Untyped Lambda-Calculus with Input-Output

Jerzy Tiuryn*
Institute of Informatics
Warsaw University
Banacha 2
02-097 Warsaw, POLAND
tiuryn@mimuw.edu.pl

Mitchell Wand†
College of Computer Science
Northeastern University
360 Huntington Avenue, 161CN
Boston, MA 02115, USA
wand@ccs.neu.edu
http://www.ccs.neu.edu/home/wand

September 12, 1995

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.

A contrasting notion of equivalence is suggested by Abramsky's applicative bisimulation [1]. This approach seeks to find some richer notion of behavior that coincides with

*This work is partly supported by NSF Grants CCR-9417382, CCR-9304144, and by Polish KBN Grant 2 P301 031 06.

†Work supported by the National Science Foundation under grants number CCR-9304144 and CCR-9404646.

operational equivalence yet avoids explicit quantification over all contexts. In a typed system, this leads to Milner's Context Lemma [8]; in an untyped system this leads to Böhm trees (for reducibility to head normal form) or to Abramsky's applicative bisimulation.

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.

Gordon [2] provides four such calculi and gives translations between them. Gordon's system involves an elaborate type system. Our goal is to study such systems without the complication of types. Our system is also a simplification of the one considered in [13].

In Section 2 we present our calculus and our notion of behavior. Our notion of behavior is based on simulation $\sqsubseteq_{sim}$[1]. 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 $\sqsubseteq$. In Section 4, we show that applicative and operational aproximation 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]. This is the most technically difficult result in the paper.

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_0M for $c_0(M)$, etc., where convenient. We will write $\lambda\vec{x}.M$ for $\lambda x_1.\lambda x_2.\ldots\lambda x_n.M$. 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.

[1]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 2.2.

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

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

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

where $\lceil 0\rceil$ and $\lceil 1\rceil$ 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_0MN, deadlocks. Note also that although the definition is framed in terms of head reduction, we shall see in Lemma 2.4 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:

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

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.

This calculus inherits the usual notions of simulation and bisimulation from its definition as a labelled transition system [9]. 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.

By specializing the general definitions to our system, we can characterize these notions as greatest fixpoints of monotone operators on relations:

Definition 2.1 *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 $(M'\lceil 1\rceil)\ R\ (N'\lceil 1\rceil)$

We can define *simulation* $\sqsubseteq_{sim}$ in the usual way as the greatest fixpoint of $[-]_{sim}$. The coinduction principle states that if $R \subseteq [R]_{sim}$, then $R \subseteq \sqsubseteq_{sim}$. Bisimulation $\cong_{bisim}$ is the greatest fixed point of $[-]_{sim} \cap ([-]_{sim})^{op}$. It is immediate that these definitions agree with the general definitions on our system.

Proposition 2.2 $\cong_{bisim} = \sqsubseteq_{sim} \cap (\sqsubseteq_{sim})^{op}$.

Proof: A routine argument. ∎

Lemma 2.3 *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 2.4 *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 2.5 *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. 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 3.1 *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 $\Omega \sqsubseteq_{ap} \lambda x.\Omega$.

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

Lemma 3.2 *If $M =_\beta M' \sqsubseteq_{ap} N' =_\beta N$, then $M \sqsubseteq_{ap} N$.*

Proof: Immediate from Lemma 2.4 and Definition 3.1. ∎

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 4.2).

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

Lemma 4.1 *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 4.2 (Substitution Lemma) *If $M \sqsubseteq_{ap} N$, then for any closing σ and any term Q with at most one free variable z,*

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

Proof: Let $\sigma_1 = [M\sigma/z]$ and $\sigma_2 = [N\sigma/z]$. 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 $Q\sigma_1 \to_h^* c_0 Q'$; the other cases for $[-]_{sim}$ are similar. We must show that $Q\sigma_2 \to_h^* c_0 Q''$ 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 $Q\sigma_1 = M\sigma \to_h^* c_0 Q'$. Since $M \sqsubseteq_{ap} N$, we have $Q\sigma_2 = z\sigma_2 = N\sigma \to_h^* c_0 Q''$ 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 $Q\sigma_1 = (M\sigma)(Q_1\sigma_1)\ldots(Q_n\sigma_1) \to_h^* c_0 Q'$. Hence $M\sigma$ must reduce to some abstraction $\lambda x.M_0$. Then we have:

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

   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

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

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

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

   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

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

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

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

as desired.

∎

Theorem 4.3 (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 4.4 (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 $C[M] \sqsubseteq C[N]$ 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 λ-terms

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

It follows immediately from Theorem 4.3 that $\cong$ is a congruence, and hence by Lemma 3.2, 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 5.1 *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 5.2 $T \subset \mathcal{H}^*$

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 5.3 $\lambda\eta \not\subseteq T$

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

Lemma 5.4 $\mathcal{B} \subset T$

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 applicative approximation is due to Abramsky [1]. Abramsky pointed out that the existing theory of Böhm trees can be viewed as an applicative bisimulation in which the existence of a head normal form is an observable event (the basic component of behavior), and he formulated a new bisimulation in which the existence of a weak head normal form is the basic observable. He also emphasized the importance of showing that applicative approximation was a precongruence.

The definition of applicative approximation by application to identical arguments (as in Definition 3.1.4) can be seen as requiring that terms have the same behavior in some restricted set of contexts, and the equivalence theorem as asserting that if terms have the same behavior in these contexts, then they have the same behavior in all contexts. The oldest such result is 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.

Howe [5, 6, 7] proposed a general framework for showing that applicative approximations were precongruences. An earlier version of this paper [12] 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 4.3.

Andrew Gordon's thesis [2] presents four different systems for input-output in a functional language and gives translations between them, but does 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 study applicative approximation for an extension of PCF with a far more elaborate type structure than ours. However, his notion of immediate behavior is also based on convergence and does not deal with potentially infinite interactions such as ours.

We have also studied the extension to possibly infinite input-output behavior in the context of PCF [11]. 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.

References

[1] Samson Abramsky. The lazy lambda calculus. In David A. Turner, editor, *Reearch 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] Jerzy Tiuryn and Mitchell Wand. Adding Input-Output to PCF (Technical Summary). submitted for publication, July 1995.

[12] Jerzy Tiuryn and Mitchell Wand. Untyped Lambda-Calculus with Input-Output (Progress Report). presented at Atlantique meeting, La Jolla, June 1995.

[13] Mitchell Wand. Compiler Correctness for Parallel Compilers. In *Functional Programming Languages and Computer Architecture*, pages 120–134, June 1995.