Untyped Lambda-Calculus with Input-Output

Jerzy Tiuryn[1] and Mitchell Wand[2]

[1] Institute of Informatics, Warsaw University, Banacha 2, 02-097 Warsaw, POLAND.‡

[2] College of Computer Science, Northeastern University, 360 Huntington Avenue, 161CN, Boston, MA 02115, USA.§

Abstract. We introduce an untyped λ-calculus with input-output, based on Gordon's continuation-passing model of input-output. This calculus is intended to allow the classification of possibly infinite input-output behaviors, such as those required for servers or distributed systems. We define two terms to be operationally approximate iff they have similar behaviors in any context. We then define a notion of applicative approximation and show that it coincides with operational approximation for these new behaviors. Last, we consider the theory of pure λ-terms under this notion of operational equivalence.

1 Introduction

In the study of the operational behavior of languages, the standard paradigm for studying equivalence of terms begins by setting up some relatively weak notion of behavior (or immediate observation). For PCF [10], this is generally taken to be convergence to an integer constant; for the untyped λ-calculus, this can be taken to be reducibility to a head normal form (Böhm) or to an abstraction (Abramsky). Two terms are said to be *operationally equivalent* iff they have the same behavior in any context. This is a useful notion of equivalence, as it asserts that the two terms can be substituted for each other at will.

These finite notions of behavior are generally satisfactory as a formalization of experiments on finite behaviors of programs: to see whether a program M has some complex finite behavior, one can easily set up a context $C[-]$ such that $C[M]$ terminates iff M has the desired behavior. If, however, we wish to classify programs that have potentially infinite behaviors, such as a program that echoes its input, then a more direct model of input-output seems necessary.

In this paper we present a system that augments the pure λ-calculus with primitive input-output operations that can read or write a single bit. The observable behavior of a term is taken to be its input-output behavior, up to

‡ E-mail: `tiuryn@mimuw.edu.pl` This work is partly supported by NSF Grants CCR-9417382, CCR-9304144, and by Polish KBN Grant 2 P301 031 06.

§ E-mail: `wand@ccs.neu.edu` WWW: `http://www.ccs.neu.edu/home/wand`. Work supported by the National Science Foundation under grants number CCR-9304144 and CCR-9404646.

bisimilarity. Our system is analogous to the CPS I/O system of Gordon [2], but without types.

Our main result is an applicative characterization of operational equivalence for this system. This characterization is similar in spirit to Milner's Context Lemma [8] for PCF, to Böhm trees (for reducibility to head normal form) or to Abramsky's applicative bisimulation (for reducibility to an abstraction). We then go on to study the discriminatory power of this notion of operational equivalence for pure λ-terms.

In Section 2 we present our calculus and our notion of behavior. Our notion of behavior is based on simulation $\sqsubseteq_{sim}$[5]. We give some examples of programs that are equivalent and inequivalent under this new notion of behavior. We then define operational equivalence or operational approximation (the corresponding one-sided notion) by quantifying over contexts in the usual way.

In Section 3, we extend our notion of behavior to define an applicative approximation (the one-sided version of applicative bisimulation) denoted $\sqsubseteq_{ap}$. In Section 4, we show that applicative and operational approximation coincide. We do this by showing that applicative approximation is a precongruence, that is, a preorder closed under the term-formation rules of the language [1].

In Section 5, we consider the λ-theory obtained by restricting applicative equivalence to pure λ-terms. We show that this theory properly extends the theory $\mathcal{B}$ of Böhm trees, is a proper subset of the theory $\mathcal{H}^*$ of equi-solvability, and is incomparable with the theory $\lambda\eta$ of β-η reduction.

We conclude with discussion of related work and some conclusions and open problems.

2 The Basic System

The syntax of our terms is given by the following:

$$M ::= v \mid (MM) \mid \lambda v.M \mid c_0(M) \mid c_1(M) \mid r(M)$$

This is the usual untyped λ-calculus, augmented by new unary term constructors c_0, c_1, and r. Let Λ denote this set of terms, and let Λ_0 denote the set of closed terms. We will write $c_0 M$ for $c_0(M)$, etc., where convenient. We will use the usual λ-calculus notions of β-reduction, head-reduction, etc. We will write $=$ for identical (α-congruent) terms, $=_\beta$ for terms that are β-convertible, $\rightarrow_h$ for head reduction, and $\rightarrow_h^*$ for a sequence of head reductions.

We turn the closed terms of this calculus into a labelled transition system with actions $\{!0, !1, ?0, ?1\}$ by the following rules:

[5] Because the systems we will consider are deterministic, it is sufficient (and traditional) to consider only simulation (or approximation), rather than bisimulation (or equivalence); see Proposition 5.

$$\frac{M \to_h^* c_0(M')}{M \xrightarrow{!0} M'} \qquad \frac{M \to_h^* c_1(M')}{M \xrightarrow{!1} M'}$$

$$\frac{M \to_h^* r(M')}{M \xrightarrow{?0} (M'\ulcorner 0 \urcorner)} \qquad \frac{M \to_h^* r(M')}{M \xrightarrow{?1} (M'\ulcorner 1 \urcorner)}$$

where $\ulcorner 0 \urcorner$ and $\ulcorner 1 \urcorner$ are the Church numerals for 0 and 1.

With these rules, we can see that c_0 and c_1 play the role of "print 0" and "print 1", and r plays the role of "read a bit". Note that these terms operate only at the top level, so they are effectively in continuation-passing style. Note that a term with no head-normal form, or a term with some head-normal form other than these four, such as $\lambda x.x$ or $c_0 M N$, deadlocks. Note also that although the definition is framed in terms of head reduction, we shall see in Lemma 7 that simulation and bisimulation in this system are closed under β-conversion.

For example, the term

$$Y(\lambda f.r(\lambda x.\,\mathrm{zero?}\,x(c_0 f)(c_1 f))) \tag{1}$$

where Y is the usual fixpoint combinator and zero? is a zero-testing term, echoes its input. For a more complex example, here is a term that echoes its input at half-speed by reading two bits, depositing them in an unbounded queue, and printing one bit from the queue:

$$\begin{aligned} &Y(\lambda fq.r(\lambda x.r(\lambda y.\,\mathrm{deq}(\mathrm{enq}(\mathrm{enq}\,qx)y) \\ &\qquad\qquad (\lambda vq'.\,\mathrm{zero?}\,v(c_0(fq'))(c_1(fq')))))) \, q_0 \end{aligned} \tag{2}$$

Here q_0 is a coding of the empty queue, $\mathrm{enq}\,q\,x$ returns the queue obtained by adding x to tail of q, and $\mathrm{deq}\,qf$ passes to f the first element on the queue q and the queue obtained by removing that first element. Such terms are of course easy to code in the untyped λ-calculus.

We begin by defining simulation in the standard way [9]:

Definition 1. Let $R \subseteq \Lambda_0 \times \Lambda_0$ be a relation on closed terms. We say R is a *simulation relation* iff whenever $M\ R\ N$ and $M \xrightarrow{a} M'$, then there exists a term N' such that $N \xrightarrow{a} N'$ and $M'\ R\ N'$. R is a *bisimulation* iff R and R^{-1} are simulations.

We say M *simulates* N ($M \sqsubseteq_{sim} N$) iff there is some simulation relation R such that $M\ R\ N$. We say M and N are bisimilar ($M \cong_{bisim} N$) iff there is some bisimulation R such that $M\ R\ N$.

It is straightforward to check that the terms (1) and (2) above are not bisimilar, whereas a term that differed from (2) only by altering the coding of queues would be bisimilar to it, even though it would not be equal to it in any reasonable λ-theory.

We can characterize simulation and bisimulation as the greatest fixpoint of monotone operators on relations:

Definition 2. If $R \subseteq \Lambda_0 \times \Lambda_0$ is any relation on closed terms, the relation $[R]_{sim} \subseteq \Lambda_0 \times \Lambda_0$ is defined by $M\ [R]_{sim}\ N$ iff the following properties hold:

1. if $M \to_h^* c_0 M'$, then there exists N' such that $N \to_h c_0 N'$ and $M'\ R\ N'$.
2. if $M \to_h^* c_1 M'$, then there exists N' such that $N \to_h c_1 N'$ and $M'\ R\ N'$.
3. if $M \to_h^* rM'$, then there exists N' such that $N \to_h rN'$ and $(M'\lceil 0 \rceil)\ R\ (N'\lceil 0 \rceil)$ and $(M'\lceil 1 \rceil)\ R\ (N'\lceil 1 \rceil)$

Lemma 3. *R is a simulation iff $R \subseteq [R]_{sim}$, and R is a bisimulation iff $R \subseteq ([R]_{sim} \cap [R^{-1}]_{sim}{}^{-1})$.*

Proof. By straightforward comparison of the definitions.

$[-]_{sim}$ is evidently monotone, so we can take its greatest fixed point:

Proposition 4. *$\sqsubseteq_{sim} = \mathrm{gfp}([-]_{sim})$, and $\cong_{bisim} = \mathrm{gfp}([-]_{sim} \cap [(-)^{-1}]_{sim}{}^{-1})$.*

Proof. Immediate from preceding lemma.

From this we get a coinduction principle that states that if $R \subseteq [R]_{sim}$, then $R \subseteq \sqsubseteq_{sim}$. It is easy to show by coinduction that $\sqsubseteq_{sim}$ is a preorder.

Proposition 5. $\cong_{bisim} = \sqsubseteq_{sim} \cap (\sqsubseteq_{sim})^{-1}$

Proof. A routine argument, depending on the determinacy of the system.

Lemma 6. *If R and S are relations on closed terms, then $[R]_{sim} \circ [S]_{sim} \subseteq [R \circ S]_{sim}$.*

Proof. Immediate.

The following lemma shows that $\sqsubseteq_{sim}$ is closed under β-conversion.

Lemma 7. *If $M =_\beta M' \sqsubseteq_{sim} N' =_\beta N$, then $M \sqsubseteq_{sim} N$*

Proof. A straightforward Church-Rosser argument shows that $=_\beta \subseteq [=_\beta]_{sim}$, so $(=_\beta \circ \sqsubseteq_{sim} \circ =_\beta) \subseteq [=_\beta]_{sim} \circ [\sqsubseteq_{sim}]_{sim} \circ [=_\beta]_{sim} \subseteq [(=_\beta \circ \sqsubseteq_{sim} \circ =_\beta)]_{sim}$, and so the desired result follows by coinduction.

Our notion of behavior immediately gives us a notion of operational approximation:

Definition 8. We say M *operationally approximates* N ($M \sqsubseteq N$) iff for every context C such that $C[M]$ and $C[N]$ are closed, $C[M] \sqsubseteq_{sim} C[N]$.

3 Applicative Approximation

Operational approximation is difficult to show in practice because it requires quantification over all contexts. Simulation $\sqsubseteq_{sim}$ is easier to establish, but it is not the same as operational approximation. For example, $I = \lambda x.x \sqsubseteq_{sim} \Omega = (\lambda x.xx)(\lambda x.xx)$, because both deadlock, but $I(c_0(I)) \not\sqsubseteq_{sim} \Omega(c_0(I))$, since the former has a !0 action but the latter does not. Hence $I \not\sqsubseteq \Omega$. Similarly $I \sqsubseteq_{sim} \lambda xy.xy$, since both deadlock, but $I \not\sqsubseteq \lambda xy.xy$, since $I(c_0 I) \not\sqsubseteq_{sim} (\lambda xy.xy)(c_0 I)$ (since the latter reduces to an abstraction and therefore deadlocks).

Thus our situation, while the ordinary situation in language theory, differs from the usual situation in concurrency theory, in which one seeks to show that bisimulation is a congruence. The experience of [1, 4] suggests that we should consider not just simulation between whole programs but simulation in every head context. Thus we are interested in the relation defined by

Definition 9. We say M *applicatively approximates* N ($M \sqsubseteq_{ap} N$) iff for every closing substitution σ and every sequence of closed terms $P_1, \ldots, P_n$,

$$((M\sigma)P_1 \ldots P_n) \sqsubseteq_{sim} ((N\sigma)P_1 \ldots P_n)$$

Note that $\Omega \sqsubseteq_{ap} M$ for all M, since Ω deadlocks in all head contexts. Furthermore $\lambda x.\Omega \sqsubseteq_{ap} M$, since for all closed P, $(\lambda x.\Omega)P =_\beta \Omega \sqsubseteq_{ap} (MP)$. Thus in our system $\lambda x.\Omega \sqsubseteq_{ap} \Omega$ and $\Omega \sqsubseteq_{ap} \lambda x.\Omega$. This differentiates our system, based on input/output behavior, from that of Abramsky [1].

Furthermore, applicative approximation is not a bisimulation, since $c_0(I) \sqsubseteq_{ap} c_o(\Omega)$ (they are bisimilar by themselves, and deadlock in any head context), but $I \not\sqsubseteq_{ap} \Omega$.

Applicative approximation can be defined coinductively:

Definition 10. Say a relation R on closed terms is an *applicative approximation* iff $M\ R\ N$ implies

1. $M \sqsubseteq_{sim} N$ and
2. for every closed term P, $(MP)\ R\ (NP)$

Define $\sqsubseteq_{ap'}$ as the largest applicative approximation, and extend this relation to open terms by saying that $M \sqsubseteq^o_{ap'} N$ iff $M\sigma \sqsubseteq_{ap'} N\sigma$ for every closing substitution σ. Then evidently $\sqsubseteq^o_{ap'}$ coincides with $\sqsubseteq_{ap}$. On the whole, however, the closed form of Definition 9 seems more useful.

β-conversion is still sound for $\sqsubseteq_{ap}$:

Lemma 11. *If $M =_\beta M' \sqsubseteq_{ap} N' =_\beta N$, then $M \sqsubseteq_{ap} N$.*

Proof. Immediate from Lemma 7 and Definition 9.

4 Main Results

To show that applicative and operational approximation coincide, we will show that

$$\sqsubseteq_{ap}^{\forall C} \subseteq \sqsubseteq \subseteq \sqsubseteq_{ap}$$

where $\sqsubseteq_{ap}^{\forall C}$ is the largest precongruence contained in $\sqsubseteq_{ap}$. We then show that applicative approximation is already a precongruence. Thus $\sqsubseteq = \sqsubseteq_{ap}$.

The difficult part of this proof is showing that $\sqsubseteq_{ap}$ is a precongruence. Here we follow the outline of [4], replacing induction on computation steps by coinduction. The key lemma is the Substitution Lemma (Lemma 13).

The first lemma is the one-sided version of "bisimulation up to $\sim$" [9, Proposition 3.4.3]:

Lemma 12. *Let S be any relation on closed terms. If $S \subseteq [(S \circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}]_{sim}$ then $S \subseteq \sqsubseteq_{sim}$.*

Proof. We have $S \subseteq [(S \circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}]_{sim}$ and $\sqsubseteq_{sim} \subseteq [\sqsubseteq_{sim}]_{sim}$, so we have

$$\begin{aligned}
(S \circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim} &\subseteq ([(S \circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}]_{sim} \circ [\sqsubseteq_{sim}]_{sim}) \cup [\sqsubseteq_{sim}]_{sim} \\
&\subseteq [(S \circ \sqsubseteq_{sim} \circ \sqsubseteq_{sim}) \cup (\sqsubseteq_{sim} \circ \sqsubseteq_{sim})]_{sim} \cup [\sqsubseteq_{sim}]_{sim} \\
&= [(S \circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}]_{sim} \cup [\sqsubseteq_{sim}]_{sim} \\
&= [(S \circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}]_{sim}
\end{aligned}$$

So by coinduction, we have $((S \circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}) \subseteq \sqsubseteq_{sim}$. Hence we have

$$S \subseteq [(S \circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}]_{sim} \subseteq [\sqsubseteq_{sim}]_{sim} \subseteq \sqsubseteq_{sim}$$

Lemma 13 Substitution Lemma. *Let M and N be closed and $M \sqsubseteq_{ap} N$. Then for and any term Q with at most one free variable z,*

$$Q[M/z] \sqsubseteq_{sim} Q[N/z]$$

Proof. Let $\sigma_1 = [M/z]$ and $\sigma_2 = [N/z]$. Let

$$S = \{(Q\sigma_1, Q\sigma_2) \mid \mathrm{fv}(Q) \subseteq \{z\}\}$$

We will show that $S \subseteq [(S \circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}]_{sim}$, so by the preceding lemma, $S \subseteq \sqsubseteq_{sim}$.

We will work out the case in which $Q\sigma_1 \rightarrow_h^* c_0 Q'$; the other cases for $[-]_{sim}$ are similar. We must show that $Q\sigma_2 \rightarrow_h^* c_0 Q''$ for some Q'' such that $(Q', Q'') \in (S \circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}$. We proceed by induction on the length of the reduction sequence. We assume that the length of the reduction sequence is k, and proceed by cases on the form of Q:

1. If $Q = z$, then $Q\sigma_1 = M \rightarrow_h^* c_0 Q'$. Since $M \sqsubseteq_{ap} N$, we have $Q\sigma_2 = z\sigma_2 = N \rightarrow_h^* c_0 Q''$ with $Q' \sqsubseteq_{sim} Q''$, as desired.

2. If $Q = c_0 Q_1$, then $Q\sigma_1 = c_0(Q_1\sigma_1)$ so $Q' = Q_1\sigma_1$. Hence $Q\sigma_2 = c_0(Q_1\sigma_2)$, so let Q'' be $(Q_1\sigma_2)$, and $(Q', Q'') \in S \subseteq (S\circ \sqsubseteq_{sim})$ as desired.
3. $Q = zQ_1 \ldots Q_n$ for $n > 0$. So $Q\sigma_1 = M(Q_1\sigma_1)\ldots(Q_n\sigma_1) \to_h^* c_0 Q'$. Hence M must reduce to some abstraction $\lambda x.M_0$. Then we have:

$$\begin{aligned} Q\sigma_1 &= M(Q_1\sigma_1)\ldots(Q_n\sigma_1) \\ &\to_h^* (\lambda x.M_0)(Q_1\sigma_1)\ldots(Q_n\sigma_1) \\ &\to_h (M_0[Q_1\sigma_1/x])(Q_2\sigma_1)\ldots(Q_n\sigma_1) \\ &= ((M_0[Q_1/x])Q_2\ldots Q_n)\sigma_1 \\ &\to_h^* c_0 Q' \end{aligned}$$

Since the last reduction is of length less than k, by the induction hypothesis we conclude that $((M_0[Q_1/x])Q_2\ldots Q_n)\sigma_2 \to_h^* c_0 Q''$ with $(Q', Q'') \in (S\circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}$. So

$$\begin{aligned} M(Q_1\sigma_2)\ldots(Q_n\sigma_2) &\to_h^* (\lambda x.M_0)(Q_1\sigma_2)\ldots(Q_n\sigma_2) \\ &\to_h (M_0[Q_1\sigma_2/x])(Q_2\sigma_2)\ldots(Q_n\sigma_2) \\ &= ((M_0[Q_1/x])Q_2\ldots Q_n)\sigma_2 \\ &\to_h^* c_0 Q'' \end{aligned}$$

Since $M \sqsubseteq_{ap} N$, it follows that

$$Q\sigma_2 = N(Q_1\sigma_2)\ldots(Q_n\sigma_2) \to_h^* c_0 Q'''$$

with $Q'' \sqsubseteq_{sim} Q'''$.

Now we have

$$(Q', Q''') \in ((S\circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim})\circ \sqsubseteq_{sim} = (S\circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}$$

as desired.
4. The only remaining possibility is that $Q = (\lambda x.Q_0)Q_1\ldots Q_n$ with $n > 0$. In this case we have

$$\begin{aligned} Q\sigma_1 &= ((\lambda x.Q_0)\sigma_1)(Q_1\sigma_1)\ldots(Q_n\sigma_1) \\ &\to_h (Q_0\sigma_1[Q_1\sigma_1/x])(Q_2\sigma_1)\ldots(Q_n\sigma_1) \\ &= ((Q_0[Q_1/x])Q_2\ldots Q_n)\sigma_1 \\ &\to_h^* c_0 Q' \end{aligned}$$

Since the last reduction sequence is of length less than k, by the induction hypothesis we have

$$((Q_0[Q_1/x])Q_2\ldots Q_n)\sigma_2 \to_h^* c_0 Q''$$

with $(Q', Q'') \in (S\circ \sqsubseteq_{sim}) \cup \sqsubseteq_{sim}$. But then

$$\begin{aligned} Q\sigma_2 &= ((\lambda x.Q_0)\sigma_2)(Q_1\sigma_2)\ldots(Q_n\sigma_2) \\ &\to_h (Q_0\sigma_2[Q_1\sigma_2/x])(Q_2\sigma_2)\ldots(Q_n\sigma_2) \\ &= ((Q_0[Q_1/x])Q_2\ldots Q_n)\sigma_2 \\ &\to_h^* c_0 Q'' \end{aligned}$$

as desired.

Theorem 14 Precongruence. *If $M \sqsubseteq_{ap} N$, then for any context C (not necessarily closed), $C[M] \sqsubseteq_{ap} C[N]$.*

Proof. The contexts C can be described by the grammar

$$C ::= [] \mid x \mid (CC) \mid (\lambda x.C) \mid c_0 C \mid c_1 C \mid rC$$

We proceed by structural induction on C.

1. If $C = []$ or $C = x$, the result is true trivially.
2. If $C = \lambda x.C'$, by the induction hypothesis $C'[M] \sqsubseteq_{ap} C'[N]$. We must show that for every closing substitution σ and closed terms $P_1, \ldots, P_n$,

$$(\lambda x.C'[M])\sigma P_1 \ldots P_n \sqsubseteq_{sim} (\lambda x.C'[N])\sigma P_1 \ldots P_n$$

   If $n = 0$, the left-hand side deadlocks, so the desired result holds. If $n > 0$, then

$$(\lambda x.C'[M])\sigma P_1 \ldots P_n =_\beta (C'[M])\sigma[P_1/x]P_2 \ldots P_n$$

   and

$$(\lambda x.C'[N])\sigma P_1 \ldots P_n =_\beta (C'[N])\sigma[P_1/x]P_2 \ldots P_n$$

   so the desired relation follows from $C'[M] \sqsubseteq_{ap} C'[N]$ and the closure of $\sqsubseteq_{sim}$ under $=_\beta$.
3. If $C = (C_1 C_2)$, by the induction hypothesis $C_1[M] \sqsubseteq_{ap} C_1[N]$ and $C_2[M] \sqsubseteq_{ap} C_2[N]$. Let σ be a closing substitution and let $P_1, \ldots, P_n$ be closed terms. We wish to show that

$$(C_1[M]C_2[M])\sigma P_1 \ldots P_n \sqsubseteq_{sim} (C_1[N]C_2[N])\sigma P_1 \ldots P_n$$

   Then we have

$$\begin{array}{ll} (C_1[M]C_2[M])\sigma P_1 \ldots P_n & \\ = & (C_1[M]\sigma)(C_2[M]\sigma)P_1 \ldots P_n \\ \sqsubseteq_{sim} & (C_1[N]\sigma)(C_2[M]\sigma)P_1 \ldots P_n \\ \sqsubseteq_{sim} & (C_1[N]\sigma)(C_2[N]\sigma)P_1 \ldots P_n \\ = & (C_1[N]C_2[N])\sigma P_1 \ldots P_n \end{array}$$

   where the two $\sqsubseteq_{sim}$ steps follow from the substitution lemma, letting Q be $z(C_2[M]\sigma)P_1 \ldots P_n$ and $(C_1[N]\sigma)zP_1 \ldots P_n$, respectively, where z is a fresh variable.

The remaining cases are similar to the application case, but easier.

Now we can state the main theorem:

Theorem 15 Equivalence. $M \sqsubseteq_{ap} N$ *iff* $M \sqsubseteq N$.

Proof. Define $\sqsubseteq_{ap}^{\forall C}$ by $M \sqsubseteq_{ap}^{\forall C} N$ iff for every context C such that $C[M]$ and $C[N]$ are closed, $C[M] \sqsubseteq_{ap} C[N]$.

Since $C[M] \sqsubseteq_{ap} C[N]$ implies $C[M] \sqsubseteq_{sim} C[N]$, it follows that $\sqsubseteq_{ap}^{\forall C} \subseteq \sqsubseteq$. To show that $\sqsubseteq \subseteq \sqsubseteq_{ap}$, assume $M \sqsubseteq N$, with free variables $x_1, \ldots x_k$, let σ be a closing substitution, and let $P_1 \ldots P_n$ be closed terms. Let $Q_i = \sigma(x_i)$ for $1 \leq i \leq k$. Let $C = (\lambda x_1 \ldots x_k.[])Q_1 \ldots Q_k P_1 \ldots P_n$. Then

$$(M\sigma)P_1 \ldots P_n =_\beta C[M] \sqsubseteq_{sim} C[N] =_\beta (N\sigma)P_1 \ldots P_n$$

Hence we have

$$\sqsubseteq_{ap}^{\forall C} \subseteq \sqsubseteq \subseteq \sqsubseteq_{ap}$$

But, by the precongruence theorem, $\sqsubseteq_{ap} \subseteq \sqsubseteq_{ap}^{\forall C}$, so all three relations coincide.

5 Operational Equivalence of Pure $\boldsymbol{\lambda}$-terms

Let us call the terms of our system with no constants (c_0, c_1, or r) *pure* terms. Define $\cong = \sqsubseteq \cap (\sqsubseteq)^{-1}$.

It follows immediately from Theorem 14 that $\cong$ is a congruence, and hence by Lemma 11, the restriction of $\cong$ to closed pure λ-terms is a λ-theory. Let us call this theory T.

We will show that this theory properly extends the theory $\mathcal{B}$ of Böhm trees, is a proper subset of the theory $\mathcal{H}^*$ of equi-solvability, and is incomparable with the theory $\lambda\eta$ of β-η reduction. We will state each of the necessary facts as lemmas.

Lemma 16. *Let M and N be pure terms (not necessarily closed) and C be a context such that $C[M]$ is solvable and $C[N]$ is unsolvable. Then $M \not\sqsubseteq N$.*

Proof. Since $C[M]$ is solvable, there exist closed terms $P_1, \ldots, P_n$ such that

$$C[M]P_1 \ldots P_n =_\beta I$$

Since $C[N]$ is unsolvable, so is $C[N]P_1 \ldots P_n$. Hence

$$C[M]P_1 \ldots P_n(c_0(I)) \not\sqsubseteq C[N]P_1 \ldots P_n(c_0(I))$$

since the former reduces to $c_0(I)$ and the latter has no action. But if it were true that $M \sqsubseteq N$, then it would follow that

$$C[M]P_1 \ldots P_n(c_0(I)) \sqsubseteq C[N]P_1 \ldots P_n(c_0(I))$$

Contradiction.

Lemma 17. $T \subset \mathcal{H}^*$

Proof. The preceding lemma and the definition of $\mathcal{H}^*$ as the set

$$\{(M, N) \mid (\forall C)[C[M] \text{ is solvable} \iff C[N] \text{ is solvable}]\}$$

show that $(M, N) \notin \mathcal{H}^*$ implies $(M, N) \notin T$. To show the inclusion is proper, observe that I and $\lambda xy.xy$ are in $\mathcal{H}^*$, but they are not in T, as noted at the beginning of Section 3.

Lemma 18. $\lambda\eta \not\subseteq T$

Proof. Again, $(I, \lambda xy.xy)$ provides a counterexample.

Lemma 19. $\mathcal{B} \subset T$

Proof. Define $\mathcal{B}$ as the set of pairs of closed pure terms with equal Böhm trees. To show $\mathcal{B} \subseteq T$, extend the notion of Böhm trees to terms with c_0, c_1, and r, by treating these symbols as free variables. The conditions for $\mathcal{B} \subseteq [\mathcal{B}]$ are easy to verify; since the theory of Böhm trees forms a λ-theory, $BT(M) = BT(N)$ implies $BT(MP) = BT(NP)$. So by coinduction, $\mathcal{B} \subseteq \sqsubseteq$, and by symmetry $\mathcal{B} \subseteq T$.

To show that the inclusion is proper, consider the terms $M = \lambda x.x\Omega(xI)$ and $N = \lambda x.x\Omega(\lambda z.xIz)$. These have distinct Böhm trees, but we claim they are T-equivalent, by showing that there is no distinguishing context $[-]P_1 \dots P_n$.

We consider the possibilities for P_1. If P_1 is unsolvable or has head normal form $\lambda x.xQ_1 \dots Q_n$, then both MP_1 and NP_1, are unsolvable, so these contexts cannot distinguish M and N.

If $P_1 =_\beta \lambda x.dQ$ for $d \in \{c_0, c_1, r\}$, then both MP_1 and NP_1 deadlock, so contexts of this form cannot distinguish M and N.

The only remaining possibility is that $P_1 =_\beta \lambda x_1x_2.Q$ for some Q. But then we have

$$\begin{aligned} MP_1 &= (\lambda x.x\Omega(xI))P_1 \\ &=_\beta P_1\Omega(P_1I) \\ &=_\beta P_1\Omega(\lambda x_2.Q[I/x_1]) \end{aligned}$$

$$\begin{aligned} NP_1 &= (\lambda x.x\Omega(\lambda z.xIz))P_1 \\ &=_\beta P_1\Omega(\lambda z.P_1Iz) \\ &=_\beta P_1\Omega(\lambda z.Q[I/x_1, z/x_2]) \\ &= P_1\Omega(\lambda x_2.Q[I/x_1]) \end{aligned}$$

where the last step is an α-conversion. So $MP_1 =_\beta NP_1$, and these P_1's cannot distinguish M from N either.

6 Related Work

The idea of operational equivalence goes back at least as far as Plotkin [10]. The idea of applicative equivalence, attempting to characterize operational equivalence by quantification over a much more restricted set of contexts, goes back to Milner's context lemma [8]. The ciu theorem of Honsell, Mason, Smith, and Talcott [4] provides a similar result for an untyped system with side-effects. Both these systems use convergence as a notion of immediate behavior.

Abramsky [1] pointed out how convergence to an abstraction could be used as an observable behavior for the untyped λ-calculus, and identified the corresponding operational equivalence as his applicative bisimulation. He also pointed out that the existing theory of Böhm trees characterized operational equivalence in which the existence of a head normal form was the observable event. He also emphasized the importance of showing that applicative approximation was a precongruence. Abramsky's notion of observation is well-suited for functional programming languages; we are interested instead in programs that interact with their environment and we therefore choose input-output behavior to be the fundamental observable. Hence for us, Ω and $\lambda x.\Omega$ are equivalent, since neither can interact with its environment in any head context.

Sangiorgi [11] generalized Abramsky's system by allowing an observable event to be either convergence to an abstraction or an input-output event. He considered classes of transition systems rather than our specific system, including the possibility of non-determinism. He used this framework to show that two pure terms are equivalent in Milner's translation into the π-calculus iff they are equivalent in every such input/output framework. He gave an applicative bisimulation and showed that it is a congruence, but did not characterize his equivalence as the largest congruence contained in some more primitive notion of behavior. We hope to extend our results to be parameterized over an input-output system in a way similar to Sangiorgi's.

Howe [5, 6, 7] proposed a general framework for showing that applicative approximations were precongruences. An earlier version of this paper [13] used Howe's method to extend $\sqsubseteq_{sim}$ to a precongruence, but that precongruence turned out to be properly contained in $\sqsubseteq$. We do not know whether Howe's method can be used to prove Theorem 14.

Andrew Gordon's thesis [2] presented four different systems for input-output in a functional language and gave translations between them, but did not consider applicative approximations. Our system is a variant of his CPS I/O system. Gordon's more recent paper [3] uses Howe's technique to extend Milner's context lemma to a version of PCF with products, sums, and recursive types. His notion of immediate behavior is also based on convergence, but he does deal with unbounded behavior via the use of recursive types.

We have also studied the extension to possibly infinite input-output behavior in the context of PCF [12]. In this context we were also able to formulate an applicative approximation that coincided with operational approximation, and to show that this new approximation was a conservative extension of the conventional notion of operational approximation.

7 Conclusions

We have formulated a notion of behavior for untyped λ-calculus based on potentially infinite interactions. We have defined an applicative approximation for this notion of behavior, and showed that it is equivalent to the corresponding notion of operational approximation.

We hope to extend the current work to consider not only single-process input-output but also CCS-style interacting processes.

Acknowledgements

We thank Greg Sullivan for his careful reading of the manuscript and many discussions. Thanks also to Andrew Gordon and Davide Sangiorgi for helpful conversations.

References

1. Samson Abramsky. The lazy lambda calculus. In David A. Turner, editor, *Research Topics in Functional Programming*, pages 65–116. Addison-Wesley, 1990.
2. Andrew D. Gordon. *Functional Programming and Input/Output*. Cambridge University Press, Cambridge, 1994.
3. Andrew D. Gordon. Bisimilarity as a Theory of Functional Programming. In *Proceedings of 11th Conference on Mathematical Foundations of Programming Semantics*, 1995.
4. Furio Honsell, Ian A. Mason, Scott Smith, and Carolyn Talcott. A Variable Typed Logic of Effects. *Information and Computation*, 119:55–90, 1995.
5. Douglas J. Howe. Equality in Lazy Computation Systems. In *Proc. 4th IEEE Symposium on Logic in Computer Science*, pages 198–203, 1989.
6. Douglas J. Howe. A Note on Proving Congruence of Bisimulation in a Generalized Lambda Calculus. unpublished manuscript, 1995.
7. Douglas J. Howe. Proving Congruence of Bisimulation in Functional Programming Languages. to appear in *Information and Computation*, 1996.
8. Robin Milner. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science*, 4:1–22, 1977.
9. Robin Milner. Operational and Algebraic Semantics of Concurrent Processes. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 1201–1242. MIT Press/Elsevier, 1990.
10. Gordon D. Plotkin. LCF Considered as a Programming Language. *Theoretical Computer Science*, 5:223–255, 1977.
11. Davide Sangiorgi. The Lazy Lambda Calculus in a Concurrency Scenario. *Information and Computation*, 111(1):120–153, May 1994. Preliminary version appeared in *Proc. 7th IEEE Symposium on Logic in Computer Science*, 1992.

12. Jerzy Tiuryn and Mitchell Wand. Adding Input-Output to PCF (Technical Summary). submitted for publication, July 1995.
13. Jerzy Tiuryn and Mitchell Wand. Untyped Lambda-Calculus with Input-Output (Progress Report). presented at Atlantique meeting, La Jolla, June 1995.