Countable Nondeterminism and Random Assignment

K. R. APT

Laboratorie Informatique Theorique et Programmation, Université Paris 7, Paris, France

AND

G. D. PLOTKIN

University of Edinburgh, Edinburgh, Scotland

Abstract. Four semantics for a small programming language involving unbounded (but countable) nondeterminism are provided. These comprise an operational semantics, two state transformation semantics based on the Egli–Milner and Smyth orders, respectively, and a weakest precondition semantics. Their equivalence is proved. A Hoare-like proof system for total correctness is also introduced and its soundness and completeness in an appropriate sense are shown. Finally, the recursion theoretic complexity of the notions introduced is studied. Admission of countable nondeterminism results in a lack of continuity of various semantic functions, and this is shown to be necessary for any semantics satisfying appropriate conditions. In proofs of total correctness, one resorts to the use of (countable) ordinals, and it is shown that all recursive ordinals are needed.

Categories and Subject Descriptors: D.2.4 [**Software Engineering**]: Program Verification—*correctness proofs*; D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*semantics*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and Verifying and Reasoning about Programs; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages; F.3.3 [**Logics and Meanings of Programs**]: Studies of Program Constructs—*control primitives*

General Terms: Languages, Theory, Verification

Additional Key Words and Phrases: Continuity, fairness, nondeterminism, powerdomains, predicate transformers

1. *Introduction*

One of the natural assumptions concerning the execution of a nondeterministic or parallel program is that of fairness. In its simplest form it states that no process is forever denied its turn for execution. The assumption of fairness implies unbounded nondeterminism. To see this, consider the well-known program

$b :=$ **true**; $x := 0$;
do $b \rightarrow x := x + 1 \; \square \; b \rightarrow b :=$ **false od**

(see Dijkstra [16, p. 76]), which always terminates, under the assumption of fairness, and assigns to x an arbitrary natural number depending on the sequence of

This work was supported in part by a grant from the Science Research Council.

Authors' addresses: K. R. Apt, LITP, Université Paris 7, 2 Place Jussieu, 75221 Paris, France; G. D. Plotkin, Department of Computer Science, University of Edinburgh, Edinburgh EH9 3JZ Scotland.

execution steps. What is more, every nondeterministic program of this kind can be translated into an appropriate unbounded nondeterministic program using the random assignment command $x := ?$ which sets x to an arbitrary integer [5, 29]. This close relation between fairness and unbounded (but countable) nondeterminism motivates us to a thorough study of the latter. As is well known, unbounded nondeterminism results in a lack of continuity of various semantic functions. For example, in Dijkstra [16, chap. 9], one can find an argument showing that admitting unbounded nondeterminism results in a noncontinuity of the weakest precondition semantics. On the other hand, Boom [11] realized that this weakest precondition semantics still can be straightforwardly defined by considering least fixed points of monotone, but noncontinuous functions. Both Broy et al [12] and Back [8] gave semantics for unbounded nondeterminism, employing variants of the discrete powerdomains in [30]. The former paper used least fixed points, but the latter only used the first ω iterates (and the subsequent difficulties motivated other, continuous semantics—see also Back [9]). Similar issues are addressed in Park [28, 29] where the assumption of fair merging is also analyzed.

In other papers the issue of complexity of these properties is raised. In particular Chandra [13] has shown that the halting problem for programs admitting unbounded nondeterminism, being complete Π^1_1, is of higher complexity than truth in the standard model of natural numbers, being Δ^1_1. Similar results concerning various assumptions of fairness and inevitability about simple nondeterministic programs were proved in Emerson and Clarke [17].

In the present paper we try to consider all these issues together, concentrating on a simple programming language with atomic commands allowing countable nondeterminism (such as random assignment). In Section 2 we discuss discrete powerdomains and their associated state transformation functions considering both the Egli–Milner ordering and the Smyth ordering. The section concludes with a systematic presentation of predicate transformers that adapts Dijkstra's healthiness conditions to the present framework and shows the isomorphism with state transformations based on Smyth powerdomains (in analogy with Plotkin [31]). In Section 3 we present an operational semantics, two state transformation semantics, and a predicate transformer semantics. The relationships between all four are shown. The section concludes with an analysis of the reasons why continuity fails in one way or another in the various published approaches to the semantics of countable nondeterminism. It is shown that no continuous least fixed-point semantics can exist satisfying a certain full abstraction property. The technique also applies to the work of De Bakker and Zucker [15]. In Section 4 we consider a Hoare-style logic for total correctness and present soundness and relative completeness results; this involves the use of countable ordinals in the assertions. In Section 5 we incorporate Chandra's ideas into our framework discussing the halting problem and the definable state transformation functions and predicate transformers. The section concludes by discussing issues related to the recursion theoretic complexity of the assertion language and by showing that the countable ordinals needed in proofs are the recursive ones (see [32]). An extended abstract covering most of the material in this paper appeared in [6].

What we have shown here is that unbounded nondeterminism admits a simple and natural characterization that can be studied by generalizing techniques used for the case of deterministic or bounded nondeterministic programs.

We believe the present work can be extended to cover some other constructs omitted in our analysis such as **or** commands, Dijkstra's guarded commands, or recursive procedures. For example, the proof system we consider is a simple

refinement of the corresponding system for total correctness of **while** programs and an appropriate system covering the case of recursion should be a similar refinement of a system dealing with the total correctness of recursive procedures (e.g., [2]). In principle, our paper also provides a framework for studying fairness via translation into a language for countable nondeterminism. A proof-theoretic approach to the problem of total correctness of fair nondeterministic programs based on this idea has been recently worked out in [5].

Finally, note that the present paper considers only countable structures, restricting further in the last section to the case of arithmetic. There are two possible directions of generalization. One would be to consider structures of any cardinality, but to allow only countable nondeterminism; one might have two-sorted structures, one sort being the natural numbers, and allow $x := ?$ only for natural number variables. This would be the natural generalization when considering the origin of the concerns of the paper in the problems associated with fairness. Another direction would be to allow structures of any cardinality and random assignment over the domain of the structure. This is a very natural mathematical generalization, regarding random assignment as the computational analog of the universal quantifier, as do Harel and Kozen [21], for example. One would expect direct applications of the work on inductive relations by Moschovakis [1, 26].

2. *Powerdomains and Predicate Transformers*

In this section we begin by collecting some general information on fixed points. Then we give the basic definitions and properties of discrete Egli–Milner and Smyth powerdomains, suitably adapted from those in Plotkin [30] and Smyth [33] to handle countable nondeterminism, and show, following the ideas in Plotkin [31], how they connect up with the discrete Smyth powerdomain.

Definition 2.1. Let P be a partial order and let A be a subset of P. Then A is *directed* if every finite subset of A has an upper bound in A; it is *countably directed* if every countable subset of A has an upper bound in A. The partial order P is a *complete partial order* (*cpo*) if every directed subset, A, of P has a least upper bound (lub), denoted by $\sqcup A$, and if there is a least element in P, denoted by $\perp$. A subset of P is *eventually constant* if it contains its own least upper bound.

For example, for any set, X, there is the *flat* cpo $X_\perp$ which is the set $X \cup \{\perp\}$ ordered by $x \sqsubseteq y$ iff $x = \perp$ or $x = y$.

Definition 2.2. Let P, Q be partial orders and let $f\colon P \to Q$ be a monotone function. Then f is *continuous* if whenever $A \subseteq P$ is a directed subset with a lub, then $f(A)$ has a lub, namely, $f(\sqcup A)$ (i.e., f preserves lubs of directed subsets); f is *strict* whenever it preserves the least element; f is ω_1-*continuous* if it preserves lubs of countably directed sets (recall that ω_1 is the first uncountable ordinal).

Definition 2.3. Let P, Q be partial orders, X a countable set. Then $P \times Q$ is the Cartesian product of P and Q ordered coordinatewise; $X \to P$ is the partial order of all functions from X to P ordered pointwise; $P \to_m Q$ is the partial order of all monotone functions from P to Q ordered pointwise.

FACT 2.1. *If P is a cpo, then so is $X \to P$; if P and Q are cpos, so are $P \times Q$ and $P \to_m Q$. Least upper bounds are calculated pointwise or coordinatewise, as the case may be.*

2.1 FIXED POINTS. For any partial order P, any monotone $f\colon P \to P$ and any ordinal λ, define f^λ by

$$f^\lambda = f\left(\bigsqcup_{\kappa<\lambda} f^\kappa\right).$$

Of course f^λ need not exist since $\bigsqcup_{k<\lambda} f^\lambda$ need not exist. (Note that $f^0 = f(\bot)$ when the least element $\bot$ of P exists.) If f^λ does not exist, then for any $\lambda' > \lambda$, $f^{\lambda'}$ does not exist either. Note that f^λ is monotonic in λ. We say that $\langle f^\lambda\rangle_\lambda$ *stabilizes* by κ if, whenever $\lambda > \kappa$, then $f^\lambda = f^\kappa$; the *closure* ordinal of f is the least ordinal κ by which the sequence stabilizes, and then f^κ is the least (pre-) fixed point of f (since $f(f^\kappa) = f^{\kappa+1} = f^\kappa$ and $f(a) \sqsubseteq a$ implies $f^\lambda \sqsubseteq a$ for all λ). If P is a cpo, then of course f^λ always exists and $\langle f^\lambda\rangle_\lambda$ stablizes. If additionally f is continuous, then it has closure ordinal $\leq\omega$, and if f is ω_1-continuous it has closure ordinal $\leq\omega_1$.

In Section 3 we need the following two well-known facts.

FACT 2.2. *Suppose P, Q are cpos and $g\colon Q \times P \to Q$ is monotone. Define $f\colon P \to Q$ by $f(a) = \mu b.g(b, a)$. Then $f = \mu h \in P \to_m Q.\ [\lambda a \in P.g(h(a), a))]$.*

In stating Fact 2.2 we use the *μ-notation* where, for any partial order P, variable a ranging over P and expression e possibly containing a and denoting an element of P, the expression $\mu a.e$ is the least element, a, of P such that $e \sqsubseteq a$ (if such an element exists).

FACT 2.3. TRANSFER LEMMA. *Suppose P, Q are partial orders, and $f\colon P \to P$, $g\colon Q \to Q$ are monotone functions, and $h\colon P \to Q$ is a strict and continuous function such that the following diagram commutes:*

$$\begin{array}{ccc} P & \xrightarrow{\ f\ } & P \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h}. \\ Q & \xrightarrow{\ g\ } & Q \end{array}$$

Then if f^λ exists, so does q^λ and indeed $g^\lambda = h(f^\lambda)$. In particular, if $\mu x.f(x)$ exists (being an f^λ), then so does $\mu x.g(x)$ and $\mu x.g(x) = h(\mu x.f(x))$.

PROOF. To see that g^λ exists when f^λ does, being $h(f^\lambda)$, let us calculate (by ordinal induction) that

$$\begin{aligned} h(f^\lambda) &= h\left(f\left(\bigsqcup_{\kappa<\lambda} f^\kappa\right)\right) && \text{(by definition of } f^\lambda) \\ &= g\left(h\left(\bigsqcup_{\kappa<\lambda} f^\kappa\right)\right) && \text{(by the diagram)} \\ &= g\left(\bigsqcup_{\kappa<\lambda} h(f^\kappa)\right) && \text{(since } h \text{ is strict and continuous)} \\ &= g\left(\bigsqcup_{\kappa<\lambda} g^\kappa\right) && \text{(by induction hypothesis)} \\ &= g^\lambda && \text{(by definition).} \end{aligned}$$

If $\mu x.fx$ is f^λ for some λ, then we have

$$h(f^\lambda) = h(f^{\lambda+1}) = h(f(f^\lambda)) = g(h(f^\lambda)).$$

But $h(f^\lambda) = g^\lambda$, by the above. So $\langle g^{\lambda'}\rangle_{\lambda'}$ stabilizes by λ and we see that $\mu x.gx$, the least fixed point of g, is $h(f^\lambda)$ as required. □

2.2 DISCRETE POWERDOMAINS. A meaning of a countably nondeterministic command is to be a function from the set of states to an appropriate partial order. A typical element of such a partial order will be the set of outcomes of all computations of the command starting in the same initial state. Depending on how we handle a possibly nonterminating computation, we arrive at two possible partial orders, both of which have already been considered in the case of bounded nondeterminism.

We explore Egli–Milner and Smyth powerdomains of flat cpos, $X_\perp$, with enough subsets to handle countable nondeterminism. To avoid some ticklish problems, we restrict X to being countable. Note that, even so, the Smyth powerdomain as defined here is *not* a cpo; we do not understand what significance, if any, this has for a possible more general theory of powerdomains for countable nondeterminism.

Egli–Milner Order. Let $\mathscr{E}(X_\perp)$ be the set of nonempty subsets of $X_\perp$ ordered by

$$A \sqsubseteq B \quad \text{iff} \quad (\forall a \in A.\ \exists b \in B.a \sqsubseteq b) \wedge (\forall b \in B.\ \exists a \in A.a \sqsubseteq b)$$

(which is the same as $A = B$ (if $\perp \notin A$) or as $A - \{\perp\} \subseteq B$ (if $\perp \in A$)).

FACT 2.4. *$\mathscr{E}(X_\perp)$ is a cpo with least element $\{\perp\}$; every countably directed subset is eventually constant; it is closed under arbitrary nonempty unions.*

PROOF. Evidently $\{\perp\}$ is the least element. If $\mathscr{F} \subseteq \mathscr{E}(X_\perp)$ is a directed family, then $\sqcup\mathscr{F} = [\cup\mathscr{F} - \{\perp\}] \cup \{\perp \mid \forall A \in \mathscr{F}.\perp \in A\}$. In case $\mathscr{F}$ is countably directed, it is easy to see that $\sqcup\mathscr{F}$—a countable set—is in $\mathscr{F}$. Closure under arbitrary nonempty unions of subsets of $\mathscr{E}(X_\perp)$ is obvious. □

Note. This is where things go wrong if X is not countable; one cannot restrict elements of $\mathscr{E}(X_\perp)$ to be countable if one wants $\mathscr{E}(X_\perp)$ to be a cpo since $\sqcup\mathscr{F}$, as defined above, need not be countable. It is a question of how many subsets one wants to allow versus how strong the required completeness properties of $\mathscr{E}(X_\perp)$ are to be.

State Transformations. A meaning of a command will thus be a function from X to $\mathscr{E}(X_\perp)$. Let $\mathrm{ET}_{X,Y}$ stand for $X \to \mathscr{E}(Y_\perp)$. We call the elements of $\mathrm{ET}_{X,Y}$ *Egli–Milner state transformers.* They are ordered pointwise.

The following functions will be needed when meaning is assigned to the composite commands considered in the next section. For example, extension will be used for the composition construct on commands.

Singleton. $\{\cdot\} \in \mathrm{ET}_{X,X}$.
Union. $\cup: \mathscr{E}(X_\perp)^2 \to \mathscr{E}(X_\perp)$. It is easily checked to be continuous.
Extension. For $f \in \mathrm{ET}_{X,Y}$ define $f^\dagger: \mathscr{E}(X_\perp) \to \mathscr{E}(Y_\perp)$ by

$$f^\dagger(A) = \cup f(A - \{\perp\}) \cup \{\perp \mid \perp \in A\}.$$

A function $g: \mathscr{E}(X_\perp) \to \mathscr{E}(Y_\perp)$ is *completely linear* if it preserves existing unions of arbitrary families of elements of $\mathscr{E}(X_\perp)$; that is, if $\mathscr{F}$ is a nonempty subset of $\mathscr{E}(X_\perp)$, then $g(\cup\mathscr{F}) = \cup g(\mathscr{F})$.

FACT 2.5. *Every $f^\dagger$ is continuous and completely linear. However, $f^\dagger$ is* not *continuous as a function of f, although it is monotonic.*

PROOF. To see $f^\dagger$ is monotonic, suppose $A \sqsubseteq B$; if $\perp \notin A$, then $A = B$ and so $f^\dagger(A) \sqsubseteq f^\dagger(B)$; otherwise, $\perp \in f^\dagger(A)$, and $A - \{\perp\} \subseteq B$, and

$$f^\dagger(A) - \{\perp\} = \bigcup f(A - \{\perp\}) - \{\perp\} \subseteq f^\dagger(B).$$

For continuity, we first note that complete linearity is trivial. Now if $\mathscr{F}$ is a directed subset of $\mathscr{E}(X_\perp)$, then there are two cases. If some A in $\mathscr{F}$ does not contain $\perp$, then $f^\dagger(\bigsqcup \mathscr{F}) = f^\dagger(A) = \bigsqcup_{B \in \mathscr{F}} f^\dagger(B)$ (by monotonicity). Otherwise

$$\begin{aligned} f^\dagger(\bigsqcup \mathscr{F}) = f^\dagger(\bigcup \mathscr{F}) &= \bigcup_{A \in \mathscr{F}} f^\dagger(A) \quad \text{(by complete linearity)} \\ &= \bigsqcup_{A \in \mathscr{F}} f^\dagger(A) \quad \text{(as } \perp \text{ is in every } A \text{ in } \mathscr{F}\text{).} \end{aligned}$$

Monotonicity of $f^\dagger$ in f is obvious. To see how continuity fails, take

$$f^{(m)}\colon N \to \mathscr{E}(\{\mathsf{T}\}_\perp),$$

where:

$$f^{(m)}(n) = \begin{cases} \{\mathsf{T}\} & (n < m), \\ \{\perp\} & (n \geq m). \end{cases}$$

Then $\{f^{(m)} \mid m \in \omega\}$ is directed and $(\bigsqcup_m f^{(m)})(n) = \{\mathsf{T}\}$. But then $(\bigsqcup_m f^{(m)})^\dagger(N) = \bigcup_{n \in N} (\bigsqcup_m f^{(m)})(n) = \{\mathsf{T}\}$, whereas $(\bigsqcup_m f^{(m)\dagger})(N) = \bigsqcup_m f^{(m)\dagger}(N) = \bigsqcup_m \bigcup_{n \in N} f^{(m)}(n) = \{\perp, \mathsf{T}\}$. (Essentially, this example appears in [14, p. 269].) □

LEMMA 2.1. *Suppose $f \in ET_{X,Y}$ and $g \in ET_{Y,Z}$. Then $(g^\dagger \circ f)^\dagger = g^\dagger \circ f^\dagger$. Also, for $\{\cdot\} : ET_{X,X}$, we have $\{\cdot\}^\dagger = id_{\mathscr{E}(X_\perp)}$, the identity function on $\mathscr{E}(X_\perp)$.*

PROOF. Take A in $\mathscr{E}(X_\perp)$. First consider the case where $\perp \notin A$. Then

$$\begin{aligned} (g^\dagger \circ f)^\dagger(A) &= \bigcup_{a \in A} g^\dagger \circ f(a) \\ &= g^\dagger\left(\bigcup_{a \in A} f(a)\right) \quad (g^\dagger \text{ is completely linear}) \\ &= g^\dagger(f^\dagger(A)). \end{aligned}$$

The other case is $A \cup \{\perp\}$ and then

$$\begin{aligned} (g^\dagger \circ f)^\dagger(A \cup \{\perp\}) &= (g^\dagger \circ f)^\dagger(A) \cup (g^\dagger \circ f)^\dagger(\perp) \\ &= g^\dagger(f^\dagger(A)) \cup \{\perp\} \\ &= g^\dagger(f^\dagger(A) \cup \{\perp\}) \\ &= g^\dagger(f^\dagger(A \cup \{\perp\})). \end{aligned}$$

It is obvious that $\{\cdot\}^\dagger = \mathrm{id}_{\mathscr{E}(X_\perp)}$. □

We can now define the composition operation ";", which will stand for the meaning of the composition of commands.

Composition. For $f \in \mathrm{ET}_{X,Y}$ and $g \in \mathrm{ET}_{Y,Z}$, define $f\,;\, g \in \mathrm{ET}_{X,Z}$ by putting $f\,;\, g = g^\dagger \circ f$.

FACT 2.6. *The composition $f\,;\, g$ is continuous in f and monotone, but not continuous, in g. Also it is associative with units the singleton functions (that is, we get a category).*

PROOF. Monotonicity is an easy calculation as is continuity in f (use the fact that $g^\dagger$ is continuous). To see that continuity in g fails, adapt the example in Fact 2.5 with $Y = N$, $X = Z = \{\top\}$, *and* f: $\top \mapsto N$. For associativity we can calculate, using Lemma 2.1,

$$f;(g;h) = (g;h)^\dagger \circ f = (h^\dagger \circ g)^\dagger \circ f = h^\dagger \circ g^\dagger \circ f$$
$$= h^\dagger \circ (f;g) = (f;g);h.$$

Finally note that $\{\cdot\};f = \lambda a \in X.f^\dagger(\{a\}) = \lambda a \in X. \cup \{f(a)\} = f$ and $f;\{\} = \{\cdot\}^\dagger \circ f = \mathrm{id} \circ f = f$. □

Note. It is the lack of continuity of $f;g$ in g that will force us (in the semantics of **while** commands) to consider least fixed points of noncontinuous functionals.

In the relational approach to nondeterminism advocated, for example, by Park [28] and Broy et al. [12], one handles nontermination by a termination set, which is the collection of all input states guaranteeing termination. It is natural then to define the collection $\mathrm{ER}_{X,Y}$ of Egli–Milner relations as

$$\{\langle R, T\rangle \in \mathscr{P}(X \times Y) \times \mathscr{P}(X) \mid \forall x \in T.R(x) \neq \varnothing\},$$

and turn it into a partial order by defining

$$\langle R, T\rangle \sqsubseteq \langle R', T'\rangle \quad \text{iff} \quad R \subseteq R' \quad \text{and} \quad T \subseteq T'$$
$$\text{and} \quad \forall x \in T.R'(x) \subseteq R(x).$$

But this is isomorphic to our approach, as we may define rel: ET $\to$ ER by putting $\mathrm{rel}(m) = \langle R_m, T_m\rangle$ where

$$xR_m y \equiv y \in m(x),$$
$$x \in T_m \equiv \bot \notin m(x).$$

Then rel is easily seen to be an isomorphism of partial orders with inverse st: ER $\to$ ET where

$$\mathrm{st}(R, T)(x) = R(x) \cup \{\bot \mid x \notin T\}.$$

Which approach to adopt is therefore, a matter of convenience or taste.

Smyth Order. Let $\mathscr{S}(X_\bot)$ be

$$\{A \subseteq X \mid A \neq \varnothing\} \cup \{X_\bot\},$$

ordered by the superset ordering

$$A \sqsubseteq B \quad \text{iff} \quad A \supseteq B.$$

The idea behind the choice of the elements of $\mathscr{S}(X_\bot)$ is first that all nonempty subsets are feasible as the results of computations, since we have countable nondeterminism (neglecting computability considerations) and the empty subset is not possible with the language we consider in the next section, as nontermination is recorded by $\bot$. Furthermore, all sets containing $\bot$ are identified, since no predicate (=postcondition) must hold on all the results if $\bot$ is possible, and so all are equally bad. The choice of superset as the ordering has a similar motivation, since any predicate holding on all results must also hold for any subset of the results. For more discussion, see [31], [33], and [35].

FACT 2.7. *$\mathscr{S}(X_\bot)$ has least element $X_\bot$ but need* not *be a cpo (although if a subset $\mathscr{F}$ has an upper bound, its lub exists and is $\bigcap\mathscr{F}$); every countably directed subset is eventually constant; it is closed under arbitrary nonempty unions.*

PROOF. Clearly $X_\perp$ is the least element. To see that $\mathscr{S}(N_\perp)$ is not a cpo, consider $N \sqsubseteq N - \{0\} \sqsubseteq N - \{0, 1\} \sqsubseteq \cdots$. If $\mathscr{F} \subseteq \mathscr{S}(X_\perp)$ has an upper bound A, then $\cap \mathscr{F} \supseteq A$ and so is nonempty and therefore the lub. If $\mathscr{F}$ is countably directed, then $\cap \mathscr{F}$ is nonempty and in $\mathscr{F}$. Closure under arbitrary nonempty unions is obvious. □

Note. The greatest lower bound of a subset always exists being its union.

Had we included the empty set in $\mathscr{S}(X_\perp)$, we would have obtained a cpo and thereby avoided the resulting difficulties (like the question of the existence of the Smyth denotational semantics, $\mathscr{D}_\mathscr{S}$ considered below). Some alterations in the definitions of the Smyth relations and predicate transformers given below would also be needed to retain results like Theorem 2.1—for example, one would have to drop the law of the excluded miracle. The empty set was included in the work of Milne and Milner [25], but there it had a natural interpretation as deadlock. As far as we can see, the decision seems to be a matter of taste.

State Transformations. The "Smyth state transformers" from X to Y are all functions m: $X \to \mathscr{S}(Y_\perp)$. They are ordered pointwise. We call this collection $\mathrm{ST}_{X,Y}$.

The following functions are needed in the next section.

Singleton. $\{\cdot\} \in \mathrm{ST}_{X,X}$.

Union. $\cup$: $\mathscr{S}(X_\perp)^2 \to \mathscr{S}(X_\perp)$. It is continuous, for, if $\mathscr{F}$, $\mathscr{G}$ are directed sets with lubs, then so is $\{A \cup B \mid A \in \mathscr{F}, B \in \mathscr{G}\}$ and $\sqcup\mathscr{F} \cup \sqcup\mathscr{G} = \sqcup\{A \cup B \mid A \in \mathscr{F}, B \in \mathscr{G}\}$.

Extension. For $f \in \mathrm{ST}_{X,Y}$ define $f^\dagger$: $\mathscr{S}(X_\perp) \to \mathscr{S}(Y_\perp)$ by

$$f^\dagger(A) = \begin{cases} \cup f(A) & (\perp \notin A), \\ Y_\perp & (\perp \in A). \end{cases}$$

FACT 2.8. *Every $f^\dagger$ is monotone, but* not *necessarily continuous, and $f^\dagger$ is completely linear (i.e., it preserves existing unions of arbitrary families of elements of $\mathscr{S}(X_\perp)$); function extension, $(\cdot)^\dagger$, is monotone but not necessarily continuous.*

PROOF. Monotonicity is clear. To see the general failure of continuity take $X = N$, $Y = \{tt, ff\}$ and define

$$f(a) = \begin{cases} \{tt\} & (a \neq 0), \\ \{ff\} & (a = 0), \end{cases}$$

and take $A^{(m)} = \{a \in N \mid a \geq m\} \cup \{0\}$. Then $\sqcup A^{(m)}$ exists, being $\{0\}$, and $f^\dagger(\sqcup A^{(m)}) = f^\dagger(\{0\}) = \{ff\}$, but $\sqcup f^\dagger(A^{(m)})$ *is* $\sqcup \{\{tt, ff\}\} = \{tt, ff\} \neq \{ff\}$. Complete linearity is clear. The monotonicity of function extension and the failure of continuity is much as before. □

LEMMA 2.2. $(g^\dagger \circ f)^\dagger = g^\dagger \circ f^\dagger$ *and* $\{\cdot\}^\dagger = id$.

PROOF. The proof is like that of Lemma 2.1. □

Composition. For $f \in \mathrm{ST}_{X,Y}$ and $g \in \mathrm{ST}_{Y,Z}$, define $f\,;g \in \mathrm{ST}_{X,Z}$ by $f\,;g = g^\dagger \circ f$.

FACT 2.9. *The composition $f\,;g$ is monotone in each argument but need not be continuous in either. Also it is associative with the singleton as unit.*

PROOF. Omitted. □

The relational approach is perhaps not quite so natural as in the Egli–Milner case. Define the set $SR_{X,Y}$ of the Smyth relations as

$$\{\langle R, T\rangle \in \mathcal{P}(X \times Y) \times \mathcal{P}(X) \mid (\forall x \in X.R(x) \neq \emptyset) \wedge (\forall x \notin T.R(x) = Y)\},$$

and partially order it by

$$\langle R, T\rangle \sqsubseteq \langle R', T'\rangle \quad \text{iff} \quad R \supseteq R' \quad \text{and} \quad T \subseteq T'.$$

Then the functions $ST \to^{rel} SR \to^{st} ST$ are defined as before and rel is an isomorphism of partial orders with inverse st.

From $\mathcal{E}(X_\perp)$ to $\mathcal{S}(X_\perp)$. Define $e_X: \mathcal{E}(X_\perp) \to \mathcal{S}(X_\perp)$ by

$$e_X(A) = \begin{cases} A & (\perp \notin A), \\ X_\perp & (\perp \in A). \end{cases}$$

(That is, $e_X(A) = \{b \in X_\perp \mid \exists a \in A.a \sqsubseteq b\}$.)

Then e_X is strict, continuous, onto, and completely linear (as is easily verified). It is important that e_X be continuous, since this is why we can live with the fact that $\mathcal{S}(X_\perp)$ is not a cpo—enough directed sets, for our purpose, will have limits since they will be images under e_X of directed sets in $\mathcal{E}(X_\perp)$.

We apply this observation in the next section where we use the strict continuous surjection

$$ET_{X,Y} \xrightarrow{\lambda m.e_X \circ m} ST_{X,Y},$$

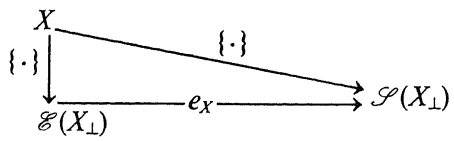
as the function h mentioned in the Transfer Lemma. In terms of the relational approach, we obtain a strict continuous surjection

$$\text{sr}: ER_{X,Y} \to ST_{X,Y},$$

where $\text{sr} = \text{rel}_{X,Y} \circ (\lambda m.e_Y \circ m) \circ \text{st}_{X,Y}$; it turns out that

$$\text{sr}(R, T) = \langle R \cup (T' \times Y), T\rangle \quad \text{where} \quad T' = X - T.$$

FACT 2.10. *The following diagram commutes:*



FACT 2.11. *For any $f \in ET_{X,Y}$ and $g \in ET_{Y,Z}$, $e_Z \circ (f; g) = (e_Y \circ f); (e_Z \circ g)$.*

PROOF. First, we show that $(e_Z \circ g)^\dagger \circ e_Y = e_Z \circ g^\dagger$. By strictness and complete linearity, it is enough to check this for arguments of the form $\{b\}$ with $b \in Y$ and we calculate

$$(e_Z \circ g)^\dagger \circ e_Y(\{b\}) = e_Z \circ g(b) = e_Z(g^\dagger(\{b\})).$$

Now we have $(e_Y \circ f); (e_Z \circ g) = (e_Z \circ g)^\dagger \circ e_Y \circ f = (e_Z \circ g^\dagger) \circ f = e_Z \circ (f; g)$. □

2.3 SMYTH POWERDOMAINS AND PREDICATE TRANSFORMERS. A predicate transformer from X to Y is any map $p: \mathcal{P}(Y) \to \mathcal{P}(X)$ from the powerset of Y to the powerset of X such that

(1) *Law of Excluded Miracle*: $p(\emptyset) = \emptyset$.
(2) *Countable Multiplicativity*: $p(\bigcap_{i\in\omega} B_i) = \bigcap_{i\in\omega} p(B_i)$.

These are the appropriate healthiness conditions. The usual healthiness conditions imply them for (2) follows from the Stability Lemma 2.4, which is shown in [31]

to hold for the usual predicate transformers. But noncontinuous transformers are allowed and must be essentially as pointed out by Dijkstra in [16, chap. 9]. That they are exactly the right conditions will appear from the isomorphism with the Smyth state transformers, which we shall show, and from the role they play in the various semantics.

We take $\mathrm{PT}_{X,Y}$ to be the set of predicate transformers from X to Y (dropping here and in $\mathrm{ST}_{X,Y}$ the subscripts when they can be understood from the context) and ordered pointwise as follows:

$$p \sqsubseteq q \quad \text{iff} \quad \forall B \subseteq Y. p(B) \subseteq q(B).$$

Now for any $m \in \mathrm{ST}_{X,Y}$ define, for $B \subseteq Y$,

$$\mathrm{wp}(m, B) = \{a \in X \mid m(a) \subseteq B\},$$

so wp is the weakest precondition function.

Note. If $\perp$ is in $m(a)$, then a is not in $\mathrm{wp}(m, B)$.

LEMMA 2.3. *The function* $wp(m, \cdot)$ *is a predicate transformer and* $wp(m, \cdot)$ *is monotone in* m.

PROOF. It is a straightforward verification that $\mathrm{wp}(m, \cdot)$ is in PT. Suppose $m \sqsubseteq m'$ and $a \in \mathrm{wp}(m, B)$. Then $m'(a) \subseteq m(a) \subseteq B$ (note reversal of order!), demonstrating the required monotonicity. □

So now we have a monotone ω: ST $\rightarrow$ PT where

$$\omega(m)(B) = \mathrm{wp}(m, B)$$

and we even show it is an isomorphism.

LEMMA 2.4. (STABILITY). *Suppose we have* p *in* $PT_{X,Y}$ *and* a *in* X *with* $a \in p(Y)$. *Then there is a nomempty set,* $min(p, a)$, *such that*

$$\forall B \subseteq Y. (a \in p(B) \leftrightarrow min(p, a) \subseteq B).$$

PROOF. Let $b_0, b_1, \ldots,$ be an enumeration of those elements b of Y such that there is a set $B \subseteq Y$ with $b \notin B$ and $a \in p(B)$. (If there are none, we can take $\min(p, a) = Y$.) Let $B_0, B_1, \ldots$ be an enumeration of subsets of Y where $b_i \notin B_i$ and $a \in p(B_i)$. Put $M = \bigcap B_i$. We now show that we can take $\min(p, a)$ to be M. Since p is a predicate transformer, it follows from condition (2) and $a \in p(B_i)$ that $a \in p(M)$ and so by condition (1) that M is nonempty. So if $M \subseteq B$, then $a \in p(B)$. Conversely, suppose that $a \in p(B)$, but that, for the sake of contradiction, $M \not\subseteq B$. Then there is some b in M but not B, and so b must be a b_i and then $b \notin B_i \supseteq \bigcap B_i = M$, contradicting $b \in M$. □

Now we can define ω^{-1}: $\mathrm{PT}_{X,Y} \rightarrow \mathrm{ST}_{X,Y}$ by

$$\omega^{-1}(p)(a) = \begin{cases} \min(p, a) & (a \in p(Y)), \\ Y_\perp & (a \notin p(Y)). \end{cases}$$

Note that this uses the fact that $\min(p, a)$ is nonempty.

LEMMA 2.5. *The function* ω^{-1} *is monotonic.*

PROOF. Suppose $p \sqsubseteq q$ and take a in X. If $\omega^{-1}(p)(a) = \perp$, we have $\omega^{-1}(p)(a) \sqsubseteq \omega^{-1}(q)(a)$. Otherwise, $a \in p(Y) \subseteq q(Y)$, and so $\omega^{-1}(p)(a) = \min(p, a)$ and $\omega^{-1}(q)(a) = \min(q, a)$. Since $a \in p(\min((p, a)) \subseteq q(\min(p, a))$, we have $\min(p, a) \supseteq \min(q, a)$ (by the Stability Lemma applied to q), which means that $\omega^{-1}(p)(a) \sqsubseteq \omega^{-1}(q)(a)$. □

THEOREM 2.1 (ISOMORPHISM). *The function $\omega : ST \cong PT$ is an isomorphism of partial orders with inverse ω^{-1}.*

PROOF. We already know from Lemmas 2.3 and 2.5 that ω and ω^{-1} are monotone; it remains to show they are inverses. First we show $\omega^{-1} \circ \omega = \mathrm{id}_{ST}$. Take m in ST and a in X. We have $m(a) = \perp$ iff $a \notin \mathrm{wp}(m, Y) = \omega(m)(Y)$. So if $m(a) = \perp$, we have $[\omega^{-1} \circ \omega'(m)](a) = \perp = m(a)$.

Otherwise, $[\omega^{-1}(\omega(m))](a) = \min(\omega(m), a)$. Now $a \in \omega(m)(B) \leftrightarrow a \in \mathrm{wp}(m, B) \leftrightarrow m(a) \subseteq B$; so by the Stability Lemma, when $m(a) \neq \perp$, $\min(\omega(m), a) = m(a)$, and so $[\omega^{-1} \circ \omega(m)](a) = m(a)$ in this case too.

Finally we show $\omega \circ \omega^{-1} = \mathrm{id}_{PT}$. For p in PT and B in Y we have

$$\begin{aligned}\omega(\omega^{-1}(p))(B) &= \mathrm{wp}(\omega^{-1}(p), B) = \{a \mid \omega^{-1}(p)(a) \subseteq B\} \\ &= \{a \mid a \in p(Y) \wedge \min(p, a) \subseteq B\} \\ &= \{a \mid a \in p(Y) \wedge a \in p(B)\} \\ &= p(B).\end{aligned}$$ □

This theorem and its analog in Plotkin [31] should have a common generalization involving various degrees of nondeterminism (and corresponding notions of continuity).

FACT 2.12

(1) $wp(\bigcup_i m_i, B) = \bigcap_i wp(m_i, B)$.
(2) $wp(\{\cdot\}, B) = B$.
(3) $wp(m\,;\, m', B) = wp(m, wp(m', B))$.

PROOF

(1) We have $\mathrm{wp}(\bigcup_i m_i, B) = \mathrm{wp}(\bigsqcap m_i, B) = \omega(\bigsqcap m_i)(B) = (\bigsqcap_i \omega(m_i))B$ (by the Isomorphism Theorem) $= \bigsqcap \mathrm{wp}(m_i, B) = \bigcap \mathrm{wp}(m_i, B)$.
(2) Obvious.
(3) We have

$$\begin{aligned}wp(m;\, m', B) &= \{a \mid m;\, m'(a) \subseteq B\} \\ &= \{a \mid m'^{\dagger}(m(a)) \subseteq B\} \\ &= \{a \mid \perp \notin m(a) \wedge \bigcup m'(m(a)) \subseteq B\} \\ &= \{a \mid \perp \notin m(a) \wedge \forall\, b \in m(a).m'(b) \subseteq B\} \\ &= \{a \mid \perp \notin m(a) \wedge m(a) \subseteq \{b \mid m'(b) \subseteq B\}\} \\ &= \{a \mid \perp \notin m(a) \wedge m(a) \subseteq \mathrm{wp}(m', B)\} \\ &= \mathrm{wp}(m, \mathrm{wp}(m', B)).\end{aligned}$$ □

Finally, we can connect up the relational approach and predicate transformers as we now have an isomorphism $\mathrm{pt} : \mathrm{SR} \cong \mathrm{PT}$ where $\mathrm{pt} = \omega \circ \mathrm{st}$. It turns out that pt is given by

$$\mathrm{pt}(R, T)(B) = \{x \in T \mid R(x) \subseteq B\}.$$

The inverse pt^{-1} is given by the formula

$$\mathrm{pt}^{-1}(p) = \langle \{\langle x, y\rangle \mid x \in p(Y) \supset y \in \min(p, x)\}, p(Y)\rangle.$$

3. *Semantic Issues*

In this section we consider four semantics of a simple programming language of commands allowing countable nondeterminism and establish the relationships between the various semantics. Then we give some general results that no reasonable continuous models (in a sense to be spelled out) exist. The first semantics is

operational being given as a transition relation between configurations and specified axiomatically. The next two are standard nondeterministic state-transformation semantics based on the two discrete powerdomains we considered in Section 2.

Here we differ from Back [8], who defines a semantics based on $\mathscr{E}(X_\perp)$ but where the semantics of **while**-loops is defined as the limit of the first ω iterates. He points out that this does not capture the correct notion of termination and then considers alternative semantics; we follow Broy et al. [12] and carry the iterates to enough stages (at most all countable ordinals) to reach the least fixed point. Then with this definition, Theorem 3.2 shows that the operational and denotational semantics are identical.

Further, Theorem 3.1, shows that the semantics based on the Smyth order is a projection, under e_X, of the semantics based on the Egli–Milner ordering, and Corollary 3.1 then relates it to the operational semantics. Then we give a predicate transformer semantics, again iterating through suitable ordinals, following Boom [11], and we show in Theorem 3.3 and Corollary 3.2 that it is isomorphic to the semantics based on the Smyth order (following the ideas in Plotkin [31]). Corollary 3.3 relates the predicate transformer semantics to the operational semantics.

Finally, we turn to the negative result that no semantics of a general type can exist. First we consider semantics based on cpos. Any such semantics should satisfy certain properties, just to fit into the program of denotational semantics founded by Scott and Strachey: The semantics should be continuously compositional and least fixed point (definitions given below). Then Theorem 3.4 shows that no such semantics can exist, which agrees with the operational semantics by giving the same equalities between programs (perhaps via a so-called abstraction function). Next, as mentioned above, we widen the framework to T_0 spaces in order to include approaches based on metric spaces [15, 27]. We feel this work shows just why noncontinuous functions have arisen in the treatment of countable nondeterminism.

Throughout the rest of the paper we consider a simple programming language whose set of commands is parameterized on the following two sets:

First, ACom is the set of *atomic commands*, ranged over by the metavariable A. Second, BExp is the set of *Boolean expressions*, ranged over by B. Then, Com is the set of *commands* of the language, ranged over by S and generated by the following grammar:

$$S ::= \textbf{skip}\ A \mid S;\ S \mid \textbf{if}\ B\ \textbf{then}\ S\ \textbf{else}\ S\ \textbf{fi} \mid \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}.$$

We assume a *countable* unanalyzed set X of states (ranged over by σ) and we further assume we are given two semantic functions:

$$\mathscr{A} : \text{ACom} \to (X \to \mathscr{P}(X) - \{\varnothing\}),$$
$$\mathscr{B} : \text{BExp} \to (X \to \{tt, ff\}),$$

where $\{tt, ff\}$ is of course the set of *truthvalues*.

The assumption that for any $\sigma \in X$ $\mathscr{A}[\![A]\!](\sigma)$ is a nonempty and (necessarily) countable subset of X means that atomic commands are assumed to be always terminating and countably nondeterministic statements. A particular choice for A might be the statement $x := ?$ meaning set x to any value. If there were only one variable that could appear in the language, we could give the semantics of $x := ?$ by putting for any σ

$$\mathscr{A}[\![x := ?]\!](\sigma) = X.$$

We now provide three different semantics for commands.

3.1. OPERATIONAL SEMANTICS. We define a function

$$\mathrm{Op}: \mathrm{Com} \to (X_\perp \to \mathscr{E}(X_\perp))$$

by considering a transition relation $\to$ between configurations, that is, pairs $\langle S, \sigma \rangle$ consisting of a command and a state. We define $\to$ by the following clauses:

(i) $\langle A, \sigma \rangle \to \langle \mathbf{skip}, \sigma' \rangle$ if $\sigma' \in \mathscr{A}[\![A]\!](\sigma)$.
(ii) If $\langle S_1, \sigma \rangle \to \langle S_1', \sigma' \rangle$ then $\langle S_1; S, \sigma \rangle \to \langle S_1', S, \sigma' \rangle$.
(iii) $\langle \mathbf{skip}\ ; S, \sigma \rangle \to \langle S, \sigma \rangle$.
(iv) (1) $\langle \mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{fi}, \sigma \rangle \to \langle S, \sigma \rangle$ if $\mathscr{B}[\![B]\!](\sigma) = tt$,
(2) $\langle \mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{fi}, \sigma \rangle \to \langle S_2, \sigma \rangle$ if $\mathscr{B}[\![B]\!](\sigma) = ff$.
(v) (1) $\langle \mathbf{while}\ B\ \mathbf{do}\ S\ \mathbf{od}, \sigma \rangle \to \langle S; \mathbf{while}\ B\ \mathbf{do}\ S\ \mathbf{od}, \sigma \rangle$ if $\mathscr{B}[\![B]\!](\sigma) = tt$,
(2) $\langle \mathbf{while}\ B\ \mathbf{do}\ S\ \mathbf{od}, \sigma \rangle \to \langle \mathbf{skip}, \sigma \rangle$ if $\mathscr{B}[\![B]\!](\sigma) = ff$.

Intuitively, $\langle S_1, \sigma \rangle \to \langle S_2, \sigma' \rangle$ means that one step of execution of S_1 in state σ can lead to state σ' with S_2 being the remainder of S_1 to be executed.

Definition 3.1. S can *diverge* from σ (equivalently, *may not terminate* from σ) iff there exists an infinite sequence $\langle S_i, \sigma_i \rangle$ $(i = 0, 1, \ldots)$ such that

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

Notes

(1) If $S \neq \mathbf{skip}$, then for any σ there are S_1 and σ' such that $\langle S, \sigma \rangle \to \langle S', \sigma' \rangle$ (i.e., S can be executed for at least one step). This property can be termed *absence of blocking* and is evidently dependent on the syntax of our programming language. For example, it does not hold in the case of Dijkstra's control structures (see [16]).
(2) The set $\{\langle S', \sigma' \rangle \mid \langle S, \sigma \rangle \to \langle S', \sigma' \rangle\}$ is always countable (since X is assumed to be countable).

Definition 3.2. We define the function Op by

$$\mathrm{Op}[\![S]\!](\sigma) = \{\sigma' \mid \langle S, \sigma \rangle \to^* \langle \mathbf{skip}, \sigma' \rangle\} \cup \{\perp \mid S \text{ can diverge from } \sigma\}.$$

Of course, $\to^*$ is the transitive reflexive closure of $\to$.

The following characterization of Op will be used in what follows.

LEMMA 3.1. *Op is the least function of type*

$$\mathscr{F}: \mathit{Com} \to (X \to \mathscr{E}(X_\perp)) \textit{ such that for all } \sigma, \mathscr{F}[\![\mathbf{skip}]\!](\sigma) = \{\sigma\},$$

and, for $S \neq \mathbf{skip}$ *and any* σ

$$\mathscr{F}[\![S]\!](\sigma) = \bigcup \{\mathscr{F}[\![S']\!](\sigma') \mid \langle S, \sigma \rangle \to \langle S', \sigma' \rangle\}.$$

The ordering we are referring to is pointwise:

$$\mathscr{F} \sqsubseteq \mathscr{F}' \quad \textit{iff for all } S \textit{ and } \sigma, \quad \mathscr{F}[\![S]\!](\sigma) \sqsubseteq \mathscr{F}'[\![S]\!](\sigma).$$

PROOF. Obviously, Op satisfies the above equations. Consider any $\mathscr{F}$ that satisfies them. By induction on the length of the derivation we get that $\langle S, \sigma \rangle \to^* \langle \mathbf{skip}, \sigma' \rangle$ implies $\sigma' \in \mathscr{F}[\![S]\!](\sigma)$. This shows that $\mathrm{Op}[\![S]\!](\sigma) \sqsubseteq \mathscr{F}[\![S]\!](\sigma)$ in the case S can diverge from σ. In the other case the relation $\to$, restricted to the pairs $\langle S', \sigma' \rangle$ such that $\langle S, \sigma \rangle \to^* \langle S', \sigma' \rangle$, is well founded. We prove by induction with respect to this well-founded ordering that $\mathrm{Op}[\![S]\!](\sigma) = \mathscr{F}[\![S]\!](\sigma)$.

The case $S =$ **skip** is clear. Otherwise,

$$\begin{aligned}\mathscr{F}[\![S]\!](\sigma) &= \bigcup \{\mathscr{F}[\![S]\!](\sigma') \mid \langle S, \sigma\rangle \to \langle S', \sigma'\rangle\} \\ &= \bigcup \{\mathrm{Op}[\![S']\!](\sigma') \mid \langle S', \sigma\rangle \to \langle S', \sigma'\rangle\} \quad \text{(by the inductive assumption)} \\ &= \mathrm{Op}[\![S]\!](\sigma). \qquad \square\end{aligned}$$

We shall also need the following facts about Op.

LEMMA 3.2

(1) $Op[\![A]\!] = \lambda\sigma \in X.\ \mathscr{A}[\![A]\!](\sigma)$.
(2) $Op[\![S_1 ; S_2]\!] = Op[\![S_1]\!]; Op[\![S_2]\!]$
(the second ";" is the composition operation defined in Section 2).
(3) $Op[\![$**if** B **then** S_1 **else** S_2 **fi**$]\!] = \lambda\sigma \in X.$ **if** $\mathscr{B}[\![B]\!](\sigma)$ **then** $Op[\![S_1]\!](\sigma)$ **else** $Op[\![S_2]\!](\sigma)$.
(4) *Let* $m = Op[\![$**while** B **do** S **od**$]\!]$. *Then for all* σ

$$m(\sigma) = \textbf{if } \mathscr{B}[\![B]\!](\sigma) \textbf{ then } (Op[\![S]\!]; m)(\sigma) \textbf{ else } \{\sigma\}.$$

PROOF

(1) Trivial.
(2) We have

$$\begin{aligned}\sigma' \in \mathrm{Op}[\![S_1 ; S_2]\!](\sigma) \quad &\text{iff} \quad \langle S_1 ; S_2, \sigma\rangle \to^* \langle \textbf{skip}, \sigma'\rangle \\ &\text{iff} \quad \exists\, \sigma_1[\langle S_1, \sigma\rangle \to^* \langle \textbf{skip}, \sigma_1\rangle \\ &\qquad\qquad \text{and } \langle S_2, \sigma_1\rangle \to^* \langle \textbf{skip}, \sigma'\rangle] \\ &\text{iff} \quad \exists\, \sigma_1[\sigma_1 \in \mathrm{Op}[\![S_1]\!](\sigma) \text{ and } \sigma' \in \mathrm{Op}[\![S_2]\!](\sigma_1)] \\ &\text{iff} \quad \sigma' \in \mathrm{Op}[\![S_1]\!]; \mathrm{Op}[\![S_2]\!](\sigma).\end{aligned}$$

Also $\perp \in \mathrm{Op}[\![S_1; S_2]\!](\sigma)$ iff $S_1; S_2$ can diverge from σ and this happens in one of these two cases:

(a) S_1 can diverge from σ;
(b) $\exists\sigma'[\langle S_1, \sigma\rangle \to^* \langle$**skip**$, \sigma'\rangle$ and S_2 can diverge from $\sigma']$ which by definition of the composition operation are just the two cases when $\perp \in (\mathrm{Op}[\![S_1]\!] ; \mathrm{Op}[\![S_2]\!])(\sigma)$.

(3) Straightforward by definition.
(4) The case in which $\mathscr{B}[\![B]\!](\sigma) = ff$ is obvious. In the other cases we have

$$\begin{aligned}m(\sigma) &= \mathrm{Op}[\![S;\ \textbf{while } B \textbf{ do } S \textbf{ od}]\!](\sigma) \quad \text{(by Lemma 3.1)} \\ &= (\mathrm{Op}[\![S]\!] ; m)(\sigma) \quad \text{(by case 2).}\end{aligned}$$ $\square$

3.2 DENOTATIONAL SEMANTICS. We define now two functions

$$\mathscr{D}_{\mathscr{E}}\colon \mathrm{Com} \to \mathrm{ET}_{X,X} \qquad \text{and} \qquad \mathscr{D}_{\mathscr{S}}\colon \mathrm{Com} \to \mathrm{ST}_{X,X}$$

by the same type of equations. Let $\mathscr{D}$ be, indifferently, $\mathscr{D}_{\mathscr{E}}$ or $\mathscr{D}_{\mathscr{S}}$. We define

(i) $\mathscr{D}[\![$**skip**$]\!] = \{\cdot\}$ *(the singleton function defined in Section 2).*
(ii) $\mathscr{D}[\![A]\!] = \lambda\sigma \in X.\ \mathscr{A}[\![A]\!](\sigma)$.
(iii) $\mathscr{D}[\![S_1 ; S_2]\!]. = \mathscr{D}[\![S_1]\!]; \mathscr{D}[\![S_2]\!]$.
(iv) $\mathscr{D}[\![$**if** B **then** S_1 **else** S_2 **fi**$]\!](\sigma) =$ **if** $\mathscr{B}[\![B]\!](\sigma)$ **then** $\mathscr{D}[\![S_1]\!](\sigma)$ **else** $\mathscr{D}[\![S_2]\!](\sigma)$.
(v) $\mathscr{D}[\![$**while** B **do** S **od**$]\!] = \mu m.\lambda\sigma \in X.$ **if** $\mathscr{B}[\![B]\!](\sigma)$ **then** $(\mathscr{D}[\![S]\!] ; m)(\sigma)$ **else** $\{\sigma\}$.

Note. $\mathscr{S}(X_\perp)$ need not be a cpo, so $\mathscr{D}_\mathscr{S}$ might not be well defined in case (v). To show this is not the case, we prove

THEOREM 3.1. *For all S, the function* $\mathscr{D}_\mathscr{S}$ *is well defined and*

$$e_X \circ \mathscr{D}_\mathscr{E}[\![S]\!] = \mathscr{D}_\mathscr{S}[\![S]\!]$$

(e_X *is the projection of* $\mathscr{E}(X_\perp)$ *onto* $\mathscr{S}(X_\perp)$ *defined in Section* 2).

PROOF. By induction on the structure of S, following the definition of $\mathscr{D}_\mathscr{E}$.

(i) By Fact 2.10.
(ii) Trivial.
(iii) We have

$$\begin{aligned} e_X \circ \mathscr{D}_\mathscr{E}[\![S_1 ; S_2]\!] &= e_X \circ (\mathscr{D}_\mathscr{E}[\![S_1]\!]; \mathscr{D}_\mathscr{E}[\![S_2]\!]) \\ &= (e_X \circ \mathscr{D}_\mathscr{E}[\![S_1]\!]); (e_X \circ \mathscr{D}_\mathscr{E}[\![S_1]\!]) \quad \text{(by Fact 2.11)} \\ &= \mathscr{D}_\mathscr{S}[\![S_1]\!]; \mathscr{D}_\mathscr{S}[\![S_2]\!] \quad \text{(by inductive assumption)} \\ &= \mathscr{D}_\mathscr{S}[\![S_1 ; S_2]\!]. \end{aligned}$$

(iv) Obvious.
(v) Let S be **while** B **do** S_1 **od**. For $\mathscr{F} \in \{\mathscr{E}, \mathscr{S}\}$ let $\Phi_\mathscr{F}: (X \to \mathscr{F}(X_\perp)) \to (X \to \mathscr{F}(X_\perp))$ be defined by $\Phi_\mathscr{F}(m) = \lambda\sigma \in X.$ **if** $\mathscr{B}[\![B]\!](\sigma)$ **then** $(\mathscr{D}_\mathscr{F}[\![S_1]\!] ; m)(\sigma)$ **else** $\{\sigma\}$.

Then both $\Phi_\mathscr{E}$ and $\Phi_\mathscr{S}$ are well defined and monotone. Now ET is a cpo, so $\mu m.\Phi_\mathscr{E}(m)$ exists being some $\Phi^\omega_\mathscr{E}$. In fact we have $\mathscr{D}_\mathscr{E}[\![S]\!] = \mu m.\Phi_\mathscr{E}(m)$. To show $\mu m.\Phi_\mathscr{S}(m)$ exists is sufficient to show that the following diagram commutes:

$$\begin{array}{ccc} \mathrm{ET} & \xrightarrow{\Phi_\mathscr{E}} & \mathrm{ET} \\ {\scriptstyle \lambda m.e_X \circ m}\downarrow & & \downarrow{\scriptstyle \lambda m.e_X \circ m} \\ \mathrm{ST} & \xrightarrow{\Phi_\mathscr{S}} & \mathrm{ST} \end{array}$$

and use the Transfer Lemma.

For any $m \in \mathrm{ET}$ and $\sigma \in X$ we have

$$\begin{aligned} (e_X \circ \Phi_\mathscr{E}(m))(\sigma) &= e_X(\textbf{if } \mathscr{B}[\![B]\!](\sigma) \textbf{ then } (\mathscr{D}_\mathscr{E}[\![S_1]\!] ; m(\sigma) \textbf{ else } \{\sigma\}) \\ &= \textbf{if } \mathscr{B}[\![B]\!](\sigma) \textbf{ then } e_X \circ (\mathscr{D}_\mathscr{E}[\![S_1]\!] ; m)(\sigma) \textbf{ else } e_X(\{\sigma\}) \\ &= \textbf{if } \mathscr{B}[\![B]\!](\sigma) \textbf{ then } ((e_X \circ \mathscr{D}_\mathscr{E}[\![S_1]\!]) ; (e_X \circ m))(\sigma) \textbf{ else } \{\sigma\} \\ &\qquad \text{(by Facts 2.10 and 2.11)} \\ &= \Phi_\mathscr{S}(e_X \circ m)(\sigma) \quad \text{(by inductive assumption).} \end{aligned}$$

Hence by the Transfer Lemma $\mu m.\Phi_\mathscr{S}(m)$ exists and

$$e_X \circ \mu m.\Phi_\mathscr{E}(m) = \mu m.\Phi_\mathscr{S}(m),$$

showing that

$$e_X \circ \mathscr{D}_\mathscr{E}[\![S]\!] = \mathscr{D}_\mathscr{S}[\![S]\!]. \qquad \square$$

To tie up $\mathscr{D}_\mathscr{E}$ with the operational semantics, we first prove the following lemma.

LEMMA 3.3. $\mathscr{D}_\mathscr{E}$ *satisfies the equations of Lemma* 3.1.

PROOF. By induction on the structure of S. The case $S =$ **skip** is clear:

$S = A$; $\mathscr{D}_{\mathscr{E}}[\![A]\!](\sigma) = \mathscr{A}[\![A]\!](\sigma)$
$= \{\sigma' \mid \langle A, \sigma\rangle \to \langle \textbf{skip}, \sigma'\rangle\}$
$= \bigcup \{\mathscr{D}_{\mathscr{E}}[\![S']\!](\sigma') \mid \langle A, \sigma\rangle \to \langle S', \sigma'\rangle\}$;

$S = S_1, S_2$; *subcase* $S_1 \neq$ **skip**
$\mathscr{D}_{\mathscr{E}}[\![S_1 ; S_2]\!](\sigma) = (\mathscr{D}_{\mathscr{E}}[\![S_1]\!] ; \mathscr{D}_{\mathscr{E}}[\![S_2]\!])(\sigma)$
$= \mathscr{D}_{\mathscr{E}}[\![S_2]\!]^\dagger(\mathscr{D}_{\mathscr{E}}[\![S_1]\!](\sigma))$ (by the definition of ;)
$= \mathscr{D}_{\mathscr{E}}[\![S_2]\!]^\dagger(\bigcup \{\mathscr{D}_{\mathscr{E}}[\![S']\!](\sigma') \mid \langle S_1, \sigma\rangle \to \langle S', \sigma'\rangle\})$ (by inductive assumption)
$= \bigcup \{\mathscr{D}_{\mathscr{E}}[\![S_2]\!]^\dagger(\mathscr{D}_{\mathscr{E}}[\![S']\!](\sigma')) \mid \langle S_1, \sigma\rangle \to \langle S', \sigma'\rangle\}$ (by complete linearity)
$= \bigcup \{\mathscr{D}_{\mathscr{E}}[\![S' ; S_2]\!](\sigma') \mid \langle S_1, \sigma\rangle \to \langle S', \sigma'\rangle\}$
$= \bigcup \{\mathscr{D}_{\mathscr{E}}[\![S'']\!](\sigma) \mid \langle S_1 ; S_2, \sigma\rangle \to \langle S'', \sigma'\rangle\}$ (by the definition of ;)

subcase $S_1 =$ **skip**

$\mathscr{D}_{\mathscr{E}}[\![\textbf{skip}; S_2]\!](\sigma) = \mathscr{D}_{\mathscr{E}}[\![S_2]\!](\sigma)$
$= \bigcup \{\mathscr{D}_{\mathscr{E}}[\![S']\!](\sigma') \mid \langle \textbf{skip}; S_2, \sigma\rangle \to \langle S', \sigma'\rangle\}$

$S =$ **if** B **then** S_1 **else** S_2 **fi**; straightforward.
$S =$ **while** B **do** S **od**; by cases whether $\mathscr{B}[\![B]\!](\sigma)$ is *tt* or *ff*
subcase: *tt*; $\mathscr{D}_{\mathscr{E}}[\![\textbf{while } B \textbf{ do } S \textbf{ od}]\!](\sigma)$
$= \mathscr{D}_{\mathscr{E}}[\![S]\!] ; \mathscr{D}_{\mathscr{E}}[\![\textbf{while } B \textbf{ do } S \textbf{ od}]\!](\sigma)$
$= \mathscr{D}_{\mathscr{E}}[\![S; \textbf{while } B \textbf{ do } S \textbf{ od}]\!](\sigma)$
$= \bigcup \{\mathscr{D}_{\mathscr{E}}[\![S']\!](\sigma') \mid \langle \textbf{while } B \textbf{ do } S \textbf{ od}, \sigma\rangle \to \langle S', \sigma'\rangle\}$

subcase: *ff*; trivial □

THEOREM 3.2. $\mathscr{D}_{\mathscr{E}} = Op$.

PROOF. By Lemmas 3.1 and 3.3, we have $\mathrm{Op} \sqsubseteq \mathscr{D}_{\mathscr{E}}$. We prove the converse by induction of S using Lemma 3.2.

A, **skip**; there $\mathscr{D}_{\mathscr{E}}[\![S]\!] = \mathrm{Op}[\![S]\!]$ by Lemma 3.2.1.
$S_1 ; S_2$; straightforward using Lemma 3.2.2.
if B **then** S_1 **else** S_2 **fi**; straightforward using Lemma 3.2.3.
while B **do** S_1 **od**; let $m = \mathrm{Op}[\![S]\!]$. We have
$\lambda\sigma \in X.$**if** $\mathscr{B}[\![B]\!](\sigma)$ **then** $(\mathscr{D}_{\mathscr{E}}[\![S_1]\!] ; m)(\sigma)$ **else** $\{\sigma\}$
$\sqsubseteq \lambda\sigma \in X.$**if** $\mathscr{B}[\![B]\!](\sigma)$ **then** $(\mathrm{Op}[\![S_1]\!] ; m)(\sigma)$ **else**$\{\sigma\}$ (by induction hypothesis)
$= m$ (by Lemma 3.2.4).

So m is a prefixed point of the functional of which $\mathscr{D}_{\mathscr{E}}[\![S]\!]$ is the least one. Thus $\mathscr{D}_{\mathscr{E}}[\![S]\!] \sqsubseteq m$. □

COROLLARY 3.1 (OPERATIONAL CHARACTERIZATION OF $D_{\mathscr{S}}$)

(1) *If S cannot diverge from σ then*

$$\sigma' \in \mathscr{D}_{\mathscr{S}}[\![S]\!](\sigma) \quad \textit{iff} \quad \langle S, \sigma\rangle \to^* \langle \textbf{skip}, \sigma'\rangle.$$

(2) $\perp \in \mathscr{D}_{\mathscr{S}}[\![S]\!](\sigma)$ *iff S can diverge from σ.*

PROOF. By Theorems 3.1 and 3.2. □

3.3 WEAKEST PRECONDITION SEMANTICS. Let PT be the set of all predicate transformers from X to X as defined in Section 2.

We define now a function $\mathscr{W}$: Com $\to$ PT, which we call the *weakest precondition semantics* (*wp semantics*).

(i) $\mathscr{W}[\![\textbf{skip}]\!] = \mathrm{id}$.
(ii) $\mathscr{W}[\![A]\!](R) = \mathrm{wp}(\mathscr{A}[\![A]\!], R)$
(where wp is the function defined in Section 2).
(iii) $\mathscr{W}[\![S_1 ; S_2]\!] = \mathscr{W}[\![S_1]\!] \circ \mathscr{W}[\![S_2]\!]$.
(iv) $\mathscr{W}[\![\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}]\!](R) = [\mathscr{B}[\![B]\!]^{-1}(tt) \cap \mathscr{W}[\![S_1]\!](R)] \cup [\mathscr{B}[\![B]\!]^{-1}(ff) \cap \mathscr{W}[\![S_2]\!](R)]$.
(v) $\mathscr{W}[\![\textbf{while } B \textbf{ do } S \textbf{ od}]\!](R) = \mu Q \subseteq X.([\mathscr{B}[\![B]\!]^{-1}(tt) \cap \mathscr{W}[\![S]\!](Q)] \cup [\mathscr{B}[\![B]\!]^{-1}(ff) \cap R])$.

It is clear that $\mathscr{W}$ is well defined, since $\mathscr{W}[\![S]\!]$ is monotone and so the corresponding function in case (v) is monotone as well and therefore has a least fixed point. However, we also wish to prove that for each S, $\mathscr{W}[\![S]\!]$ is a predicate transformer. This follows directly from the next theorem, which also, when taken together with Corollary 3.2 below, demonstrates the equivalence of the weakest precondition semantics and the Smyth state transformation semantics.

THEOREM 3.3. *For all $S \in$ Com and $R \subseteq X$ we have*

$$wp(\mathscr{D}_{\mathscr{S}}[\![S]\!], R) = \mathscr{W}[\![S]\!](R).$$

PROOF. By structural induction on S. Case (i) follows from Fact 2.12.2. Case (ii) is trivial and case (iii) follows from Fact 2.12.3. For case (iv) we calculate

$$\begin{aligned}
&\mathrm{wp}(\mathscr{D}_{\mathscr{S}}[\![\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}]\!], R)\\
&\quad = \{\sigma \mid \textbf{if } \mathscr{B}[\![B]\!](\sigma) \textbf{ then } \mathscr{D}_{\mathscr{S}}[\![S_1]\!](\sigma) \textbf{ else } \mathscr{D}_{\mathscr{S}}[\![S_2]\!](\sigma) \subseteq R\}\\
&\quad = \{\sigma \mid \mathscr{B}[\![B]\!](\sigma) = tt \wedge \mathscr{D}_{\mathscr{S}}[\![S_1]\!](\sigma) \subseteq R\}\\
&\qquad \cup \{\sigma \mid \mathscr{B}[\![B]\!](\sigma) = ff \wedge \mathscr{D}_{\mathscr{S}}[\![S_2]\!](\sigma) \subseteq R\}\\
&\quad = [\mathscr{B}[\![B]\!]^{-1}(tt) \cap \mathrm{wp}(\mathscr{D}_{\mathscr{S}}[\![S_1]\!], R)]\\
&\qquad \cup [\mathscr{B}[\![B]\!]^{-1}(ff) \cap \mathrm{wp}(\mathscr{D}_{\mathscr{S}}[\![S_2]\!], R)] \quad \text{(by induction hypothesis)}\\
&\quad = \mathscr{W}[\![\textbf{if } B \textbf{ then } S_1 \textbf{ else } S_2 \textbf{ fi}]\!](R).
\end{aligned}$$

For case (v), where $S =$ **while** B **do** S_1 **od**, we first define two functions.

Let Φ: ST $\to$ ST be defined by

$$\Phi(m)(\sigma) = \textbf{if } \mathscr{B}[\![B]\!](\sigma) \textbf{ then } (\mathscr{D}_{\mathscr{S}}[\![S_1]\!] ; m)(\sigma) \textbf{ else } \{\sigma\}$$

and G: $\mathscr{P}(X) \times \mathscr{P}(X) \to \mathscr{P}(X)$ by

$$G(Q, R) = [\mathscr{B}[\![B]\!]^{-1}(tt) \cap \mathscr{W}[\![S_2]\!](Q)] \cup [\mathscr{B}[\![B]\!]^{-1}(ff) \cap R]$$

(G is clearly monotone).

Then $\mathrm{wp}(\mathscr{D}_{\mathscr{S}}[\![\textbf{while } B \textbf{ do } S_1 \textbf{ od}]\!])(R) = \mathrm{wp}(\mu m.\Phi(m), R)$ and

$$\begin{aligned}
\mathscr{W}[\![\textbf{while } B \textbf{ do } S_1 \textbf{ od}]\!](R) &= \mu Q.G(Q, R)\\
&= (\mu f.\lambda Q.G(f(Q), Q))(R) \quad \text{(by Fact 2.2).}
\end{aligned}$$

To prove the claim it suffices now to show that $\mathrm{wp}(\mu m.\Phi(m), R) = (\mu f.\Psi(f))(R)$ where

$$\Psi\colon (\mathscr{P}(X) \underset{m}{\to} \mathscr{P}(X)) \to (\mathscr{P}(X) \underset{m}{\to} \mathscr{P}(X))$$

is defined by

$$\Psi(f)(Q) = G(f(Q), Q).$$

We prove this using the Transfer Lemma. First we prove that the following diagram commutes:

$$\begin{array}{ccc} \mathrm{ST} & \xrightarrow{\Phi} & \mathrm{ST} \\ \Big\downarrow{\scriptstyle \lambda m.\lambda R.\mathrm{wp}(m,R)} & & \Big\downarrow{\scriptstyle \lambda m.\lambda R.\mathrm{wp}(m,R)} \\ (\mathscr{P}(X) \xrightarrow[m]{} \mathscr{P}(X)) & \xrightarrow{\Psi} & (\mathscr{P}(X) \xrightarrow[m]{} \mathscr{P}(X)) \end{array}$$

We have

$$\begin{aligned} \mathrm{wp}(\Phi(m),R) &= [\mathscr{B}[\![B]\!]^{-1}(tt) \cap \mathrm{wp}(\mathscr{D}_{\mathscr{S}}[\![S_1]\!]; m, R] \\ &\quad \cup [\mathscr{B}[\![B]\!]^{-1}(ff) \cap \mathrm{wp}(\{\cdot\}, R)] \\ &= [\mathscr{B}[\![B]\!]^{-1}(tt) \cap \mathrm{wp}(\mathscr{D}_{\mathscr{S}}[\![S_1]\!], \mathrm{wp}(m,R))] \quad \text{(by Fact 2.12)} \\ &\quad \cup [\mathscr{B}[\![B]\!]^{-1}(ff) \cap R] \\ &= [\mathscr{B}[\![B]\!]^{-1}(tt) \cap \mathscr{W}[\![S_1]\!](\mathrm{wp}(m,R))] \quad \text{(by induction hypothesis)} \\ &\quad \cup [\mathscr{B}[\![B]\!]^{-1}(ff) \cap R] \\ &= G(\mathrm{wp}(m,R),R) \\ &= \Psi(\lambda Q.\mathrm{wp}(m,Q))(R), \end{aligned}$$

as required.

Clearly Φ and Ψ are monotonic and, by the Isomorphism Theorem $\mathrm{wp}(m, \cdot)$ is strict and continuous in m. Since $\mu m.\Phi(m) = \mathscr{D}_{\mathscr{S}}[\![\textbf{while } B \textbf{ do } S_1 \textbf{ od}]\!]$, $\mu m.\Phi(m)$ exists being a Φ^{ω}. We can now apply the Transfer Lemma owing to which we get that $\mu f.\Psi(f)$ exists and equals $\lambda R.\mathrm{wp}(\mu m.\Phi(m), R)$, which establishes the claim. □

COROLLARY 3.2. *For all S in Com, we have*

$$\mathscr{D}_{\mathscr{S}}[\![S]\!] = \omega^{-1}(\mathscr{W}[\![S]\!]).$$

PROOF. Immediate from Theorem 3.3 and Theorem 2.1. □

COROLLARY 3.3 (OPERATIONAL CHARACTERIZATION OF WP SEMANTICS). *For all commands S and predicates R a state σ is in $\mathscr{W}[\![S]\!](R)$ iff S cannot diverge from σ and whenever $\langle S, \sigma\rangle \to^* \langle \textbf{skip}, \sigma'\rangle$ then $\sigma' \in R$.*

PROOF. By Corollary 3.1 and Theorem 3.3. □

3.4 CONTINUOUS SEMANTICS. In all the above semantics for unbounded nondeterminism we have seen failures of continuity. Now we show that these failures are essential: In a sense to be made precise, no reasonable continuous semantics can exist. The ideas generalize to arbitrary T_0 topological spaces. In particular, we show that no reasonable semantics can exist using complete metric spaces and the Banach fixed-point theorem (as one would desire following Nivat [27] and de Bakker and Zucker [15]).

First we should decide what, in general, a continuous semantics should be, and even what a semantics should be. At the very least we should have a function

$$\mathscr{C}\colon \mathrm{Com} \to D,$$

where D is a structured set of some kind, and we call any such function a *semantics*. But more is required to follow the spirit of denotational semantics as practiced by Scott and Strachey and many others: The meaning of a command should be a function of the meaning of its parts. To formalize this idea, consider *command contexts*, which are just "commands with holes in them"; they can be defined by a grammar with the same rules as those for commands plus the rule $C ::= [\,]$; then

by $C[S]$ is meant the command obtained from C by replacing every occurrence of [ ] by S.

Definition 3.3. A semantics $\mathscr{C}$: Com $\to D$ is *compositional* if for every context C, there is a function $\mathscr{C}[\![C]\!]$: $D \to D$ such that for every command S

$$\mathscr{C}[\![C[S]]\!] = \mathscr{C}[\![C]\!](\mathscr{C}[\![S]\!]).$$

If D is a cpo, then $\mathscr{C}$ is *continuously* compositional if $\mathscr{C}[\![C]\!]$ can always be taken to be continuous.

To see the connection with the usual semantic equations, suppose there is a binary function, ;: $D^2 \to D$ and for each b in BExp there are functions if_B: $D^2 \to D$ and while_B: $D \to D$ such that the following equations always hold:

$$\begin{aligned} \mathscr{C}[\![S\ ;\ S']\!] &= \mathscr{C}[\![S]\!];\mathscr{C}[\![S']\!] \\ \mathscr{C}[\![\textbf{if } B \textbf{ then } S \textbf{ else } S' \textbf{ fi}]\!] &= \text{if}_B(\mathscr{C}[\![S]\!], \mathscr{C}[\![S]\!]) \\ \mathscr{C}[\![\textbf{while } B \textbf{ do } S \textbf{ od}]\!] &= \text{while}_B(\mathscr{C}[\![S]\!]). \end{aligned}$$

Then $\mathscr{C}$ is compositional. Furthermore, if these functions can be taken to be continuous, then $\mathscr{C}$ is continuously compositional. (As to the converses, if $\mathscr{C}$ is compositional, the required functions exist, but continuous ones need not exist even if $\mathscr{C}$ is continuously compositional. In fact a more appropriate general theory would consider contexts with several different kinds of holes; our notion is really only that of *unary* compositionality. However, since our present purpose is to establish negative results, it is actually better for us to consider the weaker unary notion.)

Usually the meaning of iterative commands is given as a least fixed point.

Definition 3.4. Let D be a partial order (cpo). A semantics $\mathscr{C}$: Com $\to D$ is a (*continuous*) *least fixed-point semantics* if it is (continuously) compositional and the (continuous) $\mathscr{C}[\![C]\!]$ can be taken so that $\mathscr{C}[\![\textbf{while } B \textbf{ do } S \textbf{ od}]\!] = \mu d \in D.\Phi(d)$, where $\Phi = \mathscr{C}[\![\textbf{if } B \textbf{ then } S; [\ ] \textbf{ else skip fi}]\!]$.

Now we decide how to formulate the requirement that the semantics be reasonable—it is clearly not enough just to ask for a continuous least fixed-point semantics since any constant $\mathscr{C}$ with range a cpo would do. It is natural to look for conditions involving the operational semantics, Op: Com $\to \text{ET}_{X,X}$. Consider the condition that the denotational semantics determines the operational semantics. This amounts to saying that, if two commands have the same denotational meaning, then they should have the same operational meaning. So define the *operational equivalence* relation between commands by

$$S \equiv S' \qquad \text{if} \qquad \text{Op}[\![S]\!] = \text{Op}[\![S']\!].$$

Then the condition is that $\mathscr{C}[\![S]\!] = \mathscr{C}[\![S']\!]$ always implies $S \equiv S'$. But we can reasonably ask for more, that denotational equivalence determines operational equivalence in all contexts. So let us say that a semantics $\mathscr{C}$ is *correct* iff for all commands S, S', and contexts C

$$\mathscr{C}[\![S]\!] = \mathscr{C}[\![S']\!] \supset \forall C.C[S] \equiv C[S'].$$

(Note that correctness is no stronger than the first condition when $\mathscr{C}$ is compositional, and, in fact, for our language of commands it is shown in Example 3.1 below that correctness is no stronger without any conditions in $\mathscr{C}$.)

Let us also say that a semantics $\mathscr{C}$ is *complete* if the converse of correctness holds, which is that for all commands S, S'

$$\forall C.C[S] \equiv C[S'] \supset \mathscr{C}[\![S]\!] = \mathscr{C}[\![S']\!].$$

If $\mathscr{C}$ is both correct and complete, we call it *fully abstract*; the idea is that it makes exactly the distinctions between commands needed to decide their operational behavior in all contexts. (See [10] and [22] for other work on full abstraction.)

It might be argued that full abstraction is too demanding and that one can use "concrete" meanings to define $\mathscr{C}$ and then provide the "abstract" meanings by an "abstraction" function. So let us say that ab: $D \to E$ is a *full abstraction* function for a semantics $\mathscr{C}$: Com $\to D$ if for all commands S, S' we have

$$\text{ab}(\mathscr{C}[\![S]\!]) = \text{ab}(\mathscr{C}[\![S']\!]) \qquad \text{iff} \qquad \forall C.C[S] \equiv C[S']$$

(and it will be obvious what implications constitute correctness and completeness for abstraction functions).

Let us see examples of semantics that fulfill as many conditions as possible.

Example 3.1. The semantics $\mathscr{D}_{\mathscr{E}}$: Com $\to \text{ET}_{X,X}$ is a least fixed-point semantics and since it is equal to Op by Theorem 3.2 it is fully abstract. (Note that this justifies our earlier comment on correctness as it shows that operational equivalence is *substitutive* in the sense that for all commands S, S'

$$S \equiv S' \supset \forall C.C[S] \equiv C[S'].)$$

But $\mathscr{D}_{\mathscr{E}}$ is not continuously compositional. For example consider the case in which $X = N$ and there are commands $x := ?$, $x := 0$, and S_m (for $m > 0$) defining the state transformers $\lambda n.N$, $\lambda n.\{0\}$, and $f^{(m)}$ where

$$f^{(m)}(n) = \begin{cases} \{0\} & (n < m) \\ \{\bot\} & (n \geq m) \end{cases}$$

(much as in the proof of Fact 2.5). Now suppose, for the sake of contradiction, that there is indeed a continuous $\Phi = \mathscr{D}_{\mathscr{E}}[\![x := ?; [\,]]\!]$ (of course, it will not be the one used to define $\mathscr{D}_{\mathscr{E}}$). But now we can calculate that

$$\begin{aligned} \lambda n.\{0\} &= \mathscr{D}_{\mathscr{E}}[\![x := ?; x := 0]\!] = \Phi(\mathscr{D}_{\mathscr{E}}[\![x := 0]\!]) = \Phi(\lambda n.\{0\}) \\ &= \Phi(\bigsqcup_m f^{(m)}) = \bigsqcup_m \Phi(f^{(m)}) = \bigsqcup_m \Phi(\mathscr{C}[\![S_m]\!]) \\ &= \bigsqcup_m \mathscr{D}_{\mathscr{E}}[\![x := ?; S_m]\!] = \bigsqcup_m \lambda n.\{0, \bot\} \\ &= \lambda n.\{0, \bot\}, \end{aligned}$$

which is the required contradiction.

Example 3.2. It is possible to find a continuous least fixed-point semantics with a full abstraction function that is, however, not continuous. Let $N_{\bot}^{\omega}$ be the cpo of all countably infinite sequences of elements of $N_{\bot}$, ordered pointwise; its elements $\sigma = \langle \sigma_n \rangle_n$ are to be thought of as oracles giving integer answer σ_n if that is not $\bot$ to query number n. The semantics is

$$\text{Oracle: Com} \to (X \to (N_{\bot}^{\omega} \to X_{\bot})).$$

We assume available a function $\mathscr{F}$: ACom $\to (X \to (N \to X))$, which indexes $\mathscr{A}$ is the sense that $\mathscr{A}[\![A]\!](x) = \mathscr{F}[\![A]\!](x)(N)$. The semantic clause for atomic commands is then

$$\text{Oracle}[\![A]\!](x)(\sigma) = \begin{cases} \mathscr{F}[\![A]\!](x)(\sigma_0) & (\text{if} \quad \sigma_0 \neq \bot), \\ \bot & (\text{if} \quad \sigma_0 = \bot). \end{cases}$$

The clause for compositions is

$$\text{Oracle}[\![S_1 ; S_2]\!](x)(\sigma) = \text{Oracle}[\![S_2]\!](\text{Oracle}[\![S_1]\!](x)(\langle \sigma_{2n} \rangle_{n\in\omega}))(\langle \sigma_{2n+1} \rangle_{n\in\omega}),$$

and the other clauses are omitted, being evident.

Each $f\colon N_\perp^\omega \to X_\perp$ can be regarded as a subset of $X_\perp$ parameterized on total oracles, and so we define a function Range: $(N_\perp^\omega \to X_\perp) \to \mathscr{E}_{(X_\perp)}$, by

$$\text{Range}(f) = \{f(w) \mid w \in N^\omega\},$$

and now define ab: $(X \to (N_\perp^\omega \to X_\perp)) \to (X \to \mathscr{E}(X_\perp))$ by

$$\text{ab}(m) = \text{Range} \circ m.$$

Then $\mathscr{D}_\mathscr{E} = \text{ab} \circ \text{Oracle}$, and so ab is indeed a full abstraction function; but, unfortunately, neither it nor Range is continuous (the proofs are omitted). Another continuous least fixed-point semantics has been given by Back [7].

Example 3.3. By a little trick we can construct a fully abstract continuously compositional semantics $\mathscr{C}\colon \text{Com} \to D$. Take D to be the flat cpo $\{\mathscr{D}_\mathscr{E}[\![S]\!] \mid S \in \text{Com}\}_\perp$ and define $\mathscr{C}$ by

$$\mathscr{C}[\![S]\!] = \mathscr{D}_\mathscr{E}[\![S]\!].$$

However $\mathscr{C}$ is not a least fixed-point semantics, although in an obvious sense it is a *maximal* fixed-point semantics. It is open whether there is such a *maximum* fixed-point semantics.

Now we state in what sense no reasonable continuous semantics can exist. First, fix the language (to some extent) by assuming that the Boolean expressions **true** and $x > 0$ are in BExp and that the atomic commands $x := 0$, $x := x \dot{-} 1$, and $x := ?$ (where x is a fixed identifier) are in ACom; assume too that $\mathscr{A}$ and $\mathscr{B}$ have the obvious definitions for these expressions and the commands to determine their operational semantics.

THEOREM 3.4. *There is no continuous least fixed-point semantics that has a continuous full abstraction function. (And in particular there is no fully abstract continuous least fixed-point semantics.)*

Note how close the above examples have come to satisfying all the conditions. The proof of Theorem 3.4 will proceed via two lemmas; the idea is as in the calculation of Example 3.1, but, in order to obtain the lub of a suitable increasing sequence like the $f^{(m)}$ there, we express $x := 0$ as an iterative program and use the least fixed-point property.

In the proofs we assume (for the sake of contradiction) that $\mathscr{C}\colon \text{Com} \to D$ is a continuous least fixed-point semantics ab: $D \to E$ is a continuous full abstraction function. The first lemma uses the notation Ω to denote the command **while true do skip od**.

LEMMA 3.4. *For all contexts C and commands S*

$$ab(\mathscr{C}[\![C[\Omega]]\!]) \sqsubseteq ab(\mathscr{C}[\![C[S]]\!]).$$

PROOF. First define S_n inductively by $S_0 = S$ and $S_{n+1} =$ **if true then skip**; S_n **else skip fi**. Since $S \equiv S_n$ and operational equivalence is substitutive, by the completeness of ab we have

$$\text{ab}(\mathscr{C}[\![C[S_n]]\!]) = \text{ab}(\mathscr{C}[\![C[S]]\!]).$$

Next define ω_n inductively by $\omega_0 = \bot$ and $\omega_{n+1} = \Phi(\omega_n)$, where

$$\Phi = \mathscr{C}[\![\textbf{if true then skip; [ ] else skip fi}]\!].$$

Now we show by induction that

$$\omega_n \sqsubseteq \mathscr{C}[\![S_n]\!].$$

For $n = 0$ this is clear. For $n + 1$ we calculate

$$\begin{aligned}\omega_{n+1} &= \Phi(\omega_n)\\ &\sqsubseteq \Phi(\mathscr{C}[\![S_n]\!]) \quad \text{(by induction hypothesis)}\\ &= \mathscr{C}[\![\textbf{if true then skip; } S_n \textbf{ else skip fi}]\!]\\ &= \mathscr{C}[\![S_{n+1}]\!].\end{aligned}$$

And now we see that

$$\begin{aligned}\mathrm{ab}(\mathscr{C}[\![C[\Omega]]\!]) &= \mathrm{ab}(\mathscr{C}[\![C]\!](\mathscr{C}[\![\Omega]\!]))\\ &= \mathrm{ab}(\mathscr{C}[\![C]\!] \sqcup \omega_n)\\ &= \bigsqcup_n \mathrm{ab}(\mathscr{C}[\![C]\!]\omega_n)\\ &\sqsubseteq \bigsqcup_n \mathrm{ab}(\mathscr{C}[\![C]\!]\mathscr{C}[\![S_n]\!]) \quad \text{(by above)}\\ &= \bigsqcup_n \mathrm{ab}(\mathscr{C}[\![C[S_n]]\!])\\ &= \mathrm{ab}(\mathscr{C}[\![C[S]]\!]) \qquad \text{(by above).}\end{aligned}$$

□

In the next lemma we use the notation W_n for an iterative command W of the form **while** B **do** S **od**, defined inductively by

$$\begin{aligned}W_0 &= \Omega,\\ W_{n+1} &= \textbf{if } B \textbf{ then } S;\ W_n \textbf{ else skip fi.}\end{aligned}$$

LEMMA 3.5 *Let* $W =$ **while** B **do** S **od** *be an iteration. Then* $ab(\mathscr{C}[\![C[W_n]]\!])$ *is increasing in n and for any context C*

$$ab(\mathscr{C}[\![C[W]]\!]) = \bigsqcup_n ab(\mathscr{C}[\![C[W_n]]\!]).$$

PROOF. Define contexts C_n inductively by $C_0 = [\ ]$ and $C_{n+1} =$ **if** B **then** S; C_n **else skip fi**. Clearly $W_n = C_n[\Omega]$ and $W_{n+1} = C_n[W_1]$, and so Lemma 3.4 tells us that $\mathrm{ab}(\mathscr{C}[\![C[W_n]]\!])$ is increasing. Equally, since $W \equiv C_n[W]$, we also have that $\mathrm{ab}(\mathscr{C}[\![C[W_n]]\!]) \sqsubseteq \mathrm{ab}(\mathscr{C}[\![C[W]]\!])$ by completeness, and so

$$\mathrm{ab}(\mathscr{C}[\![C[W]]\!]) \sqsupseteq \bigsqcup_n \mathrm{ab}(\mathscr{C}[\![C[W_n]]\!]),$$

and it remains to prove the converse relation. For this define w_n inductively by putting $w_0 = \bot$ and $w_{n+1} = \mathscr{C}[\![\textbf{if } b \textbf{ then } S;\ [\ ] \textbf{ else skip fi}]\!](w_n)$.

Note that since $\mathscr{C}$ is a continuous least fixed-point semantics $\mathscr{C}[\![W]\!] = \sqcup_{w_n}$. An easy induction like that in the proof of the previous lemma shows that

$$w_n \sqsubseteq \mathscr{C}[\![W_n]\!],$$

and we calculate that

$$\begin{aligned}\mathrm{ab}(\mathscr{C}[\![C[W]]\!]) &= \mathrm{ab}(\mathscr{C}[\![C]\!](\mathscr{C}[\![W]\!]))\\ &= \mathrm{ab}(\mathscr{C}[\![C]\!](\sqcup w_n))\\ &= \sqcup\, \mathrm{ab}(\mathscr{C}[\![C]\!](w_n)) \quad \text{(by the continuity of } \mathscr{C}[\![C]\!] \text{ and ab)}\\ &\sqsubseteq \sqcup\, \mathrm{ab}(\mathscr{C}[\![C]\!](\mathscr{C}\lfloor w_n]\!]))\\ &= \sqcup\, \mathrm{ab}(\mathscr{C}[\![C[W_n]]\!]).\end{aligned}$$

□

PROOF OF THEOREM 3.4. Let W be the iterative program **while** $x > 0$ **do** $x := x - 1$ **od**. Clearly $(x := ?;\ W) \equiv x := 0$. Let $(x := 0$ **or** $\Omega)$ abbreviate $x := ?$; **if** $x > 0$ **then** Ω **else skip fi** (lacking a nondeterministic choice command). We show by considering the approximations, W_n, of W, that $\mathrm{ab}(\mathscr{C}[\![x := 0$ **or** $\Omega]\!])$ is equal to $\mathrm{ab}(\mathscr{C}[\![x := ?;\ W]\!])$ and hence to $\mathrm{ab}(\mathscr{C}[\![x := 0]\!])$. This will prove the theorem as clearly $(x := 0$ **or** $\Omega) \not\equiv (x := 0)$ and so ab cannot be correct, as assumed.

First we see by induction that $x := ?;\ W_{n+1} \equiv x := 0$ **or** Ω. For $n = 0$ we have

$$\begin{aligned} x := ?;\ W_1 &= x := ?;\ \textbf{if } x > 0 \textbf{ then } x := x \dot{-} 1;\ \Omega \textbf{ else skip fi} \\ &\equiv (x := 0 \textbf{ or } \Omega) \quad (\text{since } (x := x \dot{-} 1;\ \Omega) \equiv \Omega). \end{aligned}$$

For $n + 1$ we have

$$\begin{aligned} x := ?;\ W_{n+2} &= x := ?;\ \textbf{if } x > 0 \textbf{ then } x := x \dot{-} 1;\ W_{n+1} \textbf{ else skip fi} \\ &\equiv x := ?;\ \textbf{if } x > 0 \textbf{ then } x := ?;\ W_{n+1} \textbf{ else skip fi} \\ &\equiv x := ?;\ \textbf{if } x > 0 \textbf{ then } (x := 0 \textbf{ or } \Omega) \textbf{ else skip fi} \qquad (\text{by induction}) \\ &\equiv (x := 0 \textbf{ or } \Omega). \end{aligned}$$

And now we can calculate that

$$\begin{aligned} \mathrm{ab}(\mathscr{C}[\![x := ?;\ W]\!]) &= \bigsqcup_{n \geq 1} \mathrm{ab}(\mathscr{C}[\![x := ?;\ W_n]\!]) \quad (\text{by Lemma 3.5 dropping the first term}) \\ &= \mathrm{ab}(\mathscr{C}[\![x := 0 \textbf{ or } \Omega]\!]) \qquad (\text{by the above}), \end{aligned}$$

thereby concluding the proof. □

Note that only a few simple instances of the completeness of the full abstraction function were used in the proof; the reader may enjoy enumerating them.

A version of Theorem 3.4 holds in any topological space (all spaces are assumed to be T_0). A certain amount of—mostly standard—topological background is needed.

Definition 3.5. If D is a cpo, the *Scott topology* on D is defined by putting V open iff (i) y is in V whenever x is and $x \sqsubseteq y$ and (ii) if the lub of a directed set is in V, then so is some element of the set. Let X be a T_0-space. The *specialization order* on X is defined by $x \sqsubseteq y$ iff $\forall$ open $V.x \in V \supset y \in V$. If x_n is a sequence of elements in X, then $\lim x_n = x$ means that, if x is in a given open set then almost all the x_n are.

Notes. The Scott topology is always T_0. The specialization order is always a partial order; in T_1-spaces, it is equality. If taken from the Scott topology, it is the original partial order. If D and E are cpos, then $f\colon D \to E$ is continuous iff it is continuous with respect to the associated topologies, and so there is no confusion between two different notions of continuity. Limits are unique in Hausdorff spaces but not in T_0-spaces where we do have that if $\lim x_n = y$ and almost all the x_n are equal, say to x, then $y \sqsubseteq x$. In a cpo if x_n is an increasing sequence, then $\lim x_n = y$ iff $y \sqsubseteq \bigsqcup x_n$. If $f\colon X \to Y$ is a continuous map of T_0-spaces and $\lim x_n = x$ in X, then $\lim f(x_n) = f(x)$ in Y; also f is monotonic with respect to the specialization order.

Now we shall see that there is no continuously compositional semantics $\mathscr{C}$: Com $\to X$ with a continuous full abstraction function ab: $X \to Y$ (in the obvious senses), where X, Y are T_0-spaces, and that the following two conditions hold:

(1) For all contexts C and commands S, $\mathrm{ab}(\mathscr{C}[\![C[\Omega]]\!]) \sqsubseteq \mathrm{ab}(\mathscr{C}[\![C[S]]\!])$.
(2) For all iteration commands W, $\lim \mathscr{C}[\![W_n]\!] = \mathscr{C}[\![W]\!]$.

(Condition (1) is just the generalization of Lemma 3.4 to T_0-spaces and condition (2) replaces the least fixed-point condition.)

First we show that the generalization of Lemma 3.5 to T_0-spaces holds. So suppose $W =$ **while** b **do** S **od** is an iteration and let C be a context. Then using (1) it follows exactly as in the proof of the lemma that $\mathrm{ab}(\mathscr{E}[\![C[W_n]]\!])$ is increasing and has $\mathrm{ab}(\mathscr{E}[\![C[W]]\!])$ as an upper bound. Let y be any other upper bound and suppose $\mathrm{ab}(\mathscr{E}[\![C[W]]\!])$ is in an open set V. Now since both ab and $\mathscr{E}[\![C]\!]$ are continuous we have by (2) that

$$\lim \mathrm{ab}(\mathscr{E}[\![C]\!]\mathscr{E}[\![W_n]\!]) = \mathrm{ab}(\mathscr{E}[\![C]\!]\mathscr{E}[\![W]\!]).$$

So almost every $\mathrm{ab}(\mathscr{E}[\![C[W_n]]\!])$ is in V and so too, therefore, is y. This proves that $\mathrm{ab}(\mathscr{E}[\![C[W]]\!]) \sqsubseteq y$ and so that the conclusion of Lemma 3.5 holds, as required. Using this we can now follow the proof of Theorem 3.4 word for word.

Nivat [21] and de Bakker and Zucker [15] proposed the use of complete metric spaces and the Banach fixed-point theorem. Let us see there is no least fixed-point semantics $\mathscr{E}$: Com $\to X$ with a continuous full abstraction function ab: $X \to Y$, where X is a complete metric space, such that for any iteration command $W =$ **while** B **do** S **od** the function $\Phi = \mathscr{E}[\![$**if** B **then** S; [ ] **else skip fi**$]\!]$ is contracting. Note that in this context least fixed point means unique fixed point and by the Banach fixed-point theorem for any x in X

$$\lim \Phi^n(x) = \mathscr{E}[\![W]\!],$$

and taking $x = \mathscr{E}[\![\Omega]\!]$ we see that (2) holds. By the general remarks it remains to establish (1) for such a $\mathscr{E}$ and ab.

PROOF OF (1). Let C be a context and S be a command. Define S_n as in the proof of Lemma 3.4. Then taking x to be $\mathscr{E}[\![S]\!]$ in the above, we see that $\lim \mathscr{E}[\![S_n]\!] = \mathscr{E}[\![\Omega]\!]$, since $\Phi^n(\mathscr{E}[\![S]\!]) = \mathscr{E}[\![S_n]\!]$. But since both $\mathscr{E}[\![C]\!]$ and ab are continuous, it follows that

$$\lim \mathrm{ab}(\mathscr{E}[\![C[S_n]]\!]) = \mathrm{ab}(\mathscr{E}[\![C[\Omega]]\!]).$$

Now, just as in the proof of Lemma 3.4, completeness implies that the $\mathrm{ab}(\mathscr{E}[\![C[S_n]]\!])$ are all equal to $\mathrm{ab}(\mathscr{E}[\![C[S]]\!])$, and (1) follows by a remark in the above notes.

In [15] de Bakker and Zucker provide a semantics and an abstraction function satisfying all the above conditions, except that the abstraction function is not continuous.

4. *Proof Theory*

In this section we consider a Hoare-like proof system for the total correctness of programs; we demonstrate its soundness, and give a relative completeness theorem after the fashion of Cook (see [2] for a survey of results of this kind). The programs are the usual **while** programs, but an additional random assignment $x := ?$ is allowed. For considerations of partial correctness of this language, it is enough to add the axiom of random assignment given below to the usual system for **while** programs to obtain a sound and complete Hoare-like proof system. This contrasts with the proof systems for total correctness that we consider here, for which additionally the **while** rule has to be modified so that the loop counter ranges over countable ordinals instead of natural numbers.

Our assertion language L contains two sorts: *data* (for program data) and *ord* (for ordinals); we assume as given some constants and function and predicate symbols, including a constant 0, of sort *ord*, and a binary predicate symbol $<$ over *ord*. We use x, y, z as variables of sort *data*; α, β, γ as variables of sort *ord*; and u as a variable of sort *data* or *ord*. t ranges over terms, which are built up from

variables, constants, and function symbols in the usual way. We use p, q, r to range over L-formulas; L also includes second-order set variables $a, b, c, \ldots$. These set variables are of arbitrary arity and sort. We write $p(a_1, \ldots, a_m, u_1, \ldots, u_n)$ to show some free variables of p (and perhaps not all). We write $a(t_1, \ldots, t_n)$ instead of the atomic formula $(t_1, \ldots, t_n) \in a$; such a formula is *well formed* if the sorts and number of the terms $t_1, \ldots, t_n$ agree with the sort and arity of a. Formulas are built up from these atomic formulas and from the usual atomic formulas $P(t_1, \ldots, t_n)$ (where P is a predicate symbol and the arity of P and the sorts of the t_i agree) by the usual Boolean connectives (conjunction, disjunction, and negation) and by quantification over variables of sorts *data* and *ord.* Although set variables cannot be quantified over, they can be bound by the least fixed-point operator. We use the following notation first introduced by Gurevich in [19]. For any formula $p(a, u_1, \ldots, u_n)$, where $a(u_1, \ldots, u_n)$ and $a(t_1, \ldots, t_n)$ are well-formed atomic formulas and every free occurrence of a in p is positive, the abstraction

$$(t_1, \ldots, t_n) \in \mu a(u_1, \ldots, u_n)p$$

is also a formula.

A positive (respectively, negative) occurrence of a set variable in a formula is defined in the usual way, with the additional stipulation that an occurrence of a set variable in the new formula is positive (respectively, negative) if it is so in p. (In other words, an occurrence is positive if it is within the scope of an even number of negation signs.) The free variables of the new formula are those of $t_1, \ldots, t_n$ and the variables of p other than $a, u_1, \ldots, u_n$. Thus, the least fixed-point operator binds $a, u_1, \ldots, u_n$. This assertion language is based on the μ-calculus of Hitchcock and Park [23]. The main difference is that we allow ordinals. This specific choice of the assertion language is needed only for completeness, as the soundness should hold for any reasonable assertion language.

Now we can finish specifying the syntax of our programming language. For convenience we only consider a fixed finite set of *data* variables, Var $= \{x_1, \ldots, x_k\}$. Boolean expressions are taken to be those quantifier-free L-formulas whose variables are all in Var and whose symbols have sorts only involving *data.* Atomic commands are taken to be of the form $x := t$ (*ordinary assignment*), where all the symbols of t have sorts only involving *data*, or $x := ?$ (*random assignment*).

Before turning to semantic issues, we give our logic and work out an example. The formulas of the logic are all L-formulas, together with all those of the form

$$\{p\}\, S\, \{q\}$$

(the latter meaning that, for all values of parameters, if σ is a state satisfying p, then every execution sequence of S from σ terminates and ends in a state satisfying q). The axioms and rules of the logic are as follows:

(1) *assignment*

$$\{p[t/x]\}x := t\{p\}$$

where $p[t/x]$ is the result of substituting t for all free occurrences of x in p.

(2) *random assignment*

$$\{p\}x := ?\{p\},$$

provided x is not free in p.

(3) **if–then–else** *rule*

$$\frac{\{p \wedge B\}\ S_1\ \{q\},\ \{p \wedge \neg B\}\ S_2\ \{q\}}{\{p\}\ \textbf{if}\ B\ \textbf{then}\ S_1\ \textbf{else}\ S_2\ \textbf{fi}\ \{q\}}.$$

(4) *composition rule*

$$\frac{\{p\}\ S_1\ \{q\},\ \{q\},\ S_2\ \{r\}}{\{p\}\ S_1\ ;\ S_2\ \{r\}}.$$

(5) **while** *rule*

$$\frac{p(\alpha) \wedge 0 < \alpha \rightarrow B,\ \{p(\alpha) \wedge 0 < \alpha\} S \{\exists \beta < \alpha \cdot p(\beta)\},\ p(0) \rightarrow \neg B}{\{\exists \alpha p(\alpha)\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p(0)\}}.$$

We call $p(\alpha)$ the *loop invariant.*

(6) *consequence rule*

$$\frac{p \rightarrow p',\ \{p'\}\ S\ \{q'\},\ q' \rightarrow q}{\{p\}\ S\ \{q\}}.$$

Call the above system T; we write $F \vdash_T \{p\}\ S\ \{q\}$ to mean that $\{p\}\ S\ \{q\}$ can be proved in T from the formulas in F. The random assignment axiom was introduced by Harel [20]. The above **while** rule is a straightforward generalization of the following **while** rule for total correctness of the usual **while** programs given in [20].

(7) **while** *rule II*

$$\frac{p(\alpha + 1) \rightarrow B,\ \{p(\alpha + 1)\}\ S\ \{p(\alpha)\},\ p(0) \rightarrow \neg B}{\{\exists \alpha p(\alpha)\}\ \textbf{while}\ B\ \textbf{do}\ S\ \textbf{od}\ \{p(0)\}}.$$

(A slightly different vocabulary is assumed here—α ranges here over natural numbers.) We show in a moment that **while** rule II is not sufficient for proofs of total correctness of the programs considered here.

Another problem that arises here is that of expressibility of total correctness of (countably) nondeterministic programs in the dynamic logic considered by Harel [20]. Total correctness of deterministic programs can be expressed in deterministic dynamic logic (DDL) by the formula $p \rightarrow \langle S \rangle q$, equivalent to our $\{p\}\ S\ \{q\}$. However, for the case of nondeterministic programs, this formula is not equivalent to $\{p\}\ S\ \{q\}$ since the modality "$\langle\ \rangle$" expresses only existence of a terminating computation. To overcome this problem, Harel [20] introduces a formula loop_S stating existence of infinite computations.

In the dynamic logic augmented by this formula, total correctness of nondeterministic programs can be expressed. Harel [20] provides an arithmetically sound and complete axiomatization of this logic, but for the case of programs admitting bounded nondeterminism only. Our completeness result suggests how to obtain a sound and relatively complete axiomatization of this logic in the presence of random assignment.

As an example proof in T, consider the following program:

$$S \equiv \textbf{while}\ B\ \textbf{do}\ S_0\ \textbf{od},$$

where

$$B \equiv x = 0 \vee 0 < y$$

and

$$S_0 \equiv \textbf{if } x = 0 \textbf{ then } y := ?; x := 1 \textbf{ else } y := y - 1 \textbf{ fi}$$

(see [16, chap. 9]).

We now wish to prove in T that $\{\text{true}\}\ S\ \{y = 0\}$ holds. To this end we assume L contains equality symbols of all sorts, the language of Peano arithmetic (with a predecessor function), a one-argument (conversion) function $\bar{-}$ of sort (*data*, *ord*), and a constant ω of sort *ord*. In the proof we use logical consequences of the laws of equality, the axioms of Peano arithmetic, and the following formulas:

(1) $\bar{0} = 0$.
(2) $\forall x, y.(\bar{x} = \bar{y} \to x = y)$.
(3) $\forall x, y.(\bar{x} < \bar{y} \leftrightarrow x < y)$.
(4) $\forall x.(\bar{x} < \omega)$.

Define $p(\alpha)$ by

$$p(\alpha) \equiv (x = 0 \to \alpha = \omega) \wedge (x \neq 0 \to \alpha = \bar{y}).$$

Intuitively speaking, for a state σ, $p(\alpha)(\sigma)$ holds if α is the smallest ordinal bigger than or equal to the number of possible iterations performed by the loop when started in σ.

We now show that $p(\alpha)$ is a loop invariant (to apply the **while** rule).

(1) Clearly $p(\alpha) \wedge 0 < \alpha \to x = 0 \vee 0 < y$.
(2) We first show

$$\{p(\alpha) \wedge 0 < \alpha \wedge x = 0\} y := ?; x = 1\ \{\exists \beta < \alpha.p(\beta)\} \qquad (*)$$

by showing that

$$[p(\alpha) \wedge 0 < \alpha \wedge x = 0] \to \forall y.\exists \beta < \alpha.p(\beta)[1/x]$$

is true: $p(\alpha) \wedge x = 0$ implies $\alpha = \omega$. So for any y put $\beta = \bar{y}$: then $\beta < \alpha$ and $p(\beta)[1/x]$ holds.

Next we show

$$\{p(\alpha) \wedge 0 < \alpha \wedge x \neq 0\}\ y := y - 1\ \{\exists \beta < \alpha.p(\beta)\}. \qquad (**)$$

We have

$$\begin{aligned} p(\alpha) \wedge 0 < \alpha \wedge x \neq 0 &\to \alpha = \bar{y} \wedge 0 < y \wedge x \neq 0 \\ &\to \alpha = \bar{y} \wedge 0 < y \wedge p(\overline{y - 1})[y - 1/y] \\ &\to \exists \beta < \alpha.p(\beta)[y - 1/y], \end{aligned}$$

which justifies $(**)$.

From $(*)$ and $(**)$ by the **if–then–else** rule

$$\{p(\alpha) \wedge 0 < \alpha\}\ S_0\ \{\exists \beta < \alpha.p(\beta)\}$$

holds.

(3) Clearly $p(0) \to (x \neq 0 \wedge y = 0) \to -B$.

By the **while** rule

$$\{\exists\alpha.p(\alpha)\}\ S\ \{p(0)\}$$

holds. Both $\exists\alpha.p(\alpha)$ and $p(0) \rightarrow y = 0$ hold: So by the consequence rule

$$\{\text{true}\}\ S\ \{y = 0\}$$

also holds.

It is easy to sketch why **while** rule II is not sufficient to prove the formula $\{\text{true}\}\ S\ \{y = 0\}$ from arithmetical assumptions. Suppose otherwise. For some formula $p(\alpha)$, we would then have

(i) The following formulas are true:

$$p(\alpha + 1) \rightarrow B, \qquad p(0) \rightarrow \neg B, \qquad \exists\alpha p(\alpha), p(0), \rightarrow y = 0.$$

(ii) $\{p(\alpha + 1)\}\ S_0\ \{p(\alpha)\}$ holds.

Soundness of T with rule (5) replaced by rule (7) implies that $\{p(\alpha + 1)\}\ S_0\ \{p(\alpha)\}$ is true when interpreted in the domain of natural numbers. Now take a state σ; for some α_0, $p_0(\alpha_0)(\sigma)$ holds. It is now easy to see that (i) and (ii) imply that α_0 is equal to the number of loop iterations performed by the program S when started in σ. However, this is not true as for the state σ satisfying $x = 0$; such a number α_0 does not exist.

It is also clear that a modification of rule (7) obtained by replacing $\{p(\alpha + 1)\}\ S\ \{p(\alpha)\}$ by $\{p(\alpha) \wedge 0 < \alpha\}\ S\ \{\exists\beta < \alpha.p(\beta)\}$ does not save the situation either. These remarks are greatly generalized in Theorem 5.4 below.

The use of parameterized loop invariants combines the technique of using loop invariants and loop counters. The **while** rule II thus uses integer-valued loop counters as opposed to the **while** rule from T, which uses ordinal-valued loop counters. The insufficiency of integer-valued loop counters to prove the above formula $\{\text{true}\}\ S\ \{y = 0\}$ was first observed by Back [9].

The use of ordinal-valued loop counters was in fact proposed already in Floyd [18]. In the proof-theoretic framework it was first incorporated by Manna and Pnueli in [24], where so-called convergence functions, with range being a well-founded set, are used. In the framework of weakest precondition semantics, the use of ordinal-valued loop counters was advocated in Boom [11].

We now pass to the problem of soundness and completeness of T and consider interpretations I of L. These are ordinary two-sorted second-order structures, but subject to the following four conditions:

(1) The domain I_{data} of sort *data* is countable.
(2) The domain I_{ord} of sort *ord* is an initial segment of the ordinals.
(3) The constant 0 denotes the least ordinal, and the related symbol $<$ denotes the strict ordering of the ordinals, restricted to I_{ord}.
(4) The domain of each of the set sorts $(s_1, \ldots, s_n)$ contains all sets of the appropriate kind, that is, all subsets of $I_{s_1} \times \cdots \times I_{s_n}$, $n \geq 0$, $s_i \in \{data, ord\}$.

Let us fix on such an interpretation I and finish specifying the semantics of our programming language. The set of states is

$$X = I_{data}^k$$

(remember there are k variables x_j); we use σ to range over elements of the set of states. Let π range over maps from all L-variables, other than those in Var, to

elements of I-domains of the appropriate sort. We write

$$I \models_{\pi,\sigma} p$$

to mean that p is true in I when the free variables of p denote the values specified by π and σ; we write $I \models p$ for $\forall \pi, \sigma.\ I \models_{\pi,\sigma} p$. The notation $(\pi, \sigma)[Q_1/a_1, \ldots, Q_m/a_m, i_1/u_1, \ldots, i_n/u_n]$ is used to denote π', σ', where π' is obtained from π by altering its value at each a_k to Q_k and at each u_l, not in Var, to i_l; σ' is obtained by similarly altering the values of σ at every coordinate j, to i_j if u_j is x_j. We make use of similar notation for π and σ alone.

The truth relation $I \models_{\pi,\sigma} p$ is defined in the usual way by induction. The only nonstandard case is when p is of the form $(t_1, \ldots, t_n) \in \mu a(u_1, \ldots, u_n).r$. We then put

$$I \models_{\pi,\sigma} p \qquad \text{iff} \qquad I \models_{\pi[R/a],\sigma} a(t_1, \ldots, t_n),$$

where

$$R = \mu Q \subseteq (I_{sort_1} \times \cdots \times I_{sort_n}).\{\langle i_1, \ldots, i_n\rangle \mid I \models_{(\pi,\sigma)[Q/a, i_1/u_1, \ldots, i_n/u_n]} r\}.$$

Here $sort_i$ is the sort of variable u_i, and we use the μ notation of Section 2.

So $I \models_{\pi,\sigma} p$ holds iff the denotation of the tuple $(t_1, \ldots, t_n)$ belongs to the least fixed point of the operator Φ, defined over I by the formula r (and the variables a, $u_1, \ldots, u_n$). That is,

$$\Phi(Q) = \{\langle i_1, \ldots, i_n\rangle \mid I \models_{(\pi,\sigma)[Q/a, i_1/u_1, \ldots, i_n/u_n]} r\}.$$

We omit the (routine) proof that Φ is monotonic since a occurs positively in the formula r.

The definition of $\mathscr{B}$: BExp $\to (X \to \{tt, ff\})$ is now obvious, and for $\mathscr{A}$ we have

$$\mathscr{A}[\![x := t]\!](\sigma) = \{\sigma[I[\![t]\!](\sigma)/x]\},$$

using an obvious notation and

$$\mathscr{A}[\![x := ?]\!](\sigma) = \{\sigma' \mid \exists i \in I_{data}.\sigma' = \sigma[1/x]\}$$

(note that condition 1 on the countability of I_{data} is implicitly used here).

Now all four semantics considered in the previous section are at our disposal; we concentrate on the weakest precondition semantics $\mathscr{W}$. For the true of Hoare assertions we put, for any p, π,

$$[p]_\pi = \{\sigma \mid I \models_{\pi,\sigma} p\},$$

and then

$$\models_1 \{p\}\ S\ \{q\} \qquad \text{iff} \qquad \forall \pi.[p]_\pi \subseteq \mathscr{W}[\![S]\!][q]_\pi.$$

By Corollary 3.1 and Theorem 3.2 this is the same as

$$\models_I \{p\}\, S\, [q]$$
$$\text{iff} \quad \forall \pi, \sigma[\sigma \in [p]_\pi \to (S \text{ cannot diverge from } \sigma \wedge \forall \sigma'(\langle S, \sigma\rangle \to^* \langle \textbf{skip}, \sigma'\rangle \to \sigma' \in [q]_\pi)],$$

which is the usual definition of total correctness. We write $\mathrm{Th}(I)$ for the set of all sentences true in I (as usual); It will also prove helpful to use $[p]_{\pi/\lambda}$ as an abbreviation for $[p]_{\pi[\lambda/\alpha]}$.

We now prove the following soundness theorem.

SOUNDNESS THEOREM. *For any formulas p, q, of L and command S if*

$$Th(I)\ r\{p\}\ S\ \{q\},$$

then

$$\models_I \{p\}\ S\ \{q\}.$$

PROOF. We only consider the cases of the random assignment axiom and the **while** rule, since the others are standard.

(1) We have to show that for any p such that x is not free in p, and for any π

$$[p]\pi \subseteq \mathscr{W}[\![x := ?]\!]([p]_\pi).$$

It is easy to see that for any π

$$\begin{aligned}\mathscr{W}[\![x := ?]\!]([p]_\pi) &= \{\sigma \mid \forall i \in I_{data},\ \sigma[i/x] \in [p]_\pi\} \\ &= [p]_\pi \quad (\text{since } x \text{ is not free in } p).\end{aligned}$$

(2) Assume

(i) $I \models p(\alpha) \wedge 0 < \alpha \rightarrow B$;
(ii) $\forall\pi\{[p(\alpha) \wedge 0 < \alpha] \subseteq \mathscr{W}[\![S]\!]([\exists\beta < \alpha.p(\beta)]_\pi)\}$;
(iii) $i \models p(0) \rightarrow \neg B$.

We are to show that

$$\forall\pi\{[\exists\alpha.p(\alpha)]_\pi \subseteq \mathscr{W}[\![\textbf{while } B \textbf{ do } S \textbf{ od}]\!]([p(0)]_\pi)\}.$$

Now fix an arbitrary π.

By the definition of $\mathscr{W}$ we have

$$\mathscr{W}[\![\textbf{while } B \textbf{ do } S \textbf{ od}]\!]([p(0)]_\pi) = \mu Q.\Phi(Q),$$

where

$$\Phi(Q) = ([B]_\pi \cap \mathscr{W}[\![S]\!](Q)) \cap ([\neg B]_\pi \cap [p(0)]_\pi).$$

Assume by induction hypothesis that, for $\kappa < \lambda$, (κ, $\lambda \in I_{ord}$), we have

$$[p]_{\pi/\kappa} \subseteq \mu Q.\Phi(Q).$$

Thus

$$\bigcup_{\kappa<\lambda} [p]_{\lambda/\kappa} \subseteq \mu Q.\Phi(Q),$$

where, by convention, the set on the left-hand side is empty if $\lambda = 0$. By the monotonicity of Φ

$$\Phi\left(\bigcup_{\kappa<\lambda} [p]_{\pi/\kappa}\right) \subseteq \mu Q.\Phi(Q).$$

On the other hand, we have by (i)–(iii)

$$[p]_{\pi/\lambda} \subseteq \Phi\left(\bigcup_{\kappa<\lambda} [p]_{\pi/\lambda}\right),$$

so finally

$$[p]_{\pi/\lambda} \subseteq \mu Q.\Phi(Q),$$

that is,

$$[\exists \alpha . p(\alpha)]_\pi \subseteq \mu Q . \Phi(Q),$$

as required. □

We now pass to the problem of completeness of our proof system. First we introduce some notation. If $p(a)$ and $q(u_1, \ldots, u_n)$ are formulas such that $a(u_1, \ldots, u_n)$ is a well-formed atomic formula and a does not occur free in q, then by $p[a \backslash q(u_1, \ldots, u_n)]$ we denote a formula obtained from p in the following way. First rename all variables of p bound by quantifiers or the least fixed-point operator μ that happen to occur free in $\forall u_1, \ldots, u_n q(u_1, \ldots, u_n)$. Then replace each subformula $a(t_1, \ldots, t_n)$ of $p(a)$ by $q(t_1, \ldots, t_n)$, meaning the formula obtained from q by substituting $t_1, \ldots, t_n$ for $u_1, \ldots, u_n$, renaming bound variables of q if need be. The following lemma clarifies this definition.

LEMMA 4.1. *We have*

$$I \vDash_{\pi,\sigma} p[a \backslash q(u_1, \ldots, u_n)] \quad \textit{iff} \quad I \vDash_{\pi[A/a],\sigma} p,$$

where

$$A = \{\langle i_1, \ldots, i_n \rangle \mid I \vDash_{(\pi,\sigma)[i_1/u_1,\ldots,i_n/u_n]} q(u_1, \ldots, u_n)\}.$$

PROOF. The proof proceeds by induction on the structure of the formula p and the details are omitted. □

As with the remarks on the truth definition of μ formulas, we see that any formula $p(a, x_1, \ldots, x_k)$, where a is a k-ary set variable of sort $(data)^k$ and where $x_1, \ldots, x_k$ are the elements of Var defines for any π an operator $\{p\}_\pi : \mathscr{P}(X) \to \mathscr{P}(X)$ where

$$\{p\}_\pi(Q) = \{\sigma \in X \mid I \vDash_{\pi[Q/a],\sigma} p(a, x_1, \ldots, x_k)\}.$$

And if a always occurs positively in p, then $\{p\}_\pi$ is a monotonic operator.

Recall that if $\{p\}_\pi$ is monotonic, then $\{p\}_\pi^\lambda$ is defined by induction by the formula

$$\{p\}_\pi^\lambda = \{p\}_\pi\left(\bigcup_{\kappa<\lambda} \{p\}_\pi^\kappa\right).$$

Since X is countable, the sequence $\langle \{p\}_\pi^\lambda \rangle_\lambda$ stabilizes at a countable ordinal.

In the subsequent investigations we need to consider a formula defining this sequence. The other result needed concerns definability of the weakest precondition semantics in the assertion language. It turns out that these definitions can be made in a greatly restricted subset of the language. We say that a formula p is *positive* iff in every subformula of the form $(t_1, \ldots, t_n) \in \mu a(x_1, \ldots, x_n)$. q occurs positively in p, every existential ordinal quantifier occurs positively, and every universal ordinal quantifier occurs negatively; and we say that a *data* formula is one whose symbols have sorts involving only *data*.

LEMMA 4.2. DEFINABILITY LEMMA

(1) *For each command S there exists a positive data formula $\Phi_S(a, x_1, \ldots, x_k)$ with free variables as indicated, where a is a k-ary set variable that always occurs positively in Φ_S such that for any $R \subseteq X$,*

$$\sigma \in \mathscr{W}[\![S]\!](R) \quad \textit{iff} \quad I \vDash_{\pi[R/a],\sigma} \Phi_S(a, x_1, \ldots, x_k).$$

(2) *For each formula* $p(a, x_1, \ldots, x_k)$ *with k-ary set variable a always occurring positively, there exists a formula* $q(\alpha, x_1, \ldots, x_k)$ *with, apart from* α *and a, the same free variables as p such that for any* π *and* λ *in* I_{ord}

$$\sigma \in \{p\}^{\lambda}_{\pi} \quad \textit{iff} \quad I \vDash_{\pi[\lambda/\alpha],\sigma} q(\alpha, x_1, \ldots, x_k).$$

Furthermore, if p is positive, so is q.

PROOF

(1) Note first that, by assumption, all variables occurring in S are from the set $\mathrm{Var} = \{x_1, \ldots, x_k\}$. If S is $x_i := ?$, then Φ_S is

$$\forall z.a(x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_k) \wedge x_i = x_i.$$

If S is $x_i := t$, then Φ_S is

$$a(x_1, \ldots, x_{i-1}, t, x_{i+1}, \ldots, x_k) \wedge x_i = x_i.$$

If S is $S_1 ; S_2$, then Φ_S is

$$\Phi_{S_1}[a \backslash \Phi_{S_2}(y_1, \ldots, y_k)].$$

If S is **if** B **then** S_1 **else** S_2 **fi**, then Φ_S is

$$(B \rightarrow \Phi_{S_1}(a, x_1, \ldots, x_k)) \wedge (\neg B \rightarrow \Phi_{S_2}(a, x_1, \ldots, x_k)).$$

If S is **while** B **do** S_1 **od**, then Φ_S is

$$(x_1, \ldots, x_k) \in \mu b(x_1, \ldots, x_k).[(\neg B \wedge a(x_1, \ldots, x_k)) \vee (\Phi_{S_1}(b, x_1, \ldots, x_k) \wedge B)],$$

where b is a fresh k-ary set variable.

The claim can be straightforwardly justified on the ground of the definition of $\mathscr{W}$. It is clear that all formulas constructed are positive-data formulas.

(2) For any subset $Q' \subset I_{ord} \times X$, let

$$Q'(\lambda) = \{\sigma \mid \langle \lambda, \sigma \rangle \in Q'\}.$$

Consider the following equation:

$$R = \left\{ \langle \lambda, \sigma \rangle \mid \lambda \in I_{ord}, \sigma \in \{p\}_{\pi}\left(\bigcup_{\beta<\lambda} R(\beta) \right) \right\}.$$

Note that the set

$$Q = \{\langle \lambda, \sigma \rangle \mid \lambda \in I_{ord}, \sigma \in \{p\}^{\lambda}_{\pi}\}$$

satisfies this equation by the definition of $\{p\}^{\lambda}_{\pi}$. Moreover, it is the only set that satisfies this equation; if some Q' satisfies it as well, then we have, by transfinite induction, $Q'(\lambda) = Q(\lambda)$ for all $\lambda \in I_{ord}$ so that $Q' = Q$. We also have for any $\lambda \in I_{ord}$

$$(*) \qquad \bigcup_{\beta<\lambda} Q(\beta) = \{\sigma \mid I \vDash_{\pi[Q/b][\lambda/\alpha],\sigma} \exists \beta < \alpha b(\beta, x_1, \ldots, x_k)\},$$

where the variables b and α do not occur in p. Now

$$\langle \lambda, \sigma \rangle \in Q,$$

iff

$$I \vDash_{\pi[\bigcup_{\beta<\lambda} Q(\beta)/a],\sigma} p(a, x_1, \ldots, x_k),$$

iff

$$I \models_{\pi[Q/b][\lambda/\alpha][\cup_{\beta<\lambda}Q(\beta)/a],\sigma} p(a, x_1, \ldots, x_k),$$

iff (by Lemma 4.1 and (*))

$$I \models_{\pi[Q/b][\lambda/\alpha],\sigma} p[a\backslash\exists\beta < \alpha b(\beta, x_1, \ldots, x_k)](x_1, \ldots, x_k).$$

We thus see that Q is the only set satisfying the equation

$$R = \{\langle \lambda, \sigma\rangle \mid I \models_{\pi[R/b][\lambda/\alpha],\sigma} p[a\backslash\exists\beta < \alpha b(\beta, x_1, \ldots, x_k)](x_1, \ldots, x_k).$$

Now let

$$q(\alpha, x_1, \ldots, x_k) =_{\text{def}} (\alpha, x_1, \ldots, x_k) \in \mu b(\beta, x_1, \ldots, x_k) p[a\backslash\exists\beta < \alpha b(\beta, x_1, \ldots, x_k)].$$

(Note this is positive if p is.)

We have by the definition for any π and $\lambda \in I_{ord}$

$$I \models_{\pi[\lambda/\alpha],\sigma} q(\alpha, x_1, \ldots, x_k),$$

iff

$$I \models_{\pi[Q/b][\lambda/\alpha],\sigma} p[a\backslash\exists\beta < \alpha b(\beta, x_1, \ldots, x_k)](x_1, \ldots, x_k),$$

iff

$$\langle \lambda, \sigma\rangle \in Q,$$

iff

$$\sigma \in \{p\}^{\lambda}_{\pi},$$

which concludes the proof. □

Note that in part (2), the formula q only defines $\{p\}^{\lambda}_{\pi}$ insofar as this is possible with the ordinals in I_{ord}. We now impose a condition in the interpretation to the effect that we have enough such ordinals. Let $\kappa(I)$ be the supremum of the closure ordinals of the $\{p\}_{\pi}$, where $p(a, x_1, \ldots, x_n)$ is a positive data formula with free variables as indicated, including the k-ary set variable a. Then the assumption is

(3) The domain I_{ord} contains all ordinals strictly less than $\kappa(I)$, together with $\kappa(I)$ if that is not a limit ordinal.

An example interpretation satisfying these conditions can be constructed by chosing I_{data} and the interpretation of the constant, function, and predicate symbols whose sort involves only *data*; this determines the $\{p\}_{\pi}$ and we can then take I_{ord} to be $\kappa(I) + 1$; one can then fill in the interpretation of 0, <, and the other symbols.

Of course, we could just have taken I_{ord} to be the countable ordinals from the beginning and avoided all the fuss. But we can now obtain quite a strong completeness theorem.

COMPLETENESS THEOREM. *For any command S and formulas p, q if $\models_I \{p\}\ S\ \{q\}$, then $Th(I) \vdash_T \{p\}\ S\ \{q\}$. Furthermore, if p, q are positive, then, in the applications of the proof rules involved in proving $\{p\}\ S\ \{q\}$, only positive formulas need to be chosen to instantiate the formula variables.*

PROOF. Assume $\models_I \{p\}\, S\, \{q\}$ holds. By induction on the structure of S we prove that $\mathrm{Th}(I) \vdash_T \{p\}\, S\, \{q\}$ holds.

Suppose S is $x := ?$. Then, $I \models p \to \forall x q$ and, by the random assignment axiom, $\mathrm{Th}(I) \vdash_T \{\forall x.q\}\ x := ?\ \{\forall x.q\}$, so by the consequence rule, $\mathrm{Th}(I) \vdash_T \{p\}\ x := ?\ \{q\}$. (Note that $\forall x q$ is positive if q is.)

The case of the assignment statement and the **if–then–else** construct are clear.

Suppose that S is $S_1\ ;\ S_2$. It is easy to see that both $\models_I \{p\}\ S_1\ \{r\}$ and $\models_I \{r\}\ S_2\ \{q\}$ hold for any formula r such that for all π

$$[r]_\pi = \mathscr{W}[\![S_2]\!]([q]_\pi).$$

The existence of such a formula follows from Lemma 4.2.1: We can take for r the formula

$$\Phi_{S_2}[a \backslash q(x_1, \ldots, x_k)]$$

(which is positive if q is). By the induction hypothesis and composition and the consequence rule

$$\mathrm{Th}(I) \vdash \{p\}\ S_1\ ;\ S_2\ \{q\}.$$

The case in which S is **while** B **do** S_1 **od** is the most complicated one. In order to prove $\{p\}\ S\ \{q\}$, we have to apply the **while** rule and for this purpose find an appropriate loop invariant $r(\alpha)$.

Consider first the positive data formula

$$p_1(a, x_1, \ldots, x_k) \equiv \neg B \vee \Phi_{S_1}(a, x_1, \ldots, x_k),$$

where a does not appear free in q or B. According to Lemma 4.2.2, the sequence $\langle \{p_1\}^\lambda_\pi \rangle_\lambda$ can be defined by a positive formula $q_1(\alpha)$. More precisely, there exists a positive formula $q_1(\alpha, x_1, \ldots, x_k)$ such that for any $\lambda \in I_{ord}$

$$I \models_{\pi[\lambda/\alpha], \sigma} q_1(\alpha, x_1, \ldots, x_k) \qquad \text{iff} \qquad \sigma \in \{p_1\}^\lambda_\pi.$$

Now define r by

$$r(\alpha, x_1, \ldots, x_k) =_{\mathrm{def}} q_1(\alpha, x_1, \ldots, x_k) \wedge \Phi_S[a \backslash q(x_1, \ldots, x_k)] \wedge (0 < \alpha \leftrightarrow B)$$

(and note that r is positive if q is). We now prove that r is a loop invariant.

To this end we have to establish three facts:

(i) $$I \models r(0) \to \neg B.$$

(ii) $$I \models \{r(\alpha) \wedge 0 < \alpha\}\ S_1\ \{\exists \beta < \alpha r(\beta)\}.$$

(iii) $$I \models r(\alpha) \wedge 0 < \alpha \to B.$$

Both (i) and (iii) follow from the definition of $r(\alpha)$. To prove (ii), we have to show that for any $\lambda > 0$, $\lambda \in I_{ord}$, and π

$$[r]_{\pi/\lambda} \subseteq \mathscr{W}[\![S_1]\!]([\exists \beta < \alpha.r(\beta)]_{\pi/\lambda}).$$

Assume that for any λ and π

$$\bigcup_{\kappa<\lambda} [r]_{\pi/\kappa} = \bigcup_{\kappa<\lambda} \{p_1\}^\kappa_\pi \cap \mathscr{W}[\![S]\!]([q]_\pi).$$

We have by the definition of r

$$[r]_{\pi/\lambda} = \{p_1\}^{\lambda}_{\pi} \cap \mathscr{W}[\![S]\!]([q]_{\pi}) \cap [B]_{\pi}$$

(by Lemmas 4.1 and 4.2.1 and as $\lambda > 0$)

$$= \{p_1\}_{\pi}\left(\bigcup_{\kappa<\lambda} \{p_1\}^{\kappa}_{\pi}\right) \cap \mathscr{W}[\![S]\!]([q]_{\pi}) \cap [B]_{\pi}$$

(by the definition of $\{p_1\}^{\lambda}_{\pi}$)

$$= \left([\neg B]_{\pi} \cup \mathscr{W}[\![S_1]\!]\left(\bigcup_{\kappa<\lambda} \{p_1\}^{\kappa}_{\pi}\right)\right) \cap \mathscr{W}[\![S]\!]([q]_{\pi}) \cap [B]_{\pi}$$

(by the definition of p_1, and Lemmas 4.1 and 4.2.1)

$$= \mathscr{W}[\![S_1]\!]\left(\bigcup_{\kappa<\lambda} \{p_1\}^{\kappa}_{\pi}\right)\right) \cap \mathscr{W}[\![S_1]\!](\mathscr{W}[\![S]\!]([q]_{\pi})) \cap [B]_{\pi}$$

(by the definition of $\mathscr{W}[\![S]\!]$)

$$= \mathscr{W}[\![S_1]\!]\left(\left(\bigcup_{\kappa<\lambda} \{p_1\}^{\kappa}_{\pi}\right) \cap \mathscr{W}[\![S]\!]([q]_{\pi})\right) \cap [B]_{\pi}$$

(as $\mathscr{W}[\![S]\!]$ is a predicate transformer)

$$\subseteq \mathscr{W}[\![S_1]\!]\left(\bigcup_{\kappa<\lambda} [r]_{\pi/\kappa}\right)$$ (by the above assumption).

It remains to prove the above assumption.

To this end it is enough to prove by induction on κ that

$$(*) \qquad [r]_{\pi/0} \cup [r]^{0}_{\pi/\kappa} = (\{p_1\}^{0}_{\pi} \cap \{p_1\}^{\kappa}_{\pi}) \cap \mathscr{W}[\![S]\!]([q]_{\pi}).$$

In the case $\kappa = 0$ we see by the definition of p_1 that

$$\{p_1\}^{0}_{\pi} = [\neg B]_{\pi} \cup \mathscr{W}[\![S_1]\!](\varnothing)$$
$$= [\neg B]_{\pi}$$ (by the law of the excluded miracle),

and so, by the definition of r that

$$[r]_{\pi/0} = [\neg B]_{\pi} \cap \mathscr{W}[\![S]\!]([q]_{\pi}),$$

establishing (*) for $\kappa = 0$.

In the case $\kappa > 0$ we calculate

$$[r]_{\pi/0} \cup [r]_{\pi/\kappa} = ([\neg B]_{\pi} \cap \mathscr{W}[\![S]\!]([q]_{\pi})) \cup (\{p_1\}^{\kappa}_{\pi} \cap \mathscr{W}[\![S]\!]([q]_{\pi}) \cap [B]_{\pi})$$
$$= ([\neg B]_{\pi} \cup \{p_1\}^{\kappa}_{\pi}) \cap \mathscr{W}[\![S]\!]([q]_{\pi})$$
$$= (\{p_1\}^{0}_{\pi} \cup \{p_1\}^{\kappa}_{\pi}) \cap \mathscr{W}[\![S]\!]([q]_{\pi}).$$

From (i)–(iii), it follows by the **while** rule and the induction hypothesis that

$$\mathrm{Th}(I) \vdash_T \{\exists \alpha.r(\alpha)\}\ \textbf{while}\ B\ \textbf{do}\ S_1\ \textbf{od}\ \{r(0)\}.$$

On the other hand

$$\models_I \{p\}\ \textbf{while}\ B\ \textbf{do}\ S_1\ \textbf{od}\ \{q\}$$

and so

$$\begin{aligned}
\forall\pi.[p]_\pi &\subseteq \mathscr{W}[\![\textbf{while } B \textbf{ do } S_1 \textbf{ od}]\!]([q]_\pi) \\
&= \mathscr{W}[\![S]\!](X) \cap \mathscr{W}[\![S]\!]([q]_\pi) \\
&= \mathscr{W}[\![S]\!](X) \cap \mathscr{W}[\![S]\!]([q]_\pi) \\
&= (\mu R.\{p_1\}_\pi(R)) \cap \mathscr{W}[\![S]\!]([q]_\pi) \quad \text{(by definition of } \mathscr{W}) \\
&= \bigcup_{\lambda \in I_{\text{ord}}} \{p_1\}_\pi^\lambda \cap \mathscr{W}[\![S]\!]([q]_\pi) \quad \text{(by assumption 5 on } I) \\
&= \bigcup_{\lambda \in I_{\text{ord}}} [r]_{\pi/\lambda} \quad \text{(by the assumption established above)} \\
&= [\exists.r(\alpha)]_\pi.
\end{aligned}$$

So

$$I = p \to \exists\alpha.r(\alpha).$$

Also

$$[r]_{\pi/0} = [\neg B]_\pi \cap \mathscr{W}[\![S]\!]([q]_p) \quad \text{(by the definition of } \mathscr{W}[\![S]\!]).$$

Thus by the consequence rule

$$\text{Th}(I) \vdash_T \{p\} \textbf{ while } B \textbf{ do } S_1 \textbf{ od } \{q\} \text{ as required.} \qquad \square$$

The use of such a powerful language as our μ-calculus contrasts with the usual situation [2, 20] in which a first-order language containing the language of Peano arithmetic suffices to obtain completeness. As we see in the next section, that will not do here. The μ-calculus gives the needed extra expressive power, as evidenced by the Definability Lemma; with this, no extra expressiveness assumption on the interpretation is needed. On the other hand, the calculus has somewhat too much expressive power, and we have given a sublanguage—the positive formulas—for which completeness still holds; in the next section, we give some evidence that this language is of the right recursion-theoretic strength. The only real difference between our language and that of Hitchcock and Park [23] lies in the use of ordinals. Now using the Definability Lemma we can see that

$$\vDash_I \{p\}\ S\ \{q\} \qquad \text{iff} \qquad I \vDash p \to \Phi_S[a \backslash q(x_1, \ldots, x_k)],$$

and so we have the following sound and complete "proof system":

$$\frac{p \to \Phi_S\ [a \backslash q(x_1, \ldots, x_k)]}{\{p\}\ S\ \{q\}}$$

for all formulas p and q; further, it does not use the ordinal sort at all, as long as p and q contain no symbols involving this sort. Thus, in principle, we can work entirely within Hitchcock and Park's language. However, we wanted to obtain a natural proof system, with syntax-directed proof rules; we also wanted to investigate the natural generalization of the standard **while** rule II to ordinals. It should be possible to find a system using definable well-founded relations rather than ordinals, and thereby work entirely within Hitchcock and Park's language. It might also be of interest to see if a completeness result is possible using a two-sorted first-order language of data and ordinals.

5. *Recursion-Theoretic Results*

In this section we gather results concerning the recursion-theoretic complexity of the constructs studied in the previous sections. To this purpose we assume that the domain of data values I_{data} is N, the set of natural numbers.

We fix the programming language as in Section 4 (but without limiting ourselves to any particular set of variables). Additionally, we adopt the following reasonable assumption: All functions and relations used in the expressions are recursive (i.e., effectively calculable), and the usual functions and relations of Peano arithmetic are available in the language.

In 5.1 we show that the halting problem for our language is Π^1_1 complete (Theorem 5.2), also noting what ordinals are associated with the computation trees of always halting computations (Theorem 5.1). In 5.2 we characterize the state transformations and predicate transformers definable in our language (Theorem 5.2). These last two results are restatements within our framework of Chandra's Theorem 2 [13]. Finally in 5.3 we discuss the recursion-theoretic complexity of our assertion language and also refine the completeness theorem of the last section showing (Theorem 5.3) what ordinals are needed in proofs.

5.1. THE HALTING PROBLEM. In the subsequent considerations we use various results from recursion theory. To make the paper self-contained, we briefly recall the definitions and results we need. All of them can be found, for example, in [32]. We assume the standard coding mechanism that assigns to each finite sequence $m_1, \ldots, m_k$ of natural numbers its code $\langle m_1, \ldots, m_k \rangle$ being a natural number; if m is the code of $m_1, \ldots, m_k$, then $m \otimes n$ is the code of $m_1, \ldots, m_k, n$. By $\{m\}$ we denote the partial recursive function with index m.

Definition 5.1. A *tree* is a set of finite sequences of natural numbers, closed under subsequences, partially ordered under the relation *is an extension of*. A tree is *well-founded* if it does not contain an infinite descending sequence, that is, if its partial ordering is well founded. A tree is *recursive* if the set of codes of its elements is recursive.

With each well-founded relation we can associate ordinals in a standard way. First ordinal 1 is attached to all its minimal elements. Next we proceed by a transfinite induction and attach to a nonminimal element a, the ordinal $\text{Ord}(a)$ being $\sup(\{\text{Ord}(b) + 1 \mid b < a\})$, which is the least ordinal greater than any attached to a lower element. When the partial ordering is a tree, the ordinal attached to the root of the tree (if nonempty, and zero, otherwise) is the ordinal associated with the tree, called the *height* of the tree.

Now an ordinal is *recursive* if it is a height of a recursive well-founded tree. There are many equivalent definitions of the recursive ordinals, and the reader is referred to [32] for more information.

A set $A \subseteq N$ is Π^1_1 if for some first-order definable (in the language of second-order Peano arithmetic with relation variables [4]) relation $R \subseteq \mathcal{N} \times N$

$$m \in A \leftrightarrow \forall a\, R(a, m),$$

for all $m \in N$. (Here $\mathcal{N}$ is the set of all subsets of N.) A set A is Σ^1_1 iff $N - A$ is Π^1_1.

Now let

$$T = \{m \mid \{m\} \text{ is the characteristic function of a well-founded recursive tree}\}.$$

The following fact will be needed in what follows.

FACT 5.1. (Rogers [32]): *T is a complete Π^1_1 set (completeness here meaning that any other Π^1_1 set is recursively one-one reducible to T).*

We now relate the material on recursive ordinals and Π^1_1 predicates to our programming language. For any configuration $\langle S, \sigma \rangle$ the converse of the relation $\rightarrow^*$ restricted to $\{\langle S', \sigma' \rangle \mid \langle S, \sigma \rangle \rightarrow^* \langle S', \sigma' \rangle\}$ is well-founded iff S cannot diverge from σ, as described above. The ordinal so associated is clearly recursive as it is also the ordinal of the *execution tree*, Exec(S, σ), of $\langle S, \sigma \rangle$. This tree is by definition the set of finite sequences $\langle S, \sigma \rangle = \langle S_0, \sigma_0 \rangle \rightarrow \cdots \rightarrow \langle S_n, \sigma_n \rangle$ ordered by the supersequence ordering; it can be considered a tree in the sense of Definition 5.1 via a standard coding of configurations, and as such the explicit description of the operational semantics given in Section 3 makes it clear that it is a recursive tree.

Conversely, we can write a command to search a given recursive tree systematically. Let Tree be the command written informally as

$y := 0$; $u := \langle\ \rangle$;
while $y = 0$ **do**
  if $\{x\}\ u = 0$ **then** $y := 1$ **fi**;
  $v := ?\ u := u \circledast v$;
od;
$u := ?$; $u := \{x\}u$

Note that the test of whether $\{x\}u = 0$ can take arbitrarily many steps and need not even terminate. Indeed Tree always terminates from σ iff $\sigma(x)$ is the code of a recursive well-founded tree (the last two commands are to check the totality of $\{\sigma(x)\}$). Moreover, consideration of the resulting execution tree shows that it must have an associated ordinal β with $\alpha < \beta \leq \omega.\alpha + 1$, where α is the ordinal of the well-founded tree coded by $\sigma(x)$. Summarizing this discussion, we have proved

THEOREM 5.1. *The ordinals associated with well-founded execution trees are recursive. Conversely, for any recursive ordinal, α, there is an execution tree with ordinal β with $\alpha < \beta \leq \omega.\alpha + 1$.*

A similar, but slightly weaker, result has been proved by J. Stavi. The reason for the "ω-blowup" of α is that we simulate passing from one node of the tree to another by finitely many steps of computation. Since $\langle S, \sigma \rangle$ always terminates iff its execution tree is well founded, we see that, modulo a standard coding, the halting set

$$H = \{\langle S, \sigma \rangle \mid \langle S, \sigma \rangle \text{ always terminates}\}$$

is Π^1_1. What is more, we have

THEOREM 5.2. *The set H is complete Π^1_1.*

PROOF. We have already noted that H is Π^1_1. Next we know from the above discussion that an integer m is in T iff the program Tree always terminates from $\sigma[m/x]$. The conclusion follows from Fact 5.1. □

An alternative characterization of Π^1_1 sets in terms of unbounded nondeterminism is provided by Harel and Kozen [21] where the key concept is the notion of acceptance instead of termination. They consider infinitely broad AND/OR trees and concentrate on definability issues only.

5.2 DEFINABILITY. We now characterize those state transformations and predicate transformers that are definable by commands. We make use of

commands with more than k variables and put $X_l = N^l$ (for $l \geq k$). Define the functions $X_k \xrightarrow{0} k$, $1 \to X_l \xrightarrow{\pi} l$, $k \to X_k$ for $1 \geq k$ by $0_{k,l}(x_1, \ldots, x_k) = (x_1, \ldots, x_k, 0, \ldots, 0)$ and $\pi_{l,k}(x_1, \ldots, x_l) = (x_1, \ldots, x_k)$.

Definition 5.2. A state transformation m in ET_{X_k,X_k} (respectively, ST_{X_k,X_k}) is *definable* iff there is a command S with $l \geq k$ variables such that $m = \pi^{\dagger}_{l,k} \circ \mathscr{D}_{\mathscr{E}}[\![S]\!] \circ 0_{k,l}$ (respectively, $\pi^{\dagger}_{l,k} \circ \mathscr{D}_{\mathscr{P}}[\![S]\!] \cdot 0_{k,l}$).

Thus, we allow ourselves extra variables; this is done to avoid coding problems and would clearly not be needed in an extension of our programming language, which had a block structure (for example).

Definition 5.3. A predicate transformer p is *definable* iff there is a command S with $l \geq k$ variables such that $p = \pi_{l,k} \circ \mathscr{W}[\![S]\!] \circ 0_{k,l}$ (where the evident extensions to sets of $\pi_{l,k}$ and $0_{k,l}$ are intended).

We need to write one more program.

LEMMA 5.1. *For any recursively enumerable relation $R \subseteq N^k \times N^k$, there is a command S_R defining the function f: $N^k \to \mathscr{E}(N^k_{\perp})$ where*

$$f(m_1, \ldots, m_k) = \begin{cases} \{(n_1, \ldots, n_k) \mid (R(m_1, \ldots, m_k, n_1, \ldots, n_k)\} & (\textit{if this is not } \varnothing) \\ \{\perp\} & (\textit{otherwise}). \end{cases}$$

PROOF. To every re set, there effectively corresponds a partial recursive function whose range is the given re set and which is total iff the set is nonempty. Therefore, there is a recursive function h such that $h(\langle m_1, \ldots, m_k \rangle)$ is an index of a recursive partial function that corresponds in this way to

$$\{\langle n_1, \ldots, n_k \rangle \mid R(m_1, \ldots, m_k, n_1, \ldots, n_k)\}.$$

So we can take S_R to be the command,

$$r := ?;\ x := \{h(\langle x_1, \ldots, x_k \rangle)\}(r);\ x_1 := (x)_1; \ldots; x_k := (x)_k. \qquad \square$$

THEOREM 5.3. DEFINABILITY

(1) *A state-transformation m in ET is definable iff R_m is recursively enumerable and $\{\langle m_1, \ldots, m_k \rangle \mid (m_1, \ldots, m_k) \in T_m\}$ is Π^1_1 (here $rel(m) = \langle R_m, T_m \rangle$ as in Section* 2.2).

(2) *A state-transformation, m, in ST is definable iff for some recursively enumerable R we have $R_m = R \cup (T'_m \times Y)$ and also $\{\langle m_1, \ldots, m_k \rangle \mid (m_1, \ldots, m_k) \in T_m\}$ is Π^1_1 (here $rel(m) = \langle R_m, T_m \rangle$ as in Section* 2.2).

(3) *A predicate transformer p is definable iff for some recursively enumerable R and Π^1_1 set T we have*

$$p(B) = \{(m_1, \ldots, m_k) \mid \langle m_1, \ldots, m_k \rangle \in T \textit{ and } R(m_1, \ldots, m_k) \subseteq B\}.$$

PROOF

(1) Suppose m is definable by a command S. Then, as the binary relation $\langle S, \sigma \rangle \to^* \langle \mathbf{skip}, \sigma' \rangle$ on states is clearly recursively enumerable, so is R_m. Also, since H is Π^1_1 from Theorem 5.2, so is T_m.

Conversely, suppose we have that R_m is recursively enumerable and that $\{\langle m_1, \ldots, m_k \rangle \mid (m_1, \ldots, m_k) \in T_m\}$ is Π^1_1. Since T is complete Π^1_1, there is a recursive function g such that $(m_1, \ldots, m_k) \in T_m$ iff $g(\langle m_1, \ldots, m_k \rangle) \in T$.

Now we can see that m is defined by the command

$v := ?$
if $v = 0$ **then** $x := g(\langle x_1, \ldots, x_k \rangle)$; Tree **fi**;
S_{R_m}

Note that the above assertions depend for their validity on Theorem 3.2.

(2) Immediate from Part (1) and the remarks on the relational approach in Section 2.2 and Theorem 3.1.

(3) Immediate from Part (2) and the remarks on the relational approach in Section 2.3 and Theorem 3.3. (One small point is that, given R and T satisfying the formula, it follows from the fact that $p(\varnothing) = \varnothing$ by the law of the excluded miracle that, if $\langle m_1, \ldots, m_k \rangle \in T$, then $R(m_1, \ldots, m_k) \neq \varnothing$.) □

Note that there is no conflict with Church's thesis, which only relates to computable partial functions. In fact it follows from Part (1) of Theorem 5.3 that any definable partial function is partial recursive. A more interesting question is whether to extend the notion of computability to the present kind of nondeterminism (Chandra's possibility (C) [13]). We incline to this view since the Π^1_1 phenomena arise from an abstraction from reality for which we do not care to specify which oracle (giving values for random assignments) actually occurs.

To relate Part (3) of the Theorem to a standard concept, recall that an operator $\Phi: \mathscr{P}(N^k) \to \mathscr{P}(N^k)$ is Π^1_1 iff, for some first-order definable (in the language of Peano arithmetic) relation $R \subseteq \mathscr{N} \times \mathscr{P}(N^k) \times N^k$,

$$\sigma \in \Phi(X) \equiv \forall a R(a, X, \sigma).$$

COROLLARY 5.1. *For any command S the predicate transformer $\mathscr{W}[\![S]\!]$ is a Π^1_1 monotonic operator.*

PROOF. Immediate from Part (3) of Theorem 5.3. □

5.3 PROOF THEORY. The assertion language we used in the proof of the completeness theorem in Section 4 is quite powerful. It is instructive to see that some simpler assertion languages are not sufficient to obtain completeness. Take, for example, the language of Peano arithmetic and its standard interpretation $\mathscr{N}$. Note that for any finitistic proof system G the set

$$\{\{p\}\ S\ \{q\} \mid \mathrm{Th}(\mathscr{N}) \vdash_G \{p\}\ S\ \{q\}\}$$

is recursively enumerable in a Δ^1_1 set and so is itself Δ^1_1.

On the other hand, the set

$$\{\{p\}\ S\ \{q\} \mid \vDash_{\mathscr{N}} \{p\}\ S\ \{q\}\}$$

is Π^1_1 complete since the Π^1_1 complete set H from Section 5.1 can be reduced to it by

$$\langle S, \sigma \rangle \in H \quad \text{iff} \quad \vDash_{\mathscr{N}} \{x_1 = \sigma_1 \wedge \cdots \wedge x_k = \sigma_k\}\ S\ \{\text{true}\},$$

and moreover the set itself is Π^1_1 since

$$\vDash \{p\}\ S\ \{q\} \quad \text{iff} \quad \forall \pi, \sigma[(\mathscr{N} \vDash_{\pi,\sigma} p \to \langle S, \sigma \rangle \in H) \wedge (\langle S, \sigma \rangle \to^* \langle \textbf{skip}, \sigma' \rangle \to \mathscr{N} \vDash_{\pi,\sigma'} q)].$$

This should be contrasted with the situation in the case of programs S admitting only bounded nondeterminism where there is completeness relative to $\mathrm{Th}(\mathscr{N})$ (see

[3]). Indeed then, the set

$$\{\{p\}\ S\ \{q\} \mid \models_{\mathscr{N}} \{p\}\ S\ \{q\}\}$$

is Δ^1_1, since the corresponding halting set H is then recursively enumerable.

The power of our assertion language is also reflected in the complexity of the truth relation of the standard model I_{st} in the case of an assertion language being an extension of Peano arithmetic.

The model I_{st} contains all natural numbers, recursive ordinals, and all sets of the appropriate kind. As we shall see, $\mathrm{Th}(I_{st})$ is at least Π^1_1 and, we conjecture, lies within Δ^1_2. We can characterize the complexity of the subcollection of positive formulas. Recall that a formula of second-order Peano arithmetic is Π^1_1 iff it has the form $\forall a.p$ where only first-order quantifiers appear in p.

LEMMA 5.2. *From every positive formula with no free ordinal variables, there is effectively obtainable a* Π^1_1 *formula of second-order arithmetic defining the same relation over the natural numbers and relations (using the standard model for both cases).*

PROOF. First we sketch how to translate the positive formulas into positive data formulas. To this end we interpret the ordinals as the set T of codes of characteristic functions of well-founded trees by means of a positive data formula $O(x)$, translating ordinal quantifiers $\forall\alpha$ and $\exists\alpha$ by the restricted natural number quantifiers $(\forall x.O(x) \to \cdots)$ and $(\exists x.O(x) \wedge \cdots)$. It is here that the restrictions on the occurrences of ordinal quantifiers in positive formulas are used to ensure the positivity of their translations. Then the ordinal constant 0 is interpreted by the code of the empty tree, and the relation $\alpha < \beta$ is interpreted by a formula $L(x, y)$ defining the relation $x <^* y$ over the integers (where $x <^* y$ iff x and y are in T and the height of the tree coded by x is strictly smaller than the height of the tree coded by y).

To define $O(x)$ within our assertion language, one notes that T is the least set of natural numbers such that

$$x \in T \quad \text{iff} \quad (\{x\}\langle\ \rangle = 0) \vee \forall y.(x \mid y) \in T,$$

where $(\cdot \mid \cdot)$ is a recursive function such that for any $n_1, \ldots, n_k$ we have

$$\{x \mid y\}\langle n_1, \ldots, n_k\rangle \simeq \{x\}\langle y, n_1, \ldots, n_k\rangle.$$

So $O(x)$ can be taken to be

$$x \in \mu a(x).(\{x\}\langle\ \rangle = 0 \vee \forall y.a(x \mid y),$$

making use of the usual abbreviating devices afforded by Peano arithmetic. For the formula $L(x, y)$, we let $x \leq^* Y$ mean that x and y are in T and $y <^* x$ does not hold, and we note that $<^*$ and $\leq^*$ are the least pair of relations such that

$$\begin{aligned} x <^* y \quad &\text{iff} \quad x \in T \wedge y \in T \wedge (\{y\}\langle\ \rangle = 1 \wedge \exists z.x \leq^* y \mid z), \\ x \leq^* y \quad &\text{iff} \quad x \in T \wedge y \in T \wedge (\{x\}\langle\ \rangle = 0 \vee \forall z.x \mid z <^* y); \end{aligned}$$

then we can use the least fixed-point operator to obtain a suitable formula $M(x, y, z)$, where $M(x, y, 0)$ defines $<^*$ and $M(x, y, 1)$ defines $\leq^*$, and that gives $L(x, y)$.

Finally we note that least fixed-point formulas $(t_1, \ldots, t_n) \in \mu a(x_1, \ldots, x_n)$ can be translated as

$$\forall a. \forall x_1 \cdots \forall x_n (p' \rightarrow a(x_1, \ldots, x_n)) \rightarrow a(t_1, \ldots, t_n)$$

(where p' is the translation of p), and one sees that applying this to a positive data formula results in a formula of second-order arithmetic that can be brought into Π^1_1 form. □

We see from the lemma, that since the true Π^1_1 sentences of second-order arithmetic form a Π^1_1 set, so do the positive sentences of our language. Furthermore, since the complete Π^1_1 set T is definable by a positive formula, we conclude that the sentences form a complete Π^1_1 set. If we are happy with the "proof system" outlined at the end of the previous section for formulas of the form $\{p\}\ S\ \{q\}$ with p and q positive formulas, then we see that, at least in the case of arithmetic, our assertion language, restricted to the positive formulas, has exactly the right recursion-theoretic strength. However, the completeness theorem for the Hoare logic merely gives a reduction to the truth of sentences of the form $\forall x_1, \ldots, \forall x_n (p \rightarrow q)$ where p and q are positive. The set of true sentences is more complicated than Π^1_1, since it includes the complete Σ^1_1 set of the formulas where q is $(0 = 1)$ (but at least it is Δ^1_2). We conjecture that this is the price paid for not having a structured proof system. Returning to the lemma, it can be shown that the converse also holds, and indeed what we have is a mild elaboration of the well-known fact that the inductive relations and the Π^1_1 relations coincide.

We now study which ordinals are needed to prove the total correctness of the programs considered. Let Ω denote the least nonrecursive ordinal. We need the following fact concerning monotonic Π^1_1 operators.

FACT 5.2 (Spector [34]). *If Φ is a Π^1_1 monotonic operator, then $|\Phi| \leq \Omega$ where $|\Phi|$ is the closure ordinal of Φ (see Section 2.1).*

We now prove the following theorem.

THEOREM 5.4. *Exactly all recursive ordinals are needed in the proofs of total correctness.*

PROOF. We show first that it is sufficient to restrict the range of ordinals in the **while** rule to all recursive ordinals.

By Lemma 5.2 every monotonic operator definable by a positive data formula is a Π^1_1 operator and hence, by Fact 5.2, has closure ordinal $\leq\Omega$, which is a limit ordinal. So if I_{ord} is Ω, assumption (5) on I is fulfilled, and the Completeness Theorem applies to I.

Suppose now for contradiction that for some recursive ordinal α_0 it is sufficient to restrict the range of ordinals needed in the **while** rule to ordinals $\leq\alpha_0$. In Section 5.1, we prove that for any recursive ordinal there exists a value m such that Exec(Tree, σ) has associated ordinal $>\alpha$ when $\sigma(x) = m$.

Choose now m_0 such that Exec(Tree, σ) is of height $>\omega.\alpha_0 + 2$ when $\sigma(x) = m_0$. We have (omitting to write I)

$$\vDash \{x = m_0 \wedge y = 0 \wedge u = \langle\ \rangle\}\ \textbf{while}\ y = 0\ \textbf{do}\ S'\ \textbf{od}\ \{\text{true}\},$$

where

$$S' \equiv \textbf{if}\ \{x\}(u) = 0\ \textbf{then}\ y := 1\ \textbf{fi};$$
$$v := ?;\ u := u \otimes v.$$

By the above claim there exists an assertion $r(\alpha)$ such that

(i) $x = m_0 \wedge y = 0 \wedge u = \langle\ \rangle \rightarrow \exists \alpha \leq \alpha_0 r(\alpha)$.
(ii) $r(\alpha) \wedge \alpha > 0 \rightarrow B, \quad r(0) \rightarrow \neg B$.
(iii) $\models \{r(\alpha) \wedge \alpha > 0\}\ S'\ \{\exists \beta < \alpha r(\beta)\}$.

Let σ be any state such that

$$\models_\sigma (x = m_0 \wedge y = 0 \wedge u = \langle\ \rangle) \quad \text{(omitting to write the unnecessary } \pi\text{)}.$$

Then Exec(**while** $y = 0$ **do** S' **od**, σ) has associated ordinal $>\omega.\alpha_0$. We now assign an ordinal to each node of this tree. By (i) for some α, $\models_\sigma r(\alpha)$. We assign $\omega \cdot \alpha$ to the root of the tree; clearly $\alpha > 0$. For any σ' such that

$$\langle S', \sigma\rangle \rightarrow^* \langle \textbf{skip}, \sigma'\rangle;$$

by (iii) $\models_{\sigma'} r(\beta)$ holds for some $\beta < \alpha$. To the corresponding node

$$\langle \textbf{while } y = 0 \textbf{ do } S' \textbf{ od},\rangle \rightarrow^* \langle \textbf{while } y = 0 \textbf{ do } S' \textbf{ od}, \sigma'\rangle,$$

we assign the value $\omega \cdot \beta$. Continuing this procedure from σ', we assign an ordinal to every node of this form. Now take a node τ to which no ordinal has been assigned. By the form of S' there exist two nodes τ_0 and τ_1 in the execution tree of the above form such that ordinals have been assigned to τ_0 and τ_1 and τ lies on a path connecting them. Let β be the ordinal assigned to τ_0 and let n be its distance from τ. We assign to τ the ordinal $\beta + n$.

In this way we find an order-preserving function from the execution tree into the ordinal $\alpha \cdot \alpha_0$. However, this is a contradiction because the least ordinal into which such a function exists is the height of this execution tree which is $>\omega \cdot \alpha_0$. □

ACKNOWLEDGMENTS. We are grateful to A. de Bruin, F. Nielson, and the referees for detailed comments on the first version of this paper. We thank J. Stavi for interesting discussions concerning the results contained in Section 5.

REFERENCES

1. ACZEL, P. An introduction to inductive definitions. In *Handbook of Mathematical Logic*, J. Barwise, Ed. North Holland Studies in Logic and the Foundations of Mathematics, vol. 90, Elsevier-North Holland, Amsterdam, 1977, pp. 739–792.
2. APT, K. R. Ten years of Hoare's logic: A survey—Part I. *ACM Trans. Program. Lang. Syst. 3*, 4 (Oct. 1981), 431–483.
3. APT, K. R. Ten years of Hoare's logic: A survey, Part II, nondeterminism. *Theoret. Comput. Sci. 28* (1984), 83–109.
4. APT, K. R., AND MAREK, W. Second order arithmetic and related topics. *Ann. Math. Logic 6* (1974), 177–209.
5. APT, K. R., AND OLDEROG, E.-R. Proof rules and transformations dealing with fairness. *Sci. Comput. Prog. 3* (1983), 65–100.
6. APT, K. R., AND PLOTKIN, G. D. A Cook's tour of countable nondeterminism. In *Proceedings ICALP '81*, S. Even and O. Kariv, Eds. Lecture Notes in Computer Science, vol. 115. Springer-Verlag, New York, 1981, pp. 479–494.
7. BACK, R. J. A continuous semantics for unbounded nondeterminism. *Theoret. Comput. Sci. 23*, 2 (1983), 187–210.
8. BACK, R. J. Semantics of unbounded non-determinism. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming*. Lecture Notes in Computer Science, vol. 85. Springer-Verlag, New York, 1980, pp. 51–63.
9. BACK, R. J. Proving total correctness of non-deterministic programs in infinitary logic. *Acta Inf. 15* (1981), 233–250.
10. BERRY, G., CURIEN, P. L., AND LEVY, J. J. Full abstraction for sequential languages: The state of the art. In *Proceedings of the French Seminar on the Applications of Algebra to Language Definition*

and Compilation (Fountainbleau, 1982), M. Nivat and J. Reynolds, Eds. Cambridge University Press, Cambridge, Mass., 1985.
11. BOOM, H. J. A weaker precondition for loops. *ACM Trans. Program. Lang. Syst. 4*, 4 (Oct. 1982), 668–677.
12. BROY, M., GNATZ, R., AND WIRSING, M. Semantics of non-deterministic and non-continuous constructs. In *Program Construction, International Summer School Marktoberdorf* (July 1978), F. L. Bauer and M. Broy, Eds. Lecture Notes in Computer Science, vol. 69. Springer-Verlag, New York, 1979, pp. 553–591.
13. CHANDRA, A. Computable non-deterministic functions. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science.* IEEE, New York, 1978, 127–131.
14. DE BAKKER, J. W. *Mathematical Theory of Program Correctness.* Prentice-Hall, Englewood Cliffs, N. J., 1980.
15. DE BAKKER, J. W., AND ZUCKER, J. I. Denotational semantics of concurrency. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing.* ACM, New York, 1982, pp. 153–158.
16. DIJKSTRA, E. W. *A Discipline of Programming.* Prentice-Hall, Englewood Cliffs, N. J., 1976.
17. EMERSON, E. A., AND CLARKE, E. M. Characterizing correctness properties of parallel programs using fixpoints. In *Proceedings of the 7th Colloquium on Automata, Languages, and Programming.* Lecture Notes in Computer Science, vol. 85, Springer-Verlag, New York, 1980, pp. 169–181.
18. FLOYD, R. W. Assigning meanings to programs. In *Proceedings of AMS Symposium in Applied Mathematics 19* (1967), 19–31.
19. GUREVICH, Y. Toward a logic tailored for computational complexity. In *Proceedings of 1983 Logic Colloquium* in Aachen, Lecture Notes in Mathematics, vol 104. Springer-Verlag, New York, 1984.
20. HAREL, D. First-order dynamic logic. In Lecture Notes in Computer Science, vol. 68. Springer-Verlag, Berlin, 1979.
21. HAREL, D., AND KOZEN, D. A programming language for the inductive sets and applications. *Inf. Cont. 63* (1984), 118–139.
22. HENNESSY, M. C. H., AND PLOTKIN, G. D. Full abstraction for a simple parallel programming language. In *Mathematical Foundations of Computer Science*, J. Becvar, Ed. Lecture Notes in Computer Science, vol. 74. Springer-Verlag, New York, 1979, pp. 108–120.
23. HITCHCOCK, P., AND PARK, D. Induction rules and termination proofs. In *Automata, Languages, and Programming*, M. Nivat, Ed. North Holland, Amsterdam, 1973.
24. MANNA, Z., AND PNUELI, A. Axiomatic approach to total correctness of programs. *Acta Inf. 3* (1974), 253–262.
25. MILNE, G., AND MILNER, R. Concurrent processes and their syntax. *J. ACM 26*, 2 (July 1979), 302–321.
26. MOSCHOVAKIS, Y. N. Elementary induction on abstract structures. North-Holland, Amsterdam, 1974.
27. NIVAT, M. Infinite words, infinite trees, infinite computations. In *Foundations of Computer Science*, J. W. de Bakker and J. van Leeuwen, Eds., vol. III, no. 2. Mathematical Centre Tracts, vol. 109, 1979, pp. 3–52.
28. PARK, D. On the semantics of fair parallelism. In *Proceedings of the Winter School on Formal Software Specification.* Lecture Notes in Computer Science, vol. 86. Springer-Verlag, New York, 1980, pp. 504–526.
29. PARK, D. A predicate transformer for weak fair iteration. In *Proceedings of the 6th IBM Symposium on Mathematical Foundations of Computer Science* (Hakone). IBM, New York, 1981.
30. PLOTKIN, G. D. A powerdomain construction. *SIAM J. Comput. 5*, 3 (1976), 452–487.
31. PLOTKIN, G. D. Dijkstra's predicate transformers and Smyth's powerdomains. In *Proceedings of the Winter School on Formal Software Specification.* Lecture Notes in Computer Science, vol. 86. Springer-Verlag, New York, 1980, pp. 527–553.
32. ROGERS, H., JR. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, New York, 1967.
33. SMYTH, M. Powerdomains. *J. Comput. Syst. Sci. 16*, 1 (1978), 23–36.
34. SPECTOR. C. Inductively defined sets of natural numbers. In: *Infinitistic Methods.* Pergamon Press, Elmsford, N.Y., 1961, pp. 97–105.
35. STOY, J. Semantic Models. In *Theoretical Foundations of Programming Methodology,* M. Broy and G. Schmidt, Eds. Reidel, Hingham, Mass., 1982, pp. 293–324.

RECEIVED JANUARY 1982; REVISED MARCH 1985; ACCEPTED JANUARY 1986