^{\star} \lambda^{\mathsf{c}}\mathsf{z}.\ [\mathsf{z}]^{7}$$

Reduct: $\lambda^{\mathsf{b}}\mathsf{y}.\ [\mathsf{y}]^{4}$

Event: $\mathsf{Ret}(9 \cdot 6, 5, \lambda^{\mathsf{b}}\mathsf{y}.\ [\mathsf{y}]^{4})$

# Example

$$\left[\left[\left(\lambda^{a}x.\ \left[[x]^{0}\ [x]^{1}\right]^{2}\right)\ \left(\lambda^{b}y.\ [y]^{4}\right)\right]^{6}\ \left[\lambda^{c}z.\ [z]^{7}\right]^{8}\right]^{9} \longrightarrow^{\star} \lambda^{c}z.\ [z]^{7}$$

# Example

$$\left[\left[\left(\lambda^{a}x.\ \left[[x]^0\ [x]^1\right]^2\right)\ \left(\lambda^{b}y.\ [y]^4\right)\right]^6\ \left[\lambda^{c}z.\ [z]^7\right]^8\right]^9 \longrightarrow^{\star} \lambda^{c}z.\ [z]^7$$

Reduct: $\left[\left[[v_b]^0\ [v_b]^1\right]^2\right]^{\mathsf{Call}(v_a,v_b)}$

Event: $\beta(9\cdot 6, v_a, v_b)$

where $v_a = \lambda^{a}x.\ \left[[x]^0\ [x]^1\right]^2$ and $v_b = \lambda^{b}y.\ [y]^4$

# Example

$$\left[\left[\left[\left[[v_\mathsf{b}]^0\ [v_\mathsf{b}]^1\right]^2\right]^{\mathsf{Call}(v_\mathsf{a},v_\mathsf{b})}\right]^6 \left[\lambda^\mathsf{c}\mathsf{z}.\ [\mathsf{z}]^7\right]^8\right]^9 \longrightarrow^\star \lambda^\mathsf{c}\mathsf{z}.\ [\mathsf{z}]^7$$

Reduct: $\left[\left[[v_\mathsf{b}]^0\ [v_\mathsf{b}]^1\right]^2\right]^{\mathsf{Call}(v_\mathsf{a},v_\mathsf{b})}$

Event: $\beta(9 \cdot 6, v_\mathsf{a}, v_\mathsf{b})$

where $v_\mathsf{a} = \lambda^\mathsf{a}\mathsf{x}.\ \left[[\mathsf{x}]^0\ [\mathsf{x}]^1\right]^2$ and $v_\mathsf{b} = \lambda^\mathsf{b}\mathsf{y}.\ [\mathsf{y}]^4$

# Example

$$\left[\left[\left[\left[[v_b]^0 \; [v_b]^1\right]^2\right]^{\mathsf{Call}(v_a,v_b)}\right]^6 \left[\lambda^c z.\ [z]^7\right]^8\right]^9 \longrightarrow^\star \lambda^c z.\ [z]^7$$

where $v_a = \lambda^a x.\ \left[[x]^0\ [x]^1\right]^2$ and $v_b = \lambda^b y.\ [y]^4$

# Example

$$\left[\left[\left[\left[[v_b]^0\ [v_b]^1\right]^2\right]^{\mathsf{Call}(v_a,v_b)}\right]^6 \left[\lambda^c z.\ [z]^7\right]^8\right]^9 \longrightarrow^\star \lambda^c z.\ [z]^7$$

Reduct: $v_b$

Event: $\mathsf{Ret}(9 \cdot 6 \cdot \mathsf{Call}(v_a, v_b) \cdot 2, 0, v_b)$

where $v_a = \lambda^a x.\ \left[[x]^0\ [x]^1\right]^2$ and $v_b = \lambda^b y.\ [y]^4$

# Example

$$\left[\left[\left[\left[\left[v_{\mathsf{b}}\ [v_{\mathsf{b}}]^{1}\right]^{2}\right]^{\mathsf{Call}(v_{\mathsf{a}},v_{\mathsf{b}})}\right]^{6}\left[\lambda^{\mathsf{c}}\mathsf{z}.\ [\mathsf{z}]^{7}\right]^{8}\right]^{9} \longrightarrow^{\star} \lambda^{\mathsf{c}}\mathsf{z}.\ [\mathsf{z}]^{7}$$

Reduct: $v_{\mathsf{b}}$

Event: $\mathsf{Ret}(9 \cdot 6 \cdot \mathsf{Call}(v_{\mathsf{a}}, v_{\mathsf{b}}) \cdot 2, 0, v_{\mathsf{b}})$

where $v_{\mathsf{a}} = \lambda^{\mathsf{a}}\mathsf{x}.\ \left[[\mathsf{x}]^{0}\ [\mathsf{x}]^{1}\right]^{2}$ and $v_{\mathsf{b}} = \lambda^{\mathsf{b}}\mathsf{y}.\ [\mathsf{y}]^{4}$

# Example

$$\left[\left[\left[\left[v_b\ [v_b]^1\right]^2\right]^{\mathsf{Call}(v_a,v_b)}\right]^6 \left[\lambda^c z.\ [z]^7\right]^8\right]^9 \longrightarrow^\star \lambda^c z.\ [z]^7$$

and so on...

where $v_a = \lambda^a x.\ \left[[x]^0\ [x]^1\right]^2$ and $v_b = \lambda^b y.\ [y]^4$

## A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $$(\!|t|\!)_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\mathrm{tr}, v) \mid t \cdot \sigma \xrightarrow{\mathrm{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$$

# A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $$(\!|t|\!)_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\mathrm{tr}, v) \mid t \cdot \sigma \xrightarrow{\mathrm{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$$

Semantic inclusions:

$$(\!|[\mathsf{x}]^p|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, \sigma(\mathsf{x})), \sigma(\mathsf{x})) \mid \sigma \in \mathcal{I}\}$$

# A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $(\!|t|\!)_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\text{tr}, v) \mid t \cdot \sigma \xrightarrow{\text{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$

Semantic inclusions:

$$(\!|[\mathsf{x}]^{p}|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, \sigma(\mathsf{x})), \sigma(\mathsf{x})) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[\lambda^{\ell}\mathsf{x}.\, t]^{p}|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, (\lambda^{\ell}\mathsf{x}.\, t) \cdot \sigma), (\lambda^{\ell}\mathsf{x}.\, t) \cdot \sigma) \mid \sigma \in \mathcal{I}\}$$

# A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $(\!|t|\!)_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\text{tr}, v) \mid t \cdot \sigma \xrightarrow{\text{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$

Semantic inclusions:

$$(\!|[\mathsf{x}]^{p}|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, \sigma(\mathsf{x})), \sigma(\mathsf{x})) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[\lambda^{\ell}\mathsf{x}.\, t]^{p}|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, (\lambda^{\ell}\mathsf{x}.\, t) \cdot \sigma), (\lambda^{\ell}\mathsf{x}.\, t) \cdot \sigma) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[[t_1]^{p_1}\ [t_2]^{p_2}]^{p}|\!)_{\mathcal{I}} \subseteq \left\{ \begin{array}{c} ( \\ \\ , \\ ) \end{array} \middle| \right.$$

# A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $(\!|t|\!)_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\mathrm{tr}, v) \mid t \cdot \sigma \xrightarrow{\mathrm{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$

Semantic inclusions:

$$(\!|[\mathrm{x}]^p|\!)_{\mathcal{I}} = \{(\mathrm{Ret}(\varepsilon, p, \sigma(\mathrm{x})), \sigma(\mathrm{x})) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[\lambda^\ell \mathrm{x}.\, t]^p|\!)_{\mathcal{I}} = \{(\mathrm{Ret}(\varepsilon, p, (\lambda^\ell \mathrm{x}.\, t) \cdot \sigma), (\lambda^\ell \mathrm{x}.\, t) \cdot \sigma) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[[t_1]^{p_1}\ [t_2]^{p_2}]^p|\!)_{\mathcal{I}} \subseteq \left\{ \begin{array}{l} (p \cdot \mathrm{tr}_1 \\ \\ \qquad\qquad\qquad , \\ \ ) \end{array} \middle| \ (\mathrm{tr}_1, \lambda^\ell \mathrm{x}.\, [t_3]^{p_3}) \in (\!|[t_1]^{p_1}|\!)_{\mathcal{I}} \right.$$

# A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $(\!|t|\!)_{\mathcal{I}} \overset{\text{def}}{=} \{(\mathrm{tr}, v) \mid t \cdot \sigma \xrightarrow{\mathrm{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$

Semantic inclusions:

$$(\!|[\mathrm{x}]^{p}|\!)_{\mathcal{I}} = \{(\mathrm{Ret}(\varepsilon, p, \sigma(\mathrm{x})), \sigma(\mathrm{x})) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[\lambda^{\ell}\mathrm{x}.\, t]^{p}|\!)_{\mathcal{I}} = \{(\mathrm{Ret}(\varepsilon, p, (\lambda^{\ell}\mathrm{x}.\, t) \cdot \sigma), (\lambda^{\ell}\mathrm{x}.\, t) \cdot \sigma) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[[t_1]^{p_1}\ [t_2]^{p_2}]^{p}|\!)_{\mathcal{I}} \subseteq \left\{ \begin{array}{l} (p \cdot \mathrm{tr}_1 +\!\!+\ p \cdot \mathrm{tr}_2 \\ \\ \qquad\qquad\qquad , \\ \quad ) \end{array} \middle| \begin{array}{l} (\mathrm{tr}_1, \lambda^{\ell}\mathrm{x}.\, [t_3]^{p_3}) \in (\!|[t_1]^{p_1}|\!)_{\mathcal{I}} \\ \wedge\ (\mathrm{tr}_2, v_2) \in (\!|[t_2]^{p_2}|\!)_{\mathcal{I}} \end{array} \right.$$

# A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $(\!|t|\!)_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\mathrm{tr}, v) \mid t \cdot \sigma \xrightarrow{\mathrm{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$

Semantic inclusions:

$$(\!|[\mathsf{x}]^{p}|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, \sigma(\mathsf{x})), \sigma(\mathsf{x})) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[\lambda^{\ell}\mathsf{x}.\, t]^{p}|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, (\lambda^{\ell}\mathsf{x}.\, t) \cdot \sigma), (\lambda^{\ell}\mathsf{x}.\, t) \cdot \sigma) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[[t_1]^{p_1}\ [t_2]^{p_2}]^{p}|\!)_{\mathcal{I}} \subseteq \left\{ \begin{array}{l} (p \cdot \mathrm{tr}_1 +\!\!+\ p \cdot \mathrm{tr}_2 +\!\!+\ \beta(p, \lambda^{\ell}\mathsf{x}.\, [t_3]^{p_3}, v_2) \\ \\ \qquad\qquad\qquad , \\ \ \ ) \end{array} \middle| \begin{array}{l} (\mathrm{tr}_1, \lambda^{\ell}\mathsf{x}.\, [t_3]^{p_3}) \in (\!|[t_1]^{p_1}|\!)_{\mathcal{I}} \\ \wedge\ (\mathrm{tr}_2, v_2) \in (\!|[t_2]^{p_2}|\!)_{\mathcal{I}} \end{array} \right\}$$

# A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $(\!|t|\!)_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\mathrm{tr}, v) \mid t \cdot \sigma \xrightarrow{\mathrm{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$

Semantic inclusions:

$$(\!|[\mathsf{x}]^p|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, \sigma(\mathsf{x})), \sigma(\mathsf{x})) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[\lambda^\ell \mathsf{x}.\, t]^p|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, (\lambda^\ell \mathsf{x}.\, t) \cdot \sigma), (\lambda^\ell \mathsf{x}.\, t) \cdot \sigma) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[[t_1]^{p_1}\ [t_2]^{p_2}]^p|\!)_{\mathcal{I}} \subseteq \left\{ \begin{array}{l|l} (p \cdot \mathrm{tr}_1 +\!\!+\ p \cdot \mathrm{tr}_2 +\!\!+\ \beta(p, \lambda^\ell \mathsf{x}.\, [t_3]^{p_3}, v_2) +\!\!+ & (\mathrm{tr}_1, \lambda^\ell \mathsf{x}.\, [t_3]^{p_3}) \in (\!|[t_1]^{p_1}|\!)_{\mathcal{I}} \\ p \cdot \mathsf{Call}(\lambda^\ell \mathsf{x}.\, [t_3]^{p_3}, v_2) \cdot \mathrm{tr}_3 & \wedge\ (\mathrm{tr}_2, v_2) \in (\!|[t_2]^{p_2}|\!)_{\mathcal{I}} \\ \qquad\qquad , & \wedge\ (\mathrm{tr}_3, v_3) \in (\!|[t_3]^{p_3}|\!)_{\mathcal{I}[\mathsf{x} \mapsto v_2]} \\ \quad ) & \end{array} \right.$$

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $(\!|t|\!)_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\mathrm{tr}, v) \mid t \cdot \sigma \xrightarrow{\mathrm{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$

Semantic inclusions:

$$(\!|[\mathsf{x}]^p|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, \sigma(\mathsf{x})), \sigma(\mathsf{x})) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[\lambda^\ell \mathsf{x}.\, t]^p|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, (\lambda^\ell \mathsf{x}.\, t) \cdot \sigma), (\lambda^\ell \mathsf{x}.\, t) \cdot \sigma) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[[t_1]^{p_1}\ [t_2]^{p_2}]^p|\!)_{\mathcal{I}} \subseteq \left\{ \begin{array}{l} (p \cdot \mathrm{tr}_1 +\!\!+\ p \cdot \mathrm{tr}_2 +\!\!+\ \beta(p, \lambda^\ell \mathsf{x}.\, [t_3]^{p_3}, v_2) +\!\!+ \\ p \cdot \mathsf{Call}(\lambda^\ell \mathsf{x}.\, [t_3]^{p_3}, v_2) \cdot \mathrm{tr}_3 +\!\!+ \\ \mathsf{Ret}(p, \mathsf{Call}(\lambda^\ell \mathsf{x}.\, [t_3]^{p_3}, v_2), v_3) \\ \qquad , \\ \quad ) \end{array} \middle| \begin{array}{l} (\mathrm{tr}_1, \lambda^\ell \mathsf{x}.\, [t_3]^{p_3}) \in (\!|[t_1]^{p_1}|\!)_{\mathcal{I}} \\ \wedge\ (\mathrm{tr}_2, v_2) \in (\!|[t_2]^{p_2}|\!)_{\mathcal{I}} \\ \wedge\ (\mathrm{tr}_3, v_3) \in (\!|[t_3]^{p_3}|\!)_{\mathcal{I}[\mathsf{x} \mapsto v_2]} \end{array} \right.$$

# A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $(\!|t|\!)_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\text{tr}, v) \mid t \cdot \sigma \xrightarrow{\text{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$

Semantic inclusions:

$$(\!|[\mathsf{x}]^{p}|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, \sigma(\mathsf{x})), \sigma(\mathsf{x})) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[\lambda^{\ell}\mathsf{x}.\, t]^{p}|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, (\lambda^{\ell}\mathsf{x}.\, t) \cdot \sigma), (\lambda^{\ell}\mathsf{x}.\, t) \cdot \sigma) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[[t_1]^{p_1}\ [t_2]^{p_2}]^{p}|\!)_{\mathcal{I}} \subseteq \left\{ \begin{array}{l} (p \cdot \text{tr}_1 +\!\!+\ p \cdot \text{tr}_2 +\!\!+\ \beta(p, \lambda^{\ell}\mathsf{x}.\,[t_3]^{p_3}, v_2) +\!\!+ \\ p \cdot \mathsf{Call}(\lambda^{\ell}\mathsf{x}.\,[t_3]^{p_3}, v_2) \cdot \text{tr}_3 +\!\!+ \\ \mathsf{Ret}(p, \mathsf{Call}(\lambda^{\ell}\mathsf{x}.\,[t_3]^{p_3}, v_2), v_3) \\ \ +\!\!+\ \mathsf{Ret}(\varepsilon, p, v_3), \\ \quad ) \end{array} \middle| \begin{array}{l} (\text{tr}_1, \lambda^{\ell}\mathsf{x}.\,[t_3]^{p_3}) \in (\!|[t_1]^{p_1}|\!)_{\mathcal{I}} \\ \wedge\ (\text{tr}_2, v_2) \in (\!|[t_2]^{p_2}|\!)_{\mathcal{I}} \\ \wedge\ (\text{tr}_3, v_3) \in (\!|[t_3]^{p_3}|\!)_{\mathcal{I}[\mathsf{x} \mapsto v_2]} \end{array} \right\}$$

# A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $\llparenthesis t \rrparenthesis_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\mathrm{tr}, v) \mid t \cdot \sigma \xrightarrow{\mathrm{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$

Semantic inclusions:

$$\llparenthesis [\mathrm{x}]^{p} \rrparenthesis_{\mathcal{I}} = \{(\mathrm{Ret}(\varepsilon, p, \sigma(\mathrm{x})), \sigma(\mathrm{x})) \mid \sigma \in \mathcal{I}\}$$

$$\llparenthesis [\lambda^{\ell}\mathrm{x}.\, t]^{p} \rrparenthesis_{\mathcal{I}} = \{(\mathrm{Ret}(\varepsilon, p, (\lambda^{\ell}\mathrm{x}.\, t) \cdot \sigma), (\lambda^{\ell}\mathrm{x}.\, t) \cdot \sigma) \mid \sigma \in \mathcal{I}\}$$

$$\llparenthesis [[t_1]^{p_1}\ [t_2]^{p_2}]^{p} \rrparenthesis_{\mathcal{I}} \subseteq \left\{ \begin{array}{l} (p \cdot \mathrm{tr}_1 \mathbin{+\!\!+} p \cdot \mathrm{tr}_2 \mathbin{+\!\!+} \beta(p, \lambda^{\ell}\mathrm{x}.\, [t_3]^{p_3}, v_2) \mathbin{+\!\!+} \\ p \cdot \mathrm{Call}(\lambda^{\ell}\mathrm{x}.\, [t_3]^{p_3}, v_2) \cdot \mathrm{tr}_3 \mathbin{+\!\!+} \\ \mathrm{Ret}(p, \mathrm{Call}(\lambda^{\ell}\mathrm{x}.\, [t_3]^{p_3}, v_2), v_3) \\ \ \mathbin{+\!\!+} \mathrm{Ret}(\varepsilon, p, v_3), \\ v_3) \end{array} \middle| \begin{array}{l} (\mathrm{tr}_1, \lambda^{\ell}\mathrm{x}.\, [t_3]^{p_3}) \in \llparenthesis [t_1]^{p_1} \rrparenthesis_{\mathcal{I}} \\ \wedge\ (\mathrm{tr}_2, v_2) \in \llparenthesis [t_2]^{p_2} \rrparenthesis_{\mathcal{I}} \\ \wedge\ (\mathrm{tr}_3, v_3) \in \llparenthesis [t_3]^{p_3} \rrparenthesis_{\mathcal{I}[\mathrm{x} \mapsto v_2]} \end{array} \right.$$

# A Trace-Collecting Semantics

Inputs: $\mathcal{I}$ is a set of value substitutions

Collecting semantics: $(\!|t|\!)_{\mathcal{I}} \stackrel{\text{def}}{=} \{(\text{tr}, v) \mid t \cdot \sigma \xrightarrow{\text{tr}}{}^{*} v,\ \sigma \in \mathcal{I}\}$

Semantic inclusions:

$$(\!|[\mathsf{x}]^{p}|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, \sigma(\mathsf{x})), \sigma(\mathsf{x})) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[\lambda^{\ell}\mathsf{x}.\, t]^{p}|\!)_{\mathcal{I}} = \{(\mathsf{Ret}(\varepsilon, p, (\lambda^{\ell}\mathsf{x}.\, t) \cdot \sigma), (\lambda^{\ell}\mathsf{x}.\, t) \cdot \sigma) \mid \sigma \in \mathcal{I}\}$$

$$(\!|[[t_1]^{p_1}\ [t_2]^{p_2}]^{p}|\!)_{\mathcal{I}} \subseteq \left\{ \begin{array}{l} (p \cdot \text{tr}_1 +\!\!+\ p \cdot \text{tr}_2 +\!\!+\ \beta(p, \lambda^{\ell}\mathsf{x}.\, [t_3]^{p_3}, v_2) +\!\!+ \\ p \cdot \mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, [t_3]^{p_3}, v_2) \cdot \text{tr}_3 +\!\!+ \\ \mathsf{Ret}(p, \mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, [t_3]^{p_3}, v_2), v_3) \\ \ +\!\!+\ \mathsf{Ret}(\varepsilon, p, v_3), \\ v_3) \end{array} \middle| \begin{array}{l} (\text{tr}_1, \lambda^{\ell}\mathsf{x}.\, [t_3]^{p_3}) \in (\!|[t_1]^{p_1}|\!)_{\mathcal{I}} \\ \wedge\ (\text{tr}_2, v_2) \in (\!|[t_2]^{p_2}|\!)_{\mathcal{I}} \\ \wedge\ (\text{tr}_3, v_3) \in (\!|[t_3]^{p_3}|\!)_{\mathcal{I}[\mathsf{x} \mapsto v_2]} \end{array} \right.$$

**Remark:** this is already an over-approximation

- ☞ The relational information between $t_1$ and $t_2$ is lost

The semantic inclusions are further abstracted to recover the $k$-CFA analysis

# Abstractions Towards $k$-CFA

$$\mathcal{I} \to \mathbb{R}(\widehat{\mathsf{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\mathsf{Clos}}) \times \mathbb{C}(\widehat{\mathsf{Clos}})$$

⟵ Family of CFAs (includes $k$-CFA)

$\gamma_{\mathbb{D}}(\phi) \downarrow \uparrow \alpha_{\mathbb{D}}(\phi)$ context projection

$$\mathcal{I} \to \mathbb{R}(\widehat{\mathsf{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\mathsf{Clos}}) \times \mathbb{C}(\widehat{\mathsf{Clos}})$$

$\gamma_{\hat{\mathbb{G}}} \downarrow$ indirection abstraction

$$\mathcal{I} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\mathsf{Clos})) \to \mathbb{R}(\mathsf{Clos}) \times \mathbb{C}(\mathsf{Clos})$$

$\gamma_{\mathsf{clos}} \downarrow$ closure abstraction

$$\mathcal{I} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\mathbb{V})) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$$

$\gamma_{\mathsf{M}} \downarrow \uparrow \alpha_{\mathsf{M}}$ independent attribute for inputs

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$$

$\gamma_{\mathsf{EC}} \downarrow \uparrow \alpha_{\mathsf{EC}}$ independent attribute for maps

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V})) \times \wp(\mathbb{C}(\mathbb{V}))$$

$\gamma_{\mathsf{IA}} \downarrow \uparrow \alpha_{\mathsf{IA}}$ independent attribute for pairs

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}))$$

$\gamma_{\mathsf{NoVal}} \downarrow \uparrow \alpha_{\mathsf{NoVal}}$ output oblivion

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}) \times \mathbb{V})$$

$\gamma_{\mathsf{h}} \downarrow \uparrow \alpha_{\mathsf{h}}$ history oblivion

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$$

$\gamma_{\mathsf{e}} \downarrow \uparrow \alpha_{\mathsf{e}}$ event selection

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$$

⟵ Concrete semantics

# Abstractions Towards $k$-CFA

$$\mathcal{T} \to \mathbb{R}(\widehat{\mathsf{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\mathsf{Clos}}) \times \mathbb{C}(\widehat{\mathsf{Clos}})$$

← Family of CFAs (includes $k$-CFA)

$\gamma_{\mathbb{D}}(\phi)$ ↓ ↑ $\alpha_{\mathbb{D}}(\phi)$ — context projection

$$\mathcal{T} \to \mathbb{R}(\widehat{\mathsf{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\mathsf{Clos}}) \times \mathbb{C}(\widehat{\mathsf{Clos}})$$

$\gamma_{\hat{\mathbb{G}}}$ ↓ — indirection abstraction

$$\mathcal{T} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\mathsf{Clos})) \to \mathbb{R}(\mathsf{Clos}) \times \mathbb{C}(\mathsf{Clos})$$

$\gamma_{\mathsf{clos}}$ ↓ — closure abstraction

$$\mathcal{T} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\mathbb{V})) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$$

$\gamma_{\mathsf{M}}$ ↓ ↑ $\alpha_{\mathsf{M}}$ — independent attribute for inputs

$$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$$

$\gamma_{\mathsf{EC}}$ ↓ ↑ $\alpha_{\mathsf{EC}}$ — independent attribute for maps

$$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V})) \times \wp(\mathbb{C}(\mathbb{V}))$$

$\gamma_{\mathsf{IA}}$ ↓ ↑ $\alpha_{\mathsf{IA}}$ — independent attribute for pairs

$$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}))$$

$\gamma_{\mathsf{NoVal}}$ ↓ ↑ $\alpha_{\mathsf{NoVal}}$ — output oblivion

$$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}) \times \mathbb{V})$$

← Introduction of $\rho$ and $C$

$\gamma_{\mathsf{h}}$ ↓ ↑ $\alpha_{\mathsf{h}}$ — history oblivion

$$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$$

$\gamma_{\mathsf{e}}$ ↓ ↑ $\alpha_{\mathsf{e}}$ — event selection

$$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$$

← Concrete semantics

# Abstractions Towards $k$-CFA

$\mathcal{J} \to \mathbb{R}(\widehat{\mathrm{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\mathrm{Clos}}) \times \mathbb{C}(\widehat{\mathrm{Clos}})$ ⟵ Family of CFAs (includes $k$-CFA)

$\gamma_{\mathbb{D}}(\phi) \downarrow \uparrow \alpha_{\mathbb{D}}(\phi)$ — context projection

$\mathcal{J} \to \mathbb{R}(\widehat{\mathrm{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\mathrm{Clos}}) \times \mathbb{C}(\widehat{\mathrm{Clos}})$

$\gamma_{\hat{\mathbb{G}}} \downarrow$ — indirection abstraction

$\mathcal{J} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\mathrm{Clos})) \to \mathbb{R}(\mathrm{Clos}) \times \mathbb{C}(\mathrm{Clos})$

$\gamma_{\mathrm{clos}} \downarrow$ — closure abstraction

$\mathcal{J} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\mathbb{V})) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$ ⟵ ≈ Big-step abstract evaluator for sets of values

$\gamma_{\mathrm{M}} \downarrow \uparrow \alpha_{\mathrm{M}}$ — independent attribute for inputs

$\mathcal{J} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$

$\gamma_{\mathrm{EC}} \downarrow \uparrow \alpha_{\mathrm{EC}}$ — independent attribute for maps

$\mathcal{J} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V})) \times \wp(\mathbb{C}(\mathbb{V}))$

$\gamma_{\mathrm{IA}} \downarrow \uparrow \alpha_{\mathrm{IA}}$ — independent attribute for pairs

$\mathcal{J} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}))$

$\gamma_{\mathrm{NoVal}} \downarrow \uparrow \alpha_{\mathrm{NoVal}}$ — output oblivion

$\mathcal{J} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}) \times \mathbb{V})$ ⟵ Introduction of $\rho$ and $C$

$\gamma_{\mathrm{h}} \downarrow \uparrow \alpha_{\mathrm{h}}$ — history oblivion

$\mathcal{J} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$

$\gamma_{\mathrm{e}} \downarrow \uparrow \alpha_{\mathrm{e}}$ — event selection

$\mathcal{J} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$ ⟵ Concrete semantics

# Abstractions Towards $k$-CFA

$$\mathcal{I} \to \mathbb{R}(\widehat{\text{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\text{Clos}}) \times \mathbb{C}(\widehat{\text{Clos}})$$

← Family of CFAs (includes $k$-CFA)

$\gamma_{\mathbb{D}}(\phi)$ ↓↑ $\alpha_{\mathbb{D}}(\phi)$ — context projection

$$\mathcal{I} \to \mathbb{R}(\widehat{\text{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\text{Clos}}) \times \mathbb{C}(\widehat{\text{Clos}})$$

$\gamma_{\hat{\mathbb{G}}}$ ↓ — indirection abstraction

$$\mathcal{I} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\text{Clos})) \to \mathbb{R}(\text{Clos}) \times \mathbb{C}(\text{Clos})$$

← Introduction of closures

$\gamma_{\text{clos}}$ ↓ — closure abstraction

$$\mathcal{I} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\mathbb{V})) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$$

← $\approx$ Big-step abstract evaluator for sets of values

$\gamma_{\text{M}}$ ↓↑ $\alpha_{\text{M}}$ — independent attribute for inputs

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$$

$\gamma_{\text{EC}}$ ↓↑ $\alpha_{\text{EC}}$ — independent attribute for maps

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V})) \times \wp(\mathbb{C}(\mathbb{V}))$$

$\gamma_{\text{IA}}$ ↓↑ $\alpha_{\text{IA}}$ — independent attribute for pairs

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}))$$

$\gamma_{\text{NoVal}}$ ↓↑ $\alpha_{\text{NoVal}}$ — output oblivion

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}) \times \mathbb{V})$$

← Introduction of $\rho$ and $C$

$\gamma_{\text{h}}$ ↓↑ $\alpha_{\text{h}}$ — history oblivion

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$$

$\gamma_{\text{e}}$ ↓↑ $\alpha_{\text{e}}$ — event selection

$$\mathcal{I} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$$

← Concrete semantics

# Abstractions Towards $k$-CFA

$\mathcal{T} \to \mathbb{R}(\widehat{\overline{\text{Clos}}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\overline{\text{Clos}}}) \times \mathbb{C}(\widehat{\overline{\text{Clos}}})$ ⟵ Family of CFAs (includes $k$-CFA)

$\gamma_{\mathbb{D}}(\phi)$ / $\alpha_{\mathbb{D}}(\phi)$ — context projection

$\mathcal{T} \to \mathbb{R}(\widehat{\text{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\text{Clos}}) \times \mathbb{C}(\widehat{\text{Clos}})$ ⟵ Introduction of indirections through names

$\gamma_{\hat{\mathbb{G}}}$ — indirection abstraction

$\mathcal{T} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\text{Clos})) \to \mathbb{R}(\text{Clos}) \times \mathbb{C}(\text{Clos})$ ⟵ Introduction of closures

$\gamma_{\text{clos}}$ — closure abstraction

$\mathcal{T} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\mathbb{V})) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$ ⟵ $\approx$ Big-step abstract evaluator for sets of values

$\gamma_{\text{M}}$ / $\alpha_{\text{M}}$ — independent attribute for inputs

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$

$\gamma_{\text{EC}}$ / $\alpha_{\text{EC}}$ — independent attribute for maps

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V})) \times \wp(\mathbb{C}(\mathbb{V}))$

$\gamma_{\text{IA}}$ / $\alpha_{\text{IA}}$ — independent attribute for pairs

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}))$

$\gamma_{\text{NoVal}}$ / $\alpha_{\text{NoVal}}$ — output oblivion

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}) \times \mathbb{V})$ ⟵ Introduction of $\rho$ and $C$

$\gamma_{\text{h}}$ / $\alpha_{\text{h}}$ — history oblivion

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$

$\gamma_{\text{e}}$ / $\alpha_{\text{e}}$ — event selection

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$ ⟵ Concrete semantics

# Abstractions Towards $k$-CFA

$\mathcal{T} \to \mathbb{R}(\widehat{\text{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\text{Clos}}) \times \mathbb{C}(\widehat{\text{Clos}})$ ⟵ Family of CFAs (includes $k$-CFA)

$\gamma_{\mathbb{D}}(\phi)$ ↓ ↑ $\alpha_{\mathbb{D}}(\phi)$ — context projection

$\mathcal{T} \to \mathbb{R}(\widehat{\text{Clos}}) \times (\mathbb{X} \xrightarrow{\text{fin}} (\mathbb{L} \times \mathbb{D})) \to \mathbb{R}(\widehat{\text{Clos}}) \times \mathbb{C}(\widehat{\text{Clos}})$ ⟵ Introduction of indirections through names

$\gamma_{\hat{\mathbb{G}}}$ ↓ — indirection abstraction

$\mathcal{T} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\text{Clos})) \to \mathbb{R}(\text{Clos}) \times \mathbb{C}(\text{Clos})$ ⟵ Introduction of closures

$\gamma_{\text{clos}}$ ↓ — closure abstraction

$\mathcal{T} \to (\mathbb{X} \xrightarrow{\text{fin}} \wp(\mathbb{V})) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$ ⟵ $\approx$ Big-step abstract evaluator for sets of values

$\gamma_{\text{M}}$ ↓ ↑ $\alpha_{\text{M}}$ — independent attribute for inputs

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V})$

$\gamma_{\text{EC}}$ ↓ ↑ $\alpha_{\text{EC}}$ — independent attribute for maps

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V})) \times \wp(\mathbb{C}(\mathbb{V}))$

$\gamma_{\text{IA}}$ ↓ ↑ $\alpha_{\text{IA}}$ — independent attribute for pairs

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}))$

$\gamma_{\text{NoVal}}$ ↓ ↑ $\alpha_{\text{NoVal}}$ — output oblivion

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{R}(\mathbb{V}) \times \mathbb{C}(\mathbb{V}) \times \mathbb{V})$ ⟵ Introduction of $\rho$ and $C$

$\gamma_{\text{h}}$ ↓ ↑ $\alpha_{\text{h}}$ — history oblivion

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$

Main idea: see next slide

$\gamma_{\text{e}}$ ↓ ↑ $\alpha_{\text{e}}$ — event selection

$\mathcal{T} \to \wp(\mathbb{X} \xrightarrow{\text{fin}} \mathbb{V}) \to \wp(\mathbb{T} \times \mathbb{V})$ ⟵ Concrete semantics

# Abstractions: The Main Ideas

$$(\lambda^{\ell}\mathsf{x}.\, t)\, v \xrightarrow{\beta(\varepsilon, \lambda^{\ell}\mathsf{x}.\, t, v)} [t[\mathsf{x} \leftarrow v]]^{\mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, t, v)}$$

# Abstractions: The Main Ideas

Contributes to the environment $\rho$:
$\{v\} \subseteq \rho(\ell, \epsilon)$

$$(\lambda^{\ell}\mathsf{x}.\, t)\; v \xrightarrow{\beta(\varepsilon, \lambda^{\ell}\mathsf{x}.\, t, v)} [t[\mathsf{x} \leftarrow v]]^{\mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, t, v)}$$

# Abstractions: The Main Ideas

Contributes to the environment $\rho$: $\{v\} \subseteq \rho(\ell, \epsilon)$

Contributes to the context (call strings)

$$(\lambda^{\ell}\mathsf{x}.\, t)\ v \xrightarrow{\beta(\varepsilon, \lambda^{\ell}\mathsf{x}.\, t, v)} [t[\mathsf{x} \leftarrow v]]^{\mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, t, v)}$$

# Abstractions: The Main Ideas

Contributes to the environment $\rho$: $\{v\} \subseteq \rho(\ell, \epsilon)$

Contributes to the context (call strings)

$$(\lambda^\ell x.\, t)\; v \xrightarrow{\beta(\varepsilon, \lambda^\ell x.\, t, v)} [t[x \leftarrow v]]^{\mathsf{Call}(\lambda^\ell x.\, t, v)}$$

$$[v]^a \xrightarrow{\mathsf{Ret}(\varepsilon, a, v)} v$$

# Abstractions: The Main Ideas

Contributes to the environment $\rho$: $\{v\} \subseteq \rho(\ell, \epsilon)$

Contributes to the context (call strings)

Contributes to the cache $C$: $\{v\} \subseteq C(a, \epsilon)$

$$(\lambda^{\ell}\mathsf{x}.\, t)\ v \xrightarrow{\beta(\varepsilon, \lambda^{\ell}\mathsf{x}.\, t, v)} [t[\mathsf{x} \leftarrow v]]^{\mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, t, v)} \qquad [v]^{a} \xrightarrow{\mathsf{Ret}(\varepsilon, a, v)} v$$

# Abstractions: The Main Ideas

Contributes to the environment $\rho$: $\{v\} \subseteq \rho(\ell, \epsilon)$

Contributes to the context (call strings)

Contributes to the cache $C$: $\{v\} \subseteq C(a, \epsilon)$

$$(\lambda^{\ell}\mathsf{x}.\, t)\; v \xrightarrow{\beta(\varepsilon, \lambda^{\ell}\mathsf{x}.\, t, v)} [t[\mathsf{x} \leftarrow v]]^{\mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, t, v)} \qquad [v]^{a} \xrightarrow{\mathsf{Ret}(\varepsilon, a, v)} v$$

Example:

$$t \xrightarrow{\beta(\delta_1, \lambda^{\ell_1}\mathsf{x}.\, t, v_1) +\!\!+ \beta(\delta_1, \lambda^{\ell_1}\mathsf{x}.\, t, v_1') +\!\!+ \mathsf{Ret}(\delta_2, p_2, v_2) +\!\!+ \mathsf{Ret}(\delta_2, p_2', v_2')}{}^{*} u$$

# Abstractions: The Main Ideas

Contributes to the environment $\rho$: $\{v\} \subseteq \rho(\ell, \epsilon)$

Contributes to the context (call strings)

Contributes to the cache $C$: $\{v\} \subseteq C(a, \epsilon)$

$$(\lambda^\ell x.\, t)\; v \xrightarrow{\beta(\varepsilon, \lambda^\ell x.\, t, v)} [t[x \leftarrow v]]^{\mathsf{Call}(\lambda^\ell x.\, t, v)} \qquad [v]^a \xrightarrow{\mathsf{Ret}(\varepsilon, a, v)} v$$

Example:

$$t \xrightarrow{\beta(\delta_1, \lambda^{\ell_1} x.\, t, v_1) \mathbin{+\!\!+} \beta(\delta_1, \lambda^{\ell_1} x.\, t, v_1') \mathbin{+\!\!+} \mathsf{Ret}(\delta_2, p_2, v_2) \mathbin{+\!\!+} \mathsf{Ret}(\delta_2, p_2', v_2')}{}^{*} u$$

$$\rho = \{(\ell_1, \delta_1) \mapsto \{v_1, v_1'\}\}$$

# Abstractions: The Main Ideas

Contributes to the environment $\rho$: $\{v\} \subseteq \rho(\ell, \epsilon)$

Contributes to the context (call strings)

Contributes to the cache $C$: $\{v\} \subseteq C(a, \epsilon)$

$$(\lambda^{\ell}\mathsf{x}.\, t)\; v \xrightarrow{\beta(\varepsilon, \lambda^{\ell}\mathsf{x}.\, t, v)} [t[\mathsf{x} \leftarrow v]]^{\mathsf{Call}(\lambda^{\ell}\mathsf{x}.\, t, v)} \qquad [v]^{a} \xrightarrow{\mathsf{Ret}(\varepsilon, a, v)} v$$

Example:

$$t \xrightarrow{\beta(\delta_1, \lambda^{\ell_1}\mathsf{x}.\, t, v_1) +\!\!+\, \beta(\delta_1, \lambda^{\ell_1}\mathsf{x}.\, t, v_1') +\!\!+\, \mathsf{Ret}(\delta_2, p_2, v_2) +\!\!+\, \mathsf{Ret}(\delta_2, p_2', v_2')}{}^{*} u$$

$$\rho = \{(\ell_1, \delta_1) \mapsto \{v_1, v_1'\}\} \qquad C = \{(p_2, \delta_2) \mapsto \{v_2\}, (p_2', \delta_2) \mapsto \{v_2'\}\}$$

# A New Look at Constraint-Based CFA

# Constraints for CFA: A Completely Different Line of Work!

**Main idea:** a term $[t]^p$ defines constraints for its CFA, that $\hat{\rho}$ and $\hat{C}$ should satisfy

Then, just solve those constraints the way the want! (F. Nielson, H. R. Nielson and Hankin 1999)

# Constraints for CFA: A Completely Different Line of Work!

**Main idea:** a term $[t]^p$ defines constraints for its CFA, that $\hat{\rho}$ and $\hat{C}$ should satisfy
Then, just solve those constraints the way the want! (F. Nielson, H. R. Nielson and Hankin 1999)

$(\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [t]^p$ means: « In the environment $\hat{\Gamma}$, and the calling context $\delta$, $\hat{\rho}$ and $\hat{C}$ are valid solutions for the CFA of $[t]^p$ »

# Constraints for CFA: A Completely Different Line of Work!

**Main idea:** a term $[t]^p$ defines constraints for its CFA, that $\hat{\rho}$ and $\hat{C}$ should satisfy
Then, just solve those constraints the way the want! (F. Nielson, H. R. Nielson and Hankin 1999)

$(\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [t]^p$ means: « In the environment $\hat{\Gamma}$, and the calling context $\delta$, $\hat{\rho}$ and $\hat{C}$ are valid solutions for the CFA of $[t]^p$ »

$$\frac{\hat{\rho}(\hat{\Gamma}(\mathrm{x})) \subseteq \hat{C}(p, \delta)}{(\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [\mathrm{x}]^p}$$

# Constraints for CFA: A Completely Different Line of Work!

**Main idea:** a term $[t]^p$ defines constraints for its CFA, that $\hat{\rho}$ and $\hat{C}$ should satisfy
Then, just solve those constraints the way the want! (F. Nielson, H. R. Nielson and Hankin 1999)

$(\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [t]^p$ means: « In the environment $\hat{\Gamma}$, and the calling context $\delta$, $\hat{\rho}$ and $\hat{C}$ are valid solutions for the CFA of $[t]^p$ »

$$\frac{\hat{\rho}(\hat{\Gamma}(\mathrm{x})) \subseteq \hat{C}(p, \delta)}{(\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [\mathrm{x}]^p} \qquad \frac{\left\{\langle \lambda^{\ell}\mathrm{x}.\, t, \hat{\Gamma}|_{\mathrm{fv}(\lambda^{\ell}\mathrm{x}.\, t)} \rangle\right\} \subseteq \hat{C}(p, \delta)}{(\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [\lambda^{\ell}\mathrm{x}.\, t]^p}$$

# Constraints for CFA: A Completely Different Line of Work!

**Main idea:** a term $[t]^p$ defines constraints for its CFA, that $\hat{\rho}$ and $\hat{C}$ should satisfy
Then, just solve those constraints the way the want! (F. Nielson, H. R. Nielson and Hankin 1999)

$(\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [t]^p$ means: « In the environment $\hat{\Gamma}$, and the calling context $\delta$, $\hat{\rho}$ and $\hat{C}$ are valid solutions for the CFA of $[t]^p$ »

$$\frac{\hat{\rho}(\hat{\Gamma}(\mathrm{x})) \subseteq \hat{C}(p, \delta)}{(\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [\mathrm{x}]^p} \qquad \frac{\left\{\langle \lambda^{\ell}\mathrm{x}.\, t, \hat{\Gamma}|_{\mathrm{fv}(\lambda^{\ell}\mathrm{x}.\, t)} \rangle\right\} \subseteq \hat{C}(p, \delta)}{(\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [\lambda^{\ell}\mathrm{x}.\, t]^p}$$

$$\frac{\begin{array}{c} (\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [t_1]^{p_1} \qquad (\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [t_2]^{p_2} \\ \forall \langle \lambda^{\ell}\mathrm{x}.\, [t_3]^{p_3}, \hat{\Gamma}_3 \rangle \in \hat{C}_1(p_1, \delta), \\ (\hat{\rho}, \hat{C}) \vdash_{\delta p}^{\hat{\Gamma}, \mathrm{x}:(\ell, \delta p)} [t_3]^{p_3} \wedge \hat{C}(p_2, \delta) \subseteq \hat{\rho}(\ell, \delta p) \wedge \hat{C}(p_3, \delta p) \subseteq \hat{C}(p, \delta) \end{array}}{(\hat{\rho}, \hat{C}) \vdash_{\delta}^{\hat{\Gamma}} [[t_1]^{p_1}\ [t_2]^{p_2}]^p}$$

# Constraints for CFA: A Completely Different Line of Work!

**Main idea:** a term $[t]^p$ defines constraints for its CFA, that $\hat{\rho}$ and $\hat{C}$ should satisfy
Then, just solve those constraints the way the want! (F. Nielson, H. R. Nielson and Hankin 1999)

$(\hat{\rho}, \hat{C}) \vdash_\delta^{\hat{\Gamma}} [t]^p$ means: « In the environment $\hat{\Gamma}$, and the calling context $\delta$, $\hat{\rho}$ and $\hat{C}$ are valid solutions for the CFA of $[t]^p$ »

$$\frac{\hat{\rho}(\hat{\Gamma}(\mathrm{x})) \subseteq \hat{C}(p, \delta)}{(\hat{\rho}, \hat{C}) \vdash_\delta^{\hat{\Gamma}} [\mathrm{x}]^p} \qquad \frac{\left\{\langle \lambda^\ell \mathrm{x}.\, t, \hat{\Gamma}|_{\mathrm{fv}(\lambda^\ell \mathrm{x}.\, t)}\rangle\right\} \subseteq \hat{C}(p, \delta)}{(\hat{\rho}, \hat{C}) \vdash_\delta^{\hat{\Gamma}} [\lambda^\ell \mathrm{x}.\, t]^p}$$

$$\frac{\begin{array}{c} (\hat{\rho}, \hat{C}) \vdash_\delta^{\hat{\Gamma}} [t_1]^{p_1} \qquad (\hat{\rho}, \hat{C}) \vdash_\delta^{\hat{\Gamma}} [t_2]^{p_2} \\ \forall \langle \lambda^\ell \mathrm{x}.\, [t_3]^{p_3}, \hat{\Gamma}_3\rangle \in \hat{C}_1(p_1, \delta), \\ (\hat{\rho}, \hat{C}) \vdash_{\delta p}^{\hat{\Gamma}, \mathrm{x}:(\ell, \delta p)} [t_3]^{p_3} \wedge \hat{C}(p_2, \delta) \subseteq \hat{\rho}(\ell, \delta p) \wedge \hat{C}(p_3, \delta p) \subseteq \hat{C}(p, \delta) \end{array}}{(\hat{\rho}, \hat{C}) \vdash_\delta^{\hat{\Gamma}} [[t_1]^{p_1}\ [t_2]^{p_2}]^p}$$

Soundness usually proved by subject reduction

# A Novel Justification for Constraint-Based CFA

$(\hat{\rho}, \hat{C}) \models_{\delta}^{\hat{\Gamma}} [t]^p$ defined **semantically**: « $(\hat{\rho}, \hat{C})$ approximates the events of the trace »

# A Novel Justification for Constraint-Based CFA

$(\hat{\rho}, \hat{C}) \models_{\delta}^{\hat{\Gamma}} [t]^p$ defined **semantically**: « $(\hat{\rho}, \hat{C})$ approximates the events of the trace »

$$(\hat{\rho}, \hat{C}) \models_{\delta}^{\hat{\Gamma}} [t]^p \quad \stackrel{\text{def}}{=} \quad \delta \cdot A_{\mathsf{EC}}(\!|[t]^p|\!)_{\gamma_{\hat{\mathbb{G}}}(\hat{\rho},\hat{\Gamma})} \mathrel{\dot{\subseteq}}^2 \gamma_{\mathsf{clos}} \circ \gamma_{\mathsf{Ind}}^{\hat{\rho}}(\hat{\rho}, \hat{C})$$

# A Novel Justification for Constraint-Based CFA

$(\hat{\rho}, \hat{C}) \models^{\hat{\Gamma}}_{\delta} [t]^p$ defined **semantically**: « $(\hat{\rho}, \hat{C})$ approximates the events of the trace »

$$(\hat{\rho}, \hat{C}) \models^{\hat{\Gamma}}_{\delta} [t]^p \quad \stackrel{\text{def}}{=} \quad \delta \cdot A_{\mathsf{EC}}(\![ [t]^p ]\!)_{\gamma_{\hat{\mathbb{G}}}(\hat{\rho}, \hat{\Gamma})} \mathrel{\dot{\subseteq}^2} \gamma_{\mathsf{clos}} \circ \gamma^{\hat{\rho}}_{\mathsf{Ind}}(\hat{\rho}, \hat{C})$$

The *same* rules become actual theorems:

$$\hat{\rho}(\hat{\Gamma}(\mathsf{x})) \subseteq \hat{C}(p, \delta) \quad \Longrightarrow \quad (\hat{\rho}, \hat{C}) \models^{\hat{\Gamma}}_{\delta} [\mathsf{x}]^p$$

$$\left\{ \langle \lambda^\ell \mathsf{x}.\, t, \hat{\Gamma}|_{\mathrm{fv}(\lambda^\ell \mathsf{x}.\, t)} \rangle \right\} \subseteq \hat{C}(p, \delta) \quad \Longrightarrow \quad (\hat{\rho}, \hat{C}) \models^{\hat{\Gamma}}_{\delta} [\lambda^\ell \mathsf{x}.\, t]^p$$

$$\left.\begin{array}{c} (\hat{\rho}, \hat{C}) \models^{\hat{\Gamma}}_{\delta} [t_1]^{p_1} \quad \wedge \quad (\hat{\rho}, \hat{C}) \models^{\hat{\Gamma}}_{\delta} [t_2]^{p_2} \\ \wedge \forall \langle \lambda^\ell \mathsf{x}.\, [t_3]^{p_3}, \hat{\Gamma}_3 \rangle \in \hat{C}_1(p_1, \delta), \\ (\hat{\rho}, \hat{C}) \models^{\hat{\Gamma}, \mathsf{x}:(\ell, \delta p)}_{\delta p} [t_3]^{p_3} \wedge \hat{C}(p_2, \delta) \subseteq \hat{\rho}(\ell, \delta p) \wedge \hat{C}(p_3, \delta p) \subseteq \hat{C}(p, \delta) \end{array}\right\} \Longrightarrow (\hat{\rho}, \hat{C}) \models^{\hat{\Gamma}}_{\delta} [[t_1]^{p_1}\, [t_2]^{p_2}]^p$$

# A Novel Justification for Constraint-Based CFA

$(\hat\rho, \hat C) \models^{\hat\Gamma}_{\delta} [t]^p$ defined **semantically**: « $(\hat\rho, \hat C)$ approximates the events of the trace »

$$(\hat\rho, \hat C) \models^{\hat\Gamma}_{\delta} [t]^p \quad \stackrel{\text{def}}{=} \quad \delta \cdot A_{\mathsf{EC}}(\![ [t]^p ]\!)_{\gamma_{\hat{\mathbb{G}}}(\hat\rho, \hat\Gamma)} \mathrel{\dot\subseteq}^2 \gamma_{\mathsf{clos}} \circ \gamma^{\hat\rho}_{\mathsf{Ind}}(\hat\rho, \hat C)$$

The *same* rules become actual theorems:

$$\hat\rho(\hat\Gamma(\mathrm{x})) \subseteq \hat C(p, \delta) \implies (\hat\rho, \hat C) \models^{\hat\Gamma}_{\delta} [\mathrm{x}]^p$$

$$\left\{\langle \lambda^\ell \mathrm{x}.\, t, \hat\Gamma|_{\mathrm{fv}(\lambda^\ell \mathrm{x}.\, t)}\rangle\right\} \subseteq \hat C(p, \delta) \implies (\hat\rho, \hat C) \models^{\hat\Gamma}_{\delta} [\lambda^\ell \mathrm{x}.\, t]^p$$

$$\left.\begin{array}{l} (\hat\rho, \hat C) \models^{\hat\Gamma}_{\delta} [t_1]^{p_1} \quad \wedge \quad (\hat\rho, \hat C) \models^{\hat\Gamma}_{\delta} [t_2]^{p_2} \\ \wedge \forall \langle \lambda^\ell \mathrm{x}.\, [t_3]^{p_3}, \hat\Gamma_3\rangle \in \hat C_1(p_1, \delta), \\ (\hat\rho, \hat C) \models^{\hat\Gamma, \mathrm{x}:(\ell, \delta p)}_{\delta p} [t_3]^{p_3} \wedge \hat C(p_2, \delta) \subseteq \hat\rho(\ell, \delta p) \wedge \hat C(p_3, \delta p) \subseteq \hat C(p, \delta) \end{array}\right\} \implies (\hat\rho, \hat C) \models^{\hat\Gamma}_{\delta} [[t_1]^{p_1}\, [t_2]^{p_2}]^p$$

Each rule is proved sound separately:

- 👍 Modular proof
- 👍 Easier to extend or adapt to new language features

# Semantic Soundness for Constraint-Based CFA

**Theorem**

*Assume* $(\hat{\rho}, \hat{C}) \models_{\varepsilon}^{\{\}} [t_0]^{p_0}$ *and* $[t_0]^{p_0} \xrightarrow{\text{tr}}{}^{*} v_0$ *and* $t_0$ *is a closed program.*
*Then:*

# Semantic Soundness for Constraint-Based CFA

**Theorem**

*Assume* $(\hat{\rho}, \hat{C}) \vDash_{\varepsilon}^{\{\}} [t_0]^{p_0}$ *and* $[t_0]^{p_0} \xrightarrow{\mathrm{tr}}{}^{*} v_0$ *and* $t_0$ *is a closed program.*
*Then:*

- *Soundness for* $\hat{\rho}$*:*
  *If* $\beta(\pi, \lambda^{\ell}\mathrm{x}.\, t, v) \in \mathrm{tr}$*, then:*
  *there exists an abstract closure* $\hat{c} \in \hat{\rho}(\ell, \mathrm{calls}(\pi))$ *such that* $v \in \gamma^{\hat{\rho}}_{\mathsf{clnd}}\{\hat{c}\}$

# Semantic Soundness for Constraint-Based CFA

**Theorem**

*Assume* $(\hat{\rho}, \hat{C}) \models_{\varepsilon}^{\{\}} [t_0]^{p_0}$ *and* $[t_0]^{p_0} \xrightarrow{\mathrm{tr}}{}^{*} v_0$ *and* $t_0$ *is a closed program.*
*Then:*

- *Soundness for* $\hat{\rho}$*:*
  *If* $\beta(\pi, \lambda^{\ell}\mathrm{x}.\, t, v) \in \mathrm{tr}$*, then:*
  *there exists an abstract closure* $\hat{c} \in \hat{\rho}(\ell, \mathrm{calls}(\pi))$ *such that* $v \in \gamma^{\hat{\rho}}_{\mathsf{clnd}}\{\hat{c}\}$
- *Soundness for* $\hat{C}$*:*
  *If* $\mathrm{Ret}(\pi, p, v) \in \mathrm{tr}$*, then:*
  *there exists an abstract closure* $\hat{c} \in \hat{C}(p, \mathrm{calls}(\pi))$ *such that* $v \in \gamma^{\hat{\rho}}_{\mathsf{clnd}}\{\hat{c}\}$

## Semantic Soundness for Constraint-Based CFA

**Theorem**

*Assume* $(\hat{\rho}, \hat{C}) \models_{\varepsilon}^{\{\}} [t_0]^{p_0}$ *and* $[t_0]^{p_0} \xrightarrow{\mathrm{tr}}{}^{*} v_0$ *and* $t_0$ *is a closed program.*
*Then:*

- *Soundness for* $\hat{\rho}$*:*
  *If* $\beta(\pi, \lambda^{\ell}\mathrm{x}.\, t, v) \in \mathrm{tr}$*, then:*
  *there exists an abstract closure* $\hat{c} \in \hat{\rho}(\ell, \mathrm{calls}(\pi))$ *such that* $v \in \gamma^{\hat{\rho}}_{\mathsf{clnd}}\{\hat{c}\}$
- *Soundness for* $\hat{C}$*:*
  *If* $\mathsf{Ret}(\pi, p, v) \in \mathrm{tr}$*, then:*
  *there exists an abstract closure* $\hat{c} \in \hat{C}(p, \mathrm{calls}(\pi))$ *such that* $v \in \gamma^{\hat{\rho}}_{\mathsf{clnd}}\{\hat{c}\}$

Stated here for $\infty$-CFA for the sake of simplicity

A similar statement holds for $k$-CFA and more generally for the $\phi$-CFA family

# Beyond $k$-CFA

# $k^\star$: A New Strategy for Context Sensitivity

- Standard $k$-CFA: limits call sites to the $k$ most recent ones
  - An issue: $k$ is chosen *statically*
  - It is hard to predict how deep in the stack it is necessary to look to retain precise analysis results

# $k^\star$: A New Strategy for Context Sensitivity

- Standard $k$-CFA: limits call sites to the $k$ most recent ones
  - An issue: $k$ is chosen *statically*
  - ☞ It is hard to predict how deep in the stack it is necessary to look to retain precise analysis results
- New strategy $k^\star$: limits call sites to at most $k$ *repetitions*
  - The depth of the stack is *not* statically bounded
  - Still, the number of call strings is finite
  - ☞ Loses information only for recursively called call sites
  - Cost is similar to $k$-CFA in practice
  - See table on next slide

# Evaluation on classic micro-benchmarks: $0$-CFA vs. $1$-CFA vs. $1^\star$-CFA

| Program | $\mathbf{0}$-CFA | | | | $\mathbf{1}$-CFA | | | | $\mathbf{1}^*$-CFA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | States | Edges | Iter | Result | States | Edges | Iter | Result | States | Edges | Iter | Result |
| ack | 35 | 54 | 3 | = | 107 | 183 | 4 | B | 185 | 321 | 5 | = |
| ack_cps | 54 | 78 | 5 | = | 161 | 250 | 7 | B | 187 | 302 | 7 | = |
| binomial | 37 | 59 | 3 | = | 99 | 161 | 3 | B | 151 | 263 | 4 | = |
| blur | 42 | 60 | 6 | ⊖ | 64 | 81 | 10 | B° | 93 | 128 | 8 | =° |
| church | 117 | 251 | 17 | ⊖ | 269 | 427 | 21 | B | 322 | 446 | 25 | ⊖ |
| delta_delta | 6 | 5 | 2 | =° | 8 | 7 | 3 | B° | 10 | 9 | 4 | =° |
| eta | 11 | 12 | 3 | ⊖ | 14 | 14 | 3 | B° | 14 | 14 | 3 | =° |
| facehugger | 37 | 50 | 4 | = | 58 | 74 | 5 | B | 76 | 102 | 6 | = |
| fact | 15 | 19 | 3 | = | 25 | 33 | 4 | B | 35 | 47 | 4 | = |
| fact_cps | 30 | 37 | 7 | = | 60 | 71 | 7 | B | 65 | 83 | 7 | = |
| fact_tailrec | 24 | 30 | 5 | = | 37 | 47 | 5 | B | 49 | 64 | 5 | = |
| hmca100 | 1307 | 11902 | 4 | ⊖ | 1605 | 2002 | 4 | B° | 1605 | 2002 | 4 | =° |
| hmca200 | 2607 | 43802 | 4 | ⊖ | 3205 | 4002 | 4 | B° | 3205 | 4002 | 4 | =° |
| hmca300 | 3907 | 95702 | 4 | ⊖ | 4805 | 6002 | 4 | B° | 4805 | 6002 | 4 | =° |
| kcfa2 | 28 | 36 | 5 | = | 54 | 75 | 5 | B | 91 | 97 | 5 | ⊕° |
| kcfa3 | 36 | 45 | 6 | = | 79 | 119 | 6 | B | 167 | 173 | 6 | ⊕° |
| mc91 | 16 | 22 | 3 | = | 37 | 56 | 4 | B | 59 | 90 | 5 | = |
| mc91_cps | 31 | 40 | 5 | = | 70 | 96 | 8 | B | 68 | 94 | 6 | = |
| mc91_tailrec | 34 | 47 | 5 | = | 74 | 102 | 5 | B | 116 | 181 | 5 | = |
| mj09 | 28 | 31 | 7 | = | 42 | 44 | 7 | B | 44 | 44 | 7 | = |
| sat | 55 | 76 | 11 | = | 156 | 278 | 14 | B | 102 | 113 | 11 | ⊕° |
| sat3 | 49 | 68 | 10 | = | 105 | 160 | 12 | B | 86 | 94 | 10 | ⊕° |
| shivers | 8 | 7 | 3 | =° | 11 | 10 | 4 | B° | 14 | 13 | 5 | =° |
| shivers2 | 31 | 37 | 6 | =° | 49 | 55 | 8 | B° | 49 | 53 | 9 | =° |
| tak | 35 | 59 | 3 | = | 137 | 256 | 4 | B | 241 | 448 | 5 | = |
| tak_cps | 62 | 92 | 7 | = | 227 | 368 | 12 | B | 136 | 222 | 9 | = |
| tak_4d | 49 | 88 | 3 | = | 231 | 464 | 4 | B | 421 | 834 | 5 | = |

# ∇CFA: Introducing Widening in CFA

- In CFA: two sources of divergence
  - An infinite number of contexts (call stacks)
  - An infinite number of produced values (closures)
- In $k$-CFA: use of finite domains to limit both (last $k$ call sites)

☞ Ensures the convergence of the analysis

☞ Precludes the use of expressive numeric domains (intervals...)

# ∇CFA: Introducing Widening in CFA

- In CFA: two sources of divergence
  - An infinite number of contexts (call stacks)
  - An infinite number of produced values (closures)
- In $k$-CFA: use of finite domains to limit both (last $k$ call sites)

  ☞ Ensures the convergence of the analysis

  ☞ Precludes the use of expressive numeric domains (intervals...)

- **A new point in the design space:** ∇CFA
  - ★ Finite contexts: parameterised by the strategy on contexts ($k$, $k^\star$, ...)
  - ★ An infinite domain for values
    - ☞ A recursive (unbounded) abstract domain for closures
  - ★ Use of widening to ensure termination
    - ☞ Integrates well with expressive abstract domains for integers!
- Experimental results are promising

# The Abstract Domain for ∇CFA

A recursively defined abstract domain:

Abstract closures

$$\mathsf{Clos}^\sharp \stackrel{\text{def}}{=} (\mathbb{V} \xrightarrow{\text{fin}} \mathbb{G}^\sharp) + \{\top\}$$

Abstract environments

$$\mathbb{G}^\sharp \stackrel{\text{def}}{=} \mathbb{X} \xrightarrow{\text{fin}} \mathsf{Clos}^\sharp$$

# The Abstract Domain for ∇CFA

A recursively defined abstract domain:

Abstract closures

$$\mathrm{Clos}^\sharp \stackrel{\text{def}}{=} (\mathbb{V} \xrightarrow{\text{fin}} \mathbb{G}^\sharp) + \{\top\} \qquad \mathbb{G}^\sharp \stackrel{\text{def}}{=} \mathbb{X} \xrightarrow{\text{fin}} \mathrm{Clos}^\sharp$$

Finite disjunction of code + environment

Any code

Abstract environments

# The Abstract Domain for ∇CFA

A recursively defined abstract domain:

$$\mathrm{Clos}^\sharp \overset{\text{def}}{=} (\mathbb{V} \xrightarrow{\text{fin}} \mathbb{G}^\sharp) + \{\top\} \qquad \mathbb{G}^\sharp \overset{\text{def}}{=} \mathbb{X} \xrightarrow{\text{fin}} \mathrm{Clos}^\sharp$$

Abstract closures

Finite disjunction of code + environment

Any code

Abstract environments

⚠ The abstract values *cannot* express recursively defined solutions

# The Abstract Domain for ∇CFA

A recursively defined abstract domain:

$$\mathsf{Clos}^\sharp \stackrel{\text{def}}{=} (\mathbb{V} \xrightarrow{\text{fin}} \mathbb{G}^\sharp) + \{\mathsf{T}\} \qquad \mathbb{G}^\sharp \stackrel{\text{def}}{=} \mathbb{X} \xrightarrow{\text{fin}} \mathsf{Clos}^\sharp$$



- ⚠ The abstract values *cannot* express recursively defined solutions
- ☞ In standard CFA: *indirections* are used to express cyclic solutions

# The Abstract Domain for ∇CFA

A recursively defined abstract domain:

$$\mathrm{Clos}^\sharp \stackrel{\text{def}}{=} (\mathbb{V} \xrightarrow{\text{fin}} \mathbb{G}^\sharp) + \{\top\} \qquad \mathbb{G}^\sharp \stackrel{\text{def}}{=} \mathbb{X} \xrightarrow{\text{fin}} \mathrm{Clos}^\sharp$$

Abstract closures

Finite disjunction of code + environment

Any code

Abstract environments

- ⚠ The abstract values *cannot* express recursively defined solutions
- ☞ In standard CFA: *indirections* are used to express cyclic solutions

Widening: union + truncation

The height of $\Gamma_1^\sharp \nabla_{\mathbb{G}}^\sharp \Gamma_2^\sharp$ is no greater than the height of $\Gamma_1^\sharp$

- ☞ The parts that are too high are replaced with ⊤
- ☞ A violent approximation, that remains precise enough for most examples

# ∇CFA: Definition

$$\text{analyzer} : (\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp$$

Environment

Abstract value for the result

Call string

Program

# ∇CFA: Definition

Input widening

Environment

Abstract value
for the result

$$\text{analyzer} : (\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp$$

Call string

Program

# ∇CFA: Definition

$$\text{analyzer} : (\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp \stackrel{\text{def}}{=} \mu(\text{analyze})$$

Input widening

Environment

Abstract value for the result

Call string

Program

Top-down fixpoint solver

# ∇CFA: Definition

Input widening

Environment

Abstract value for the result

$$\text{analyzer} : (\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp \stackrel{\text{def}}{=} \mu(\text{analyze})$$

Top-down fixpoint solver

Call string

Program

Open recursion

$$\text{analyze} : ((\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp) \to ((\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp)$$

# ∇CFA: Definition

Input widening
Environment
Abstract value for the result
Call string
Program
Top-down fixpoint solver
Open recursion

$$\text{analyzer} : (\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp \stackrel{\text{def}}{=} \mu(\text{analyze})$$

$$\text{analyze} : ((\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp) \to ((\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp)$$

$$\text{analyze}(\text{analyze}_{\mathsf{rec}})(\text{iw}, \delta, \Gamma^\sharp, [\mathsf{x}]^p) \stackrel{\text{def}}{=} \Gamma^\sharp(\mathsf{x})$$

# ∇CFA: Definition

Input widening

Environment

Abstract value for the result

$$\text{analyzer} : (\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp \stackrel{\text{def}}{=} \mu(\text{analyze})$$

Top-down fixpoint solver

Call string

Program

Open recursion

$$\text{analyze} : ((\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp) \to ((\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp)$$

$$\text{analyze}(\text{analyze}_{\mathsf{rec}})(\text{iw}, \delta, \Gamma^\sharp, [\mathsf{x}]^p) \stackrel{\text{def}}{=} \Gamma^\sharp(\mathsf{x})$$

$$\text{analyze}(\text{analyze}_{\mathsf{rec}})(\text{iw}, \delta, \Gamma^\sharp, [\lambda^\ell \mathsf{x}.\, t]^p) \stackrel{\text{def}}{=} \{\langle \lambda^\ell \mathsf{x}.\, t, \Gamma^\sharp|_{\text{fv}(\lambda^\ell \mathsf{x}.\, t)} \rangle\}$$

# ∇CFA: Definition

Input widening · Environment · Abstract value for the result · Top-down fixpoint solver · Call string · Program · Open recursion

$$\text{analyzer} : (\text{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp \stackrel{\text{def}}{=} \mu(\text{analyze})$$

$$\text{analyze} : ((\text{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp) \to ((\text{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp)$$

$$\text{analyze}(\text{analyze}_{\mathsf{rec}})(\text{iw}, \delta, \Gamma^\sharp, [\mathsf{x}]^p) \stackrel{\text{def}}{=} \Gamma^\sharp(\mathsf{x})$$

$$\text{analyze}(\text{analyze}_{\mathsf{rec}})(\text{iw}, \delta, \Gamma^\sharp, [\lambda^\ell \mathsf{x}.\, t]^p) \stackrel{\text{def}}{=} \{\langle \lambda^\ell \mathsf{x}.\, t, \Gamma^\sharp|_{\text{fv}(\lambda^\ell \mathsf{x}.\, t)}\rangle\}$$

$$\text{analyze}(\text{analyze}_{\mathsf{rec}})(\text{iw}, \delta, \Gamma^\sharp, [[t_1]^{p_1}\ [t_2]^{p_2}]^p) \stackrel{\text{def}}{=}$$
$$\mathsf{let}\ V_1 = \text{analyze}_{\mathsf{rec}}(\mathsf{false}, \delta, \Gamma^\sharp, [t_1]^{p_1})\ \mathsf{in}$$
$$\mathsf{if}\ V_1 = \top\ \mathsf{then}\ \top\ \mathsf{else}$$
$$\mathsf{let}\ V_2 = \text{analyze}_{\mathsf{rec}}(\mathsf{false}, \delta, \Gamma^\sharp, [t_2]^{p_2})\ \mathsf{in}$$
$$\mathsf{let}\ \text{iw}' = \text{maximal}(\phi(\delta p))\ \mathsf{in}$$
$$\bigcup^{\mathsf{clos}^\sharp}_{\langle \lambda^\ell \mathsf{x}.\, [t_3]^{p_3}, \Gamma_3^\sharp\rangle \in V_1} \text{analyze}_{\mathsf{rec}}(\text{iw}', \phi(\delta p), (\Gamma_3^\sharp, \mathsf{x} : V_2), [t_3]^{p_3})$$

# ∇CFA: Definition

Input widening

Environment

Abstract value for the result

$$\text{analyzer} : (\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp \stackrel{\text{def}}{=} \mu(\text{analyze})$$

Top-down fixpoint solver

Call string

Program

Open recursion

$$\text{analyze} : ((\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp) \to ((\mathsf{bool} \times \mathbb{D} \times \mathbb{G}^\sharp \times \mathcal{T}) \to \mathsf{Clos}^\sharp)$$

$$\text{analyze}(\text{analyze}_{\mathsf{rec}})(\text{iw}, \delta, \Gamma^\sharp, [\mathsf{x}]^p) \stackrel{\text{def}}{=} \Gamma^\sharp(\mathsf{x})$$

$$\text{analyze}(\text{analyze}_{\mathsf{rec}})(\text{iw}, \delta, \Gamma^\sharp, [\lambda^\ell \mathsf{x}.\, t]^p) \stackrel{\text{def}}{=} \{\langle \lambda^\ell \mathsf{x}.\, t, \Gamma^\sharp|_{\text{fv}(\lambda^\ell \mathsf{x}.\, t)} \rangle\}$$

$$\text{analyze}(\text{analyze}_{\mathsf{rec}})(\text{iw}, \delta, \Gamma^\sharp, [[t_1]^{p_1}\ [t_2]^{p_2}]^p) \stackrel{\text{def}}{=}$$

$\quad \mathsf{let}\ V_1 = \text{analyze}_{\mathsf{rec}}(\mathsf{false}, \delta, \Gamma^\sharp, [t_1]^{p_1})\ \mathsf{in}$

$\quad \mathsf{if}\ V_1 = \top\ \mathsf{then}\ \top\ \mathsf{else}$

$\quad \mathsf{let}\ V_2 = \text{analyze}_{\mathsf{rec}}(\mathsf{false}, \delta, \Gamma^\sharp, [t_2]^{p_2})\ \mathsf{in}$

$\quad \mathsf{let}\ \text{iw}' = \text{maximal}(\phi(\delta p))\ \mathsf{in}$

$\quad \bigcup^{\mathsf{clos}^\sharp}_{\langle \lambda^\ell \mathsf{x}.\, [t_3]^{p_3}, \Gamma_3^\sharp \rangle \in V_1} \text{analyze}_{\mathsf{rec}}(\text{iw}', \phi(\delta p), (\Gamma_3^\sharp, \mathsf{x} : V_2), [t_3]^{p_3})$

Controls whether the solver should widen inputs

# Experimental evaluation

Experiments: $0$-CFA vs. $1$-CFA vs. $1^\star$-CFA vs. $1$-$\nabla$CFA vs. $1^\star$-$\nabla$CFA

- Measures: size of search space, number of iterations, precision of the results
- $1^\star$ strategy: more precise results at a cost similar to "last 1 call site" strategy
- $\nabla$CFA always performs less iterations than standard CFA
- $\nabla$CFA exploits infinite numeric domains (intervals), with benefit!
- $\nabla$CFA less precise on recursive CPS programs (requires cyclic abstract values)
- $1^\star$-$\nabla$CFA is a new, promising analysis for control flow!
- See table on next slide

# Experimental results on classic micro-benchmarks: 1-CFA vs. 1-∇CFA vs. $1^\star$-∇CFA

| Program | 1-CFA | | | | 1-∇CFA | | | | $1^\star$-∇CFA | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | States | Edges | Iter | Result | States | Edges | Iter | Result | States | Edges | Iter | Result |
| ack | 107 | 183 | 4 | B | 177 | 397 | 5 | ⊕ | 276 | 579 | 3 | = |
| ack_cps | 161 | 250 | 7 | B | 162 | 289 | 3 | ⊖ | 292 | 548 | 3 | ⊖ |
| binomial | 99 | 161 | 3 | B | 148 | 274 | 4 | ⊕ | 254 | 531 | 4 | ⊕ |
| blur | 64 | 81 | 10 | B° | 54 | 61 | 2 | =° | 55 | 61 | 2 | =° |
| church | 269 | 427 | 21 | B | 403 | 636 | 3 | ⊖ | 444 | 642 | 3 | ⊖ |
| delta_delta | 8 | 7 | 3 | B° | 8 | 7 | 2 | =° | 10 | 9 | 2 | =° |
| eta | 14 | 14 | 3 | B° | 14 | 14 | 2 | =° | 14 | 14 | 2 | =° |
| facehugger | 58 | 74 | 5 | B | 76 | 117 | 4 | ⊕ | 74 | 99 | 4 | ⊕ |
| fact | 25 | 33 | 4 | B | 25 | 39 | 4 | ⊕ | 36 | 53 | 4 | ⊕ |
| fact_cps | 60 | 71 | 7 | B | 60 | 81 | 2 | ⊖ | 76 | 104 | 2 | ⊖ |
| fact_tailrec | 37 | 47 | 5 | B | 41 | 58 | 3 | ⊕ | 50 | 65 | 3 | ⊕ |
| hmca100 | 1605 | 2002 | 4 | B° | 1605 | 2002 | 2 | =° | 1605 | 2002 | 2 | =° |
| hmca200 | 3205 | 4002 | 4 | B° | 3205 | 4002 | 2 | =° | 3205 | 4002 | 2 | =° |
| hmca300 | 4805 | 6002 | 4 | B° | 4805 | 6002 | 2 | =° | 4805 | 6002 | 2 | =° |
| kcfa2 | 54 | 75 | 5 | B | 59 | 83 | 2 | = | 91 | 97 | 2 | ⊕° |
| kcfa3 | 79 | 119 | 6 | B | 88 | 131 | 2 | = | 167 | 173 | 2 | ⊕° |
| mc91 | 37 | 56 | 4 | B | 40 | 68 | 3 | ⊕ | 74 | 125 | 2 | ⊕ |
| mc91_cps | 70 | 96 | 8 | B | 82 | 125 | 3 | ⊖ | 92 | 130 | 2 | ⊖ |
| mc91_tailrec | 74 | 102 | 5 | B | 104 | 180 | 4 | ⊕ | 118 | 193 | 3 | = |
| mj09 | 42 | 44 | 7 | B | 43 | 47 | 2 | ⊕ | 44 | 44 | 2 | ⊕° |
| sat | 156 | 278 | 14 | B | 192 | 303 | 2 | = | 129 | 166 | 3 | ⊕° |
| sat3 | 105 | 160 | 12 | B | 140 | 208 | 2 | = | 96 | 109 | 2 | ⊕° |
| shivers | 11 | 10 | 4 | B° | 14 | 18 | 2 | =° | 17 | 21 | 2 | =° |
| shivers2 | 49 | 55 | 8 | B° | 49 | 56 | 2 | =° | 49 | 56 | 2 | =° |
| tak | 137 | 256 | 4 | B | 195 | 401 | 3 | = | 402 | 849 | 3 | = |
| tak_cps | 227 | 368 | 12 | B | 250 | 388 | 2 | ⊖ | 225 | 398 | 2 | ⊖ |
| tak_4d | 231 | 464 | 4 | B | 315 | 683 | 3 | = | 797 | 1757 | 3 | = |

# Ongoing work

## A more precise abstract domain for abstract closures:

- Can express *recursively-defined* sets of values
- Idea: an abstract domain with *cycles*
  - Like standard CFA, *without* the name indirections

$$\left\{ (\lambda x.\, f\,(g\,x)) \left[ \begin{array}{l} f \mapsto \{(\lambda x.\, x)[\,]\} \\ g \mapsto \mu\alpha.\left\{ (\lambda x.\, x), (\lambda x.\, f\,(g\,x)) \left[ \begin{array}{l} f \mapsto \{(\lambda x.\, x)[\,]\} \\ g \mapsto \alpha \end{array} \right] \right\} \end{array} \right] \right\}$$

  - Similar to the idea of recursive types
  - Algorithms similar to unification on term-graphs
- How is it related to tree automata? (Comon et al. 2007)
- Exploit sharing as in Mauborgne's Ph.D. thesis? (Mauborgne 1999)

# How to get the artefact

Demo!

- Try out the demo in your browser!
  `https://people.irisa.fr/Benoit.Montagu/projects/cfa/demo/cfa.html`
- OCaml sources available on ACM Digital Library and at
  `https://people.irisa.fr/Benoit.Montagu/projects/cfa/sources/cfa.tar.bz2`
- Ask me if you want to try out the most recent versions

# Conclusion and Future Work

Take-back-home ideas:

- Traces with control-flow events: a semantic foundation for CFA
- A semantic, modular justification for constraint-based CFA
- A new strategy for limiting context: at most $k$ repetitions in call strings
- $\nabla$CFA: a new analysis for control flow, with widening

  ☞ Faster convergence ☞ Can exploit expressive numeric domains

# Conclusion and Future Work

Take-back-home ideas:

- Traces with control-flow events: a semantic foundation for CFA
- A semantic, modular justification for constraint-based CFA
- A new strategy for limiting context: at most $k$ repetitions in call strings
- $\nabla$CFA: a new analysis for control flow, with widening

  ☞ Faster convergence ☞ Can exploit expressive numeric domains

Future work:

- More experiments, on larger programs
- A more expressive abstract domain for closures
- More features: ADTs, exceptions, mutable state, resource usage...
- Can traces explain polymorphic splitting (Wright and Jagannathan 1998)? pushdown-CFA (Vardoulakis and Shivers 2011)?
- Relational version of CFA (expanding on ICFP'20 stable relations paper)

See you virtually at PLDI'21!

# References i


- Comon, H. et al. (2007). *Tree Automata Techniques and Applications*. Available on: `http://www.grappa.univ-lille3.fr/tata`. Release October, 12th 2007.
- Horn, David Van and Matthew Might (2010). “Abstracting Abstract Machines”. In: *Proceedings of the 15th ACM SIGPLAN international conference on Functional programming - ICFP ’10*. ACM Press.
- Jones, Neil D. and Alan Mycroft (1986). “Data Flow Analysis of Applicative Programs Using Minimal Function Graphs”. In: *Proceedings of the 13th ACM SIGACT-SIGPLAN symposium on Principles of programming languages - POPL ’86*. ACM Press.
- Mauborgne, Laurent (Nov. 1999). “Representation of Sets of Trees for Abstract Interpretation”. PhD thesis. École Polytechnique.
- Nielson, Flemming, Hanne Riis Nielson and Chris Hankin (1999). *Principles of Program Analysis*. Springer Berlin Heidelberg.

- Shivers, Olin (1991). "The Semantics of Scheme Control-Flow Analysis". In: *Proceedings of the 1991 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation*. PEPM '91. New York, NY, USA: Association for Computing Machinery, pp. 190–198. ISBN: 0897914333.
- Vardoulakis, Dimitrios and Olin Shivers (2011). "Pushdown flow analysis of first-class control". In: *Proceeding of the 16th ACM SIGPLAN international conference on Functional Programming, ICFP 2011, Tokyo, Japan, September 19-21, 2011*. Ed. by Manuel M. T. Chakravarty, Zhenjiang Hu and Olivier Danvy. ACM, pp. 69–80.
- Wright, Andrew K. and Suresh Jagannathan (1998). "Polymorphic Splitting: An Effective Polyvariant Flow Analysis". In: *ACM Trans. Program. Lang. Syst.* 20.1, pp. 166–207.