

Bisimulation for pure untyped $\lambda\mu$ -calculus (Extended Abstract)

S. B. Lassen

Soeren.Lassen@cl.cam.ac.uk

<http://www.cl.cam.ac.uk/~sb121/>

University of Cambridge Computer Laboratory

January 6, 1999

Abstract

This paper develops two bisimulation theories for the pure untyped $\lambda\mu$ -calculus of Parigot. The calculus extends Church's λ -calculus with control operators similar to `call/cc` in Scheme.

The first bisimulation theory is an extension of Ong's Lévy-Longo tree theory for the pure λ -calculus. Following Sangiorgi, the theory is presented co-inductively in terms of bisimulation. The second bisimulation theory is an extension of Abramsky's theory of applicative bisimulation for functional languages. Both are 'lazy' call-by-name theories, based on reduction to weak head normal form.

The development of the theories is operationally-based. First, weak head reduction and evaluation to weak head normal form are characterised in terms of new small-step and big-step structural operational semantics. They are used to give elementary congruence proofs for the bisimulation equivalences. The bisimulation principles appear to be the first co-inductive operational theory for a λ -calculus with control operators.

1 Introduction and related work

Parigot's $\lambda\mu$ -calculus [19] extends the λ -calculus with first-class access to the flow of control, or the current continuation. It was introduced as a term assignment for natural deduction proofs in classical logic. Following Griffin [9], it is one

of several typed λ -calculi with control operators to extend the Curry-Howard correspondence between typed λ -calculus and logic, or the formulae-as-types principle, from intuitionistic logic to classical logic.

This paper is concerned with the untyped calculus, using it as a vehicle for exploring the operational theory of control operators. The $\lambda\mu$ -calculus is chosen because its particular syntax is convenient for expressing the useful, new small-step and big-step structural operational semantics (SOS) that are present in §3. They have had a significant impact on the development of the bisimulation theories in this paper: first, they suggested a definitions of bisimulation for the calculus; second, they are key to the central congruence proofs. Similar, but rather more complex, SOS were presented by Ong and Stewart [18] for a simply-typed call-by-value $\lambda\mu$ -calculus. For the same calculus, de Groote [5] presents a small-step weak head reduction relation both as global rewrite axioms and in a rule-based form. These two presentations lead to different relations which only correspond on well-typed programs. In earlier work, Felleisen et al [6, 29] give small-step operational semantics for control operators as rewrite systems expressed in terms of evaluation contexts (a syntactic representation of continuations). These rewrite semantics are not "structural" in the style of Plotkin's SOS [20]. Other operational semantics of control operators and the $\lambda\mu$ -calculus [6, 26, 27, 25, 4, 5, 24] are in the form of abstract machines operating on configurations (e.g., tuples of terms, stacks, and

environments) which are different from the pure syntax. None of the many alternative operational semantics for control operators would support the development of the bisimulation theories as well as the SOS presented here.

The first bisimulation theory presented in this paper extends the Lévy–Longo tree theory for the untyped λ -calculus, following Sangiorgi’s co-inductive formulation as “open bisimulation” [22]. Lévy–Longo trees for untyped λ -terms were introduced in [16]. They are a lazy variant of Barendregt’s Böhm trees [3] (where *lazy* refers to its relationship with Abramsky and Ong’s ‘lazy’ λ -theories, based on weak head normal forms [2], as opposed to Barendregt and Wadsworth’s ‘sensible’ λ -theories [3, 28], based on head normal forms). The Lévy–Longo tree theory is the equivalence relation on λ -terms induced by Lévy–Longo tree equality. This theory was developed by Ong [17] along the lines of Barendregt’s [3] development of the Böhm tree theory. The central congruence proofs of the induced tree equivalences between λ -terms are semantically-based, either by connections to domain-theoretic models of the λ -calculus or by continuity properties of the tree models. In [14], a novel operationally-based development of the tree theories is presented, including elementary congruence proofs for the tree equivalences, based on an operational bisimulation account of the tree equivalences. Here, this development is extended to the $\lambda\mu$ -calculus.

The second bisimulation theory presented in this paper is an adaptation of Abramsky’s applicative bisimulation to the $\lambda\mu$ -calculus. It is a coarser (larger) theory than the open bisimulation theory. Existing operational results about program equivalence in λ -calculi with control operators are based on confluent reduction calculi [7, 19] or so-called CIU context lemmas [26, 27, 4]. Reduction calculi are finer-grained (smaller) theories than the bisimulation theory. In general, they do not identify objects with bisimilar infinite behaviour, e.g., different fixed point combinators in the pure calculus or infinite streams. CIU theorems are context lemmas that characterise contextual equivalence in terms of a restricted set of term contexts. It remains to investigate their rela-

tionship with the applicative bisimulation in this paper. An example in §5 suggests that applicative bisimulation is better for some problems.

Outline The paper is organised as follows. §2 introduces the syntax and reductions of the $\lambda\mu$ -calculus. §3 presents new small-step and big-step operational semantics for the calculus and a proof of their correspondence. §4 and §5 develop two bisimulation theories for the calculus, an extension of Sangiorgi’s open bisimulation and an adaptation of Abramsky’s applicative bisimulation, respectively. The main results are that the two bisimulation equivalences are congruences.

2 The $\lambda\mu$ -calculus

This section introduces the syntax and reductions of the pure $\lambda\mu$ -calculus.

Let M , N and P range over terms of the calculus. From the pure λ -calculus we have variables x, y, z, \dots (an infinite supply is assumed), λ -abstractions $\lambda x. M$, and function application $M N$. The $\lambda\mu$ -calculus adds μ -abstraction and *naming* in one construct, $\mu\alpha. [\beta]M$, where $\alpha, \beta, \gamma, \dots$ range over a separate infinite set of variables called *names*.

$$\begin{aligned} \text{Terms } M, N, P ::= & x \mid \lambda x. M \\ & \mid M N \mid \mu\alpha. [\beta]M \end{aligned}$$

It is convenient to introduce an auxiliary syntactic category of named terms of the form $[\beta]M$, ranged over by A and B . They are disjoint from terms. Then we can see μ -abstraction, $\mu\alpha. A$, and naming, $[\beta]M$, and separate syntactic constructors, and express the syntax grammar as a mutually inductive definition:

$$\begin{aligned} \text{Terms } M, N, P ::= & x \mid \lambda x. M \\ & \mid M N \mid \mu\alpha. A \\ \text{Named terms } A, B ::= & [\beta]M \end{aligned}$$

The scope of λ -abstraction, μ -abstraction, and naming all extend as far to the right as possible. Application associates to the left. For instance, $\lambda x. \mu\alpha. [\beta]M N P = \lambda x. (\mu\alpha. [\beta]((M N) P))$.

We have the usual notion of free and bound variables from the λ -calculus. Let $\text{FV}(M)$ and $\text{FV}(A)$ denote the sets of free variables in M and A , respectively. A term or named term with no free variables is said to be closed. We have similar notions of free and bound names: in a named term $[\beta]M$, β is a free name; μ -abstraction, $\mu\alpha. A$, binds the name α . We let $\text{FN}(M)$ and $\text{FN}(A)$ denote the sets of free names in M and A , respectively. Terms are identified up to renaming of bound variables and names.

Variables range over terms. The usual capture-avoiding substitution of a term N for free occurrences of x in a term M or a named term A is denoted $M[N/x]$ and $A[N/x]$, respectively. Syntactically, names range over named term contexts with a hole, \bullet , as a place holder for an (unnamed) term—more specifically, names range over (named) applicative contexts \mathbb{A} , generated by the grammar:

$$\text{Applicative contexts } \mathbb{A} ::= [\alpha]\bullet\vec{N}$$

A named term $[\alpha]M$ can be read as the term M filled in for the hole in the context α (so $\alpha[M]$ would have been a more fitting syntax). The (second-order) substitution of an applicative context \mathbb{A} for a name α in the term M is written $M\langle\mathbb{A}/\alpha\rangle$, or just $M\langle\beta/\alpha\rangle$ if $\mathbb{A} = [\beta]\bullet$; and similarly for substitution into named terms. The full definition is as follows.

$$\begin{aligned} x\langle\mathbb{A}/a\rangle &\stackrel{\text{def}}{=} x \\ (\lambda x. M)\langle\mathbb{A}/a\rangle &\stackrel{\text{def}}{=} \lambda x. (M\langle\mathbb{A}/a\rangle) \\ (\mu b. A)\langle\mathbb{A}/a\rangle &\stackrel{\text{def}}{=} \mu b. (A\langle\mathbb{A}/a\rangle), \quad \text{where } b \neq a \\ (MN)\langle\mathbb{A}/a\rangle &\stackrel{\text{def}}{=} (M\langle\mathbb{A}/a\rangle)(N\langle\mathbb{A}/a\rangle) \\ ([b]M)\langle\mathbb{A}/a\rangle &\stackrel{\text{def}}{=} \begin{cases} \mathbb{A}[M\langle\mathbb{A}/a\rangle], & \text{if } b = a \\ [b](M\langle\mathbb{A}/a\rangle), & \text{if } b \neq a \end{cases} \end{aligned}$$

Operationally, names range over continuations: μ -abstraction, $\mu\alpha. [\beta]M$, binds the name α to the current continuation and then passes the result of M to the continuation bound to β . The reduc-

tion rules of the calculus are:

$$\begin{aligned} (\lambda x. M) N &\rightarrow M[N/x] & (\beta) \\ (\mu\alpha'. A) N &\rightarrow \mu\alpha. A\langle[\alpha]\bullet N/[\alpha']\bullet\rangle & (\mu) \\ \mu\alpha. [\beta]\mu\gamma. A &\rightarrow \mu\alpha. A\langle\beta/\gamma\rangle & (\rho) \\ \mu\alpha. [\alpha]M &\rightarrow M, \quad \text{if } a \notin \text{FN}(M) & (\theta) \end{aligned}$$

Let \rightarrow denote the compatible closure of the four reductions. Parigot [19] shows that the reflexive, transitive closure \rightarrow^* is confluent. However, this result and, indeed, the reduction relation itself, will not be needed for the operational development in the remainder of the paper, but we shall compare the bisimulation equivalences that will be introduced later on against the reduction axioms.

Remark 1. The (ρ) axiom is defined on terms, as in [5], rather than on named terms as in Parigot's simpler formulation: $[\beta]\mu\gamma. A \rightarrow A\langle\beta/\gamma\rangle$. In the present formulation, reduction is expressed on terms only, in keeping with the other relations which will be introduced in sequel.

3 Operational semantics

This section defines small-step and big-step operational semantics, specifying weak head reduction to weak head normal form. A proof of correspondence between the two is given.

First, we define a weak head transition relation between named terms as follows:

$$\begin{aligned} (\text{trans.}\beta\textcircled{a}) \quad &[\alpha](\lambda y. M) N\vec{P} \mapsto_{\text{wh}} [\alpha]M[N/y]\vec{P} \\ (\text{trans.}\mu\rho) \quad &[\alpha](\mu\beta. A) \vec{P} \mapsto_{\text{wh}} A\langle[\alpha]\bullet\vec{P}/[\beta]\bullet\rangle \end{aligned}$$

Recall the definition of weak head normal forms (whnf's) from the pure λ -calculus:

$$\text{whnf's } W ::= x\vec{N} \mid \lambda x. M$$

The normal forms of the weak head reduction relation on named terms are exactly the named whnf's:

$$\text{Named whnf's } \Upsilon, \Psi ::= [\alpha]W$$

Remark 2. $A \mapsto_{\text{wh}} B$ implies $\mu\alpha. A \rightarrow^+ \mu\alpha. B$, for any α , so a transition sequence can be seen as a “coarse-grained” reduction sequence. If one wishes, one can derive a transition relation $>$ on terms by the two rules:

$$\frac{[\alpha]M \mapsto_{\text{wh}} [\alpha]N}{M > N} \quad \alpha \notin \text{FN}(MN)$$

$$\frac{[\alpha]M \mapsto_{\text{wh}} B}{\mu\beta. [\alpha]M > \mu\beta. B} \quad \beta \neq \alpha \text{ or } \alpha \in \text{FN}(M)$$

This corresponds to a (coarse-grained) call-by-name version of Ong and Stewart’s transition relation in [18]. One notices that if M is a λ -term (contains no μ -abstractions) then, by (trans. $\beta\textcircled{\@}$), $[a]M \mapsto_{\text{wh}} [a]N$ and $M > N$ if and only if M reduces to N by a conventional weak head reduction in the λ -calculus.

The weak head transition relation on named terms can be specified as a small-step structural operational semantics, given by the structurally inductive axioms and rule:

$$\text{(trans.}\beta\text{)} \quad [\alpha](\lambda y. M) N \mapsto_{\text{wh}} [\alpha]M[N/y]$$

$$\text{(trans.}\rho\text{)} \quad [\alpha]\mu\beta. A \mapsto_{\text{wh}} A\langle\alpha/\beta\rangle$$

$$\text{(trans.}\textcircled{\@}\text{)} \quad \frac{[\alpha']M \mapsto_{\text{wh}} B, \alpha' \notin \text{FN}(M)}{[\alpha]MN \mapsto_{\text{wh}} B\langle[\alpha]\bullet N/[\alpha']\bullet\rangle}$$

Lemma 3.1. *For all named terms A and B , $A \mapsto_{\text{wh}} B$ can be derived from (trans. β), (trans. ρ) and (trans. $\textcircled{\@}$) if and only if $A \mapsto_{\text{wh}} B$ is an instance of (trans. $\beta\textcircled{\@}$) or (trans. $\mu\rho$).*

Proof. The forward direction is by induction on the derivation of $A \mapsto_{\text{wh}} B$. The reverse direction is by induction on the length of \vec{P} in (trans. $\beta\textcircled{\@}$) and (trans. $\mu\rho$). \square

Next, we introduce a big-step evaluation semantics of reduction to whnf. It is specified as a ternary relation, $A \rightsquigarrow_{\text{wh}}^p \Psi$, between named terms A , non-negative integers p , and named whnf’s Ψ .

The integer p counts the number of transitions in the derivation. The relation is defined inductively by the following rules.

$$\text{(eval.}\lambda\text{)} \quad [\alpha]\lambda x. M \overset{0}{\rightsquigarrow}_{\text{wh}} [\alpha]\lambda x. M$$

$$\text{(eval.}\rho\text{)} \quad \frac{A\langle\alpha/\beta\rangle \overset{p}{\rightsquigarrow}_{\text{wh}} \Psi}{[\alpha]\mu\beta. A \overset{p+1}{\rightsquigarrow}_{\text{wh}} \Psi}$$

$$\text{(eval.}\textcircled{\@}\beta\text{)} \quad \frac{[\alpha']M \overset{p_1}{\rightsquigarrow}_{\text{wh}} [\alpha']\lambda x. P, \alpha' \notin \text{FN}(M) \quad P\langle[\alpha]\bullet N/[\alpha']\bullet\rangle[N/x] \overset{p_2}{\rightsquigarrow}_{\text{wh}} \Psi}{[\alpha]MN \overset{p_1+p_2+1}{\rightsquigarrow}_{\text{wh}} \Psi}$$

$$\text{(eval.}\textcircled{\@}\mu\text{)} \quad \frac{[\alpha']M \overset{p}{\rightsquigarrow}_{\text{wh}} [\beta]\lambda x. P, \beta \neq \alpha' \notin \text{FN}(M)}{[\alpha]MN \overset{p}{\rightsquigarrow}_{\text{wh}} [\beta]\lambda x. P\langle[\alpha]\bullet N/[\alpha']\bullet\rangle}$$

A binary relation between named terms A and named whnf’s Ψ is obtained by erasing the integer superscripts from all the rules, or we can simply define $A \rightsquigarrow_{\text{wh}} \Psi$ to mean that $A \overset{p}{\rightsquigarrow}_{\text{wh}} \Psi$ for some p .

On closed terms, evaluation corresponds to terminating transition sequences:

$$A \overset{p}{\rightsquigarrow}_{\text{wh}} \Psi \Leftrightarrow A \mapsto_{\text{wh}}^p \Psi, \text{ for closed } A, \Psi$$

More generally, this correspondence holds for open terms as well if we extend the evaluation relation to open terms by including the next two rules in its inductive definition.

$$\text{(eval.}\nu\text{)} \quad [a]x \overset{0}{\rightsquigarrow}_{\text{wh}} [a]x$$

$$\text{(eval.}\textcircled{\@}\nu\text{)} \quad \frac{[\alpha']M \overset{p}{\rightsquigarrow}_{\text{wh}} [\beta]x\vec{P}, \alpha' \notin \text{FN}(M)}{[\alpha]MN \overset{p}{\rightsquigarrow}_{\text{wh}} ([\beta]x\vec{P})\langle[\alpha]\bullet N/[\alpha']\bullet\rangle}$$

Theorem 3.2. *$A \rightsquigarrow_{\text{wh}} \Psi$ if and only if $A \mapsto_{\text{wh}}^p \Psi$.*

The remainder of this section lists the lemmas that are used in establishing the theorem.

First some easy properties of the transition relation and the evaluation relation:

Lemma 3.3.

- (1) \mapsto_{wh} is deterministic.
- (2) There is a B such that $A \mapsto_{\text{wh}} B$ iff A is a named non-whnf term.
- (3) \mapsto_{wh} is closed under substitution:

$$A \mapsto_{\text{wh}} B \Rightarrow A[M/x] \mapsto_{\text{wh}} B[M/x] \ \& \ A\langle \mathbb{A}/[\alpha]\bullet \rangle \mapsto_{\text{wh}} B\langle \mathbb{A}/[\alpha]\bullet \rangle$$

Lemma 3.4.

- (1) $\rightsquigarrow_{\text{wh}}$ is deterministic.
- (2) $A \xrightarrow{p}_{\text{wh}} B$ implies B is a named whnf; and $A \xrightarrow{0}_{\text{wh}} \Psi$ iff $A = \Psi$.
- (3) $\rightsquigarrow_{\text{wh}}$ commutes with name contraction:

$$A\langle \alpha/\beta \rangle \xrightarrow{p}_{\text{wh}} \Upsilon \Leftrightarrow \exists \Psi. A \xrightarrow{p}_{\text{wh}} \Psi \ \& \ \Psi\langle \alpha/\beta \rangle = \Upsilon$$

The forward direction of the theorem is straightforward:

Lemma 3.5. $A \xrightarrow{p}_{\text{wh}} \Psi$ implies $A \mapsto_{\text{wh}}^p \Psi$.

Proof. By induction on the derivation of $A \xrightarrow{p}_{\text{wh}} \Psi$. \square

The reverse direction is surprisingly difficult. It is proved simultaneously with a commutation property of evaluation and name substitution:

Lemma 3.6.

- (1) $A \mapsto_{\text{wh}}^p \Psi$ implies $A \xrightarrow{p}_{\text{wh}} \Psi$.
- (2) $A \xrightarrow{q}_{\text{wh}} \Psi$ and $\Psi\langle [\alpha]\bullet N/[\beta]\bullet \rangle \xrightarrow{p}_{\text{wh}} \Upsilon$ imply $A\langle [\alpha]\bullet N/[\beta]\bullet \rangle \xrightarrow{q+p}_{\text{wh}} \Upsilon$.

Proof. (1) and (2) are proved simultaneously by induction on p . For each p , first (1) is proved by induction on the derivation of the first transition step from A according to the small-step structural operational semantics, and then (2) is proved by induction on the derivation of $A \xrightarrow{q}_{\text{wh}} \Psi$. \square

4 Open bisimulation

This section extends Sangiorgi’s co-inductive bisimulation presentation of Lévy–Longo tree equivalence, called “open bisimulation” [22], to the $\lambda\mu$ -calculus. The resulting bisimulation equivalence is a congruence.

The bisimulation equivalence, \approx_{L} , is defined co-inductively as the greatest symmetric relation on terms such that

$$M \approx_{\text{L}} M' \Rightarrow \forall \alpha, \beta. \forall W. [\alpha]M \rightsquigarrow_{\text{wh}} [\beta]W \Rightarrow \exists W'. [\alpha]M' \rightsquigarrow_{\text{wh}} [\beta]W' \ \& \ W \langle \approx_{\text{L}} \rangle_{\text{whnf}} W'$$

where, for any term relation R , $\langle R \rangle_{\text{whnf}}$ is the relation on whnf’s given by:

- (i) $\lambda x. M \langle R \rangle_{\text{whnf}} \lambda x. M' \Leftrightarrow M R M'$
- (ii) $x M_1 \dots M_n \langle R \rangle_{\text{whnf}} x M'_1 \dots M'_n \Leftrightarrow M_i R M'_i \text{ for } 1 \leq i \leq n$

That is, two $\lambda\mu$ -terms are related by \approx_{L} when, evaluated with the same naming, they both diverge or both reach equally named whnf’s whose sub-terms are again related. Apart from the naming, the definition is exactly the same as Sangiorgi’s for the pure λ -calculus. Indeed, it is immediate that the open bisimulation equivalence, Lévy–Longo tree equivalence, from the pure λ -calculus is included in the \approx_{L} bisimulation equivalence for the $\lambda\mu$ -calculus: Lévy–Longo tree equivalence is symmetric and is easily seen to satisfy the defining implication for \approx_{L} in place of $\rightsquigarrow_{\text{wh}}$, thus it is included in \approx_{L} which is the largest such relation by definition.

The crucial properties we need to verify for \approx_{L} , to establish that it a useful semantic equivalence relation, are

- (1) equivalence: \approx_{L} is reflexive, transitive, and symmetric;
- (2) compatibility: \approx_{L} is preserved by all the syntactic constructors of the calculus; and
- (3) \approx_{L} includes the four reduction axioms (β), (μ), (ρ), and (θ);

(4) consistency: not all terms are equated by \approx_L .

Congruence is the conjunction of items (1) and (2).

The proofs of items (1) and (4) are standard.

With regard to (3), we see that each of the four reduction axioms holds for \approx_L by naming both sides and then check that either both sides diverge or both sides evaluate to the same named whnf. In the latter case the result follows by reflexivity of \approx_L . For instance, consider

$$\mu\alpha. [\alpha]M \rightarrow M, \text{ if } a \notin \text{FN}(M) \quad (\Theta)$$

Choosing any name β , we see that $[\beta]\mu\alpha. [\alpha]M \xrightarrow{p}_{\text{wh}} \Psi$ if and only if $[\beta]M \xrightarrow{p-1}_{\text{wh}} \Psi$, by (eval.p) and the assumption that $a \notin \text{FN}(M)$. Clearly, $\Psi \langle \approx_L \rangle_{\text{whnf}} \Psi$ as \approx_L is reflexive.

The remainder of this section is devoted to proving item (2), that \approx_L is compatible, following the operationally-based development of Böhm tree and Lévy–Longo tree theories in [14]. Here, operationally-based means that the development uses elementary techniques of inductive definitions and proofs by rule induction, as in Howe’s method for proving congruence of bisimulation equivalences in functional languages [11]. It is based purely on the bisimulation formulation of \approx_L , without reference to continuity, semantic models, or tree representations. Indeed, it is left to the interested reader to derive an associated notion of Lévy–Longo trees for the $\lambda\mu$ -calculus from the definition of bisimulation.

We define a useful bisimulation operator $\langle \cdot \rangle_{\text{wh}}$ on term relations. If R is a relation between terms, $\langle R \rangle_{\text{wh}}$ is the relation between terms defined by:

$$\begin{aligned} M \langle R \rangle_{\text{wh}} M' &\stackrel{\text{def}}{=} \\ &\forall \alpha, \beta. \forall W. [\alpha]M \rightsquigarrow_{\text{wh}} [\beta]W \Rightarrow \\ &\quad \exists W'. [\alpha]M' \rightsquigarrow_{\text{wh}} [\beta]W' \ \& \\ &\quad W \langle R \rangle_{\text{whnf}} W' \end{aligned}$$

We call a term relation R a *bisimulation* if it is a post-fixed point of $\langle \cdot \rangle_{\text{wh}}$, that is, $R \subseteq \langle R \rangle_{\text{wh}}$. We see that \approx_L is defined as the greatest bisimulation.

Further, we need to define a suitable context closure operator on term relations. First, we define

the *compatible refinement*, \widehat{R} of a relation R , as the relation between terms with identical outermost syntactic constructor and with immediate subterms pairwise related by R , defined as follows.

$$\begin{aligned} x \widehat{R} x, & \quad \text{for all variables } x \\ \lambda x. M \widehat{R} \lambda x. M', & \quad \text{if } M R M' \\ MN \widehat{R} M'N', & \quad \text{if } M R M', N R N' \\ \mu\alpha. [\beta]M \widehat{R} \mu\alpha. [\beta]M', & \quad \text{if } M R M' \end{aligned}$$

Then we define for every term relation R , its (*second-order substitutive*) *context closure*, R^{SSC} , inductively as the smallest relation closed under the rules:

$$\begin{aligned} (\text{ssc}.R) \quad & \frac{M R M'}{M R^{\text{SSC}} M'} \\ (\text{ssc}.comp) \quad & \frac{M \widehat{R^{\text{SSC}}} M'}{M R^{\text{SSC}} M'} \\ (\text{ssc}.subst) \quad & \frac{M R^{\text{SSC}} M' \quad N R^{\text{SSC}} N'}{M[N/x] R^{\text{SSC}} M'[N'/x]} \\ (\text{ssc}.subst2) \quad & \frac{M R^{\text{SSC}} M' \quad N R^{\text{SSC}} N'}{M\langle[\alpha]\bullet N/[\beta]\bullet\rangle R^{\text{SSC}} M'\langle[\alpha]\bullet N'/[\beta]\bullet\rangle} \end{aligned}$$

Rule (ssc.R) says that R^{SSC} contains R ; (ssc.comp) says that R^{SSC} is *compatible* and implies that R^{SSC} is closed under contexts; and (ssc.subst) and (ssc.subst2) say that R^{SSC} is *substitutive*, with regard to both variable substitution and name substitution. Except for (ssc.subst2) this is the definition of the substitutive context closure operator, used extensively in the relational proofs in [12, 13, 15, 14]. It can be seen as an alternative to Howe’s construction of the congruence candidate relation in [11].

We call a symmetric term relation R a *bisimulation up to context* if $R \subseteq \langle R^{\text{SSC}} \rangle_{\text{wh}}$. The main lemma says that if R a bisimulation up to context then R^{SSC} is a bisimulation.

Lemma 4.1 (Main Lemma). $R \subseteq \langle R^{\text{SSC}} \rangle_{\text{wh}}$ implies $R^{\text{SSC}} \subseteq \langle R^{\text{SSC}} \rangle_{\text{wh}}$.

The proof is analogous to the case for the pure λ -calculus, presented in [14]. It uses some specific

properties of the operational semantics for the $\lambda\mu$ -calculus that fall out of the lemmas in the proof of Theorem 3.2. In outline, the proof proceeds by nested inductions (1) on the length of transition sequences and (2) on derivations of relation judgments of the form $MR_v^{\text{SSC}}M'$ according to the inductive definition of R^{SSC} .

As a corollary of the main lemma we get that \approx_L contains its context closure and, hence, that \approx_L is compatible. Therefore:

Theorem 4.2. \approx_L is a congruence.

As another corollary we get a so-called bisimulation up to context proof principle [23, 12] for the $\lambda\mu$ -calculus. This may be a useful tool for reasoning about recursive programs in the $\lambda\mu$ -calculus, but is left for future work.

5 Applicative bisimulation

Now a form of applicative bisimulation [1, 10] is formulated for the $\lambda\mu$ -calculus. The bisimulation equivalence is a congruence and it contains the open bisimulation equivalence from the previous section.

Let us start by defining the associated bisimulation operator on term relations, $\langle \cdot \rangle_{\text{wh}}^{\text{app}}$. It maps every relation S on closed terms to the relation $\langle S \rangle_{\text{wh}}^{\text{app}}$ on closed terms, defined by:

$$M \langle S \rangle_{\text{wh}}^{\text{app}} M' \stackrel{\text{def}}{\iff} \forall \alpha, \beta. \forall W. [\alpha]M \rightsquigarrow_{\text{wh}} [\beta]W \Rightarrow \exists W'. [\alpha]M' \rightsquigarrow_{\text{wh}} [\beta]W' \ \& \ [\beta]W \langle S \rangle_{\text{whnf}}^{\text{app}} [\beta]W'$$

where $\langle S \rangle_{\text{whnf}}^{\text{app}}$ is the relation on closed, named λ -abstractions given by:

$$[\beta]\lambda x. N \langle S \rangle_{\text{whnf}}^{\text{app}} [\beta]\lambda x. N' \stackrel{\text{def}}{\iff} \forall \beta'. \forall \text{closed } P. N \langle [\beta'] \bullet P / [\beta] \bullet \rangle [P/x] S N' \langle [\beta'] \bullet P / [\beta] \bullet \rangle [P/x]$$

An *applicative bisimulation* S is a symmetric relation which is a post-fixed point of $\langle \cdot \rangle_{\text{wh}}^{\text{app}}$, that is, $S \subseteq \langle S \rangle_{\text{wh}}^{\text{app}}$. Let applicative bisimulation equivalence, \sim , be defined co-inductively as the greatest applicative bisimulation.

Remark 3. Morally, we can think of $[\beta]W \langle S \rangle_{\text{whnf}}^{\text{app}} [\beta]W'$ as testing the two named terms in all term contexts of the form $(\mu\beta. \blacksquare)P$ (where \blacksquare is a named-term hole, a place-holder for a named sub-term); the definition of $\langle S \rangle_{\text{whnf}}^{\text{app}}$ reduces the resulting compound terms, $(\mu\beta. [\beta]W)P$ and $(\mu\beta. [\beta]W')P$, by (μ) and (β) before relating them by S . (The (β) step is akin to Howe's definition of applicative bisimulation in [10].) In other words, this is the syntactically well-formed way of applying the named whnf's $[\beta]W$ and $[\beta]W'$ to all closed terms P . This is the justification for calling this form of bisimulation “applicative”.

We extend \sim to a relation on open terms, \sim° , by open extension:

$$M \sim^\circ M' \stackrel{\text{def}}{\iff} \forall \text{closed } \vec{N}. M[\vec{N}/\vec{x}] \sim M'[\vec{N}/\vec{x}]$$

for all terms M and M' with free variables in $\{\vec{x}\}$.

By the same type of argument as for \approx_L we see that \sim° is a consistent equivalence relation and includes the four reduction axioms. The inclusion of the reduction axioms also follows from the following result:

Proposition 5.1. $\approx_L \subseteq \sim^\circ$.

Proof. By the properties (1)–(3) of \approx_L , established in the previous section, and by the above analysis, Remark 3, of $\langle \cdot \rangle_{\text{whnf}}^{\text{app}}$ in terms of term contexts and (μ) and (β) reductions. \square

The inclusion is strict. This is demonstrated by the following example:

$$M \stackrel{\text{def}}{=} \mu\alpha. [\beta]\lambda y. \Omega \\ N \stackrel{\text{def}}{=} \Theta \lambda x. \mu\alpha. [\beta]\lambda y. x$$

where Ω is a diverging term, e.g., $\Omega \stackrel{\text{def}}{=} (\lambda x. x x)(\lambda x. x x)$, and Θ is Turing's fixed point combinator,

$$\Theta \stackrel{\text{def}}{=} (\lambda g. \lambda f. f (g g f)) (\lambda g. \lambda f. f (g g f))$$

For any name γ ,

$$[\gamma]M \rightsquigarrow_{\text{wh}} [\beta]\lambda y. \Omega \\ [\gamma]N \rightsquigarrow_{\text{wh}} [\beta]\lambda y. N$$

We see that $\Omega \not\approx_L N$ because for any name γ' , $[\gamma']\Omega$ diverges and $[\gamma']N$ evaluates to the named whnf $[\beta]\lambda y. N$. But

$$[\beta]\lambda y. \Omega \langle \sim \rangle_{\text{whnf}}^{\text{app}} [\beta]\lambda y. N$$

holds because $[\gamma']N \langle [\beta'] \bullet P / [\beta] \bullet \rangle [P/y]$ diverges for all P and γ' , as one finds by tracing transitions. Hence $M \sim N$.

We now prove that \sim° is compatible. The congruence proof for \approx_L in the preceding section does not carry over to \sim° directly. The proof relies on evaluation of open terms and this interacts badly with the two-layered definition of \sim° , first as a relation \sim on closed terms, and then lifted to open terms by open extension. The proof we give for \sim° is a mixture of the congruence proof for \approx_L in the preceding section and Howe's congruence proof for applicative bisimulation for lazy λ -calculus [11].

We define \sim^\blacklozenge , the (*second-order substitutive*) *compatible extension* of \sim , inductively as the smallest relation closed under the rules:

$$\text{(scand.right)} \quad \frac{M \sim^\blacklozenge M'' \quad M'' \sim^\circ M'}{M \sim^\blacklozenge M'}$$

$$\text{(scand.comp)} \quad \frac{M \widehat{\sim^\blacklozenge} M'}{M \sim^\blacklozenge M'}$$

$$\text{(scand.subst2)} \quad \frac{M \sim^\blacklozenge M' \quad N \sim^\blacklozenge N'}{M \langle [\alpha] \bullet N / [\beta] \bullet \rangle \sim^\blacklozenge M' \langle [\alpha] \bullet N' / [\beta] \bullet \rangle}$$

(Howe's relation for the λ -calculus is the smallest relation closed under the first two rules only; cf. [21].)

Lemma 5.2. \sim^\blacklozenge is substitutive:

$$M \sim^\blacklozenge M' \ \& \ N \sim^\blacklozenge N' \Rightarrow M[N/x] \sim^\blacklozenge M'[N'/x]$$

Proof. By induction on the derivation of $M \sim^\blacklozenge M'$. \square

Rule (scand.comp) says that \sim^\blacklozenge is compatible which implies that it is also reflexive. Therefore (scand.right) implies that \sim^\blacklozenge contains \sim° . It remains to prove the reverse containment, then \sim° is compatible too. We prove that \sim^\blacklozenge , restricted to closed terms, is a post-fixed point of the applicative bisimulation operator:

Lemma 5.3 (Main Lemma). For all closed M and M' , $M \sim^\blacklozenge M' \Rightarrow M \langle \sim^\blacklozenge \rangle_{\text{wh}}^{\text{app}} M'$.

Proof. More generally, we prove for all closed M and M' ,

$$\begin{aligned} M \sim^\blacklozenge M' \ \& \ [\alpha]M \xrightarrow{p}_{\text{wh}} [\beta]\lambda x. N \Rightarrow \\ \exists N'. [\alpha]M' \rightsquigarrow_{\text{wh}} [\beta]\lambda x. N' \ \& \\ \forall \beta'. N \langle [\beta'] \bullet x / [\beta] \bullet \rangle \sim^\blacklozenge N' \langle [\beta'] \bullet x / [\beta] \bullet \rangle \end{aligned}$$

by nested inductions (1) on p and (2) on the derivation of $M \sim^\blacklozenge M'$, as in the proof of the Main Lemma 4.1 for open bisimulation. \square

In a standard fashion, using determinacy of evaluation and certain symmetries, we deduce from the main lemma that \sim^\blacklozenge is contained in \sim° , as required to show that \sim° is compatible. Hence:

Theorem 5.4. \sim° is a congruence.

From this result one can derive that \sim° is included in “lazy” contextual equivalence for the calculus, the notion of contextual equivalence, or observational equivalence, one obtains by observing termination in the operational semantics. Is the inclusion strict? Let

$$\begin{aligned} \Xi &\stackrel{\text{def}}{=} \lambda x. \lambda y. x x \\ N &\stackrel{\text{def}}{=} \mu \alpha. [\alpha]\lambda y. \mu \alpha'. [\alpha]\Xi \end{aligned}$$

Then $\Xi \not\approx N$ because

$$\begin{aligned} [\alpha]\Xi \rightsquigarrow_{\text{wh}} [\alpha]\lambda y. \Xi \\ [\alpha]N \rightsquigarrow_{\text{wh}} [\alpha]\lambda y. \mu \alpha'. [\alpha]\Xi \end{aligned}$$

and

$$\begin{aligned} \Xi \langle [\beta'] \bullet P / [\alpha] \bullet \rangle [P/y] &= \Xi \\ (\mu \alpha'. [\alpha]\Xi) \langle [\beta'] \bullet P / [\alpha] \bullet \rangle [P/y] &= \mu \alpha'. [\beta']\Xi P \end{aligned}$$

and

$$\begin{aligned} [\alpha']\Xi \rightsquigarrow_{\text{wh}} [\alpha']\lambda y. \Xi \\ [\alpha']\mu \alpha'. [\beta']\Xi P \rightsquigarrow_{\text{wh}} [\beta']\lambda y. \Xi \end{aligned}$$

and $[\alpha']\lambda y. \Xi \langle \sim \rangle_{\text{whnf}}^{\text{app}} [\beta']\lambda y. \Xi$ fails if $\alpha' \neq \beta'$. But I conjecture that Ξ is contextually equivalent to N . (The only useful term contexts for distinguishing Ξ from N appear to be those of the form $\mathbb{C}[\bullet P]$ but $\Xi P \rightarrow^2 \Xi$ and $N P \rightarrow^6 \Xi$ and it follows that $\mathbb{C}[\Xi P] \sim \mathbb{C}[N P]$, hence they have the same termination behaviour.) If so, contextual equivalence is not included in applicative bisimulation.

6 Future work

The new small-step and big-step operational semantics and the bisimulation theories presented here have been developed with the aim of being applied to other variants of the $\lambda\mu$ -calculus, e.g., the call-by-value calculus or the call-by-name calculus with η -conversion. The operational semantics and applicative bisimulation theory should also be applicable to the typed versions of the calculus. Moreover, there are many other related λ -calculi with control operators for which it would be interesting to see if our operational theory is applicable. These applications and extensions of our results are left for future work.

The interrelationship between our applicative bisimulation theory and the operational theories based on CIU theorems [27, 4] deserves further study. For pure call-by-name λ -calculus, the two characterisations are easily interderivable, but that is not necessarily the case for the $\lambda\mu$ -calculus: for instance, it is not clear how the example that we used to distinguish \approx_L and \sim° in §5 can be proved by means of a CIU theorem for the calculus.

References

- [1] S. Abramsky. The lazy lambda calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pages 65–116. Addison-Wesley, 1990.
- [2] S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105(2):159–267, 1993.
- [3] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. Number 103 in Studies in Logic and the Foundations of Mathematics. North-Holland, revised edition, 1984.
- [4] G. M. Bierman. A computational interpretation of the lambda-mu calculus. In L. Brim, J. Gruska, and J. Zlatuska, editors, *Proc. 23rd Mathematical Foundations of Computer Science, Brno, Czech Republic*, volume 1450, pages 336–345. Springer-Verlag, 1998.
- [5] P. de Groote. An environment machine for the lambda-mu-calculus. To appear in *Mathematical Structures in Computer Science*, 1998.
- [6] M. Felleisen and D. P. Friedman. Control operators, the SECD-machine, and the λ -calculus. In M. Wirsing, editor, *Formal Description of Programming Concepts III, Proc. IFIP TC2 Working Conference, Gl. Avernæs, 1986*. IFIP, North-Holland, 1987.
- [7] M. Felleisen and R. Hieb. The revised report on the syntactic theories of sequential control and state. *Theoretical Computer Science*, 103:235–271, 1992.
- [8] A. D. Gordon and A. M. Pitts, editors. *Higher Order Operational Techniques in Semantics*. Publications of the Newton Institute. Cambridge University Press, 1998.
- [9] T. G. Griffin. The formulae-as-types notion of control. In *Proc. 17th ACM Symposium on Principles of Programming Languages, San Fransisco, CA*, pages 47–57, 1990.
- [10] D. J. Howe. Equality in lazy computation systems. In *Proc. 4th Annual IEEE Symposium on Logic in Computer Science*, 1989.
- [11] D. J. Howe. Proving congruence of bisimulation in functional programming languages. *Information and Computation*, 124(2):103–112, 1996.
- [12] S. B. Lassen. Relational reasoning about contexts. In Gordon and Pitts [8], pages 91–135.
- [13] S. B. Lassen. *Relational Reasoning about Functions and Nondeterminism*. PhD thesis, Dept. of Computer Science, Univ. of Aarhus, Dec. 1998. BRICS Dissertation Series DS-98-2.
- [14] S. B. Lassen. Bisimulation in untyped lambda calculus: Böhm trees and bisimulation up to context. In M. Mislove, editor, *MFPS XV, New Orleans, LA*, volume 20 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1999.
- [15] S. B. Lassen and A. K. Moran. Unique fixed point induction for McCarthy’s amb. In *Proceedings of the 24th International Symposium on Mathematical Foundations of Computer Science (September 6-10, 1999, Szklarska Poreba, Poland)*, volume 1672 of *Lecture Notes in Computer Science*, pages 198–208. Springer-Verlag, 1999.
- [16] G. Longo. Set-theoretical models of lambda calculus: Theories, expansions and isomorphisms. *Annals of Pure and Applied Logic*, 24:153–188, 1983.
- [17] C.-H. L. Ong. *The Lazy Lambda Calculus: An Investigation into the Foundations of Functional Programming*. PhD thesis, University of London, 1988.

- [18] C.-H. L. Ong and C. A. Stewart. A curry-howard foundation for functional computation with control. In *Proc. 24th ACM Symposium on Principles of Programming Languages*, 1997.
- [19] M. Parigot. $\lambda\mu$ -calculus: An algorithmic interpretation of classical natural deduction. In A. Voronkov, editor, *Proceedings Intl. Conf. on Logic Programming and Automated Reasoning, LPAR '92, St Petersburg*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer-Verlag, 1992.
- [20] G. D. Plotkin. A structural approach to operational semantics. Lecture Notes DAIMI FN-19, Dept. of Computer Science, Univ. of Aarhus, 1981.
- [21] D. Sands. Operational theories of improvement in functional languages (extended abstract). In *Proceedings of the Fourth Glasgow Workshop on Functional Programming*, Workshops in Computing Series, pages 298–311. Springer-Verlag, 1991.
- [22] D. Sangiorgi. The lazy lambda calculus in a concurrency scenario. *Information and Computation*, 111(1):120–153, 1994.
- [23] D. Sangiorgi. On the bisimulation proof method. Technical Report LFCS-94-299, University of Edinburgh, Aug. 1994.
- [24] P. Selinger. An implementation of the call-by-name $\lambda\mu\nu$ -calculus. Available from URL <http://www.math.lsa.umich.edu/~selinger/lammunu/>, 1998.
- [25] T. Streicher and B. Reus. Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, 1998.
- [26] C. Talcott. A theory for program and data specification. *Theoretical Computer Science*, 104:129–159, 1993.
- [27] C. Talcott. Reasoning about functions with effects. In Gordon and Pitts [8], pages 347–390.
- [28] C. P. Wadsworth. The relation between computational and denotational properties for Scott's D_∞ -models of the lambda-calculus. *SIAM Journal on Computing*, 5(3):488–521, 1976.
- [29] A. K. Wright and M. Felleisen. A syntactic approach to type soundness. *Information and Computation*, 115(1):38–94, 1994.