« Bi-inductive Structural Semantics and its Abstraction »

Patrick Cousot
École normale supérieure
45 rue d'Ulm, 75230 Paris cedex 05, France
Patrick.Cousot@ens.fr   www.di.ens.fr/~cousot

(joint work with Radhia Cousot)

Departmental Seminar — Department of Computing, Imperial College London

Wednesday July 4th, 2007

1. Motivation

Contents

Motivation

– We look for a formalism to specify abstract program semantics

    from definitional semantics ...

    to static program analysis algorithms

  handling the many different styles of presentations found in the literature (rules, fixpoint, equations, constraints, ...) in a uniform way

– A simple generalization of inductive definitions from sets to posets seems adequate.

On the importance of defining both finite and infinite behaviors

- Example of the *choice operator* $E_1 \mid E_2$ where:

$$\begin{array}{lll} & E_1 \Longrightarrow a \quad E_2 \Longrightarrow b & \text{termination} \\ \text{or} & E_1 \Longrightarrow \bot \quad E_2 \Longrightarrow \bot & \text{non-termination} \end{array}$$

- The *finite behavior* of $E_1 \mid E_2$ is:

$$a \mid b \Longrightarrow a \qquad a \mid b \Longrightarrow b \quad .$$

2. Semantics of the Eager λ-calculus

[1] P. Cousot & R. Cousot. Bi-inductive Structural Semantics. SOS 2007, July 9, 2007, Wroclaw, Poland.

- But for the case $\bot \mid \bot \Longrightarrow \bot$, the *infinite behaviors* of $E_1 \mid E_2$ depend on the choice method:

Non-deterministic	Parallel	Eager	Mixed left-to-right	Mixed right-to-left
$\bot \mid b \Longrightarrow b$	$\bot \mid b \Longrightarrow b$			$\bot \mid b \Longrightarrow b$
$\bot \mid b \Longrightarrow \bot$		$\bot \mid b \Longrightarrow \bot$	$\bot \mid b \Longrightarrow \bot$	$\bot \mid b \Longrightarrow \bot$
$a \mid \bot \Longrightarrow a$	$a \mid \bot \Longrightarrow a$		$a \mid \bot \Longrightarrow a$	
$a \mid \bot \Longrightarrow \bot$		$a \mid \bot \Longrightarrow \bot$	$a \mid \bot \Longrightarrow \bot$	$a \mid \bot \Longrightarrow \bot$

- Nondeterministic: an internal choice is made initially to evaluate E_1 or to evaluate E_2;
- Parallel: evaluate E_1 and E_2 concurrently, with an unspecified scheduling, and return the first available result a or b;
- Mixed left-to-right: evaluate E_1 and then either return its result a or evaluate E_2 and return its result b;
- Mixed right-to-left: evaluate E_2 and then either return its result b or evaluate E_1 and return its result a;
- Eager: evaluate both E_1 and E_2 and return either results if both terminate.

Syntax

Syntax of the Eager λ-calculus

$$
\begin{array}{rcll}
x, y, z, \ldots & \in & \mathbb{X} & \text{variables} \\
c & \in & \mathbb{C} & \text{constants } (\mathbb{X} \cap \mathbb{C} = \emptyset) \\
c & ::= & 0 \mid 1 \mid \ldots & \\
v & \in & \mathbb{V} & \text{values} \\
v & ::= & c \mid \lambda x \cdot a & \\
e & \in & \mathbb{E} & \text{errors} \\
e & ::= & c\ a \mid e\ a & \\
a, a', a_1, \ldots, b, , \ldots & \in & \mathbb{T} & \text{terms} \\
a & ::= & x \mid v \mid a\ a' &
\end{array}
$$

Trace Semantics

Example I: Finite Computation

function argument

$((\lambda x \cdot x\ x)\ (\lambda y \cdot y))\ ((\lambda z \cdot z)\ 0)$

$\longrightarrow$ evaluate function

$((\lambda y \cdot y)\ (\lambda y \cdot y))\ ((\lambda z \cdot z)\ 0)$

$\longrightarrow$ evaluate function, cont'd

$(\lambda y \cdot y)\ ((\lambda z \cdot z)\ 0)$

$\longrightarrow$ evaluate argument

$(\lambda y \cdot y)\ 0$

$\longrightarrow$ apply function to argument

0 *a value!*

Example II: Infinite Computation

function argument

$(\lambda x \cdot x\ x)\ (\lambda x \cdot x\ x)$

$\longrightarrow$ apply function to argument

$(\lambda x \cdot x\ x)\ (\lambda x \cdot x\ x)$

$\longrightarrow$ apply function to argument

$(\lambda x \cdot x\ x)\ (\lambda x \cdot x\ x)$

$\longrightarrow$ apply function to argument

$\ldots$ *non termination!*

Example III: Erroneous Computation

function argument

$((\lambda x \cdot x\ x)\ ((\lambda z \cdot z)\ 0))\ ((\lambda y \cdot y)\ 0)$

$\rightarrow$ evaluate argument

$((\lambda x \cdot x\ x)\ ((\lambda z \cdot z)\ 0))\ 0$

$\rightarrow$ evaluate function

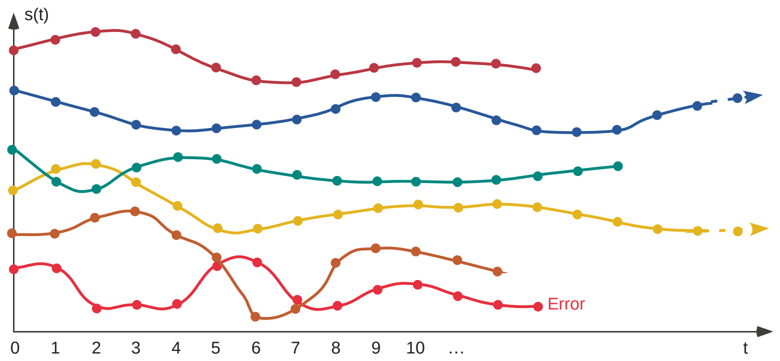
$((\lambda x \cdot x\ x)\ 0)\ 0$

$\rightarrow$ evaluate function, cont'd

$(0\ 0)\ 0$

a runtime error!

Finite, Infinite and Erroneous Trace Semantics



Traces

- $\mathbb{T}^\star$ (resp. $\mathbb{T}^+$, $\mathbb{T}^\omega$, $\mathbb{T}^\propto$ and $\mathbb{T}^\infty$) be the set of finite (resp. nonempty finite, infinite, finite or infinite, and nonempty finite or infinite) sequences of terms
- ϵ is the empty sequence $\epsilon \bullet \sigma = \sigma \bullet \epsilon = \sigma$.
- $|\sigma| \in \mathbb{N} \cup \{\omega\}$ is the length of $\sigma \in \mathbb{T}^\propto$. $|\epsilon| = 0$.
- If $\sigma \in \mathbb{T}^+$ then $|\sigma| > 0$ and $\sigma = \sigma_0 \bullet \sigma_1 \bullet \ldots \bullet \sigma_{|\sigma|-1}$.
- If $\sigma \in \mathbb{T}^\omega$ then $|\sigma| = \omega$ and $\sigma = \sigma_0 \bullet \ldots \bullet \sigma_n \bullet \ldots$.

Operations on Traces

- For $a \in \mathbb{T}$ and $\sigma \in \mathbb{T}^\infty$, we define $a@\sigma$ to be $\sigma' \in \mathbb{T}^\infty$ such that $\forall i < |\sigma| : \sigma'_i = a\ \sigma_i$

$$\sigma = \sigma_0 \quad \sigma_1 \quad \sigma_2 \quad \sigma_3 \quad \ldots \quad \sigma_i \quad \ldots$$

$$a@\sigma = a\ \sigma_0 \quad a\ \sigma_1 \quad a\ \sigma_2 \quad a\ \sigma_3 \quad \ldots \quad a\ \sigma_i \quad \ldots$$

Example

- $\mathsf{a} = (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})$
- $\sigma = ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0) \bullet 0$
- $\mathsf{a}@\sigma =$

  $(\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})@((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0) \bullet 0 =$

  $((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0)) \bullet ((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ 0)$

Example

- $\sigma = ((\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{x}\ \mathsf{x})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})) \bullet ((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})) \bullet (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})$
- $\mathsf{b} = ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0)$
- $(\sigma@\mathsf{b})$

  $=$

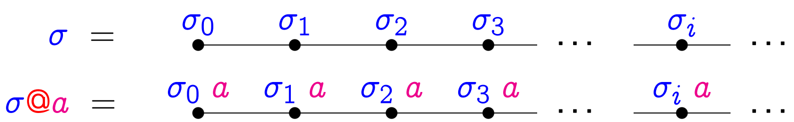
  $(((\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{x}\ \mathsf{x})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})) \bullet ((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})) \bullet (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})@((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0))$

  $=$

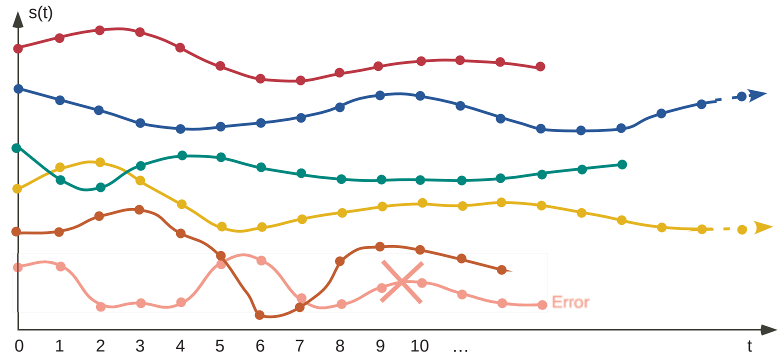
  $(((\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{x}\ \mathsf{x})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y}))\ ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0)) \bullet (((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y}))\ ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0)) \bullet$
  $((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0))$

Operations on Traces (Cont'd)

- Similarly for $\mathsf{a} \in \mathbb{T}$ and $\sigma \in \mathbb{T}^\infty$, $\sigma@a$ is σ' where $\forall i < |\sigma| : \sigma'_i = \sigma_i\ a$



Finite and Infinite Trace Semantics



Bifinitary Trace Semantics $\vec{\mathbb{S}}$ of the Eager λ-calculus [1] [CC92]

$$v \in \vec{\mathbb{S}},\quad v \in \mathbb{V}$$

$$\frac{a[x \leftarrow v] \bullet \sigma \in \vec{\mathbb{S}}}{(\lambda x \cdot a)\ v \bullet a[x \leftarrow v] \bullet \sigma \in \vec{\mathbb{S}}} \sqsubseteq,\quad v \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{a@\sigma \in \vec{\mathbb{S}}} \sqsubseteq,\quad a \in \mathbb{V}$$

$$\frac{\sigma \bullet v \in \vec{\mathbb{S}}^{+},\ (a\ v) \bullet \sigma' \in \vec{\mathbb{S}}}{(a@\sigma) \bullet (a\ v) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\quad v, a \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{\sigma@b \in \vec{\mathbb{S}}} \sqsubseteq$$

$$\frac{\sigma \bullet v \in \vec{\mathbb{S}}^{+},\ (v\ b) \bullet \sigma' \in \vec{\mathbb{S}}}{(\sigma@b) \bullet (v\ b) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\quad v \in \mathbb{V}$$

[1] Note: $a[x \leftarrow b]$ is the capture-avoiding substitution of b for all free occurences of x within a. We let FV(a) be the free variables of a. We define the call-by-value semantics of closed terms (without free variables) $\overline{\mathbb{T}} \triangleq \{a \in \mathbb{T} \mid FV(a) = \emptyset\}$.

Bifinitary Trace Semantics $\vec{\mathbb{S}}$ of the Eager λ-calculus [1] [CC92]

$$v \in \vec{\mathbb{S}},\quad v \in \mathbb{V}$$

$$\frac{a[x \leftarrow v] \bullet \sigma \in \vec{\mathbb{S}}}{(\lambda x \cdot a)\ v \bullet a[x \leftarrow v] \bullet \sigma \in \vec{\mathbb{S}}} \sqsubseteq,\quad v \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{a@\sigma \in \vec{\mathbb{S}}} \sqsubseteq,\quad a \in \mathbb{V}$$

$$\frac{\sigma \bullet v \in \vec{\mathbb{S}}^{+},\ (a\ v) \bullet \sigma' \in \vec{\mathbb{S}}}{(a@\sigma) \bullet (a\ v) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\quad v, a \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{\sigma@b \in \vec{\mathbb{S}}} \sqsubseteq$$

$$\frac{\sigma \bullet v \in \vec{\mathbb{S}}^{+},\ (v\ b) \bullet \sigma' \in \vec{\mathbb{S}}}{(\sigma@b) \bullet (v\ b) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\quad v \in \mathbb{V}$$

[1] Note: $a[x \leftarrow b]$ is the capture-avoiding substitution of b for all free occurences of x within a. We let FV(a) be the free variables of a. We define the call-by-value semantics of closed terms (without free variables) $\overline{\mathbb{T}} \triangleq \{a \in \mathbb{T} \mid FV(a) = \emptyset\}$.

Bifinitary Trace Semantics $\vec{\mathbb{S}}$ of the Eager λ-calculus [1] [CC92]

$$v \in \vec{\mathbb{S}},\quad v \in \mathbb{V}$$

$$\frac{a[x \leftarrow v] \bullet \sigma \in \vec{\mathbb{S}}}{(\lambda x \cdot a)\ v \bullet a[x \leftarrow v] \bullet \sigma \in \vec{\mathbb{S}}} \sqsubseteq,\quad v \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{a@\sigma \in \vec{\mathbb{S}}} \sqsubseteq,\quad a \in \mathbb{V}$$

$$\frac{\sigma \bullet v \in \vec{\mathbb{S}}^{+},\ (a\ v) \bullet \sigma' \in \vec{\mathbb{S}}}{(a@\sigma) \bullet (a\ v) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\quad v, a \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{\sigma@b \in \vec{\mathbb{S}}} \sqsubseteq$$

$$\frac{\sigma \bullet v \in \vec{\mathbb{S}}^{+},\ (v\ b) \bullet \sigma' \in \vec{\mathbb{S}}}{(\sigma@b) \bullet (v\ b) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\quad v \in \mathbb{V}$$

[1] Note: $a[x \leftarrow b]$ is the capture-avoiding substitution of b for all free occurences of x within a. We let FV(a) be the free variables of a. We define the call-by-value semantics of closed terms (without free variables) $\overline{\mathbb{T}} \triangleq \{a \in \mathbb{T} \mid FV(a) = \emptyset\}$.

Bifinitary Trace Semantics $\vec{\mathbb{S}}$ of the Eager λ-calculus [1] [CC92]

$$v \in \vec{\mathbb{S}},\quad v \in \mathbb{V}$$

$$\frac{a[x \leftarrow v] \bullet \sigma \in \vec{\mathbb{S}}}{(\lambda x \cdot a)\ v \bullet a[x \leftarrow v] \bullet \sigma \in \vec{\mathbb{S}}} \sqsubseteq,\quad v \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{a@\sigma \in \vec{\mathbb{S}}} \sqsubseteq,\quad a \in \mathbb{V}$$

$$\frac{\sigma \bullet v \in \vec{\mathbb{S}}^{+},\ (a\ v) \bullet \sigma' \in \vec{\mathbb{S}}}{(a@\sigma) \bullet (a\ v) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\quad v, a \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{\sigma@b \in \vec{\mathbb{S}}} \sqsubseteq$$

$$\frac{\sigma \bullet v \in \vec{\mathbb{S}}^{+},\ (v\ b) \bullet \sigma' \in \vec{\mathbb{S}}}{(\sigma@b) \bullet (v\ b) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\quad v \in \mathbb{V}$$

[1] Note: $a[x \leftarrow b]$ is the capture-avoiding substitution of b for all free occurences of x within a. We let FV(a) be the free variables of a. We define the call-by-value semantics of closed terms (without free variables) $\overline{\mathbb{T}} \triangleq \{a \in \mathbb{T} \mid FV(a) = \emptyset\}$.

Non-Standard Meaning of the Rules

The rules

$$\mathcal{R} = \left\{ \frac{P_i}{C_i} \sqsubseteq \;\middle|\; i \in \Delta \right\}$$

define

$$\mathsf{lfp}^{\sqsubseteq}\, F[\![\mathcal{R}]\!]$$

where the *consequence operator* is

$$F[\![\mathcal{R}]\!](T) = \bigsqcup \left\{ C \;\middle|\; P \sqsubseteq T \wedge \frac{P}{C} \sqsubseteq \in \mathcal{R} \right\}$$

and ...

Bifinitary Trace Semantics $\vec{\mathbb{S}}$ of the Eager λ-calculus [1] [CC92]

$$\mathsf{v} \in \vec{\mathbb{S}},\ \mathsf{v} \in \mathbb{V} \qquad \frac{\mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \bullet \sigma \in \vec{\mathbb{S}}}{(\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{a})\ \mathsf{v} \bullet \mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \bullet \sigma \in \vec{\mathbb{S}}} \sqsubseteq,\ \mathsf{v} \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{\mathsf{a}@\sigma \in \vec{\mathbb{S}}} \sqsubseteq,\ \mathsf{a} \in \mathbb{V} \qquad \frac{\sigma \bullet \mathsf{v} \in \vec{\mathbb{S}}^{+},\ (\mathsf{a}\ \mathsf{v}) \bullet \sigma' \in \vec{\mathbb{S}}}{(\mathsf{a}@\sigma) \bullet (\mathsf{a}\ \mathsf{v}) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\ \mathsf{v}, \mathsf{a} \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{\sigma@\mathsf{b} \in \vec{\mathbb{S}}} \sqsubseteq \qquad \frac{\sigma \bullet \mathsf{v} \in \vec{\mathbb{S}}^{+},\ (\mathsf{v}\ \mathsf{b}) \bullet \sigma' \in \vec{\mathbb{S}}}{(\sigma@\mathsf{b}) \bullet (\mathsf{v}\ \mathsf{b}) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\ \mathsf{v} \in \mathbb{V}$$

[1] Note: $\mathsf{a}[\mathsf{x} \leftarrow \mathsf{b}]$ is the capture-avoiding substitution of b for all free occurences of x within a. We let FV(a) be the free variables of a. We define the call-by-value semantics of closed terms (without free variables) $\overline{\mathbb{T}} \triangleq \{\mathsf{a} \in \mathbb{T} \mid \mathrm{FV}(\mathsf{a}) = \emptyset\}$.

The Computational Lattice

Given $S, T \in \wp(\mathbb{T}^{\infty})$, we define

- $S^{+} \triangleq S \cap \mathbb{T}^{+}$ — finite traces
- $S^{\omega} \triangleq S \cap \mathbb{T}^{\omega}$ — infinite traces
- $S \sqsubseteq T \triangleq S^{+} \subseteq T^{+} \wedge S^{\omega} \supseteq T^{\omega}$ — computational order
- $\langle \wp(\mathbb{T}^{\infty}), \sqsubseteq, \mathbb{T}^{\omega}, \mathbb{T}^{+}, \sqcup, \sqcap \rangle$ is a complete lattice

Example

$$\frac{\sigma \bullet \mathsf{v} \in \vec{\mathbb{S}}^{+},\ (\mathsf{a}\ \mathsf{v}) \bullet \sigma' \in \vec{\mathbb{S}}}{(\mathsf{a}@\sigma) \bullet (\mathsf{a}\ \mathsf{v}) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq, \quad \mathsf{v}, \mathsf{a} \in \mathbb{V}\ .$$

- $\sigma \bullet \mathsf{v} = ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0) \bullet 0 \in \in \vec{\mathbb{S}}^{+}$
- $(\mathsf{a}\ \mathsf{v}) \bullet \sigma' = (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ 0 \bullet 0 \in \vec{\mathbb{S}}$
- $(\mathsf{a}@\sigma) \bullet (\mathsf{a}\ \mathsf{v}) \bullet \sigma'$
  $=$
  $((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})@((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0) \bullet\ 0) \bullet 0$
  $=$
  $(\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0) \bullet (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ 0 \bullet 0 \in \vec{\mathbb{S}}$

Bifinitary Trace Semantics $\vec{\mathbb{S}}$ of the Eager λ-calculus [1] [CC92]

$$\mathsf{v} \in \vec{\mathbb{S}},\ \mathsf{v} \in \mathbb{V} \qquad \frac{\mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \bullet \sigma \in \vec{\mathbb{S}}}{(\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{a})\ \mathsf{v} \bullet \mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \bullet \sigma \in \vec{\mathbb{S}}} \sqsubseteq,\ \mathsf{v} \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{\mathsf{a}@\sigma \in \vec{\mathbb{S}}} \sqsubseteq,\ \mathsf{a} \in \mathbb{V} \qquad \frac{\sigma \bullet \mathsf{v} \in \vec{\mathbb{S}}^{+},\ (\mathsf{a}\ \mathsf{v}) \bullet \sigma' \in \vec{\mathbb{S}}}{(\mathsf{a}@\sigma) \bullet (\mathsf{a}\ \mathsf{v}) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\ \mathsf{v}, \mathsf{a} \in \mathbb{V}$$

$$\frac{\sigma \in \vec{\mathbb{S}}^{\omega}}{\sigma@\mathsf{b} \in \vec{\mathbb{S}}} \sqsubseteq \qquad \frac{\sigma \bullet \mathsf{v} \in \vec{\mathbb{S}}^{+},\ (\mathsf{v}\ \mathsf{b}) \bullet \sigma' \in \vec{\mathbb{S}}}{(\sigma@\mathsf{b}) \bullet (\mathsf{v}\ \mathsf{b}) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq,\ \mathsf{v} \in \mathbb{V}$$

[1] Note: $\mathsf{a}[\mathsf{x} \leftarrow \mathsf{b}]$ is the capture-avoiding substitution of b for all free occurences of x within a. We let FV(a) be the free variables of a. We define the call-by-value semantics of closed terms (without free variables) $\overline{\mathbb{T}} \triangleq \{\mathsf{a} \in \mathbb{T} \mid \mathrm{FV}(\mathsf{a}) = \emptyset\}$.

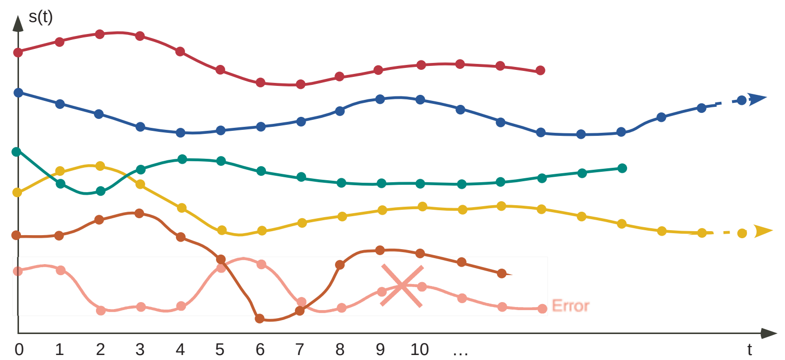
Relational Semantics

Example

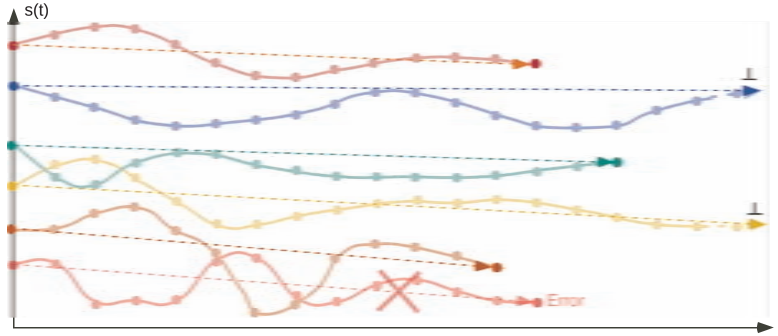
$$\frac{\sigma \bullet \mathsf{v} \in \vec{\mathbb{S}}^{+},\ (\mathsf{v}\ \mathsf{b}) \bullet \sigma' \in \vec{\mathbb{S}}}{(\sigma@\mathsf{b}) \bullet (\mathsf{v}\ \mathsf{b}) \bullet \sigma' \in \vec{\mathbb{S}}} \sqsubseteq, \quad \mathsf{v} \in \mathbb{V}$$

- $\sigma \bullet \mathsf{v} = ((\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{x}\ \mathsf{x})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})) \bullet ((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})) \bullet (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y}) \in \vec{\mathbb{S}}^{+}$
- $(\mathsf{v}\ \mathsf{b}) \bullet \sigma' = (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0) \bullet (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ 0 \bullet 0 \in \vec{\mathbb{S}}$
- $(\sigma@\mathsf{b}) \bullet (\mathsf{v}\ \mathsf{b}) \bullet \sigma'$
  $=$
  $(((\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{x}\ \mathsf{x})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})) \bullet ((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y}))@((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0)) \bullet$
  $((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0)) \bullet (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ 0 \bullet 0$
  $=$
  $((\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{x}\ \mathsf{x})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y}))\ ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0) \bullet ((\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y}))\ ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0)$
  $\bullet\ (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ ((\boldsymbol{\lambda}\,\mathsf{z} \cdot \mathsf{z})\ 0) \bullet (\boldsymbol{\lambda}\,\mathsf{y} \cdot \mathsf{y})\ 0 \bullet 0 \in \vec{\mathbb{S}}$

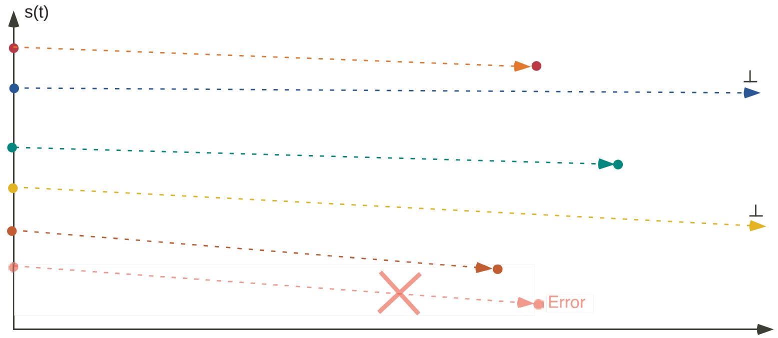
Trace Semantics



Relational Semantics = α(Trace Semantics)



Relational Semantics



Abstraction to the Bifinitary Relational Semantics of the Eager λ-calculus

remember the input/output behaviors,
forget about the intermediate computation steps

$$\alpha(T) \stackrel{\text{def}}{=} \{\alpha(\sigma) \mid \sigma \in T\}$$

$$\alpha(\sigma_0 \bullet \sigma_1 \bullet \ldots \bullet \sigma_n) \stackrel{\text{def}}{=} \langle \sigma_0,\ \sigma_n \rangle$$

$$\alpha(\sigma_0 \bullet \ldots \bullet \sigma_n \bullet \ldots) \stackrel{\text{def}}{=} \langle \sigma_0,\ \bot \rangle$$

Bifinitary Relational Semantics of the Eager λ-calculus

$$\mathsf{v} \Longrightarrow \mathsf{v}, \quad \mathsf{v} \in \mathbb{V}$$

$$\frac{\mathsf{a} \Longrightarrow \bot}{\mathsf{a}\ \mathsf{b} \Longrightarrow \bot}\sqsubseteq \qquad \frac{\mathsf{b} \Longrightarrow \bot}{\mathsf{a}\ \mathsf{b} \Longrightarrow \bot}\sqsubseteq, \quad \mathsf{a} \in \mathbb{V}$$

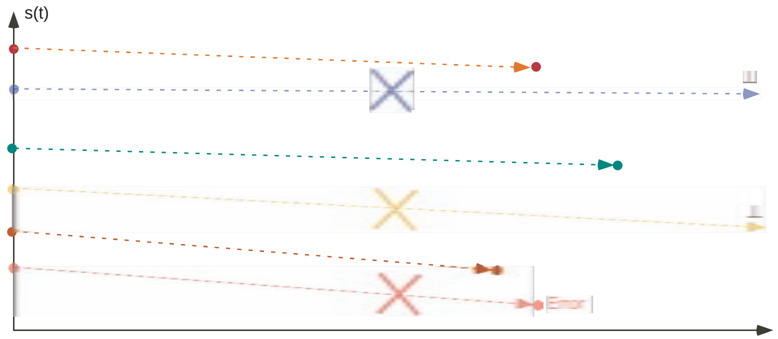
$$\frac{\mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \Longrightarrow r}{(\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{a})\quad \mathsf{v} \Longrightarrow r}\sqsubseteq, \quad \mathsf{v} \in \mathbb{V},\ r \in \mathbb{V} \cup \{\bot\}$$

$$\frac{\mathsf{a} \Longrightarrow \mathsf{v}, \quad \mathsf{v}\ \mathsf{b} \Longrightarrow r}{\mathsf{a}\ \mathsf{b} \Longrightarrow r}\sqsubseteq, \quad \mathsf{v} \in \mathbb{V},\ r \in \mathbb{V} \cup \{\bot\}$$

$$\frac{\mathsf{b} \Longrightarrow \mathsf{v}, \quad \mathsf{a}\ \mathsf{v} \Longrightarrow r}{\mathsf{a}\ \mathsf{b} \Longrightarrow r}\sqsubseteq, \quad \mathsf{a} \in \mathbb{V},\ \mathsf{v} \in \mathbb{V},\ r \in \mathbb{V} \cup \{\bot\}\ .$$

Natural Semantics

Natural Semantics $= \alpha$(Relational Semantics)



Abstraction to the Natural Big-Step Semantics of the Eager λ-calculus

remember the finite input/output behaviors,
forget about non-termination

$$\alpha(T) \stackrel{\text{def}}{=} \bigcup\{\alpha(\sigma) \mid \sigma \in T\}$$

$$\alpha(\langle\sigma_0,\ \sigma_n\rangle) \stackrel{\text{def}}{=} \{\langle\sigma_0,\ \sigma_n\rangle\}$$

$$\alpha(\langle\sigma_0,\ \bot\rangle) \stackrel{\text{def}}{=} \varnothing$$

Natural Big-Step Semantics of the Eager λ-calculus [Kah88]

$$\mathsf{v} \Longrightarrow \mathsf{v}, \quad \mathsf{v} \in \mathbb{V}$$

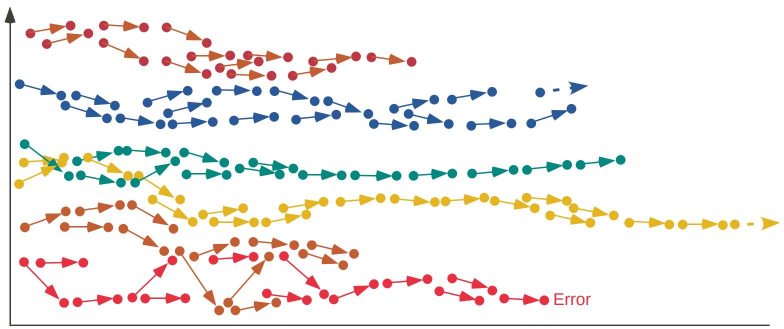
$$\frac{\mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \Longrightarrow r}{(\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{a}) \quad \mathsf{v} \Longrightarrow r} \subseteq, \quad \mathsf{v} \in \mathbb{V},\ r \in \mathbb{V}$$

$$\frac{\mathsf{a} \Longrightarrow \mathsf{v}, \quad \mathsf{v}\ \mathsf{b} \Longrightarrow r}{\mathsf{a}\ \mathsf{b} \Longrightarrow r} \subseteq, \quad \mathsf{v} \in \mathbb{V},\ r \in \mathbb{V}$$

$$\frac{\mathsf{b} \Longrightarrow \mathsf{v}, \quad \mathsf{a}\ \mathsf{v} \Longrightarrow r}{\mathsf{a}\ \mathsf{b} \Longrightarrow r} \subseteq, \quad \mathsf{a} \in \mathbb{V},\ \mathsf{v} \in \mathbb{V},\ r \in \mathbb{V}\ .$$

Transition Semantics

Transition Semantics = α(Trace Semantics)



Abstraction to the Transition Semantics of the Eager λ-calculus

remember execution steps,
forget about their sequencing

$$\alpha(T) \stackrel{\text{def}}{=} \bigcup\{\alpha(\sigma) \mid \sigma \in T\}$$

$$\alpha(\sigma_0 \bullet \sigma_1 \bullet \ldots \bullet \sigma_n) \stackrel{\text{def}}{=} \{\langle \sigma_i,\ \sigma_{i+1}\rangle \mid 0 \leqslant i \wedge i < n\}$$

$$\alpha(\sigma_0 \bullet \ldots \bullet \sigma_n \bullet \ldots) \stackrel{\text{def}}{=} \{\langle \sigma_i,\ \sigma_{i+1}\rangle \mid i \geqslant 0\}$$

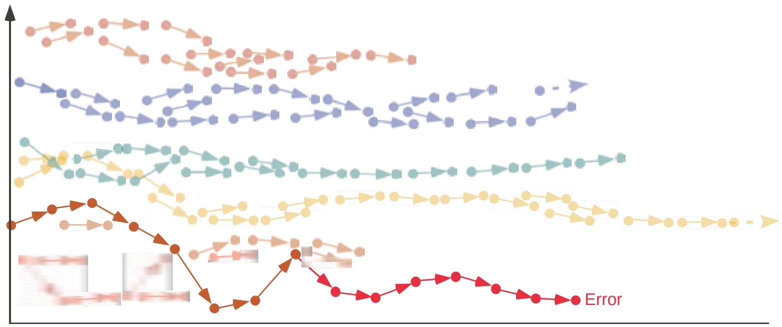
Transition Semantics of the Eager λ-calculus [Plo81]

$$((\lambda\, \mathsf{x} \cdot \mathsf{a})\ \mathsf{v}) \longrightarrow \mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}]$$

$$\frac{\mathsf{a}_0 \longrightarrow \mathsf{a}_1}{\mathsf{a}_0\ \mathsf{b} \longrightarrow \mathsf{a}_1\ \mathsf{b}} \subseteq$$

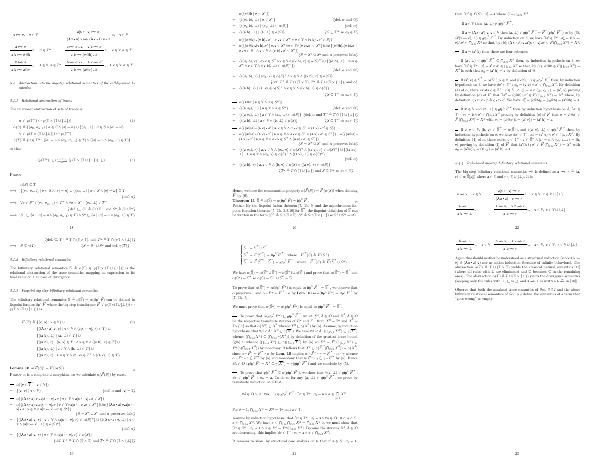
$$\frac{\mathsf{b}_0 \longrightarrow \mathsf{b}_1}{\mathsf{v}\ \mathsf{b}_0 \longrightarrow \mathsf{v}\ \mathsf{b}_1} \subseteq .$$

Approximation



$((\lambda x \cdot x\ x)\ ((\lambda z \cdot z)\ 0))\ (\lambda y \cdot y) \rightarrow ((\lambda x \cdot x\ x)\ 0)\ (\lambda y \cdot y)$
$\rightarrow (0\ 0)\ (\lambda y \cdot y)$ *an error!*

The Abstract Semantics are Correct by Calculational Design



3. Bi-inductive Structural Definitions

[2] P. Cousot & R. Cousot. Bi-inductive Structural Semantics. SOS 2007, July 9, 2007, Wroclaw, Poland.

Syntax

– $\ell, \ell_1, \ldots, \ell_n \in \mathbb{L}$ language
– $\ell \ ::= \ \ell_1, \ldots, \ell_n$ derivation relation
– The "syntactic subcomponent" relation $\prec$ on $\mathbb{L}$:

$$\ell' \prec \ell \triangleq \ell \ ::= \ \ell_1, \ldots, \ell', \ldots \ell_n$$

is
- irreflexive
- finite left images $(\forall \ell \in \mathbb{L} : |\{\ell' \in \mathbb{L} \mid \ell' \prec \ell\}| \in \mathbb{N})$
- well-founded

– Example: a, b, ... ::= x | λ x · a | a b defines a $\prec$ λ x · a, a $\prec$ a b and b $\prec$ a b.

Semantic domains

For each "syntactic component" $\ell \in \mathbb{L}$, we consider a *semantic domain*

$$\langle \mathcal{D}_\ell,\ \sqsubseteq_\ell,\ \bot_\ell,\ \sqcup_\ell \rangle$$

which is assumed to be a directed complete partial order (dcpo).

Variables

– To write definitions we use *variables* X_ℓ, Y_ℓ, ... ranging over the semantic domains $\mathcal{D}_\ell$ of syntactic components $\ell \in \mathbb{L}$.

Transformers

– For derivations $\ell \ ::= \ \ell_1, \ldots, \ell_n$ we consider *transformers*

$$F^i_\ell \in \mathcal{D}_\ell \times \mathcal{D}_{\ell_1} \ldots \times \mathcal{D}_{\ell_n} \longmapsto \mathcal{D}_\ell$$

When $n = 0$, we have $F^i_\ell \in \mathcal{D}_\ell \longmapsto \mathcal{D}_\ell$

– The transformers are assumed to be $\sqsubseteq_\ell$-monotone in their first parameter [2]

[2] $\forall i \in \Delta_\ell,\ \ell_1, \ldots, \ell_n \prec \ell,\ X, Y \in \mathcal{D}_\ell, X_1 \in \mathcal{D}_{\ell_1}, \ldots, X_n \in \mathcal{D}_{\ell_n}\colon\ X \sqsubseteq_\ell Y \Longrightarrow F^i_\ell(X, X_1, \ldots, X_n) \sqsubseteq_\ell F^i_\ell(Y, X_1, \ldots, X_n)$.

Alternatives

– For each "syntactic component" $\ell \in \mathbb{L}$, we let Δ_ℓ be indexed sequences (totally ordered sets) of alternatives/definition cases.

– Given a set S,

$$\begin{array}{lll} & \langle x_i,\ i \in \Delta_\ell \rangle \in \Delta_\ell \mapsto S & \text{indexed sequence} \\ \approx & \displaystyle\prod_{i \in \Delta_\ell} x_i \in \prod_{i \in \Delta_\ell} S & \text{cartesian product} \end{array}$$

Join

– For each "syntactic component" $\ell \in \mathbb{L}$, the *join*

$$\curlyvee_\ell \in (\Delta_\ell \longmapsto \mathcal{D}_\ell) \longmapsto \mathcal{D}_\ell$$

is used to gather alternatives in formal definitions

– The join operator is assumed to be componentwise $\sqsubseteq_\ell$-monotone[3]

– $\displaystyle\bigcurlyvee_{i \in \Delta_\ell} X_i \triangleq \curlyvee_\ell(\prod_{i \in \Delta_\ell} X_i)$, for short

– If the order of presentation of the alternatives is irrelevant Δ_ℓ is a set and the join is associative, commutative, and $\sqsubseteq_\ell$-monotone

[3] $\forall \langle X_i,\ i \in \Delta_\ell\rangle : \forall \langle Y_i,\ i \in \Delta_\ell\rangle : (\forall i \in \Delta_\ell : X_i \sqsubseteq_\ell Y_i) \Longrightarrow \curlyvee_\ell(\prod_{i \in \Delta_\ell} X_i) \sqsubseteq_\ell \curlyvee_\ell(\prod_{i \in \Delta_\ell} Y_i)$.

Fixpoint definitions

A *fixpoint definition* for all $\ell \in \mathbb{L}$ such that $\ell ::= \ell_1, \ldots, \ell_n$ has the form

$$\mathcal{S}_f[\![\ell]\!] = \mathsf{lfp}^{\sqsubseteq_\ell} \boldsymbol{\lambda} X \cdot \bigcurlyvee_{i \in \Delta_\ell} F^i_\ell(X, \mathcal{S}_f[\![\ell_1]\!], \ldots, \mathcal{S}_f[\![\ell_n]\!]) \ .$$

where $\mathsf{lfp}^{\sqsubseteq}$ is the partially defined $\sqsubseteq$-least fixpoint operator on a poset $\langle P, \sqsubseteq\rangle$.

Lemma 1 *$\forall \ell \in \mathbb{L} : \mathcal{S}_f[\![\ell]\!]$ is well defined.* □

Fixpoint definitions, particular cases

– without fixpoint:

$$\bigcurlyvee_{i \in \Delta_\ell} F^i_\ell(\mathcal{S}_f[\![\ell_1]\!], \ldots, \mathcal{S}_f[\![\ell_n]\!]) = \mathsf{lfp}^{\sqsubseteq_\ell} \boldsymbol{\lambda} X \cdot \bigcurlyvee_{i \in \Delta_\ell} F^i_\ell(\mathcal{S}_f[\![\ell_1]\!], \ldots, \mathcal{S}_f[\![\ell_n]\!])$$

– and without join:

$$F^i_\ell(\mathcal{S}_f[\![\ell_1]\!], \ldots, \mathcal{S}_f[\![\ell_n]\!]) = \mathsf{lfp}^{\sqsubseteq_\ell} \boldsymbol{\lambda} X \cdot \bigcurlyvee_{i' \in \{i\}} F^{i'}_\ell(\mathcal{S}_f[\![\ell_1]\!], \ldots, \mathcal{S}_f[\![\ell_n]\!]).$$

Example 1: fixpoint big-step maximal trace semantics

The bifinitary trace semantics $\vec{\mathbb{S}} \in \wp(\overline{\mathbb{T}}^\infty)$ is

$$\vec{\mathbb{S}} \triangleq \mathsf{lfp}^{\sqsubseteq} \vec{F}$$

where $\vec{F} \in \wp(\overline{\mathbb{T}}^\infty) \mapsto \wp(\overline{\mathbb{T}}^\infty)$ is

$$\begin{aligned}
\vec{F}(S) \triangleq\ & \{\mathsf{v} \in \overline{\mathbb{T}}^\infty \mid \mathsf{v} \in \mathbb{V}\} \cup & \text{(a)}\\
& \{(\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{a})\ \mathsf{v} \bullet \mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \bullet \sigma \mid \mathsf{v} \in \mathbb{V} \wedge \mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \bullet \sigma \in S\} \cup & \text{(b)}\\
& \{\sigma @ \mathsf{b} \mid \sigma \in S^\omega\} \cup & \text{(c)}\\
& \{(\sigma @ \mathsf{b}) \bullet (\mathsf{v}\ \mathsf{b}) \bullet \sigma' \mid \sigma \neq \epsilon \wedge \sigma \bullet \mathsf{v} \in S^+ \wedge \mathsf{v} \in \mathbb{V} \wedge (\mathsf{v}\ \mathsf{b}) \bullet \sigma' \in S\} \cup & \text{(d)}\\
& \{\mathsf{a} @ \sigma \mid \mathsf{a} \in \mathbb{V} \wedge \sigma \in S^\omega\} \cup & \text{(e)}\\
& \{(\mathsf{a} @ \sigma) \bullet (\mathsf{a}\ \mathsf{v}) \bullet \sigma' \mid \mathsf{a}, \mathsf{v} \in \mathbb{V} \wedge \sigma \neq \epsilon \wedge \sigma \bullet \mathsf{v} \in S^+ \wedge (\mathsf{a}\ \mathsf{v}) \bullet \sigma' \in S\}\ . & \text{(f)}
\end{aligned}$$

We have $\mathbb{L} = \{\bullet\}$ (no structural induction), $\Delta_\bullet \triangleq \{a, b, c, d, e, f\}$ where $\vec{F}^i_\bullet(S)$, $i \in \Delta_\bullet$ is defined by equation (i). The join operator is chosen in binary form as $\curlyvee_\bullet \triangleq \cup$.

Example 2: fixpoint small-step maximal trace semantics

– The small-step maximal trace semantics $\stackrel{\infty}{\longrightarrow}$ of a transition relation $\longrightarrow$ is

$$\stackrel{n}{\not\longrightarrow} \triangleq \{\sigma \in \mathbb{T}^+ \mid |\sigma| = n > 0 \wedge \forall i : 0 \leqslant i < n-1 : \sigma_i \longrightarrow \sigma_{i+1}\} \quad \text{partial traces}$$

$$\stackrel{n}{\longrightarrow} \triangleq \{\sigma \in \stackrel{n}{\not\longrightarrow} \mid \sigma_{n-1} \in \mathbb{V}\} \quad \text{maximal execution traces of length } n$$

$$\stackrel{+}{\longrightarrow} \triangleq \bigcup_{n>0} \stackrel{n}{\longrightarrow} \quad \text{maximal finite execution traces}$$

$$\stackrel{\omega}{\longrightarrow} \triangleq \{\sigma \in \mathbb{T}^\omega \mid \forall i \in N : \sigma_i \longrightarrow \sigma_{i+1}\} \quad \text{infinite execution traces}$$

$$\stackrel{\infty}{\longrightarrow} \triangleq \stackrel{+}{\longrightarrow} \cup \stackrel{\omega}{\longrightarrow} \quad \text{maximal finite and diverging execution traces.}$$

– Junction $\mathbin{;}$ of set of traces:
$S \mathbin{;} T \triangleq S^\omega \cup \{\sigma_0 \bullet \ldots \bullet \sigma_{|\sigma|-2} \bullet \sigma' \mid \sigma \in S^+ \wedge \sigma_{|\sigma|-1} = \sigma'_0 \wedge \sigma' \in T\}$

– Small-step transformer $\vec{f} \in \wp(\overline{\mathbb{T}}^\infty) \mapsto \wp(\overline{\mathbb{T}}^\infty)$:

$$\vec{f}(T) \triangleq \{\mathsf{v} \in \overline{\mathbb{T}}^\infty \mid \mathsf{v} \in \mathbb{V}\} \cup \stackrel{2}{\not\longrightarrow} \mathbin{;} T \qquad (1)$$

– Small-step maximal trace semantics $\stackrel{\infty}{\longrightarrow}$ in fixpoint form:

$$\stackrel{\infty}{\longrightarrow} = \mathsf{lfp}^{\sqsubseteq} \vec{f}\ .$$

– The big-step and small-step trace semantics are the same

$$\vec{\mathbb{S}} = \stackrel{\infty}{\longrightarrow}\ .$$

Constraint-based definitions

A *constraint-based definition* has the form:

> *$\langle \mathcal{S}_e[\![\ell]\!],\ \ell \in \mathbb{L}\rangle$ is the componentwise $\sqsubseteq_\ell$-least $\langle X_\ell,\ \ell \in \mathbb{L}\rangle$ satisfying the system of constraints (inequations)*

$$\left\{ \begin{array}{l} \displaystyle\bigvee_{i \in \Delta_\ell}{}_{\ell}\ F^i_\ell(X_\ell, \textstyle\prod_{\ell' \prec \ell} X_{\ell'}) \sqsubseteq_\ell X_\ell \\ \ell \in \mathbb{L} \quad . \end{array} \right.$$

Rule-based definitions

– A *rule-based definition* is a sequence of rules of the form

$$\frac{X_\ell}{F^i_\ell(X_\ell, \prod_{\ell' \prec \ell} \mathcal{S}_r[\![\ell']\!])} \sqsubseteq_\ell \qquad \ell \in \mathbb{L}, i \in \Delta_\ell$$

where the premise and conclusion are elements of the $\langle \mathcal{D}_\ell,\ \sqsubseteq_\ell\rangle$ cpo.

– If F^i_ℓ does not depend upon the premise X_ℓ, it is an axiom

Rule-based definitions in logical form

$$\frac{X_\ell \sqsubseteq_\ell \mathcal{S}_r[\![\ell]\!]}{F^i_\ell(X_\ell, \prod_{\ell' \prec \ell} \mathcal{S}_r[\![\ell']\!]) \sqsubseteq_\ell \mathcal{S}_r[\![\ell]\!]} \sqsubseteq_\ell \qquad \ell \in \mathbb{L},\ X_\ell \in \mathcal{D}_\ell, i \in \Delta_\ell$$

To make thejoin $\curlyvee_\ell$ explicit, we can write

$$\frac{X_\ell \sqsubseteq_\ell \mathcal{S}_r[\![\ell]\!]}{\curlyvee_{\ell\, i \in \Delta_\ell} F^i_\ell(X_\ell, \prod_{\ell' \prec \ell} \mathcal{S}_r[\![\ell']\!]) \sqsubseteq_\ell \mathcal{S}_r[\![\ell]\!]} \sqsubseteq_\ell \qquad \ell \in \mathbb{L},\ X_\ell \in \mathcal{D}_\ell\ .$$

4. Abstraction

Proofs

– A $D \in \mathcal{D}_\ell$ is *provable* if and only if it has a *proof* that is a transfinite sequence [4] $D_0, \ldots, D_\lambda$ of elements of $\mathcal{D}_\ell$ such that
  - $D_0 = \bot_\ell$, $D_\lambda = D$ and
  - for all $0 < \delta \leqslant \lambda$, $D_\delta \sqsubseteq_\ell \curlyvee_{\ell\, i \in \Delta_\ell} F^i_\ell(\bigsqcup_{\ell\, \beta<\delta} D_\beta, \prod_{\ell' \prec \ell} \mathcal{S}_r[\![\ell']\!])$.

– The *meaning* of a rule-based definition is

$$\mathcal{S}_r[\![\ell]\!] \triangleq \bigsqcup_\ell \{D \in \mathcal{D}_\ell \mid D \text{ is provable}\}\ .$$

[4] In the classical case [Acz77], the fixpoint operator is continuous whence proofs are finite.

Kleenian abstraction

– $\langle \mathcal{D}, \sqsubseteq, \bot, \sqcup \rangle$, $\langle \mathcal{D}^\sharp, \sqsubseteq^\sharp, \bot^\sharp, \sqcup^\sharp \rangle$ dcpos
– $F \in \mathcal{D} \mapsto \mathcal{D}$, $F^\sharp \in \mathcal{D}^\sharp \mapsto \mathcal{D}^\sharp$ monotone
– $\alpha \in \mathcal{D} \mapsto \mathcal{D}^\sharp$ strict and continuous on chains of $\mathcal{D}$
– $\alpha \circ F = F^\sharp \circ \alpha$, commutation condition

$\Longrightarrow \alpha(\mathsf{lfp}^{\sqsubseteq} F) = \mathsf{lfp}^{\sqsubseteq^\sharp} F^\sharp$

OK for abstracting finite behaviors, not infinite ones

Tarskian abstraction

- $\langle \mathcal{D}, \sqsubseteq, \bot, \sqcup \rangle$, $\langle \mathcal{D}^\sharp, \sqsubseteq^\sharp, \bot^\sharp, \sqcup^\sharp \rangle$ dcpos
- $F \in \mathcal{D} \mapsto \mathcal{D}$, $F^\sharp \in \mathcal{D}^\sharp \mapsto \mathcal{D}^\sharp$ monotone
- $\alpha \in \mathcal{D} \mapsto \mathcal{D}^\sharp$ preserves meets
- $F^\sharp \circ \alpha \sqsubseteq^\sharp \alpha \circ F$, semi-commutation condition
- $\forall y \in \mathcal{D}^\sharp : (F^\sharp(y) \sqsubseteq^\sharp y) \Longrightarrow (\exists x \in \mathcal{D} : \alpha(x) = y \wedge F(x) \sqsubseteq x$

  $\Longrightarrow \alpha(\mathsf{lfp}^{\sqsubseteq} F) = \mathsf{lfp}^{\sqsubseteq^\sharp} F^\sharp$

OK for abstracting infinite behaviors, not finite ones $\Rightarrow$ abstract by parts.

5. Conclusion

Requirements

- Both convergence/termination and divergence/nonterminating behaviors are needed in static strictness analysis [Myc80], safety & security analysis, typing [Cou97, Ler06], etc;
- Such static analyzes must be proved correct with respect to a semantics chosen at an appropriate level of abstraction (small-step/big-step trace/relational/natural semantics);

Requirements satisfaction

- The bifinite extension of OS should satisfy the need for formal finite and infinite semantics, at various levels of abstraction and using various equivalent presentations (fixpoints, equational, constraints and inference rules) needed in static program analysis.

THE END

Bibliography

[Acz77] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, pages 739–782. Elsevier, 1977.

[CC92] P. Cousot and R. Cousot. Inductive definitions, semantics and abstract interpretation. In *19th POPL*, pages 83–94, Albuquerque, NM, US, 1992. ACM Press.

[Cou97] P. Cousot. Types as abstract interpretations, invited paper. In *24th POPL*, pages 316–331, Paris, FR, Jan. 1997. ACM Press.

[Kah88] G. Kahn. Natural semantics. In K. Fuchi and M. Nivat, editors, *Programming of Future Generation Computers*, pages 237–258. Elsevier, 1988.

[Ler06] X. Leroy. Coinductive big-step operational semantics. In P. Sestoft, editor, *Proc. 15th ESOP '2006*, Vienna, AT, LNCS 3924, pages 54–68. Springer, 27–28 Mar. 2006.

[Myc80] A. Mycroft. The theory and practice of transforming call-by-need into call-by-value. In B. Robinet, editor, *Proc. 4th Int. Symp. on Programming*, Paris, FR, 22–24 Apr. 1980, LNCS 83, pages 270–281. Springer, 1980.

THE END, THANK YOU

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, DK, Sep. 1981.