A Category-theoretic Semantics for Unbounded Indeterminacy

Prakash Panangaden
James Russell*

88-957
December 1988

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Supported in part by an NSF Graduate Fellowship

A Category-theoretic Semantics for Unbounded Indeterminacy

Prakash Panangaden
James Russell*
Computer Science Department, Cornell University

December 9, 1988

Abstract

In this paper we give a category-theoretic semantics for a simple imperative language featuring unbounded indeterminacy. This semantics satisfies the categorical analogues of continuity and has the meaning of while loops defined as co-limits of ω-diagrams. Furthermore, it collapses via an abstraction function to a semantics that is fully abstract, and coincides with the operational semantics. The abstraction function is the only discontinuous function appearing in our semantics.

1 Introduction

In the recent book "Fairness" by Nissim Francez [5] the following quote appears:

> One of the more challenging problems in modeling fairness properties is to reconcile infinite fair behaviors with approximations and limits.

The presence of fairness usually points to the presence of unbounded indeterminacy [2]. In a recent paper, Apt and Plotkin [3] showed that it is impossible to have a continuous, least fixed-point, fully abstract semantics using domain theory. This proof does not depend on the details of any particular powerdomain construction. We show that one can get a "continuous" semantics using category theoretic methods that were developed

*Supported in part by an NSF graduate fellowship

by Lehmann [6]. The meanings of while loops are given by co-limits of ω-diagrams. The semantics that we provide, is adequate, though while not fully abstract. However it collapses, via an abstraction function, to a semantics that coincides with operational equality and *is* fully abstract. Apt and Plotkin [3] give a semantics for this language that is fully abstract but it is not continuous. The failure of continuity in our case is isolated to the abstraction function.

Lehmann's category theoretic approach to powerdomains was based on the idea that morphisms in a category could convey a more precise notion of approximation than partial orders could. He developed category-theoretic generalizations of many standard domain-theoretic concepts including the Smyth powerdomain [12]. He did not use these constructions for defining the semantics of any languages. Several years later, Samson Abramsky [1] sketched a semantics for an applicative language with unbounded indeterminacy and investigated mathematical properties of the resulting powerdomains.

We view our work as the beginning of a study of category-theoretic techniques for modeling unbounded indeterminacy. Recent work has shown that there are more "vicious" forms of unbounded indeterminacy than the one described here or in Apt and Plotkin. In particular we know that various commonly considered dataflow primitives, such as fair merge, are not even monotone in a suitable sense [8,9]. We are interested in developing fixed-point theories for reasoning about such primitives as well.

The rest of this paper is organized as follows. Section 2 describes the language and its operational semantics. In section 3 we describe categorical powerdomains; this material is essentially a summary of the relevant parts of Lehmann's thesis [6]. Sections 4 and 5 describe the semantics and provide some examples. In section 6 we establish the relationship between the operational semantics and the denotational semantics.

2 Operational Semantics of the Language

We describe a simple language for non-determinism based on the one presented in Apt and Plotkin [3]. Our language is a simplified version of that one, the state consisting of the value of a single integer variable, usually called x. It should be clear that the treatment extends readily to any 'flat' domain of states.

The Atomic Commands are the ones that change the state, i.e. set the value of x.

$$A \ ::= \ x := n \mid x := x - 1 \mid x := x + 1 \mid x :=?$$

The intended meaning of $x :=?$ is non-deterministic assignment to x of any value from the underlying domain (which in this case is the integers). The details of the Boolean expressions are not important, since they are intended to be side-effect free. We assume at least the ability to detect when x is 0.

$$B \ ::= \ x = 0 \mid x \neq 0 \mid \cdots$$

Finally, we have the following Commands:

$$S \ ::= \ A \mid \mathsf{skip} \mid S_1; S_2 \mid \mathsf{if}\ B\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \mathsf{fi} \mid \mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}$$

We give the operational semantics via a one-step transition function **Com** $\times$ **States** $\rightarrow$ **Com** $\times$ **States** $\cup$ **States**. We assume the existence of a boolean evaluation function $\mathcal{B}[\![\cdot]\!]$: **Bexp** $\rightarrow$ (**States** $\rightarrow$ $\{\mathsf{true}, \mathsf{false}\}$); we explicitly assume that the evaluation of boolean expressions terminates. In what follows we use σ to range over **States**.

$\langle x := n, \sigma\rangle \ \rightarrow n$
$\langle x := x - 1, \sigma\rangle \ \rightarrow \sigma - 1$
$\langle x := x + 1, \sigma\rangle \ \rightarrow \sigma + 1$
$\langle x :=?, \sigma\rangle \ \rightarrow n\ \forall n \in D$
$\langle \mathsf{skip}, \sigma\rangle \ \rightarrow \sigma$
$\langle S_1; S_2, \sigma\rangle \ \rightarrow \langle S_1'; S_2, \sigma'\rangle$ if $\langle S_1, \sigma\rangle \ \rightarrow \langle S_1', \sigma'\rangle$
$\langle S_1; S_2, \sigma\rangle \ \rightarrow \langle S_2, \sigma'\rangle$ if $\langle S_1, \sigma\rangle \ \rightarrow \sigma'$
$\langle \mathsf{if}\ B\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \mathsf{fi}, \sigma\rangle \ \rightarrow \langle S_1, \sigma\rangle$ if $\mathcal{B}[\![B]\!]\sigma = \mathsf{true}$
$\langle \mathsf{if}\ B\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \mathsf{fi}, \sigma\rangle \ \rightarrow \langle S_2, \sigma\rangle$ if $\mathcal{B}[\![B]\!]\sigma = \mathsf{false}$
$\langle \mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}, \sigma\rangle \ \rightarrow \langle S; \mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}, \sigma\rangle$ if $\mathcal{B}[\![B]\!]\sigma = \mathsf{true}$
$\langle \mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}, \sigma\rangle \ \rightarrow \sigma$ if $\mathcal{B}[\![B]\!]\sigma = \mathsf{false}$

We define the notation $\langle S, \sigma\rangle \uparrow$ to mean that there exists an infinite sequence of transitions

$$\langle S, \sigma\rangle = \langle S_0, \sigma_0\rangle \rightarrow \langle S_1, \sigma_1\rangle \rightarrow \langle S_2, \sigma_2\rangle \rightarrow \cdots,$$

and we define the operational meaning function $Op[\![]\!]$ by

$$Op[\![S]\!]\sigma \stackrel{\mathrm{def}}{=} \{\sigma' | \langle S, \sigma\rangle \rightarrow^* \sigma'\} \cup \{\bot | \langle S, \sigma\rangle \uparrow\}.$$

3 Categorical Powerdomains

Powerdomains were originally introduced as the domain-theoretic analogues of powersets and were intended to be used for semantic treatments of indeterminacy [11,12]. The approach was generalized to categories by Lehmann [6]. Lehmann's approach was to use categories as the semantic spaces rather than domains. His idea was that a 'more detailed' notion of approximation between elements can be expressed by using morphisms between objects in a category than by using partial orders. Dually, one may view traditional domain theory as a special case of a category theoretic approach in which the homsets are at most singletons.

Abramsky [1] used Lehmann's construction to model unbounded indeterminacy. A categorical approach, but with a different construction for the powerdomain, was also used by Panangaden [7] to model dataflow networks with fair merge. Recently it has been realized that one can use categories to model lambda calculi [4].

Though Lehmann's original construction was for domains that were generalized to categories, in this paper we assume that the underlying domain is a partial order. As already noted, a poset can be viewed as a category in which the arrows are unique; we will often adopt this view of our domain when discussing the powerdomain construction.

Given a domain D the construction of what we will call the *categorical powerdomain* of D, $CP(D)$, is straightforward. The objects of the category are taken to be multisets - sets with repetitions - of the elements of D. Given multisets A and B an arrow $G : A \longrightarrow B$ is defined as a set $G \subseteq \mathit{Arrows}(D)$ such that for each $b \in B$, there is a unique arrow $a \longrightarrow b, a \in A$ in the set G. In our case where D is a poset, arrows are always unique and we may represent an arrow $a \longrightarrow b$ by the pair $\langle a, b\rangle$, with $a \sqsubseteq b$. Hence, we will use the following definition of an arrow G:

$$\begin{aligned} G \subseteq A \times B \text{ such that } & \langle a, b\rangle \in G \Rightarrow a \sqsubseteq b \;\& \\ & \forall b \in B \exists! a \in A . \langle a, b\rangle \in G. \end{aligned}$$

Composition of arrows and identity arrows in $CP(D)$ are straightforward.

The categorical analog of a continuous function is a functor that preserves colimits of ω-diagrams. The definition of f a functor automatically assures the analog of monotonicity - i.e. if there is an arrow $a \xrightarrow{r} b$, then there is an arrow $f(a) \xrightarrow{f(r)} f(b)$.

We now present two theorems about $CP(D)$ originally due to Lehmann.

These establish that $CP(D)$ has the properties that one associates with a complete partial order.

Theorem 1. $\{\perp\}$ is the initial object.

Proof: For all $d \in D$ we have $\perp \sqsubseteq d$, hence for any object B of $CP(D)$ there is a unique arrow $\{\perp\} \longrightarrow B$ given by $G = \{\langle\perp, b\rangle | b \in B\}$.

Theorem 2. All ω-diagrams have colimits

Proof (sketch): Consider a diagram $X_0 \xrightarrow{G_0} X_1 \xrightarrow{G_1} X_2 \xrightarrow{G_2} \cdots$.

Let S be the collection of all chains $Q = \{q_0, q_1, \ldots\}$ s.t. for all $i, q_i \in X_i$ and $\langle q_i, q_{i+1}\rangle \in X_i$. We define the colimiting object X^* by $X^* = \mathrm{colim}(X_i) = \biguplus_{Q \in S} \sqcup Q$. The colimiting arrows $X_i \xrightarrow{G_i^*} X^*$ are given by $G_i^* = \biguplus_{Q \in S}\{\langle x, \sqcup Q\rangle | x \in X_i \cap Q\}$. We omit the details of commutativity and couniversality.

Note that in the above (and in the sequel) we use the 'tagged union' symbol $\biguplus$ for unions that yield multisets. The intention is that multiple copies of the same element are not identified.

4 Semantics of the Language

In this section we use the categorical powerdomain $CP(D)$ defined above to give a denotational semantics to our language. First, we need to define our domains. Our base domain D will be an unrelated set of states (which in this case is the integers ω). However, since non-termination is a possibility in the language, we must consider $CP(D_\perp)$, the categorical powerdomain of D with $\perp$ added. Our semantics will then be a function $\mathbf{Com} \to (D \to CP(D_\perp))$, i.e. the meaning of commands will be given as ω-colimit preserving functors from D to $CP(D_\perp)$.

Before we can give the semantics, there are three auxiliary functors we must define:

Singleton $\{\cdot\}$: This takes an element d of D to the object $\{d\}$ (the singleton multiset containing one copy of d) of $CP(D_\perp)$. It takes an arrow $a \longrightarrow b$ in D to the arrow $\{a\} \xrightarrow{G} \{b\}$ given by $G = \{\langle a, b\rangle\}$. It is easily seen that this is ω-colimit preserving.

Lifting $(\cdot)^\dagger$: This is a functor between the functor categories $(D \to CP(D_\perp))$ and $(CP(D_\perp) \to CP(D_\perp))$. Note that $\perp \notin D$, so a functor $f : (D \to CP(D_\perp))$ is not defined on $\perp$, while multisets $A \in CP(D_\perp)$ may contain $\perp$. For this reason it is necessary that lifting specify explicitly the action of $f^\dagger$ and $\eta^\dagger$ on $\perp$.

$$f^\dagger(A) \stackrel{\text{def}}{=} \biguplus_{a\in A} f(a) \uplus \biguplus_{\perp\in A}\{\perp\}$$

$$f^\dagger(G : A \longrightarrow B) \stackrel{\text{def}}{=} \biguplus_{\substack{\langle a,\, b\rangle \in G \\ a \neq \perp}} f(a \longrightarrow b) \uplus \biguplus_{\substack{p\langle \perp,\, b\rangle \in G \\ b \neq \perp}} \{\langle \perp,\, y\rangle | y \in f(b)\} \uplus \biguplus_{\langle \perp,\, \perp\rangle \in G} \{\langle \perp,\, \perp\rangle\}$$

The above describes what lifting does to objects (which in this case are functors). We now describe what lifting does to arrows (which are natural transformations). Recall that natural transformations are maps from objects to arrows.

Given $\eta : f \longrightarrow g$, we have $\eta^\dagger : f^\dagger \longrightarrow g^\dagger$, given by

$$\eta^\dagger(A) \stackrel{\text{def}}{=} \biguplus_{\substack{a \in A \\ a \neq \perp}} \eta(a) \uplus \biguplus_{\perp \in A} \{\langle \perp,\, \perp\rangle\}.$$

Two important properties of lifting are that $g^\dagger$ is ω-colimit preserving if g is, and that $(\cdot)^\dagger$ is itself an ω-colimit preserving functor.

Sequential composition $\cdot\,;\cdot$: $f; g$ is shorthand for $g^\dagger \circ f$, and as such it inherits the desirable ω-colimit preservation properties from lifting.

We now describe a semantic function $\mathcal{D}[\![\cdot]\!] : \mathbf{Com} \rightarrow (D \rightarrow CP(D_\perp))$. Note that since D is completely 'flat', when it is interpreted as a category the only arrows are the identities. Thus, when we describe the functors in the semantics we will specify only their action on objects, since functors by definition preserve identity arrows. If we were to give a semantics for a non-flat domain, we would be required to explicitly specify the action of the semantic functors on the arrows of the domain, since in general this is not trivial. We use the variable d to refer to elements of D.

$\mathcal{D}[\![x :=?]\!] = \lambda d.D$

$\mathcal{D}[\![x := c]\!] = \lambda d.\{c\}$

$\mathcal{D}[\![x := x - 1]\!] = \lambda d.\{d - 1\}$

$\mathcal{D}[\![x := x + 1]\!] = \lambda d.\{d + 1\}$

$\mathcal{D}[\![\mathsf{skip}]\!] = \lambda d.\{d\}$

$\mathcal{D}[\![S_1; S_2]\!] = \mathcal{D}[\![S_1]\!]; \mathcal{D}[\![S_2]\!]$

$$\mathcal{D}[\![\mathsf{if}\ B\ \mathsf{then}\ S_1\ \mathsf{else}\ S_2\ \mathsf{fi}]\!] = \lambda d. \begin{cases} \mathcal{D}[\![S_1]\!]d & \text{if } \mathcal{B}[\![B]\!]d = \mathsf{true} \\ \mathcal{D}[\![S_2]\!]d & \text{if } \mathcal{B}[\![B]\!]d = \mathsf{false} \end{cases}$$

We want to define the meaning of a while loop as a colimit, and since the meaning is a functor, it must be the colimit of a diagram in the functor category $D \to CP(D_\perp)$. This is the category whose objects are functors $D \to CP(D_\perp)$ and whose arrows are natural transformations. The initial object is the functor $\lambda d.\{\perp\}$, as is easily verified. We will call this Ω. Define

$$\mathcal{D}[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!] = \mathrm{colim}(W_0 \xrightarrow{\eta_0} W_1 \xrightarrow{\eta_1} W_2 \xrightarrow{\eta_2} \cdots)$$

where W_i and η_i are defined as follows:

$W_0 = \Omega = \lambda d.\{\perp\}$

For $n \geq 1$: $W_n = \lambda d.\begin{cases} (\mathcal{D}[\![S]\!]; W_{n-1})d & \text{if } \mathcal{B}[\![B]\!]d = \mathsf{true} \\ \mathcal{D}[\![\mathsf{skip}]\!]d & \text{if } \mathcal{B}[\![B]\!]d = \mathsf{false} \end{cases}$

$\eta_0 x : W_0 x \longrightarrow W_1 x = \begin{cases} \biguplus_{x' \in \mathcal{D}[\![S]\!]x}\{\langle\perp, \perp\rangle\} & \text{if } \mathcal{B}[\![B]\!]x = \mathsf{true} \\ \{\langle\perp, x\rangle\} & \text{if } \mathcal{B}[\![B]\!]x = \mathsf{false} \end{cases}$

For $n \geq 1$: $\eta_n x : W_n x \longrightarrow W_{n+1} x =$

$$\begin{cases} \biguplus_{\substack{x' \in \mathcal{D}[\![S]\!]x \\ x' \neq \perp}} \eta_{n-1} x' \uplus \biguplus_{\perp \in \mathcal{D}[\![S]\!]x} \{\langle\perp, \perp\rangle\} & \text{if } \mathcal{B}[\![B]\!]x = \mathsf{true} \\ \{\langle x, x\rangle\} & \text{if } \mathcal{B}[\![B]\!]x = \mathsf{false} \end{cases}$$

The colimit is determined pointwise; i.e. for all d, $\mathcal{D}[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d = \mathrm{colim}(W_0 d \xrightarrow{\eta_0 d} W_1 d \xrightarrow{\eta_1 d} W_2 d \xrightarrow{\eta_2 d} \cdots)$.

5 Some Examples

Example 1.

$\mathcal{D}[\![\mathsf{while\ true\ do\ skip\ od}]\!]x =$

$$\begin{aligned} & \mathrm{colim}((\lambda d.\{\perp\})x \longrightarrow (\lambda d.\{d\}; \lambda d.\{\perp\})x \longrightarrow (\lambda d.\{d\}; \lambda d.\{d\}; \lambda d.\{\perp\})x \longrightarrow \cdots) \\ = \; & \mathrm{colim}(\{\perp\} \longrightarrow \{\perp\} \longrightarrow \{\perp\} \longrightarrow \cdots) \\ = \; & \{\perp\} \end{aligned}$$

Thus, $\mathcal{D}[\![\mathsf{while\ true\ do\ skip\ od}]\!] = \lambda d.\{\perp\} = \Omega$. This is a pleasant property, since it means we can replace Ω in the definition of $\mathcal{D}[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]$ by $\mathcal{D}[\![\mathsf{loop}]\!]$, where $\mathsf{loop} \equiv \mathsf{while\ true\ do\ skip\ od}$.

Example 2. $\mathcal{D}[\![x := 0]\!]d = \{0\}$

Example 3.

$\mathcal{D}[\![\mathsf{while}\ x > 0\ \mathsf{do}\ x := x - 1\ \mathsf{od}]\!]d =$

$$\begin{aligned} & \mathrm{colim}(\overbrace{\{\perp\} \longrightarrow \{\perp\} \longrightarrow \cdots \{\perp\}}^{d \text{ times}} \longrightarrow \{0\} \longrightarrow \{0\} \longrightarrow \cdots) \\ = \; & \{0\} \end{aligned}$$

Thus, $\mathcal{D}[\![\mathsf{while}\ x > 0\ \mathsf{do}\ x := x - 1\ \mathsf{od}]\!] = \lambda d.\{0\} = \mathcal{D}[\![x := 0]\!]$.

Example 4. $\mathcal{D}[\![x :=?; \mathsf{while}\ x > 0\ \mathsf{do}\ x := x - 1\ \mathsf{od}]\!] =$

$$
\begin{array}{llllll}
\mathrm{colim}(& \{\bot, & \bot, & \bot, & \ldots & \} \\
 & \{0, & \bot, & \bot, & \ldots & \} \\
 & \{0, & 0, & \bot, & \ldots & \} \\
 & & & \cdots) & &
\end{array}
$$

$= \{0, 0, 0, \ldots\}$

Of course, we should have expected this, since

$$
\begin{aligned}
\mathcal{D}[\![x :=?; \mathsf{while}\ x > 0\ \mathsf{do}\ x := x - 1\ \mathsf{od}]\!] &= \mathcal{D}[\![x :=?; x := 0]\!] \\
&= (\lambda d.\{0\})^{\dagger} \circ (\lambda d.D) \\
&= \biguplus_{d \in D} \{0\}.
\end{aligned}
$$

This may seem unnatural, but it is a necessary effect of our approach that $\mathcal{D}[\![x :=?; \mathsf{while}\ x > 0\ \mathsf{do}\ x := x - 1\ \mathsf{od}]\!]$ is different from $\mathcal{D}[\![x := 0]\!]$. The use of multisets and functors provides a more detailed description of the approximations, but at the same time can increase the cardinality of the approximate objects to account fo non-deterministic behavior.

The semantics of programs over the determinate portion of our language (i.e. without "$x :=?$") will always be a singleton, and will satisfy all the usual semantic relations.

6 Relationship with the Operational Semantics

In this section we begin by investigating properties of the operations semantics $Op[\![]\!]$, ultimately proving its compositionality. We then demonstrate that an abstraction of the denotational semantics $\mathcal{D}[\![]\!]$ exactly coincides with $Op[\![]\!]$. From these, the full-abstraction of the abstracted semantics follows as a corollary.

Lemma 1. For all statements S and states d,

$$
Op[\![S]\!]d = \bigcup_{\substack{S',\, d' \text{ s.t.} \\ \langle S,\, d\rangle \to \langle S',\, d'\rangle}} Op[\![S']\!]d' \cup \{d' | \langle S,\, d\rangle \to d'\}.
$$

Proof: Straightforward from the transition relations.

Lemma 2. For all statements S_1, S_2, and states d,

$$Op[\![S_1; S_2]\!]d = \bigcup_{\substack{d' \in Op[\![S_1]\!]d \\ d' \neq \bot}} Op[\![S_2]\!]d' \cup \{\bot \mid \bot \in Op[\![S_1]\!]d\}.$$

Proof: Straightforward from the transition relations.

Theorem 3. If $Op[\![S]\!] = Op[\![S']\!]$, then for all contexts $C[\cdot]$, $Op[\![C[S]]\!] = Op[\![C[S']]\!]$.

Proof: The proof is a structural induction on the context $C[\cdot]$.

<u>Case $C \equiv \textsf{if } B \textsf{ then } [\cdot] \textsf{ else } T \textsf{ fi}$:</u> From Lemma 1 it follows that, for all d,

$$\begin{aligned}
Op[\![\textsf{if } B \textsf{ then } S \textsf{ else } T \textsf{ fi}]\!]d &= \begin{cases} Op[\![S]\!]d & \text{if } \mathcal{B}[\![B]\!]d = \textsf{true} \\ Op[\![T]\!]d & \text{if } \mathcal{B}[\![B]\!]d = \textsf{false} \end{cases} \\
\text{(by hypothesis)} &= \begin{cases} Op[\![S']\!]d & \text{if } \mathcal{B}[\![B]\!]d = \textsf{true} \\ Op[\![T]\!]d & \text{if } \mathcal{B}[\![B]\!]d = \textsf{false} \end{cases} \\
\text{(by lemma)} &= Op[\![\textsf{if } B \textsf{ then } S' \textsf{ else } T \textsf{ fi}]\!]d
\end{aligned}$$

<u>Case $C \equiv \textsf{if } B \textsf{ then } [\cdot] \textsf{ else } T \textsf{ fi}$:</u> Similar to above.

<u>Case $C \equiv [\cdot]; T$:</u> We know from Lemma 2 that

$$\begin{aligned}
Op[\![S; T]\!]d &= \bigcup_{\substack{d' \in Op[\![S]\!]d \\ d' \neq \bot}} Op[\![T]\!]d' \cup \{\bot \mid \bot \in Op[\![S]\!]d\} \\
\text{(by hyp.)} &= \bigcup_{\substack{d' \in Op[\![S']\!]d \\ d' \neq \bot}} Op[\![T]\!]d' \cup \{\bot \mid \bot \in Op[\![S']\!]d\} \\
\text{(by lemma)} &= Op[\![S'; T]\!]d
\end{aligned}$$

<u>Case $C \equiv T; [\cdot]$:</u> As above, we know that

$$Op[\![T; S]\!]d = \bigcup_{\substack{d' \in Op[\![T]\!]d \\ d' \neq \bot}} Op[\![S]\!]d' \cup \{\bot \mid \bot \in Op[\![T]\!]d\}$$

$$
\begin{aligned}
\text{(by hyp.)} &= \bigcup_{\substack{d' \in Op[\![T]\!]d \\ d' \neq \bot}} Op[\![S']\!]d' \cup \{\bot \mid \bot \in Op[\![T]\!]d\} \\
\text{(by lemma)} &= Op[\![T; S']\!]d
\end{aligned}
$$

<u>Case $C \equiv$ while B do $[\cdot]$ od:</u> We show that for all d,

$$Op[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d \subseteq Op[\![\mathsf{while}\ B\ \mathsf{do}\ S'\ \mathsf{od}]\!]d.$$

By symmetry, the reverse must also be true, and the desired equality follows.

Case 1: If $\mathcal{B}[\![B]\!]d = \mathsf{false}$, then

$$Op[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d = Op[\![\mathsf{while}\ B\ \mathsf{do}\ S'\ \mathsf{od}]\!]d = \{d\}$$

Case 2: Suppose $\mathcal{B}[\![B]\!]d = \mathsf{true}$, and let $d' \in Op[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d$, $d' \neq \bot$. Then $\exists n \geq 1$ and a sequence $d = d_0, d_1, \ldots, d_n = d'$ such that

$$
\begin{aligned}
&d_{i+1} \in Op[\![S]\!]d_i \text{ for } i = 0, \ldots, n-1, \\
&\mathcal{B}[\![B]\!]d_i = \mathsf{true} \text{ for } i = 0, \ldots, n-1, \\
&\text{and } \mathcal{B}[\![B]\!]d_n = \mathsf{false}.
\end{aligned}
$$

By the hypothesis, this means

$$
\begin{aligned}
&d_{i+1} \in Op[\![S']\!]d_i \text{ for } i = 0, \ldots, n-1, \\
&\mathcal{B}[\![B]\!]d_i = \mathsf{true} \text{ for } i = 0, \ldots, n-1, \\
&\text{and } \mathcal{B}[\![B]\!]d_n = \mathsf{false},
\end{aligned}
$$

which is equivalent to

$$d' \in Op[\![\mathsf{while}\ B\ \mathsf{do}\ S'\ \mathsf{od}]\!]d.$$

Case 3: Suppose $\mathcal{B}[\![B]\!]d = \mathsf{true}$, and $\bot \in Op[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d$. Then there exists a sequence $d = d_0, d_1, \ldots$ such that

$$
\begin{aligned}
&d_{i+1} \in Op[\![S]\!]d_i \text{ for all } i, \\
&\text{and } \mathcal{B}[\![B]\!]d_i = \mathsf{true} \text{ for all } i.
\end{aligned}
$$

By the hypothesis, this means

$$
\begin{aligned}
&d_{i+1} \in Op[\![S']\!]d_i \text{ for all } i, \\
&\text{and } \mathcal{B}[\![B]\!]d_i = \mathsf{true} \text{ for all } i,
\end{aligned}
$$

or equivalently,

$$\perp \in Op[\![\text{while } B \text{ do } S' \text{ od}]\!]d.$$

∎

Definition ab : **Multiset** $\rightarrow$ **Set** is the function that takes a multiset to its underlying set: $ab(X) \stackrel{\text{def}}{=} \{x | x \in X\}$.

Note that ab is not a continuous function (as can be seen by applying ab to the sets in Example 4). This is the only aspect of our semantics that is not continuous.

Theorem 4. $ab(\mathcal{D}[\![S]\!]) = Op[\![S]\!]$ for all commands S.

Proof: The proof is a structural induction by cases.

Case $S \equiv A$: For the atomic commands A, it is trivially the case that

$$Op[\![A]\!]d = ab(\mathcal{D}[\![A]\!]d) \text{ for all } d \text{ in } D.$$

Case $S \equiv \text{skip}$: Also trivially,

$$Op[\![\text{skip}]\!]d = \{d\} = ab(\mathcal{D}[\![\text{skip}]\!]d) \text{ for all } d \text{ in } D.$$

Case $S \equiv \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}$: From Lemma 1 it follows that, for all d,

$$\begin{aligned} Op[\![\text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}]\!]d &= \begin{cases} Op[\![S_1]\!]d & \text{if } \mathcal{B}[\![B]\!]d = \text{true} \\ Op[\![S_2]\!]d & \text{if } \mathcal{B}[\![B]\!]d = \text{false} \end{cases} \\ \text{(by ind. hyp.)} &= \begin{cases} ab(\mathcal{D}[\![S_1]\!]d) & \text{if } \mathcal{B}[\![B]\!]d = \text{true} \\ ab(\mathcal{D}[\![S_2]\!]d) & \text{if } \mathcal{B}[\![B]\!]d = \text{false} \end{cases} \\ &= ab(\mathcal{D}[\![\text{if } B \text{ then } S_1 \text{ else } S_2 \text{ fi}]\!]d) \end{aligned}$$

Case $S \equiv S_1; S_2$: From the definition of sequential composition we have

$$\begin{aligned} \mathcal{D}[\![S_1; S_2]\!]d &= (\mathcal{D}[\![S_2]\!])^{\dagger}(\mathcal{D}[\![S_1]\!]d) \\ &= \biguplus_{\substack{d' \in \mathcal{D}[\![S_1]\!]d \\ d' \neq \perp}} \mathcal{D}[\![S_2]\!]d' \uplus \biguplus_{\perp \in \mathcal{D}[\![S_1]\!]d} \{\perp\}. \end{aligned}$$

This means that, for all d,

$$
\begin{aligned}
ab(\mathcal{D}[\![S_1; S_2]\!]d) &= \bigcup_{\substack{d' \in ab(\mathcal{D}[\![S_1]\!]d) \\ d' \neq \perp}} ab(\mathcal{D}[\![S_2]\!]d') \cup \{\perp | \perp \in ab(\mathcal{D}[\![S_1]\!]d)\} \\
\text{(by ind. hyp.)} &= \bigcup_{\substack{d' \in Op[\![S_1]\!]d \\ d' \neq \perp}} Op[\![S_2]\!]d' \cup \{\perp | \perp \in Op[\![S_1]\!]d\} \\
\text{(by Lemma 2)} &= Op[\![S_1; S_2]\!]d
\end{aligned}
$$

<u>Case $S \equiv$ while B do S od:</u> This case proceeds via two lemmas.

Lemma 3. For all d, $Op[\![\text{while } B \text{ do } S \text{ od}]\!]d \subseteq ab(\mathcal{D}[\![\text{while } B \text{ do } S \text{ od}]\!]d)$

Proof: Let $d' \in Op[\![\text{while } B \text{ do } S \text{ od}]\!]d$.

Case 1: If $\mathcal{B}[\![B]\!]d = \mathsf{false}$, then $d = d'$, and

$$
\begin{aligned}
\mathcal{D}[\![\text{while } B \text{ do } S \text{ od}]\!] &= \operatorname{colim}(W_0 d \to W_1 d \to \ldots) \\
&= \operatorname{colim}(\{\perp\} \to \{d\} \to \ldots) \\
&= \{d\},
\end{aligned}
$$

so $d' \in ab(\mathcal{D}[\![\text{while } B \text{ do } S \text{ od}]\!]d)$.

Case 2: Suppose $\mathcal{B}[\![B]\!]d = \mathsf{true}$, and $d' \neq \perp$. Then $\exists n \geq 1$ and a sequence $d = d_0, d_1, \ldots, d_n = d'$ such that

$d_{i+1} \in Op[\![S]\!]d_i$ (equivalently $\langle S, d_i\rangle \to^* d_{i+1}$) for $i = 0, \ldots, n-1$,
$\mathcal{B}[\![B]\!]d_i = \mathsf{true}$ for $i = 0, \ldots, n-1$,
and $\mathcal{B}[\![B]\!]d_n = \mathsf{false}$.

Now note that if $\mathcal{B}[\![B]\!]d_i = \mathsf{true}$, then for any $j \geq 1$

$$
\begin{aligned}
W_j d_i &= (\mathcal{D}[\![S]\!]; W_{j-1})d_i \\
&= \biguplus_{\substack{e \in \mathcal{D}[\![S]\!]d_i \\ e \neq \perp}} W_{j-1}e \uplus \{\perp | \perp \in \mathcal{D}[\![S]\!]d_i\},
\end{aligned}
$$

so given that $d_{i+1} \in Op[\![S]\!]d_i$ implies that $d_{i+1} \in \mathcal{D}[\![S]\!]d_i$ by the induction hypothesis, we have that

$$W_{j-1}d_{i+1} \subseteq W_j d_i, \text{ for } i < n.$$

Also, $\mathcal{B}[\![B]\!]d_n = \mathsf{false}$ implies that

$$W_j d_n = \{d_n\} \text{ for any } j > 0.$$

This, together with the chain $d = d_0, d_1, \ldots, d_n = d'$ defined above, give us

$$\{d'\} = \{d_n\} = W_1 d_n \subseteq W_2 d_{n-1} \subseteq \cdots \subseteq W_n d_1 \subseteq W_{n+1} d_0 = W_{n+1} d.$$

Now, if we look at the definition of the natural transformation η_j, for $j > 0$, we see that

$$\begin{aligned} d' \neq \perp,\ d' \in W_{n+1} \quad &\Rightarrow \quad d' \in W^* d \\ &\Leftrightarrow \quad d' \in \mathcal{D}[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d \\ &\Rightarrow \quad d' \in ab(\mathcal{D}[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d). \end{aligned}$$

Case 3: Suppose $\mathcal{B}[\![B]\!]d = \mathsf{true}$, and $d' = \perp$. Then there exists a sequence $d = d_0, d_1, \ldots$ such that

$$\begin{aligned} &d_{i+1} \in Op[\![S]\!]d_i \text{ for all } i, \\ &\text{and } \mathcal{B}[\![B]\!]d_i = \mathsf{true} \text{ for all } i. \end{aligned}$$

We wish to show that $\mathcal{D}[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d$ contains $\perp$, or (by unfolding the definition) that the diagram

$$W_0 d_0 \xrightarrow{\eta_0 d_0} W_1 d_0 \xrightarrow{\eta_1 d_0} W_2 d_0 \xrightarrow{\eta_2 d_0} \cdots$$

contains an infinite chain $\{\perp\} \longrightarrow \{\perp\} \longrightarrow \cdots$).

Noting that by the induction hypothesis, $d_{i+1} \in Op[\![S]\!]d_i$ for all i, it is a straightforward induction to show that, for any $n \geq 0$

$$\eta_n d_j : W_n d_j \longrightarrow W_{n+1} d_j \ni \langle \perp, \perp \rangle \text{ for all } j \geq 0.$$

$\mathcal{B}[\![B]\!]d_j = \mathsf{true}$ for all $j \geq 0$ implies

$$\eta_0 d_j : W_0 d_j \longrightarrow W_1 d_j \ni \langle \perp, \perp \rangle \text{ for all } j \geq 0$$

from the definition of η_0. For $n > 1$, we have that, for all $j \geq 0$

$$\begin{aligned} \eta_n d_j : W_n d_j \longrightarrow W_{n+1} d_j \quad &\ni \quad \biguplus_{x \in \mathcal{D}[\![S]\!]d_j} \eta_{n-1} x \\ (\text{since } d_{j+1} \in \mathcal{D}[\![S]\!]d_j) \quad &\ni \quad \eta_{n-1} d_{j+1} \\ (\text{by induction}) \quad &\ni \quad \langle \perp, \perp \rangle. \end{aligned}$$

Thus,

$$\begin{aligned} d' = \bot & \in & \operatorname{colim}(W_0 d_0 \xrightarrow{\eta_0 d_0} W_1 d_0 \xrightarrow{\eta_1 d_0} W_2 d_0 \xrightarrow{\eta_2 d_0} \cdots) \\ & = & \mathcal{D}[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d, \end{aligned}$$

and we have $d' \in ab(\mathcal{D}[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d)$. ∎

Lemma 4. For all d, $ab(\mathcal{D}[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d) \subseteq Op[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d$

Proof: Let $d' \in \mathcal{D}[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d$. Then

$$d' \in \operatorname{colim}(W_0 d \to W_1 d \to \cdots) \Rightarrow d' = \sqcup Q,$$

where Q is a chain through the diagram, as defined in Theorem 2. From the definition of the W_i and the η_i, we know that the only distinct elements of Q are $\bot$ and d'. Thus, $d' \in Q$, which implies $d' \in W_n d$ for some finite n. Let m be the least such index. Now, assuming $d' \neq \bot$, there is a sequence $d = d_0, d_1, \ldots, d_m = d'$ such that

$$\begin{aligned} & d_{i+1} \in \mathcal{D}[\![S]\!]d_i \text{ for } i = 0, \ldots, m-1, \\ & \mathcal{B}[\![B]\!]d_i = \mathsf{true} \text{ for } i = 0, \ldots, m-1, \\ & \text{and } \mathcal{B}[\![B]\!]d_m = \mathsf{false}. \end{aligned}$$

But this means that (by induction hypothesis)

$$\begin{aligned} & d_{i+1} \in Op[\![S]\!]d_i \text{ for } i = 0, \ldots, m-1, \\ & \mathcal{B}[\![B]\!]d_i = \mathsf{true} \text{ for } i = 0, \ldots, m-1, \\ & \text{and } \mathcal{B}[\![B]\!]d_m = \mathsf{false}. \end{aligned}$$

This we know is equivalent to $d' = d_m \in Op[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d$.

If $d' = \bot$, then there is an infinite sequence $d = d_0, d_1, \ldots$ such that

$$\begin{aligned} & d_{i+1} \in \mathcal{D}[\![S]\!]d_i \text{ for all } i, \\ & \text{and } \mathcal{B}[\![B]\!]d_i = \mathsf{true} \text{ for all } i. \end{aligned}$$

But this means that (by induction hypothesis)

$$\begin{aligned} & d_{i+1} \in Op[\![S]\!]d_i \text{ for all } i, \\ & \text{and } \mathcal{B}[\![B]\!]d_i = \mathsf{true} \text{ for all } i, \end{aligned}$$

which implies $\bot = d' \in Op[\![\mathsf{while}\ B\ \mathsf{do}\ S\ \mathsf{od}]\!]d$. ∎

Corollary 1 (Full-abstraction) For all statements S and S' we have

$$\mathcal{D}[\![S]\!] = \mathcal{D}[\![S']\!]$$

if and only if

$$Op[\![C[S]]\!] = Op[\![C[S']]\!] \text{ for all contexts } C[\cdot].$$

Proof: $\mathcal{D}[\![S]\!] = \mathcal{D}[\![S']\!]$ is equivalent to $Op[\![S]\!] = Op[\![S']\!]$ by Theorem 4. If we take $C[\cdot]$ to be the empty context, we see that $Op[\![C[S]]\!] = Op[\![C[S']]\!]$ implies that $Op[\![S]\!] = Op[\![S']\!]$; Theorem 3 gives us the reverse implication.

7 Conclusions

The result in section 6 is the main result of this paper. The main achievement is that the semantics generalizes the usual notion of continuous least fixed-point semantics. The failure of continuity, as required by Apt and Plotkin's result, is isolated to the abstraction function.

The framework we have developed is general enough to accommodate domains that are not 'flat', and even Lehmann's categorically generalized domains. We are developing a semantics for a language with infinite output streams, together with a full-abstraction result for this setting. Additional future work includes a detailed comparison of this approach with strictly domain theoretic approaches to the same problem.

References

[1] S. Abramsky. On semantic foundations for applicative multiprogramming. In J. Diaz, editor, *Proceedings of the Tenth International Conference On Automata, Languages And Programming*, pages 1–14, New York, 1983. Springer-Verlag.

[2] K. R. Apt and E.-R. Olderog. Proof rules and transformations dealing with fairness. *Sci. Comput. Prog.*, 3:65–100, 1983.

[3] K. R. Apt and G. D. Plotkin. Countable nondeterminism and random assignment. *Journal Of The ACM*, 33(4):724–767, 1986.

[4] T. Coquand. Categories of embeddings. In *Proceedings of the Third IEEE Symposium on Logic In Computer Science*, 1988.

[5] N. Francez. *Fairness.* Springer-Verlag, 1986.

[6] D. Lehmann. *Categories for Fixed-point Semantics.* PhD thesis, University of Warwick, 1976.

[7] P. Panangaden. Abstract interpretation and indeterminacy. In *Proceedings of the 1984 CMU Seminar on Concurrency*, pages 497–511, 1985. LNCS 197.

[8] P. Panangaden and V. Shanbhogue. On the expressive power of indeterminate primitives. Technical Report 87-891, Cornell University, Computer Science Department, November 1987.

[9] P. Panangaden and E. W. Stark. Computations, residuals and the power of indeterminacy. In Timo Lepisto and Arto Salomaa, editors, *Proceedings of the Fifteenth ICALP*, pages 439–454. Springer-Verlag, 1988. Lecture Notes in Computer Science 317.

[10] G. D. Plotkin. Lecture notes on domain theory. The Pisa Notes.

[11] G. D. Plotkin. A powerdomain construction. *SIAM Journal of Computing*, 5(3):452–487, 1976.

[12] M. B. Smyth. Powerdomains. *Journal of Computer and System Sciences*, 16:23–36, 1978.