Bi-inductive Structural Semantics

(Extended Abstract)

Patrick Cousot[1]

Département d'informatique, École normale supérieure, 45 rue d'Ulm, 75230 Paris cedex 05, France

Radhia Cousot[2]

CNRS & École polytechnique, 91128 Palaiseau cedex, France

Abstract

We propose a simple order-theoretic generalization of set-theoretic inductive definitions. This generalization covers inductive, co-inductive and bi-inductive definitions and is preserved by abstraction. This allows the structural operational semantics to describe simultaneously the finite/terminating and infinite/diverging behaviors of programs. This is illustrated on the structural bifinitary small/big-step trace/relational/operational semantics of the call-by-value λ-calculus.

Keywords: fixpoint definition, inductive definition, co-inductive definition, bi-inductive definition, structural operational semantics, SOS, trace semantics, relational semantics, small-step semantics, big-step semantics, divergence semantics, abstraction.

1 Introduction

The connection between the use of fixpoints in *denotational semantics* [17] and the use of rule-based inductive definitions in *axiomatic semantics* [10] and *structural operational semantics* (SOS) [19,20,21] can be made by a generalization of inductive definitions [1] to include co-inductive definitions [8]. It is then possible to generalize *natural semantics* describing finite input/output behaviors [12] so as to also include infinite behaviors [7]. This is necessary since the definition of the infinite behaviors cannot be derived from the finite big-step SOS behaviors.

Example 1.1 Let us consider the choice operator $E_1 \mid E_2$ where the evaluation of E_1 either terminates (returning the value a, written $E_1 \Longrightarrow a$) or does not terminate (written $E_1 \Longrightarrow \bot$). Similarly, the big-step semantics of E_2 is $E_2 \Longrightarrow b$ for a

[1] Patrick.Cousot@ens.fr, www.enseignement.polytechnique.fr/profs/informatique/Radhia.Cousot/

[2] Radhia.Cousot@polytechnique.fr, www.di.ens.fr/~cousot/

terminating evaluation returning b or $E_2 \Longrightarrow \bot$ for non-termination. Let us consider several possible semantics for the choice operator:

- 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.

The corresponding finite big-step behaviors, as described in natural semantics [12], are all defined as follows:

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

But for the case $\bot \mid \bot \Longrightarrow \bot$, the infinite behaviors are all different:

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

Since the natural semantics defines the finite behaviors but not the diverging behaviors, an interpretation of the big-step evaluation rules as Horn clauses implemented in Prolog [2,9] will have its diverging behaviors determined by the implementation (e.g. Prolog interpreter with left-to-right evaluation). □

The paper develops and illustrates the use of "bi-inductive" definitions in operational semantics which enable both finitary and infinitary behaviors to be described simultaneously [7,8].

The general methodology consists in extending Hilbert proof systems [1] by replacing the powerset $\langle \wp(U), \subseteq \rangle$ of the universe U by a partial order $\langle \mathcal{D}, \sqsubseteq \rangle$. Beyond the classical inductive definitions $\langle \wp(U), \subseteq \rangle$, this extension includes the co-inductive definitions $\langle \wp(U), \supseteq \rangle$ and bi-inductive definitions mixing inductive and co-inductive definitions [7,8]. This extension also copes with compositional structural definitions as found in denotational semantics or SOS. This is illustrated by definitions of the semantics of the call-by-value λ-calculus.

We introduce an original big-step trace semantics that gives operational meaning to both convergent and divergent behaviors of programs. The compositional structural definition mixes induction for finite behaviors and co-induction for infinite behaviors while avoiding duplication of rules between the two cases. This big-step trace semantics excludes erroneous behaviors that go wrong. The other semantics are then systematically derived by abstraction.

The big-step trace semantics is first abstracted to a relational semantics and then to the standard big-step or natural semantics. These abstraction are sound and complete in that the big-step trace and relational semantics describe the same converging or diverging behaviors while the big-step trace and natural semantics describe the same finite behaviors. The big-step trace semantics is then abstracted into a small-step semantics, by collecting transitions along traces. This abstraction is sound but incomplete in that the traces generated by the small-step semantics describes convergent, divergent, but also erroneous behaviors of programs. This shows that trace-based operational semantics can be much more informative that small-step operational semantics.

2 Bi-inductive structural definitions and their abstraction

2.1 Structural order-theoretic inductive definitions

We introduce different forms of structural order-theoretic inductive definitions and prove their equivalence.

We formalize the *syntax* of a language $\mathbb{L}$ as a binary relation $\prec$ on $\mathbb{L}$ to be understood as the "strict syntactic subcomponent" relation on $\mathbb{L}$. $\langle \mathbb{L}, \prec\rangle$ is therefore a well-founded set, $\prec$ is irreflexive (inducing the reflexive $\preccurlyeq$), and $\prec$ has finite left images $\forall \ell \in \mathbb{L} : |\{\ell' \in \mathbb{L} \mid \ell' \prec \ell\}| \in \mathbb{N}$ ($|S|$ is the cardinality of set S, $\mathbb{N}$ is the set of natural numbers). Hence we can write $\ell ::= \ell_1, \ldots, \ell_n$ for the tuple of elements $\prod_{\ell' \prec \ell} \ell' = \ell_1, \ldots, \ell_n$ such that $\{\ell_1, \ldots, \ell_n\} = \{\ell' \in \mathbb{L} \mid \ell' \prec \ell\}$.

For example, for the language $\mathbb{L}$ of lambda terms a, b, ... $::= \mathsf{x} \mid \boldsymbol{\lambda}\mathsf{x} \bullet \mathsf{a} \mid \mathsf{a}\ \mathsf{b}$, we can define $\mathsf{a} \prec \boldsymbol{\lambda}\mathsf{x} \bullet \mathsf{a}$, $\mathsf{a} \prec \mathsf{a}\ \mathsf{b}$ and $\mathsf{b} \prec \mathsf{a}\ \mathsf{b}$ so $\mathsf{a}\ \mathsf{b} ::= \mathsf{a}, \mathsf{b}$. In case no structural i.e. syntax-directed reasoning is needed, $\mathbb{L}$ can be chosen as a singleton and $\prec$ as false.

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).

For each "syntactic component" $\ell \in \mathbb{L}$, we consider *variables* X_ℓ, Y_ℓ, ... ranging over the semantic domain $\mathcal{D}_\ell$. We drop the subscript ℓ when the corresponding semantic domain is clear from the context (e.g. the semantic domain is the same for all "syntactic components" i.e. $\forall \ell \in \mathbb{L} : \mathcal{D}_\ell = \mathcal{D}$).

For each "syntactic component" $\ell \in \mathbb{L}$, we let Δ_ℓ be indexed sequences (totally ordered sets). We write $\prod_{i \in \Delta_\ell} x_i$ when considering the sequence $\langle x_i,\ i \in \Delta_\ell\rangle \in \Delta_\ell \mapsto S$ of elements of a set S as a vector of $\prod_{i \in \Delta_\ell} S$.

For each element $i \in \Delta_\ell$ of the sequence, we consider transformers $F^i_\ell \in \mathcal{D}_\ell \times \mathcal{D}_{\ell_1} \ldots \times \mathcal{D}_{\ell_n} \longmapsto \mathcal{D}_\ell$ where $n = |\{\ell' \in \mathbb{L} \mid \ell' \prec \ell\}|$ and $\{\ell_1, \ldots, \ell_n\} = \{\ell' \in \mathbb{L} \mid \ell' \prec \ell\}$. 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, that is $\forall i \in \Delta_\ell,\ \ell_1, \ldots, \ell_n \prec \ell,\ X, Y \in \mathcal{D}_\ell, X_1 \in \mathcal{D}_{\ell_1}, \ldots, X_n \in \mathcal{D}_{\ell_n}$: $X \sqsubseteq_\ell Y \Longrightarrow F^i_\ell(X, X_1, \ldots, X_n) \sqsubseteq_\ell F^i_\ell(Y, X_1, \ldots, X_n)$.

For each "syntactic component" $\ell \in \mathbb{L}$, the *join* $\curlyvee_\ell \in (\Delta_\ell \longmapsto \mathcal{D}_\ell) \longmapsto \mathcal{D}_\ell$ is assumed to be componentwise $\sqsubseteq_\ell$-monotone $(\forall \langle X_i,\ i \in \Delta_\ell\rangle : \forall \langle Y_i,\ i \in \Delta_\ell\rangle : (\forall i \in$

$\Delta_\ell : X_i \sqsubseteq_\ell Y_i) \Longrightarrow \bigcurlyvee_\ell(\prod_{i\in\Delta_\ell} X_i) \sqsubseteq_\ell \bigcurlyvee_\ell(\prod_{i\in\Delta_\ell} Y_i))$. The join operator is used to gather alternatives in formal definitions. For brevity, we write $\curlyvee_\ell(\prod_{i\in\Delta_\ell} X_i) = \bigcurlyvee_{\ell\, i\in\Delta_\ell} X_i$, leaving implicit the fact that the X_i should be considered in the total order given by the sequence Δ_ℓ.

Most often, the order of presentation of these alternatives in the formal definition is not significant. In this case, Δ_ℓ is just a set and the join may often be defined in term of a *binary join* $\curlyvee_\ell \in (\mathcal{D}_\ell \times \mathcal{D}_\ell) \longmapsto \mathcal{D}_\ell$, which is assumed to be associative, commutative, and $\sqsubseteq_\ell$-monotone, as $\curlyvee_\ell(\prod_{i\in\Delta_\ell} X_i) \triangleq \bigcurlyvee_{\ell\, i\in\Delta_\ell} X_i$. The binary join may be different form the least upper bound (lub) $\sqcup_\ell$ of the semantic domain $\mathcal{D}_\ell$.

A *fixpoint definition* has the form

$$\forall \ell \in \mathbb{L} : \mathcal{S}_f[\![\ell]\!] = \mathbf{lfp}^{\sqsubseteq_\ell}\ \boldsymbol{\lambda} X \cdot \bigcurlyvee_{\ell\, i\in\Delta_\ell} F^i_\ell(X, \prod_{\ell'\prec\ell} \mathcal{S}_f[\![\ell']\!])$$

where $\mathbf{lfp}^{\sqsubseteq}$ is the partially defined $\sqsubseteq$-least fixpoint operator on a poset $\langle P,\ \sqsubseteq\rangle$. To emphasize structural composition, we also let $\{\ell_1, \ldots, \ell_n\} = \{\ell' \in \mathbb{L} \mid \ell' \prec \ell\}$ and write

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

Lemma 2.1 *$\forall \ell \in \mathbb{L} : \mathcal{S}_f[\![\ell]\!]$ is well defined.*

Definitions needing no fixpoint or join can withal be encompassed as fixpoints such as $\bigcurlyvee_{\ell\, i\in\Delta_\ell} F^i_\ell(\mathcal{S}_f[\![\ell_1]\!], \ldots, \mathcal{S}_f[\![\ell_n]\!]) = \mathbf{lfp}^{\sqsubseteq_\ell}\ \boldsymbol{\lambda} X \cdot \bigcurlyvee_{\ell\, i\in\Delta_\ell} F^i_\ell(\mathcal{S}_f[\![\ell_1]\!], \ldots, \mathcal{S}_f[\![\ell_n]\!])$ or without join $F^i_\ell(\mathcal{S}_f[\![\ell_1]\!], \ldots, \mathcal{S}_f[\![\ell_n]\!]) = \mathbf{lfp}^{\sqsubseteq_\ell}\ \boldsymbol{\lambda} X \cdot \bigcurlyvee_{\ell\, i'\in\{i\}} F^{i'}_\ell(\mathcal{S}_f[\![\ell_1]\!], \ldots, \mathcal{S}_f[\![\ell_n]\!])$.

An *equational 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 equations*

$$\begin{cases} X_\ell = \bigcurlyvee_{\ell\, i\in\Delta_\ell} F^i_\ell(X_\ell, \prod_{\ell'\prec\ell} X_{\ell'}) \\ \ell \in \mathbb{L} \quad . \end{cases}$$

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)*

$$\begin{cases} \bigcurlyvee_{\ell\, i\in\Delta_\ell} F^i_\ell(X_\ell, \prod_{\ell'\prec\ell} X_{\ell'}) \sqsubseteq_\ell X_\ell \\ \ell \in \mathbb{L} \quad . \end{cases}$$

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. When understanding the rule in logical form (where the premise is a statement that is assumed to be true and from which a conclusion can be drawn), the following form might be preferred.

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

If F^i_ℓ does not depend upon the premise X_ℓ, it is an axiom. In such presentations, the join $\curlyvee_\ell$ of the alternatives is left implicit [3]. To make it explicit, we rewrite such definitions in the form

$$\frac{X_\ell \sqsubseteq_\ell \mathcal{S}_r[\![\ell]\!]}{\bigcurlyvee_{i \in \Delta_\ell}^{\ell} F^i_\ell(X_\ell, \prod_{\ell' \prec \ell} \mathcal{S}_r[\![\ell']\!]) \sqsubseteq_\ell \mathcal{S}_r[\![\ell]\!]} \sqsubseteq_\ell \qquad \ell \in \mathbb{L},\ X_\ell \in \mathcal{D}_\ell\ . \tag{1}$$

The formal definition of the join makes explicit whether the order of presentation of the rules does matter, or not. When it doesn't, the join can be defined using a binary associative and commutative join. This binary join can even be left implicit and, by associativity and commutativity, the rules can be given in any order. This will be the case for our examples.

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 $0 < \delta \leqslant \lambda$, $D_\delta \sqsubseteq_\ell \bigcurlyvee_{i \in \Delta_\ell}^{\ell} F^i_\ell(\bigsqcup_{\beta<\delta}^{\ell} D_\beta, \prod_{\ell' \prec \ell} \mathcal{S}_r[\![\ell']\!])$.

The *meaning* of a rule-based definition (1) is

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

The above order-theoretic inductive definitions are all equivalent:

Theorem 2.2 $\forall \ell \in \mathbb{L} : \mathcal{S}[\![\ell]\!] \triangleq \mathcal{S}_f[\![\ell]\!] = \mathcal{S}_e[\![\ell]\!] = \mathcal{S}_c[\![\ell]\!] = \mathcal{S}_r[\![\ell]\!]$.

This generalization of [1] could also include a game-theoretic version. The closure-condition version [1] is also easy to adapt.

[3] This is the case in classical Hilbert's formal systems.

[4] In the classical case [1], the fixpoint operator is continuous whence proofs are finite.

Example 2.3 The classical inductive definition [1] of the subset $\mathcal{S}$ of a universe U by rules $\left\{\frac{P_i}{c_i} \;\middle|\; i \in I\right\}$ where $P_i \subseteq U$ and $c_i \in U$, $i \in I$ can be written $\dfrac{X \subseteq \mathcal{S}}{\{c_i \mid P_i \subseteq X\} \subseteq \mathcal{S}}\subseteq$, $i \in I$ or $\dfrac{P_i \subseteq X, \quad X \subseteq \mathcal{S}}{c_i \in \mathcal{S}}\subseteq$, $i \in I$ that is $\dfrac{P_i \subseteq \mathcal{S}}{c_i \in \mathcal{S}}\subseteq$, $i \in I$ for short. So $\langle \mathbb{L}, \preccurlyeq\rangle \triangleq \langle \bullet, =\rangle$, $\langle \mathcal{D}_\bullet, \sqsubseteq_\bullet, \bot_\bullet, \sqcup_\bullet\rangle \triangleq \langle \wp(\mathcal{U}), \subseteq, \varnothing, \cup\rangle$, $\Delta_\bullet \triangleq I$, $F^i_\bullet \in \wp(U) \mapsto \wp(U)$ is $F^i_\bullet(X) \triangleq \{c_i \mid P_i \subseteq X\}$ and $\curlyvee_\bullet \triangleq \bigcup$ thus defining $\mathcal{S} = \mathbf{lfp}^{\subseteq}\ \boldsymbol{\lambda} X \cdot \{c_i \mid i \in I \wedge P_i \subseteq X\}$. □

2.2 Bi-semantic domains

To account for terminating/finite and diverging/infinite program behaviors, we consider *bi-semantic domains* consisting, for each $\ell \in \mathbb{L}$, of a finitary semantic domain (of finite program behaviors) $\langle \mathcal{D}^+_\ell, \sqsubseteq^+_\ell, \bot^+_\ell, \bigsqcup^+_\ell\rangle$ and a infinitary semantic codomain (of infinite program behaviors) $\langle \mathcal{D}^-_\ell, \sqsubseteq^-_\ell, \bot^-_\ell, \bigsqcup^-_\ell\rangle$ which are assumed to be dcpos [17] (respectively complete lattices). They are combined into a bi-semantic domain (of bifinite program behaviors) $\mathcal{D}_\ell$ thanks to a projection $\pi^+_\ell \in \mathcal{D}_\ell \mapsto \mathcal{D}^+_\ell$, a coprojection $\pi^-_\ell \in \mathcal{D}_\ell \mapsto \mathcal{D}^-_\ell$, and a constructor $\pi_\ell \in \mathcal{D}^+_\ell \times \mathcal{D}^-_\ell \mapsto \mathcal{D}_\ell$ satisfying $\forall x \in \mathcal{D}^+_\ell, y \in \mathcal{D}^-_\ell : \pi^+_\ell(\pi_\ell(x, y)) = x$ and $\pi^-_\ell(\pi_\ell(x, y)) = y$ while $\forall X \in \mathcal{D} : \pi_\ell(\pi^+_\ell(X), \pi^-_\ell(X)) = X$. Examples are the Cartesian product, disjoint union or union of disjoint sets. The bi-semantic domain $\langle \mathcal{D}_\ell, \sqsubseteq_\ell, \bot_\ell, \sqcup_\ell\rangle$ is then a dcpo (respectively a complete lattice) by defining $X^+ \triangleq \pi^+_\ell(X)$, $X^- \triangleq \pi^-_\ell(X)$, $X \sqsubseteq_\ell Y \triangleq (X^+ \sqsubseteq^+_\ell Y^+) \wedge (X^- \sqsubseteq^-_\ell Y^-)$, and $\bigsqcup_{\ell\, i\in I} X_i \triangleq \pi_\ell(\bigsqcup^+_{\ell\, i\in I} X^+_i, \bigsqcup^-_{\ell\, i\in I} X^-_i)$.

2.3 Abstraction

We consider a simple form of abstraction based on a continuous abstraction function α [6], which includes the particular case of a Galois connection [5] (denoted $\langle P, \preccurlyeq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Q, \sqsubseteq\rangle$, or $\langle P, \preccurlyeq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows\!\!\!\!\twoheadrightarrow}} \langle Q, \sqsubseteq\rangle$ when α is onto, where $\langle P, \preccurlyeq\rangle$ and $\langle Q, \sqsubseteq\rangle$ are posets, and $\forall x \in P : \forall y \in Q : \alpha(x) \sqsubseteq y \Longleftrightarrow x \preccurlyeq \gamma(y)$).

For all $\ell \in \mathbb{L}$, we let $\langle \overline{\mathcal{D}}_\ell, \overline{\sqsubseteq}_\ell, \overline{\bot}_\ell, \overline{\sqcup}_\ell\rangle$ be dcpos, $\overline{F}^i_\ell \in \overline{\mathcal{D}}_\ell \times \overline{\mathcal{D}}_{\ell_1} \ldots \times \overline{\mathcal{D}}_{\ell_n} \longmapsto \overline{\mathcal{D}}_\ell$ $i \in \Delta_\ell$ be monotone in their first parameter, and define the abstract semantics $\overline{\mathcal{S}}_f[\![\ell]\!]$ in one of the equivalent forms of **Th. 2.2**.

If $\alpha_\ell \in \mathcal{D}_\ell \longmapsto \overline{\mathcal{D}}_\ell$, we say that the abstract semantics $\langle\overline{\mathcal{S}}[\![\ell]\!], \ell \in \mathbb{L}\rangle$ is *sound* with respect to the concrete semantics $\langle \mathcal{S}[\![\ell]\!], \ell \in \mathbb{L}\rangle$ if and only if $\forall \ell \in \mathbb{L} : \alpha_\ell(\mathcal{S}[\![\ell]\!]) \overline{\sqsubseteq}_\ell \overline{\mathcal{S}}[\![\ell]\!]$. If is *complete* whenever $\forall \ell \in \mathbb{L} : \overline{\mathcal{S}}[\![\ell]\!] \overline{\sqsubseteq}_\ell \alpha_\ell(\mathcal{S}[\![\ell]\!])$.

3 Structural order-theoretic inductive definitions of the semantics of the call-by-value λ-calculus

The syntax of the λ-calculus with constants is

$$\begin{array}{rcll} \mathsf{x}, \mathsf{y}, \mathsf{z}, \ldots & \in & \mathbb{X} & \text{variables} \\ \mathsf{c} & \in & \mathbb{C} & \text{constants } (\mathbb{X} \cap \mathbb{C} = \varnothing) \\ \mathsf{c} & ::= & \mathsf{0} \mid \mathsf{1} \mid \ldots & \end{array}$$

$$
\begin{array}{rcll}
\mathsf{v} & \in & \mathbb{V} & \text{values} \\
\mathsf{v} & ::= & \mathsf{c} \mid \boldsymbol{\lambda}\,\mathsf{x} \bullet \mathsf{a} & \\
\mathsf{e} & \in & \mathbb{E} & \text{errors} \\
\mathsf{e} & ::= & \mathsf{c}\ \mathsf{a} \mid \mathsf{e}\ \mathsf{a} & \\
\mathsf{a}, \mathsf{a}', \mathsf{a}_1, \ldots, \mathsf{b}, , \ldots & \in & \mathbb{T} & \text{terms} \\
\mathsf{a} & ::= & \mathsf{x} \mid \mathsf{v} \mid \mathsf{a}\ \mathsf{a}' &
\end{array}
$$

We write $\mathsf{a}[\mathsf{x} \leftarrow \mathsf{b}]$ for the capture-avoiding substitution of b for all free occurrences 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) $\overline{\mathbb{T}} \triangleq \{\mathsf{a} \in \mathbb{T} \mid \mathrm{FV}(\mathsf{a}) = \varnothing\}$.

The application $(\boldsymbol{\lambda}\,\mathsf{x} \bullet \mathsf{a}\ \mathsf{v})$ of a function $\boldsymbol{\lambda}\,\mathsf{x} \bullet \mathsf{a}$ to a value v is evaluated by substitution $\mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}]$ of the actual parameter v for the formal parameter x in the function body a. This cannot be understood as induction on the program syntax since $\mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}]$ is not in general a strict syntactic subcomponent of $(\boldsymbol{\lambda}\,\mathsf{x} \bullet \mathsf{a}\ \mathsf{v})$. Hence the various semantics below cannot be defined by structural induction of the syntax of λ-expressions. So the framework of Sect. 2.1 is instantiated with $\mathbb{L} = \{\bullet\}$ and $\prec$ is defined to be false on $\mathbb{L}$ which prevents the use of structural induction on program syntax. For brevity we omit the void syntactic component $\bullet$ writing e.g. F for $F[\![\bullet]\!]$, $\mathcal{D}$ for $\mathcal{D}_\bullet$, Δ for $\Delta_\bullet$, etc.

We introduce a maximal trace semantics describing terminating and diverging computations. The trace semantics is then abstracted into a relational [20] and then an operational semantics [15]. Each semantics can be defined using small steps or big steps of computation. Each semantics can be defined in fixpoint or rule-based form.

Semantics		Fixpoint definition		Rule-based definition	
		big-step	small-step	big-step	small-step
Trace	$\vec{\mathbb{S}}$	$\mathbf{lfp}^{\sqsubseteq}\, \vec{F}$	$\mathbf{lfp}^{\sqsubseteq}\, \vec{f}$	$\vec{\Longmapsto}$	$\vec{\twoheadrightarrow}$
Relational	$\overset{\frown}{\mathbb{S}}$	$\mathbf{lfp}^{\sqsubseteq}\, \overset{\frown}{F}$	$\mathbf{lfp}^{\sqsubseteq}\, \overset{\frown}{f}$	$\Longrightarrow$	$\twoheadrightarrow$
Operational	$\mathbb{S}$		$\mathbf{lfp}^{\subseteq}\, f = \mathbf{gfp}^{\subseteq}\, f$		$\longrightarrow$.

4 Big-step maximal trace semantics of the call-by-value λ-calculus

We let $\mathbb{T}^\star$ (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 where ϵ is the empty sequence $\epsilon \bullet \sigma = \sigma \bullet \epsilon = \sigma$. We let $|\sigma| \in \mathbb{N} \cup \{\omega\}$ be 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 $\sigma = \sigma_0 \bullet \ldots \bullet \sigma_n \bullet \ldots$. Given $S, T \in \wp(\mathbb{T}^\infty)$, we define $S^+ \triangleq S \cap \mathbb{T}^+$, $S^\omega \triangleq S \cap \mathbb{T}^\omega$ and $S \sqsubseteq T \triangleq S^+ \subseteq T^+ \wedge S^\omega \supseteq T^\omega$, so that the *trace domain* $\langle \wp(\mathbb{T}^\infty), \sqsubseteq, \mathbb{T}^\omega, \mathbb{T}^+, \sqcup, \sqcap \rangle$ is a complete lattice. For $\mathsf{a} \in \mathbb{T}$ and $\sigma \in \mathbb{T}^\infty$, we define $a@\sigma$ to be $\sigma' \in \mathbb{T}^\infty$ such that $\forall i < |\sigma| : \sigma'_i = a\ \sigma_i$ and similarly $\sigma@a$ is σ' such that $\forall i < |\sigma| : \sigma'_i = \sigma_i\ a$.

4.1 Fixpoint big-step maximal trace semantics

The bifinitary trace semantics $\vec{\mathbb{S}} \in \wp(\overline{\mathbb{T}}^{\infty})$ of the closed call-by-value λ-calculus $\overline{\mathbb{T}}$ can be specified in fixpoint form

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

where the set of traces transformer $\vec{F} \in \wp(\overline{\mathbb{T}}^{\infty}) \mapsto \wp(\overline{\mathbb{T}}^{\infty})$ describes big steps of computation

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

The definition of $\vec{F}$ has (a) for termination, (b) for call-by-value β-reduction, (c) and (d) for left reduction under applications and (e) and (f) for right reduction under applications, corresponding to left-to-right evaluation. (b), (d) and (f) cope both with terminating and diverging traces. In the framework of Sect. 2.1, we have $\Delta_{\bullet} \triangleq \{a, b, c, d, e, f\}$ where $\vec{F}^{i}_{\bullet}(S)$, $i \in \Delta_{\bullet}$ is defined by equation (i). The join operator is chosen in binary form as $\curlyvee_{\bullet} \triangleq \cup$.

We observe that ($S^{+} \triangleq S \cap \mathbb{T}^{+}$, $S^{\omega} \triangleq S \cap \mathbb{T}^{\omega}$ so $S^{+} \cap S^{\omega} = \emptyset$)

$$
(2)\quad \begin{cases}
\vec{\mathbb{S}} = \vec{\mathbb{S}}^{+} \cup \vec{\mathbb{S}}^{\omega} \\
\vec{\mathbb{S}}^{+} = \vec{F}(\vec{\mathbb{S}}^{+}) = \mathbf{lfp}^{\subseteq} \vec{F}^{+} \quad \text{where} \quad \vec{F}^{+}(S) \triangleq \vec{F}(S^{+}) \\
\vec{\mathbb{S}}^{\omega} = (\vec{F}(\vec{\mathbb{S}}^{+} \cup \vec{\mathbb{S}}^{\omega}))^{\omega} = \mathbf{gfp}^{\subseteq} \vec{F}^{\omega} \quad \text{where} \quad \vec{F}^{\omega}(S) \triangleq (\vec{F}(\vec{\mathbb{S}}^{+} \cup S^{\omega}))^{\omega}\ .
\end{cases}
$$

The bifinitary trace semantics $\vec{\mathbb{S}}$ is *suffix-closed* in that

$$\forall \sigma \in \mathbb{T}^{\infty} : \mathsf{a} \bullet \sigma \in \vec{\mathbb{S}} \Longrightarrow \sigma \in \vec{\mathbb{S}}\ .$$

The bifinitary trace semantics $\vec{\mathbb{S}}$ is *total* in that it excludes intermediate or result errors

$$\forall \mathsf{a} \in \overline{\mathbb{T}} : \nexists \sigma, \sigma' \in \overline{\mathbb{T}}^{\infty}, \mathsf{e} \in \mathbb{E} : \mathsf{a} \bullet \sigma \bullet \mathsf{e} \bullet \sigma' \in \vec{\mathbb{S}}\ .$$

The finite maximal traces are *blocking* in that the result of a finite computation is always a final value

$$\forall \sigma \in \mathbb{T}^{\infty} \cup \{\epsilon\} : \sigma \bullet \mathsf{b} \in \vec{\mathbb{S}}^{+} \Longrightarrow \mathsf{b} \in \mathbb{V}\ .$$

4.2 Rule-based big-step maximal trace semantics

The maximal trace semantics $\vec{\mathbb{S}}$ can also be defined as follows

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

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

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

Defining $\vec{\mathbb{S}}[\![\mathsf{a}]\!] \triangleq \{\mathsf{a} \bullet \sigma \mid \mathsf{a} \bullet \sigma \in \vec{\mathbb{S}}\}$, $\vec{\mathbb{S}}^{+}[\![\mathsf{a}]\!] \triangleq \{\mathsf{a} \bullet \sigma \mid \mathsf{a} \bullet \sigma \in \vec{\mathbb{S}}^{+}\}$, and $\vec{\mathbb{S}}^{\omega}[\![\mathsf{a}]\!] \triangleq \{\mathsf{a} \bullet \sigma \mid \mathsf{a} \bullet \sigma \in \vec{\mathbb{S}}^{\omega}\}$, we can also write for brevity

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

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

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

Observe that the inductive definition of $\vec{\mathbb{S}}[\![\mathsf{a}]\!]$ should neither be understood as a structural induction on a (since $\mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \not\prec (\boldsymbol{\lambda}\,\mathsf{x} \cdot \mathsf{a})\ \mathsf{v}$) nor as *action induction* [16] (because of infinite traces). The definition could be split in inductive rules for termination and co-inductive rules for divergence, as shown in (2), but the above bi-inductive definition avoids the duplication of common rules. Defining $\mathsf{a} \Longmapsto \sigma \triangleq \sigma \in \vec{\mathbb{S}}[\![\mathsf{a}]\!]$, we can also write

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

$$\frac{\mathsf{a} \Longmapsto \sigma}{\mathsf{a}\ \mathsf{b} \Longmapsto \sigma@\mathsf{b}} \sqsubseteq, \quad \sigma \in T^{\omega} \qquad \frac{\mathsf{a} \Longmapsto \sigma \bullet \mathsf{v}, \quad \mathsf{v}\ \mathsf{b} \Longmapsto \sigma'}{\mathsf{a}\ \mathsf{b} \Longmapsto (\sigma@\mathsf{b}) \bullet \sigma'} \sqsubseteq, \quad \mathsf{v} \in \mathbb{V}, \sigma \in T^{+}$$

$$\frac{\mathsf{b} \Longmapsto \sigma}{\mathsf{a}\ \mathsf{b} \Longmapsto \mathsf{a}@\sigma} \sqsubseteq, \quad \mathsf{a} \in \mathbb{V}, \sigma \in T^{\omega} \qquad \frac{\mathsf{b} \Longmapsto \sigma \bullet \mathsf{v}, \quad \mathsf{a}\ \mathsf{v} \Longmapsto \sigma'}{\mathsf{a}\ \mathsf{b} \Longmapsto (\mathsf{a}@\sigma) \bullet \sigma'} \sqsubseteq, \quad \mathsf{a}, \mathsf{v} \in \mathbb{V}, \sigma \in T^{+}\ .$$

5 Abstraction of the big-step trace semantics into the big-step relational semantics of the call-by-value λ-calculus

The *relational abstraction* of sets of traces is

$$
\begin{aligned}
(3)\qquad \alpha &\in \wp(\mathbb{T}^\infty) \mapsto \wp(\mathbb{T} \times (\mathbb{T} \cup \{\bot\}))\\
\alpha(S) &\triangleq \{\langle \sigma_0,\, \sigma_{n-1}\rangle \mid \sigma \in S \wedge |\sigma| = n\} \cup \{\langle \sigma_0,\, \bot\rangle \mid \sigma \in S \wedge |\sigma| = \omega\}\\
\gamma &\in \wp(\mathbb{T} \times (\mathbb{T} \cup \{\bot\})) \mapsto \wp(\mathbb{T}^\infty)\\
\gamma(T) &\triangleq \{\sigma \in \mathbb{T}^\infty \mid (|\sigma| = n \wedge \langle \sigma_0,\, \sigma_{n-1}\rangle \in T) \vee (|\sigma| = \omega \wedge \langle \sigma_0,\, \bot\rangle \in T)\}
\end{aligned}
$$

so that

$$\langle \wp(\mathbb{T}^\infty),\, \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(\mathbb{T} \times (\mathbb{T} \cup \{\bot\})),\, \subseteq\rangle\ .$$

The *bifinitary relational semantics* $\widehat{\vec{\mathbb{S}}} \triangleq \alpha(\vec{\mathbb{S}}) \in \wp(\mathbb{T} \times (\mathbb{T} \cup \{\bot\}))$ is the relational abstraction of the trace semantics mapping an expression to its final value or $\bot$ in case of divergence.

5.1 Fixpoint big-step bifinitary relational semantics

The bifinitary relational semantics $\widehat{\vec{\mathbb{S}}} \triangleq \alpha(\vec{\mathbb{S}}) = \alpha(\mathbf{lfp}^{\sqsubseteq}\, \vec{F})$ can be defined in fixpoint form as $\mathbf{lfp}^{\sqsubseteq}\, \widehat{\vec{F}}$ where the big-step transformer $\widehat{\vec{F}} \in \wp(\mathbb{T} \times (\mathbb{T} \cup \{\bot\})) \mapsto \wp(\mathbb{T} \times (\mathbb{T} \cup \{\bot\}))$ is

$$
\begin{aligned}
\widehat{\vec{F}}(T) \triangleq\ & \{\langle \mathsf{v},\, \mathsf{v}\rangle \mid \mathsf{v} \in \mathbb{V}\} \cup\\
& \{\langle (\boldsymbol{\lambda}\,\mathsf{x} \bullet \mathsf{a})\ \mathsf{v},\, r\rangle \mid \mathsf{v} \in \mathbb{V} \wedge \langle \mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}],\, r\rangle \in T\} \cup\\
& \{\langle (\mathsf{a}\ \mathsf{b}),\, \bot\rangle \mid \langle \mathsf{a},\, \bot\rangle \in T\} \cup\\
& \{\langle (\mathsf{a}\ \mathsf{b}),\, r\rangle \mid \langle \mathsf{a},\, \mathsf{v}\rangle \in T^+ \wedge \mathsf{v} \in \mathbb{V} \wedge \langle (\mathsf{v}\ \mathsf{b}),\, r\rangle \in T\} \cup\\
& \{\langle (\mathsf{a}\ \mathsf{b}),\, \bot\rangle \mid \mathsf{a} \in \mathbb{V} \wedge \langle \mathsf{b},\, \bot\rangle \in T\} \cup\\
& \{\langle (\mathsf{a}\ \mathsf{b}),\, r\rangle \mid \mathsf{a}, \mathsf{v} \in \mathbb{V} \wedge \langle \mathsf{b},\, \mathsf{v}\rangle \in T^+ \wedge \langle (\mathsf{a}\ \mathsf{v}),\, r\rangle \in T\}\ .
\end{aligned}
$$

Theorem 5.1 *We have $\alpha(\vec{F}(S)) = \widehat{\vec{F}}(\alpha(S))$ and so $\widehat{\vec{\mathbb{S}}} \triangleq \alpha(\vec{\mathbb{S}}) = \alpha(\mathbf{lfp}^{\sqsubseteq}\, \vec{F}) = \mathbf{lfp}^{\sqsubseteq}\, \widehat{\vec{F}}$.*

5.2 Rule-based big-step bifinitary relational semantics

The big-step bifinitary relational semantics $\Longrightarrow$ is defined as $\mathsf{a} \Longrightarrow r \triangleq \langle \mathsf{a},\, r\rangle \in \alpha(\vec{\mathbb{S}}[\![\mathsf{a}]\!])$ where $\mathsf{a} \in \mathbb{T}$ and $r \in \mathbb{T} \cup \{\bot\}$. It is

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

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

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

Again this should neither be understood as a structural induction (since $\mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \not\prec (\boldsymbol{\lambda}\,\mathsf{x} \bullet \mathsf{a})\ \mathsf{v}$) nor as action induction (because of infinite behaviors). The abstraction $\alpha(T) \triangleq T \cap (\mathbb{T} \times \mathbb{T})$ yields the classical natural semantics [12] (where all rules with $\bot$ are eliminated and $\sqsubseteq$ becomes $\subseteq$ in the remaining ones). The abstraction $\alpha(T) \triangleq T \cap (\mathbb{T} \times \{\bot\})$ yields the divergence semantics (keeping only the rules with $\bot$, $\sqsubseteq$ is $\supseteq$, and $\mathsf{a} \Longrightarrow \bot$ is written $\mathsf{a} \overset{\infty}{\Longrightarrow}$ in [15]).

Observe that both the maximal trace semantics of *Sec. 4.1* and the above bifinitary relational semantics of *Sec. 5* define the semantics of a term that "goes wrong" as empty.

The above big-step bifinitary relational semantics $\Longrightarrow$ is equivalent but not identical to the standard big-step semantics which bifinitary generalization would be

$$\mathsf{v} \Longrightarrow \mathsf{v}, \quad \mathsf{v} \in \mathbb{V} \qquad \frac{\mathsf{a} \Longrightarrow \boldsymbol{\lambda}\,\mathsf{x} \bullet \mathsf{c}, \quad \mathsf{b} \Longrightarrow \mathsf{v}', \quad \mathsf{c}[\mathsf{x} \leftarrow \mathsf{v}'] \Longrightarrow r}{\mathsf{a}\ \mathsf{b} \Longrightarrow r}\sqsubseteq, \quad \begin{array}{l}\mathsf{v}, \mathsf{v}' \in \mathbb{V},\\ r \in \mathbb{V} \cup \{\bot\}\end{array}$$

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

We have chosen to break evaluations of applications in smaller chunks instead so as to enforce evaluation of the function before that of the arguments and to make explicit the reduction step in the trace semantics.

6 Abstraction of the big-step trace semantics into the small-step operational semantics of the call-by-value λ-calculus

The one-step reduction semantics abstracts the trace semantics by collecting all transitions along any trace.

The *small-step abstraction of traces* is

$$\begin{aligned}\alpha_s &\in \wp(\mathbb{T}^\infty) \mapsto \wp(\mathbb{T} \times \mathbb{T})\\ \alpha_s(S) &\triangleq \{\langle \sigma_i,\ \sigma_{i+1}\rangle \mid \sigma \in S \wedge 0 \leqslant i \wedge i + 1 < |\sigma|\}\ .\end{aligned}$$

Since the bifinitary trace semantics is suffix-closed, we can also use

$$\begin{aligned}\alpha &\in \wp(\mathbb{T}^\infty) \mapsto \wp(\mathbb{T} \times \mathbb{T})\\ \alpha(S) &\triangleq \{\langle \sigma_0,\ \sigma_1\rangle \mid \sigma \in S \wedge |\sigma| > 1\}\end{aligned}$$

so that we have $\alpha_s(S) = \alpha(S)$ whenever S is suffix-closed. By defining $\overline{\wp}(\mathbb{T}^\infty)$ to be the set of suffix-closed and blocking subsets of $\mathbb{T}^\infty$ and $\gamma(\tau)$ to be the set of maximal traces generated by the transition relation $\tau \in \wp(\mathbb{T} \times \mathbb{T})$ that is

$$
\begin{aligned}
\gamma^+(\tau) &\triangleq \{\sigma \in \mathbb{T}^+ \mid \forall i < |\sigma| : \langle \sigma_i,\ \sigma_{i+1}\rangle \in \tau \wedge \forall \mathsf{a} \in \mathbb{T} : \langle \sigma_{<|\sigma|-1},\ \mathsf{a}\rangle \notin \tau\}\\
\gamma^\omega(\tau) &\triangleq \{\sigma \in \mathbb{T}^\omega \mid \forall i \in \mathbb{N} : \langle \sigma_i,\ \sigma_{i+1}\rangle \in \tau\}\\
\gamma(\tau) &\triangleq \gamma^+(\tau) \cup \gamma^\omega(\tau)\ ,
\end{aligned}
$$

we have

$$\langle \overline{\wp}(\mathbb{T}^\infty),\ \subseteq\rangle \xleftarrow[\alpha]{\gamma}\!\!\twoheadrightarrow \langle \wp((\mathbb{T} \setminus \mathbb{V}) \times \mathbb{T}),\ \subseteq\rangle\ .$$

6.1 Small-step operational semantics

The small-step operational semantics or transition semantics $\mathbb{S}$ is defined by α-overapproximation $\alpha_s(\vec{\mathbb{S}}) = \alpha(\vec{\mathbb{S}})$ of the bifinitary trace semantics $\vec{\mathbb{S}}$.

$$
\begin{aligned}
&\text{(4)} \qquad \mathbb{S} \triangleq \mathbf{lfp}^{\subseteq}\ f\\
&f(\tau) \triangleq \{\langle ((\boldsymbol{\lambda}\,\mathsf{x} \centerdot \mathsf{a})\ \mathsf{v},\ \mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}]\rangle\} \cup \{\langle \mathsf{a}_0\ \mathsf{b},\ \mathsf{a}_1\ \mathsf{b}\rangle \mid \langle \mathsf{a}_0,\ \mathsf{a}_1\rangle \in \tau\} \cup\\
&\qquad\quad \{\langle \mathsf{v}\ \mathsf{b}_0,\ \mathsf{v}\ \mathsf{b}_1\rangle \mid \langle \mathsf{b}_0,\ \mathsf{b}_1\rangle \in \tau\}\ .
\end{aligned}
$$

The rule-based presentation of (4) has a call-by-value β-reduction axiom plus two context rules for reducing under applications, corresponding to left-to-right evaluation [20]. $\mathsf{a} \longrightarrow \mathsf{b}$ stands for $\langle \mathsf{a},\ \mathsf{b}\rangle \in \mathbb{S}$.

$$((\boldsymbol{\lambda}\,\mathsf{x} \centerdot \mathsf{a})\ \mathsf{v}) \longrightarrow \mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}] \qquad \frac{\mathsf{a}_0 \longrightarrow \mathsf{a}_1}{\mathsf{a}_0\ \mathsf{b} \longrightarrow \mathsf{a}_1\ \mathsf{b}}\subseteq \qquad \frac{\mathsf{b}_0 \longrightarrow \mathsf{b}_1}{\mathsf{v}\ \mathsf{b}_0 \longrightarrow \mathsf{v}\ \mathsf{b}_1}\subseteq\ .$$

The inductive definition of $\mathbb{S}$ can also be understood as co-inductive since $\mathbf{lfp}^{\subseteq}\ f = \mathbf{gfp}^{\subseteq}\ f$.

We have $\alpha \circ \vec{F} \circ \gamma \mathrel{\dot{\subseteq}} f$. Indeed $\alpha \circ \vec{F} \circ \gamma \mathrel{\dot{\subsetneq}} f$ since a single transition cannot anticipate whether the future computation can "go wrong". For example $((\boldsymbol{\lambda}\,\mathsf{x} \centerdot \mathsf{x}\ 0)\ 0) \longrightarrow (0\ 0) \in f \circ f(\varnothing)$ while $((\boldsymbol{\lambda}\,\mathsf{x} \centerdot \mathsf{x}\ 0)\ 0) \longrightarrow (0\ 0) \notin \alpha \circ \vec{F} \circ \gamma \circ \alpha \circ \vec{F} \circ \gamma(\varnothing)$ since there is no trace of the form $\sigma \bullet ((\boldsymbol{\lambda}\,\mathsf{x} \centerdot \mathsf{x}\ 0)\ 0) \bullet (0\ 0) \bullet \sigma'$ in $\vec{F} \circ \gamma \circ \alpha \circ \vec{F} \circ \gamma(\varnothing)$. It follows that the small-step operational semantics or transition semantics $\mathbb{S}$ is sound but incomplete in that the set $\gamma(\mathbb{S})$ of maximal traces generated by the transition relation $\mathbb{S}$ includes the bifinitary trace semantics $\vec{\mathbb{S}}$ plus spurious traces for computations that can "go wrong" that is terminate with a runtime error $\mathsf{e} \in \mathbb{E}$.

7 Small-step maximal trace semantics of the call-by-value λ-calculus

The small-step maximal trace semantics $\overset{\infty}{\longrightarrow}$ of a transition relation $\longrightarrow$ is defined as

$$\begin{array}{lll}
\xrightarrow{n}\!\!\!\!\!\!\rightharpoonup \triangleq \{\sigma \in \mathbb{T}^{+} \mid |\sigma| = n > 0 \wedge \forall i : 0 \leqslant i < n-1 : \sigma_i \longrightarrow \sigma_{i+1}\} & & \text{partial traces} \\
\xrightarrow{n} \triangleq \{\sigma \in \overset{n}{\twoheadrightarrow} \mid \sigma_{n-1} \in \mathbb{V}\} & & \text{maximal execution traces of length } n \\
\xrightarrow{+} \triangleq \bigcup_{n>0} \xrightarrow{n} & & \text{maximal finite execution traces} \\
\xrightarrow{\omega} \triangleq \{\sigma \in \mathbb{T}^{\omega} \mid \forall i \in \mathbb{N} : \sigma_i \longrightarrow \sigma_{i+1}\} & & \text{infinite execution traces} \\
\xrightarrow{\infty} \triangleq \xrightarrow{+} \cup \xrightarrow{\omega} & & \text{maximal finite and diverging execution traces.}
\end{array}$$

7.1 Fixpoint small-step maximal trace semantics

To express the small-step maximal trace semantics $\xrightarrow{\infty}$ in fixpoint form, let us define the junction $\fatsemi$ of set of traces as

$$S \fatsemi T \triangleq S^{\omega} \cup \{\sigma_0 \bullet \ldots \bullet \sigma_{|\sigma|-2} \bullet \sigma' \mid \sigma \in S^{+} \wedge \sigma_{|\sigma|-1} = \sigma'_0 \wedge \sigma' \in T\} \ ,$$

and the small-step set of traces transformer $\vec{f} \in \wp(\overline{\mathbb{T}}^{\infty}) \mapsto \wp(\overline{\mathbb{T}}^{\infty})$

$$\vec{f}(T) \triangleq \{\mathsf{v} \in \overline{\mathbb{T}}^{\infty} \mid \mathsf{v} \in \mathbb{V}\} \cup \overset{2}{\twoheadrightarrow} \fatsemi T \tag{5}$$

describing small steps of computation. We have

$$\xrightarrow{\infty} = \mathbf{lfp}^{\sqsubseteq} \vec{f} \ .$$

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

$$\vec{\mathbb{S}} = \xrightarrow{\infty} \ .$$

7.2 Rule-based small-step maximal trace semantics

The maximal trace semantics $\vec{\mathbb{S}} = \xrightarrow{\infty} = \mathbf{lfp}^{\sqsubseteq} \vec{f}$ where $\vec{f}$ is defined by (5) can be defined inductively with small-steps as

$$\mathsf{v} \in \vec{\mathbb{S}}, \quad \mathsf{v} \in \mathbb{V} \qquad \frac{\mathsf{a} \longrightarrow \mathsf{b}, \quad \mathsf{b} \bullet \sigma \in \vec{\mathbb{S}}}{\mathsf{a} \bullet \mathsf{b} \bullet \sigma \in \vec{\mathbb{S}}} \sqsubseteq$$

that is, writing $\mathsf{a} \Mapsto \sigma$ for $\sigma \in \vec{\mathbb{S}}$ and $\sigma_0 = \mathsf{a}$

$$\mathsf{v} \Mapsto \mathsf{v}, \quad \mathsf{v} \in \mathbb{V} \qquad \frac{\mathsf{a} \longrightarrow \mathsf{b}, \quad \mathsf{b} \Mapsto \sigma}{\mathsf{a} \Mapsto \mathsf{a} \bullet \sigma} \sqsubseteq$$

8 Small-step bifinitary relational semantics of the call-by-value λ-calculus

The bifinitary relational semantics was defined as $\widehat{\mathbb{S}} \triangleq \alpha(\vec{\mathbb{S}})$ (where α is the relational abstraction of sets of traces (3)) and given in big-step form in *Sec. 5*. It can be given in small-step form by abstraction of the small-step bifinitary maximal trace semantics of *Sec. 7.1*.

8.1 Fixpoint small-step bifinitary relational semantics

The bifinitary relational semantics $\overset{\frown}{\mathbb{S}} \triangleq \alpha(\vec{\mathbb{S}}) = \alpha(\mathbf{lfp}^{\sqsubseteq}\ \vec{f})$ can be defined in fixpoint form as $\mathbf{lfp}^{\sqsubseteq}\ \overset{\frown}{f}$ where the small-step transformer $\overset{\frown}{f} \in \wp(\mathbb{T} \times (\mathbb{T} \cup \{\perp\})) \mapsto \wp(\mathbb{T} \times (\mathbb{T} \cup \{\perp\}))$ is

$$\begin{aligned}\overset{\frown}{f}(R) \triangleq\ & \{\langle \mathsf{v},\ \mathsf{v}\rangle \mid \mathsf{v} \in \mathbb{V}\} \cup \\ & \{\langle (\boldsymbol{\lambda}\,\mathsf{x} \bullet \mathsf{a})\ \mathsf{v},\ r\rangle \mid \mathsf{v} \in \mathbb{V} \wedge \langle \mathsf{a}[\mathsf{x} \leftarrow \mathsf{v}],\ r\rangle \in R\} \cup \\ & \{\langle \mathsf{a}_0\ \mathsf{b},\ r\rangle \mid \mathsf{a}_0 \longrightarrow \mathsf{a}_1 \wedge \langle \mathsf{a}_1\ \mathsf{b},\ r\rangle \in R\} \cup \\ & \{\langle \mathsf{v}\ \mathsf{b}_0,\ r\rangle \mid \mathsf{b}_0 \longrightarrow \mathsf{b}_1 \wedge \langle \mathsf{v}\ \mathsf{b}_1,\ r\rangle \in R\}\ .\end{aligned}$$

8.2 Rule-based small-step bifinitary relational semantics

The bifinitary rule-base form is ($\mathsf{a} \Rrightarrow \mathsf{b}$ stands for $\langle \mathsf{a},\ \mathsf{b}\rangle \in \overset{\frown}{\mathbb{S}}$ and $r \in \mathbb{V} \cup \{\perp\}$)

$$\mathsf{v} \Rrightarrow \mathsf{v}, \quad \mathsf{v} \in \mathbb{V} \qquad \frac{\mathsf{a} \longrightarrow \mathsf{b}, \quad \mathsf{b} \Rrightarrow r}{\mathsf{a} \Rrightarrow r} \sqsubseteq$$

9 Conclusion

Divergence/nonterminating behaviors are needed in static program analysis [18] or typing [3,15]. Such divergence information is part of the classical order-theoretic fixpoint denotational semantics [17] but not explicit in small-step/abstract-machine-based operational semantics [19,20,21] and absent of big-step/natural operational semantics [12]. A standard approach is therefore to generate an execution trace semantics from a (labelled) transition system/small-step operational semantics, using either an order-theoretic [4] or metric [23] fixpoint definition or else a categorical definition as a final coalgebra for a behaviour functor (modeling the transition relation) up to a weak bisimulation [11,14,22] or using an equational definition for recursion in an order-enriched category [13]. However, execution traces are not always at an appropriate level of abstraction. Finite and infinite behaviors can be both handled by SOS when extended to bi-inductive structural bifinitary small/big-step trace/relational/operational semantics. Sound (and sometimes complete) abstractions are essential to establish this hierarchy of semantics [4]. This 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.

Acknowledgements

We thank the anonymous referees for their helpful comments and suggestions.

References

[1] 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.

[2] I. Attali, J. Chazarain, and S. Gilette. Incremental evaluation of natural semantics specifications. In M. Bruynooghe and M. Wirsing, editors, *Proc. 4th Int. Symp. PLILP '92*, Leuven, BE, 26–28 Aug. 1992, LNCS 631, pages 87–99. Springer, 1992.

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

[4] P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoret. Comput. Sci.*, 277(1—2):47–103, 2002.

[5] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *6th POPL*, pages 269–282, San Antonio, TX, 1979. ACM Press.

[6] P. Cousot and R. Cousot. Abstract interpretation frameworks. *J. Logic and Comp.*, 2(4):511–547, Aug. 1992.

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

[8] P. Cousot and R. Cousot. Compositional and inductive semantic definitions in fixpoint, equational, constraint, closure-condition, rule-based and game-theoretic form, invited paper. In P. Wolper, editor, *Proc. 7th Int. Conf. CAV '95*, Liège, BE, LNCS 939, pages 293–308. Springer, 3–5 Jul. 1995.

[9] Th. Despeyroux. TYPOL: a formalism to implement natural semantics. Tech. rep.RT-0094, INRIA Sophia Antipolis, Mar. 1988.

[10] C.A.R. Hoare. An axiomatic basis for computer programming. *Comm. ACM*, 12(10):576–580, Oct. 1969.

[11] B. Jacobs and J. Rutten. A tutorial on (co)algebras and (co)induction. *EATCS Bulletin*, 62:222–269, 1997.

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

[13] B. Klin. Adding recursive constructs to bialgebraic semantics. *J. Logic and Alg. Prog.*, 60-61:259–286, Jul. -Dec. 2004.

[14] B. Klin. Bialgebraic methods in structural operational semantics. *ENTCS*, 175(1):33–43, May 2007.

[15] 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.

[16] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, chapter 19, pages 1201–1242. Elsevier, 1990.

[17] P.D. Mosses. Denotational semantics. In J. van Leeuwen, editor, *Formal Models and Semantics*, volume B of *Handbook of Theoretical Computer Science*, chapter 11, pages 575–631. Elsevier, 1990.

[18] 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.

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

[20] G.D. Plotkin. The origins of structural operational semantics. *J. Logic and Alg. Prog.*, 60–61:3–15, Jul. -Dec. 2004.

[21] G.D. Plotkin. A structural approach to operational semantics. *J. Logic and Alg. Prog.*, 60–61:17–139, Jul. -Dec. 2004.

[22] D. Turi and G.D. Plotkin. Towards a mathematical operational semantics. In *Proc. 12th LICS '1997*, pages 280–291, Warsaw, PL, 29 June – 2 Jul. 1997. IEEE Comp. Soc. Press.

[23] F. van Breugel. An introduction to metric semantics: operational and denotational models for programming and specification languages. *Theoret. Comput. Sci.*, 258:1–98, ě 2001.