From Operational Semantics to Domain Theory

Ian A. Mason*

University of New England, Armidale, NSW Australia 2350
iam@cs.stanford.edu

Scott F. Smith†

The Johns Hopkins University, Baltimore, Maryland 21218
scott@cs.jhu.edu

and

Carolyn L. Talcott*

Stanford University, Stanford, California 94305
clt@sail.stanford.edu

Abstract

This paper builds domain theoretic concepts upon an operational foundation. The basic operational theory consists of a single step reduction system from which an operational ordering and equivalence on programs are defined. The theory is then extended to include concepts from domain theory, including the notions of directed set, least upper bound, complete partial order, monotonicity, continuity, finite element, ω-algebraicity, full abstraction, and least fixed point properties. We conclude by using these concepts to construct a (strongly) fully abstract continuous model for our language. In addition we generalize a result of Milner and prove the uniqueness of such models.

Contents

*Partially supported by NSF grants CCR-8917606 and CCR-8915663, and DARPA contract NAG2-703.
†Partially supported by NSF grants CCR-9109070 and CCR-9301340.

1 Introduction

This paper presents a bottom-up approach to the construction of semantic domains from an underlying operational semantics. There is a practical motivation for taking this approach. The power of domain theory is well-known; however, it suffers a shortcoming that often limits its usefulness. For many languages, program equivalence induced from domain constructions does not correspond exactly to operational equivalence. This is known as the full abstraction problem. No such problem is encountered in a bottom-up approach as the structure is built using operational tools alone. A complete discussion of full abstraction is outside the scope of this paper, see for instance [Stoughton, 1988, Bloom, 1990].

We study a variant of the untyped call-by-value lambda calculus enriched with arithmetic, pairing, and branching primitives. The syntax and semantics of our language is defined in section 2. In section 3, a basic operational theory of this language is developed, consisting of an operational approximation relation $a \lesssim b$, and the corresponding equivalence relation $a \cong b$. $a \cong b$ means no program context can distinguish a and b. In section 3.1 we give an alternate characterization of $\lesssim$ as equivalent to the ordering $\lesssim^{\text{ciu}}$. This characterization is an analogue of Milner's context lemma [Milner, 1977] and is used to prove a number of properties such as extensionality of $\lesssim$. $\lesssim$ however fails to satisfy several basic domain theoretic requirements. In sections 3.3 and 3.4 we show by simple computability arguments that $\lesssim$ does not form a CPO and that the operational analogue of continuity fails. A basic operational theory thus needs new concepts to be developed further.

Section 4 presents an extension to the basic operational theory. We define a simple ordering $\lesssim_{\text{s}}$ on $\lesssim$-directed sets of expressions, and the corresponding equivalence $\cong_{\text{s}}$. This ordering extends $\lesssim$: $a \lesssim b$ iff $\{a\} \lesssim_{\text{s}} \{b\}$. One view of the use of $\lesssim_{\text{s}}$ is that it allows an expression, a, to be decomposed into a $\lesssim$-directed set of expressions, A, such that $\{a\} \cong_{\text{s}} A$. Then, to prove properties about a it suffices to prove properties about the set A. A particular instance of this is the case of fixed points. We show

$$\{\lambda x.\mathtt{bot}, f(\lambda x.\mathtt{bot}), f(f(\lambda x.\mathtt{bot})), \ldots, f^k(\lambda x.\mathtt{bot}), \ldots\} \cong_{\text{s}} \{\mathtt{fix}(f)\},$$

for a functional f and suitable fixed-point combinator $\mathtt{fix}$. This property is an analogue to the least fixed-point principle from domain theory,

$$\bigsqcup\{[\![\lambda x.\mathtt{bot}]\!], [\![f(\lambda x.\mathtt{bot})]\!], [\![f(f(\lambda x.\mathtt{bot}))]\!], \ldots, [\![f^k(\lambda x.\mathtt{bot})]\!], \ldots\} = [\![\mathtt{fix}(f)]\!],$$

for a domain with valuation function $[\![\cdot]\!]$. The basic theory of $\lesssim_{\text{s}}$ is developed in sections 4.1 and 4.2. We establish an alternative characterization, $\lesssim_{\text{s}}^{\text{ciu}}$, of $\lesssim_{\text{s}}$ mirroring the alternate characterization $\lesssim^{\text{ciu}}$ given for $\lesssim$. Extensionality of $\lesssim_{\text{s}}$ and the above fixed-point property are proved. The goal is to show $\lesssim_{\text{s}}$ induces a CPO, but this does not follow directly. Next in section 4.4 we consider the structure of the *finite* expressions. We show that the finite elements are the image of syntactically definable projection functions, an idea taken from [Abadi et al., 1991b]. In sections 4.5 and 4.6 we use the finite elements to show that $\lesssim_{\text{s}}$-directed sets (of $\lesssim$-directed sets) have $\lesssim_{\text{s}}$-least upper bounds, and that the resulting CPO is ω-algebraic. Furthermore, application is continuous.

In section 5 we study the general notion of a model of a functional call-by-value programming language with numbers and pairing. Our approach builds on the work of Milner [Milner, 1977] and Meyer [Meyer, 1982]. We begin by defining the notion of an **FLD** domain (functional programming language domain). These are reflexive domains with an extensional partial ordering $\sqsubseteq$ reflecting degrees of definedness. Next we define a notion of **FLEM** (functional language environment model)

for interpreting expressions in an **FLD** domain. We classify these models according to what properties they possess. We then construct a model, using $\sqsubseteq_{\mathrm{s}}$, that is (strongly) fully abstract, continuous, and ω-algebraic, using the results of section 4.6. Following Milner [Milner, 1977] we show all such models are isomorphic, generalizing his result to the untyped case. Full abstraction alone is not enough to prove uniqueness of models in the untyped case, we need to slightly strengthen the condition to so-called *strong* full abstraction to obtain uniqueness.

1.1 Related Work

There is a considerable corpus of work developing basic operational theories found in the literature, though mostly for call-by-name languages. A number of properties are desired, including congruence and extensionality of $\cong$. Researchers that have developed methods to directly prove basic operational properties include [Milner, 1977], [Talcott, 1985], [Howe, 1989], [Bloom, 1990], [Jim and Meyer, 1991], [Mason and Talcott, 1991], and [Gordon, 1994]. It also should be mentioned that for simple untyped functional languages like the one studied here, domain models may be altered by various means to give fully abstract models [Abramsky, 1990, Ong, 1988]. [Talcott, 1985] studies general notions of equivalence for languages based on the call-by-value lambda calculus, and develops several schemes for establishing properties of such relations. [Howe, 1989] proves congruence for a class of languages with a particular style of operational semantics. This schema succeeds in capturing many simple functional programming language features. [Jim and Meyer, 1991, Ong, 1992, Howe, 1995] present other such schemata. [Mason and Talcott, 1991] have proven respect of computation and congruence for more complex languages than the language presented here—the languages have continuations as first-class objects and a global state. This work gives anecdotal evidence that the results presented herein may apply to more complex languages, a subject for future work.

An earlier version of some of this work treated a call-by-name variant [Smith, 1992]. This work could be said to be a descendant of [Milner, 1977], and may ultimately be traced back to results for the pure λ-calculus [Hyland, 1976, Wadsworth, 1976].

2 The Syntax and Semantics

In this paper, a simple untyped call-by-value functional language with numbers and pairing is studied. The syntax and execution semantics is first presented.

2.1 Syntax

To present the syntax of our language we assume given a countable set of variables $\mathbb{X}$ and natural numbers $\mathbb{N}$. We let x, y, z range over $\mathbb{X}$, and n range over $\mathbb{N}$. Operators for the language are as follows.

DEFINITION 2.1 (OPERATORS $\mathbb{O}$) The unary, binary, and ternary operators are:

$$\mathbb{O}_1 = \{\texttt{isnat}, \texttt{pred}, \texttt{succ}, \texttt{ispr}, \texttt{fst}, \texttt{snd}\} \quad \mathbb{O}_2 = \{\texttt{app}\} \quad \mathbb{O}_3 = \{\texttt{br}\} \quad \mathbb{O}_n = \emptyset \text{ for } n > 3$$

$$\mathbb{O} = \bigcup_{n \in \mathbb{N}} \mathbb{O}_n$$

The operators are largely self-explanatory; `app` is function application, `br` is conditional branching, and `isnat` and `ispr` recognize numbers and pairs, respectively. Recognizers are an important feature of untyped programming languages such as Lisp and Scheme, and since an untyped language

is being studied here, they are a natural feature to include. We also define the extended operators to include pairing: $\mathbb{O}_1^+ = \mathbb{O}_1$, $\mathbb{O}_2^+ = \mathbb{O}_2 \cup \{\mathtt{pr}\}$, $\mathbb{O}_3^+ = \mathbb{O}_3$, $\mathbb{O}^+ = \mathbb{O}_1^+ \cup \mathbb{O}_2^+ \cup \mathbb{O}_3^+$. We let $\mathtt{op} \in \mathbb{O}$ range over operators, and $\mathtt{op}^+ \in \mathbb{O}^+$ range over extended operators. The pairing operator is given a special status for technical reasons.

The set of λ-abstractions $\mathbb{L}$, value pairs $\mathbb{P}$, value expressions $\mathbb{V}$, and expressions $\mathbb{E}$ are defined, mutually recursively, as the least sets satisfying the following equations.

DEFINITION 2.2 (SYNTAX OF EXPRESSIONS $\mathbb{L}$, $\mathbb{P}$, $\mathbb{V}$, $\mathbb{E}$)

$$\mathbb{P} = \mathtt{pr}(\mathbb{V},\mathbb{V}) \qquad \mathbb{L} = \lambda\mathbb{X}.\mathbb{E} \qquad \mathbb{E} = \mathbb{V} + \bigcup_{n\in\mathbb{N}} \mathbb{O}_n^+(\overbrace{\mathbb{E},\ldots,\mathbb{E}}^{n}) \qquad \mathbb{V} = \mathbb{X} + \mathbb{N} + \mathbb{L} + \mathbb{P}$$

We let v range over $\mathbb{V}$, and a, b, c, d, e range over $\mathbb{E}$.

λ is a binding operator and free and bound variables of expressions are defined as usual. Two expressions are considered equal if they are the same up to renaming of bound variables. $\mathrm{FV}(a)$ is the set of free variables of a. For any syntactic domain Y and set of variables X we let Y_X be the elements of Y with free variables in X. A *closed expression* is an expression with no free variables, thus $\mathbb{E}_\emptyset$ is the set of all closed expressions. $a^{\{x:=b\}}$ is the result of substituting b for the free occurrences of x in a taking care not to trap free variables of b.

A *value substitution* is a finite map from variables to values. We let σ range over value substitutions (i.e. $\sigma \in \mathbb{X} \xrightarrow{\text{finite}} \mathbb{V}$). a^σ is the result of simultaneous substitution of free occurrences of $x \in \mathrm{Dom}(\sigma)$ in a by $\sigma(x)$, again taking care not to trap variables. Value substitutions play an important role due to the call-by-value nature of the language.

A value substitution, σ, is a finite map from variables to values (i.e. $\sigma \in \mathbb{X} \xrightarrow{\text{finite}} \mathbb{V}$). a^σ is the result of simultaneous substitution of free occurrences of $x \in \mathrm{Dom}(\sigma)$ in a by $\sigma(x)$, again taking care not to trap variables. Substitutions σ are defined on values only due to the call-by-value nature of the language.

Several syntactic abbreviations will be made to aid in readability of programs. These include

$$\begin{array}{lcl}
a\mathtt{(}b\mathtt{)} & = & \mathtt{app}(a,b) \\
\mathtt{t} & = & 1 \\
\mathtt{f} & = & 0 \\
\mathtt{if}(a,b,c) & = & \mathtt{br}(a,\lambda x.b,\lambda x.c)\mathtt{(0)} \qquad \text{where } x \text{ is new} \\
\mathtt{bot} & = & (\lambda x.x\mathtt{(}x\mathtt{)})(\lambda x.x\mathtt{(}x\mathtt{)}) \\
\mathtt{fix} & = & \lambda y.(\lambda x.\lambda z.y\mathtt{(}x\mathtt{(}x\mathtt{))(}z\mathtt{)})(\lambda x.\lambda z.y\mathtt{(}x\mathtt{(}x\mathtt{))(}z\mathtt{)}) \\
\mathtt{islam}(a) & = & \mathtt{if}(\mathtt{ispr}(a),\mathtt{f},\mathtt{if}(\mathtt{isnat}(a),\mathtt{f},\mathtt{t})) \\
\mathtt{nateq}(a,b) & = & \mathtt{fix}(\lambda z.\lambda x.\lambda y.\mathtt{if}(\mathtt{iszero}(x), \\
 & & \qquad \mathtt{if}(\mathtt{iszero}(y),\mathtt{t},\mathtt{f}), \\
 & & \qquad \mathtt{if}(\mathtt{iszero}(y,\mathtt{f},z\mathtt{(}x\mathtt{)(}y\mathtt{)}))))) \\
\mathtt{iszero}(a) & = & \mathtt{if}(\mathtt{isnat}(a),\mathtt{if}(a,\mathtt{f},\mathtt{t}),\mathtt{f}) \\
a_1 \circ \ldots \circ a_n & = & \lambda x.a_1\mathtt{(}\ldots\mathtt{(}a_n\mathtt{(}x\mathtt{))}\ldots\mathtt{)}
\end{array}$$

Observe typewriter font parentheses $a\mathtt{(}b\mathtt{)}$ abbreviate function application. `if` is a defined construct because arguments to operators are evaluated before the operator is applied. `fix` is a call-by-value version of the standard fixed-point combinator for functionals. Note that `bot` is an expression, not a value.

2.2 Semantics

The operational semantics of expressions is given by a reduction relation $\mapsto$, using the convenient notion of a reduction context (a.k.a. evaluation context) taken from [Felleisen et al., 1987]. $C[\bullet]$ denotes a *context*, an expression C with occurrences of holes "$\bullet$".

DEFINITION 2.3 (CONTEXTS $\mathbb{C}$)

$$\mathbb{C} = \{\bullet\} + \mathbb{X} + \mathbb{N} + \lambda\mathbb{X}.\mathbb{C} + \bigcup_{n\in\mathbb{N}} \mathbb{O}_n^+(\overbrace{\mathbb{C},\ldots,\mathbb{C}}^{n})$$

We let C range over $\mathbb{C}$. $C[a]$ denotes the result of replacing all holes in C by a. Free variables of a may become bound in this process.

DEFINITION 2.4 (REDICES $\mathbb{E}_{\mathrm{rdx}}$) The set of redices, $\mathbb{E}_{\mathrm{rdx}}$, is defined as

$$\mathbb{E}_{\mathrm{rdx}} = \bigcup_{n\in\mathbb{N}} \mathbb{O}_n(\overbrace{\mathbb{V},\ldots,\mathbb{V}}^{n})$$

Redices are either immediately available for execution, $\mathtt{succ}(0)$, or are *stuck*, $\mathtt{succ}(\lambda x.x)$. In this presentation stuck computations are treated as divergences for simplicity. Observe that $\mathtt{pr}(v, v')$ is not a redex since only operators in $\mathbb{O}$ may be used to form redices—this is the technical reason for having different sets $\mathbb{O}$ and $\mathbb{O}^+$. Reduction contexts $\mathbb{R}$ determine the subexpression that is to be reduced next.

DEFINITION 2.5 (REDUCTION CONTEXTS $\mathbb{R}$) The set of reduction contexts, $\mathbb{R}$, is the subset of $\mathbb{C}$ defined by

$$\mathbb{R} = \{\bullet\} + \bigcup_{n\in\mathbb{N},m\in\mathbb{N}} \mathbb{O}_{m+n+1}^+(\overbrace{\mathbb{V},\ldots,\mathbb{V}}^{m},\mathbb{R},\overbrace{\mathbb{E},\ldots,\mathbb{E}}^{n})$$

We let R range over $\mathbb{R}$. In expression $R[a]$, R denotes the continuation for the computation a. Reduction contexts are used in evaluation as follows. In order to perform one step of computation of some none-value expression a, it is *uniquely* decomposed into $a = R[b]$ for some R and redex b by the following Lemma. Uniqueness of decomposition implies evaluation is deterministic.

LEMMA 2.6 (DECOMPOSITION) Either $a \in \mathbb{V}$ or a can be written uniquely as $R[b]$ where $b \in \mathbb{E}_{\mathrm{rdx}}$.

DEFINITION 2.7 (EVALUATION $\mapsto$) The evaluation relation $\mapsto$ is the transitive, reflexive closure of the single-step evaluation relation $\mapsto_1$, which is generated by the following clauses:

(beta) $R[(\lambda x.a)(v)] \mapsto_1 R[a^{\{x:=v\}}]$

(br) $R[\mathtt{br}(v, v_1, v_2)] \mapsto_1 \begin{cases} R[v_1] & \text{if } v \neq \mathtt{f} \text{ and } v \notin \mathbb{X} \\ R[v_2] & \text{if } v = \mathtt{f} \end{cases}$

(isnat) $R[\mathtt{isnat}(v)] \mapsto_1 \begin{cases} R[\mathtt{t}] & \text{if } v \in \mathbb{N} \\ R[\mathtt{f}] & \text{if } v \in \mathbb{P} \cup \mathbb{L} \end{cases}$

(ispr) $R[\mathtt{ispr}(v)] \mapsto_1 \begin{cases} R[\mathtt{t}] & \text{if } v \in \mathbb{P} \\ R[\mathtt{f}] & \text{if } v \in \mathbb{N} \cup \mathbb{L} \end{cases}$

(pred) $R[\mathtt{pred}(n+1)] \mapsto_1 R[n]$ for $n \in \mathbb{N}$

(succ) $R[\mathtt{succ}(n)] \mapsto_1 R[n+1]$ for $n \in \mathbb{N}$

(fst) $R[\mathtt{fst}(\mathtt{pr}(v_0, v_1))] \mapsto_1 R[v_0]$

(snd) $R[\mathtt{snd}(\mathtt{pr}(v_0, v_1))] \mapsto_1 R[v_1]$

Note it is possible to compute with open expressions using the above definition. a is *defined* (written $a\downarrow$) if it evaluates to a value, $a\uparrow$, otherwise. The relation $\ll$ orders definedness of expressions.

DEFINITION 2.8 (DEFINEDENESS $\downarrow$, $\uparrow$, $\ll$) For $a, b, v \in \mathbb{E}_\emptyset$,

$$a\downarrow \Leftrightarrow (\exists v)(a \mapsto v) \qquad a\uparrow \Leftrightarrow \neg(a\downarrow) \qquad a \ll b \Leftrightarrow a\downarrow \Rightarrow b\downarrow$$

A few simple properties concerning computation are the following.

LEMMA 2.9 (UNIFORMITY OF REDUCTION)

(i) $b_0 = b_1$ if $a \mapsto_1 b_i$ for $i < 2$

(ii) $a \mapsto_1 b \Rightarrow a^\sigma \mapsto_1 b^\sigma$

(iii) $a \mapsto_1 b \Rightarrow R[a] \mapsto_1 R[b]$

3 Operational Approximation and Equivalence

In this section we define the operational approximation and equivalence relations on expressions and study their general properties. Operational equivalence formalizes the notion of equivalence as black-boxes. Treating programs as black boxes entails only observing what values they produce, and not how they produce them.

DEFINITION 3.1 (OPERATIONAL RELATIONS $\precsim$, $\cong$)

$$a \precsim b \Leftrightarrow (\forall C \in \mathbb{C} \mid C[a], C[b] \in \mathbb{E}_\emptyset)(C[a] \ll C[b])$$

$$a \cong b \Leftrightarrow a \precsim b \wedge b \precsim a$$

LEMMA 3.2 (ELEMENTARY $\precsim$ / $\cong$ PROPERTIES)

(i) $\precsim$ / $\cong$ are nontrivial, in particular $\neg(0 \precsim 1)$.

(ii) $\precsim$ is transitive and reflexive (a pre-order).

(iii) $\cong$ is an equivalence relation.

(iv) $\precsim$ is a pre-congruence, i.e. $a \precsim b$ implies $C[a] \precsim C[b]$.

(v) $\cong$ is a congruence, i.e. $a \cong b$ implies $C[a] \cong C[b]$.

(vi) If $a \precsim b$, then for $v \in \mathbb{V}_\emptyset$, $a^{\{x:=v\}} \precsim b^{\{x:=v\}}$.

PROOF: For (i), let $C = \mathtt{if}(\bullet, 0, \mathtt{bot})$. (ii)–(iii) are direct by inspection of the definitions. For (iv), this follows by the observation that contexts compose. Assume $a \precsim b$; to show $C[a] \precsim C[b]$, by the definitions we must show $C'[C[a]]\downarrow$ implies $C'[C[b]]\downarrow$, and this is direct from assumption $a \precsim b$, picking the context to be $C'[C]$. (v) is direct from (iv). For (vi), we must show $C[a^{\{x:=v\}}]\downarrow \Rightarrow C[b^{\{x:=v\}}]\downarrow$ for arbitrary C (it suffices to pick C that contains no x). Picking $C' = (\lambda x.C[\bullet])(v)$, from the assumption $a \precsim b$ we obtain $(\lambda x.C[a])(v)\downarrow \Rightarrow (\lambda x.C[b])(v))\downarrow$, which by computing is what we wanted to show. □

3.1 The CIU Theorem for $\precsim$

Several alternate formulations of operational equivalence $\cong^{\text{alt}}$ have been developed. These have the common feature that equivalence is defined using a restricted set of observing contexts. Trivially, $a \cong b$ implies $a \cong^{\text{alt}} b$. The significance of these alternate formulations is that the converse also holds. What is then gained is not a different notion of equivalence, but simpler methods for establishing equivalence.

We define such an alternate notion in this paper, restricting contexts to be **C**losed **I**nstances of all **U**ses of an expression. This equivalence is thus called **CIU** equivalence, $\cong^{\text{ciu}}$, following [Mason and Talcott, 1991]. $a \cong^{\text{ciu}} b$ means a and b behave identically when closed (the *closed instances* part) and placed in any reduction context R (the *uses* part). As alluded to above, we can prove $\cong$ is the same as $\cong^{\text{ciu}}$, so we have gained a simpler characterization of operational equivalence. This result is the cornerstone of the (non-trivial) equational theory of $\cong$.

DEFINITION 3.3 (CIU ORDERING, $\precsim^{\text{ciu}}$)

$$a \precsim^{\text{ciu}} b \;\Leftrightarrow\; (\forall R, \sigma \;\big|\; R[a^\sigma], R[b^\sigma] \in \mathbb{E}_\emptyset)(R[a^\sigma] \ll R[b^\sigma]),$$

(Recall the convention that σ ranges over value substitutions.)

THEOREM 3.4 (CIU) $a \precsim b \;\Leftrightarrow\; a \precsim^{\text{ciu}} b$.

This theorem is a corollary of theorem 4.6, which is proved in section 4.2. An important application of theorem 3.4 is the proof of the following theorem.

THEOREM 3.5 ($\precsim$ EXTENSIONALITY) For $v_0, v_1 \in \mathbb{L}$,

$$v_0 \precsim v_1 \;\Leftrightarrow\; (\forall v)(v_0\texttt{(}v\texttt{)} \precsim v_1\texttt{(}v\texttt{)}).$$

PROOF: $\Rightarrow$ is direct from the pre-congruence of $\precsim$. For $\Leftarrow$, assume $(\forall v)(v_0\texttt{(}v\texttt{)} \precsim v_1\texttt{(}v\texttt{)})$ and show $v_0 \precsim v_1$ for $v_0, v_1 \in \mathbb{L}$. By Theorem 3.4, it suffices to assume v_0, v_1 are closed and to show $a^{\{z:=v_0\}}\!\downarrow$ implies $a^{\{z:=v_1\}}\!\downarrow$ for arbitrary $a \in \mathbb{E}_z$, by induction on computation length. Assume $a^{\{z:=v_0\}}\!\downarrow$. Consider the first step of computation performed. Suppose a steps uniformly for all $v' \in \mathbb{L}$, i.e. there exists an a' such that $a^{\{z:=v'\}} \mapsto_1 a'^{\{z:=v'\}}$ for all $v' \in \mathbb{L}$. Then, the conclusion follows directly by the induction hypothesis. So, consider steps not uniform in $\mathbb{L}$. By inspection of the rules, it must have been an (app) step starting from $a = R[z\texttt{(}v\texttt{)}]$ for some R, v. Then, $(R[v_0\texttt{(}v\texttt{)}])^{\{z:=v_0\}}\!\downarrow$ implies $(R[v_0\texttt{(}v\texttt{)}])^{\{z:=v_1\}}\!\downarrow$ by the induction hypothesis, and the uniformity of this step in the z occurring free in R, v. And, since v_0 now sits in an applicative reduction context here, the assumption may be applied to yield $(R[v_1\texttt{(}v\texttt{)}])^{\{z:=v_1\}}\!\downarrow$. □

We now list a collection of basic $\cong$ / $\precsim$ properties, all derivable from the CIU Theorem. These properties will be used implicitly in proofs that follow.

LEMMA 3.6 (BASIC $\precsim$ / $\cong$ PROPERTIES)

(i) $\texttt{bot} \precsim a$.

(ii) For $a \in \mathbb{E}_\emptyset$, $a\!\uparrow$ iff $a \cong \texttt{bot}$.

(iii) $R[\texttt{bot}] \cong \texttt{bot}$.

(iv) $\cong$ respects computation, i.e. $a \cong b$ if $a \mapsto b$.

(v) For $D \in \{\mathbb{L}, \mathbb{P}, \{0\}, \{1\}, \ldots\}$, if a, $b \in \mathbb{E}_\emptyset$, $a \precsim b$, and $a \mapsto v$ for $v \in D$, then $b \mapsto v'$ for some $v' \in D$.

(vi) If $a \in \mathtt{pr}(\mathbb{E}, \mathbb{E})$, then $a \cong \mathtt{pr}(\mathtt{fst}(a), \mathtt{snd}(a))$.

(vii) If $v \in \mathbb{L}$ and $y \notin \mathrm{FV}(v)$, then $v \cong \lambda y.v(y)$.

PROOF: We prove each case in turn, implicitly allowing switches between $\precsim$ and $\precsim^{\mathrm{ciu}}$ by Theorem 3.4. (i) is direct since $R[\mathtt{bot}]\uparrow$. For (ii), $a\uparrow$ iff $(\forall R)R[a]\uparrow$ iff $((\forall R)R[a]\uparrow$ iff $R[\mathtt{bot}]\uparrow)$ iff $a \cong^{\mathrm{ciu}} \bot$. For (iii), it suffices to show $R[\mathtt{bot}] \cong^{\mathrm{ciu}} \mathtt{bot}$, which is direct from the definition of $\cong^{\mathrm{ciu}}$. For (iv), by Lemma 2.9 cases (ii) and (iii), $R[a^\sigma] \mapsto R[b^\sigma]$, so $R[a^\sigma]\downarrow$ iff $R[b^\sigma]\downarrow$ and $a \cong^{\mathrm{ciu}} b$ by the definition of $\cong^{\mathrm{ciu}}$. For (v), for each D the proof is similar, we prove it for $D = \mathbb{L}$. There are two cases where $b \mapsto_1 v'$ for $v' \in \mathbb{L}$ fails: either $b\uparrow$, or $v' \notin \mathbb{L}$. The former is ruled out as follows: by (ii), then $b \cong \mathtt{bot}$, and by (iv) $v \precsim b \cong \mathtt{bot}$, but $v \precsim \mathtt{bot}$ is a contradiction for the case $C = \bullet$. Now supposing $v' \notin \mathbb{L}$, it suffices to show $v \precsim v'$ fails, by (iv). picking $C = \mathtt{if}(\mathtt{islam}(\bullet), 1, \mathtt{bot})$ accomplishes this: $C[v]\downarrow$ but $C[v']\uparrow$. For (vi), let $a = \mathtt{pr}(a_0, a_1)$. It suffices to show $R[\mathtt{pr}(a_0, a_1)]\downarrow$ iff $R[\mathtt{pr}(\mathtt{fst}(\mathtt{pr}(a_0, a_1)), \mathtt{snd}(\mathtt{pr}(a_0, a_1)))]\downarrow$. Observing $R[\mathtt{pr}(\mathtt{fst}(\mathtt{pr}(a_0, a_1)), \mathtt{snd}(\mathtt{pr}(a_0, a_1)))] \mapsto R[\mathtt{pr}(a_0, a_1)]$ establishes the result. For (vii), By Theorem 3.5 it suffices to show $v(v') \cong (\lambda y.v(y))(v')$, which is direct by computing via (iv) above. □

3.2 Other Notions of Ordering and Equivalence

We briefly compare $\precsim^{\mathrm{ciu}}$ with some other characterizations of $\precsim$ found in the literature, in particular applicative approximation $\precsim^{\mathrm{app}}$ of [Jim and Meyer, 1991] and applicative bisimulation $\precsim^{\mathrm{bisim}}$ of [Abramsky, 1990]. The main difference centers on the fact that our R may be of the form $v(\bullet)$, while $\precsim^{\mathrm{app}}$ / $\precsim^{\mathrm{bisim}}$ have no such case in their definition. Here we show for this particular language all notions are in fact equivalent.

The distinction between call-by-name and call-by-value becomes important here; in a call-by-name reduction system [Smith, 1992], arguments to functions are not first computed to a value, and $v(\bullet)$ would thus not be included amongst the reduction contexts. Without this case $\precsim^{\mathrm{ciu}}$ and $\precsim^{\mathrm{bisim}}$/$\precsim^{\mathrm{app}}$ are of a very similar character and may be easily shown equivalent.

Consider then the call-by-value case under study here. What we show is the $v(\bullet)$ case may be removed from the definition of reduction context, giving an ordering $\precsim^{\mathrm{app}}$ such that $\precsim^{\mathrm{app}}$ is equivalent to $\precsim^{\mathrm{ciu}}$. We lastly show $\precsim^{\mathrm{app}}$ equivalent to $\precsim^{\mathrm{bisim}}$, and thus all notions are equivalent. For simplicity, we work over closed terms only.

DEFINITION 3.7 (APPLICATIVE APPROXIMATION $\precsim^{\mathrm{app}}$) Let the applicative reduction contexts be $\mathbb{R}^{\mathrm{app}} = \mathbb{R} - \mathbb{R}[\mathtt{app}(\mathbb{V}, \mathbb{R})]$, with P ranging over $\mathbb{R}^{\mathrm{app}}$, and for $a, b \in \mathbb{E}_\emptyset$ define

$$a \precsim^{\mathrm{app}} b \;\Leftrightarrow\; (\forall P \;\big|\; P[a], P[b] \in \mathbb{E}_\emptyset)(P[a] \ll P[b])).$$

LEMMA 3.8 $(\forall a, b \in \mathbb{E}_\emptyset)(a \precsim^{\mathrm{app}} b \;\Leftrightarrow\; a \precsim^{\mathrm{ciu}} b)$.

PROOF: The $\Leftarrow$ direction is trivial from the definitions. We prove the $\Rightarrow$ direction. A rough idea of this proof is as follows. What the $v(\bullet)$ case does upon execution is copy the hole value to multiple points in the function v; however, this copying operation has no real effect since the hole is not touched, and it is only when the hole is touched that its contents matter. What the proof then needs to do is to keep track of these hole values as they are copied around. The difficult part of

the proof is that the hole could contain a higher-order object: the hole could be copied into v, later applied to some value which returns a functional result, which in turn gets copied around some more, and applied at some still later time, etc. All of these intermediate points must be recorded in a list of applicative uses of the hole value. This list is finite because the computation eventually terminates.

For closed a we let $[\![a]\!]$ denote the value expression a reduces to. $[\![a]\!] = v$ if $a \mapsto v$, and write $[\![a]\!]\uparrow$ if no such value expression exists. It suffices to prove the result for the case a and b are values v_0 and v_1: first, we show it suffices for a to be a value v_0. Proceed by cases on whether $a\downarrow$. If not, then the Lemma vacuously holds. If so, then $a \mapsto v_0$ for some v_0, and then by Lemma 3.6 (iv), it then suffices to show $v_0 \precsim^{\text{app}} b \Rightarrow v_0 \precsim^{\text{ciu}} b$ by Lemma 2.9 (iii). Now, proceed by cases on whether $b\downarrow$. If not, then $v_1 \precsim^{\text{app}} b$ is false by the case $P = \bullet$, so the Lemma vacuously holds. If $b\downarrow$, then $b \mapsto v_1$ for some v_1, and by reasoning analogous to that for $a\downarrow$ above, it then suffices to show $v_0 \precsim^{\text{app}} v_1 \Rightarrow v_0 \precsim^{\text{ciu}} v_1$. Now, assuming $v_0 \precsim^{\text{app}} v_1$, the following generalized statement is proven:

$$(\forall e, P_1, \ldots, P_n \mid e \in \mathbb{E}_{\{x_1,\ldots,x_n\}}, P_i \in \mathbb{R}^{\text{app}}{}_{\{x_1,\ldots,x_{i-1}\}})$$
$$((\bigwedge_{\substack{1\le i\le n\\ j<2}} [\![{P_i}^{\sigma[v_j]}[v_j]]\!]\downarrow) \Rightarrow e^{\sigma[v_0]}\downarrow \Rightarrow e^{\sigma[v_1]}\downarrow)$$

where $\sigma[v_j]$ abbreviates $\{x_1 := [\![P_1[v_j]]\!]\} \circ \ldots \circ \{x_n := [\![P_n[v_j]]\!]\}$ for $j < 2$, $e^{\{y_0:=[\![e_0]\!]\}\circ\{y_1:=[\![e_1]\!]\}}$ abbreviates $(e^{\{y_1:=[\![e_1^{\{y_0:=[\![e_0]\!]\}}]\!]\}})^{\{y_0:=[\![e_0]\!]\}}$, and each x_i is distinct. From this, $v_0 \precsim^{\text{ciu}} v_1$ follows by picking $n = 1$ and $P_1 = \bullet$. We proceed by induction on the length of the computation of $e^{\sigma[v_0]}$. Consider the next step of computation; if no x_i is touched, the induction hypothesis establishes the conclusion. Consider then the cases where x_i is touched for some $i \le n$. We focus on the case of function application, $e = R[\mathtt{app}(x_i, v)]$. By assumption, $R^{\sigma[v_0]}[\mathtt{app}([\![{P_i}^{\sigma[v_0]}[v_0]]\!], v^{\sigma[v_0]})]\downarrow$, so $R^{\sigma[v_0]}[[\![\mathtt{app}({P_i}^{\sigma[v_0]}[v_0], v^{\sigma[v_0]})]\!]]\downarrow$ and in fewer steps since the application has also been computed to a value in the latter.

Define $\sigma'[v_j] = \sigma[v_j] \circ \{x_{n+1} := [\![\mathtt{app}(P_i[v_j], v)]\!]\}$, $j < 2$ and fresh x_{n+1}, and apply the induction hypothesis for the substitution σ' and expression $R[x_{n+1}]$. Observe that $[\![\mathtt{app}({P_i}^{\sigma'[v_1]}, v^{\sigma'[v_1]})]\!]\downarrow$ by assumption $v_0 \precsim v_1$ and the fact that x_{n+1} does not occur free in P_i or v. Thus, we may conclude $R^{\sigma[v_1]}[[\![\mathtt{app}({P_i}^{\sigma[v_1]}[v_1], v^{\sigma[v_1]})]\!]]\downarrow$, and by the definition of $[\![\cdot]\!]$, $R^{\sigma[v_1]}[\mathtt{app}([\![{P_i}^{\sigma[v_1]}[v_1]]\!], v^{\sigma[v_1]})]\downarrow$. □

DEFINITION 3.9 (APPLICATIVE BISIMULATION $\precsim^{\text{bisim}}$) $\precsim^{\text{bisim}}$ is the greatest relation such that for all $a, b \in \mathbb{E}_\emptyset$, $a \precsim^{\text{bisim}} b$ iff

(i) $a \mapsto \lambda x.a' \Rightarrow b \mapsto \lambda x.b' \wedge (\forall v)((\lambda x.a')(v) \precsim^{\text{bisim}} (\lambda x.b')(v))$

(ii) $a \mapsto \mathtt{pr}(a_0, a_1) \Rightarrow b \mapsto \mathtt{pr}(b_0, b_1) \wedge a_0 \precsim^{\text{bisim}} b_0 \wedge a_1 \precsim^{\text{bisim}} b_1$

(iii) $a \mapsto n \Rightarrow b \mapsto n$ for $n \in \mathbb{N}$.

Bisimulation orderings have received considerable attention, e.g. [Ong, 1992, Pitts and Stark, 1993, Gordon, 1994, Pitts, 1994, Howe, 1995]. They have the advantage alluded to above of lacking the $v(\bullet)$ case. One consequence of this is all cases are then "destructive" on the expression, and a coinduction (*i.e.*, greatest fixed-point induction) principle is thus sound. The $\precsim^{\text{ciu}}$ characterization has no coinduction principle. We have never had difficulty establishing properties for $\precsim^{\text{ciu}}$ even though there is no coinduction principle, but it is reasonable to expect coinduction to simplify some proofs. Bisimulation has a significant disadvantage when compared to $\precsim^{\text{ciu}}$, however: it does not

easily extend to languages with state or explicit control operators (an important aim of this paper is to use techniques that are as widely applicable as possible). Although it is possible to define a bisimulation ordering for languages with state that is a congruence [Ritter and Pitts, 1995], there has yet been no bisimulation ordering defined which exactly corresponds to the operational ordering [Pitts and Stark, 1993]. The $\cong^{\mathrm{ciu}}$ form of equivalence in a memory-based language can be shown to correspond to $\cong$; see [Mason and Talcott, 1991, Honsell et al., 1995] for complete definitions and proofs (the latter citation contains a more complete proof of the CIU Theorem).

LEMMA 3.10 (ORDERING EQUIVALENCES)

$$(\forall a, b \in \mathbb{E}_\emptyset)(a \mathrel{\lesssim}^{\mathrm{bisim}} b \Leftrightarrow a \mathrel{\lesssim}^{\mathrm{app}} b \Leftrightarrow a \mathrel{\lesssim}^{\mathrm{ciu}} b \Leftrightarrow a \lesssim b)$$

PROOF: Using the previous Lemma and Theorem 3.4 we only need $a \mathrel{\lesssim}^{\mathrm{bisim}} b \Leftrightarrow a \mathrel{\lesssim}^{\mathrm{app}} b$. The forward direction follows by showing

$$a \mathrel{\lesssim}^{\mathrm{bisim}} b \Rightarrow (\forall P)(P[a] \mathrel{\lesssim}^{\mathrm{bisim}} P[b])$$

by an induction on the size of P. The reverse direction follows by coinduction on the bisimulation definition. □

Ong [Ong, 1992] has proved congruence of $\cong^{\mathrm{bisim}}$ for a family of languages that include call-by-value languages, giving a direct means for showing $\cong$ and $\cong^{\mathrm{bisim}}$ are identical relations.

3.3 The Lack of $\lesssim$-Least Upper Bounds

Operational approximation $\lesssim$ is a pre-order. We will show in this section that this pre-order is not complete. In the following section we will show that there are functions denoted by λ-expressions that are not continuous. These failures are due to missing (uncomputable) limit points. Thus we must look for an alternative pre-order to realize our goal of developing domain theoretic tools in an operational setting.

First, some preliminaries and notation for directed sets of terms are defined. For technical reasons, we only allow directed sets with finitely many free variables, otherwise a directed set may contain all the variables $\mathbb{X}$ free and problems may arise in obtaining fresh variables.

DEFINITION 3.11 ($\lesssim$-DIRECTED SETS Δ) A set A is *directed* iff for all $a, b \in A$, there is some $c \in A$ where $a \lesssim c$ and $b \lesssim c$. We define Δ_X to be the $\lesssim$-directed subsets of $\mathbb{E}_X$ for finite $X \subset \mathbb{X}$ and let $\Delta = \bigcup_{X \in \mathbf{P}_\omega(\mathbb{X})} \Delta_X$.

We let A, B range over Δ, and V range over Δ such that $V \subseteq \mathbb{V}$. We allow directed sets of expressions to be used as subexpressions with the convention $C[A] = \{C[a] \mid a \in A\}$. Value substitutions σ extend pointwise to sets of expressions: $A^\sigma = \{a^\sigma \mid a \in A\}$. Both of these operations clearly preserve directedness:

LEMMA 3.12 (Δ CLOSURE CONDITIONS) If $A, A_1, \ldots, A_n \in \Delta$, $\mathtt{op}^+ \in \mathbb{O}_n^+$, and σ is a value substitution, then

(i) $\lambda x.A \in \Delta$

(ii) $\mathtt{op}^+(A_1, \ldots, A_n) \in \Delta$

(iii) $C[A] \in \Delta$

(iv) $A^\sigma \in \Delta$

We write $\bigsqcup A = a$ to mean that a is a $\precsim$-least upper bound of A.

DEFINITION 3.13 (LEAST UPPER BOUND $\bigsqcup A = a$) $\bigsqcup A = a$ ("A has least upper bound a") iff $a_0 \precsim a$ for all $a_0 \in A$, and for all b, if $a_0 \precsim b$ for all $a_0 \in A$, then $a \precsim b$.

This is a partial operation as the following theorem reveals.

THEOREM 3.14 (INCOMPLETENESS) The pre-order $\langle \mathbb{E}_\emptyset, \precsim \rangle$ is not complete; there exists a directed set $A \in \Delta_\emptyset$ with no $\precsim$-least upper bound.

PROOF: Let ϕ be an uncomputable function mapping $\mathbb{N}$ to $\{0,1\}$. Define the $\precsim$-directed set D_0 as

$$D_0 = \{f_k \mid k \in \mathbb{N}\} \quad \text{where} \quad f_k(n) \mapsto \begin{cases} \phi(n) & \text{for } n \leq k, \\ \texttt{bot} & \text{otherwise} \end{cases}$$

observing that functions f_k are computable since they have finitely many non-`bot` values. This set cannot have an upper bound, for any upper bound is a computation coding uncomputable ϕ. $\precsim$ is thus not complete. Also note that some directed sets have upper bounds, but no least upper bound. The set

$$D_1 = \{d_k \mid k \in \mathbb{N}\} \quad \text{where} \quad d_k(n) \mapsto \begin{cases} 0 & \text{if } n \leq k \text{ and } \phi(n) = 0, \\ \texttt{bot} & \text{otherwise} \end{cases}$$

has the upper bound $d = \lambda x.0$, but for instance assuming $\phi(k) = 1$,

$$\lambda x.\texttt{if}(\texttt{nateq}(x,k), \texttt{bot}, 0)$$

is a smaller upper bound. It is easy to show by a computability argument that no least upper bound of D_1 exists. □

3.4 The Failure of $\precsim$-Continuity

DEFINITION 3.15 (CONTINUITY) For $f \in \mathbb{L}_\emptyset$, f is *continuous* iff $\bigsqcup A = a$ implies $\bigsqcup f\texttt{(}A\texttt{)} = f\texttt{(}a\texttt{)}$.

THEOREM 3.16 (DISCONTINUITY) Application is not continuous. There exist A, a, and g such that $\bigsqcup A = a$, but not $\bigsqcup g\texttt{(}A\texttt{)} = g\texttt{(}a\texttt{)}$.

PROOF: The failure of continuity is shown by counterexample. Before giving A, a, and g, let us motivate their construction. As already observed, the key problem is the missing points corresponding to uncomputable functions. We define A with least upper bound a, is such a way that a is least for artificial reasons, i.e. because the "ideal" least upper bound is uncomputable. We can then detect this artificiality by applying a function g to A and a that makes the "ideal" least upper bound of $g\texttt{(}A\texttt{)}$ computable again, demonstrating a discontinuity in g's behavior.

DEFINITION 3.17 Expressions a_k, a, g, and c_k, for $k \in \mathbb{N}$, and directed set A are defined as follows:

$a_k = \lambda x.\texttt{if}(\texttt{islam}(x), \texttt{if}(\texttt{nateq}(x\texttt{(}f_k\texttt{)}, 0), 0, \texttt{bot}), 0)$

$A = \{a_k \mid k \in \mathbb{N}\}$

$a = \lambda x.0$

$g = \lambda x.x\texttt{(}\lambda y.\texttt{bot}\texttt{)}$

$c_k = \lambda x.\texttt{if}(\texttt{nateq}(x\texttt{(}0\texttt{)}, \phi(0)), \ldots \texttt{if}(\texttt{nateq}(x\texttt{(}k\texttt{)}, \phi(k)), 0, \texttt{bot}), \ldots, \texttt{bot})$

A is trivially directed. Before proving $\bigsqcup A = a$, some auxiliary lemmas are established. The functions c_k are "checker" functions that recognize expressions at least as defined as f_k.

LEMMA 3.18 For all k and e, if $c_k(e) \mapsto 0$ then $f_k \sqsubseteq\!\!\!\sim e$.

PROOF: By induction on k, using Theorem 3.5. □

LEMMA 3.19 Given some a_{ub} such that $a_k \sqsubseteq\!\!\!\sim a_{\text{ub}}$ for every $a_k \in A$, it then follows that $a_{\text{ub}}(\lambda x.b) \mapsto 0$, and furthermore $a_{\text{ub}}(\lambda x.b)$ computes uniformly in b.

PROOF: Suppose the computation were not uniform. By inspection of the rules, $\lambda x.b$ must then be applied (`islam` executes independently of the function body). Consider the first such application in the course of computation: $a_{\text{ub}}(\lambda x.b) \mapsto R[(\lambda x.b)(v)]$. Thus by uniformity, we also have $a_{\text{ub}}(c_k) \mapsto R[c_k(v)]$, for all k. If $c_k(v)\downarrow$ for all k, by Lemma 3.18 $f_k \sqsubseteq\!\!\!\sim v$ for all k, but v would thus have the behavior of ϕ and contradict its uncomputability. Thus, $c_n(v)\uparrow$ for some n and thus $a_{\text{ub}}(c_n)\uparrow$, but $a_n(c_n)\downarrow$, contradicting the assumption that $a_n \sqsubseteq\!\!\!\sim a_{\text{ub}}$. Thus, the computation is uniform. So since $a_0 \sqsubseteq\!\!\!\sim a_{\text{ub}}$ and $a_0(\lambda x.0) \mapsto 0$, $a_{\text{ub}}(\lambda x.b) \mapsto 0$. □

LEMMA 3.20 $\bigsqcup A = a$.

PROOF: $a_k \sqsubseteq\!\!\!\sim a$ for all $a_k \in A$ trivially. Suppose for all $a_k \in A$, $a_k \sqsubseteq\!\!\!\sim a_{\text{ub}}$ for some a_{ub}, show $a \sqsubseteq\!\!\!\sim a_{\text{ub}}$. We in fact show something stronger, $a_{\text{ub}} \cong a$. By Theorem 3.5, it suffices to show that $a_{\text{ub}}(v) \cong 0$. We proceed by cases on $v \in \mathbb{L}$.
CASE $v \in \mathbb{L}$: Then by Lemma 3.19, $a_{\text{ub}}(v) \cong 0$.
CASE $v \notin \mathbb{L}$: Then by inspection of the definition, $a_k(v) \cong 0$, so since $a_k \sqsubseteq\!\!\!\sim a_{\text{ub}}$, $0 \sqsubseteq\!\!\!\sim a_{\text{ub}}(v)$ and thus $a_{\text{ub}}(v) \cong 0$. □

The proof of the theorem is now straightforward. $\bigsqcup A = a$ by the previous Lemma, but $g(a_k) \cong \texttt{bot}$ for all k, and $g(a) \cong 0$ and clearly $\bigsqcup\{\texttt{bot}\} = 0$ fails. □

4 Directed Set Ordering and Equivalence

In the previous section we demonstrated that the pre-order $\langle \mathbb{E}_\emptyset, \sqsubseteq\!\!\!\sim \rangle$ was not complete due to the lack of limit points for directed sets. In this section we rectify this shortcoming, and in the process of doing so prove other useful results, including an operational analogue to the least fixed-point theorem. The primary tool used is a simple pre-ordering, $\sqsubseteq\!\!\!\sim_{\text{s}}$, defined on $\sqsubseteq\!\!\!\sim$-directed sets of expressions. One view is that these sets serve to represent the uncomputable limit points. This pre-ordering also has the nice property that $a \sqsubseteq\!\!\!\sim b \Leftrightarrow \{a\} \sqsubseteq\!\!\!\sim_{\text{s}} \{b\}$. The obvious definition of $\sqsubseteq\!\!\!\sim_{\text{s}}$,

$$A \sqsubseteq\!\!\!\sim_{\text{s}}^{\text{obvious}} B \Leftrightarrow (\forall a \in A)(\exists b \in B)(a \sqsubseteq\!\!\!\sim b)$$

is not particularly useful since Lemma 4.11 below will fail: there is a functional f such that

$$\{\texttt{fix}(f)\} \sqsubseteq\!\!\!\sim_{\text{s}}^{\text{obvious}} \{\lambda x.\texttt{bot}, f(\lambda x.\texttt{bot}), f(f(\lambda x.\texttt{bot})), \ldots, f^k(\lambda x.\texttt{bot}), \ldots\}$$

would fail to hold. This has the additional consequence that extensionality fails for $\sqsubseteq\!\!\!\sim_{\text{s}}^{\text{obvious}}$.

To motivate a more useful definition of $A \sqsubseteq\!\!\!\sim_{\text{s}} B$ we note that for the purpose of observing termination, any context can use only a finite amount of information about what fills its holes.

Thus what we care about is that for any use (context) of an element a in A, there is some element b of B that can be used without losing termination. Note that different contexts may require different elements of B.

We first formalize the above intuition, defining the approximation relation $\lesssim_s$. We then establish some basic properties, including a least-fixed point property for $\cong_s$. Next, we develop the theory of $\lesssim$-finite expressions. A syntactic notion of projection, π^n, is defined and used to characterize the finite expressions. The theory of finite expressions is used to show $\lesssim_s$-least upper bounds of $\lesssim_s$-directed sets of expressions always exist, and $\lesssim_s$ is thus a complete pre-order. We conclude by demonstrating that $\lesssim_s$ is ω-algebraic, and that the natural extensions of the primitive operations are continuous.

4.1 Basic Properties of $\lesssim_s$

We begin with the definition of $\lesssim_s$ alluded to above, recalling from the previous section that Δ is the set of $\lesssim$-directed sets and A and B range over Δ.

DEFINITION 4.1 (SET RELATIONS $\lesssim_s$, $\cong_s$) For $A, B \in \Delta$, define

$$A \lesssim_s B \Leftrightarrow (\forall a \in A)(\forall C \in \mathbb{C} \mid C[A], C[B] \subseteq \mathbb{E}_\emptyset)(\exists b \in B)(C[a] \ll C[b])$$

$$A \cong_s B \Leftrightarrow A \lesssim_s B \wedge B \lesssim_s A$$

Some basic properties of $\lesssim_s$ include the following.

LEMMA 4.2 (ELEMENTARY $\lesssim_s$ / $\cong_s$ PROPERTIES)

(i) $\lesssim_s$ is a pre-congruence: $A \lesssim_s B \Rightarrow C[A] \lesssim_s C[B]$.

(ii) $\cong_s$ is a congruence: $A \cong_s B \Rightarrow C[A] \cong_s C[B]$.

(iii) $\{a\} \lesssim_s \{b\} \Leftrightarrow a \lesssim b$.

(iv) $A \lesssim_s \{b\} \Leftrightarrow (\forall a \in A)(a \lesssim b)$.

(v) $a \in A \Rightarrow \{a\} \lesssim_s A$.

PROOF: (i)-(v) are direct from the definitions. □

A counterexample to $A \lesssim_s B \wedge a \in A \Rightarrow (\exists b \in B)(a \lesssim b)$ is given by the A and B of Lemma 4.11 below. The following two lemmas relate $A \cong_s \{a\}$ to the domain notion of both A having lub a and A having glb a.

LEMMA 4.3 (GREATEST LOWER BOUND) $\{a\} \cong_s A$ iff $\{a\} \lesssim_s A$ and for all a', if $\{a'\} \lesssim_s A$, then $a' \lesssim a$.

PROOF: The forward implication is trivial. For the reverse implication we need only show $A \lesssim_s \{a\}$ (the other direction of $\cong_s$ is given). This is achieved by showing $a' \lesssim a$ for arbitrary $a' \in A$. Since $\{a'\} \lesssim_s A$, this follows directly by assumption. □

Returning to the discussion of $\lesssim$-directed set lubs in the previous section, we concluded the notion of lub, $\bigsqcup A \cong a$, was not useful since it was not continuous. Using this new ordering we can define a related property, $A \cong_s \{a\}$. By the congruence of $\cong_s$, we have $f(A) \cong_s \{f(a)\}$, the continuity property that failed for lub. Thus, the two properties must be different. Their difference is captured in the following Lemma.

LEMMA 4.4 $A \cong_{\mathrm{s}} \{a\} \Rightarrow \bigsqcup A \cong a$, and the converse fails.

PROOF: For the forward implication, it suffices to consider the case when A and a are closed. $A \sqsubseteq_{\mathrm{s}} \{a\}$ is trivial. Suppose $A \sqsubseteq_{\mathrm{s}} \{a'\}$. We show $a \sqsubseteq a'$. Expanding the definition of $\sqsubseteq$, we assume $R[a]\downarrow$, and show $R[a']\downarrow$. $\{a\} \sqsubseteq_{\mathrm{s}} A$, so $R[a'']\downarrow$ for some $a'' \in A$; thus, by assumption, $R[a']\downarrow$.
To see that converse fails, suppose the contrary. Assuming $\bigsqcup A \cong a$, we have $A \cong_{\mathrm{s}} \{a\}$, and $f(A) \cong_{\mathrm{s}} \{f(a)\}$. Applying the first case of this lemma, $\bigsqcup f(A) = f(a)$, i.e. continuity, contradicting Theorem 3.16. □

4.2 The CIU Theorem for $\sqsubseteq_{\mathrm{s}}$

As was the case for $\sqsubseteq$, we desire an alternate characterization of $\sqsubseteq_{\mathrm{s}}$, $\sqsubseteq_{\mathrm{s}}^{\mathrm{ciu}}$, that uses fewer contexts in its definition, and which we may show equivalent to $\sqsubseteq_{\mathrm{s}}$. The analogy is very close, $\sqsubseteq_{\mathrm{s}}^{\mathrm{ciu}}$ differs from $\sqsubseteq_{\mathrm{s}}$ in the same manner $\sqsubseteq^{\mathrm{ciu}}$ differs from $\sqsubseteq$: we replace all contexts with closed instances of all uses. One of many uses of this characterization will be to prove the extensionality of $\sqsubseteq_{\mathrm{s}}$, Lemma 4.10 below.

DEFINITION 4.5 (CIU SET ORDERING $\sqsubseteq_{\mathrm{s}}^{\mathrm{ciu}}$) For $A, B \in \Delta$

$$A \sqsubseteq_{\mathrm{s}}^{\mathrm{ciu}} B \Leftrightarrow (\forall a \in A)(\forall \sigma, R \mid R[A^\sigma], R[B^\sigma] \subseteq \mathbb{E}_\emptyset)(\exists b \in B)(R[a^\sigma] \ll R[b^\sigma])$$

The main characterization we desire is,

THEOREM 4.6 (SET ORDERING CIU) $A \sqsubseteq_{\mathrm{s}} B \Leftrightarrow A \sqsubseteq_{\mathrm{s}}^{\mathrm{ciu}} B$.

We give a proof that synthesizes ideas from proofs in [Smith, 1992, Mason and Talcott, 1991, Howe, 1989]. We first give an informal overview of the proof. The $(\Rightarrow)$ direction is not difficult, since $\sqsubseteq_{\mathrm{s}}^{\mathrm{ciu}}$ has a smaller collection of contexts to distinguish expressions than $\sqsubseteq_{\mathrm{s}}$ has. $(\Leftarrow)$ is the difficult direction. This proof uses the observation that it suffices to show $\sqsubseteq_{\mathrm{s}}^{\mathrm{ciu}}$ is a pre-congruence. To establish this, we prove lemmas that establish pre-congruence for single constructors: operators $\mathbb{O}^+$ (Lemma 4.7) and λx (Lemma 4.8) may be placed around sets of expressions while preserving $\sqsubseteq_{\mathrm{s}}^{\mathrm{ciu}}$.

LEMMA 4.7 (SET ORDERING OPERATOR CIU) If $A \sqsubseteq_{\mathrm{s}}^{\mathrm{ciu}} B$ then $\mathtt{op}^+(\bar{c}, A, \bar{d}) \sqsubseteq_{\mathrm{s}}^{\mathrm{ciu}} \mathtt{op}^+(\bar{c}, B, \bar{d})$ for any $\mathtt{op}^+ \in \mathbb{O}^+$.

PROOF: Pick arbitrary $\mathtt{op}^+$, R, σ such that $R[(\mathtt{op}^+(\bar{c}, A, \bar{d}))^\sigma], R[(\mathtt{op}^+(\bar{c}, B, \bar{d}))^\sigma] \subseteq \mathbb{E}_\emptyset$. Since $\mathtt{op}^+$ does not bind, σ may be factored in, so it suffices to show for arbitrary $a \in A$ and arbitrary closed $\bar{c}$ and $\bar{d}$ that

$$R[\mathtt{op}^+(\bar{c}, a^\sigma, \bar{d}))]\downarrow \Rightarrow (\exists b \in B)(R[\mathtt{op}^+(\bar{c}, b^\sigma, \bar{d}))]\downarrow)$$

We proceed by induction on the length of the computation of the assumption.

Assume the conclusion is true for all $\bar{c}, \bar{d}$ with shorter computations. Proceed by cases on whether all elements $\bar{c}$ are values. Suppose so (or if $\bar{c}$ is empty): then, define $R_0 = R[\mathtt{op}^+(\bar{c}, \bullet, \bar{d}))]$, and the conclusion follows directly by assumption. Suppose not. Then there is some c_i such that $c_i \notin \mathbb{V}$ and $c_k \in \mathbb{V}$ for $k < i$. This means we have reduction context

$$R_0 = R[\mathtt{op}^+(c_0, \ldots, c_{i-1}, \bullet, c_{i+1}, \ldots, c_n, a^\sigma, \bar{d})],$$

and by Lemma 2.9 (ii),

$$R[\mathtt{op}^+(c_0,\ldots,c_{i-1},c_i,c_{i+1},\ldots,c_n,a^\sigma,\bar{d})] \mapsto_1 R[\mathtt{op}^+(c_0,\ldots,c_{i-1},c_i',c_{i+1},\ldots,c_n,a^\sigma,\bar{d})],$$

so by induction hypothesis the conclusion is direct. □

LEMMA 4.8 (SET ORDERING LAMBDA CIU) If $A \sqsubseteq^{\text{ciu}}_{\simeq_s} B$, then $\lambda x.A \sqsubseteq^{\text{ciu}}_{\simeq_s} \lambda x.B$.

PROOF: It suffices to prove the Lemma for the case $A, B \subseteq \mathbb{E}_{\{x\}}$, for by the definition of $\sqsubseteq^{\text{ciu}}_{\simeq_s}$ the conclusion then follows. Given arbitrary $R \in \mathbb{R}_\emptyset$, show for fixed $a \in A$

$$R[\lambda x.a]\downarrow \Rightarrow (\exists b \in B)(R[\lambda x.b]\downarrow).$$

Generalize this statement to

$$e^{\{z:=\lambda x.a\}}\downarrow \Rightarrow (\exists b \in B)(e^{\{z:=\lambda x.b\}}\downarrow),$$

for $e \in \mathbb{E}_{\{z\}}$. The original goal follows by letting $e = R[z]$. Proceed by induction on the length of the computation of the assumption. Consider whether e is uniform in z, i.e. whether

$$e \mapsto_1 e'$$

for some e'. If it is uniform, then by Lemma 2.9 (ii),

$$e^{\{z:=\lambda x.e''\}} \mapsto_1 e'^{\{z:=\lambda x.e''\}}, \text{for all } e'',$$

and the result follows directly by induction hypothesis.

Consider the case where e is stuck, i.e. does not reduce. Since $e^{\{z:=\lambda x.a\}}\downarrow$, it does not get stuck when a λ-value is substituted for z. By inspection of the rules, replacing z with a λ-value causes a stuck computation to become un-stuck in two cases. The first is if the redex is $\mathtt{isnat}(z)$ or $\mathtt{ispr}(z)$; but these cases are still uniform for any λ-value and reasoning analogous to the previous uniform case applies. The only other non-uniform case is where $e = R[z(v)]$ for some R, v, containing z possibly free. Consider this case. By inspection of the (app) rule, we have the following:

$$R[(\lambda x.e')(v)] \mapsto_1 R[e'^{\{x:=v\}}],$$

for all expressions e', v, R. In particular, it holds for e' being a or any $b \in B$. It thus suffices to show

$$(\exists b \in B)((R[b^{\{x:=v\}}])^{\{z:=\lambda x.b\}}\downarrow).$$

By the induction hypothesis,

$$(\exists b' \in B)(R[a^{\{x:=v\}}])^{\{z:=\lambda x.b'\}}\downarrow.$$

Then by assumption $A \sqsubseteq^{\text{ciu}}_{\simeq_s} B$, a above can be replaced by some $b'' \in B$ (take σ in the definition of $\sqsubseteq^{\text{ciu}}_{\simeq_s}$ to be $\{x := v^{\{z:=\lambda x.b'\}}\}$), giving

$$(\exists b'' \in B)(R[b''^{\{x:=v\}}])^{\{z:=\lambda x.b'\}}\downarrow.$$

By the directedness of B, we can find b such that $b', b'' \sqsubseteq_\simeq b$, and this means first that

$$(R[b''^{\{x:=v\}}])^{\{z:=\lambda x.b\}}\downarrow.$$

Now by Lemma 3.2 case (vi), $b''^{\{x:=v'\}} \sqsubseteq_\simeq b^{\{x:=v'\}}$ for $v' = v^{\{z:=\lambda x.b\}}$, so

$$(R[b^{\{x:=v\}}])^{\{z:=\lambda x.b\}}\downarrow.$$

□

LEMMA 4.9 ($\sqsubseteq^{\text{ciu}}_{\lesssim_s}$ PRE-CONGRUENCE) $\sqsubseteq^{\text{ciu}}_{\lesssim_s}$ is a pre-congruence, $A \sqsubseteq^{\text{ciu}}_{\lesssim_s} B$ implies $C[A] \sqsubseteq^{\text{ciu}}_{\lesssim_s} C[B]$.

PROOF: We proceed by induction on the size of C. For the base case, either $C = \bullet$ or $C = v$ for $v \in \mathbb{N}$. For the former the result follows by assumption, and for the latter by reflexivity. Otherwise we consider two cases: $C = \lambda x.C_0$; and $C = \mathtt{op}^+(C_0, \ldots, C_n)$ for $\mathtt{op}^+ \in \mathbb{O}^+$. In the first case $C_0[A] \sqsubseteq^{\text{ciu}}_{\lesssim} C_0[B]$ by the induction hypothesis. Thus $\lambda x.(C_0[A]) \sqsubseteq^{\text{ciu}}_{\lesssim} \lambda x.(C_0[B])$ by Lemma 4.8, and hence $(\lambda x.C_0)[A] \sqsubseteq^{\text{ciu}}_{\lesssim} (\lambda x.C_0)[B]$.

In the second case, by induction hypothesis, $C_i[A] \sqsubseteq^{\text{ciu}}_{\lesssim} C_i[B]$, for $i \leq n$. Then, by Lemma 4.7,

$$\begin{aligned}
&\mathtt{op}^+(C_0[A], C_1[A] \ldots C_n[A]) \sqsubseteq^{\text{ciu}}_{\lesssim} \mathtt{op}^+(C_0[B], C_1[A] \ldots C_n[A]) \\
&\quad \sqsubseteq^{\text{ciu}}_{\lesssim} \mathtt{op}^+(C_0[B], C_1[B] \ldots C_n[A]) \\
&\quad \vdots \quad \vdots \\
&\quad \sqsubseteq^{\text{ciu}}_{\lesssim} \mathtt{op}^+(C_0[B], C_1[B] \ldots C_n[B]).
\end{aligned}$$

□

We now prove the main theorem.

PROOF OF THEOREM 4.6: For the forward direction, pick C such that $C[e] \mapsto R[e^\sigma]$ for all e; one such C is

$$C = (\lambda x_1, \ldots, \lambda x_n.R[\bullet])\,\mathtt{(}v_1\mathtt{)} \ldots \mathtt{(}v_n\mathtt{)},$$

where $\mathrm{Dom}(\sigma) = \{x_1, \ldots x_n\}$ and $\sigma(x_i) = v_i$ for $i < n$.

For the reverse direction, we assume $C[a]\!\downarrow$ and find $b \in B$ such that $C[b]\!\downarrow$. By Lemma 4.9, $C[A] \sqsubseteq^{\text{ciu}}_{\lesssim_s} C[B]$. From the definition of $\sqsubseteq^{\text{ciu}}_{\lesssim_s}$, pick $C[a] \in C[A]$, $R = \bullet$, $\sigma = \emptyset$, gives $C[b] \in C[B]$ such that $C[b]\!\downarrow$. □

Note that Theorem 3.4 now follows from Theorem 4.6 and part (iii) of Lemma 4.2.

LEMMA 4.10 ($\sqsubseteq_{\lesssim_s}$ EXTENSIONALITY) For directed $V_0, V_1 \subseteq \mathbb{L}$, $V_0 \sqsubseteq_{\lesssim_s} V_1 \Leftrightarrow (\forall V \in \Delta_\emptyset)(V_0\mathtt{(}V\mathtt{)} \sqsubseteq_{\lesssim_s} V_1\mathtt{(}V\mathtt{)})$.

PROOF: The forward direction follows from the pre-congruence of $\sqsubseteq_{\lesssim_s}$. For other direction assume $(\forall V)(V_0\mathtt{(}V\mathtt{)} \sqsubseteq_{\lesssim_s} V_1\mathtt{(}V\mathtt{)})$. To show $V_0 \sqsubseteq_{\lesssim_s} V_1$ for $V_0, V_1 \subseteq \mathbb{L}$ it suffices, by Theorem 4.6, to assume V_0, V_1 are closed and to show for fixed $v_0 \in V_0$ and all $a \in \mathbb{E}_z$, $a^{\{z:=v_0\}}\!\downarrow$ implies $a^{\{z:=v_1\}}\!\downarrow$ for some $v_1 \in V_1$. Proceed by induction on computation length. Assume $a^{\{z:=v_0\}}\!\downarrow$. Consider the next step of computation performed. If $a^{\{z:=v'\}} \mapsto_1 a'^{\{z:=v'\}}$ for some a' uniformly for all $v' \in \mathbb{L}$, then the conclusion follows directly by the induction hypothesis. So, consider steps not uniform in $\mathbb{L}$. By inspection of the rules, it must have been an (app) step starting from $a = R[z\mathtt{(}v\mathtt{)}]$ for some R, v. Then, $(R[v_0\mathtt{(}v\mathtt{)}])^{\{z:=v_0\}}\!\downarrow$ implies $(R[v_0\mathtt{(}v\mathtt{)}])^{\{z:=v_1\}}\!\downarrow$ by the uniformity of this step in the z occurring free in R, v and the induction hypothesis. And, since v_0 is now being applied, the assumption may be used at $V = \{v\}$ to yield $(R[v_1'\mathtt{(}v\mathtt{)}])^{\{z:=v_1\}}\!\downarrow$ for some $v_1' \in V_1$. Now, since V_1 is directed, pick $v_1'' \in V_1$ such that $v_1'' \sqsupseteq\!\!\gtrsim v_1', v_1$ and we have $(R[v_1''\mathtt{(}v\mathtt{)}])^{\{z:=v_1''\}}\!\downarrow$. □

4.3 Fixed Point Properties

We establish some basic properties of fixed points: fixed points are equivalent to their set of finite unrollings, a least fixed-point property holds, and fixed-point induction is justified. We make the following abbreviation: for a functional $f = \lambda x.\lambda y.a$, define $f^0 = \lambda x.\mathtt{bot}$ and $f^{n+1} = f(f^n)$. The key lemma is the following.

LEMMA 4.11 (FIXED POINT APPROXIMATION) For a functional f,

$$\{\mathtt{fix}(f)\} \cong_s \{f^n \mid n \in \mathbb{N}\}.$$

PROOF: Without loss of generality take f to be closed, for from this case the result follows for arbitrary f by Theorem 4.6. The $\sqsupseteq_{\approx_s}$ direction follows by induction on n; consider then proving $\sqsubseteq_{\approx_s}$. First note $\mathtt{fix}(f) \cong u(u)$ where $u = \lambda x.\lambda z.f(x(x))(z)$, so it suffices to show $\{u(u)\} \sqsubseteq_{\approx_s} \{f^k \mid k \in \mathbb{N}\}$. Expanding definitions, the desired result is

$$(\forall a \in \mathbb{E}_x)(a^{\{x:=u(u)\}}\downarrow \Rightarrow (\exists k)(a^{\{x:=f^k\}}\downarrow)).$$

Assume $a^{\{x:=u(u)\}}\downarrow$, proceed by induction on the length of this computation to show the above statement. Consider the next step of computation performed on $a^{\{x:=u(u)\}}$. If the step is uniform in $u(u)$, the conclusion follows directly by induction hypothesis. Then, consider non-uniform steps; all such cases can easily be seen to be of the form

$$a^{\{x:=u(u)\}} = R^{\{x:=u(u)\}}[u(u)] \mapsto_1 R^{\{x:=u(u)\}}[\lambda z.f(u(u))(z)],$$

we show $R^{\{x:=f^k\}}[f^k]\downarrow$ for some k. By the induction hypothesis, $R^{\{x:=f^{k_0}\}}[\lambda z.f(f^{k_0})(z)]\downarrow$ for some k_0, so since $f^{k_0} \sqsubseteq f^{k_0+1}$ and $\lambda z.f(f^{k_0})(z) \cong f^{k_0+1}$ by extensionality, $R^{\{x:=f^{k_0+1}\}}[f^{k_0+1}]\downarrow$, and letting k be k_0+1, the desired conclusion has been reached. □

LEMMA 4.12 (LEAST FIXED POINT) For a functional f, $\mathtt{fix}(f) \cong f(\mathtt{fix}(f))$, and $(\forall a)(\lambda x.a \cong f(\lambda x.a) \Rightarrow \mathtt{fix}(f) \sqsubseteq \lambda x.a)$

PROOF: The first half follows by computing; for the second half, suppose for arbitrary $\lambda x.a$ that $\lambda x.a \cong f(\lambda x.a)$. $\{f^k \mid k \in \mathbb{N}\} \sqsubseteq_{\approx_s} \{\lambda x.a\}$ follows from showing $f^k \sqsubseteq \lambda x.a$ by induction on k. By Lemma 4.11, $\{f^k \mid k \in \mathbb{N}\} \cong_s \{\mathtt{fix}(f)\}$. Thus, $\{\mathtt{fix}(f)\} \sqsubseteq_{\approx_s} \{\lambda x.a\}$, so $\mathtt{fix}(f) \sqsubseteq \lambda x.a$ by Lemma 4.2 (iii). □

One of the most useful induction principles for proving facts about fixed points is Scott fixed-point induction ([deBakker and Scott, 1969]; see also [Manna, 1974]). The justification of fixed-point induction necessitates functions be continuous in a domain. All that is needed to justify fixed-point induction here is Lemma 4.11.

THEOREM 4.13 (ATOMIC FIXED POINT INDUCTION) For a functional f, if for all k, $C[f^k] \sqsubseteq C'[f^k]$, then $C[\mathtt{fix}(f)] \sqsubseteq C'[\mathtt{fix}(f)]$.

PROOF: By Lemma 4.11 and congruence of $\cong_s$, $\{C[f^k] \mid k \in \mathbb{N}\} \cong_s \{C[\mathtt{fix}(f)]\}$ and $\{C'[f^k] \mid k \in \mathbb{N}\} \cong_s \{C'[\mathtt{fix}(f)]\}$. Then, using the fact $\{C[f^k] \mid k \in \mathbb{N}\} \sqsubseteq_{\approx_s} \{C'[f^k] \mid k \in \mathbb{N}\}$ (by definition of $\sqsubseteq_{\approx_s}$) and the above equivalences, the result is immediate. □

It is a simple matter to extend this theorem to logical formulas in which statements $C[\mathtt{fix}(f)] \sqsubseteq C'[\mathtt{fix}(f)]$ occur, although only certain *admissible* formulas admit a fixed-point induction principle [Paulson, 1987, Igarashi, 1972].

4.4 Finite Expressions

An important tool in the further development of the theory of $\sqsubseteq$ and $\sqsubseteq_{\mathrm{s}}$ are the finite expressions. Most importantly here, they will be used to show $\sqsubseteq_{\mathrm{s}}$ is complete. Finite expressions are critical to a number of constructions, including the ideal model construction [MacQueen et al., 1984, Abadi et al., 1991b]. Construction of finite expressions is the first point in the paper where the presence of recognizers `ispr` and `isnat` in the language become critical. The following definition of finite expressions relies on the representation of limits by directed sets of expressions.

DEFINITION 4.14 (FINITE EXPRESSIONS $\mathbb{E}^\omega$) The set of finite expressions $\mathbb{E}^\omega$ is defined by

$$\mathbb{E}^\omega = \{d \in \mathbb{E}_\emptyset \mid (\forall A \in \Delta_\emptyset)(\{d\} \sqsubseteq_{\mathrm{s}} A \Rightarrow (\exists a \in A)(d \sqsubseteq a))\}$$

We hereafter let d range over $\mathbb{E}^\omega$. Note we define *closed* finite expressions only; it is possible to generalize to allow open finite expressions, but the resulting definitions are more complex, and for our purposes working over closed expressions suffices. Finite expressions *per se* are of little use without a stronger characterization of their structure. To this end we define expressions π^n that compute the "finite projections" familiar from the inverse limit domain construction. The key results are:

(i) The finite approximation property, $\{\lambda x.x\} \cong_{\mathrm{s}} \{\pi^n \mid n \in \mathbb{N}\}$ (Theorem 4.19);

(ii) the range of π^n is finite (modulo $\cong$) for each n (Lemma 4.21);

(iii) $\mathbb{E}^\omega$ may be characterized as the union of the images of the finite projections π^n (Lemma 4.23).

The idea of syntactically defined projection functions is found in [Abadi et al., 1991b], though the uses we put them to here are significantly different.

DEFINITION 4.15 (FINITE PROJECTIONS π^n) The projection functional π, finite projections π^n, and infinite projection π^∞ are defined as follows.

$$\begin{aligned}
\pi &= \lambda y.\lambda x.\\
&\qquad \mathtt{if}(\mathtt{isnat}(x), \mathtt{if}(\mathtt{iszero}(x), 0, \mathtt{succ}(y(\mathtt{pred}(x)))),\\
&\qquad \mathtt{if}(\mathtt{ispr}(x), \mathtt{pr}(y(\mathtt{fst}(x)), y(\mathtt{snd}(x))),\\
&\qquad \mathtt{if}(\mathtt{islam}(x), y \circ x \circ y\\
&\qquad \mathtt{bot})))\\
\pi^0 &= \lambda x.\mathtt{bot}\\
\pi^{n+1} &= \pi(\pi^n)\\
\pi^\infty &= \mathtt{fix}(\pi)
\end{aligned}$$

Observe the syntactic construction of projections π^n would be impossible without recognizers `isnat`, `ispr`. The following lemma serves to characterize the basic properties of the projections.

LEMMA 4.16 (ELEMENTARY π^n/π^∞ PROPERTIES)

(fix) $\quad \pi^\infty \cong_{\mathrm{s}} \{\pi^n \mid n \in \mathbb{N}\}$

(idemp) $\quad \pi^n \circ \pi^n \cong \pi^n, \quad \pi^\infty \circ \pi^\infty \cong \pi^\infty$

(compose) $\pi^m \circ \pi^n \cong \pi^{\min(m,n)}$

(order) $\pi^n \sqsubseteq \pi^{n+1} \sqsubseteq \pi^\infty$

(num.0) $\pi^n(m) \cong \mathtt{bot} \qquad \text{if } m \geq n$

(num.+) $\pi^n(m) \cong m \qquad \text{if } m < n, \qquad \pi^\infty(m) \cong m$

(pair) $\pi^{n+1}(\mathtt{pr}(v_0, v_1)) \cong \mathtt{pr}(\pi^n(v_0), \pi^n(v_1)), \quad \pi^\infty(\mathtt{pr}(v_0, v_1)) \cong \mathtt{pr}(\pi^\infty(v_0), \pi^\infty(v_1))$

(fun.0) $\pi^1(\lambda x.e) \cong \lambda x.\mathtt{bot}$

(fun.+) $\pi^{n+1}(\lambda x.a) \cong \pi^n \circ \lambda x.a \circ \pi^n, \quad \pi^\infty(\lambda x.a) \cong \pi^\infty \circ \lambda x.a \circ \pi^\infty$

(prune) $\pi^n(a) \sqsubseteq a, \quad \pi^\infty(a) \sqsubseteq a$

PROOF: (fix) follows from Lemma 4.11. The first part of (idemp) follows by induction on n and computation. The second part follows from (fix) and the congruence of $\cong_s$, using context $C = \bullet \circ \bullet$. (compose) may be proved by showing $\pi^{n+1} \circ \pi^n \cong \pi^n$ and $\pi^n \circ \pi^{n+1} \cong \pi^n$ by induction on n, and then composing one of these two facts $m - n$ or $n - m$ times. (order) follows by induction on n. The first part of (prune) is proved by induction on n and the previous π^n facts, noting it suffices to consider the case where a is a closed value. The second part is proved from the first part and (fix): $\{\pi^\infty(a)\} \cong_s \{\pi^n(a) \mid n \in \mathbb{N}\} \sqsubseteq_s \{\lambda x.x\}$. The remaining cases are obvious from the definitions. □

π^∞ may be characterized as nothing but a fancy identity function; from this the finite approximation property alluded to previously will be an immediate corollary.

THEOREM 4.17 (IDENTITY OF π^∞) $\pi^\infty \cong \lambda x.x$

To prove this, we inductively define $\tau(a)$ and $\tau(R)$ as follows:

$\tau(x) = x$

$\tau(n) = n$

$\tau(\mathtt{op}(a_0, \ldots, a_n)) = \pi^\infty(\mathtt{op}(\tau(a_0), \ldots, \tau(a_n)))$

$\tau(\mathtt{pr}(a_0, a_1)) = \mathtt{pr}(\tau(a_0), \tau(a_1))$

$\tau(\lambda x.a) = \pi^\infty \circ \lambda x.\tau(a) \circ \pi^\infty$

$\tau(R) = \tau(R[x])^{\{x:=\bullet\}}$

The τ-expressions are an intermediate form that expresses how π^∞ subexpressions can distill throughout an expression in the course of computing $\pi^\infty(a)$. Basic properties of these expressions include the following.

LEMMA 4.18

(i) For $a \in \mathbb{E}_\emptyset$, $\pi^\infty(\tau(a)) \cong \tau(a)$.

(ii) $\tau(a) \sqsubseteq a$, and $\tau(R[x]) \sqsubseteq R[x]$.

(iii) $\tau(R[b]) = \tau(R)[\tau(b)]$, and $\tau(a^{\{x:=v\}}) = \tau(a)^{\{x:=\tau(v)\}}$.

PROOF: For (i), proceed by induction on the structure of a. If $a \in \mathbb{N}$, the result follows by $\mathbb{N}$ induction. If $a = \mathtt{op}(a_0, \ldots, a_n)$, then $\tau(a) = \pi^\infty(\mathtt{op}(\tau(a_0), \ldots, \tau(a_n)))$, so by Lemma 4.16 (idemp),

$$\pi^\infty(\tau(a)) \cong \pi^\infty(\pi^\infty(\mathtt{op}(\tau(a_0), \ldots, \tau(a_n)))) \cong \pi^\infty(\mathtt{op}(\tau(a_0), \ldots, \tau(a_n))) \cong \tau(a).$$

If $a \in \mathbb{P}$, then $\tau(a) = \mathtt{pr}(\tau(a_0), \tau(a_1))$. Observe if either $\tau(a_0)\uparrow$ or $\tau(a_1)\uparrow$, $\tau(a)\uparrow$ and the result is then trivial. So, assume without loss of generality that $\tau(a_i) \mapsto v_i$, $i < 2$. Then, by Lemma 4.16 (pair), $\pi^\infty(\mathtt{pr}(v_0, v_1)) \cong \mathtt{pr}(\pi^\infty(v_0), \pi^\infty(v_1))$, so substituting gives $\pi^\infty(\mathtt{pr}(\tau(a_0), \tau(a_1))) \cong \mathtt{pr}(\pi^\infty(\tau(a_0)), \pi^\infty(\tau(a_1)))$ and the result follows by induction. Finally, if $a = \lambda x.a'$, $\tau(a)$ is of the form $\pi^\infty \circ \lambda x.\tau(a') \circ \pi^\infty$, and $\pi^\infty(\tau(a)) \cong \pi^\infty \circ \tau(a) \circ \pi^\infty$ by computing, which expands to $\pi^\infty \circ \pi^\infty \circ \lambda x.\tau(a') \circ \pi^\infty \circ \pi^\infty$ and by Lemma 4.16 (idemp) simplifies then to $\pi^\infty \circ \lambda x.\tau(a') \circ \pi^\infty \cong \tau(a)$. For (ii), this easily follows by induction on the structure of a/R, using Lemma 4.16 (prune) and Theorem 3.5. Property (iii) follows by structural induction on R or a. □

PROOF OF THEOREM 4.17: The $\sqsubseteq$ direction follows from Lemma 4.16 (prune) and Theorem 3.5. For the $\sqsupseteq$ direction, we successively rephrase the statement five times. It suffices to show for all a that $R[a]\downarrow \Rightarrow R[\pi^\infty(a)]\downarrow$ by Theorems 3.5 and 3.4. For this it then suffices to show $R[a]\downarrow \Rightarrow R[\pi^\infty(\tau(a))]\downarrow$ by Lemma 4.18 (ii). Then, by Lemma 4.18 (i), it suffices to show $R[a]\downarrow \Rightarrow R[\tau(a)]\downarrow$. Next, generalizing it suffices to show $a_0\downarrow \Rightarrow \tau(a_0)\downarrow$ by Lemma 4.18 (ii) and (iii): $R[a]\downarrow \Rightarrow \tau(R[a])\downarrow \Rightarrow \tau(R)[\tau(a)]\downarrow \Rightarrow R[\tau(a)]\downarrow$. And lastly, to show this it suffices to show $a_0 \mapsto_1 a_1 \Rightarrow \tau(a_0) \cong \tau(a_1)$, for the conclusion then follows by induction on computation length and the observation that $\tau(v) \in \mathbb{V}$ for any value v.

So, assume $a_0 \mapsto_1 a_1$, show $\tau(a_0) \cong \tau(a_1)$. Consider this step of computation; $a_0 = R[a]$ for some redex a, proceed by cases on the form of a.

If $a = \mathtt{app}(\lambda x.c, v)$, then $a_1 = R[c^{\{x:=v\}}]$. By inspection of the definitions of $\tau(a)$ and $\tau(R)$, $\tau(a_0)$ must be of the form

$$\tau(R)[\pi^\infty(\mathtt{app}(\pi^\infty \circ \lambda x.\tau(c) \circ \pi^\infty, \tau(v)))].$$

Computing from this point yields

$$\cong \tau(R)[\pi^\infty(\pi^\infty(\mathtt{app}(\lambda x.\tau(c), \pi^\infty(\tau(v)))))]$$

$$\cong \tau(R)[\pi^\infty(\mathtt{app}(\lambda x.\tau(c), \tau(v)))] \qquad \text{by Lemmas 4.16 (idemp) and 4.18 (i)}$$

$$\cong \tau(R)[\pi^\infty(\tau(c)^{\{x:=\tau(v)\}})]$$

$$\cong \tau(R)[\tau(c)^{\{x:=\tau(v)\}}] \qquad \text{by Lemma 4.18 (i) and (iii),}$$

and $\tau(R[c^{\{x:=v\}}]) = \tau(R)[\tau(c)^{\{x:=\tau(v)\}}]$ by Lemma 4.18 (iii).

If a is any other redex, the proof is similar to the previous case. □

The Finite Approximation Theorem is now a simple corollary.

THEOREM 4.19 (FINITE APPROXIMATION) $\{\pi^n \mid n \in \mathbb{N}\} \cong_s \{\lambda x.x\}$.

PROOF: $\{\pi^n \mid n \in \mathbb{N}\} \cong_s \{\pi^\infty\} \cong_s \{\lambda x.x\}$ by Lemmas 4.16 and 4.17. □

Now we justify the use of the term "finite", and characterize all finite expressions d as those equivalent to the projection of some expression, $d \cong \pi^n(e)$.

DEFINITION 4.20 ($\mathbb{E}^n$) Define $\mathbb{E}^n = \{a \mid a \in \mathbb{E}_\emptyset \wedge a \cong \pi^n(a)\}$.

LEMMA 4.21 (FINITE CARDINALITY) For all $n \in \mathbb{N}$, $\mathbb{E}^n$ contains finitely many $\cong$-distinct expressions.

PROOF: The proof is by induction on n, using Theorem 3.5 and the observation that there are only finitely many functions that map a finite set to a finite set. □

LEMMA 4.22 (FINITENESS OF PROJECTIONS) For all $a \in \mathbb{E}_\emptyset$ and $n \in \mathbb{N}$, $\pi^n(a)$ is finite.

PROOF: Given an arbitrary set A with $\{\pi^n(a)\} \lesssim_s A$, find $a_0 \in A$ such that $\pi^n(a) \lesssim a_0$. Observe that $\pi^n(A)$ is directed and of finite cardinality (modulo $\cong$). Thus there is some $a_0 \in A$ such that $\pi^n(a') \lesssim \pi^n(a_0)$ for all $a' \in A$. Hence $\{\pi^n(a)\} \cong_s \{\pi^n(\pi^n(a))\} \lesssim_s \pi^n(A) \lesssim_s \{\pi^n(a_0)\} \lesssim_s \{a_0\}$ and thus $\pi^n(a) \lesssim a_0$. □

LEMMA 4.23 (FINITE CHARACTERIZATION) $\mathbb{E}^\omega = \bigcup_{n\in\mathbb{N}} \mathbb{E}^n$

PROOF: We show d is finite iff $d \cong \pi^n(a)$ for some $n \in \mathbb{N}$. The backwards implication follows from Lemma 4.22. To prove the forward implication, pick $d \in \mathbb{E}^\omega$, and show $\pi^n(d) \cong d$, for some n. $d \lesssim_s \{\pi^n(d) \mid n \in \mathbb{N}\}$ by Theorem 4.19. Thus, by finiteness of d, $d \lesssim \pi^n(d)$ for some n. Furthermore, $\pi^n(d) \lesssim d$ by 4.16, so $\pi^n(d) \cong d$. □

We conclude by showing $\lesssim$ is ω-algebraic.

DEFINITION 4.24 (FINITE PROJECTION $\Pi(a)$) For $a \in \mathbb{E}_\emptyset$ define

$$\Pi(a) = \{d \in \mathbb{E}^\omega \mid d \lesssim a\}$$

LEMMA 4.25 ($\lesssim$ ω-ALGEBRAICITY) For $a \in \mathbb{E}_\emptyset$,

(i) $\Pi(a) \in \Delta_\emptyset$, and $\Pi(a) \cong_s \{a\}$

(ii) $\bigsqcup \Pi(a) = a$.

PROOF: (i) follows directly from Theorem 4.19 and Lemma 4.23. (ii) follows from (i) and Lemma 4.4. □

4.5 The Existence of $\lesssim_s$-Least Upper Bounds

In this section we show that $\lesssim_s$-directed sets have least upper bounds. As for finite expressions, we restrict our attention to closed expressions. Note that $A \lesssim_s B$ does not imply that $A \cup B$ is a $\lesssim$-directed set. Consider the following simple example.

$$f_k(x) = \begin{cases} 1 & \text{if } x \notin \mathbb{N} \text{ or } x < k \text{ or } x \text{ is even.} \\ \texttt{bot} & \text{otherwise.} \end{cases}$$

$$g_k(x) = \begin{cases} 1 & \text{if } x \notin \mathbb{N} \text{ or } x < k \text{ or } x \text{ is odd.} \\ \texttt{bot} & \text{otherwise.} \end{cases}$$

Then

LEMMA 4.26 $\{g_k \mid k \in \mathbb{N}\} \cong_s \{f_k \mid k \in \mathbb{N}\} \cong_s \{\lambda x.1\}$ but $\{g_k \mid k \in \mathbb{N}\} \cup \{f_k \mid k \in \mathbb{N}\}$ is not $\lesssim$-directed.

Consequently the lub operation on $\lesssim_s$-directed sets cannot be a simple union operation. To construct lubs we use the theory of finite expressions developed in the previous section, lifting those results to consider finite approximants of directed *sets* of expressions instead of finite approximants to a single expression.

We begin by making some observations that enable us to restrict our attention to $\lesssim$-directed subsets of $\mathbb{E}^\omega$.

DEFINITION 4.27 (SET FINITE PROJECTION $\Pi(A)$) $\Pi(A) = \bigcup_{a \in A} \Pi(a)$.

LEMMA 4.28 (ELEMENTARY Π PROPERTIES) For $A \in \Delta_\emptyset$,

(i) $\Pi(A) \in \Delta_\emptyset$,

(ii) $\Pi(A) = \{d \in \mathbb{E}^\omega \mid (\exists a \in A)(d \sqsubseteq a)\}$

(iii) $\Pi(A)$ is $\cong$-closed

(iv) $\Pi(A) \cong_s A$

PROOF: (i) and (iv) follow immediately from Lemma 4.25, and the remainder are direct from the definitions. □

DEFINITION 4.29 (DIRECTED SETS $\Delta_\emptyset$) $\Delta_\emptyset$ is the set of $\sqsubseteq_s$-directed subsets of $\Delta_\emptyset$.

DEFINITION 4.30 ($\sqsubseteq_s$-LEAST UPPER BOUNDS) For $S \in \Delta_\emptyset$, $\bigsqcup S = A$ for some $A \in \Delta$ iff A is a $\sqsubseteq_s$-least upper bound of S.

LEMMA 4.31 (EXISTENCE $\sqsubseteq_s$-LEAST UPPER BOUNDS) If $S \in \Delta_\emptyset$, then its least upper bound exists and equals:

$$\bigsqcup S = \bigcup_{A \in S} \Pi(A) = \{d \in \mathbb{E}^\omega \mid (\exists A \in S)(\exists a \in A)(d \sqsubseteq a)\}$$

PROOF: First, we show that $\bigcup_{A \in S} \Pi(A)$ is directed. Let $a_0, a_1 \in \bigcup_{A \in S} \Pi(A)$. We must find an upper bound of these two points. By definition of a_0 and a_1, that means $a_i \in \Pi(A_i)$ for some $A_i \in S$, $i < 2$. $\{a_i\} \sqsubseteq_s \Pi(A')$, $i < 2$, for $A' \in S$ by $\sqsubseteq_s$-directedness of S. Thus, by the definition of finiteness, there are a'_i for $a'_i \in \Pi(A')$, $i < 2$, such that $a_i \sqsubseteq a'_i$, $i < 2$. Since $\Pi(A')$ is directed, there is an $a_2 \in \Pi(A')$ such that $a'_i \sqsubseteq a_2$, $i < 2$. Thus, $a_i \sqsubseteq a_2$, the upper bound we sought.

To prove $\bigcup_{A \in S} \Pi(A)$ is an upper bound, let $A \in S$. We must show $A \sqsubseteq_s \bigcup_{A \in S} \Pi(A)$. Since $A \cong_s \Pi(A)$, the result is trivial by set inclusion. To prove $\bigcup_{A \in S} \Pi(A)$ is least, let A_0 be such that $A \sqsubseteq_s A_0$ for each $A \in S$. We must show $\bigcup_{A \in S} \Pi(A) \sqsubseteq_s A_0$. For this, let $a \in \bigcup_{A \in S} \Pi(A)$, and thus $a \in A$ for some $A \in S$. $\{a\} \sqsubseteq_s A_0$ by definition of $\sqsubseteq_s$, so the result follows directly. □

4.6 $\sqsubseteq_s$-Continuity and ω-Algebraicity.

In Section 4.5, we showed that $<\Delta_\emptyset, \sqsubseteq_s>$ is a complete pre-order, and thus the quotiented $<\Delta_\emptyset/\cong_s, \sqsubseteq_s>$ is a complete partial order. In this section, we show it is also ω-algebraic and that the $\mathbb{O}^+$-induced operations are continuous. We begin by showing that there is a natural choice of $\cong_s$ equivalence class representative, the $\sqsubseteq$-downward-closed directed sets of finite elements $\Pi(A)$. ω-algebraicity and continuity will be proved for this representation.

LEMMA 4.32 (FINITE SET REPRESENTATION) For $A_0, A_1 \in \Delta_\emptyset$,

$$A_0 \sqsubseteq_s A_1 \Leftrightarrow \Pi(A_0) \subseteq \Pi(A_1)$$
$$A_0 \cong_s A_1 \Leftrightarrow \Pi(A_0) = \Pi(A_1)$$

The elements of the CPO, $\Delta^\omega_\emptyset$, are the $\Pi(A)$.

DEFINITION 4.33 (FINITE DIRECTED SETS $\Delta^\omega_\emptyset$) $\Delta^\omega_\emptyset = \{\Pi(A) \mid A \in \Delta_\emptyset\}$

We let D range over $\Delta^\omega_\emptyset$. Note that $\Delta^\omega_\emptyset$ is the set of $\lesssim$-directed subsets of $\mathbb{E}^\omega$ that are downward closed (order ideals).

THEOREM 4.34 (CPO) $<\Delta^\omega_\emptyset, \subseteq>$ is a Complete Partial Order, with $\bigsqcup S = \bigcup S$.

PROOF: $<\Delta_\emptyset, \subseteq>$ is a CPO directly from Lemmas 4.31 and 4.32. $\bigsqcup S$ for $\subseteq$-directed $S \subseteq \Delta^\omega_\emptyset$ is precisely $\cup S$ by Lemma 4.31 above and the observation that $\Pi(\Pi(A)) = \Pi(A)$. □

A corollary of Lemma 4.32 is that $<\Delta_\emptyset/\cong_s, \lesssim_s>$ and $<\Delta^\omega_\emptyset, \subseteq>$ are isomorphic structures.

LEMMA 4.35 (FINITE ELEMENTS) $\{\Pi(d) \mid d \in \mathbb{E}^\omega\}$ are the finite elements of CPO $<\Delta^\omega_\emptyset, \subseteq>$.

PROOF: We show that $\Pi(d) \subseteq \cup D$ implies $\Pi(d) \subseteq \Pi(A)$ for some $\Pi(A) \in D$. Since the Π operation produces $\lesssim$-downward closed sets, it suffices to show $d \in \cup D$ implies $d \in \Pi(A)$ for some $\Pi(A) \in D$, and this is immediate. □

Now we define the $\mathbb{O}^+$-induced operations on $\Delta^\omega_\emptyset$.

DEFINITION 4.36 (INDUCED OPERATIONS) For $\mathtt{op}^+ \in \mathbb{O}^+$ we define $\mathtt{op}^+ : (\Delta^\omega_\emptyset)^n \to \Delta^\omega_\emptyset$ as follows:

$$\mathtt{op}^+(D_1, \ldots, D_n) = \{d \in \mathbb{E}^\omega \mid (\exists d_1 \in D_1, \ldots, d_n \in D_n)(d \lesssim \mathtt{op}^+(d_1, \ldots, d_n))\}.$$

Then we have the following:

THEOREM 4.37 (CONTINUITY) $<\Delta^\omega_\emptyset, \subseteq>$ is continuous: all $\mathtt{op}^+ \in \mathbb{O}^+$ are continuous in each argument.

PROOF: For simplicity consider a unary operator $\mathtt{op}^+ \in \mathbb{O}^+_1$, the general case is similar. It suffices to show $\bigcup_{D \in S} \mathtt{op}^+(D) = \mathtt{op}^+(\bigcup S)$, i.e.

$$\bigcup_{D \in S} \{d_0 \mid (\exists d \in D)(d_0 \lesssim \mathtt{op}^+(d))\} = \{d_0 \mid (\exists d \in \bigcup S)(d_0 \lesssim \mathtt{op}^+(d))\}.$$

These are clearly the same sets. □

Note, in particular that application, $\mathtt{app} \in \mathbb{O}_2$, is continuous in both its arguments.

THEOREM 4.38 (ω-ALGEBRAIC) $<\Delta^\omega_\emptyset, \subseteq>$ is ω-Algebraic.

PROOF: Note that by Lemma 4.35 there are only countably many finite elements, and for $D \in \Delta^\omega_\emptyset$,

$$\bigsqcup\{\Pi(d) \mid \Pi(d) \subseteq D\} = \bigcup\{\Pi(d) \mid \Pi(d) \subseteq D\} = D.$$

□

The CPO construction of this section relies on properties of finite expressions; it is also possible to derive a continuous CPO from a $\lesssim_s$ ordering by a simpler ideal completion construction [Smith, 1992]. The construction in this section has the advantage that we can show that the ordering $\lesssim_s$ itself is a complete pre-order, and is ω-algebraic.

5 Constructing and Characterizing Models

In this section we study the general notion of a model for a functional call-by-value programming language with numbers and pairing. Our approach builds on the work of Milner [Milner, 1977] and Meyer [Meyer, 1982]. We begin by defining the notion of an **FLD** domain (functional programming language domain). These are reflexive domains with an extensional partial ordering $\sqsubseteq$ reflecting degrees of definedness. Next we define a notion of **FLEM** (functional language environment model) for interpreting expressions in an **FLD** domain. We classify these models according to what properties they possess. The classifications are adequacy (**AD**), full abstraction (**FA**), strong full abstraction (**SFA**), completeness (**CPO**), continuity (**CON**), ω-algebraic (**ALG**), least fixed-point (**LFP**), and standard (**STD**). Strong full abstraction is a strengthening of full abstraction, discussed in more detail below. A standard model is the well-known notion from logic, here meaning all points in the model correspond to computations.

We construct a standard model using $\sqsubseteq_{\simeq}$, and show that this model is fully abstract but not continuous, using the results of section 3.4. We then show that all standard, fully abstract models are isomorphic, and thus no such model is continuous. We then construct a model, using $\sqsubseteq_{\simeq_s}$, that is strongly fully abstract, continuous, and ω-algebraic, using the results of section 4.6. Following Milner [Milner, 1977] we show all such models are isomorphic.

5.1 The Notion of Model

We use the usual lifting operator, $D_\perp = D \cup \{\perp\}$, adding distinguished element $\perp$ to arbitrary set D. $D_\perp \stackrel{\text{strict}}{\longrightarrow} D'_\perp$ denotes the space of strict functions from $D_\perp$ to $D'_\perp$, i.e. functions $\phi \in (D_\perp \to D'_\perp)$ where $\phi(\perp) = \perp$. Define $\mathit{lift}(\phi)$ to take $\phi \in (D \to D'_\perp)$ and lift it to the strict $D_\perp \stackrel{\text{strict}}{\longrightarrow} D'_\perp$. If $\phi \in (D_\perp \times \ldots \times D_\perp) \to D_\perp$ we say it is strict if it is strict in each argument. A general notion of domain is now defined.

DEFINITION 5.1 (**FLD** DOMAIN) An **FLD** domain is a structure

$$\mathcal{S} = (\mathcal{D}, \mathcal{N}, \mathcal{P}, \mathcal{L}, \mathcal{F}, \Phi, \boldsymbol{\Pi}, \boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2, +_{\mathcal{N}}, -_{\mathcal{N}}, \iota, \sqsubseteq)$$

where

$$\begin{array}{l}
\mathcal{D} = \mathcal{N} + \mathcal{P} + \mathcal{L}, \text{ a disjoint sum} \\
\iota \in \mathbb{N} \stackrel{\text{bijection}}{\longrightarrow} \mathcal{N}, \\
+_{\mathcal{N}}, -_{\mathcal{N}} : \mathcal{N}^2 \to \mathcal{N}, \\
\mathcal{F} \subseteq \mathcal{D}_\perp \stackrel{\text{strict}}{\longrightarrow} \mathcal{D}_\perp, \\
\Phi \in \mathcal{L} \stackrel{\text{bijection}}{\longrightarrow} \mathcal{F}, \\
\boldsymbol{\Pi} \in \mathcal{D}_\perp \times \mathcal{D}_\perp \stackrel{\text{strict}}{\longrightarrow} \mathcal{P}_\perp, \text{ also satisfying } \boldsymbol{\Pi} \in \mathcal{D} \times \mathcal{D} \to \mathcal{P}, \\
\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2 \in \mathcal{P}_\perp \stackrel{\text{strict}}{\longrightarrow} \mathcal{D}_\perp, \text{ also satisfying } \boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2 \in \mathcal{P} \to \mathcal{D}
\end{array}$$

such that

(i) $\iota : \langle \mathbb{N}, +, - \rangle \cong \langle \mathcal{N}, +_{\mathcal{N}}, -_{\mathcal{N}} \rangle$ as standard first order structures.

(ii) $\boldsymbol{\Pi}_1(\boldsymbol{\Pi}(\delta, \delta')) = \delta, \quad \boldsymbol{\Pi}_2(\boldsymbol{\Pi}(\delta, \delta')) = \delta'$ for $\delta, \delta' \in \mathcal{D}$,

(iii) $\boldsymbol{\Pi}(\boldsymbol{\Pi}_1(p), \boldsymbol{\Pi}_2(p)) = p$ for $p \in \mathcal{P}$

and $\sqsubseteq \subseteq \mathcal{D}_\perp \times \mathcal{D}_\perp$ has the following properties:

1. $\sqsubseteq$ is a partial order (transitive, reflexive, anti-symmetric), with $<\mathcal{N}_\perp, \sqsubseteq>$ being flat.
2. $\perp \sqsubseteq \delta$ for all $\delta \in \mathcal{D}$,
3. monotone: for every $\phi \in \mathcal{F} \cup \{\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2\}$, $\delta_0 \sqsubseteq \delta_1$ implies $\phi(\delta_0) \sqsubseteq \phi(\delta_1)$; similarly $\boldsymbol{\Pi}$ is monotone in each argument.
4. extensional: For $\phi \in \mathcal{F}$, $\phi(\delta) \sqsubseteq \phi'(\delta)$ for all $\delta \in \mathcal{D}_\perp$ iff $\Phi^{-1}(\phi) \sqsubseteq \Phi^{-1}(\phi')$.

We let δ range over $\mathcal{D}$; it will be clear from context what particular set $\mathcal{D}$ is. Let the collection of environments be $\mathbf{Env} = \mathbb{X} \to \mathcal{D}$. Now that the algebraic structure is defined we define the requirements an environment model must meet.

DEFINITION 5.2 (ENVIRONMENT MODELS (**FLEM**)) The set **FLEM** of functional language environment models consists of structures $\mathcal{M} = (\mathcal{S}, [\![\cdot]\!]\cdot)$ where $\mathcal{S}$ is a **FLD** domain, and $[\![\cdot]\!]\cdot \in \mathbb{E} \times \mathbf{Env} \to \mathcal{D}_\perp$ satisfies the following:

(i) $\mathit{lift}(\boldsymbol{\lambda}\delta : \mathcal{D}.[\![a]\!](\rho\{x := \delta\})) \in \mathcal{F}$ for $\rho \in \mathbf{Env}$, $a \in \mathbb{E}$,

(ii) $[\![x]\!]\rho = \rho(x)$ for $x \in \mathbb{X}$,

(iii) $[\![n]\!]\rho = \iota(n)$ for $n \in \mathbb{N}$,

(iv) $[\![\mathtt{app}(a,b)]\!]\rho = (\Phi([\![a]\!]\rho))([\![b]\!]\rho)$ if $[\![a]\!]\rho \in \mathcal{L}$, else $\perp$,

(v) $[\![\lambda x.a]\!]\rho = \Phi^{-1}(\phi)$ where $\phi = \mathit{lift}(\boldsymbol{\lambda}\delta : \mathcal{D}.[\![a]\!](\rho\{x := \delta\}))$,

(vi) $[\![\mathtt{pr}(a,b)]\!]\rho = \boldsymbol{\Pi}([\![a]\!]\rho, [\![b]\!]\rho)$

(vii) $[\![\mathtt{fst}(a)]\!]\rho = \boldsymbol{\Pi}_1([\![a]\!]\rho)$ if $[\![a]\!]\rho \in \mathcal{P}$, else $\perp$,

(viii) $[\![\mathtt{snd}(a)]\!]\rho = \boldsymbol{\Pi}_2([\![a]\!]\rho)$ if $[\![a]\!]\rho \in \mathcal{P}$, else $\perp$,

(ix) $[\![\mathtt{ispr}(a)]\!]\rho = 1$ if $[\![a]\!]\rho \in \mathcal{P}$, 0 if $[\![a]\!]\rho \in \mathcal{L} \cup \mathcal{N}$, otherwise $\perp$,

(x) $[\![\mathtt{isnat}(a)]\!]\rho = 1$ if $[\![a]\!]\rho \in \mathcal{N}$, 0 if $[\![a]\!]\rho \in \mathcal{L} \cup \mathcal{P}$, otherwise $\perp$,

(xi) $[\![\mathtt{pred}(a)]\!]\rho = n -_{\mathcal{N}} \iota(1)$ if $[\![a]\!]\rho = n$ for $n \in \mathcal{N}$, otherwise $\perp$,

(xii) $[\![\mathtt{succ}(a)]\!]\rho = n +_{\mathcal{N}} \iota(1)$ if $[\![a]\!]\rho = n$ for $n \in \mathcal{N}$, otherwise $\perp$,

(xiii) $[\![\mathtt{br}(a,b,c)]\!]\rho = [\![b]\!]\rho$ if $[\![a]\!]\rho = 1$, $[\![c]\!]\rho$ if $1 \neq [\![a]\!]\rho \neq \perp$, otherwise $\perp$,

Note that the first condition is simply a closure condition on the set $\mathcal{F}$. Modulo this closure condition the nature of $[\![\cdot]\!]\cdot$ is completely determined by the structure of the underlying **FLD** domain $\mathcal{S}$ as we shall show in Lemma 5.20. For closed a, $[\![a]\!]$ abbreviates $[\![a]\!]\,\emptyset$.

LEMMA 5.3 (SUBSTITUTION) $[\![a]\!]\rho\{x := [\![v]\!]\rho\} = [\![a\{x := v\}]\!]\rho$.

PROOF: By induction on the structure of a. □

LEMMA 5.4 $[\![\cdot]\!]\cdot$ respects computation: $a \mapsto b \Rightarrow [\![a]\!] = [\![b]\!]$.

PROOF: Direct from properties (ii)–(xiv) of Definition 5.2 and Lemma 5.3. □

We let $\mathcal{M}$ range over **FLEM** models and $D \subseteq \mathcal{D}$. If $\mathcal{M} \in \mathbf{FLEM}$, then we let $\mathcal{D}_{\text{def}}$ be the set of definable elements of $\mathcal{D}$, those elements that interpret some closed expression. $\mathcal{F}_{\text{def}} \subseteq \mathcal{F}$ are the definable functions.

DEFINITION 5.5 ($\mathcal{D}_{\text{def}}$ $\mathcal{F}_{\text{def}}$ DEFINABLE ELEMENTS AND FUNCTIONS)

$$\mathcal{D}_{\text{def}} = \{\delta \mid (\exists a \in \mathbb{E}_\emptyset)(\llbracket a \rrbracket = \delta)\}$$
$$\mathcal{F}_{\text{def}} = \{\phi \mid (\exists \delta \in \mathcal{D}_{\text{def}})(\Phi(\delta) = \phi)\} = \{\phi \mid (\exists a \in \mathbb{L}_\emptyset)(\Phi(\llbracket a \rrbracket) = \phi)\}$$

DEFINITION 5.6 (FINITE) $\bigsqcup D$ denotes the $\sqsubseteq$-least upper bound of directed D, if it exists. We say $\delta \in \mathcal{D}$ is *finite* if for any directed $D \subseteq \mathcal{D}$ such that $\delta \sqsubseteq \bigsqcup D$, there is some $\delta' \in D$ such that $\delta \sqsubseteq \delta'$.

5.2 Classification of Models

In this section we define some important properties of **FLEM** models, and establish some relationships between these properties.

DEFINITION 5.7 (**FLEM** MODEL CLASSIFICATIONS) Given an $\mathcal{M} \in \mathbf{FLEM}$, define

ADEQUACY: $\mathcal{M} \in \mathbf{AD}$ iff for all closed a ($a \mapsto v$ for some v iff $\llbracket a \rrbracket \neq \bot$)

FULL ABSTRACTION: $\mathcal{M} \in \mathbf{FA}$ iff $\mathcal{M} \in \mathbf{AD}$ and for all $\delta_0, \delta_1 \in \mathcal{D}_{\text{def}}$, $\delta_0 \sqsubseteq \delta_1$ iff for all $\phi \in \mathcal{F}_{\text{def}}$, $\phi(\delta_0) \neq \bot$ implies $\phi(\delta_1) \neq \bot$.

STRONG FULL ABSTRACTION: $\mathcal{M} \in \mathbf{SFA}$ iff $\mathcal{M} \in \mathbf{AD}$ and for all $\delta_0, \delta_1 \in \mathcal{D}$, $\delta_0 \sqsubseteq \delta_1$ iff for all $\phi \in \mathcal{F}_{\text{def}}$, $\phi(\delta_0) \neq \bot$ implies $\phi(\delta_1) \neq \bot$.

COMPLETE: $\mathcal{M} \in \mathbf{CPO}$ iff $\mathcal{M} \in \mathbf{AD}$ and all $\sqsubseteq$-directed sets $D \subseteq \mathcal{D}_\bot$ have a lub, $\bigsqcup D$.

CONTINUOUS: $\mathcal{M} \in \mathbf{CON}$ iff $\mathcal{M} \in \mathbf{CPO}$ and for all $\phi \in \mathcal{F}$ and directed sets $D \subseteq \mathcal{D}_\bot$, $\phi(\bigsqcup D) = \bigsqcup\{\phi(\delta) \mid \delta \in D\}$.

ω-ALGEBRAIC: $\mathcal{M} \in \mathbf{ALG}$ if $\mathcal{D}$ has countably many finite elements, and for each $\delta \in \mathcal{D}$, letting $D = \{\delta_0 \mid \delta_0 \text{ is finite and } \delta_0 \sqsubseteq \delta\}$, D is directed and $\bigsqcup D = \delta$.

LEAST FIXED-POINT: $\mathcal{M} \in \mathbf{LFP}$ if $\mathcal{M} \in \mathbf{CPO}$ and

(i) Defining $\mathit{fix}(\phi) = \bigsqcup\{\overbrace{\phi(\ldots\phi}^{n}(\boldsymbol{\lambda}x.\ \bot)\ldots) \mid n \in \mathbb{N}\}$ where $\boldsymbol{\lambda}x.\ \bot$ is the everywhere $\bot$ function, $\mathit{fix}(\phi) = \phi(\mathit{fix}(\phi))$ for all $\phi \in \mathcal{F}$.

(ii) $\Phi(\llbracket \texttt{fix} \rrbracket)(\phi) = \mathit{fix}(\phi)$, for all $\phi \in \mathcal{F}$.

STANDARD: $\mathcal{M} \in \mathbf{STD}$ if $\mathcal{M} \in \mathbf{AD}$ and $\mathcal{D}_{\text{def}} = \mathcal{D}$, i.e. if $\llbracket \cdot \rrbracket\emptyset$ is a surjection.

Some trivial consequences of the definitions are summed up in the following lemma.

LEMMA 5.8

(i) $\mathbf{SFA} \subseteq \mathbf{FA} \subseteq \mathbf{AD}$.

(ii) $\mathbf{CON} \subseteq \mathbf{CPO} \subseteq \mathbf{AD}$.

PROOF: Trivial by inspection of the definitions. □

Full abstraction above is defined over functions in the domain; the following Lemma demonstrates this is equivalent to the standard definition of full abstraction.

LEMMA 5.9 $\mathcal{M} \in \mathbf{FA} \Leftrightarrow (\mathcal{M} \in \mathbf{AD} \wedge (\forall a_0, a_1 \in \mathbb{E}_\emptyset)(a_0 \sqsubseteq\!\!\!\sim a_1 \Leftrightarrow [\![a_0]\!] \sqsubseteq [\![a_1]\!]))$.

PROOF:

$$a_0 \sqsubseteq\!\!\!\sim a_1 \Leftrightarrow (\forall C \in \mathbb{C}_\emptyset)(C[a_0]\!\downarrow \Rightarrow C[a_1]\!\downarrow) \Leftrightarrow (\forall C)([\![C[a_0]]\!] \neq \bot \Rightarrow [\![C[a_1]]\!] \neq \bot)$$

by 3.4 and adequacy, respectively. Define $\phi = \Phi[\![\lambda x.C[x]]\!]$ and observe $\phi([\![a_i]\!]) = [\![C[x]\{x := [\![a_i]\!]\}]\!] = [\![C[a_i]]\!]$, $i < 2$. Thus,

$$(\forall C)([\![C[a_0]]\!] \neq \bot \Rightarrow [\![C[a_1]]\!] \neq \bot) \Leftrightarrow (\forall \phi \in \mathcal{F}_{\text{def}})(\phi([\![a_0]\!]) \neq \bot \Rightarrow \phi([\![a_1]\!]) \neq \bot) \Leftrightarrow [\![a_0]\!] \sqsubseteq [\![a_1]\!]$$

by definition of $\phi \in \mathcal{F}_{\text{def}}$, and assumption $\mathcal{M} \in \mathbf{FA}$, respectively. □

Strong full abstraction is an extension of full abstraction to require $\delta_0 \sqsubseteq \delta_1$ on non-definable elements δ_0, δ_1 to mean δ_0 and δ_1 are indistinguishable by any definable functions $\phi \in \mathcal{F}_{\text{def}}$. The analogous result to 5.9 in the case of strong full abstraction is the following lemma.

DEFINITION 5.10 Working over $\mathcal{M} \in \mathbf{CPO}$, we may define $[\![A]\!] = \bigsqcup\{[\![a]\!] \mid a \in A\}$ for $A \in \Delta_\emptyset$.

LEMMA 5.11 (SET FULL ABSTRACTION) If $\mathcal{M} \in \mathbf{SFA} \cap \mathbf{CON}$, then for any $A, B \in \Delta_\emptyset$

$$A \sqsubseteq\!\!\!\sim_{\text{s}} B \Leftrightarrow [\![A]\!] \sqsubseteq [\![B]\!].$$

PROOF: For the forward implication, we begin by assuming that $[\![A]\!] \not\sqsubseteq [\![B]\!]$. Thus by **SFA** there exists an $\phi \in \mathcal{F}_{\text{def}}$ such that $\phi([\![A]\!]) \neq \bot$ while $\phi([\![B]\!]) = \bot$. Now $\phi([\![B]\!]) = \phi(\bigsqcup\{[\![b]\!] \mid b \in B\}) = \bigsqcup\{\phi([\![b]\!]) \mid b \in B\}) = \bot$ by continuity. Thus $\phi([\![b]\!]) = \bot$ for all $b \in B$. Similarly since $\phi([\![A]\!]) \neq \bot$ we have that there is an $a \in A$ with $\phi([\![a]\!]) \neq \bot$. Thus $A \not\sqsubseteq\!\!\!\sim_{\text{s}} B$.

For the reverse implication, assume that $[\![A]\!] \sqsubseteq [\![B]\!]$ and choose $a \in A$, $\phi \in \mathcal{F}_{\text{def}}$ such that $\phi([\![a]\!]) \neq \bot$. Thus $\phi([\![A]\!]) \neq \bot$, and consequently $\phi([\![B]\!]) \neq \bot$. This last fact also implies $\phi([\![b]\!]) \neq \bot$ for some $b \in B$. □

In [Milner, 1977] a straightforward induction argument establishes that, in the simply typed lambda calculus, a fully abstract, continuous model with ω-algebraic base types that contains the finite projections is also ω-algebraic. In the untyped framework, whether or not an analogous result remains true is an open question. In particular it is open whether all continuous fully abstract models are ω-algebraic. The difference between the simply typed and untyped cases is in a simply typed language a finite number of applications or projections will always produce an expression of base type, but some untyped computations (such as $\texttt{fix}(\lambda x.\lambda y.x)$) have no such property. Similar problems will arise in typed languages with recursive types. Thus there could in principle be an element of the model of this infinitary form which is not expressible as the lub of a collection of finite elements. Strong full abstraction is introduced to close this gap in the untyped case; we show the following.

THEOREM 5.12 $\mathbf{SFA} \cap \mathbf{CON} \subseteq \mathbf{ALG}$.

To prove this theorem we first establish a series of four simple lemmas.

LEMMA 5.13 For $\mathcal{M} \in \mathbf{SFA} \cap \mathbf{CON}$, $[\![\lambda x.x]\!] = \bigsqcup\{[\![\pi^n]\!] \mid n \in \mathbb{N}\}$.

PROOF: By Lemma 5.11 and Theorem 4.19. □

DEFINITION 5.14 (SEMANTIC PROJECTION) Define $\boldsymbol{\pi}^n = \Phi([\![\pi^n]\!]) \in \mathcal{F}$, the semantic meaning of the projection function.

LEMMA 5.15 For $\mathcal{M} \in \mathbf{FLEM}$, $\boldsymbol{\pi}^k(\delta) \sqsubseteq \delta$ for all k.

PROOF: The proof is by induction on k. The base case is trivial. We proceed by cases on the form of δ. If $\delta \in \mathcal{N}$, then the result is also trivial if $\delta = 0$; otherwise, computing π^k by Lemma 5.4 we derive $\boldsymbol{\pi}^k(\delta) = \boldsymbol{\pi}^{k-1}(\delta-1)+1$, and applying the induction hypothesis, $\boldsymbol{\pi}^{k-1}(\delta-1)+1 \sqsubseteq \delta-1+1 = \delta$, completing this case. If $\delta \in \mathcal{P}$, then by computation we derive $\boldsymbol{\pi}^k(\delta) = \boldsymbol{\Pi}(\boldsymbol{\pi}^{k-1}(\boldsymbol{\Pi}_1(\delta)), \boldsymbol{\pi}^{k-1}(\boldsymbol{\Pi}_2(\delta)))$. Applying the induction hypothesis yields $\boldsymbol{\Pi}(\boldsymbol{\pi}^{k-1}(\boldsymbol{\Pi}_1(\delta)), \boldsymbol{\pi}^{k-1}(\boldsymbol{\Pi}_2(\delta))) \sqsubseteq \boldsymbol{\Pi}(\boldsymbol{\Pi}_1(\delta), \boldsymbol{\Pi}_2(\delta))$, and by the definitions $\boldsymbol{\Pi}(\boldsymbol{\Pi}_1(\delta), \boldsymbol{\Pi}_2(\delta)) = \delta$, completing this case. If $\delta \in \mathcal{L}$, then by computing we derive $\Phi(\boldsymbol{\pi}^k(\delta)) = \boldsymbol{\lambda}\delta_0.\boldsymbol{\pi}^{k-1}(\Phi(\delta)(\boldsymbol{\pi}^{k-1}(\delta_0))) \in \mathcal{F}$. Using $\sqsubseteq$ extensionality, it suffices to show $\boldsymbol{\pi}^{k-1}(\Phi(\delta)(\boldsymbol{\pi}^{k-1}(\delta_0))) \sqsubseteq \Phi(\delta)(\delta_0)$ for arbitrary δ_0, and this follows directly from the induction hypothesis. □

We now characterize the finite elements of fully abstract, continuous models. The next two Lemmas are domain analogues of Lemmas 4.21 and 4.23 on directed sets of expressions, respectively.

DEFINITION 5.16 $\mathcal{D}^k = \{\delta \in \mathcal{D} \mid \delta = \boldsymbol{\pi}^k(\delta)\}$, $\mathcal{D}^\omega = \cup_{k\in\mathbb{N}}\mathcal{D}^k$.

LEMMA 5.17 For $\mathcal{M} \in \mathbf{FA} \cap \mathbf{CON}$, $\mathcal{D}^k$ is of finite cardinality for each $k \in \mathbb{N}$, and $\mathcal{D}^\omega$ is countable.

PROOF: The countability of $\mathcal{D}^\omega$ is direct from the finiteness of the $\mathcal{D}^k$. The latter is proved by induction on k. The base case is trivial, $\mathcal{D}^0$ is empty by observing $[\![\texttt{bot}]\!] = \bot$ from the fact that $\mathcal{M} \in \mathbf{AD}$. For the induction case, it suffices to prove $\mathcal{D}^k \cap \mathcal{L}$, $\mathcal{D}^k \cap \mathcal{N}$, $\mathcal{D}^k \cap \mathcal{P}$ are each finite sets.

For $\mathcal{D}^k \cap \mathcal{N}$ observe that $\{\delta \mid \delta = \boldsymbol{\pi}^k(\delta) \wedge \delta \in \mathcal{N}\} = \{\delta \mid \delta = 0 \vee \delta = \boldsymbol{\pi}^{k-1}(\delta-1)+1\}$ which is finite by the finiteness of $\{\delta \mid \delta = \boldsymbol{\pi}^{k-1}(\delta)\}$ by inductive assumption. Similarly, $\mathcal{D}^k \cap \mathcal{P}$ is seen to be finite.

For $\mathcal{D}^k \cap \mathcal{L}$ observe that $\{\delta \mid \delta = \boldsymbol{\pi}^k(\delta) \wedge \delta \in \mathcal{L}\} = \{\delta \mid \Phi(\delta) = \boldsymbol{\lambda}\delta_0.\boldsymbol{\pi}^{k-1}(\Phi(\delta)(\boldsymbol{\pi}^{k-1}(\delta_0))) \in \mathcal{F}\}$. By inductive assumption each of these functions is restricted to a finite domain and codomain, so there can only be finitely many such functions. □

LEMMA 5.18 For $\mathcal{M} \in \mathbf{SFA} \cap \mathbf{CON}$, δ finite iff $\delta \in \mathcal{D}^\omega$.

PROOF: For the forward implication, $\bigsqcup\{\boldsymbol{\pi}^k(\delta) \mid n \in \mathbb{N}\} = \delta$ by Lemma 5.13 and continuity, so by the finiteness of δ, $\delta \sqsubseteq \boldsymbol{\pi}^k(\delta)$ for some k. Since $\boldsymbol{\pi}^k(\delta) \sqsubseteq \delta$ by Lemma 5.15, $\boldsymbol{\pi}^k(\delta) = \delta$.

For the reverse implication observe that by Lemma 5.17, each $\mathcal{D}^k$ set is of finite cardinality, thus no infinite ascending chains may be defined in $\mathcal{D}^k$, so all its elements must be finite. □

PROOF OF 5.12: There are countably many finite elements by Lemma 5.17. Next, given δ, set $D = \{\delta_0 \mid \delta_0 \sqsubseteq \delta \wedge \delta_0 \text{ finite }\}$. We show $\bigsqcup D = \delta$. $\bigsqcup D \sqsubseteq \delta$ follows pointwise so it suffices to show $\bigsqcup D \sqsupseteq \delta$.

$$\begin{aligned}
\bigsqcup D &= \bigsqcup\{\boldsymbol{\pi}^k(\delta_0) \mid \boldsymbol{\pi}^k(\delta_0) \sqsubseteq \delta\} && \text{by Lemma 5.18}\\
&\sqsupseteq \bigsqcup\{\boldsymbol{\pi}^k(\delta) \mid k \in \mathbb{N}\} && \text{by Lemma 5.15}\\
&= \delta && \text{by Lemma 5.13.}
\end{aligned}$$

□

5.3 Isomorphisms between Models

In this section we define the notion of an isomorphism between two models. This notion will play an important role in demonstrating that certain properties uniquely characterize the models which satisfy them.

DEFINITION 5.19 (ISOMORPHISM OVER **FLEM**) Given two elements of **FLEM**, $\mathcal{M}_i$ for $i < 2$,

$\mathcal{M}_i = (\mathcal{S}_i, [\![\cdot]\!]_i \cdot)$

where

$\mathcal{S} = (\mathcal{D}_i, \mathcal{N}_i, \mathcal{P}_i, \mathcal{L}_i, \mathcal{F}_i, \Phi_i, \boldsymbol{\Pi}^i, \boldsymbol{\Pi}^i_1, \boldsymbol{\Pi}^i_2, +^i_{\mathcal{N}}, -^i_{\mathcal{N}}, \iota_i, \sqsubseteq_i)$

we say they are isomorphic, $\mathcal{M}_0 \sim \mathcal{M}_1$, iff there exists a $\Sigma \in \mathcal{D}_0 \stackrel{\text{bijection}}{\longrightarrow} \mathcal{D}_1$ such that

(i) Σ maps $\mathcal{P}_0$ to $\mathcal{P}_1$, $\mathcal{L}_0$ to $\mathcal{L}_1$, and $\mathcal{N}_0$ to $\mathcal{N}_1$ commuting with ι_i.

(ii) Σ is order-preserving, $\delta_0 \sqsubseteq_0 \delta_1 \Leftrightarrow \Sigma(\delta_0) \sqsubseteq_1 \Sigma(\delta_1)$

(iii) $(\forall \delta_0 \in \mathcal{L}_0, \delta_1 \in \mathcal{D}_0)(\mathit{lift}(\Sigma)(\Phi_0(\delta_0)(\delta_1)) = \Phi_1(\Sigma(\delta_0))(\Sigma(\delta_1)))$

(iv) $(\forall \delta_0, \delta_1 \in \mathcal{D}_0)(\Sigma(\boldsymbol{\Pi}^0(\delta_0, \delta_1)) = \boldsymbol{\Pi}^1(\Sigma(\delta_0), \Sigma(\delta_1))$

(v) $(\forall \delta \in \mathcal{P}_0)\Sigma(\boldsymbol{\Pi}^0_1(\delta)) = \boldsymbol{\Pi}^1_1(\Sigma(\delta))$

(vi) $(\forall \delta \in \mathcal{P}_0)\Sigma(\boldsymbol{\Pi}^0_2(\delta)) = \boldsymbol{\Pi}^1_2(\Sigma(\delta))$

An alternate, but equivalent definition, is to define $\hat{\Sigma} = \boldsymbol{\lambda}\phi.\mathit{lift}(\Sigma) \circ \phi \circ \mathit{lift}(\Sigma^{-1})$, and require the following hold in place of (iii) above:

1. $\hat{\Sigma} : \mathcal{F}_0 \to \mathcal{F}_1$
2. $\Phi_0 = \hat{\Sigma}^{-1} \circ \Phi_1 \circ \Sigma$

Note that the definition of isomorphism does not mention $[\![\cdot]\!]_i \cdot$. This is because it is uniquely determined by the underlying structure. The following lemma validates this observation.

LEMMA 5.20 (EVALUATION ISOMORPHISM) Suppose that $\mathcal{M}_0 \sim \mathcal{M}_1$ via Σ. Then

$(\forall a \in \mathbb{E})(\forall \rho \in \mathbf{Env}_0 = \mathbb{X} \to \mathcal{D}_0)(\mathit{lift}(\Sigma)([\![a]\!]_0\rho) = [\![a]\!]_1(\Sigma \circ \rho))$

PROOF: The proof is by induction on the structure of a. We provide the interesting cases.
CASE $a = \mathtt{app}(b, c)$: For simplicity we shall assume that $[\![a]\!]_0\rho \neq \perp$.

$\mathit{lift}(\Sigma)([\![a]\!]_0\rho) =$
$= \mathit{lift}(\Sigma)((\Phi_0([\![b]\!]_0\rho))([\![c]\!]_0\rho))$
$= \Phi_1(\Sigma([\![b]\!]_0\rho))(\Sigma([\![c]\!]_0\rho)))$ by isomorphism property (iii)
$= (\Phi_1([\![b]\!]_1\Sigma \circ \rho))([\![c]\!]_1\Sigma \circ \rho)$ by induction hypothesis
$= [\![a]\!]_1\Sigma \circ \rho$

CASE $a = \lambda x.b$:

$$
\begin{aligned}
&\mathit{lift}(\Sigma)([\![a]\!]_0\rho) = [\![a]\!]_1\Sigma \circ \rho \\
&\quad \Leftrightarrow\ [\![a]\!]_0\rho = \mathit{lift}(\Sigma^{-1})([\![a]\!]_1\Sigma \circ \rho) \\
&\quad \Leftrightarrow\ (\forall \delta_0 \in \mathcal{D}_0)(\mathit{lift}(\Sigma)(\Phi_0([\![a]\!]_0\rho)(\delta_0)) = \Phi_0(\mathit{lift}(\Sigma^{-1})([\![a]\!]_1\Sigma \circ \rho))(\delta_0)) \qquad \text{since } \Phi_0 \text{ bijective} \\
&\quad \Leftrightarrow\ (\forall \delta_0 \in \mathcal{D}_0)(\mathit{lift}(\Sigma)(\Phi_0([\![a]\!]_0\rho)(\delta_0)) = \Phi_1([\![a]\!]_1\Sigma \circ \rho)(\Sigma(\delta_0))) \qquad \text{by isomorphism property (iii)} \\
&\quad \Leftrightarrow\ (\forall \delta_0 \in \mathcal{D}_0)(\mathit{lift}(\Sigma)([\![b]\!]_0\rho\{x := \delta_0\}) = \Phi_1([\![b]\!]_1(\Sigma \circ \rho)\{x := \Sigma(\delta_0)\})) \\
&\quad \Leftrightarrow\ (\forall \delta_0 \in \mathcal{D}_0)(\mathit{lift}(\Sigma)([\![b]\!]_0\rho\{x := \delta_0\}) = \Phi_1([\![b]\!]_1\Sigma \circ (\rho\{x := \delta_0\}))
\end{aligned}
$$

which follows by induction hypothesis. □

5.4 Existence and Uniqueness of Models

In this section we construct models with various combinations of properties. We also show that there are combinations of properties that have exactly one (up to isomorphism) model that satisfies them. We begin by constructing the canonical standard model, the term model.

DEFINITION 5.21 Define term model $\mathcal{M}^t$ as follows:

$$\mathcal{M}^t = (\mathcal{S}^t, [\![\cdot]\!]_i \cdot)$$

where

$$\mathcal{S} = (\mathcal{D}^t, \mathcal{N}^t, \mathcal{P}^t, \mathcal{L}^t, \mathcal{F}^t, \Phi^t, \boldsymbol{\Pi}^t, \boldsymbol{\Pi}^t_1, \boldsymbol{\Pi}^t_2, +^t_{\mathcal{N}}, -^t_{\mathcal{N}}, \iota^t, \sqsubseteq^t)$$

and

$$
\begin{aligned}
&[v] = \{v' \in \mathbb{V}_\emptyset \mid v \cong v'\} \\
&\mathcal{N}^t = \{[v] \mid v \in \mathbb{N}\} \\
&\mathcal{P}^t = \{[v] \mid v \in \mathbb{P}\} \\
&\mathcal{L}^t = \{[v] \mid v \in \mathbb{L}\} \\
&\mathcal{D}^t = \mathcal{N}^t + \mathcal{P}^t + \mathcal{L}^t \\
&[\![a]\!]^t\rho = \begin{cases} [v] & \text{if } a^\rho \mapsto v \text{ (where } a^{\{x:=[v]\}} = a^{\{x:=v\}}\text{)} \\ \bot & \text{otherwise} \end{cases} \\
&\Phi^t([\lambda x.a]) = \mathit{lift}(\boldsymbol{\lambda}[v] \in \mathcal{D}^t.[\![a]\!]^t\{x := [v]\}), \quad \mathcal{F}^t = \mathrm{Rng}(\Phi^t) \\
&\boldsymbol{\Pi}^t([v_1], [v_2]) = [\mathtt{pr}(v_1, v_2)] \\
&\boldsymbol{\Pi}^t_1([\mathtt{pr}(v_1, v_2)]) = [v_1] \\
&\boldsymbol{\Pi}^t_2([\mathtt{pr}(v_1, v_2)]) = [v_2] \\
&\sqsubseteq^t = \{\bot\} \times \mathcal{D}^t_\bot \cup \{\mathtt{<}[v_1], [v_2]\mathtt{>} \mid v_1 \sqsubseteq\!\!\!\sim\ v_2\}
\end{aligned}
$$

noting that Φ^t, $\boldsymbol{\Pi}^t$, $\boldsymbol{\Pi}^t_1$, and $\boldsymbol{\Pi}^t_2$ are in fact functions since any member of the equivalence class returns the same value.

One obvious property it is important not to forget is that the term model is indeed a standard model and is fully abstract.

LEMMA 5.22 (**STD**) $\mathcal{M}^t \in \mathbf{STD} \cap \mathbf{FA}$.

PROOF: First we show $\mathcal{M}^t \in \mathbf{FLEM}$. Φ^t is easily shown to be bijective: it is onto by definition, and into by $\cong$ extensionality. The properties of $\boldsymbol{\Pi}^t$, $\boldsymbol{\Pi}^t_1$, and $\boldsymbol{\Pi}^t_2$ are similarly direct. $\sqsubseteq^t$ is a partial order since $\lesssim$ is a pre-order and $\sqsubseteq^t$ is anti-symmetric by the quotienting operation. $\sqsubseteq$ Property 2. is direct from the definition, 3. follows by $\lesssim$ pre-congruence, and 4. by $\lesssim$ extensionality. $[\![\cdot]\!]^t\cdot$ is an environment model by a simple structural induction. Thus, $\mathcal{M}^t \in \mathbf{FLEM}$. $\mathcal{M}^t \in \mathbf{FA}$ is direct from Lemma 5.9. □

A standard model must have no extra points, so the only room for variance is to have alternate notions of $\sqsubseteq$. If we require $\sqsubseteq$ to be fully abstract we fix its value at all points so there is no room for variance and all models are then isomorphic.

THEOREM 5.23 All $\mathcal{M} \in \mathbf{FA} \cap \mathbf{STD}$ are isomorphic.

PROOF: It suffices to show for arbitrary $\mathcal{M} = (\mathcal{D}, \Phi, \boldsymbol{\Pi}, \boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2, \sqsubseteq, [\![\cdot]\!]\cdot) \in \mathbf{FA} \cap \mathbf{STD}$ that $\mathcal{M} \sim \mathcal{M}^t$. Define $\Sigma([v]) = [\![v]\!]$. Note this indeed defines a function since by full abstraction of $\mathcal{M}$ all $v_0 \cong v_1 \in [v]$ must map to the same point in $\mathcal{M}$.

First, Σ is a bijection: it is onto by the standardness of $\mathcal{M}$, and is into by the full abstraction of $\mathcal{M}$. We now proceed to establish isomorphism requirements (i)–(vi). For property (i), we show first that Σ maps $\mathcal{P}^t$ to $\mathcal{P}$. By the definitions of $\mathcal{P}^t$ and Σ it suffices to show $[\![\mathtt{pr}(v_0, v_1)]\!] \in \mathcal{P}$, and this is direct from the definitions. The cases for $\mathcal{N}$ and $\mathcal{L}$ are similar. For property (ii), the $\bot$ cases are direct and $[v_0] \sqsubseteq^t [v_1] \Leftrightarrow v_0 \lesssim v_1 \Leftrightarrow \Sigma([\![v_0]\!]) \sqsubseteq \Sigma([\![v_1]\!])$, the former by definition and the latter by $\mathcal{M}$ full abstraction. For property (iii), we show $\mathit{lift}(\Sigma)(\Phi^t([v_0])([v_1])) = \Phi([\![v_0]\!])([\![v_1]\!])$. Since $[v_0] \in \mathcal{L}^t$, $v_0 = \lambda x.a$. Proceed by cases on whether $v_0\,(v_1)\downarrow$. If not, $\mathit{lift}(\Sigma)(\Phi^t([v_0])([v_1])) = \bot$ by definition and $\Phi([\![v_0]\!])([\![v_1]\!]) = [\![v_0\,(v_1)]\!] = \bot$ by the definition of $[\![\cdot]\!]\cdot$ for applications and by adequacy of $\mathcal{M}$, respectively. Consider then the case $v_0\,(v_1)\downarrow$.

$$\mathit{lift}(\Sigma)(\Phi^t([v_0])([v_1])) = \Sigma([v_2]) = [\![a\{x := v_1\}]\!] = [\![a]\!]\{x := [\![v_1]\!]\} = \Phi([\![\lambda x.a]\!])([\![v_1]\!])$$

by the definition of Φ^t (where $v_0\,(v_1) \mapsto v_2$), Lemma 5.4, Lemma 5.3, and the definition of $[\![\cdot]\!]\cdot$, respectively. Properties (iv)-(vi) are direct from the definitions. □

Define $\mathbf{CON}^-$ to be **CON** but without the requirement $\mathbf{CON} \subseteq \mathbf{CPO}$.

DEFINITION 5.24 ($\mathbf{CON}^-$) $\mathcal{M} \in \mathbf{CON}^-$ iff for all $\phi \in \mathcal{F}$ and directed sets $D \subseteq \mathcal{D}_\bot$, if $\bigsqcup D$ is defined then $\phi(\bigsqcup D) = \bigsqcup\{\phi(\delta) \mid \delta \in D\}$.

We then may show no fully abstract standard model is continuous or complete.

COROLLARY 5.25 $\mathbf{FA} \cap \mathbf{STD} \cap \mathbf{CON}^- = \emptyset = \mathbf{FA} \cap \mathbf{STD} \cap \mathbf{CPO}$.

PROOF: These are immediate from Theorem 5.23 and Theorem 3.16, section 3.3. □

Now we construct a model, $\mathcal{M}^s$, that is in $\mathbf{SFA} \cap \mathbf{CON}$. The ultimate goal will be to show $\mathcal{M}^s$ is the unique continuous, strongly fully abstract model. Elements of $\mathcal{D}^s$ are $\cong_s$-equivalence classes of directed sets of expressions. Following the development of section 4.6, we pick a particular representative of the equivalence class to make proofs easier, the downward-closed sets of finite elements $\Delta^\omega_\emptyset$. Over this set, $\lesssim_s$ is not only a pre-order, it is a partial order.

DEFINITION 5.26

$$\mathcal{M}^s = (\mathcal{S}^s, [\![\cdot]\!]_i \cdot)$$

where

$$\mathcal{S} = (\mathcal{D}^s, \mathcal{N}^s, \mathcal{P}^s, \mathcal{L}^s, \mathcal{F}^s, \Phi^s, \boldsymbol{\Pi}^s, \boldsymbol{\Pi}^s_1, \boldsymbol{\Pi}^s_2, +^s_{\mathcal{N}}, -^s_{\mathcal{N}}, \iota^s, \sqsubseteq^s)$$

and

$$\Pi(A) = \bigcup_{a \in A} \{d \in \mathbb{E}^\omega \mid d \mathrel{\underline{\lesssim}} a\}$$

$$\mathcal{D}^s_\perp = \Delta^\omega_\emptyset = \{\Pi(A) \mid A \in \Delta_\emptyset\}$$

$$\mathcal{L}^s = \{D \in \mathcal{D}^s_\perp \mid D \cap \mathbb{L} \neq \emptyset\}$$

$$\mathcal{P}^s = \{D \in \mathcal{D}^s_\perp \mid D \cap \mathbb{P} \neq \emptyset\}$$

$$\mathcal{N}^s = \{D \in \mathcal{D}^s_\perp \mid D \cap \mathbb{N} \neq \emptyset\}$$

$$\mathcal{D}^s = \mathcal{N}^s + \mathcal{P}^s + \mathcal{L}^s$$

$$[\![a]\!]^s \rho = \Pi(\{a^\rho\})$$

$$\Phi^s(\Pi(\lambda x.D_0)) = \mathit{lift}(\boldsymbol{\lambda} D \in \mathcal{D}^s.[\![D_0\{x := D\}]\!]^s), \quad \mathcal{F}^s = \mathrm{Rng}(\Phi^s)$$

$$\boldsymbol{\Pi}^s(\Pi(D_1), \Pi(D_2)) = \Pi(\mathtt{pr}(D_1, D_2))$$

$$\boldsymbol{\Pi}^s_1(\Pi(\mathtt{pr}(D_1, D_2))) = \Pi(D_1)$$

$$\boldsymbol{\Pi}^s_2(\Pi(\mathtt{pr}(D_1, D_2))) = \Pi(D_2)$$

$$\sqsubseteq^s \;=\; \mathrel{\underline{\lesssim}}_{\mathrm{s}}$$

Recall that by Lemma 4.32, $\mathrel{\underline{\lesssim}}_{\mathrm{s}}$ restricted to $\Delta^\omega_\emptyset$ is just $\subseteq$, and by Definition 5.10, $[\![A]\!] = \bigsqcup\{[\![a]\!] \mid a \in A\}$, for $A \in \mathcal{D}^s$.

LEMMA 5.27 $\mathcal{M}^s \in \mathbf{SFA} \cap \mathbf{CON}$.

PROOF: First we establish $\mathcal{M}^t \in \mathbf{FLEM}$. Φ^s is onto by definition. To see that it is into, suppose not. Then there would be $\lambda x.D_0 \not\cong \lambda x.D_1$ and $(\lambda x.D_0)(D) \cong (\lambda x.D_1)(D)$ for all D by definition of Φ^s, but this contradicts $\mathrel{\underline{\lesssim}}_{\mathrm{s}}$ extensionality, Lemma 4.10. The required properties of $\boldsymbol{\Pi}^s$, $\boldsymbol{\Pi}^s_1$, and $\boldsymbol{\Pi}^s_2$ are similarly direct. $\sqsubseteq^s$ is a partial order over $\Delta^\omega_\emptyset$ by Lemma 4.34. $\sqsubseteq$ Property 2. is direct from the definition, 3. follows by $\mathrel{\underline{\lesssim}}_{\mathrm{s}}$ pre-congruence, and 4. by $\mathrel{\underline{\lesssim}}_{\mathrm{s}}$ extensionality. $[\![\cdot]\!]^s \cdot$ is an environment model by a simple structural induction. Thus, $\mathcal{M}^s \in \mathbf{FLEM}$. Next we show that $\mathcal{M}^s \in \mathbf{SFA}$. Expanding definitions, this amounts to showing $\Pi(D_0) \cong_{\mathrm{s}} \Pi(D_1)$ iff for all $\Pi(D) \in \mathcal{L}^s$, $\Pi(D)(\Pi(D_0))\downarrow \Leftrightarrow \Pi(V)(\Pi(D_1))\downarrow$. This in turn is direct from the definition of $\cong_{\mathrm{s}}$ and Theorem 4.6. That $\mathcal{M}^s \in \mathbf{CPO}$ follows directly from Lemma 4.34. $\mathcal{M}^s \in \mathbf{CON}$ is a consequence of Lemma 4.37. □

5.5 Milner's Uniqueness Theorem

In this section we prove an untyped version of Milner's uniqueness theorem [Milner, 1977]: all continuous, fully abstract models of the typed lambda calculus that articulate their base domains are isomorphic. In the untyped framework we prove that all continuous, *strongly fully abstract* models

are isomorphic. This slight weakening is due to the open question, raised earlier, of whether or not continuous fully abstract models are ω-algebraic in the untyped case. We begin by characterizing the finite elements in strongly fully abstract, continuous **FLEM** models, leading to a proof that all finite elements of such models are definable.

DEFINITION 5.28 (GLB $\sqcap(a,b)$)

$$\begin{aligned}\sqcap(a,b) = \ & \mathtt{if}(\mathtt{isnat}(a), \mathtt{if}(\mathtt{isnat}(b), \mathtt{if}(\mathtt{nateq}(a,b), a, \mathtt{bot}), \mathtt{bot}),\\ & \mathtt{if}(\mathtt{ispr}(a), \mathtt{if}(\mathtt{ispr}(b), \mathtt{pr}(\sqcap(\mathtt{fst}(a), \mathtt{fst}(b)), \sqcap(\mathtt{snd}(a), \mathtt{snd}(b))), \mathtt{bot}),\\ & \mathtt{if}(\mathtt{islam}(a), \mathtt{if}(\mathtt{islam}(b), \lambda y.\sqcap(\mathtt{app}(a,y), \mathtt{app}(b,y)), \mathtt{bot}), \mathtt{bot})))\\ \sqcap(\delta_0,\delta_1) = \ & [\![\sqcap(x,y)]\!]\{x := \delta_0\}\{y := \delta_1\}\end{aligned}$$

LEMMA 5.29 (GREATEST LOWER BOUND) For $\mathcal{D} \in \mathbf{FLEM}$, $\delta_0, \delta_1 \in \mathcal{D}^k$, $\sqcap(\delta_0,\delta_1) \sqsubseteq \delta_0, \delta_1$, and if some other $\delta \sqsubseteq \delta_0, \delta_1$, then $\delta \sqsubseteq \sqcap(\delta_0,\delta_1)$.

PROOF: By induction on k, using $\sqsubseteq$ extensionality to prove the function case. □

Define sets of definable finite elements as follows: $\mathcal{D}^k_{\text{def}} = \mathcal{D}^k \cap \mathcal{D}_{\text{def}}$, $\mathcal{D}^\omega_{\text{def}} = \mathcal{D}^\omega \cap \mathcal{D}_{\text{def}}$, $\mathcal{F}^k_{\text{def}} = \Phi(\mathcal{D}^k_{\text{def}} \cap \mathcal{L})$.

LEMMA 5.30 For $\mathcal{M} \in \mathbf{SFA} \cap \mathbf{CON}$,

$$(\forall k)(\delta_0, \delta_1 \in \mathcal{D}^k \Rightarrow (\forall \phi \in \mathcal{F}^{k+1}_{\text{def}})(\phi(\delta_0) \neq \bot \Rightarrow \phi(\delta_1) \neq \bot) \Rightarrow \delta_0 \sqsubseteq \delta_1).$$

PROOF: The result is trivial for $k = 0$, assume $k > 0$. By the definition of **SFA**, to show $\delta_0 \sqsubseteq \delta_1$ it suffices to show

$$(\forall \phi \in \mathcal{F}_{\text{def}})(\phi(\delta_0) \neq \bot \Rightarrow \phi(\delta_1) \neq \bot)$$

Further, without loss of generality we may restrict ϕ in the above to have range $\{0, \bot\}$. Letting $\phi' = \boldsymbol{\pi}^k \circ \phi \circ \boldsymbol{\pi}^k$,

$$\phi'(\delta_i) \neq \bot \Leftrightarrow \boldsymbol{\pi}^k(\phi(\delta_i)) \neq \bot \Leftrightarrow \phi(\delta_i) \neq \bot, \quad i < 2,$$

by observing $\boldsymbol{\pi}^k \circ \boldsymbol{\pi}^k = \boldsymbol{\pi}^k$, and $\boldsymbol{\pi}^k(0) = 0$ and $\boldsymbol{\pi}^k(\bot) = \bot$. So, it suffices to show

$$(\forall \phi \in \mathcal{F}_{\text{def}})(\phi'(\delta_0) \neq \bot \Rightarrow \phi'(\delta_1) \neq \bot)$$

And, $\phi' = \Phi(\boldsymbol{\pi}^{k+1}(\Phi^{-1}(\phi)))$ by the definition of $[\![\cdot]\!]\cdot$, so $\phi \in \mathcal{F}^{k+1}_{\text{def}}$ and the above goal then corresponds to our assumption. □

We now prove the key lemma: all finite elements in the domain are the interpretations of some expression in the model. This means the finite expressions of $\mathbb{E}$ and finite elments of $\mathcal{D}$ coincide.

LEMMA 5.31 (DEFINABLE) For $\mathcal{M} \in \mathbf{SFA} \cap \mathbf{CON}$, all finite elements in $\mathcal{D}$ are definable: $\mathcal{D}^\omega = \mathcal{D}^\omega_{\text{def}}$.

PROOF: We prove $\mathcal{D}^k = \mathcal{D}^k_{\text{def}}$ by induction on k, from this $\mathcal{D}^\omega = \mathcal{D}^\omega_{\text{def}}$ follows directly. Suppose there existed an undefinable $\delta \in \mathcal{D}^k$, $\delta \notin \mathcal{D}^k_{\text{def}}$. for each $\delta' \in \mathcal{D}^k_{\text{def}}$, either $\delta \sqsubseteq \delta'$ or not. Group the former δ' into the set D_a, the rest into D_b. We consider two cases depending on whether or not $D_a = \emptyset$.

CASE $D_a \neq \emptyset$: Both sets are finite by Lemma 5.17. Define $\delta_\sqcap = \sqcap D_a$; $\delta_\sqcap \in \mathcal{D}_{\text{def}}$ by Lemma 5.29. Thus, $\delta \sqsubseteq \delta_\sqcap$ and $\delta \neq \delta_\sqcap$. So, by Lemma 5.30, there is a definable $\phi_\sqcap \in \mathcal{F}_{\text{def}}^{k+1}$ with $\phi_\sqcap(\delta_\sqcap) \neq \bot$, $\phi_\sqcap(\delta) = \bot$.

Next consider D_b; for its (finite) members $\delta_1, \ldots, \delta_m \in D_b$, by Lemma 5.30 again, there are $\phi_1, \ldots, \phi_m \in \mathcal{F}_{\text{def}}^{k+1}$ such that $\phi_i(\delta) \neq \bot$ and $\phi_i(\delta_i) = \bot$.

Letting $\phi^{-1} = a$ such that $\Phi([\![a]\!]) = \phi$, defined on all $\phi \in \mathcal{F}_{\text{def}}$, form expressions

$$H_1 = \lambda x.\mathtt{seq}(\phi_1^{-1}(x), \ldots, \phi_m^{-1}(x), \phi_\sqcap^{-1}(x))$$
$$H_2 = \lambda x.\mathtt{seq}(\phi_1^{-1}(x), \ldots, \phi_m^{-1}(x))$$

and observe $H_1 \cong H_2$: it suffices to show $\mathtt{app}(H_1, a) \cong \mathtt{app}(H_2, a)$ for $a \cong \pi^k(a)$ by Theorem 3.5 and the finiteness of $\phi \in \mathcal{F}_{\text{def}}^{k+1}$. This is clear, because both H_1 and H_2 converge for $\delta' \in D_a$, while they both diverge for $\delta' \in D_b$, and that covers all (definable) finite elements at level k. However, clearly $\Phi([\![H_1]\!])(\delta)$ and $\Phi([\![H_2]\!])(\delta)$ are distinct, as the first is equivalent to $\bot$ and the second to 1, so $[\![H_1]\!] \neq [\![H_2]\!]$. Thus, full abstraction is contradicted, so there must have been no such δ to begin with.

CASE $D_a = \emptyset$: In this case form expressions

$$H_1 = \lambda x.\mathtt{seq}(\phi_1^{-1}(x), \ldots, \phi_m^{-1}(x))$$
$$H_2 = \bot$$

and reason as in the previous case. □

We may now prove Milner's Uniqueness Theorem.

THEOREM 5.32 (MILNER'S UNIQUENESS THEOREM) All $\mathcal{M} \in \mathbf{SFA} \cap \mathbf{CON}$ are isomorphic.

PROOF: We establish this result by showing for arbitrary $\mathcal{M} \in \mathbf{SFA} \cap \mathbf{CON}$ that $\mathcal{M} \sim \mathcal{M}^s$. Define $\Sigma \in \mathcal{D}^s \to \mathcal{D}$ to be $\boldsymbol{\lambda} D.[\![D]\!]$, recalling $[\![A]\!] = \bigsqcup\{[\![a]\!] \mid a \in A\}$. We now proceed to show all the requirements of the isomorphism definition are satisfied. To verify that Σ is into, suppose $[\![D_0]\!] = [\![D_1]\!]$, we show $D_0 = D_1$. By Lemma 5.11, $D_0 \cong_{\text{s}} D_1$, thus $D_0 = D_1$ by Lemmas 4.28 and 4.32, uses of which we refrain from citing hereafter. To verify that Σ is onto, we pick an arbitrary $\delta \in \mathcal{D}$ and show $\Sigma(A) = \delta$ for some A. By Lemma 5.12, and Lemma 5.31, letting $D = \bigsqcup\{\delta_0 \in \mathcal{D}_{\text{def}}^{\omega} \mid \delta_0 \sqsubseteq \delta\}$, we have $\delta = \bigsqcup D$. For $\delta \in D$, let δ^{-1} be $[\![\delta^{-1}]\!] = \delta$, let and $D^{-1} = \{\delta_0^{-1} \mid \delta_0 \in D\}$. Pick A to be $\Pi(D^{-1})$. Then,

$$\begin{aligned}
\Sigma(\Pi(D^{-1})) = \delta \quad &\text{iff} \quad [\![\Pi(D^{-1})]\!] = \delta \\
&\text{iff} \quad \bigsqcup\{[\![\Pi(\delta^{-1})]\!] \mid \delta^{-1} \in D^{-1}\} = \bigsqcup D \\
&\text{iff} \quad (\forall \delta_0 \in D)([\![\Pi(\delta_0^{-1})]\!] = \delta_0) \\
&\text{iff} \quad (\forall \delta_0 \in D)([\![\delta_0^{-1}]\!] = \delta_0)
\end{aligned}$$

and the final equation is trivial by definition. Property (i) is direct by inspection of the definition of Σ and $[\![\cdot]\!]\cdot$. Property (ii) is also direct by Lemma 5.11. To verify property (iii), we show that $\mathit{lift}(\Sigma)(\Phi^s(D_0)(D_1)) = \Phi(\Sigma(D_0))(\Sigma(D_1))$. $D_0 \in \mathcal{L}^s$, and thus, without loss, may be written $D_0 = \lambda x.A$. Working from the right side of the equation,

$$\begin{aligned}
\Phi(\Sigma(\lambda x.A))(\Sigma(D_1)) &= \Phi([\![\lambda x.A]\!])([\![D_1]\!]) \\
&= [\![A]\!]\{x := [\![D_1]\!]\}
\end{aligned}$$

$$
\begin{aligned}
&= [\![A\{x := D_1\}]\!] \qquad \text{By Lemma 5.3} \\
&= \Sigma(A\{x := D_1\}) \quad \text{if } A\{x := D_1\}\downarrow, \qquad \text{otherwise} \quad = \bot \\
&= \Sigma(\Phi^s(D_0)(D_1)) \quad \text{if } \Phi^s(D_0)(D_1) \neq \bot, \qquad \text{otherwise} \quad = \bot \\
&= \mathit{lift}(\Sigma)(\Phi^s(D_0)(D_1)).
\end{aligned}
$$

To verify property (iv) observe that

$$
\begin{aligned}
\Sigma(\boldsymbol{\Pi}^s(D_0, D_1) &= \Sigma(\mathtt{pr}(D_0, D_1)) \\
&= [\![\mathtt{pr}(D_0, D_1)]\!] \\
&= \boldsymbol{\Pi}([\![D_0]\!], [\![D_1]\!]) \\
&= \boldsymbol{\Pi}(\Sigma(D_0), \Sigma(D_1))
\end{aligned}
$$

The remaining properties are proved similarly. □

COROLLARY 5.33 $\mathbf{SFA} \cap \mathbf{CON} \subseteq \mathbf{LFP}$

6 Concluding remarks

A topic for future work is to consider the generality of techniques employed in this paper. There appear to be no significant problems in defining a useful $\precsim_{\mathrm{s}}$ relation on enriched languages with features such as state and explicit control operators. The basic theory of $\precsim_{\mathrm{s}}$ strongly parallels the basic theory of $\precsim$ developed within (in fact the proofs for $\precsim_{\mathrm{s}}$ are very minor generalizations on proofs $\precsim$, see sections 3.1 and 4.2), and for instance in [Mason and Talcott, 1991] it is shown how a basic theory of $\precsim$ may be developed for languages with state. However, it is unclear if the notion of finite expression will generalize to languages with features such as state and continuations. The presence or absense of recognizers `ispr`, `isnat` are irrelevant for all results up to the finite expressions of section 4.4. However, they play a key rôle in the construction of finite expressions. It is an open problem if finite expressions can be constructed for a language without recognizers. Recognizers are desired for untyped languages, and indeed are present in untyped languages such as Lisp and Scheme, so their inclusion here is appropriate. There also have been a number of proposals for type recognizers of some form as a feature of typed languages [Abadi et al., 1991a].

Acknowledgements

We wish to thank the anonymous referees for helpful comments, and one of the referees in particular for ideas leading to a cleaner proof of Theorem 3.16.

References

[Abadi et al., 1991a] Abadi, M., Cardelli, L., Pierce, B., and Plotkin, G. (1991a). Dynamic typing in a statically typed language. *Transactions on Programming Languages and Systems*, 13(2):237–268.

[Abadi et al., 1991b] Abadi, M., Pierce, B., and Plotkin, G. (1991b). Faithful ideal models for recursive polymorphic types. *International Journal of Foundations of Computer Science*, 2(1):1–21.

[Abramsky, 1990] Abramsky, S. (1990). The lazy lambda calculus. In *Research Topics in Functional Programming*, pages 65–116. Addison-Wesley.

[Bloom, 1990] Bloom, B. (1990). Can LCF be topped? *Information and Computation*, 87:264–301.

[deBakker and Scott, 1969] deBakker, J. W. and Scott, D. (1969). A theory of programs. unpublished Notes.

[Felleisen et al., 1987] Felleisen, M., Friedman, D., and Kohlbecker, E. (1987). A syntactic theory of sequential control. *Theoretical Computer Science*, 52:205–237.

[Gordon, 1994] Gordon, A. D. (1994). *Functional Programming and Input/Output*. Cambridge University Press.

[Honsell et al., 1995] Honsell, F., Mason, I. A., Smith, S. F., and Talcott, C. L. (1995). A variable typed logic of effects. *Information and Computation*, 119(1):55–90.

[Howe, 1989] Howe, D. J. (1989). Equality in lazy computation systems. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science*, pages 198–203. IEEE.

[Howe, 1995] Howe, D. J. (1995). Proving congruence of bisimulation in functional programming languages. Preprint.

[Hyland, 1976] Hyland, J. M. E. (1976). A survey of some useful partial order relations on terms of the lambda calculus. In Boehm, C., editor, *Lambda Calculus and Computer Science Theory*, volume 37 of *Lecture Notes in Computer Science*, pages 83–93. Springer-Verlag.

[Igarashi, 1972] Igarashi, S. (1972). Admissibility of fixed-point induction in first-order logic of typed theories. Technical Report Stan-CS-72-287, Stanford University Computer Science Department.

[Jim and Meyer, 1991] Jim, T. and Meyer, A. (1991). Full abstraction and the context lemma. In *Theoretical Aspects of Computer Science*, volume 526 of *Lecture Notes in Computer Science*, pages 131–151. Springer-Verlag.

[MacQueen et al., 1984] MacQueen, D. B., Plotkin, G., and Sethi, R. (1984). An ideal model of types. In *Conference Record of the Eleventh Annual ACM Symposium on Principles of Programming Languages*.

[Manna, 1974] Manna, Z. (1974). *Mathematical Theory of Computation*. McGraw-Hill.

[Mason and Talcott, 1991] Mason, I. A. and Talcott, C. L. (1991). Equivalence in functional languages with effects. *Journal of Functional Programming*, 1:287–327.

[Meyer, 1982] Meyer, A. R. (1982). What is a model of the lambda calculus? *Information and Computation*, 52:87–122.

[Milner, 1977] Milner, R. (1977). Fully abstract models of typed λ-calculi. *Theoretical Computer Science*, 4:1–22.

[Ong, 1988] Ong, C.-H. L. (1988). Fully abstract models of the lazy lambda calculus. In *Symposium on the Foundations of Computer Science*, pages 368–376.

[Ong, 1992] Ong, C.-H. L. (1992). The concurrent lambda calculus i: a general precongruence theorem for applicative bisimulation. Proceedings on Seminars on Parallel Programming Systems, Department of Information Systems and Computer Science, National University of Singapore.

[Paulson, 1987] Paulson, L. C. (1987). *Logic and Computation: Interactive Proof with Cambridge LCF*. Cambridge.

[Pitts, 1994] Pitts, A. M. (1994). A co-induction principle for recursively defined domains. *Theoretical Computer Science*, 124:195–219.

[Pitts and Stark, 1993] Pitts, A. M. and Stark, I. D. B. (1993). Observable properties of higher order functions that dynamically create local names, or: What's new? In *Mathematical Foundations of Computer Science, Proc. 18th Int. Symp., Gdańsk, 1993*, volume 711 of *Lecture Notes in Computer Science*, pages 122–141. Springer-Verlag, Berlin.

[Ritter and Pitts, 1995] Ritter, E. and Pitts, A. M. (1995). A fully abstract translation between a λ-calculus with reference types and standard ml. In *2nd Int. Conf. on Typed Lambda Calculus and Applications, Edinburgh, 1995*, volume 902 of *Lecture Notes in Computer Science*, pages 397–413. Springer-Verlag, Berlin.

[Smith, 1992] Smith, S. F. (1992). From operational to denotational semantics. In *MFPS 1991*, volume 598 of *Lecture notes in Computer Science*, pages 54–76. Springer-Verlag.

[Stoughton, 1988] Stoughton, A. (1988). *Fully abstract models of programming languages*. Research Notes in theoretical computer science. Pitman.

[Talcott, 1985] Talcott, C. L. (1985). *The essence of Rum: A theory of the intensional and extensional aspects of Lisp-type computation*. PhD thesis, Stanford University.

[Wadsworth, 1976] Wadsworth, C. (1976). Relation between computational and denotational properties for Scott's D^∞ models of the λ-calculus. *SIAM J Computing*.