« Specification and Abstraction of Semantics »

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

Radhia Cousot
CNRS & École polytechnique
Route de Saclay
91128 Palaiseau Cedex, France
Radhia.Cousot@polytechnique.fr
www.polytechnique.edu/Radhia.Cousot

A Tribute Workshop and Festival to Honor Neil D. Jones

Datalogisk Institut, Københavns Universitet, København, Denmark— 25–26 August, 2007

Contents

1. Souvenir, Souvenir

Neil D. Jones

An explorer of automatic semantics-based program manipulation

A Long Common Professional Interest and Collaboration

– Semantique I;
– Semantique II;
– Atlantique;
– Daedalus;

Many more shared events

- Århus workshop in 81,
- . . .
- POPL'97 in Paris,
- . . .
- POPL'04 in Venice
- . . .
- Decision to start ASTRÉE
- . . .
- VMCAI'2009

Happy Souvenirs

2. Specification and abstraction of semantics

Motivation

Motivation

– We look for a formalism to specify abstract program semantics

from definitional semantics ...

to static program analysis algorithms

- coping with termination & non-termination,
- 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:

$$E_1 \Longrightarrow a \quad E_2 \Longrightarrow b \qquad \text{termination}$$

$$\text{or} \quad E_1 \Longrightarrow \bot \quad E_2 \Longrightarrow \bot \qquad \text{non-termination}$$

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

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

– But for the case $\perp \mid \perp \Longrightarrow \perp$, 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
$\perp \mid b \Longrightarrow b$	$\perp \mid b \Longrightarrow b$			$\perp \mid b \Longrightarrow b$
$\perp \mid b \Longrightarrow \perp$		$\perp \mid b \Longrightarrow \perp$	$\perp \mid b \Longrightarrow \perp$	$\perp \mid b \Longrightarrow \perp$
$a \mid \perp \Longrightarrow a$	$a \mid \perp \Longrightarrow a$		$a \mid \perp \Longrightarrow a$	
$a \mid \perp \Longrightarrow \perp$		$a \mid \perp \Longrightarrow \perp$	$a \mid \perp \Longrightarrow \perp$	$a \mid \perp \Longrightarrow \perp$

- 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.

Bi-inductive Structural Definitions

Over-simplified for the presentation!

Inductive definitions

Set-theoretic [Acz77]

$\langle \wp(\mathcal{U}),\ \subseteq \rangle$	universe
$\dfrac{P}{c} \in \mathcal{R} \quad (P \in \wp(\mathcal{U}), c \in \mathcal{U})$	rules
$F(X) \triangleq \left\{ c \;\middle|\; \exists \dfrac{P}{c} \in \mathcal{R} : P \subseteq X \right\}$	transformer
$\mathbf{lfp}^{\subseteq} F \in \wp(\mathcal{U})$	fixpoint def.
$\subseteq\!-\text{least } X : F(X) = X$	equational def.
$\subseteq\!-\text{least } X : F(X) \subseteq X$	constraint def.
$\left\{ \dfrac{X}{c} \;\middle|\; X \subseteq \mathcal{U} \wedge c \in F(X) \right\}$	rules

Inductive definitions

Set-theoretic [Acz77]	Order-theoretic	
$\langle \wp(\mathcal{U}),\ \subseteq \rangle$	$\langle \mathcal{D},\ \sqsubseteq \rangle$	universe
$\dfrac{P}{c} \in \mathcal{R} \quad (P \in \wp(\mathcal{U}), c \in \mathcal{U})$	$\dfrac{P}{C} \in \mathcal{R} \quad (P, C \in \mathcal{D})$	rules
$F(X) \triangleq \left\{ c \;\middle\vert\; \exists \dfrac{P}{c} \in \mathcal{R} : P \subseteq X \right\}$	$F(X) \triangleq \bigsqcup \left\{ C \;\middle\vert\; \exists \dfrac{P}{c} \in \mathcal{R} : P \sqsubseteq X \right\}$	transformer
$\mathsf{lfp}^{\subseteq}\ F \in \wp(\mathcal{U})$	$\mathsf{lfp}^{\sqsubseteq}\ F \in \mathcal{D}$	fixpoint def.
$\subseteq-\text{least}\ X : F(X) = X$	$\sqsubseteq-\text{least}\ X : F(X) = X$	equational def.
$\subseteq-\text{least}\ X : F(X) \subseteq X$	$\sqsubseteq-\text{least}\ X : F(X) \sqsubseteq X$	constraint def.
$\left\{ \dfrac{X}{c} \;\middle\vert\; X \subseteq \mathcal{U} \wedge c \in F(X) \right\}$	$\left\{ \dfrac{X}{F(X)} \;\middle\vert\; X \in \mathcal{D} \right\}$	rules

Semantics of the Eager λ-calculus

Syntax

Syntax of the Eager λ-calculus

$$
\begin{array}{rcll}
\mathsf{x}, \mathsf{y}, \mathsf{z}, \ldots & \in & \mathbb{X} & \text{variables} \\
\mathsf{c} & \in & \mathbb{C} & \text{constants } (\mathbb{X} \cap \mathbb{C} = \emptyset) \\
\mathsf{c} & ::= & \mathsf{0} \mid \mathsf{1} \mid \ldots & \\
\mathsf{v} & \in & \mathbb{V} & \text{values} \\
\mathsf{v} & ::= & \mathsf{c} \mid \boldsymbol{\lambda}\, \mathsf{x} \cdot \mathsf{a} & \\
\mathsf{e} & \in & \mathbb{E} & \text{errors} \\
\mathsf{e} & ::= & \mathsf{c}\ \mathsf{a} \mid \mathsf{e}\ \mathsf{a} & \\
\mathsf{a}, \mathsf{a}', \mathsf{a}_1, \ldots, \mathsf{b}, , \ldots & \in & \mathbb{T} & \text{terms} \\
\mathsf{a} & ::= & \mathsf{x} \mid \mathsf{v} \mid \mathsf{a}\ \mathsf{a}' &
\end{array}
$$

Trace Semantics

Example I: Finite Computation

function   argument

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

$\rightarrow$ evaluate function

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

$\rightarrow$ evaluate function, cont'd

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

$\rightarrow$ evaluate argument

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

$\rightarrow$ apply function to argument

0 *a value!*

Example II: Infinite Computation

$$
\begin{array}{ll}
& \color{blue}{\text{function}} \quad \color{magenta}{\text{argument}} \\
& \color{blue}{(\boldsymbol{\lambda}}\,\color{magenta}{x}\,\color{blue}{\cdot\, x\; x)}\;\; \color{magenta}{(\boldsymbol{\lambda}\, x \cdot x\; x)} \\
\longrightarrow & \qquad\qquad\qquad \color{blue}{\text{apply function to argument}} \\
& \color{blue}{(\boldsymbol{\lambda}}\,\color{magenta}{x}\,\color{blue}{\cdot\, x\; x)}\;\; \color{magenta}{(\boldsymbol{\lambda}\, x \cdot x\; x)} \\
\longrightarrow & \qquad\qquad\qquad \color{blue}{\text{apply function to argument}} \\
& \color{blue}{(\boldsymbol{\lambda}}\,\color{magenta}{x}\,\color{blue}{\cdot\, x\; x)}\;\; \color{magenta}{(\boldsymbol{\lambda}\, x \cdot x\; x)} \\
\longrightarrow & \qquad\qquad\qquad \color{blue}{\text{apply function to argument}}
\end{array}
$$

... *non termination!*

Example III: Erroneous Computation

function argument

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

$\rightarrow$ evaluate argument

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

$\rightarrow$ evaluate function

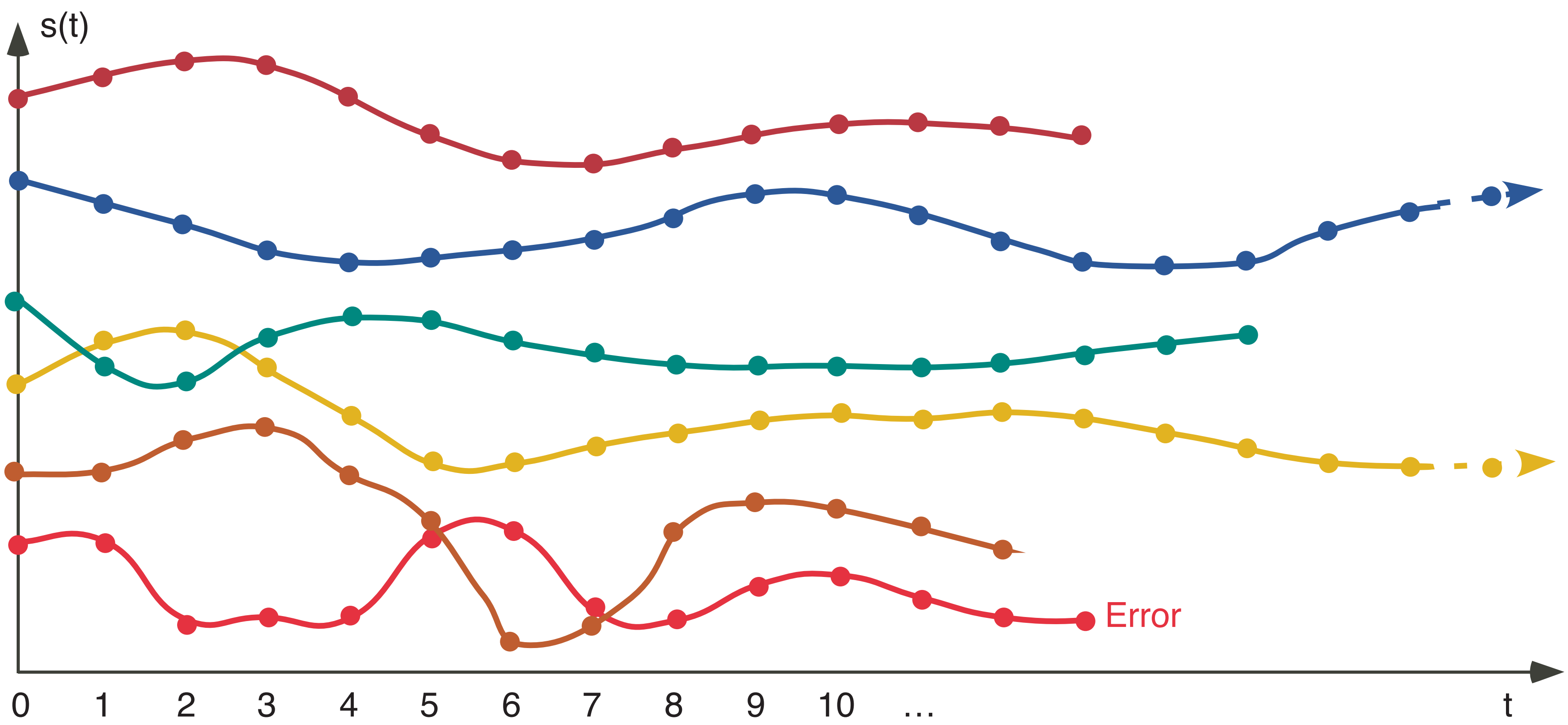
$$((\boldsymbol{\lambda}\, x \cdot x\ x)\ 0)\ 0$$

$\rightarrow$ 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 $\epsilon \bullet \sigma = \sigma \bullet \epsilon = \sigma$.

– $|\sigma| \in N \cup \{\omega\}$ 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

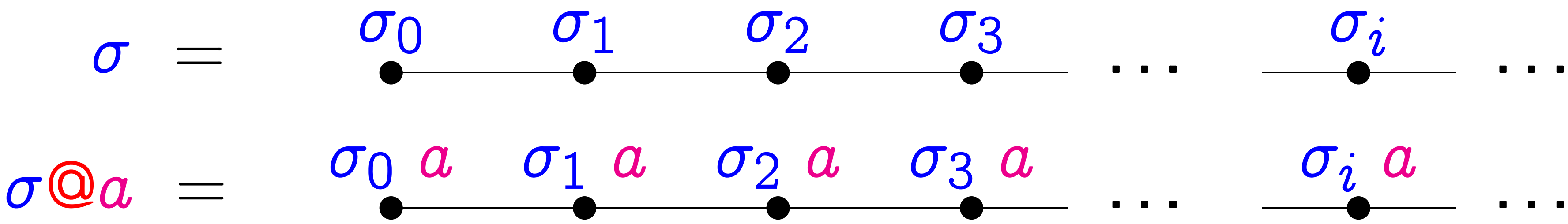
– For a $\in \mathbb{T}$ and $\sigma \in \mathbb{T}^\infty$, we define $a@\sigma$ to be $\sigma' \in \mathbb{T}^\infty$ such that $\forall i < |\sigma| : \sigma'_i = a\ \sigma_i$

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

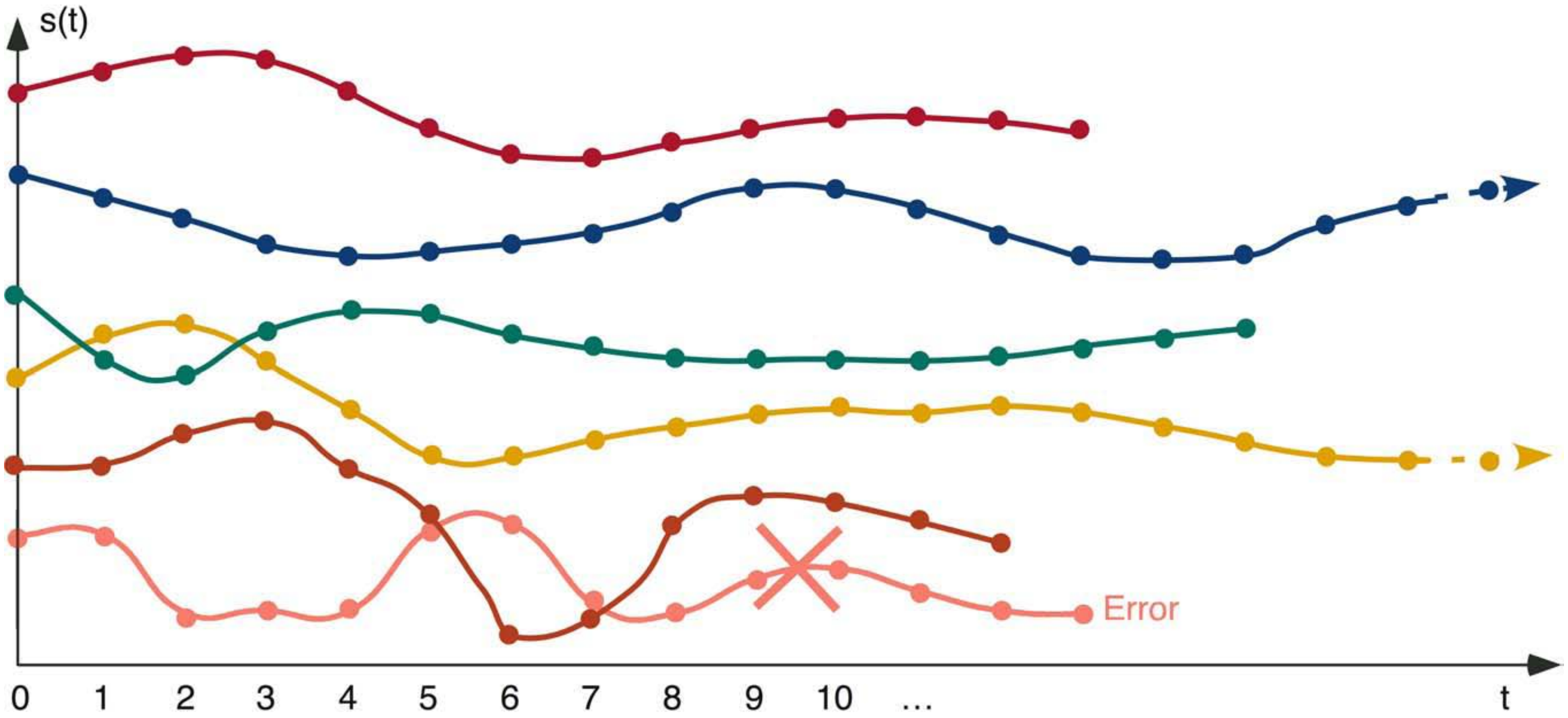
$$a@\sigma \quad = \quad a\ \sigma_0 \;\; a\ \sigma_1 \;\; a\ \sigma_2 \;\; a\ \sigma_3 \;\; \ldots \;\; a\ \sigma_i \;\; \ldots$$

Operations on Traces (Cont'd)

– Similarly for 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 $\vec{\mathbb{S}}$



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

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

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

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

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

---

[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 $\mathrm{FV}(\mathsf{a})$ be the free variables of a. We define the call-by-value semantics of closed terms (without free variables) $\overline{\mathbb{T}} \triangleq \{\mathsf{a} \in \mathbb{T} \mid \mathrm{FV}(\mathsf{a}) = \emptyset\}$.

Fixpoint big-step maximal trace semantics

The bifinitary trace semantics is

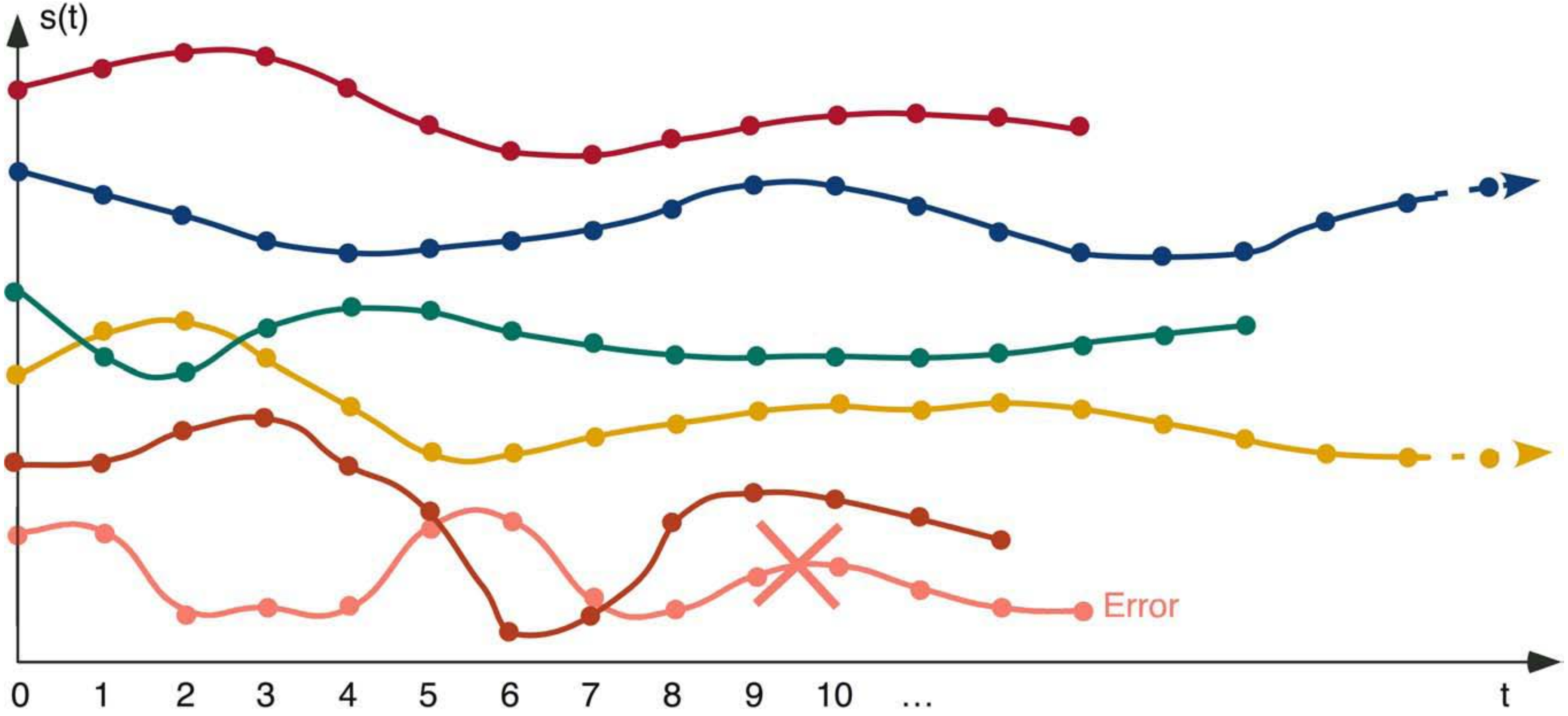
$$\vec{\mathbb{S}} = \mathsf{lfp}^{\sqsubseteq}\ \vec{F}$$

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

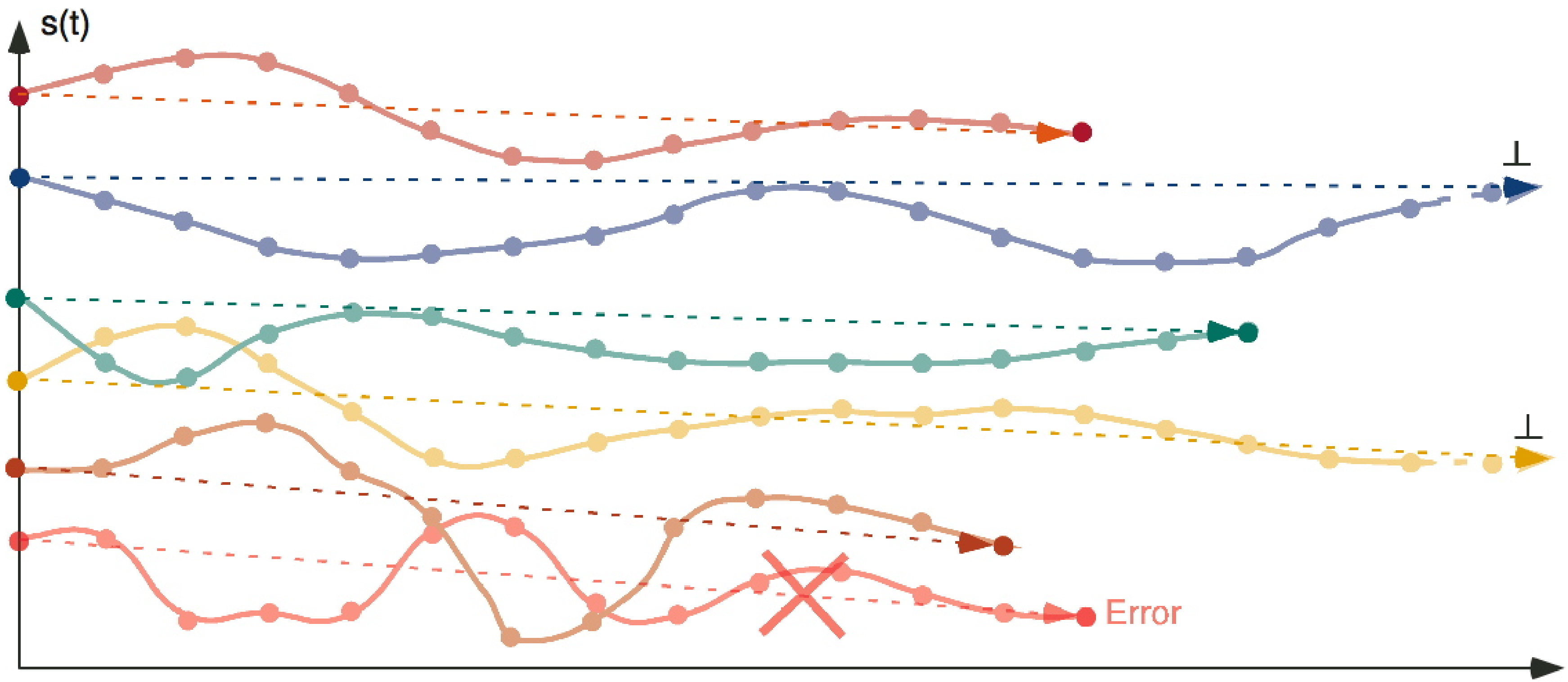
$$\begin{aligned}
\vec{F}(S) \triangleq\ & \{\mathsf{v} \in \overline{\mathbb{T}}^{\infty} \mid \mathsf{v} \in \mathbb{V}\} \cup \\
& \{(\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{a})\ \mathsf{v} \bullet \mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \bullet \sigma \mid \mathsf{v} \in \mathbb{V} \wedge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

Relational Semantics

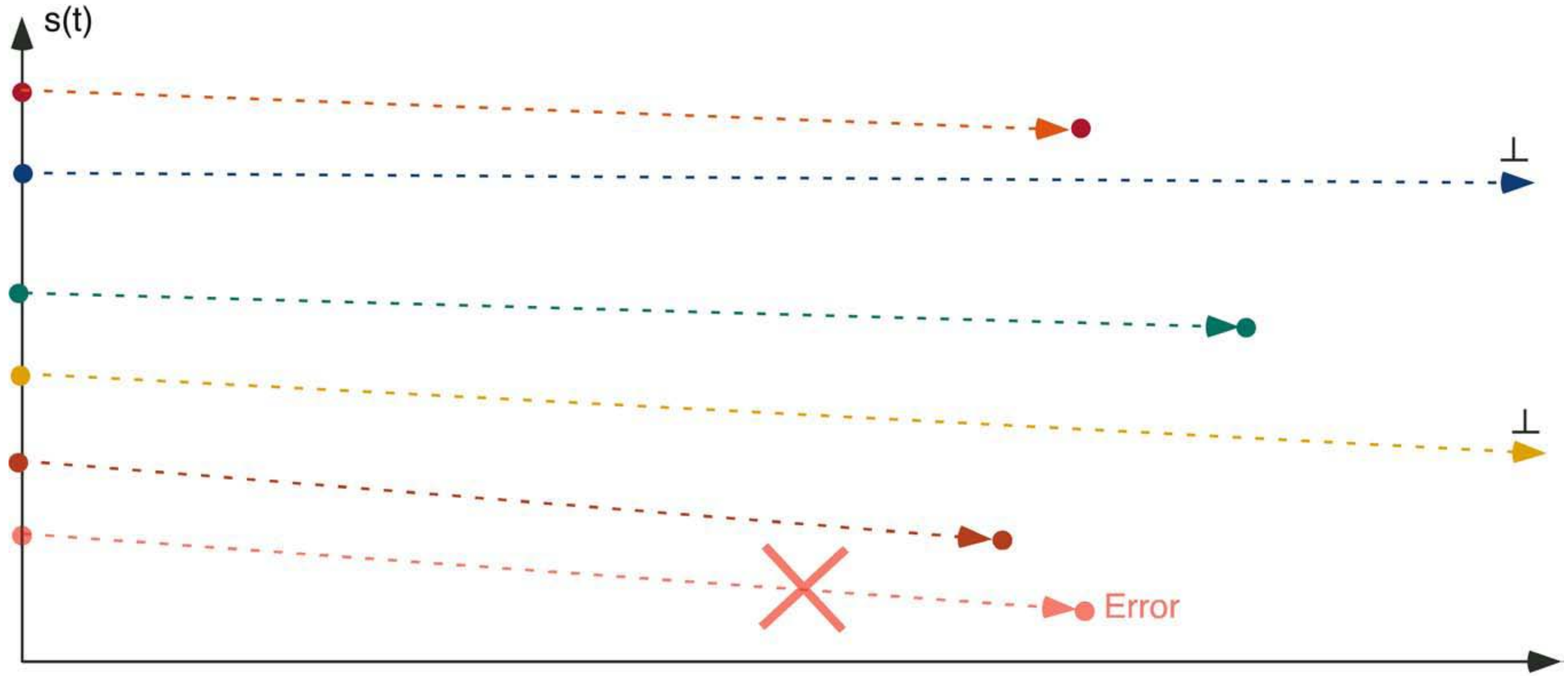
Trace Semantics



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

$$
\begin{aligned}
\alpha(T) &\stackrel{\text{def}}{=} \{\alpha(\sigma) \mid \sigma \in T\} \\
\alpha(\sigma_0 \bullet \sigma_1 \bullet \ldots \bullet \sigma_n) &\stackrel{\text{def}}{=} \sigma_0 \Longrightarrow \sigma_n \\
\alpha(\sigma_0 \bullet \ldots \bullet \sigma_n \bullet \ldots) &\stackrel{\text{def}}{=} \sigma_0 \Longrightarrow \bot
\end{aligned}
$$

Bifinitary Relational Semantics of the Eager λ-calculus

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

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

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

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

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

On the computational ordering $\sqsubseteq$

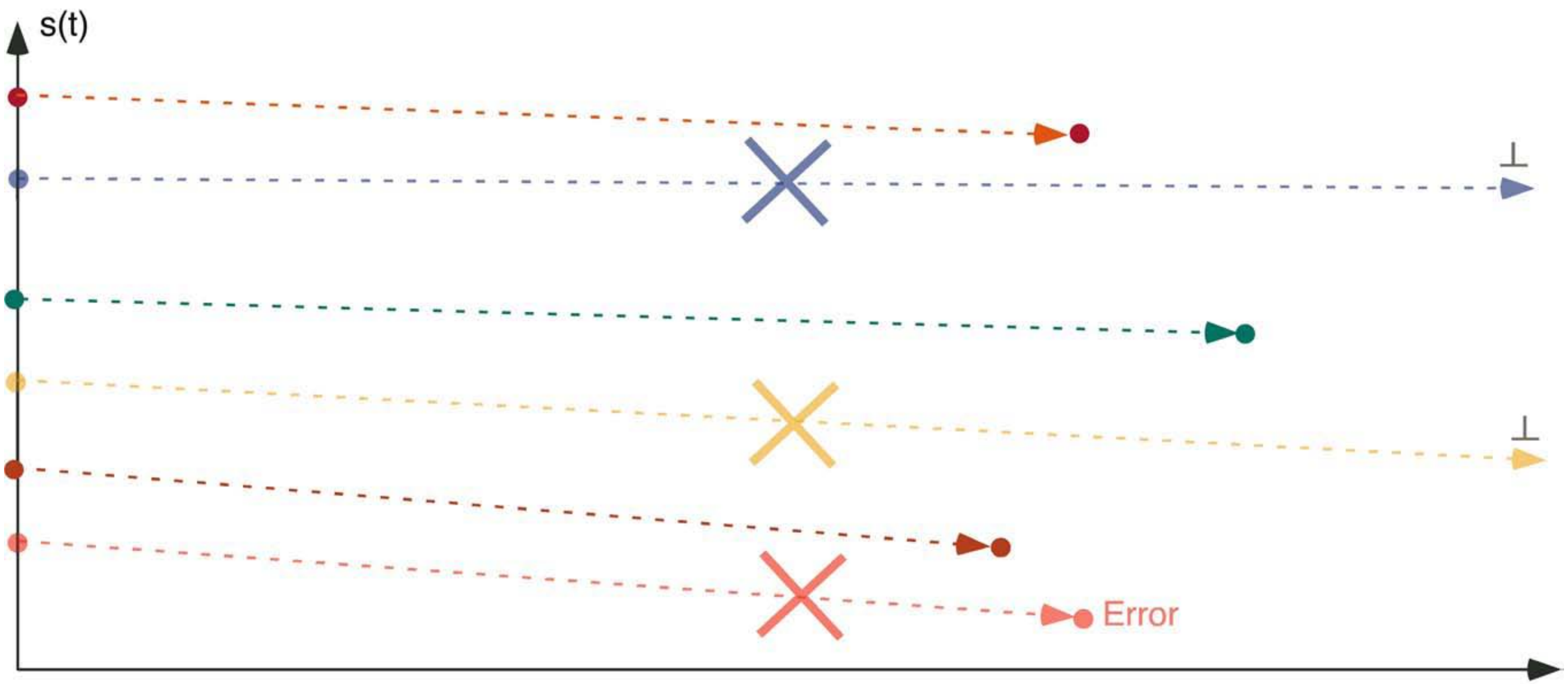
- For the bifinitary trace semantics $\vec{\mathbb{S}}$, we could replace the computational ordering $\sqsubseteq$ by $\supseteq$ (thus taking greatest fixpoints for $\subseteq$);
- Impossible for the bifinitary relational semantics!
- Counter-example: the greatest fixpoint starts by assuming that we have the terminating execution

$$(\boldsymbol{\lambda}\, \mathsf{x} \cdot \mathsf{x}\ \mathsf{x})(\boldsymbol{\lambda}\, \mathsf{x} \cdot \mathsf{x}\ \mathsf{x}) \Longrightarrow (\boldsymbol{\lambda}\, \mathsf{x} \cdot \mathsf{x}\ \mathsf{x})(\boldsymbol{\lambda}\, \mathsf{x} \cdot \mathsf{x}\ \mathsf{x})$$

then the call rule $\dfrac{\mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \Longrightarrow r}{(\boldsymbol{\lambda}\, \mathsf{x} \cdot \mathsf{a})\quad \mathsf{v} \Longrightarrow r}\sqsubseteq, \quad \mathsf{v} \in \mathbb{V},\ r \in \mathbb{V} \cup \{\bot\}$ will preserve this invalid hypothesis!

Natural Semantics

Natural Semantics = α(Relational Semantics)



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

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

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

$$\alpha(\sigma_0 \Longrightarrow \sigma_n) \stackrel{\text{def}}{=} \{\sigma_0 \Longrightarrow \sigma_n\}$$

$$\alpha(\sigma_0 \Longrightarrow \bot) \stackrel{\text{def}}{=} \varnothing$$

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

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

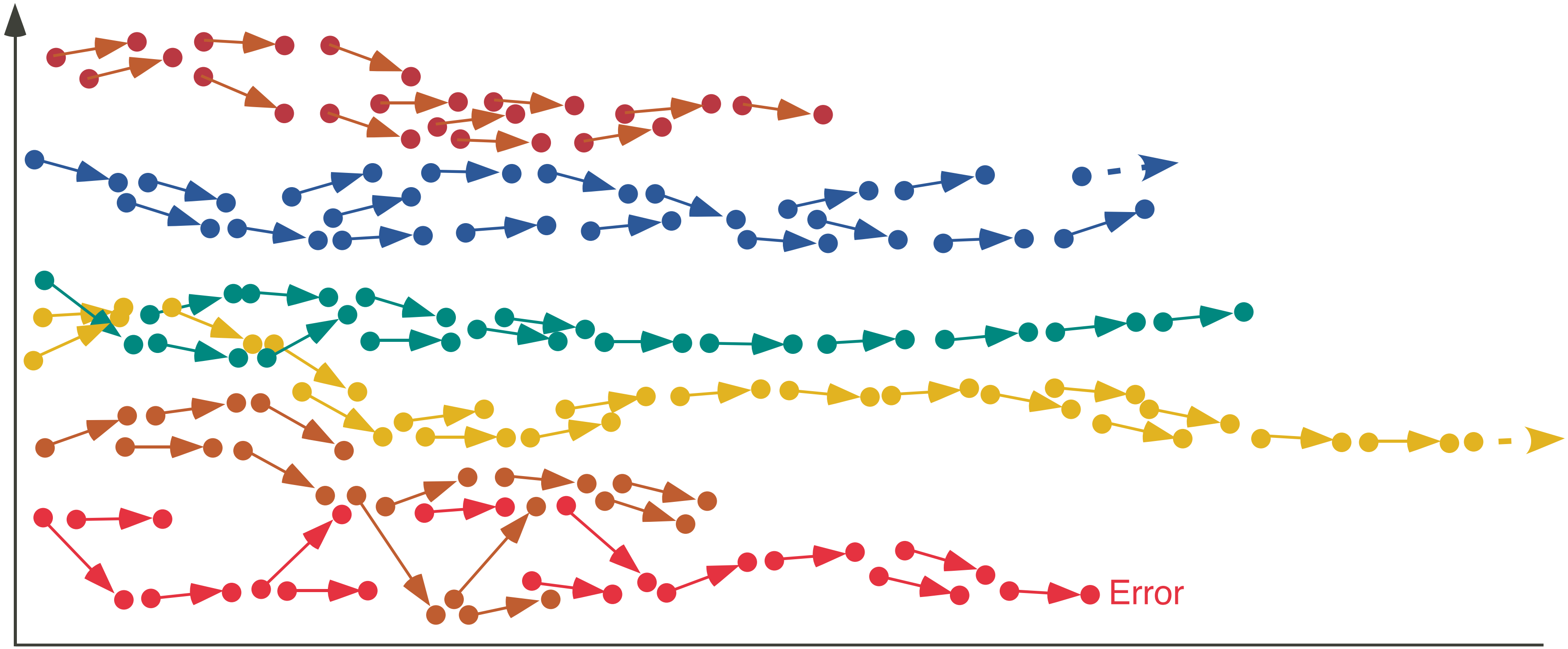
$$\frac{\mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \Longrightarrow r}{(\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{a}) \quad \mathsf{v} \Longrightarrow r} \subseteq, \quad \mathsf{v} \in \mathbb{V},\ r \in \mathbb{V}$$

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

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

Transition Semantics

Transition Semantics = α(Trace Semantics)



Abstraction to the Transition Semantics of the Eager λ-calculus

remember execution steps,
forget about their sequencing

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

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

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

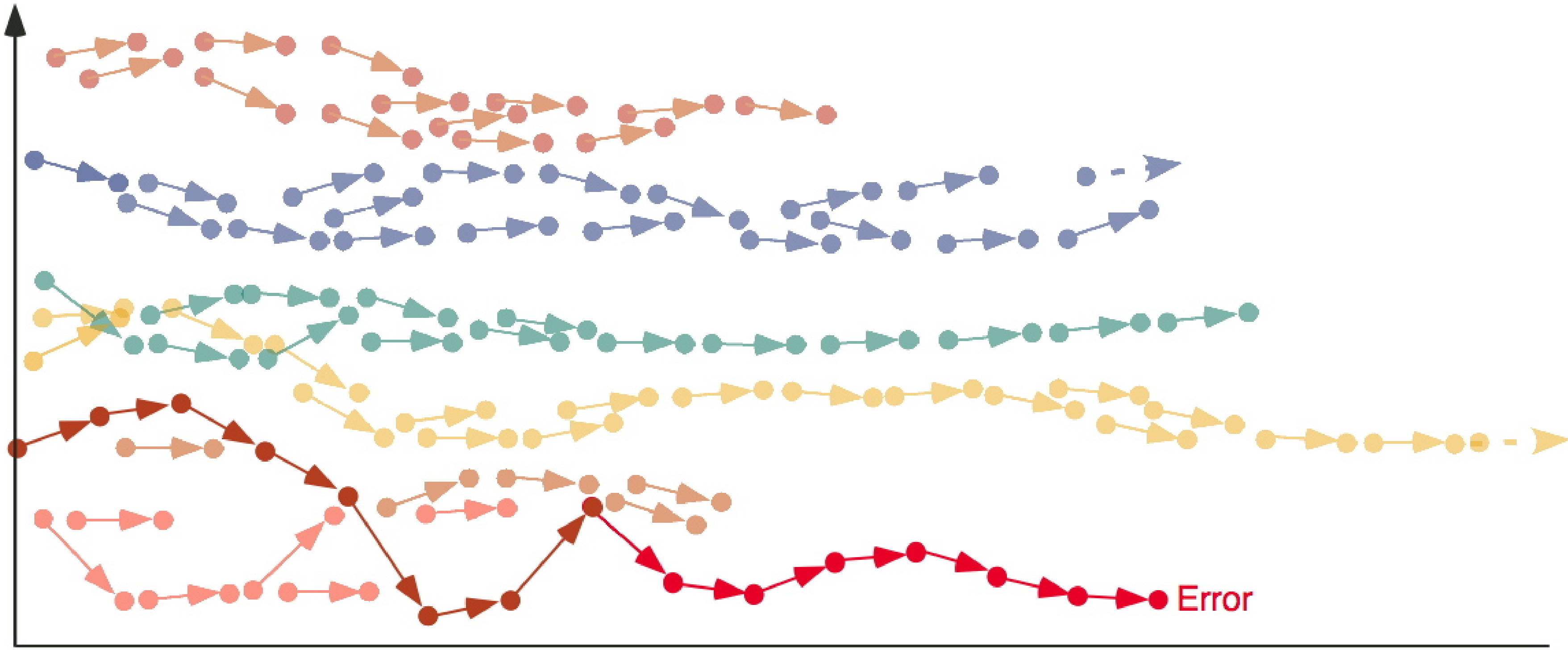
Transition Semantics of the Eager λ-calculus [Plo81]

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

$$\frac{a_0 \longrightarrow a_1}{a_0\ b \longrightarrow a_1\ b} \subseteq$$

$$\frac{b_0 \longrightarrow b_1}{v\ b_0 \longrightarrow v\ b_1} \subseteq .$$

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!*

Abstraction

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

- $\langle \mathcal{D}, \sqsubseteq, \perp, \sqcup \rangle$, $\langle \mathcal{D}^\sharp, \sqsubseteq^\sharp, \perp^\sharp, \sqcup^\sharp \rangle$ 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.

Conclusion

Conclusion

- Both finite and infinite semantics are needed in static analysis (such as strictness, [Myc80]), typing [Cou97, Ler06], etc;
- Such static analyzes must be proved correct with respect to a semantics chosen at an various level of abstraction (small-step/big-step trace/relational/natural semantics);
- Static analyzes use various equivalent presentations (fixpoints, equational, constraints and inference rules)
- The bifinite extension of SOS *might* satisfy these needs.

THE END, THANK YOU

Neil, for such a long friendship and cooperation

Best wishes for your new constraintless research career

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.

[Cou97] P. Cousot. Types as abstract interpretations, invited paper. In *24^{th} 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. 15^{th} 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. 4^{th} Int. Symp. on Programming*, Paris, FR, 22–24 Apr. 1980, LNCS 83, pages 270–281. Springer, 1980.

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