Semantics (with Proof Details) of Toy Command Languages for Iteration and Recursion

or

· **This Is (**surely more than just **Almost) Right** (cf. [**Nip2**])

Peter Hoffman
Pure Mathematics, University of Waterloo

Abstact. *This paper is a purely expository account for mathematicians of the topics in the title. We do give five different ways of 'semanticizing' iteration, and prove in detail their mutual equivalence. And we treat recursion four times, with increasing generality, proving the equivalence of the fixpoint and the syntactic unfolding approaches.*

What follows is a very leisurely exposition, giving definitions of the semantics for what seem to be the two most fundamental non-trivial constructs in programming languages of the imperative kind (as opposed to functional programming, logic programming, etc ...). In the case of *recursion*, we spiral (upwards, I hope!) through four stages, single procedure variable up to unrestricted mutual recursion. Only the last seven pages of those four sections are needed for a complete and rigorous treatment relating 'fixpoint' to 'syntactic unfolding operational' semantics. In the case of *iteration*, more is done on definitions than in [**W**]. And we give complete proofs, filling in many details left out in that book, with the partial motivation of underlining the fact that the use of theorem-proving software for doing this, [**Nip2**], though fascinating, is hardly necessary. We have made a joking second title above along the lines of [**Nip2**]'s which praises with faint damns (well, only one). Details of a 'CS-pedagogical' nature in [**W**], such as explicit treatment of mathematically trivial matters related to numerals, calculations of $1^{\underline{\text{st}}}$order terms ('arithmetic expressions'), and truth evaluations of quantifier-free $1^{\underline{\text{st}}}$order formulas ('Boolean expressions') are all simply ignored here as being beside the point for our aims, and unnecessarily tied to a particular $1^{\underline{\text{st}}}$order language of arithmetic.

It would be surprising if there were anything really new in this paper, other than some choices of notation and perhaps having all this classical stuff actually written out in a form free of CS jargon. It's not free of mathematical logic jargon but is (hopefully) immediately readable by students of mathematics. It is assumed that the reader is familiar with the basic setup for $1^{\underline{\text{st}}}$order logic. See [**book**] for the notation used here in that respect, if need be.

I have little doubt that Dana Scott would have at least sketched for himself on scrap paper some form of the stuff here on recursive programming

(and more), when beginning to rigorize Strachey's pioneering work back in the '60's, leading to their denotational semantics. But texts for CS students seem to shy away from much rigorous detail on recursive imperative programming, even just usual semantics, much less Floyd-Hoare logic. (To deviate from the topic:—as far as I can see, the latter still has a few open natural general questions about total correctness re recursion—and this for parameter-free procedures.) So, for readers, this paper's motivation is entirely pedagogical.

But I have a second motivation. In dealing with the most general form of parameter-free recursion (whether mutual or merely recursion with a single procedure variable), I was and am puzzled about the possibility of getting a straightforward 'unfolding' definition of the operational semantics in the case where 'free' calls to 'other than X'-procedure variables are allowed within the body of a declared procedure for procedure variable X, sometimes called *non-simple recursion* (and similarly in the mutual case). So I felt the need to write out the 'fixpoint-style' definition and the proofs of what is required to make the definition work, and also to at least prove the identity relating it to the semantics of the 'syntactic unfoldings' of the body (even if I still don't know whether that identity could be used as a definition of the semantics in the most general case). But see [**depth**], a follow-up to this paper, where recursion depth is formalized and used to give another way of defining semantics, and also of giving a rather different form of syntactic unfolding which *can* serve as a definition if desired. Also, in the tour-de-force [**Old**], the Floyd-Hoare logic of very much 'non-toy' Algol-style languages is studied. The hornet's nest of interaction between variable declarations and procedures *with parameters* is the central point of interest, in connection with Clarke's famous theorem on the non-existence of Cook-complete such systems. Producing proofs of the counterexamples to dropping his requirements for that theorem to hold is the major accomplishment, taking the form of a general theorem on the soundness and completeness of such a Floyd-Hoare system. Olderog's "copy-rule" semantics and other details are excruciatingly intricate, probably unavoidably. But those semantics are a version of unfolding semantics, so likely could be simplified for the austere toy language of recursion here.

I would of course be grateful to any reader who can point me to places in the literature useful to change the above-confessed ignorance to knowledge.

What follows is divided into five technical sections, curiously numbered 0 , $1-$, $1+$, $\infty-$ and $\infty+$ (preceded by a pair of preludes). To partially explain this bizarre numerology:

Section 0 presents and relates five somewhat distinct versions of the semantics for a toy language of iteration (a language which is more austere than the usual "while-language", because it ignores 'if-then-else', which is in any case simulatable using only 'while-do' and sequencing when the underlying $1^{\underline{st}}$order language is non-trivial). The second (jocular) title of this paper is a take-off on [**Nip2**], where (as in our Section 0) only the while-language is dealt with. I do realize that the "almost" in [**Nip2**]'s title refers to an error in a proof further along, re "axiomatic semantics", that is, Floyd-Hoare logic. One does, however, find it surprising that, even as late as 1993, basic undergraduate textbooks emanating from major CS centers were still running into difficulties with getting all proof details checked out for the (almost minimal) while-language. At least this paper may serve to disabuse those who misunderstand the motivation for [**Nip2**] as possibly coming from any actual difficulty for humans to write out all the details on these topics without the help of theorem-proving software. Nipkow does indicate that [**Nip2**]'s motivation is elsewhere. But some later papers from the ISABEL/HOL school seem to leave the impression that such software is indispensable on these topics. It certainly isn't if iteration and/or recursion are the only 'hard' programming features involved.

As to the pairs (1 or ∞ , $\mp$) labelling the other four sections, these sections are all parallel to each other, the first three being redundant, as it is easy to specialize the final section, $\infty+$, to each of the others. The 1 or ∞ tell us whether we are dealing respectively with recursion on a single procedure variable or with mutual recursion. The $-$ and $+$ tell us whether respectively we do not or do allow free occurrences of $callX$ within the body declared in a recursion where X is *not* one of the procedure variables 'recursed on'. What with the close parallels and automated mathematical typesetting, it wasn't all that more onerous to type up all four sections, rather than just $\infty+$. And the generalizations from $1-$ to $1+$ and to $\infty-$ are logically independent of each other, but rather technical, so separating them out hopefully has some pedagogical value to more than just the author. Some students may be happy just to check out $1-$. The toy language there is still Turing equivalent—i.e. is able to compute any recursive function.

Note that the 'if-then-else'-command constructor does get dealt with in the sections on recursion. So the reader who wants details for the full while-language can pretty much just import those bits back into Section 0 and write out the details for herself. And then she can try the following exercise.

Exercise. Show that the 'if-then-else'-command can be simulated 'up to one spurious variable' in $\mathcal{A}ten$ in that, for the semantic function m defined in various places below, we have that (omitting a few brackets)

$m(ite(H)(C)(D))$ and

$$m(x \dot{\leftarrow} 0 \ ; \ whdo(H \wedge x \approx 0)(C; x \dot{\leftarrow} 1) \ ; \ whdo(\neg H \wedge x \approx 0)(D; x \dot{\leftarrow} 1)) \ ,$$

(where x is some variable not occurring in any of C, D or H) would be 'equal' if we regard two states as 'the same' as long as they agree away from x. This assumes the $1^{\underline{\text{st}}}$order language has a couple of fixed terms (0 and 1 here) which take different values in the interpretation.

Prelude A : Fixed Points.

Firstly note that the symbol $\subset$ here simply means inclusion of sets, **not necessarily strict.** (We have $X \subset X$ for any set X.)

Set Inclusions.

Definitions. Let P be any set. The set of all its subsets will be denoted 2^P. A function $F : 2^P \to 2^P$ is *monotone*, or *monotonic*, if and only if

$$\alpha \subset \beta \implies F(\alpha) \subset F(\beta) \ .$$

A *least fixed point* of an $F : 2^P \to 2^P$ is a (clearly unique) $\lambda \in 2^P$ such that
(i) $F(\lambda) = \lambda$ and (ii) for all μ, if $F(\mu) = \mu$ then $\lambda \subset \mu$.
Such a λ , if it exists, will be denoted $lix(F)$.

Theorem P1. *Any monotone map $F : 2^P \to 2^P$ has a least fixed point. It is in fact given as*

$$\lambda \ := \ \bigcap\{ \ \alpha \mid \alpha \in \mathcal{A}_F \ \} \ := \ \bigcap \mathcal{A}_F \ ,$$

where

$$\mathcal{A}_F \ := \ \{ \ \alpha \in 2^P \mid F(\alpha) \subset \alpha \ \} \ .$$

Proof. (Note that $\mathcal{A}_F \neq \emptyset$ since $P \in \mathcal{A}_F$.) If $F(\mu) = \mu$, then $\mu \in \mathcal{A}_F$, and so $\lambda \subset \mu$, since λ is the intersection of sets, one of which is μ. Thus, as long as it *is* a fixed point, λ will certainly be the least one.

We have $\lambda \subset \alpha$ for all $\alpha \in \mathcal{A}_F$, so $F(\lambda) \subset F(\alpha) \subset \alpha$ for all such α, the first 'inequality' by monotony of F. And so $\underline{F(\lambda) \subset \lambda}$ by the latter's definition as an intersection. But now by monotony again we have $F(F(\lambda)) \subset F(\lambda)$, so $F(\lambda) \in \mathcal{A}_F$, giving $\underline{\lambda \subset F(\lambda)}$, and we're done.

Definition. A function $F : 2^P \to 2^P$ is *cts* if and only if, for any chain $\alpha_0 \subset \alpha_1 \subset \alpha_2 \subset \cdots$, we have $F(\bigcup_i \alpha_i) \ = \ \bigcup_i F(\alpha_i)$.

Remarks. It is easy to see that a cts F is necessarily monotonic, and so has a least fixed point. Actually, a more direct proof comes from the fact that, for cts F, the fixed point is given by (as proved more generally further down)

$$lix(F) \ = \ \bigcup_{i \geq 0} F^i(\emptyset) \qquad (***)$$

Theorem P2. *Let $\{F_N : 2^P \to 2^P\}_{N\geq 1}$ be a sequence of maps such that $F_N(\alpha) \subset F_{N+1}(\alpha)$ for all N and α. We have the following.*

(i) *If each F_N is monotonic, then*

$$\bigcup_N F_N \quad \textit{is monotonic, where} \quad (\bigcup_N F_N)(\alpha) \;:=\; \bigcup_N F_N(\alpha)\ ,$$

and

$$lix(\bigcup_N F_N) \;\supset\; \bigcup_N lix(F_N)\ .$$

(ii) *There are examples showing that we cannot strengthen* (i) *by asserting equality.*

(iii) *However, if each F_N is cts, then*

$$\bigcup_N F_N \quad \textit{is cts} \qquad \textit{and} \qquad lix(\bigcup_N F_N) \;=\; \bigcup_N lix(F_N)\ .$$

Proof. To begin, assume only monotonicity for the F_N. Let

$$g \;:=\; \bigcup_N F_N \quad ; \quad \beta_N \;:=\; lix(F_N) \quad ; \quad \beta \;:=\bigcup_N \beta_N\ .$$

Firstly, if $\gamma \subset \delta$, then $F_N(\gamma) \subset F_N(\delta)$ for all N. And thus

$$g(\gamma) \;=\; \bigcup_N F_N(\gamma) \;\subset\; \bigcup_N F_N(\delta) \;=\; g(\delta)\ ,$$

so g is indeed monotonic.

Now suppose that γ satisfies $g(\gamma) \subset \gamma$. Then $F_N(\gamma) \subset \gamma$ for all N. So $\beta_N \subset \gamma$ for all N, since γ will be one of the sets intersected to produce β_N via the formula in **P1**. Thus $\beta \subset \gamma$. Since $lix(g)$ is a choice for γ, we therefore have $\beta \subset lix(g)$, proving (i).

Let us continue assuming only monotonicity for the F_N to isolate what is needed for the counterexample in (ii), and to get (iii) quickly with a sudden upgrade to 'ctsness'. It suffices to prove $g(\beta) \subset \beta$ for (iii), and that reduces to $F_N(\beta) \subset \beta$ for all N.

If $K \geq N$, then $F_N(\beta_K) \subset F_K(\beta_K) = \beta_K \subset \beta$.

If $K \leq N$, then $F_N(\beta_K) \subset F_N(\beta_N) = \beta_N \subset \beta$.

The first inclusion of that last line follows from $\beta_K \subset \beta_N$, which itself follows from part of the previous paragraph with N and K reversed, namely $F_K(\beta_N) \subset \beta_N$.

So the last two paragraphs tell us that $F_N(\beta_K) \subset \beta$ for all N and K.

But we need $F_N(\bigcup_K \beta_K) \subset \beta$. That is now immediate, once we assume that F_N is cts, so (iii) is done, since

$$F_N(\bigcup_K \beta_K) \;=\; \bigcup_K F_N(\beta_K) \;\subset\; \bigcup_K \beta \;=\; \beta \;.$$

An example for (ii) is concoctable as follows—take P to be the natural numbers, and define F_N so that it maps all infinite subsets to P itself, but maps any finite set α to $\alpha \cup \{2, 4, \cdots, 2N\}$. It is not at all onerous to check: that F_N is monotonic; that $F_N(\alpha) \subset F_{N+1}(\alpha)$; that $\beta_N = \{2, 4, \cdots, 2N\}$; that β is the set of all even natural numbers; but that g maps all infinite sets to P, and any finite set α to the union of α with all even natural numbers. Thus g has only P as a fixed point, and so $lix(g) = P \neq \beta$.

The Product of Set Inclusions.

Looking at what was just done, the reader should observe that one never sees "$\in \alpha$", only "$\subset \alpha$". The following corresponding results have proofs which are word-for-word the same, except for replacing $\subset, \cap$ and $\cup$ by $\sqsubset, \sqcap$ and $\sqcup$. (And having read those following paragraphs, one of course realizes that the whole thing must have a generalization in the form of an abstraction which applies to fairly general partial orders.)

Here we change 2^P to $X \;:=\; 2^{P_1} \times 2^{P_2} \times \cdots \times 2^{P_k}$ for any sets P_i. (As it happens, the application in Sections $\infty*$ will have all the P_i the same as each other.)

Definitions. Define a binary relation $\sqsubset$ on X by

$$(\alpha_1, \cdots, \alpha_k) \sqsubset (\beta_1, \cdots, \beta_k) \qquad \Longleftrightarrow \qquad \alpha_i \subset \beta_i \;\text{ for all } i \;.$$

A function $F : X \to X$ is *monotone* if and only if

$$\alpha \sqsubset \beta \implies F(\alpha) \sqsubset F(\beta) \;.$$

A *least fixed point* of an $F : X \to X$ is a $\lambda \in X$ (clearly unique because $\lambda \sqsubset \mu$ and $\mu \sqsubset \lambda$ implies $\lambda = \mu$) such that

(i) $F(\lambda) = \lambda$ and (ii) for all μ, if $F(\mu) = \mu$ then $\lambda \sqsubset \mu$.

Such a λ , if it exists, will be denoted $lix(F)$.

Theorem P1^k. *Any monotone map $F : X \to X$ has a least fixed point. It is in fact given as*

$$\sqcap\{\ \alpha \mid \alpha \in \mathcal{A}_F\ \} := \sqcap\mathcal{A}_F\ ,$$

where

$$\mathcal{A}_F\ :=\ \{\ \alpha \in X \mid F(\alpha) \sqsubset \alpha\ \} \quad \text{and, with} \quad \pi_i(\beta_1, \cdots, \beta_k)\ :=\ \beta_i\ ,$$

$$\sqcap\mathcal{A}\ :=\ (\bigcap_{\alpha\in\mathcal{A}} \pi_1\alpha, \cdots, \bigcap_{\alpha\in\mathcal{A}} \pi_k\alpha)\ .$$

Definition. A function $F : X \to X$ is *cts* if and only if, for any chain $\alpha_0 \sqsubset \alpha_1 \sqsubset \alpha_2 \sqsubset \cdots$, we have $F(\bigsqcup_i \alpha_i)\ =\ \bigsqcup_i F(\alpha_i)$.
Here, $\bigsqcup_i(\alpha_{1i}, \alpha_{2i}, \cdots, \alpha_{ki})$ means $(\bigcup_i \alpha_{1i}, \bigcup_i \alpha_{2i}, \cdots, \bigcup_i \alpha_{ki})$.

Remarks. It is easy to see that a cts F is necessarily monotonic, and so has a least fixed point. Actually, a more direct proof comes from the fact that the fixed point is given by

$$lix(F)\ =\ \bigsqcup_{i\geq 0} F^i(\vec{\phi})\ ,$$

where $\vec{\phi}$ is $(\emptyset, \emptyset, \cdots, \emptyset)$.

This follows from the definition of lix with one application of 'ctsness': Calling the right-hand side just above λ, we have

$$F(\lambda)\ =\ F(\bigsqcup_i F^i(\vec{\phi}))\ =\ \bigsqcup_i F(F^i(\vec{\phi}))\ =\ \lambda\ ,$$

yielding the first condition in the definition. The second one is done by an easy induction on i showing that $F^i(\vec{\phi}) \sqsubset \alpha$ for any α satisfying $F(\alpha) = \alpha$. And so $\lambda = \bigsqcup_{i\geq 0} F^i(\vec{\phi})\ \sqsubset \bigsqcup_{i\geq 0} \alpha\ = \alpha$ for such an α, as required.

Theorem P2^k. *Let $\{F_N : X \to X\}$ be a sequence of maps such that $F_N(\alpha) \sqsubset F_{N+1}(\alpha)$ for all $N \geq 1$ and all α. We have the following.*

(i) *If each F_N is monotonic, then*

$$\bigsqcup_N F_N \quad \textit{is monotonic, where} \quad (\bigsqcup_N F_N)(\alpha)\ :=\ \bigsqcup_N F_N(\alpha)\ ,$$

and

$$lix(\bigsqcup_N F_N)\ \sqsupset\ \bigsqcup_N lix(F_N)\ .$$

(ii) *There are examples showing that we cannot strengthen* (i) *by asserting equality.*

(iii) *However, if each F_N is cts, then*

$$\bigsqcup_N F_N \quad \textit{is cts} \qquad \textit{and} \qquad lix(\bigsqcup_N F_N) \;=\; \bigsqcup_N lix(F_N)\ .$$

Relations.

When P has the form $Q \times Q$, we can define *composition* of relations, i.e. of elements of $2^{Q\times Q}$, as follows.

Definition. Define $\alpha \circ \beta$ by

$$(q, q'') \in \alpha \circ \beta \quad \Longleftrightarrow \quad \exists\ q' \ \text{ with } \ (q, q') \in \beta \ \text{ and } \ (q', q'') \in \alpha\ .$$

Note that a relation can be thought of as a function from Q to the set of subsets of Q—(mapping an element of Q to the set of elements to which it is related). This is the usual way to identify $2^{Q\times Q}$ with $(2^Q)^Q$. But the composition just defined is not related to composition of such functions, which is undefined. Rather it relates to generalizing from the case where both of the relations to be composed happen to be graphs of functions (see the definition below), the answer being the graph of the usual composition of those functions.

Lemma P3. *If $\alpha \subset \alpha'$ and $\beta \subset \beta'$, then $\alpha \circ \beta \subset \alpha' \circ \beta'$.*

Lemma P4. *If $\{\alpha_N\}$ and $\{\beta_N\}$ are sequences of relations such that $\alpha_1 \subset \alpha_2 \subset \cdots$ and $\beta_1 \subset \beta_2 \subset \cdots$, then*

$$(\bigcup_{N\geq 1} \alpha_N) \circ (\bigcup_{N\geq 1} \beta_N) \;=\; \bigcup_{N\geq 1} (\alpha_N \circ \beta_N)\ .$$

Definition. A relation α is the *graph of a function* if and only if

$$\forall\ q,\ q',\ q''\ :\ \ \text{if } (q, q') \in \alpha \text{ and } \ (q, q'') \in \alpha\ ,\ \text{then } q' = q''.$$

Lemma P5. *If relations α and β are both graphs of functions, then so is $\alpha \circ \beta$.*

The proofs are very brief manipulations with the definitions.

Prelude B : Common Notation re commands etc.

Assume we have a particular $1^{\underline{\text{st}}}$order language with equality (using $\approx$ for it, to keep syntax and semantics at arms-length), and with its special strings called *terms* and *formulas*. This language will be fixed but arbitrary, all five command languages, $\mathcal{A}ten$ and $\mathcal{R}ec_{**}$, of the following five sections being based on it. They are also based on a fixed interpretation, $\mathcal{N}$, of the $1^{\underline{\text{st}}}$order language. The best pair of examples is the language of $1^{\underline{\text{st}}}$order number theory, with $\mathcal{N}$ taken to be the set of natural numbers with its usual addition, multiplication and ordering, and its elements 0 and 1. Having come to regard "$\models$" as carrying too many slightly different meanings to be much use pedagogically any more, we abbreviate "H is true at s" as $H\mathsf{tt}@s$, and similarly use ff for "false".

Let $\mathcal{V}ar$ be the set of variables in the $1^{\underline{\text{st}}}$order language. Define the set of states to consist of all the functions from $\mathcal{V}ar$ to the interpretation $\mathcal{N}$, i.e.

$$\mathcal{S}te \ := \ \mathcal{N}^{\mathcal{V}ar} \ .$$

If $s \in \mathcal{S}te$, $x \in \mathcal{V}ar$ and $n \in \mathcal{N}$, define $s_{x \mapsto n}$ to be the state which agrees with s except that it maps x to n, i.e.

$$s_{x \mapsto n}(y) \ := \ \begin{cases} n & \text{if } y = x \ ; \\ s(y) & \text{if } y \neq x \ . \end{cases}$$

In each of the following five sections, we shall define by structural induction a set of strings of symbols called *commands*. In Section 0, the set is the language $\mathcal{A}ten$ of 'iteration' (almost the same as a language called while, much studied in basic theoretical CS). In Sections $\#\sigma$, for $\# = 1$ or ∞, and $\sigma = \mp$, they are the languages $\mathcal{R}ec_{\#\sigma}$ of 'recursion'. In all five cases, the set of *basic semantic values* is

$$\mathcal{B}sm \ := \ 2^{\mathcal{S}te \times \mathcal{S}te} \ .$$

As mentioned in the previous section, that set can be identified with $(2^{\mathcal{S}te})^{\mathcal{S}te}$, which is the set of all functions from $\mathcal{S}te$ to the collection of all subsets of $\mathcal{S}te$. In each case, the semantics of the language $\mathcal{L}$ is, at its most basic level, a function

$$m \ : \ \mathcal{L} \ \longrightarrow \ \mathcal{B}sm \ .$$

Figuring out such definitions and their properties is the basic objective of each of those sections.

Now each of these $\mathcal{L}$'s together with its semantics m will turn out to be (quite *un*surprisingly) a *deterministic* command language. The meaning of that adjective is that (in its alter ego as a function from states to sets of states) for each command C the 'function' $m(C)$ applied to a given state produces either the empty set or a singleton. The latter two possibilities correspond respectively to the intuitive idea of a program C taking an input state and either producing no output ('doing an infinite loop') or producing a unique output state. Going back to the actual $\mathcal{B}sm$ as the set of all subsets of $\mathcal{S}te \times \mathcal{S}te$, being deterministic corresponds to no s occurring as the 1st coordinate more than once in any $m(C)$.

Though most of us intuit a computer as acting like this, it turns out to be simpler to de-emphasize this 'deterministicism' and deal with m as defined more generally to possibly give, for a given command and input state, a set of states which may even be infinite. Non-determinism is very important in lots of other theoretical considerations, and it is desirable to formulate everything as much as possible without assuming the command language is deterministic.

In the sections on recursion, there will be an atomic command $callX$, one for each X in a set $\mathcal{P}cv$ of *procedure variables* (distinct from the previous $1^{\underline{\text{st}}}$order variables). The basic semantics of $callX$ will always be that $m(callX)$ is the empty set. However, the real use of $callX$ is to occur within the command telling the machine to "do C recursively, treating calls to X as saying to go back and start doing C again (with input as the new state which the machine has arrived at)". As to syntax, this command is the string ∇XC in both of the sections $1-$ and $\infty-$. (Not to be patronizing, but note that the "∇" and the "X" are single symbols, whereas the "C" is in general a string of symbols. See [**sing**], particularly its Section 2, for lots of examples.) Much of the work here centers on defining precisely and explaining the semantics of these '∇-commands'. It follows from one's intuitive idea of what the command is 'all about' that one should get the same effect by taking a copy of C instead, but then modifying it by substituting that ∇XC-command for each $callX$ occurring as a subcommand of C—strictly speaking, not *all* occurrences, only those which are *free*—not governed by another ∇.

This leads to the idea of a *fixed point*. Roughly speaking, one should

go back to C itself, but regard the above occurrences of $callX$ as a kind of free variable which is to take 'every possible semantic value'. And then see what the semantics of the whole of such a 'modified C' turns out to be, and, by the observation above, seek a choice of the 'variable semantics of $callX$' which produces the **same** semantics for the 'modified C'. So *that* semantic value is a fixed point of a suitable function, a point 'mapped to itself' by the function.

Stated in more technical language, fixing X, first one defines a command C_D in a precise manner, capturing the result of replacing each free occurrence of $callX$ within the command C by the command D. Then the previous paragraph can be summarized by an 'equation' $C_{\nabla XC} \sim \nabla XC$, where "$\sim$" roughly means "has the same semantics as". More precisely : $m(C_{\nabla XC}) = m(\nabla XC)$. In any case, one is seeking a D, or at least its semantics, such that $C_D \sim D$, so D is a "fixed point up to semantics" for the function best written as $[D \mapsto C_D]$. (Overzealous λ-philes would write $\lambda D.C_D$, thereby cutting down some audiences by about 95%) .

The idea is moderately subtle, but the details of carrying it out are really quite simple, and even can be made to seem so with a thoughtful choice of notation (not invariably done in the past). In any case, the above leads us to define a more complicated semantic function, with $\mathcal{L}$ as any of the four $\mathcal{R}ec_{\#\sigma}$,

$$\mathcal{M} \ : \ \mathcal{L} \times \mathcal{B}sm^{\mathcal{P}cv} \ \longrightarrow \ \mathcal{B}sm \ .$$

And then $m(C)$ will be defined to be $\mathcal{M}(C, \Phi)$ where Φ maps all procedure variables to the empty set. To relate this to the above: one needs to consider any possible value for the putative semantics of $callX$ for all the various X in $\mathcal{P}cv$, and this is just a function from $\mathcal{P}cv$ to $\mathcal{B}sm$, that is, an element of the '2nd coordinate', $\mathcal{B}sm^{\mathcal{P}cv}$, in the domain of the function $\mathcal{M}$ above. So the value of $\mathcal{M}$ on a pair (C, Θ) above gives the basic semantic value of what C would become if all the $callX$'s for all X were instead some mythical commands whose basic semantics were given by the 2nd coordinate Θ. That's a rough intuitive explanation of the main point in those sections.

Another way of thinking about the display just above is the following. Many others will, in specifying their syntax, simply drop the "*call*", and have just plain procedure variables X occurring in commands. Then the display above is analogous to the situation in $1^{\underline{\text{st}}}$order logic, where the semantics function is thought of as a function of two 'variables', namely, the interpre-

tation used and the assignment of variables. In the display we also have a function of two 'variables', and the second one comes from a set, $\mathcal{B}sm^{\mathcal{P}cv}$, which is essentially an assignment of procedure variables.

The logical/mathematical/CSish athlete who wants only a streamlined version of the semantics of recursive programming, and has no interest in seeing stuff about iteration, should turn to the final seven pages before the appendix (the 2nd part of the last section), and skip everything else.

But here we should also explain briefly the notation occurring for the other commands. Those are very standard commands in all Algol-style command languages such as C, C++, and Pascal, so everybody understands what they are supposed to mean, and we'll give no further motivation for the semantic definitions when we come to them later. Our notation is however quite non-standard :

Assignment: $x \leftarrow\!\!\div t$—the machine is expected to get out its handy pocket calculator, and work out the value of the term t, using the input state for the values of any variables occurring in the term. Then the output state agrees with the input, except that the 'value' of x gets changed to the value just calculated.

Sequencing: $(C; D)$—do C, and, if it converges, i.e. doesn't 'infinitely loop', then do D with input, of course, the state which 'C outputted'.

if-then-else: $ite(H)(C)(D)$—calculate the truth value of the 'condition' H, and then do either C or D, depending on whether that value is true, tt, or false, ff.

while-do: $whdo(H)(C)$—check H, if false, quit; but, if true, do C, then, as long as that effort doesn't 'loop forever', repeat the previous and continue to do so until and only if the value ff comes up for H.

The last four are the usual while-language. Our $\mathcal{A}ten$ omits ite. Each of the four $\mathcal{R}ec$ languages omits $whdo$, and includes some form of the previous ∇-commands, as well as the $call$-command.

The semantics we spend all the time discussing is just various mathematically precise ways of saying what has been discussed intuitively in the previous eight paragraphs.

Those who have read other material on this type of thing might be wondering both why we don't go into the general theory of CPOs, Scott domains etc., and why only the set inclusion ordering, plus cartesian products of it,

are needed. I don't know the answer to the second question yet, and that for the first one is: "pedagogical reasons". Perhaps a few people who read this will get curious about what might be the best general abstract version, and then maybe even get excited about category theory and other generalities which seem essential in lots of theoretical CS. But for pure recursion and iteration, they are far from being essential.

Section 0 : Iteration Semantics.

In this section we present five ways to define the semantics of a particular language, christened $\mathcal{A}ten$, which is just about the simplest possible Turing-equivalent imperative language. Here it is, one atomic command and two command constructors—assignment, sequencing and 'while-do' :

$$x \leftarrow t \quad || \quad (C; D) \quad | \quad whdo(H)(C)$$

where C and D are names for commands (the strings which are the elements of that language); x is any variable in $1^{\underline{\text{st}}}$order; t is any $1^{\underline{\text{st}}}$order term; H is any *quantifier-free* $1^{\underline{\text{st}}}$order formula.

The intuitive meanings of these three commands were given just a few paragraphs back.

Below we give those meanings in several formal, mathematical ways. That's what semantics is all about. The main job will be to prove that these different ways of doing it are equivalent to each other.

There is one form of the semantics on this page, and one on each of the next four pages, along with a few theorems, in particular saying that all these five forms of the semantics agree with each other. The proofs of the theorems are left for an appendix at the end of the paper.

Giant Step Operational Semantics.

Define an 'm-function', as discussed in the previous section,

$$m_{_{GSO}} \ : \ \mathcal{A}ten \longrightarrow \mathcal{B}sm \ ,$$

by the following explicit, once-and-for-all formulae (where $s(t)$ is the value of the state s, calculated on the term t in the usual way):

$$\begin{aligned}
&m_{_{GSO}}(x \leftarrow t) \ := \ \{(s, s_{x \mapsto s(t)}) \mid s \in \mathcal{S}te\} \ . \\
&m_{_{GSO}}((C; D)) \ := \ m_{_{GSO}}(D) \circ m_{_{GSO}}(C) \ . \\
&m_{_{GSO}}(whdo(H)(C)) \ := \ \{(s, s') \mid \exists N \geq 0 \ , \\
&\qquad\qquad \exists s = s_0 \ , \ s_1 \ , \ \cdots \ , \ s_N = s' \ \text{ such that } H\mathbf{ff}@s_N \\
&\qquad\qquad\qquad \text{but for } 0 \leq i < N, \ H\mathbf{tt}@s_i \text{ and } (s_i, s_{i+1}) \in m_{_{GSO}}(C)\} \ .
\end{aligned}$$

(Why CSers never want to write this last part of the definition in that form is one of those things which completely escapes me; perhaps it's some abhorrence of the "$\exists N$".)

Note that the last clause in the definition gives a set which, for s with $H\mathbf{ff}@s$, includes (s, s) and no other (s, s'). That's exactly the case $N = 0$.

Denotational Fixpoint Semantics.

Define a second 'm-function' (not really different, as we see in a theorem below),

$$m_{_{DFX}} \ : \ \mathcal{A}ten \longrightarrow \mathcal{B}sm \ ,$$

in exactly the same way as just above for assignment and sequencing commands. But for 'while-do', define

$$m_{_{DFX}}(whdo(H)(C)) \ := \ lix[\ f_{H,C} : \mathcal{B}sm \to \mathcal{B}sm\ ] \ ,$$

where

$$f_{H,C}(\alpha) \ := \ \{\ (s,s) \mid H\mathbf{ff}@s\ \} \ \cup \ \{\ (s,s') \in \alpha \circ m_{_{DFX}}(C) \mid H\mathbf{tt}@s\ \} \ .$$

Note that this uses (inductively) that $m_{_{DFX}}(C)$ is 'already defined'.

Proposition DFX1. *The function $f_{H,C}$ is monotone* and so $m_{_{DFX}}$ is well-defined.

Exercise. Give another proof by showing that $f_{H,C}$ is actually cts.

The language $\mathcal{A}ten$ is a bit pathetic from the software point-of-view, though not the computability point-of-view. We won't need ctsness since we'll soon pass along to languages for recursion, rather than dwelling on iteration. The reader might like to look up the Ackermann function, and write programs for it with this iteration language (give yourself the luxury of *ite* as well) and then with recursion. The first program will likely be about two orders of magnitude longer than the second, illustrating the point about the software point-of-view.

Theorem DFX2. *The semantic functions already defined are actually the same function, i.e.*

$$m_{_{DFX}} \ = \ m_{_{GSO}} \ .$$

Big Step Operational Semantics.

Define $\mathcal{C}$, a subset of $\mathcal{S}te \times \mathcal{A}ten \times \mathcal{S}te$, to be the intersection of all those subsets $\mathcal{B} \subset \mathcal{S}te \times \mathcal{A}ten \times \mathcal{S}te$ for which (i) to (iv) below hold for all s, s', s'', x, t, H, C and D :

(i) $(s\ ,\ x \mathrel{\dot{\leftarrow}} t\ ,\ s_{x \mapsto s(t)}) \ \in\ \mathcal{B}$.

(ii) $H\mathtt{ff@}s \implies (s\ ,\ whdo(H)(C)\ ,\ s) \ \in\ \mathcal{B}$.

(iii) If $(s\ ,\ C\ ,\ s') \ \in\ \mathcal{B}$ and $(s'\ ,\ D\ ,\ s'') \ \in\ \mathcal{B}$, then $(s\ ,\ (C;D)\ ,\ s'') \ \in\ \mathcal{B}$.

(iv) That $(s\ ,\ whdo(H)(C)\ ,\ s'') \ \in\ \mathcal{B}$ follows from: [$H\mathtt{tt@}s$, and, for some state s', we have both $(s\ ,\ C\ ,\ s') \ \in\ \mathcal{B}$ and $(s'\ ,\ whdo(H)(C)\ ,\ s'') \ \in\ \mathcal{B}$].

We are intersecting a non-empty collection of sets since $\mathcal{S}te \times \mathcal{A}ten \times \mathcal{S}te$ itself qualifies as a $\mathcal{B}$; that is, it satisfies the four conditions.

Definition. A *BS-derivation of* $(s\ ,\ C\ ,\ s')$ is a finite sequence,

$$(s_1\ ,\ C_1\ ,\ s_1')\ ,\ (s_2\ ,\ C_2\ ,\ s_2')\ ,\ \cdots\cdots\ ,\ (s_n\ ,\ C_n\ ,\ s_n')\ ,$$

whose last term is $(s\ ,\ C\ ,\ s')$, and such that, for each i with $1 \leq i \leq n$, at least one of (I) to (IV) below hold (actually, ... *exactly* one..., by unique readability):

(I) $(s_i\ ,\ C_i\ ,\ s_i')$ is $(s\ ,\ x \mathrel{\dot{\leftarrow}} t\ ,\ s_{x \mapsto s(t)})$ for some s, t and x.

(II) $(s_i\ ,\ C_i\ ,\ s_i')$ has the form $(s\ ,\ whdo(H)(C)\ ,\ s)$ where H is $\mathtt{ff@}s$.

(III) $(s_i\ ,\ C_i\ ,\ s_i')$ is $(s\ ,\ (C;D)\ ,\ s'')$ for some s, s', s'', C and D such that both $(s\ ,\ C\ ,\ s')$ and $(s'\ ,\ D\ ,\ s'')$ occur as sequence terms before the ith.

(IV) $(s_i\ ,\ C_i\ ,\ s_i')$ has the form $(s\ ,\ whdo(H)(C)\ ,\ s'')$ where H is $\mathtt{tt@}s$, and such that, for some state s', we have both $(s\ ,\ C\ ,\ s')$ and $(s'\ ,\ whdo(H)(C)\ ,\ s'')$ occur as sequence terms before the ith.

(This is like a Hilbert-style proof system; in **[W]**, the corresponding one is Gentzen-style.)

Theorem BSO1. *The set $\mathcal{C}$ consists of exactly those $(s\ ,\ C\ ,\ s')$ for which there is at least one BS-derivation.*

Sketch Proof. This is very analogous to those for similar theorems in algebra, for example, asserting the equality of two descriptions ('inner vs. outer' or 'effective vs. pie-in-the-sky') for the group generated by a given set or for the ring generated by a given set. One shows that the set $\mathcal{D}$ of derivables is a $\mathcal{B}$ as above and so $\mathcal{C} \subset \mathcal{D}$. On the other hand, one can verify $\mathcal{D} \subset \mathcal{B}$ for any $\mathcal{B}$ as above, and so $\mathcal{D} \subset \mathcal{C}$. Thus $\mathcal{D} = \mathcal{C}$, as required.

Definition. Define $m_{_{BSO}} : \mathcal{A}ten \longrightarrow \mathcal{B}sm$ by $m_{_{BSO}}(C) := \{\ (s, s')\ |\ (s\ ,\ C\ ,\ s') \in \mathcal{C}\ \}$.

Pompous Remark. A fancy way of reformulating the definition goes as follows. The canonical bijection

$$2^{\mathcal{S}te \times \mathcal{A}ten \times \mathcal{S}te} \ \longrightarrow\ (2^{\mathcal{S}te \times \mathcal{S}te})^{\mathcal{A}ten} = \mathcal{B}sm^{\mathcal{A}ten}$$

maps $\mathcal{C}$ to $m_{_{BSO}}$.

Theorem BSO2. *All three semantic functions already defined are actually the same function, i.e.*

$$m_{_{BSO}} \ =\ m_{_{GSO}} \ =\ m_{_{DFX}}\ .$$

Tiny Step Operational Semantics.

Define yet another 'm-function' (not really different, as we see in a theorem below),

$$m_{_{TSO}} \ : \ \mathcal{A}ten \longrightarrow \mathcal{B}sm \ ,$$

in the following, somewhat indirect, way. (This is called *transition semantics* in **[Nip2]**.)

First, here is the list of so-called '1-step transitions': For all x, t, s, C, C', D and H,

$$(x \doteq t, s) \ \stackrel{1}{\rightarrow} \ (bugga, s_{x \mapsto s(t)})$$

$$((bugga; D), s) \ \stackrel{1}{\rightarrow} \ (D, s)$$

$$[\ C \neq C' \ \text{ and } \ (C, s) \ \stackrel{1}{\rightarrow} \ (C', s')\ ] \quad \Longrightarrow \quad ((C; D), s) \ \stackrel{1}{\rightarrow} \ ((C'; D), s')$$

$$H\mathbf{ff}@s \quad \Longrightarrow \quad (whdo(H)(C), s) \ \stackrel{1}{\rightarrow} \ (bugga, s)$$

$$H\mathbf{tt}@s \quad \Longrightarrow \quad (whdo(H)(C), s) \ \stackrel{1}{\rightarrow} \ ((C; whdo(H)(C)), s)$$

Firstly, *bugga* is merely some fixed command which we intuit as 'doing absolutely nothing'. So let's just choose some particular variable x_0 and define *bugga* to be $x_0 \doteq x_0$. It follows that $(bugga, s) \ \stackrel{1}{\rightarrow} \ (bugga, s)$.

Next, you should interpret the above five displays as simply defining a relation '$\stackrel{1}{\rightarrow}$' ; that is, as picking out a bunch of ordered pairs of objects, each of which is itself an ordered pair in the set $\mathcal{A}ten \times \mathcal{S}te$. The intuitive idea is that the command to the left of the '$\stackrel{1}{\rightarrow}$', initiated with the state next to it as input, will, after a single step of the computation, arrive at the state on the right, with 'what-remains-to-be-done' being the command on the right-hand side of the '$\stackrel{1}{\rightarrow}$' .

So of course *that* relation is not really what is wanted, but rather its *transitive closure*, namely '$\stackrel{n}{\rightarrow}$ for some n ', as defined below:

$(C, s) \ \stackrel{n}{\rightarrow} \ (C', s') \quad \Longleftrightarrow \quad \exists$ a sequence of length n

$$(C, s) = (C_0, s_0) \ \stackrel{1}{\rightarrow} \ (C_1, s_1) \ \stackrel{1}{\rightarrow} \ (C_2, s_2) \ \stackrel{1}{\rightarrow} \quad \cdots \quad \stackrel{1}{\rightarrow} \ (C_n, s_n) \ = \ (C', s') \ .$$

Definition. Define $m_{_{TSO}} \ : \ \mathcal{A}ten \longrightarrow \mathcal{B}sm$ by

$$(s, s') \in m_{_{TSO}}(C) \quad \Longleftrightarrow \quad (C, s) \ \stackrel{n}{\rightarrow} \ (bugga, s') \ \text{ for some } n \geq 0 \ .$$

Theorem TSO. *All the semantic functions already defined are actually the same function, i.e.*

$$m_{_{TSO}} \ = \ m_{_{BSO}} \ = \ m_{_{GSO}} \ = \ m_{_{DFX}} \ .$$

Stepwise Grundge Semantics.

Define yet another (again not really different) 'm-function',

$$m_{_{SGR}} \ : \ \mathcal{A}ten \longrightarrow \mathcal{B}sm \ ,$$

in the following way. This is again indirect, but rather more specific than tiny step semantics, in that it writes down explicitly (or at least inductively) the sequence of states produced by a command with a given input, as it is undergoing 'implementation'.

First, define for each $s \in \mathcal{S}te$, by induction on $i \geq 1$, then, for fixed i by induction on $C \in \mathcal{A}ten$, an object $Q_i(C,s) \in \mathcal{S}te \cup \{\nexists\}$, where, however suggestive, $\nexists$ is merely the name for some chosen (platonic) mathematical object which does not appear anywhere in this paper or any named heretofore by any human being. (*That's sort of a joke.*)

$$Q_1(x \div\!\!= t, s) \ := \ s_{x \mapsto s(t)} \quad ; \quad Q_{i+1}(x \div\!\!= t, s) \ := \ \nexists \ .$$

$$Q_1((C;D),s) \ := \ Q_1(C,s) \ ;$$

$$Q_{i+1}((C;D),s) \ := \ \begin{cases} Q_{i+1}(C,s) & \text{if } Q_{i+1}(C,s) \neq \nexists \ ; \\ Q_{i+1-\ell}(D, Q_\ell(C,s)) & \text{if } \exists \ell \leq i \text{ with} \\ & Q_\ell(C,s) \neq \nexists = Q_{\ell+1}(C,s) \ . \end{cases}$$

$$Q_1(whdo(H)(C),s) \ := \ \begin{cases} s & \text{if } H\mathbf{ff}@s \ ; \\ Q_1(C,s) & \text{if } H\mathbf{tt}@s \ ; \end{cases}$$

$$Q_{i+1}(whdo(H)(C),s) := \begin{cases} \nexists & \text{if } H\mathbf{ff}@s \ ; \\ Q_{i+1}(C,s) & \text{if } H\mathbf{tt}@s \text{ and } Q_{i+1}(C,s) \neq \nexists \ ; \\ Q_{i+1-\ell}(whdo(H)(C), Q_\ell(C,s)) & \text{if } H\mathbf{tt}@s \text{ and } \exists \ell \leq i \text{ with} \\ & Q_\ell(C,s) \neq \nexists = Q_{\ell+1}(C,s) \ . \end{cases}$$

Theorem SGR1. *This function is well-defined, as is clear from the following claim : By an induction on (i,C), performed simultaneously with making the above definition, the 'computation sequence'*

$$Q_*(C,s) \ := \ \{ \ Q_1(C,s) \ , \ Q_2(C,s) \ , \ Q_3(C,s) \ , \ \cdots \ \}$$

either consists entirely of states, or else it begins with a finite (positive) number of states followed by nothing but (an infinite sequence of) $\nexists$'s.

To simplify : $Q_1(C,s) \neq \nexists$, and, if $Q_k(C,s) = \nexists$, then $Q_{k+1}(C,s) = \nexists$.

Definitions. If the 2nd case in the theorem holds, select the unique ℓ with $Q_\ell(C,s) \neq \nexists = Q_{\ell+1}(C,s)$ and define

$$OUT(C,s) \ := \ Q_\ell(C,s) \ .$$

Now define the desired function $m_{_{SGR}} \ : \ \mathcal{A}ten \longrightarrow \mathcal{B}sm$ by

$$m_{_{SGR}}(C) \ := \ \{ \ (s, OUT(C,s)) \mid s \in \mathcal{S}te \text{ and } OUT(C,s) \text{ exists } \} \ .$$

Theorem SGR2. *All the semantic functions are actually the same function, i.e.*

$$m_{_{SGR}} \ = \ m_{_{TSO}} \ = \ m_{_{BSO}} \ = \ m_{_{GSO}} \ = \ m_{_{DFX}} \ .$$

The gigantic definition further up is perhaps best summarized by saying how it defines $Q_*(C,s)$:
For assignment commands, it writes down $s_{x \mapsto s(t)}$, then a never-ending string of $\nexists$'s.
For sequencing, it writes down Q_* for the first command. If that terminates, i.e. produces a never-ending string of $\nexists$'s to finish, it doesn't write those, but rather immediately writes down Q_* for the second command with the new state just arrived at.
For while-do, it checks the condition. If false, it writes down the input state, then the never-ending string of $\nexists$'s. If true, it writes down Q_* for the 'ingredient command'. If that terminates, i.e. produces a never-ending string of $\nexists$'s to finish, it doesn't write those, but rather starts over at the beginning with the new state just arrived at.
And, by **SGR1**, the unhappy need to actually inspect any of those never-ending strings of $\nexists$'s is nicely avoided! If an execution terminates, one does find that out; though if it doesn't, one might not. The reader might enjoy checking that fact directly for each of the five definitions of the semantics. Turing would not have it otherwise.

Remarks on the Appendix proofs of semantic equivalence.

(i) That the *same* answer always arises for an assignment command—i.e. that $m_{***}(x \doteqdot t) := \{(s, s_{x \mapsto s(t)}) \mid s \in \mathcal{S}te\}$ for any of the five $***$'s—is boringly simple in all cases, so we'll leave that out each time in those proofs, and just do the sequencing and while-do cases of each induction on commands.

(ii) **[Nip2]** claims "logical simplicity" results from an emphasis on machine theorem proving. Even if some of the details are a bit tedious, I guess that I'm maintaining (about the straightforward, once-and-for-all formula that is the giant step semantics) that proving, for all the other $***$ the fact that $m_{***} = m_{GSO}$, is conceptually about as straightforward as it gets. That's how each proof is done in the appendix. There seems to be nothing at all 'logically' complex in all this.

(iii) However, it would be interesting to see whether the 'tiny steps' of the TSO and the SGR semantics effectively coincide. And so that may be done below in some later version of this paper. If they do coincide, that would give a second proof of either **SGR2** or **TSO**.

(iv) The last, stepwise grundge, definition makes it immediately clear that $\mathcal{A}ten$ with its semantics is deterministic, hardly surprising. The corresponding fact needed to see this from the tiny step definition (and used a few more times below) is the following lemma.

Lemma. *For each (C, s) there is exactly one (C', s') such that $(C, s) \xrightarrow{1} (C', s')$. When $C = bugga$, we have $C' = C$ and $s' = s$. When $C \neq bugga$, we have $C' \neq C$.*

Sketch Proof. This uses unique readability of the strings which are commands (and so do the proofs of several other results here!). But that, together with inspecting the definition, does it; proceed by induction on C. Note that, when doing the case where C is sequencing, you've already proved that $(bugga, s) \xrightarrow{1} (bugga, s)$ is the only transition from $bugga$.

(v) The fact of deterministicism is also very easy to see from the giant step semantics, by induction on commands. It is interesting that the two or three approaches in **[W]** turn out to be the ones from which it is hardest to concoct a rigorous proof of this 'obvious' fact.

Section 1- : Single Simple Recursion.

The language $\mathcal{R}ec_{1-}$ is given by a structural induction as

$$x \leftarrow\!\!\div t \mid callY \;\; || \;\; (C;D) \mid ite(H)(C)(D) \mid \nabla YC$$

where C and D are names for commands (the strings which are the elements of that language); x is any variable in $1^{\underline{\text{st}}}$order; t is any $1^{\underline{\text{st}}}$order term; H is any quantifier-free $1^{\underline{\text{st}}}$order formula; and Y is a particular procedure variable, the only one in $\mathcal{P}cv$.

Intuitively, the command ∇YC is intended to convey what most of you will roughly understand by the following program from somebody somewhere's recursive Algol fragment:

$$\textsf{begin} \;\; \textit{declare } Y \textit{ to be } C \;;\; \textit{do } Y \;\; \textsf{end} \;.$$

Semantics is of course supposed to pin down precisely what is to be understood, at least up to the point when someone else tries to get a physical machine to carry it out!

As explained in Prelude B, we define

$$m : \mathcal{R}ec_{1-} \to \mathcal{B}sm \qquad \text{by} \qquad m(C) := \mathcal{M}(C, \emptyset) \;,$$

where

$$\mathcal{M} : \mathcal{R}ec_{1-} \times \mathcal{B}sm \;\to\; \mathcal{B}sm$$

is defined below by induction on commands. [We can simplify $(\mathcal{B}sm)^{\mathcal{P}cv}$ to just $\mathcal{B}sm$ here , as $\mathcal{P}cv = \{Y\}$ is merely a singleton.]

Definition and Theorem 1.1−. *By induction on C, the five clauses just below define $\mathcal{M}(C,\alpha)$ in such a way that, for each C, the map $[\beta \mapsto \mathcal{M}(C,\beta)]$ in the 5th clause is monotonic* (which justifies that clause via **P1**).

$\mathcal{M}(x \leftarrow\!\!\div t, \alpha) \;:=\; \{(s, s_{x\mapsto s(t)}) \mid s \in \mathcal{S}te\}$ *(independent of α) .*

$\mathcal{M}(callY, \alpha) \;:=\; \alpha \;.$

$\mathcal{M}((C;D), \alpha) \;:=\; \mathcal{M}(D,\alpha) \circ \mathcal{M}(C,\alpha) \;.$

$\mathcal{M}(ite(H)(C)(D), \alpha) \;:=\; \{(s,s') \mid H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C,\alpha);$ or
$H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(D,\alpha)\} \;.$

$$\mathcal{M}(\nabla YC,\alpha) \;:=\; lix[\,\beta \mapsto \mathcal{M}(C,\beta) \;:\; \mathcal{B}sm \to \mathcal{B}sm\,] \quad \textit{(independent of } \alpha\textit{)}\,.$$

Proof of monotonicity. For $C = callY$, the map is the identity function. And for C an assignment- or ∇-command, it is a constant function. So assume that monotonicity holds for $C = D$ and $C = E$ and that $\beta \subset \beta'$. Then, using **P3**, we get, for $C = (D;E)$,

$$\mathcal{M}((D;E),\beta) \;=\; \mathcal{M}(E,\beta)\circ\mathcal{M}(D,\beta) \;\subset\; \mathcal{M}(E,\beta')\circ\mathcal{M}(D,\beta') \;=\; \mathcal{M}((D;E),\beta')\,.$$

Finally, for $C = ite(H)(D)(E)$, supposing that $(s,s') \in \mathcal{M}(ite(H)(D)(E),\beta)$, we see that

if $H\mathbf{tt}@s$, then $(s,s') \in \mathcal{M}(D,\beta) \subset \mathcal{M}(D,\beta')$, so $(s,s') \in \mathcal{M}(ite(H)(D)(E),\beta')$; whereas

if $H\mathbf{ff}@s$, then $(s,s') \in \mathcal{M}(E,\beta) \subset \mathcal{M}(E,\beta')$, so $(s,s') \in \mathcal{M}(ite(H)(D)(E),\beta')$.

Definition. For any commands C and D, define by induction on C a command C_D—(AKA, in our inimitable substitution notation, $|C|^{all\ free\ callY \to D}$):

$$(x \leftarrow\!\!\!\!\cdot\, t)_D := x \leftarrow\!\!\!\!\cdot\, t \;\;;\;\; (callY)_D := D$$

$$(C';C'')_D := (C'_D;C''_D) \;;\; ite(H)(C')(C'')_D := ite(H)(C'_D)(C''_D) \;;\; (\nabla YC)_D := \nabla YC\,.$$

The last piece of the definition indicates that occurrences of $callY$ in ∇YC are not 'free'—the "∇" is like a quantifier acting on Y.

Proposition 1.2−. *For any C and D in $\mathcal{R}ec_{1-}$ and $\alpha \in \mathcal{B}sm$, we have*

$$\mathcal{M}(C_D,\alpha) \;=\; \mathcal{M}(C,\mathcal{M}(D,\alpha))\,.$$

Proof by induction on C. When $C = callY$, both sides are $\mathcal{M}(D,\alpha)$. When C has no free $callY$, we get $C_D = C$ and $\mathcal{M}(C,\beta)$ is independent of β, so that deals with assignment- and ∇-commands. When $C = (C';C'')$, we have

$$\mathcal{M}((C';C'')_D,\alpha) \;=\; \mathcal{M}((C'_D;C''_D),\alpha) \;=\; \mathcal{M}(C''_D,\alpha)\circ\mathcal{M}(C'_D,\alpha) \;=$$

$$\mathcal{M}(C'',\mathcal{M}(D,\alpha))\circ\mathcal{M}(C',\mathcal{M}(D,\alpha)) \;=\; \mathcal{M}((C';C''),\mathcal{M}(D,\alpha))\,.$$

Finally, for $C = ite(H)(C')(C'')$,

$$\mathcal{M}(ite(H)(C')(C'')_D, \alpha) \;=\; \mathcal{M}(ite(H)(C'_D)(C''_D), \alpha) \;=$$

$$\{(s,s') \mid H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C'_D, \alpha);\ \text{or} \quad H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(C''_D, \alpha)\} \quad =$$

$$\{(s,s') \mid H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C', \mathcal{M}(D,\alpha));\ \text{or} \quad H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(C'', \mathcal{M}(D,\alpha))\}$$

$$= \mathcal{M}(ite(H)(C')(C''), \mathcal{M}(D,\alpha)) \;.$$

Definition of "Unfolding". Define $C^{<N>}$ by induction on $N \geq 0$ as follows :

$$C^{<0>} \;:=\; callY \quad , \quad C^{<N+1>} \;:=\; C_{C^{<N>}} \;.$$

Corollary 1.3−. *For any C, we have*

$$\bigcup_{N\geq 0} m(C^{<N>}) \;\subset\; m(\nabla Y C) \;.$$

Proof. Proceed by induction on N to show $m(C^{<N>}) \;\subset\; m(\nabla Y C)$:

$$m(C^{<0>}) \;=\; \mathcal{M}(callY, \emptyset) \;=\; \emptyset \;\subset m(\nabla Y C) \;.$$

$$m(C^{<N+1>}) \;=\; m(C_{C^{<N>}}) \;=\; \mathcal{M}(C_{C^{<N>}}, \emptyset) \;=$$

$$\mathcal{M}(C, \mathcal{M}(C^{<N>}, \emptyset)) \;\subset\; \mathcal{M}(C, \mathcal{M}(\nabla Y C, \emptyset)) \;=\; \mathcal{M}(\nabla Y C, \emptyset) \;=\; m(\nabla Y C) \;.$$

The inclusion uses the inductive hypothesis and monotony. The equality just before it uses **1.2−**, and the one after it uses the fact that $\mathcal{M}(\nabla Y C, \emptyset)$ is a fixed point of $[\ \beta \mapsto \mathcal{M}(C,\beta)\ :\ \mathcal{B}sm \to \mathcal{B}sm\ ]$.

Proposition 1.4−. *If $\alpha_0 \subset \alpha_1 \subset \alpha_2 \subset \cdots$ is a chain in $\mathcal{B}sm$, then, for any C, we have*

$$\mathcal{M}(C, \bigcup_{N\geq 0} \alpha_N) \;=\; \bigcup_{N\geq 0} \mathcal{M}(C, \alpha_N) \;.$$

This says that $[\alpha \mapsto \mathcal{M}(C,\alpha)]$ is cts, but it's not till the next section that the 'cts concept' is really used.

Proof by induction on C. When $C = callY$, both sides are $\bigcup_{N\geq 0} \alpha_N$. When C has no free $callY$, the set $\mathcal{M}(C,\beta)$ is independent of β, so the result is clear for assignment- or ∇-commands. When $C = (C';C'')$, we have

$$\mathcal{M}((C';C''),\bigcup_{N\geq 0}\alpha_N) \;=\; \mathcal{M}(C'',\bigcup_{N\geq 0}\alpha_N)\circ\mathcal{M}(C',\bigcup_{N\geq 0}\alpha_N) \;=$$

$$\bigcup_{N\geq 0}\mathcal{M}(C'',\alpha_N)\circ\bigcup_{N\geq 0}\mathcal{M}(C',\alpha_N) \;=\; \bigcup_{N\geq 0}(\mathcal{M}(C'',\alpha_N)\circ\mathcal{M}(C',\alpha_N)) \;=$$

$$\bigcup_{N\geq 0}\mathcal{M}((C';C''),\alpha_N)\;,$$

the penultimate equality using **P4** and monotony. Finally, for $C = ite(H)(C')(C'')$,

$$(s,s')\in\mathcal{M}(ite(H)(C')(C''),\bigcup_{N\geq 0}\alpha_N) \iff$$

$$[H\mathbf{tt}@s \text{ and } (s,s')\in\mathcal{M}(C',\bigcup_{N\geq 0}\alpha_N) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s,s')\in\mathcal{M}(C'',\bigcup_{N\geq 0}\alpha_N)] \iff$$

$$[H\mathbf{tt}@s \text{ and } (s,s')\in\bigcup_{N\geq 0}\mathcal{M}(C',\alpha_N) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s,s')\in\bigcup_{N\geq 0}\mathcal{M}(C'',\alpha_N)] \iff$$

$$\exists N\,[H\mathbf{tt}@s \text{ and } (s,s')\in\mathcal{M}(C',\alpha_N) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s,s')\in\mathcal{M}(C'',\alpha_N)] \iff$$

$$\exists N\,[(s,s')\in\mathcal{M}(ite(H)(C')(C''),\alpha_N) \iff (s,s')\in\bigcup_{N\geq 0}\mathcal{M}(ite(H)(C')(C''),\alpha_N)\;.$$

Corollary 1.5−. *For any C, we have*

$$m(\nabla CY) \;\subset\; \bigcup_{N\geq 0} m(C^{<N>})\;.$$

Proof. Using **1.2−** and **1.4−** for the 3rd and 2nd equalities respectively, and noting that the penultimate one just adds a $\emptyset$,

$$\mathcal{M}(C,\bigcup_{N\geq 0} m(C^{<N>}) \;=\; \mathcal{M}(C,\bigcup_{N\geq 0}\mathcal{M}(C^{<N>},\emptyset)) \;=\; \bigcup_{N\geq 0}\mathcal{M}(C,\mathcal{M}(C^{<N>},\emptyset)) \;=$$

$$\bigcup_{N\geq 0}\mathcal{M}(C_{C^{<N>}},\emptyset) \;=\; \bigcup_{N\geq 0}\mathcal{M}(C^{<N+1>},\emptyset) \;=\; \bigcup_{N\geq 0}\mathcal{M}(C^{<N>},\emptyset) \;=\; \bigcup_{N\geq 0} m(C^{<N>})\,.$$

Thus, $\bigcup_{N\geq 0} m(C^{<N>})$ is a fixed point of $[\ \beta\mapsto\mathcal{M}(C,\beta)\ :\ \mathcal{B}sm\to\mathcal{B}sm\ ]$, and so must contain the least such fixed point, as required.

Theorem 1.6−. *The 'fixpoint' and the 'unfolding operational' definitions of the semantics of a recursive command agree; that is, for any C, we have*

$$m(\nabla CY) \;=\; \bigcup_{N\geq 0} m(C^{<N>}) \;.$$

This is immediate from the two corollaries.

Afterthought 1.7−. One can now give two quite different proofs of the unsurprising result that $\mathcal{R}ec_{1-}$ with its semantics is a *deterministic* command language. That is, all $m(C)$ are graphs of (partial) functions—i.e. of functions from a subset of $\mathcal{S}te$ to $\mathcal{S}te$.

On the one hand, directly from the definitions one can prove that if β is a graph then so is $\mathcal{M}(C,\beta)$. And then take $\beta = \emptyset$. To sketch the former, the best I know would prove simultaneously by induction on C both that fact and the fact that, for all relations β, if $\mathcal{M}(C,\beta) \subset \beta$, then there is a graph $\gamma \subset \beta$ such that $\mathcal{M}(C,\gamma) \subset \gamma$. The latter for D makes proof of the former easy for the case when C has the form ∇YD, and then the latter itself becomes trivial for that case. The other cases of the induction are quite mechanical.

On the other hand, the 'unfolding operational' definition is the right-hand side of the display above, where one gives the overall definition of $m(D)$ first by induction on the number of subcommands of D which have the form ∇YE for various E, and for that parameter fixed, by induction on D. The point is that the parameter takes one larger value on ∇CY than it does on all the $C^{<N>}$. And then to prove 'graphiness' one notes that an increasing union of graphs is a graph. Why does the union increase? Using monotonicity,

$$m(C^{<N>}) \;=\; \mathcal{M}(C^{<N>},\emptyset) \;\subset\; \mathcal{M}(C^{<N>},\mathcal{M}(C,\emptyset)) \;=\; \mathcal{M}((C^{<N>})_C,\emptyset) \;=\; m(C^{<N+1>})\,.$$

From where does that last equality come? There's more than one way to unfold a cat— we have another unsurprising fact :

$$(C^{<a>})_{C^{<b>}} \;=\; C^{<a+b>} \quad \text{for all } a \text{ and } b\;.$$

This is very quickly proved by induction on a from a simple generality : $(C_D)_E = C_{D_E}$, whose proof, for the 99th time, proceeds by a simple-minded bruteforce induction on C.

Question to denotational semanticizing experts : Don't you think that systematically using the adjoint makes this come out a lot simpler for beginners—that is, regarding $\mathcal{M}$ as a function from $\mathcal{R}ec \times \mathcal{B}sm$ to $\mathcal{B}sm$, rather than as a function from $\mathcal{R}ec$ to the functions from $\mathcal{B}sm$ to itself? In the next section, religiously observing the conventions of denotational semantics produces an even more opaque object, a function from $\mathcal{R}ec$ to the functions from the functions from $\mathcal{P}cv$ to $\mathcal{B}sm$ to $\mathcal{B}sm$!! (No, that's not a typo. In [**deBa**] one finds plenty of very nearly unreadable stretches along these lines, at least for an old dog like me.) That we will conscientiously object to imitating. The adjoint is far more friendly!

Is this really slicker?

Given C, let $F(\beta) = \mathcal{M}(C, \beta)$, and calculate as follows.

$$m(\nabla CY) \;=\; \mathcal{M}(\nabla CY, \emptyset) \;=\; lix(F)$$

$$= \bigcup_{n=0}^{\infty} F^n(\emptyset) \;=\; \emptyset \;\cup\; F(\emptyset) \;\cup\; F^2(\emptyset) \;\cup\; F^3(\emptyset) \;\cup\; \cdots$$

$$= \mathcal{M}(C, \emptyset) \;\cup\; \mathcal{M}(C, F(\emptyset)) \;\cup\; \mathcal{M}(C, F(F(\emptyset))) \;\cup\; \cdots$$

$$= \mathcal{M}(C, \emptyset) \;\cup\; \mathcal{M}(C, \mathcal{M}(C, \emptyset)) \;\cup\; \mathcal{M}(C, \mathcal{M}(C, \mathcal{M}(C, \emptyset))) \;\cup\; \cdots$$

$$= m(C) \;\cup\; \mathcal{M}(C_C, \emptyset) \;\cup\; \mathcal{M}(C, \mathcal{M}(C_C, \emptyset)) \;\cup\; \cdots$$

$$= m(C) \;\cup\; \mathcal{M}(C^{<2>}, \emptyset) \;\cup\; \mathcal{M}(C, \mathcal{M}(C^{<2>}, \emptyset)) \;\cup\; \cdots$$

$$= m(C) \;\cup\; m(C^{<2>}) \;\cup\; \mathcal{M}(C_{C^{<2>}}, \emptyset)) \;\cup\; \cdots$$

$$= m(C) \;\cup\; m(C^{<2>}) \;\cup\; \mathcal{M}(C^{<3>}, \emptyset)) \;\cup\; \cdots$$

$$= m(C^{<1>}) \;\cup\; m(C^{<2>}) \;\cup\; m(C^{<3>}) \;\cup\; \cdots \;=\; \bigcup_N m(C^{<N>}) \;\; .$$

The 3rd equality uses the alternative formula $(***)$ for the fixed point. In reality, this is not much shorter than what we already did: That alternative formula is known to be correct only after knowing that F is cts, so **1.4**− has to be proved. And the formula in **1.2**− is used more than a finite number of times in the calculation just above. However, we can replace the arguments for both corollaries by the manipulation above. And in fact the proof of monotonicity in **1.1**− is unnecessary—but we already knew it was redundant, in case you hadn't noticed, since cts implies monotonic. Below,

we shall write Section $\infty+$ in the most austere and economic style that we know, by employing these shortcuts.

But I think the most interesting thing here is the connection between that beautiful, simple formula for *lix* and the desire to show that operational and fixpoint semantics coincide. The book [**deBa**] avoids dealing with the connection between the operational and fixpoint semantics uniquely for this type of language with recursive commands (or even giving the operational semantics), and its author is a serious follower of Dana Scott. And I've never seen our **1.2**− before. So I may be surprised to later learn that this connection had gone unnoticed.

Section 1+ : Single General Recursion.

In the previous section, having only one procedure variable, namely Y, made things simpler in a few superficial ways, and also precluded *mutual* recursion. But what to us here is the somewhat significant simplification is that there could be no 'free' $callX$, where $X \neq Y$, occurring in ∇YC, i.e. in C. Such an occurrence would produce an interesting effect at times, for example when one had a command ∇XD for which ∇YC was a subcommand of D. It would have been perfectly feasible to have allowed $\mathcal{P}cv$ to be general in the last section (that is, any collection of procedure variables), as long as we imposed the condition that the occurrence above was not allowed. Details in that section would have really not changed. But that generality would be a waste of time. Here we still avoid mutual recursion, but do permit the above 'irregularity' or 'lack of simplicity'—hence the "general" of the section title.

Because we need to replace variables with 'brand new ones' in the definition of substitution below (reminiscent of "renaming bound variables" in logic), we had better assume that $\mathcal{P}cv$ is an infinite set from now on.

Thus the language here, called $\mathcal{R}ec_{1+}$, is given by an identical structural induction to that for $\mathcal{R}ec_{1-}$, except that the "Y" in $callY$ and ∇YC can be any element from the now general (typically countably infinite) set $\mathcal{P}cv$ of procedure variables.

In order to produce a 'fixpoint' definition, as explained in Prelude B, and generalizing the previous section, we define

$$\mathcal{M} : \mathcal{R}ec_{1+} \times \mathcal{B}sm^{\mathcal{P}cv} \ \rightarrow \ \mathcal{B}sm$$

below by induction on commands. Then we can define the basic semantics of commands

$$m : \mathcal{R}ec_{1+} \rightarrow \mathcal{B}sm \qquad \text{by} \qquad m(C) := \mathcal{M}(C, \Phi) \ ,$$

where Φ maps all of $\mathcal{P}cv$ to $\emptyset$, the empty set. (Effectively, the 'real' semantics of the atomic *call* commands is to universally diverge.)

As with our notation for states, recall that $\Theta_{Y \mapsto \beta}$ is the element of $\mathcal{B}sm^{\mathcal{P}cv}$ which agrees everywhere with Θ, except that it maps Y to β.

Definition and Theorem 1.1+. *By induction on C, the five clauses just below define $\mathcal{M}(C,\Theta)$ in such a way that, for each C,Θ and Y, the map in the 5th clause, namely $[\beta \mapsto \mathcal{M}(C,\Theta_{Y\mapsto\beta})]$, is monotonic* (which justifies that clause via **P1**).

$\mathcal{M}(x \doteqdot t,\Theta) \;:=\; \{(s, s_{x\mapsto s(t)}) \mid s \in \mathcal{S}te\}$ *(independent of Θ) .*

$\mathcal{M}(callX,\Theta) \;:=\; \Theta(X)$.

$\mathcal{M}((C;D),\Theta) \;:=\; \mathcal{M}(D,\Theta)\circ\mathcal{M}(C,\Theta)$.

$\mathcal{M}(ite(H)(C)(D),\Theta) \;:=\; \{(s,s') \mid H\mathbf{tt}@s$ and $(s,s') \in \mathcal{M}(C,\Theta)$; or $H\mathbf{ff}@s$ and $(s,s') \in \mathcal{M}(D,\Theta)\}$.

$\mathcal{M}(\nabla YC,\Theta) \;:=\; lix[\beta \mapsto \mathcal{M}(C,\Theta_{Y\mapsto\beta}) \;:\; \mathcal{B}sm \to \mathcal{B}sm\,]$.

Proof of monotonicity. For $C = callY$, the map is the identity function. And for C an assignment command or $callX$ with $X \neq Y$, it is a constant function. For the next two command constructors, assume that monotonicity holds for $C = D$ and $C = E$ and that $\beta \subset \beta'$. Then, using **P3**, we get, for $C = (D;E)$,

$$\mathcal{M}(C,\Theta_{Y\mapsto\beta}) \;=\; \mathcal{M}(E,\Theta_{Y\mapsto\beta})\circ\mathcal{M}(D,\Theta_{Y\mapsto\beta}) \;\subset\; \mathcal{M}(E,\Theta_{Y\mapsto\beta'})\circ\mathcal{M}(D,\Theta_{Y\mapsto\beta'}) \;=\; \mathcal{M}(C,\Theta_{Y\mapsto\beta'})\,.$$

For $C = ite(H)(D)(E)$, supposing that $(s,s') \in \mathcal{M}(ite(H)(D)(E),\Theta_{Y\mapsto\beta})$, we see that
if $H\mathbf{tt}@s$, then $(s,s') \in \mathcal{M}(D,\Theta_{Y\mapsto\beta}) \subset \mathcal{M}(D,\Theta_{Y\mapsto\beta'})$,
so $(s,s') \in \mathcal{M}(ite(H)(D)(E),\Theta_{Y\mapsto\beta'})$;
whereas if $H\mathbf{ff}@s$, then $(s,s') \in \mathcal{M}(E,\Theta_{Y\mapsto\beta}) \subset \mathcal{M}(E,\Theta_{Y\mapsto\beta'})$,
so $(s,s') \in \mathcal{M}(ite(H)(D)(E),\Theta_{Y\mapsto\beta'})$.

For $C = \nabla YD$, the function at issue, namely $[\beta \mapsto \mathcal{M}(C,\Theta_{Y\mapsto\beta})]$, is

$$[\beta \mapsto lix[\alpha \mapsto \mathcal{M}(D,(\Theta_{Y\mapsto\beta})_{Y\mapsto\alpha})]]\,.$$

But $(\Theta_{Y\mapsto\beta})_{Y\mapsto\alpha} \;=\; \Theta_{Y\mapsto\alpha}$, so the function is constant.

Finally, for $C = \nabla XD$ where $X \neq Y$ and where monotonicity holds for D and any choice of Θ, we are dealing with the function

$$[\beta \mapsto lix[\alpha \mapsto \mathcal{M}(D,(\Theta_{Y\mapsto\beta})_{X\mapsto\alpha})]]\,.$$

To prove its monotonicity, fix $\beta \subset \beta_1$. Define $F(\alpha) := \mathcal{M}(D, \Theta_{X\mapsto\alpha, Y\mapsto\beta})$. (The order in Θ's subscript is irrelevant.) Define another self-map, F_1, of $\mathcal{B}sm$ by the same formula as F except using β_1 in place of β. We are required to show that the least fixed point for F is a subset of that for F_1. In order to do that, we will use the simple formula for lix in **P1**. Applying the inductive hypothesis for D with $\Theta_{X\mapsto\alpha}$, it is immediate that

$$\forall\alpha \;\; F(\alpha) \;\subset\; F_1(\alpha) \;.$$

This immediately implies

$$F_1(\alpha) \;\subset\; \alpha \quad\Longrightarrow\quad F(\alpha) \;\subset\; \alpha$$

Thus, as required

$$lix(F) \;=\; \bigcap\{\; \alpha \mid F(\alpha) \subset \alpha \;\} \;\subset\; \bigcap\{\; \alpha \mid F_1(\alpha) \subset \alpha \;\} \;=\; lix(F_1) \;.$$

Diversion concerning substitution.

Substitution of strings, replacing substrings of other strings, tends to be a minor bugbear in logic, and a major one in theoretical CS. There are at least two types of mess-ups that can happen :

(i) Simultaneous performance of two or more substitutions usually does not give the same result as successively doing those substitutions. So we shall need to give the somewhat more involved definitions of simultaneous substitutions in the two following sections, to get what we want.

(ii) When there is a notion of variables being 'free' versus 'bound', one wants to substitute for a free variable occurrence in such a way that free occurrences within the string (being stuffed into the 'big' one in place of that variable) remain free in the 'new big one'. (Pardon the language!) This is our concern in this section, in defining $C_{D/Y}$, which is supposed to be, for some procedure variable Y, the command obtained by substituting command D for all free occurrences, in command C, of $callY$.

The main aim is to prove (and the only place these unfortunate technicalities play a role here is in proving) the formula

$$\mathcal{M}(C_{D/Y}, \Theta) \;=\; \mathcal{M}(C, \Theta_{Y\mapsto\mathcal{M}(D,\Theta)}) \;.$$

But before getting into it, let us give an example showing that a simple-minded definition which doesn't account for (ii) above would give a counter-example to that formula.

Take $X \neq Y$, and define

$$pseudo(\nabla XE)_{D/Y} \ := \ \nabla X(E_{D/Y}) \ .$$

Now change $C_{D/Y}$ to $pseudo(\nabla XE)_{D/Y}$ in the formula-to-be-proved, with $D = callX$ and $E = callY$. Then the left-hand side in that formula is (using the other parts below of the definition of $C_{D/Y}$):

$$\mathcal{M}(\nabla X callX, \Theta) \ = \ lix[\beta \mapsto \mathcal{M}(callX, \Theta_{X\mapsto\beta})] \ = \ lix[\beta \mapsto \Theta_{X\mapsto\beta}(X)] \ = \ lix[\beta \mapsto \beta] \ = \ \emptyset \, .$$

But the right-hand side in the formula is

$$\mathcal{M}(\nabla X callY, \Theta_{Y\mapsto\mathcal{M}(callX,\Theta)}) \ = \ \mathcal{M}(\nabla X callY, \Theta_{Y\mapsto\Theta(X)}) \ =$$

$$lix[\beta \mapsto \mathcal{M}(callY, (\Theta_{Y\mapsto\Theta(X)})_{X\mapsto\beta})] \ = \ lix[\beta \mapsto (\Theta_{Y\mapsto\Theta(X)})_{X\mapsto\beta}(Y)] \ =$$

$$lix[\beta \mapsto \Theta(X)] \ = \ \Theta(X) \ .$$

Thus, when Θ is chosen with $\Theta(X) \neq \emptyset$, the identity fails, were we to use this naive definition of substitution.

End of diversion.

Definition. For any commands C and D, and any $Y \in \mathcal{P}cv$, define, by induction on the number of symbols in the string C, a new command denoted $C_{D/Y}$—(AKA, in our inimitable substitution notation, $|C|^{all\ free\ call\not{Y}\mapsto D}$), where $X \neq Y$ both places below :

$$(x \dot{\leftarrow} t)_{D/Y} := x \dot{\leftarrow} t \ \ ; \ \ (callY)_{D/Y} := D \ \ ; \ \ (callX)_{D/Y} := callX$$

$$(C';C'')_{D/Y} := (C'_{D/Y};C''_{D/Y}) \ ; \ ite(H)(C')(C'')_{D/Y} := ite(H)(C'_{D/Y})(C''_{D/Y}) \ ;$$

$$(\nabla YE)_{D/Y} := \nabla YE \ .$$

This penultimate piece of the definition indicates that occurrences of $callY$ in ∇YE are not 'free'—the "∇" is like a quantifier acting on Y. The final piece of the definition has the technicalities referred to in the diversion. Recall $X \neq Y$ and define

$$(\nabla XE)_{D/Y} := \nabla X'(E'_{D/Y}) \ ,$$

where $E' := E_{callX'/X}$ (noting that E' has the same number of symbols as E, two less than that for ∇XE, so the inductive definition is meaningful) and where X' is chosen to be the first procedure variable (temporarily give $\mathcal{P}cv$ a linear order for this purpose only) such that

$$X' \neq X \ ; \ X' \neq Y \ ; \ X' \notin D \ ; \ X' \notin E \ \text{(meaning that no } callX' \text{ occurs)} .$$

[The right-hand side two displays above sloppily uses a pair of brackets not really acceptable in strict syntax; that criticism applies equally to the wildly popular 'backslash notation' for substitution, made even more unreadable by its use also in place of our $s_{x\mapsto t}$, which has nothing to do with substitution. The '| |-notation' suffers neither of these drawbacks.]

Proposition 1.2+. *For any C and D in $\mathcal{R}ec_{1+}$, any $Y \in \mathcal{P}cv$, and any $\Theta \in \mathcal{B}sm^{\mathcal{P}cv}$, we have*

$$\mathcal{M}(C_{D/Y}, \Theta) \ = \ \mathcal{M}(C, \Theta_{Y\mapsto\mathcal{M}(D,\Theta)}) .$$

[This gives a utility to defining $\mathcal{M}(-,\Theta)$ for Θ other than Φ, quite apart from its utility to formulating fixpoint definitions.]

Proof by induction on the number of symbols in C. When C is $callY$, both sides are $\mathcal{M}(D,\Theta)$. When C has no free $callY$, we get $C_{D/Y} = C$ and $\mathcal{M}(C,\Theta)$ is independent of $\Theta(Y)$, so that deals with assignment commands and with $callX$ for all $X \neq Y$. When $C = (C';C'')$, we have

$$\mathcal{M}((C';C'')_{D/Y},\Theta) \ = \ \mathcal{M}((C'_{D/Y};C''_{D/Y}),\Theta) \ = \ \mathcal{M}(C''_{D/Y},\Theta)\circ\mathcal{M}(C'_{D/Y},\Theta) \ =$$

$$\mathcal{M}(C'',\Theta_{Y\mapsto\mathcal{M}(D,\Theta)}) \circ \mathcal{M}(C',\Theta_{Y\mapsto\mathcal{M}(D,\Theta)}) \ = \ \mathcal{M}((C';C''),\Theta_{Y\mapsto\mathcal{M}(D,\Theta)}) .$$

For $C = ite(H)(C')(C'')$,

$$\mathcal{M}(ite(H)(C')(C'')_{D/Y},\Theta) \ = \ \mathcal{M}(ite(H)(C'_{D/Y})(C''_{D/Y}),\Theta) \ =$$

$$\{(s,s') \mid H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C'_{D/Y},\Theta); \text{ or } H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(C''_{D/Y},\Theta)\} \ =$$

$$\{(s,s') \mid H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C',\Theta_{Y\mapsto\mathcal{M}(D,\Theta)}); \text{ or } H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(C'',\Theta_{Y\mapsto\mathcal{M}(D,\Theta)})\} \ =$$

$$\mathcal{M}(ite(H)(C')(C''),\Theta_{Y\mapsto\mathcal{M}(D,\Theta)}) .$$

When $C = \nabla YE$, we have $C_{D/Y} = C$ and both sides, $\mathcal{M}(C_{D/Y},\Theta)$ and $\mathcal{M}(C,\Theta_{Y\mapsto\mathcal{M}(D,\Theta)})$, become $lix[\beta \mapsto \mathcal{M}(E,\Theta_{Y\mapsto\beta})]$, since

$$(\Theta_{Y\mapsto\mathcal{M}(D,\Theta)})_{Y\mapsto\beta} = \Theta_{Y\mapsto\beta} .$$

When $C = \nabla X E$ with $X \neq Y$, and inductively noting that the identity holds when C is E or E' with any choice of Θ, the left side is

$$\mathcal{M}(C_{D/Y}, \Theta) \;=\; \mathcal{M}(\nabla X'(E'_{D/Y}), \Theta) \;=\; lix[\beta \mapsto \mathcal{M}(E'_{D/Y}, \Theta_{X' \mapsto \beta})] \;=$$

$$lix[\beta \mapsto \mathcal{M}(E', (\Theta_{X' \mapsto \beta})_{Y \mapsto \mathcal{M}(D, \Theta_{X' \mapsto \beta})})] \;=$$

$$lix[\beta \mapsto \mathcal{M}(E', \Psi)] \quad \text{where} \;\; \Psi \;:=\; (\Theta_{X' \mapsto \beta})_{Y \mapsto \mathcal{M}(D,\Theta)}$$

$$= \; lix[\beta \mapsto \mathcal{M}(E, \Psi_{X \mapsto \mathcal{M}(call X', \Psi)})] \;=\; lix[\beta \mapsto \mathcal{M}(E, \Psi_{X \mapsto \Psi(X')})] \;=$$

$$lix[\beta \mapsto \mathcal{M}(E, \Psi_{X \mapsto \beta})] \;=\; lix[\beta \mapsto \mathcal{M}(E, ((\Theta_{X' \mapsto \beta})_{Y \mapsto \mathcal{M}(D,\Theta)})_{X \mapsto \beta})] \;=$$

$$lix[\beta \mapsto \mathcal{M}(E, (\Theta_{Y \mapsto \mathcal{M}(D,\Theta)})_{X \mapsto \beta})] \,.$$

The right side,

$$\mathcal{M}(C, \Theta_{Y \mapsto \mathcal{M}(D,\Theta)}) \;=\; \mathcal{M}(\nabla X E, \Theta_{Y \mapsto \mathcal{M}(D,\Theta)})$$

is that latter least fixed point, and so the proof is complete, except for explaining the long sequence of equalities in the earlier display, including one simple lemma :

From left-to-right, the equalities hold by: definition of $(\nabla X E)_{D/Y}$; definition of ∇-command semantics ; inductive hypothesis applied to E' ; the lemma below and because $X' \notin D$; definition of E' and the inductive hypothesis applied to E ; semantics of the *call*-command ; $\Psi(X') = \beta$; the $(X' \mapsto \beta)$-part of the definition of Ψ ; overall definition of Ψ ; the lemma below and because $X' \notin E$ and X', X and Y are distinct procedure variables, which allows us to permute the three subscripts .

The free occurrences of $callX$ in a command are:
none for assignment commands, or for $callY$ when $Y \neq X$;
the whole command for $callX$;
those in C and/or in D for $(C; D)$ and for $ite(H)(C)(D)$;
none for $\nabla X C$;
and finally, those in C for $\nabla Y C$ if $X \neq Y$.

Lemma. *If a command E has no free $callX$ then, for any Ψ and γ,*

$$\mathcal{M}(E, \Psi_{X \mapsto \gamma}) \;=\; \mathcal{M}(E, \Psi) \,.$$

The proof is by induction on E. The cases of atomic commands are easy as usual, and of sequencing and '*ite*ing' are the expected manipulations.

When $E = \nabla VF$ with $X \neq V$, the right-hand side is

$$lix[\ \beta \mapsto \mathcal{M}(F, \Psi_{V\mapsto\beta})\ ]\ ,$$

whereas the left-hand side is

$$lix[\ \beta \mapsto \mathcal{M}(F, (\Psi_{X\mapsto\gamma})_{V\mapsto\beta})\ ]\ ,$$

But $X \neq V$, so we can permute those last subscripts, and then drop the $X \mapsto \gamma$ one, by the inductive hypothesis applied to F, as required.

When $E = \nabla XF$, both sides are

$$lix[\beta \mapsto \mathcal{M}(F, \Psi_{X\mapsto\beta})]\ , \quad \text{since} \quad (\Psi_{X\mapsto\gamma})_{X\mapsto\beta} = \Psi_{X\mapsto\beta}\ .$$

Definition of "Unfolding". Define $C_Y^{<N>}$ by induction on $N \geq 0$ as follows :

$$C_Y^{<0>}\ :=\ callY \quad , \quad C_Y^{<N+1>}\ :=\ C_{C_Y^{<N>}/Y}\ .$$

Corollary 1.3+. *For any C, we have*

$$\bigcup_{N\geq 0} m(C_Y^{<N>})\ \subset\ m(\nabla CY)\ .$$

Proof. Proceed by induction on N to show $m(C_Y^{<N>})\ \subset\ m(\nabla CY)$:

$$m(C_Y^{<0>})\ =\ \mathcal{M}(callY, \Phi)\ =\ \Phi(Y)\ =\ \emptyset\ \subset m(\nabla CY)\ .$$

$$m(C_Y^{<N+1>})\ =\ m(C_{C_Y^{<N>}/Y})\ =\ \mathcal{M}(C_{C_Y^{<N>}/Y}, \Phi)\ =\ \mathcal{M}(C, \Phi_{Y\mapsto\mathcal{M}(C_Y^{<N>},\Phi)})\ =$$
$$\mathcal{M}(C, \Phi_{Y\mapsto m(C_Y^{<N>})})\ \subset\ \mathcal{M}(C, \Phi_{Y\mapsto m(\nabla CY)})\ =\ m(\nabla CY)\ .$$

The inclusion uses the inductive hypothesis and monotonicity. The equality two before it uses **1.2+**, and the one after it is the fact that $m(\nabla CY)$, which equals $\mathcal{M}(\nabla CY, \Phi)$, is a fixed point of

$$[\ \beta \mapsto \mathcal{M}(C, \Phi_{Y\mapsto\beta})\ :\ \mathcal{B}sm \to \mathcal{B}sm\ ]\ .$$

Proposition 1.4+ *If $\alpha_0 \subset \alpha_1 \subset \alpha_2 \subset \cdots$ is a chain in $\mathcal{B}sm$, then, for any C and Θ, we have*

$$\mathcal{M}(C, \Theta_{Y \mapsto \bigcup_{N\geq 0} \alpha_N}) \;=\; \bigcup_{N\geq 0} \mathcal{M}(C, \Theta_{Y\mapsto \alpha_N}) \;.$$

Equivalently, $[\alpha \mapsto \mathcal{M}(C, \Theta_{Y\mapsto\alpha})]$ is cts.

Proof by induction on C. When $C = callY$, both sides are $\bigcup_{N\geq 0} \alpha_N$. When $C = callX$ for $X \neq Y$, both sides are $\Theta(X)$. The result is clear for assignment commands, where $\mathcal{M}(C, \Theta)$ is independent of Θ.

When $C = (C'; C'')$, we have

$$\mathcal{M}((C';C''), \Theta_{Y\mapsto\bigcup_{N\geq 0}\alpha_N}) \;=\; \mathcal{M}(C'', \Theta_{Y\mapsto\bigcup_{N\geq 0}\alpha_N}) \circ \mathcal{M}(C', \Theta_{Y\mapsto\bigcup_{N\geq 0}\alpha_N}) \;=$$

$$\bigcup_{N\geq 0} \mathcal{M}(C'', \Theta_{Y\mapsto\alpha_N}) \circ \bigcup_{N\geq 0} \mathcal{M}(C', \Theta_{Y\mapsto\alpha_N}) \;=$$

$$\bigcup_{N\geq 0} \left(\mathcal{M}(C'', \Theta_{Y\mapsto\alpha_N}) \circ \mathcal{M}(C', \Theta_{Y\mapsto\alpha_N})\right) \;=\; \bigcup_{N\geq 0} \mathcal{M}((C';C''), \Theta_{Y\mapsto\alpha_N}) \;,$$

the penultimate equality using **P4**.

For $C = ite(H)(C')(C'')$,

$$(s, s') \in \mathcal{M}(ite(H)(C')(C''), \Theta_{Y\mapsto\bigcup_{N\geq 0}\alpha_N}) \iff$$

$$[H\mathbf{tt}@s \text{ and } (s, s') \in \mathcal{M}(C', \Theta_{Y\mapsto\bigcup_{N\geq 0}\alpha_N}) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s, s') \in \mathcal{M}(C'', \Theta_{Y\mapsto\bigcup_{N\geq 0}\alpha_N})] \iff$$

$$[H\mathbf{tt}@s \text{ and } (s, s') \in \bigcup_{N\geq 0} \mathcal{M}(C', \Theta_{Y\mapsto\alpha_N}) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s, s') \in \bigcup_{N\geq 0} \mathcal{M}(C'', \Theta_{Y\mapsto\alpha_N})] \iff$$

$$\exists N\, [H\mathbf{tt}@s \text{ and } (s, s') \in \mathcal{M}(C', \Theta_{Y\mapsto\alpha_N}) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s, s') \in \mathcal{M}(C'', \Theta_{Y\mapsto\alpha_N})] \iff$$

$$\exists N\, [(s, s') \in \mathcal{M}(ite(H)(C')(C''), \Theta_{Y\mapsto\alpha_N}) \;.$$

When $C = \nabla YE$, note that, for any α, the set

$$\mathcal{M}(\nabla YE, \Theta_{Y\mapsto\alpha}) \;=\; lix[\beta \mapsto \mathcal{M}(C, (\Theta_{Y\mapsto\alpha})_{Y\mapsto\beta})\,] \;=\; lix[\beta \mapsto \mathcal{M}(C, \Theta_{Y\mapsto\beta})\,] \;,$$

which is independent of α. So here the two sides of the identity merely assert that a set, and an infinite union of 'copies' of the same set, do not differ.

When $C = \nabla XE$ for $X \neq Y$, and the result holds for E, we have

$$\mathcal{M}(\nabla XE, \Theta_{Y\mapsto\bigcup_{N\geq 0}\alpha_N}) \;=\; lix[\beta \mapsto \mathcal{M}(E, (\Theta_{Y\mapsto\bigcup_{N\geq 0}\alpha_N})_{X\mapsto\beta}) \,] \;=$$

$$lix[\beta \mapsto \mathcal{M}(E, (\Theta_{X\mapsto\beta})_{Y\mapsto\bigcup_{N\geq 0}\alpha_N}) \,] \;=\; lix[\beta \mapsto \bigcup_{N\geq 0} \mathcal{M}(E, (\Theta_{X\mapsto\beta})_{Y\mapsto\alpha_N}) \,] \,,$$

whereas

$$\bigcup_{N\geq 0} \mathcal{M}(\nabla XE, \Theta_{Y\mapsto\alpha_N}) \;=\; \bigcup_{N\geq 0} lix[\beta \mapsto \mathcal{M}(E, (\Theta_{Y\mapsto\alpha_N})_{X\mapsto\beta}) \,] \,.$$

Since the subscripts on Θ clearly 'commute', the equality of the right-hand sides is fundamentally 'moving the union symbol', and is immediate from **P2**(iii). We can use the case of the latter with F being the function mapping β to $\mathcal{M}(E, \Theta_{X\mapsto\beta})$, because the inductive assumption gives the required 'ctsness' of that F.

Corollary 1.5+. *For any C, we have*

$$m(\nabla CY) \;\subset\; \bigcup_{N\geq 0} m(C_Y^{<N>}) \,.$$

Proof. Using **1.2+** and **1.4+** for the 3rd and 2nd equalities respectively,

$$\mathcal{M}(C, \Phi_{Y\mapsto\bigcup_{N\geq 0} m(C_Y^{<N>})}) \;=\; \mathcal{M}(C, \Phi_{Y\mapsto\bigcup_{N\geq 0}\mathcal{M}(C_Y^{<N>},\Phi)}) \;=\; \bigcup_{N\geq 0} \mathcal{M}(C, \Phi_{Y\mapsto\mathcal{M}(C_Y^{<N>},\Phi)}) \;=$$

$$\bigcup_{N\geq 0} \mathcal{M}(C_{C_Y^{<N>}/Y}, \Phi) \;=\; \bigcup_{N\geq 0} \mathcal{M}(C_Y^{<N+1>}, \Phi) \;=\; \bigcup_{N\geq 0} \mathcal{M}(C_Y^{<N>}, \Phi) \;=\; \bigcup_{N\geq 0} m(C_Y^{<N>}) \,.$$

For the penultimate equality, we've just added one empty set to the union. Thus, the set $\bigcup_{N\geq 0} m(C_Y^{<N>})$ is a fixed point of

$$[\; \beta \mapsto \mathcal{M}(C, \Phi_{Y\mapsto\beta}) \;:\; \mathcal{B}sm \to \mathcal{B}sm \;] \,,$$

and so must contain the least such fixed point, as required.

Theorem 1.6+. *We have*

$$m(\nabla CY) \;=\; \bigcup_{N\geq 0} m(C_Y^{<N>}) \,.$$

This is immediate from the two corollaries.

Afterthought 1.7+. One has the unsurprising result that $\mathcal{R}ec_{1+}$ with its semantics is a *deterministic* command language. That is, all $m(C)$ are graphs of (partial) functions—i.e. of functions from a subset of $\mathcal{S}te$ to $\mathcal{S}te$. But here I do not know how to convert the previous theorem into an 'unfolding operational' definition, so that line of proof from Section 1− is not available. One proves inductively on C that, if $\Theta(X)$ is a graph for all $X \in \mathcal{P}cv$, then $\mathcal{M}(C, \Theta)$ is also a graph. And then take $\Theta = \Phi$.

Is this really slicker?

Given C and Y, let $F(\beta) = \mathcal{M}(C, \Phi_{Y \mapsto \beta})$, and calculate as follows.

$$m(\nabla CY) \;=\; \mathcal{M}(\nabla CY, \Phi) \;=\; lix(F)$$

$$= \bigcup_{n=0}^{\infty} F^n(\emptyset) \;=\; \emptyset \;\cup\; F(\emptyset) \;\cup\; F^2(\emptyset) \;\cup\; \cdots$$

$$= \mathcal{M}(C, \Phi_{Y \mapsto \emptyset}) \;\cup\; \mathcal{M}(C, \Phi_{Y \mapsto F(\emptyset)}) \;\cup\; \mathcal{M}(C, \Phi_{Y \mapsto F(F(\emptyset))}) \;\cup\; \cdots$$

$$= \mathcal{M}(C, \Phi) \;\cup\; \mathcal{M}(C, \Phi_{Y \mapsto \mathcal{M}(C,\Phi)}) \;\cup\; \mathcal{M}(C, \Phi_{Y \mapsto \mathcal{M}(C, \Phi_{Y \mapsto \mathcal{M}(C,\Phi)})}) \;\cup\; \cdots$$

$$= m(C) \;\cup\; \mathcal{M}(C_{C/Y}, \Phi) \;\cup\; \mathcal{M}(C, \Phi_{Y \mapsto \mathcal{M}(C_{C/Y}, \Phi)}) \;\cup\; \cdots$$

$$= m(C) \;\cup\; \mathcal{M}(C_Y^{<2>}, \Phi) \;\cup\; \mathcal{M}(C, \Phi_{Y \mapsto \mathcal{M}(C_Y^{<2>}, \Phi)}) \;\cup\; \cdots$$

$$= m(C) \;\cup\; m(C_Y^{<2>}) \;\cup\; \mathcal{M}(C_{C_Y^{<2>}/Y}, \Phi) \;\cup\; \cdots$$

$$= m(C) \;\cup\; m(C_Y^{<2>}) \;\cup\; \mathcal{M}(C_Y^{<3>}, \Phi) \;\cup\; \cdots$$

$$= m(C_Y^{<1>}) \;\cup\; m(C_Y^{<2>}) \;\cup\; m(C_Y^{<3>}) \;\cup\; \cdots \;=\; \bigcup_N m(C_Y^{<N>}) \;.$$

The 3rd equality uses the alternative formula $(***)$ for the fixed point. In reality, this is not much shorter than what we already did: That alternative formula is known to be correct only after knowing that F is cts, so **1.4+** has to be proved. And the formula in **1.2+** is used more than a finite number of times. However, we can replace the arguments for both corollaries by the manipulation above. And in fact the proof of monotonicity in **1.1+** is unnecessary—but we already knew it was redundant, in case you hadn't noticed, since cts implies monotonic. Below, we shall write Section ∞+

in the most austere and economic style that we know, by employing these shortcuts.

But I think the most interesting thing here is the connection between that beautiful, simple formula for *lix* and the desire to show that operational and fixpoint semantics coincide. The book [**deBa**] avoids dealing with the connection between the operational and fixpoint semantics uniquely for this type of language with recursive commands (or even giving the operational semantics), and its author is a serious follower of Dana Scott. And I've never seen our **1.2+** before. So I may be shocked to later learn that this connection had gone unnoticed.

Section ∞- : Mutual Simple Recursion.

Following Harel [**H**], the adjective "simple" refers to the sections which are labeled "$-$"; and that means we study recursion with the restriction that free occurrences of $callX$ are forbidden in the body of a declared procedure if X is not one of the variables involved in the mutual recursion. [This is exactly the level of generality where a sound, Cook-complete Floyd-Hoare proof system is known to me to exist [**mutu**]. For the most general recursion as specified in Section $\infty+$ ahead (or even the non-mutual case of Section 1+), I know of none. Again, a reader who can convert my ignorance to knowledge would be much appreciated, am I in error on this, where I am referring to a Floyd-Hoare proof system for *both* partial *and total* correctness.]

So here we allow $\mathcal{P}cv$ to be general. Otherwise, the only change (from $\mathcal{R}ec_{1-}$ or $\mathcal{R}ec_{1+}$) to the syntax in getting $\mathcal{R}ec_{\infty-}$ is to

$$\text{replace } \nabla YC \text{ by } \nabla_j \vec{C}/\vec{Y} \ , \quad \text{which is short for } \quad \nabla_j C_1 Y_1 C_2 Y_2 \cdots C_k Y_k \ .$$

In the latter, we have distinct procedure variables $Y_i \in \mathcal{P}cv$, various commands C_i, and a choice of (j,k) with $1 \leq j \leq k$. **The one proviso, as indicated above, is that $\nabla_j \vec{C}/\vec{Y}$ is a legal member of $\mathcal{R}ec_{\infty-}$ only if no free occurrence of $callX$ exists in any of the commands C_i , if $X \neq Y_\ell$ for all ℓ.**

Intuitively, the command $\nabla_j \vec{C}/\vec{Y}$ is intended to convey what most of you will roughly understand by the following program from somebody somewhere's recursive Algol fragment:

$$\text{begin } \ declare\ Y_1 \ to\ be\ C_1, \ \cdots \ , declare\ Y_k \ to\ be\ C_k; \ do\ Y_j \ \text{ end } .$$

All the precise syntactic discussion of this recursive command etc., including the *call*, assignment, sequencing, and *ite* commands as well, should be given in a single giant structural inductive definition which also contains the definition of this freeness:

The free occurrences of $callX$ in a command are:
none for assignment commands, or for $callY$ when $Y \neq X$;
the whole command for $callX$;
those in C and/or in D for $(C;D)$ and for $ite(H)(C)(D)$;
none for $\nabla_j \vec{C}/\vec{Y}$ if $X = Y_i$ for some i;

and finally (a case which won't happen here),
those in the various C_i for $\nabla_j \vec{C}/\vec{Y}$ if $X \neq Y_i$ for all i .

In a sense, this section is merely a 'vectorial' version of a mixture of the last two sections. No extra subtlety is involved other than making the fixed point definition over a k-fold product of copies of $\mathcal{B}sm$. The 'fixpoint' definition of the semantics of the recursive command, and the semantics of the other commands, both take the general form exactly as in the previous section, i.e.

$$\mathcal{M} : \mathcal{R}ec_{\infty-} \times \mathcal{B}sm^{\mathcal{P}cv} \ \rightarrow \ \mathcal{B}sm \ ,$$

given below by induction on commands. Then we can define the basic semantics of commands

$$m : \mathcal{R}ec_{\infty-} \rightarrow \mathcal{B}sm \qquad \text{by} \qquad m(C) := \mathcal{M}(C, \Phi) \ ,$$

where Φ maps all of $\mathcal{P}cv$ to $\emptyset$, the empty set. (Effectively, the 'real' semantics of the atomic *call* commands is to universally diverge.)

Suppose given $\vec{C}/\vec{Y}$ as above. Let $\vec{\beta} \in \mathcal{B}sm^k$.
As with our notation for states, define $\Theta_{\vec{Y}\mapsto\vec{\beta}}$ to be the element of $\mathcal{B}sm^{\mathcal{P}cv}$ which agrees everywhere with Θ, except that it maps each Y_i to β_i.
By $\vec{\mathcal{M}}(\vec{C}, \Theta) \in \mathcal{B}sm^k$ we mean the element whose ith component is $\mathcal{M}(C_i, \Theta)$.
Define $\overrightarrow{\nabla C/\ Y} \ \in \ (\mathcal{R}ec_{\infty-})^k$ to be the element whose jth component is $\nabla_j \vec{C}/\vec{Y}$.

Definition and Theorem $\infty.1-$**.** *By induction on C, the five clauses just below define $\mathcal{M}(C, \Theta)$ in such a way that, for each $(C, \Theta, \vec{Y})$, the map in the 5th clause, namely $[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{C}, \Theta_{\vec{Y}\mapsto\vec{\beta}}) \ : \ \mathcal{B}sm^k \rightarrow \mathcal{B}sm^k \ ]$, is monotonic with respect to the ordering, $\sqsubset$, discussed in the 2nd part of Prelude A.* (This justifies that 5th clause via $\mathbf{P1}^k$)

$\mathcal{M}(x \leftarrow\!\!\div t, \Theta) \ := \ \{(s, s_{x\mapsto s(t)}) \mid s \in \mathcal{S}te\}$ *(independent of Θ) ;*

$\mathcal{M}(callX, \Theta) \ := \ \Theta(X)$ *;*

$\mathcal{M}((C; D), \Theta) \ := \ \mathcal{M}(D, \Theta) \circ \mathcal{M}(C, \Theta)$

$\mathcal{M}(ite(H)(C)(D), \Theta) \ := \ \{(s, s') \mid H\mathbf{tt}@s$ and $(s, s') \in \mathcal{M}(C, \Theta)$; or
$H\mathbf{ff}@s$ and $(s, s') \in \mathcal{M}(D, \Theta)\}$;

$\vec{\mathcal{M}}(\overrightarrow{\nabla C/\ Y}, \Theta) \ := \ lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{C}, \Theta_{\vec{Y}\mapsto\vec{\beta}})] \ : \ \mathcal{B}sm^k \rightarrow \mathcal{B}sm^k \ ]$.

Proof of monotonicity. Just this once, let us write down textually explicit column vectors to formulate exactly what needs to be proved (but simply take $k = 3$ for purposes of illustration):

$$\begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} \sqsubset \begin{pmatrix} \beta'_1 \\ \beta'_2 \\ \beta'_3 \end{pmatrix} \quad \Longrightarrow \quad \begin{pmatrix} \mathcal{M}(C_1, \Theta_{\vec{Y}\mapsto\vec{\beta}}) \\ \mathcal{M}(C_2, \Theta_{\vec{Y}\mapsto\vec{\beta}}) \\ \mathcal{M}(C_3, \Theta_{\vec{Y}\mapsto\vec{\beta}}) \end{pmatrix} \sqsubset \begin{pmatrix} \mathcal{M}(C_1, \Theta_{\vec{Y}\mapsto\vec{\beta}'}) \\ \mathcal{M}(C_2, \Theta_{\vec{Y}\mapsto\vec{\beta}'}) \\ \mathcal{M}(C_3, \Theta_{\vec{Y}\mapsto\vec{\beta}'}) \end{pmatrix}$$

Now one can get from $\vec{\beta}$ to $\vec{\beta}'$ in various ways by 'changing one β_i at a time'. And the 'inequality' on the right-hand side of the implication is just a conjunction of set inclusions. So the whole thing is best proved by reducing and reformulating as follows :

If Θ and Ψ agree, except that $\Theta(Y) \subset \Psi(Y)$ for some one particular procedure variable Y, then $\mathcal{M}(C, \Theta) \subset \mathcal{M}(C, \Psi)$.

This is just a slightly more abstract formulation of the corresponding result, **1.1+**, in the last section. So, for all but the recursive command, the paragraphs just below simply repeat the argument there.

For $C = callY$, the conclusion is a duplicate of the hypothesis. And for C an assignment command or $callX$ with $X \neq Y$, the conclusion is equality. For the next two command constructors, assume that the result holds for $C = D$ and $C = E$. Then, using **P3**, we get, for $C = (D; E)$,

$$\mathcal{M}(C, \Theta) \;=\; \mathcal{M}(E, \Theta) \circ \mathcal{M}(D, \Theta) \;\subset\; \mathcal{M}(E, \Psi) \circ \mathcal{M}(D, \Psi) \;=\; \mathcal{M}(C, \Psi) \; .$$

For $C = ite(H)(D)(E)$, supposing that $(s, s') \in \mathcal{M}(ite(H)(D)(E), \Theta)$, we see that
if $H\mathbf{tt}@s$, then $(s, s') \in \mathcal{M}(D, \Theta) \subset \mathcal{M}(D, \Psi)$, so $(s, s') \in \mathcal{M}(ite(H)(D)(E), \Psi)$;
whereas
if $H\mathbf{ff}@s$, then $(s, s') \in \mathcal{M}(E, \Theta) \subset \mathcal{M}(E, \Psi)$, so $(s, s') \in \mathcal{M}(ite(H)(D)(E), \Psi)$.

Finally, for $C = \nabla_j \vec{C}/\vec{Y}$, we'll prove it for all j simultaneously.We need

$$lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{C}, \Theta_{\vec{Y}\mapsto\vec{\beta}})] \;\sqsubset\; lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{C}, \Psi_{\vec{Y}\mapsto\vec{\beta}})] \; .$$

If $Y = Y_i$ for some i, then $\Theta_{\vec{Y}\mapsto\vec{\beta}} = \Psi_{\vec{Y}\mapsto\vec{\beta}}$, so that one is pretty easy.

If $Y \neq Y_i$ for all i, then $\Theta_{\vec{Y}\mapsto\vec{\beta}}$ and $\Psi_{\vec{Y}\mapsto\vec{\beta}}$ agree on all procedure variables except Y, where the 'correct' inclusion holds, and so we can apply the inductive hypothesis to each C_j with those two elements of $\mathcal{B}sm^{\mathcal{P}cv}$, i.e. we have

$$\mathcal{M}(C_j, \Theta_{\vec{Y}\mapsto\vec{\beta}}) \subset \mathcal{M}(C_j, \Psi_{\vec{Y}\mapsto\vec{\beta}}) \; .$$

But now we need only observe that for monotonic functions F and G from $\mathcal{B}sm^k$ to itself, if $F(\vec{\beta}) \sqsubset G(\vec{\beta})$ for all $\vec{\beta}$, then $lix(F) \sqsubset lix(G)$:

$$lix(F) \;=\; \sqcap \{\, \vec{\alpha} \mid F(\vec{\alpha}) \sqsubset \vec{\alpha} \,\} \;\sqsubset\; \sqcap \{\, \vec{\alpha} \mid G(\vec{\alpha}) \sqsubset \vec{\alpha} \,\} \;=\; lix(G) \;.$$

We are applying this with

$$F(\vec{\beta}) := \vec{\mathcal{M}}(\vec{C}, \Theta_{\vec{Y} \mapsto \vec{\beta}}) \qquad ; \qquad G(\vec{\beta}) := \vec{\mathcal{M}}(\vec{C}, \Psi_{\vec{Y} \mapsto \vec{\beta}}) \;.$$

Definition. For any C and $\vec{D}/\vec{Y}$, define by induction on C a command $C_{\vec{D}/\vec{Y}}$—(AKA, in our inimitable substitution notation, $|C|^{all\ free\ call\not{Y}_i \to D_i}$) :

$$(x \leftarrow\!\!\div t)_{\vec{D}/\vec{Y}} := x \leftarrow\!\!\div t \;\; ; \;\; (call Y_i)_{\vec{D}/\vec{Y}} := D_i \;\; ; \;\; (call X)_{\vec{D}/\vec{Y}} := call X \text{ if } X \neq Y_i \text{ for all } i \;;$$

$$(C'; C'')_{\vec{D}/\vec{Y}} := (C'_{\vec{D}/\vec{Y}}; C''_{\vec{D}/\vec{Y}}) \;\; ; \;\; ite(H)(C')(C'')_{\vec{D}/\vec{Y}} := ite(H)(C'_{\vec{D}/\vec{Y}})(C''_{\vec{D}/\vec{Y}}) \;\; ;$$

$$(\nabla_j \vec{E}/\vec{Z})_{\vec{D}/\vec{Y}} \;:=\; \nabla_j \vec{E}/\vec{Z} \;\; .$$

The last piece of the definition is simple because we have excluded the possibility of there existing a Y_i different from all the Z_ℓ for which a free $call Y_i$ occurs in some E_m. Intuitively, the construction $C_{\vec{D}/\vec{Y}}$ removes each free occurrence of any $call Y_i$ in C and replaces it with D_i.

Proposition $\infty.2-$**.** *For any C and $\vec{D}/\vec{Y}$, and any $\Theta \in \mathcal{B}sm^{\mathcal{P}cv}$, we have*

$$\mathcal{M}(C_{\vec{D}/\vec{Y}}, \Theta) \;=\; \mathcal{M}(C, \Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)}) \;.$$

[This gives a utility to defining $\mathcal{M}(-, \Theta)$ for Θ other than Φ, quite apart from its utility to formulating fixpoint definitions.]

Proof by induction on C**.** When $C = call Y_i$, both sides are $\mathcal{M}(D_i, \Theta)$.
When $C = call X$ for X differing from all Y_i, both sides are $\Theta(X)$.
When C is an assignment command, both sides are the usual.
When $C = (C'; C'')$, we have

$$\mathcal{M}((C'; C'')_{\vec{D}/\vec{Y}}, \Theta) \;=\; \mathcal{M}((C'_{\vec{D}/\vec{Y}}; C''_{\vec{D}/\vec{Y}}), \Theta) \;=\; \mathcal{M}(C''_{\vec{D}/\vec{Y}}, \Theta) \circ \mathcal{M}(C'_{\vec{D}/\vec{Y}}, \Theta) \;=$$

$$\mathcal{M}(C'', \Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)}) \circ \mathcal{M}(C', \Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)}) \;=\; \mathcal{M}((C'; C''), \Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)}) \;.$$

For $C = ite(H)(C')(C'')$,

$$\mathcal{M}(ite(H)(C')(C'')_{\vec{D}/\vec{Y}}, \Theta) \;=\; \mathcal{M}(ite(H)(C'_{\vec{D}/\vec{Y}})(C''_{\vec{D}/\vec{Y}}), \Theta) \;=$$

$$\{(s,s') \mid H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C'_{\vec{D}/\vec{Y}}, \Theta);\ \text{or } H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(C''_{\vec{D}/\vec{Y}}, \Theta)\} \;=$$

$$\{(s,s') \mid H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C', \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)});\ \text{or } H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(C'', \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)})\} \;=$$

$$\mathcal{M}(ite(H)(C')(C''), \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)}) \,.$$

When $\vec{C} = \overrightarrow{\nabla E/Z}$, we have

$$\vec{\mathcal{M}}(\vec{C}_{\vec{D}/\vec{Y}}, \Theta) \;=\; \vec{\mathcal{M}}(\vec{C}, \Theta) \;=\; lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, \Theta_{\vec{Z}\mapsto\vec{\beta}})] \,,$$

whereas

$$\vec{\mathcal{M}}(\vec{C}, \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)}) \;=\; lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)})_{\vec{Z}\mapsto\vec{\beta}})] \,.$$

By letting $\vec{X}$ be a list of those Y_i which do not occur in the list $\vec{Z}$, and letting $\vec{\gamma}$ be the list of those $\mathcal{M}(D_\ell, \Theta)$ corresponding to the Y_ℓ which remain as X_j's, we can rewrite the last right-hand side as

$$lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{X}\mapsto\vec{\gamma}})_{\vec{Z}\mapsto\vec{\beta}})] \;=\; lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Z}\mapsto\vec{\beta}})_{\vec{X}\mapsto\vec{\gamma}})] \,.$$

The result is then immediate, taking $\Psi \;:=\; \Theta_{\vec{Z}\mapsto\vec{\beta}}$ from the next lemma.

Lemma. *If a command E has no free callX_i then, for any Ψ and $\vec{\gamma}$,*

$$\mathcal{M}(E, \Psi_{\vec{X}\mapsto\vec{\gamma}}) \;=\; \mathcal{M}(E, \Psi) \,.$$

The proof is by induction on E. The cases of atomic commands are easy as usual, and of sequencing and '*ite*ing' are the expected manipulations. When $E = \nabla_j \vec{F}/\vec{V}$, the list of right-hand sides for all the different j is

$$lix[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{F}, \Psi_{\vec{V}\mapsto\vec{\beta}})\ ] \,,$$

whereas the the list of left-hand sides is

$$lix[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{F}, (\Psi_{\vec{X}\mapsto\vec{\gamma}})_{\vec{V}\mapsto\vec{\beta}})\ ] \,,$$

As above, we can replace $\vec{X}$ by a 'subvector', containing only those X_i not in $\vec{V}$; and then commute the subscripts on Ψ. But now the inductive hypothesis applies to each F_ℓ, so the proof is done.

Definition of "Unfolding". Define, by induction on $N \geq 0$, the command $(\vec{C}/\vec{Y})_j^{<N>}$, simultaneously letting $\overrightarrow{C/Y}^{<N>}$ be the 'vector' whose jth component is $(\vec{C}/\vec{Y})_j^{<N>}$, as follows :

$$(\vec{C}/\vec{Y})_j^{<0>} \;:=\; callY_j \quad , \quad (\vec{C}/\vec{Y})_j^{<N+1>} \;:=\; (C_j)_{\overrightarrow{(C/Y}^{<N>}\,)/\vec{Y}} \; .$$

Corollary $\infty.3-$**.** *For any $\vec{C}/\vec{Y}$, we have* (with $\vec{m}$ analogous to $\vec{\mathcal{M}}$),

$$\bigsqcup_{N\geq 0} \vec{m}(\overrightarrow{C/Y}^{<N>}) \;\sqsubset\; \vec{m}(\overrightarrow{\nabla C/\, Y}) \; .$$

Proof. Proceed by induction on N to show $\vec{m}(\overrightarrow{C/Y}^{<N>}) \;\sqsubset\; \vec{m}(\overrightarrow{\nabla C/\, Y})$:

$$\vec{m}(\overrightarrow{C/Y}^{<0>}) \;=\; \vec{\mathcal{M}}([callY_1, \cdots, callY_k], \Phi) \;=\; [\mathcal{M}(callY_1, \Phi), \cdots, \mathcal{M}(callY_k, \Phi)]$$

$$= \; [\Phi(Y_1), \cdots, \Phi(Y_k)] \;=\; [\emptyset, \cdots, \emptyset] \;\sqsubset \vec{m}(\overrightarrow{\nabla C/\, Y}) \; .$$

And for the inductive step,

$$\vec{m}(\overrightarrow{C/Y}^{<N+1>}) \;=\; \vec{\mathcal{M}}(\overrightarrow{C/Y}^{<N+1>}, \Phi) \;=\; \vec{\mathcal{M}}(\vec{C}_{\overrightarrow{(C/Y}^{<N>}\,)/\vec{Y}}, \Phi) \;=$$

$$\vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\mathcal{M}}(\overrightarrow{C/Y}^{<N>},\Phi)}) \;\sqsubset\; \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\mathcal{M}}(\overrightarrow{\nabla C/\, Y},\Phi)})$$

$$= \; \vec{\mathcal{M}}(\overrightarrow{\nabla C/\, Y}, \Phi) \;=\; \vec{m}(\overrightarrow{\nabla C/\, Y}) \; .$$

The 'inequality' uses the inductive hypothesis and monotonicity. The equality before it uses $\infty.2-$, and the one after it is the fact that $\vec{\mathcal{M}}(\overrightarrow{\nabla C/\, Y}, \Phi)$ is a fixed point of

$$[\; \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\beta}}) \;:\; \mathcal{B}sm^k \to \mathcal{B}sm^k \;] \; .$$

Proposition $\infty.4-$**.** *If* $\overrightarrow{\alpha_0} \sqsubset \overrightarrow{\alpha_1} \sqsubset \overrightarrow{\alpha_2} \sqsubset \cdots$ *is a chain in* $\mathcal{B}sm^k$*, then, for any* C*, we have*

$$\mathcal{M}(C, \Theta_{\vec{Y} \mapsto \bigsqcup_{N\geq 0} \overrightarrow{\alpha_N}}) \;=\; \bigcup_{N\geq 0} \mathcal{M}(C, \Theta_{\vec{Y} \mapsto \overrightarrow{\alpha_N}}) \;.$$

Equivalently, [$\vec{\alpha} \mapsto \mathcal{M}(C, \Theta_{\vec{Y} \mapsto \vec{\alpha}})$] *is cts.* (in a slightly more general sense than earlier defined, since the domain and range are different ordered sets, except when $k = 1$).

Proof by induction on C**.** The result for assignment commands takes the form of asserting that a union of copies of a fixed set is just that set. When $C = callX$ where $X \neq Y_\ell$ for any ℓ, the same is true, the set being $\Theta(X)$. When $C = callY_\ell$, the assertion says that

$$(\bigsqcup_{N\geq 0} \overrightarrow{\alpha_N})_\ell \;=\; \bigcup_{N\geq 0} (\overrightarrow{\alpha_N})_\ell \;,$$

which is merely the definition of $\bigsqcup_N$. When $C = (C'; C'')$, we have

$$\mathcal{M}((C'; C''), \Theta_{\vec{Y} \mapsto \bigsqcup_{N\geq 0} \vec{\alpha}_N}) \;=\; \mathcal{M}(C'', \Theta_{\vec{Y} \mapsto \bigsqcup_{N\geq 0} \vec{\alpha}_N}) \circ \mathcal{M}(C', \Theta_{\vec{Y} \mapsto \bigsqcup_{N\geq 0} \vec{\alpha}_N}) \;=$$

$$\bigcup_{N\geq 0} \mathcal{M}(C'', \Theta_{\vec{Y} \mapsto \vec{\alpha}_N}) \circ \bigcup_{N\geq 0} \mathcal{M}(C', \Theta_{\vec{Y} \mapsto \vec{\alpha}_N}) \;=$$

$$\bigcup_{N\geq 0} (\mathcal{M}(C'', \Theta_{\vec{Y} \mapsto \vec{\alpha}_N}) \circ \mathcal{M}(C', \Theta_{Y \mapsto \vec{\alpha}_N})) \;=\; \bigcup_{N\geq 0} \mathcal{M}((C'; C''), \Theta_{\vec{Y} \mapsto \vec{\alpha}_N}) \;,$$

the penultimate equality using **P4**.

For $C = ite(H)(C')(C'')$,

$$(s, s') \in \mathcal{M}(ite(H)(C')(C''), \Theta_{Y \mapsto \bigsqcup_{N\geq 0} \vec{\alpha}_N}) \iff$$

$$[H\mathbf{tt}@s \text{ and } (s, s') \in \mathcal{M}(C', \Theta_{Y \mapsto \bigsqcup_{N\geq 0} \vec{\alpha}_N}) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s, s') \in \mathcal{M}(C'', \Theta_{Y \mapsto \bigsqcup_{N\geq 0} \vec{\alpha}_N})] \iff$$

$$[H\mathbf{tt}@s \text{ and } (s, s') \in \bigcup_{N\geq 0} \mathcal{M}(C', \Theta_{Y \mapsto \vec{\alpha}_N}) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s, s') \in \bigcup_{N\geq 0} \mathcal{M}(C'', \Theta_{Y \mapsto \vec{\alpha}_N})] \iff$$

$$\exists N\, [H\mathbf{tt}@s \text{ and } (s, s') \in \mathcal{M}(C', \Theta_{Y \mapsto \vec{\alpha}_N}) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s, s') \in \mathcal{M}(C'', \Theta_{Y \mapsto \vec{\alpha}_N})] \iff$$

$$\exists N\, [(s, s') \in \mathcal{M}(ite(H)(C')(C''), \Theta_{Y \mapsto \vec{\alpha}_N})] \iff (s, s') \in \bigcup_{N\geq 0} \mathcal{M}(ite(H)(C')(C''), \Theta_{Y \mapsto \vec{\alpha}_N}) \;.$$

When $\vec{C} = \overrightarrow{\nabla E/Z}$, we must prove that

$$\vec{\mathcal{M}}(\overrightarrow{\nabla E/Z}, \Theta_{\vec{Y}\mapsto\bigsqcup_{N\geq 0}\overrightarrow{\alpha_N}}) \;=\; \bigsqcup_{N\geq 0} \vec{\mathcal{M}}(\overrightarrow{\nabla E/Z}, \Theta_{\vec{Y}\mapsto\overrightarrow{\alpha_N}}) \;.$$

The right-hand side is

$$\bigsqcup_{N\geq 0} lix[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Y}\mapsto\vec{\alpha}_N})_{\vec{Z}\mapsto\vec{\beta}}\ ].$$

The left-hand side is

$$lix[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Y}\mapsto\bigsqcup_N \vec{\alpha}_N})_{\vec{Z}\mapsto\vec{\beta}}\ ].$$

Once again, we may eliminate any of the Y_i which happen to be in $\vec{Z}$; i.e. let $\vec{X}$ be a list containing exactly the Y_i which are not in $\vec{Z}$. Also, let $\mathsf{new}k$ be the reduced value of k which is the length of the vector $\vec{X}$. And also form, for each $\vec{\alpha}_N$, that $\vec{\gamma}_N \in \mathcal{P}cv^{\mathsf{new}k}$ which agrees with $\vec{\alpha}_N$ in each slot where we have retained that particular procedure variable in $\vec{Y}$ when forming $\vec{X}$. So just change $\vec{Y}$ to $\vec{X}$, and $\vec{\alpha}_N$ to $\vec{\gamma}_N$, at the unique place each occurs in each display. Now we can also interchange the order of the subscripts to Θ in both displays. So we are left to prove that

$$lix[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Z}\mapsto\vec{\beta}})_{\vec{X}\mapsto\bigsqcup_N \vec{\gamma}_N})\ ] \;=\; \bigsqcup_{N\geq 0} lix[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Z}\mapsto\vec{\beta}})_{\vec{X}\mapsto\vec{\gamma}_N})\ ].$$

But, by the induction hypothesis applied to $\vec{E}$, the left-hand side is

$$lix[\ \vec{\beta} \mapsto \bigsqcup_N \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Z}\mapsto\vec{\beta}})_{\vec{X}\mapsto\vec{\gamma}_N})\ ]\ ,$$

which agrees with the right-hand side, by $\mathbf{P2}^{\mathsf{new}k}$, since the function

$$[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Z}\mapsto\vec{\beta}})_{\vec{X}\mapsto\vec{\gamma}_N})\ ]$$

is cts for every N by the inductive hypothesis, and since the Nth such function is $\sqsubset$-dominated by the one for $N+1$, using monotonicity.

Corollary $\infty.5-$**.** *For any* $\vec{C}/\vec{Y}$*, we have*

$$\vec{m}(\overrightarrow{\nabla C/\ Y}) \ \sqsubset \ \bigsqcup_{N\geq 0} \vec{m}(\overrightarrow{C/Y}^{<N>}) \ .$$

Proof. Using $\infty.2-$ and $\infty.4-$ for the 3rd and 2nd equalities respectively,

$$\vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto \bigsqcup_{N\geq 0}\vec{m}(\overrightarrow{C/Y}^{<N>})}) \ = \ \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto \bigsqcup_{N\geq 0}\vec{\mathcal{M}}(\overrightarrow{C/Y}^{<N>},\Phi)}) \ =$$

$$\bigsqcup_{N\geq 0} \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto \vec{\mathcal{M}}(\overrightarrow{C/Y}^{<N>},\Phi)}) \ = \ \bigsqcup_{N\geq 0} \vec{\mathcal{M}}(\vec{C}_{(\overrightarrow{C/Y}^{<N>})/\vec{Y}}, \Phi) \ =$$

$$\bigsqcup_{N\geq 0} \vec{\mathcal{M}}(\overrightarrow{C/Y}^{<N+1>}, \Phi) \ = \ \bigsqcup_{N\geq 0} \vec{\mathcal{M}}(\overrightarrow{C/Y}^{<N>}, \Phi) \ = \ \bigsqcup_{N\geq 0} \vec{m}(\overrightarrow{C/Y}^{<N>}) \ .$$

For the penultimate equality, we've just added one vector, consisting of nothing but empty sets, to the 'union'. Thus, the 'vector of sets' $\bigsqcup_{N\geq 0}\vec{m}(\overrightarrow{C/Y}^{<N>})$ is a fixed point of

$$[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\beta}}) \ : \ \mathcal{B}sm^k \to \mathcal{B}sm^k \ ] \ ,$$

and so must contain the least such fixed point, as required.

Theorem $\infty.6-$**.** *We have*

$$\vec{m}(\overrightarrow{\nabla C/\ Y}) \ = \ \bigsqcup_{N\geq 0} \vec{m}(\overrightarrow{C/Y}^{<N>}) \ .$$

This is immediate from the two corollaries.

Afterthought $\infty.7-$**.** One has the unsurprising result that $\mathcal{R}ec_{\infty-}$ with its semantics is a *deterministic* command language. That is, all $m(C)$ are graphs of (partial) functions—i.e. of functions from a subset of $\mathcal{S}te$ to $\mathcal{S}te$. One proves inductively on C that, if $\Theta(X)$ is a graph for all $X \in \mathcal{P}cv$, then $\mathcal{M}(C, \Theta)$ is also a graph. And then take $\Theta = \Phi$.

Here I do happen to know how to convert the previous theorem into an 'unfolding operational' definition. This proved useful in [**mutu**], when setting up a sound, Cook-complete Floyd-Hoare proof system for this command language and its semantics, for both partial and total correctness, when the

1st-order language on which all this is based is taken to be that of 1st-order number theory, and the interpretation is taken to be the natural numbers. So one can give another proof of deterministicism here, using the operational semantics whose only divergence from the fixpoint semantics is to use the right-hand side in $\infty.6-$ as the definition of its left-hand side, rather than using lix.

Is this really slicker?

Given $\vec{C}/\vec{Y}$, let $\vec{F}(\vec{\beta}) = \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\beta}})$, recall that $\vec{\phi}$ is the k-tuple of nothing but $\emptyset$'s, and calculate as follows.

$$\vec{m}(\overrightarrow{\nabla C/Y}) \;=\; \vec{\mathcal{M}}(\overrightarrow{\nabla C/Y}, \Phi) \;=\; lix(\vec{F})$$

$$=\; \bigsqcup_{n=0}^{\infty} \vec{F}^n(\vec{\phi}) \;=\; \vec{\phi} \;\sqcup\; \vec{F}(\vec{\phi}) \;\sqcup\; \vec{F}^2(\vec{\phi}) \;\sqcup\; \vec{F}^3(\vec{\phi}) \;\sqcup\; \cdots$$

$$=\; \vec{\mathcal{M}}(C, \Phi_{\vec{Y}\mapsto\vec{\phi}}) \;\sqcup\; \vec{\mathcal{M}}(C, \Phi_{\vec{Y}\mapsto\vec{F}(\vec{\phi})}) \;\sqcup\; \vec{\mathcal{M}}(C, \Phi_{\vec{Y}\mapsto\vec{F}(\vec{F}(\vec{\phi}))}) \;\sqcup\; \cdots$$

$$=\; \vec{\mathcal{M}}(\vec{C}, \Phi) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{C},\Phi)}) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{C},\Phi_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{C},\Phi)})}) \;\sqcup\; \cdots$$

$$=\; \vec{m}(\vec{C}) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}_{\vec{C}/\vec{Y}}, \Phi) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\mathcal{M}(C_{\vec{C}/\vec{Y}},\Phi)}) \;\sqcup\; \cdots$$

$$=\; \vec{m}(\vec{C}) \;\sqcup\; \vec{\mathcal{M}}(\overrightarrow{C/Y}^{<2>}, \Phi) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\mathcal{M}}(\overrightarrow{C/Y}^{<2>},\Phi)}) \;\sqcup\; \cdots$$

$$=\; \vec{m}(\vec{C}) \;\sqcup\; \vec{m}(\overrightarrow{C/Y}^{<2>}) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}_{\overrightarrow{(C/Y}^{<2>})/\vec{Y}}, \Phi) \;\sqcup\; \cdots$$

$$=\; \vec{m}(\vec{C}) \;\sqcup\; \vec{m}(\overrightarrow{C/Y}^{<2>}) \;\sqcup\; \vec{\mathcal{M}}(\overrightarrow{C/Y}^{<3>}, \Phi) \;\sqcup\; \cdots$$

$$=\; \vec{m}(\overrightarrow{C/Y}^{<1>}) \sqcup \vec{m}(\overrightarrow{C/Y}^{<2>}) \sqcup \vec{m}(\overrightarrow{C/Y}^{<3>}) \sqcup \cdots \;=\; \bigsqcup_{N} \vec{m}(\overrightarrow{C/Y}^{<N>}) \,.$$

The 3rd equality uses the alternative formula $(***)$ for the fixed point. In reality, this is not much shorter than what we already did: That alternative formula is known to be correct only after knowing that $\vec{F}$ is cts, so $\infty.4-$ has to be proved. And the formula in $\infty.2-$ is used more than a finite number of times. However, we can replace the arguments for both corollaries by the manipulation above. And in fact the proof of monotonicity in $\infty.1-$ is unnecessary—but we already knew it was redundant, in case you hadn't

noticed, since cts implies monotonic. Below, we shall write Section $\infty+$ in the most austere and economic style that we know, by employing these shortcuts.

But I think the most interesting thing here is the connection between that beautiful, simple formula for *lix* and the desire to show that operational and fixpoint semantics coincide. The book [**deBa**] avoids dealing with the connection between the operational and fixpoint semantics uniquely for this type of language with recursive commands (or even giving the operational semantics), and its author is a serious follower of Dana Scott. And I've never seen our $\infty.2-$ before. So I may be shocked to later learn that this connection had gone unnoticed.

Section $\infty+$: Mutual General Recursion.

Since we've now gone from "simple" to "general", here we merely drop the restriction from the previous section that free occurrences of $callX$ are forbidden in the body of a declared procedure, if X is not one of the variables involved in the mutual recursion.

Part A—Paralleling the last three sections.

The changes needed from the previous section are essentially details concerning substitution and the last part of the proof of $\infty.2+$, which is more complicated than that of $\infty.2-$. The extra stuff needed here is the mutual recursion version of the matters discussed in the **Diversion** in Section 1+.

The reader must be getting fed up with all this pseudo-repetition, so here let's just indicate the changes needed from the previous section to take account of dropping that restriction.

Definition. For any command C and any $\vec{D}/\vec{Y}$, define, by induction on the number of symbols in the string C, a command $C_{\vec{D}/\vec{Y}}$. This is done exactly as was done in the previous section for all but ∇-commands. For them, use a 'vectorial version' of the corresponding definition two sections back. We will use the convention

$$\vec{B}_{\vec{D}/\vec{Y}} \ := \ (B_{1\vec{D}/\vec{Y}}, B_{2\vec{D}/\vec{Y}}, \cdots) \ .$$

Given $\vec{E}/\vec{Z}$, vectors of length ℓ, and $\vec{D}/\vec{Y}$, vectors of length k, define

$$\overrightarrow{\nabla E/Z}_{\vec{D}/\vec{Y}} \ := \ (\ \nabla_1 \vec{E}'_{\vec{D}/\vec{Y}}/\vec{Z}' \ , \ \nabla_2 \vec{E}'_{\vec{D}/\vec{Y}}/\vec{Z}' \ , \ \cdots, \nabla_\ell \vec{E}'_{\vec{D}/\vec{Y}}/\vec{Z}' \ ) \ ,$$

where $\vec{Z}'$ is chosen to be the first procedure variable vector (temporarily give $\mathcal{P}cv^\ell$ a linear order for this purpose only) such that

$$\vec{Z}' \cap \vec{Z} = \emptyset \ ; \ \vec{Z}' \cap \vec{Y} = \emptyset \ ; \ \vec{Z}' \notin \vec{D} \cup \vec{E} \ \text{(meaning that no symbol } Z'_i \text{ occurs)} \ ,$$

and where $\vec{E}' \ := \ \vec{E}_{call\vec{Z}'/\vec{Z}}$, commands whose lengths are the same as those in $\vec{E}$, less than those in $\overrightarrow{\nabla E/Z}$, so the definition makes sense inductively.

Identical to the previous section, we have

Proposition $\infty.2+$. *For any C and $\vec{D}/\vec{Y}$, and any $\Theta \in \mathcal{B}sm^{\mathcal{P}cv}$, we have*

$$\mathcal{M}(C_{\vec{D}/\vec{Y}}, \Theta) \ = \ \mathcal{M}(C, \Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)}) \ .$$

The proofs of all but the ∇-command case are identical those in the previous section, though here we proceed by induction on the number of symbols in C.

When $C = \nabla_i \vec{E}/\vec{Z}$, we do it 'vectorially', i.e. for all i simultaneously. We have

$$\vec{\mathcal{M}}(\overrightarrow{\nabla E/Z}_{\vec{D}/\vec{Y}}, \Theta) \;=\; \vec{\mathcal{M}}(\overrightarrow{\nabla E'_{\vec{D}/\vec{Y}}/Z'}, \Theta) \qquad \text{definition of } \overrightarrow{\nabla E/Z}_{\vec{D}/\vec{Y}}$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}'_{\vec{D}/\vec{Y}}, \Theta_{\vec{Z}' \mapsto \vec{\beta}})] \qquad \text{definition of } \nabla\text{−command semantics}$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}', (\Theta_{\vec{Z}' \mapsto \vec{\beta}})_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta_{\vec{Z}' \mapsto \vec{\beta}})})] \qquad \text{inductive hypothesis applied to all } E'_i$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}', \Psi)] \text{ where } \Psi := (\Theta_{\vec{Z}' \mapsto \vec{\beta}})_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)} \quad \text{lemma below and since } \vec{Z}' \notin \vec{D}$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, \Psi_{\vec{Z} \mapsto \vec{\mathcal{M}}(call \vec{Z}', \Psi)})] \quad \text{defn. of } \vec{E}' \text{ and inductive hypothesis for the } E_i$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, \Psi_{\vec{Z} \mapsto \Psi(\vec{Z}')})] \qquad \text{semantics of } call\text{−commands}$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, \Psi_{\vec{Z} \mapsto \vec{\beta}})] \qquad (\vec{Z}' \mapsto \vec{\beta})\text{−part of the definition of } \Psi$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, ((\Theta_{\vec{Z}' \mapsto \vec{\beta}})_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)})_{\vec{Z} \mapsto \vec{\beta}})] \qquad \text{overall definition of } \Psi$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, ((\Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)})_{\vec{Z} \mapsto \vec{\beta}})_{\vec{Z}' \mapsto \vec{\beta}})] \qquad \text{because } \vec{Z}' \text{ disjoint from } \vec{Z} \text{ and } \vec{Y}$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)})_{\vec{Z} \mapsto \vec{\beta}})] \qquad \text{lemma below and since } \vec{Z}' \notin \vec{E}$$

$$= \vec{\mathcal{M}}(\overrightarrow{\nabla E/Z}, \Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)}) \qquad \text{definition of } \nabla\text{−command semantics}$$

as required.

(In Section 1+, we split this proof into two cases. But that was unnecessary, only perhaps conceptually simpler. And here, no such division makes sense.)

Lemma. *If $\vec{E}$ has no free $call\vec{Z}$ then, for any Ψ and $\vec{\gamma}$,*

$$\vec{\mathcal{M}}(\vec{E}, \Psi_{\vec{Z} \mapsto \vec{\gamma}}) \;=\; \vec{\mathcal{M}}(\vec{E}, \Psi) \;.$$

This is a slightly more 'vectorial' statement of the lemma in the previous section, and the proof is the same.

Part B—Ruthlessly Efficient Version of the most General Case (more in research literature style).

Here we give our most efficient version of recursion semantics, if not the most understandable. We'll refer back only to Preludes A and B. You can erase everything from the beginning of Section 1− ; it is all quickly derivable as special cases of the following seven pages. To save space we

(i) prove 'ctsness' right away, since it can be done and monotony follows;

(ii) use the manipulation at the end of the previous three sections (more precisely, the last one) to avoid the analogues of the proofs of the corollaries from the previous sections.

The language $\mathcal{R}ec_{\infty+}$ is given by a structural induction as

$$x \Leftarrow t \mid callY \;\; || \;\; (C;D) \mid ite(H)(C)(D) \mid \nabla_j C_1 Y_1 C_2 Y_2 \cdots C_k Y_k$$

The names C and D are names for commands (the strings which are the elements of that language); x is any variable in $1^{\underline{\text{st}}}$order; t is any $1^{\underline{\text{st}}}$order term; H is any quantifier-free $1^{\underline{\text{st}}}$order formula; and Y is a procedure variable, an element of $\mathcal{P}cv$, which is assumed to be an infinite set. In the last command, we have distinct procedure variables $Y_i \in \mathcal{P}cv$, various commands C_i, and a choice of (j,k) with $1 \leq j \leq k$. The last command is abbreviated to $\nabla_j \vec{C}/\vec{Y}$.

For the logical/mathematical/CSish athlete who skipped to here from the preludes, it may be worth mentioning that the command $\nabla_j \vec{C}/\vec{Y}$ is intended to convey a sort of recursive Algol fragment:

$$\textsf{begin} \;\; declare\ Y_1\ to\ be\ C_1, \;\; \cdots \;, declare\ Y_k\ to\ be\ C_k; \;\; do\ Y_j \;\; \textsf{end}\ .$$

Suppose given $\vec{C}/\vec{Y}$ as above. Let $\vec{\beta} \in \mathcal{B}sm^k$.
As with our notation for states, define $\Theta_{\vec{Y} \mapsto \vec{\beta}}$ to be the element of $\mathcal{B}sm^{\mathcal{P}cv}$ which agrees everywhere with Θ, except that it maps each Y_i to β_i.
By $\vec{\mathcal{M}}(\vec{C}, \Theta) \in \mathcal{B}sm^k$ we mean the element whose ith component is $\mathcal{M}(C_i, \Theta)$, as defined in the theorem below.
Define $\overrightarrow{\nabla C/\ Y} \in (\mathcal{R}ec_{\infty+})^k$ to be the element whose jth component is $\nabla_j \vec{C}/\vec{Y}$.

Definition and Theorem 1. *By induction on C, the five clauses just below define $\mathcal{M}(C,\Theta) \in \mathcal{B}sm$ in such a way that, for each $(C,\Theta,\vec{Y})$, the map in the 5th clause, namely $[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{C},\Theta_{\vec{Y}\mapsto\vec{\beta}}) \ : \ \mathcal{B}sm^k \to \mathcal{B}sm^k\ ]$, is cts, and therefore monotonic, with respect to the ordering, $\sqsubset$, discussed in the 2nd part of Prelude A.* (This justifies that 5th clause via $\mathbf{P1}^k$)

$\mathcal{M}(x \div\!\!= t,\Theta) \ := \ \{(s, s_{x\mapsto s(t)}) \mid s \in \mathcal{S}te\}$ *(independent of Θ) .*

$\mathcal{M}(callX,\Theta) \ := \ \Theta(X)$.

$\mathcal{M}((C;D),\Theta) \ := \ \mathcal{M}(D,\Theta) \circ \mathcal{M}(C,\Theta)$.

$\mathcal{M}(ite(H)(C)(D),\Theta) \ := \ \{(s,s') \mid H\mathbf{tt}@s$ and $(s,s') \in \mathcal{M}(C,\Theta)$; or
$H\mathbf{ff}@s$ and $(s,s') \in \mathcal{M}(D,\Theta)\}$.

$\vec{\mathcal{M}}(\overrightarrow{\nabla C/\ Y},\Theta) \ := \ lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{C},\Theta_{\vec{Y}\mapsto\vec{\beta}})] \ : \ \mathcal{B}sm^k \to \mathcal{B}sm^k\ ]$.

Proof by induction on C. Except for the last case, we prove it componentwise, as indicated in the display just below.

Let $\{\overrightarrow{\alpha_N}\}$ be a chain of elements in $\mathcal{B}sm^k$, as in the definition of *cts*. When $C = callY_\ell$, the assertion to be proved, namely

$$\mathcal{M}(C,\Theta_{\vec{Y}\mapsto\bigsqcup_{N\geq 0}\overrightarrow{\alpha_N}}) \ = \ \bigcup_{N\geq 0} \mathcal{M}(C,\Theta_{\vec{Y}\mapsto\overrightarrow{\alpha_N}}) \ .$$

says that

$$(\bigsqcup_{N\geq 0} \overrightarrow{\alpha_N})_\ell \ = \ \bigcup_{N\geq 0} (\overrightarrow{\alpha_N})_\ell \ ,$$

which is merely the definition of $\bigsqcup_N$. The result for assignment commands takes the form of asserting that a union of copies of a fixed set is just that set. When $C = callX$ where $X \neq Y_\ell$ for any ℓ, the same is true, the set being $\Theta(X)$.

When $C = (C';C'')$, we have

$$\mathcal{M}((C';C''),\Theta_{\vec{Y}\mapsto\bigsqcup_{N\geq 0}\vec{\alpha}_N}) \ = \ \mathcal{M}(C'',\Theta_{\vec{Y}\mapsto\bigsqcup_{N\geq 0}\vec{\alpha}_N}) \circ \mathcal{M}(C',\Theta_{\vec{Y}\mapsto\bigsqcup_{N\geq 0}\vec{\alpha}_N}) \ =$$

$$\bigcup_{N\geq 0} \mathcal{M}(C'',\Theta_{\vec{Y}\mapsto\vec{\alpha}_N}) \circ \bigcup_{N\geq 0} \mathcal{M}(C',\Theta_{\vec{Y}\mapsto\vec{\alpha}_N}) \ =$$

$$\bigcup_{N\geq 0} (\mathcal{M}(C'',\Theta_{\vec{Y}\mapsto\vec{\alpha}_N}) \circ \mathcal{M}(C',\Theta_{Y\mapsto\vec{\alpha}_N})) \ = \ \bigcup_{N\geq 0} \mathcal{M}((C';C''),\Theta_{\vec{Y}\mapsto\vec{\alpha}_N}) \ ,$$

the penultimate equality using **P4**, plus the monotony which follows from the ctsness of the inductive hypothesis.

For $C = ite(H)(C')(C'')$,

$(s,s') \in \mathcal{M}(ite(H)(C')(C''), \Theta_{Y \mapsto \bigsqcup_{N\geq 0} \vec{\alpha}_N}) \iff$

$[H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C', \Theta_{Y \mapsto \bigsqcup_{N\geq 0} \vec{\alpha}_N}) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(C'', \Theta_{Y \mapsto \bigsqcup_{N\geq 0} \vec{\alpha}_N})] \iff$

$[H\mathbf{tt}@s \text{ and } (s,s') \in \bigcup_{N\geq 0} \mathcal{M}(C', \Theta_{Y \mapsto \vec{\alpha}_N}) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s,s') \in \bigcup_{N\geq 0} \mathcal{M}(C'', \Theta_{Y \mapsto \vec{\alpha}_N})] \iff$

$\exists N\, [H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C', \Theta_{Y \mapsto \vec{\alpha}_N}) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(C'', \Theta_{Y \mapsto \vec{\alpha}_N})] \iff$

$\exists N\, [(s,s') \in \mathcal{M}(ite(H)(C')(C''), \Theta_{Y \mapsto \vec{\alpha}_N})] \iff (s,s') \in \bigcup_{N\geq 0} \mathcal{M}(ite(H)(C')(C''), \Theta_{Y \mapsto \vec{\alpha}_N})\,.$

Note that putting the "$\exists N$" at the far left of the penultimate line depends on monotony, which follows from the ctsness of the inductive hypothesis.

When $\vec{C} = \overrightarrow{\nabla E/Z}$, we must prove that

$$\vec{\mathcal{M}}(\overrightarrow{\nabla E/Z}, \Theta_{\vec{Y} \mapsto \bigsqcup_{N\geq 0} \overrightarrow{\alpha_N}}) \;=\; \bigsqcup_{N\geq 0} \vec{\mathcal{M}}(\overrightarrow{\nabla E/Z}, \Theta_{\vec{Y} \mapsto \overrightarrow{\alpha_N}})\,.$$

The right-hand side is

$$\bigsqcup_{N\geq 0} lix[\, \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Y} \mapsto \vec{\alpha}_N})_{\vec{Z} \mapsto \vec{\beta}}\,].$$

The left-hand side is

$$lix[\, \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Y} \mapsto \bigsqcup_N \vec{\alpha}_N})_{\vec{Z} \mapsto \vec{\beta}}\,].$$

We may eliminate any of the Y_i which happen to be in $\vec{Z}$; i.e. let $\vec{X}$ be a list containing exactly the Y_i which are not in $\vec{Z}$. Also, let $\mathsf{new}k$ be the reduced value of k which is the length of the vector $\vec{X}$. And also form, for each $\vec{\alpha}_N$, that $\vec{\gamma}_N \in \mathcal{P}cv^{\mathsf{new}k}$ which agrees with $\vec{\alpha}_N$ in each slot where we have retained that particular procedure variable in $\vec{Y}$ when forming $\vec{X}$. So just change $\vec{Y}$ to $\vec{X}$, and $\vec{\alpha}_N$ to $\vec{\gamma}_N$, at the unique place each occurs in each display. Now we can also interchange the order of the subscripts to Θ in both displays. So we are left to prove that

$$lix[\, \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Z} \mapsto \vec{\beta}})_{\vec{X} \mapsto \bigsqcup_N \vec{\gamma}_N})\,] \;=\; \bigsqcup_{N\geq 0} lix[\, \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Z} \mapsto \vec{\beta}})_{\vec{X} \mapsto \vec{\gamma}_N})\,].$$

But, by the induction hypothesis applied to $\vec{E}$, the left-hand side is

$$lix[\ \vec{\beta} \mapsto \bigsqcup_N \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Z}\mapsto\vec{\beta}})_{\vec{X}\mapsto\vec{\gamma}_N})\ ]\ ,$$

which agrees with the right-hand side, by $\mathbf{P2}^{\mathsf{new}k}$, since the function

$$[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Z}\mapsto\vec{\beta}})_{\vec{X}\mapsto\vec{\gamma}_N})\ ]$$

is cts for every N by the inductive hypothesis, and since the Nth such function is $\sqsubset$-dominated by the one for $N+1$, using monotonicity, which follows from the ctsness of the inductive hypothesis applied to $\vec{E}$.

The free occurrences of $callX$ in a command are:
none for assignment commands, or for $callY$ when $Y \neq X$;
the whole command for $callX$;
those in C and/or in D for $(C;D)$ and for $ite(H)(C)(D)$;
none for $\nabla_j\vec{C}/\vec{Y}$ if $X = Y_i$ for some i;
and finally , those in the various C_i for $\nabla_j\vec{C}/\vec{Y}$ if $X \neq Y_i$ for all i .

Lemma. *If $\vec{E}$ has no free $call\vec{Z}$ then, for any Ψ and $\vec{\gamma}$,*

$$\vec{\mathcal{M}}(\vec{E}, \Psi_{\vec{Z}\mapsto\vec{\gamma}}) \ = \ \vec{\mathcal{M}}(\vec{E}, \Psi)\ .$$

The proof is by induction on E. The cases of atomic commands are easy as usual, and of sequencing and 'iteing' are the expected manipulations. When $E = \nabla_j\vec{F}/\vec{V}$, the list of right-hand sides for all the different j is

$$lix[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{F}, \Psi_{\vec{V}\mapsto\vec{\beta}})\ ]\ ,$$

whereas the the list of left-hand sides is

$$lix[\ \vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{F}, (\Psi_{\vec{X}\mapsto\vec{\gamma}})_{\vec{V}\mapsto\vec{\beta}})\ ]\ ,$$

As above, we can replace $\vec{X}$ by a 'subvector', containing only those X_i not in $\vec{V}$; and then commute the subscripts on Ψ. But now the inductive hypothesis applies to each F_ℓ, so the proof is done.

Definition. For any command C and any $\vec{D}/\vec{Y}$, define, by induction on the number of symbols in the string C, a command $C_{\vec{D}/\vec{Y}}$, as follows.

$(x \leftarrow\!\!\div t)_{\vec{D}/\vec{Y}} := x \leftarrow\!\!\div t$; $(callY_i)_{\vec{D}/\vec{Y}} := D_i$; $(callX)_{\vec{D}/\vec{Y}} := callX$ if $X \neq Y_i$ for all i ;

$(C';C'')_{\vec{D}/\vec{Y}} := (C'_{\vec{D}/\vec{Y}};C''_{\vec{D}/\vec{Y}})$; $ite(H)(C')(C'')_{\vec{D}/\vec{Y}} := ite(H)(C'_{\vec{D}/\vec{Y}})(C''_{\vec{D}/\vec{Y}})$.

In completing this definition, we shall use the convention

$$\vec{B}_{\vec{D}/\vec{Y}} \ := \ (B_{1\vec{D}/\vec{Y}}, B_{2\vec{D}/\vec{Y}}, \cdots) \ .$$

Given $\vec{E}/\vec{Z}$, vectors of length ℓ, and $\vec{D}/\vec{Y}$, vectors of length k, define

$$\overrightarrow{\nabla E/Z}_{\vec{D}/\vec{Y}} \ := \ (\ \nabla_1\vec{E}'_{\vec{D}/\vec{Y}}/\vec{Z}' \ , \ \nabla_2\vec{E}'_{\vec{D}/\vec{Y}}/\vec{Z}' \ , \ \cdots, \nabla_\ell\vec{E}'_{\vec{D}/\vec{Y}}/\vec{Z}' \ ) \ ,$$

where $\vec{Z}'$ is chosen to be the first procedure variable vector (temporarily give $\mathcal{P}cv^\ell$ a linear order for this purpose only) such that

$\vec{Z}' \cap \vec{Z} = \emptyset$; $\vec{Z}' \cap \vec{Y} = \emptyset$; $\vec{Z}' \notin \vec{D} \cup \vec{E}$ (meaning that no symbol Z'_i occurs) ,

and where $\vec{E}' \ := \ \vec{E}_{call\vec{Z}'/\vec{Z}}$, commands whose lengths are the same as those in $\vec{E}$, less than those in $\overrightarrow{\nabla E/Z}$, so the definition makes sense inductively.

Theorem 2. *For any C and $\vec{D}/\vec{Y}$, and any $\Theta \in \mathcal{B}sm^{\mathcal{P}cv}$, we have*

$$\mathcal{M}(C_{\vec{D}/\vec{Y}}, \Theta) \ = \ \mathcal{M}(C, \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)}) \ .$$

Proof by induction on the number of symbols in C. When C is $callY_i$, both sides are $\mathcal{M}(D_i, \Theta)$.
When $C = callX$ for X differing from all Y_i, both sides are $\Theta(X)$.
When C is an assignment command, both sides are the usual.
When $C = (C';C'')$, we have

$$\mathcal{M}((C';C'')_{\vec{D}/\vec{Y}}, \Theta) \ = \ \mathcal{M}((C'_{\vec{D}/\vec{Y}};C''_{\vec{D}/\vec{Y}}), \Theta) \ = \ \mathcal{M}(C''_{\vec{D}/\vec{Y}}, \Theta) \circ \mathcal{M}(C'_{\vec{D}/\vec{Y}}, \Theta) \ =$$

$$\mathcal{M}(C'', \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)}) \circ \mathcal{M}(C', \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)}) \ = \ \mathcal{M}((C';C''), \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)}) \ .$$

For $C = ite(H)(C')(C'')$,

$$\mathcal{M}(ite(H)(C')(C'')_{\vec{D}/\vec{Y}}, \Theta) \ = \ \mathcal{M}(ite(H)(C'_{\vec{D}/\vec{Y}})(C''_{\vec{D}/\vec{Y}}), \Theta) \ =$$

$$\{(s,s') \mid H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C'_{\vec{D}/\vec{Y}}, \Theta); \text{ or } H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(C''_{\vec{D}/\vec{Y}}, \Theta)\} \ =$$

$$\{(s,s') \mid H\mathbf{tt}@s \text{ and } (s,s') \in \mathcal{M}(C', \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)}); \text{ or } H\mathbf{ff}@s \text{ and } (s,s') \in \mathcal{M}(C'', \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)})\} \ =$$

$$\mathcal{M}(ite(H)(C')(C''), \Theta_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{D},\Theta)}) \ .$$

When $C = \nabla_i \vec{E}/\vec{Z}$, we do it 'vectorially', i.e. for all i simultaneously. We have

$$\vec{\mathcal{M}}(\overrightarrow{\nabla E/Z}_{\vec{D}/\vec{Y}}, \Theta) \;=\; \vec{\mathcal{M}}(\overrightarrow{\nabla E'_{\vec{D}/\vec{Y}}/Z'}, \Theta) \qquad \text{definition of } \overrightarrow{\nabla E/Z}_{\vec{D}/\vec{Y}}$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}'_{\vec{D}/\vec{Y}}, \Theta_{\vec{Z}' \mapsto \vec{\beta}})] \qquad \text{definition of } \nabla\text{−command semantics}$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}', (\Theta_{\vec{Z}' \mapsto \vec{\beta}})_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta_{\vec{Z}' \mapsto \vec{\beta}})})] \qquad \text{inductive hypothesis applied to all } E'_i$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}', \Psi)] \text{ where } \Psi := (\Theta_{\vec{Z}' \mapsto \vec{\beta}})_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)} \quad \text{lemma above and since } \vec{Z}' \notin \vec{D}$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, \Psi_{\vec{Z} \mapsto \vec{\mathcal{M}}(call \vec{Z}', \Psi)})] \quad \text{defn. of } \vec{E}' \text{ and inductive hypothesis for the } E_i$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, \Psi_{\vec{Z} \mapsto \Psi(\vec{Z}')})] \qquad \text{semantics of } call\text{−commands}$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, \Psi_{\vec{Z} \mapsto \vec{\beta}})] \qquad (\vec{Z}' \mapsto \vec{\beta})\text{−part of the definition of } \Psi$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, ((\Theta_{\vec{Z}' \mapsto \vec{\beta}})_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)})_{\vec{Z} \mapsto \vec{\beta}})] \qquad \text{overall definition of } \Psi$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, ((\Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)})_{\vec{Z} \mapsto \vec{\beta}})_{\vec{Z}' \mapsto \vec{\beta}})] \qquad \text{because } \vec{Z}' \text{ disjoint from } \vec{Z} \text{ and } \vec{Y}$$

$$= lix[\vec{\beta} \mapsto \vec{\mathcal{M}}(\vec{E}, (\Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)})_{\vec{Z} \mapsto \vec{\beta}})] \qquad \text{lemma above and since } \vec{Z}' \notin \vec{E}$$

$$= \vec{\mathcal{M}}(\overrightarrow{\nabla E/Z}, \Theta_{\vec{Y} \mapsto \vec{\mathcal{M}}(\vec{D}, \Theta)}) \qquad \text{definition of } \nabla\text{−command semantics}$$

as required.

Definition of "Unfolding". Define, by induction on $N \geq 0$, the command $(\vec{C}/\vec{Y})^{<N>}_j$, simultaneously letting $\overrightarrow{C/Y}^{<N>}$ be the 'vector' whose jth component is $(\vec{C}/\vec{Y})^{<N>}_j$, as follows :

$$(\vec{C}/\vec{Y})^{<0>}_j \;:=\; callY_j \quad , \quad (\vec{C}/\vec{Y})^{<N+1>}_j \;:=\; (C_j)_{(\overrightarrow{C/Y}^{<N>})/\vec{Y}} \; .$$

Recall that $m(C) \;:=\; \mathcal{M}(C, \Phi)$, where Φ maps all procedure variables to the empty set, $\emptyset$. The vector $\vec{m}(\vec{C})$ has components $m(C_i)$.

Theorem 3. *We have*

$$\vec{m}(\overrightarrow{\nabla C/\ Y}) = \bigsqcup_{N \geq 0} \vec{m}(\overrightarrow{C/Y}^{<N>}) \text{ ; equivalently, for all } j, \; m(\nabla_j \vec{C}/\vec{Y}) = \bigcup_{N \geq 0} m((\vec{C}/\vec{Y})^{<N>}_j) \; .$$

Proof. Given $\vec{C}/\vec{Y}$, let $\vec{F}(\vec{\beta}) = \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\beta}})$, recall that $\vec{\phi}$ is the k-tuple of nothing but $\emptyset$'s, and calculate as follows.

$$\vec{m}(\overrightarrow{\nabla C/Y}) \;=\; \vec{\mathcal{M}}(\overrightarrow{\nabla C/Y}, \Phi) \;=\; lix(\vec{F})$$

$$=\; \bigsqcup_{n=0}^{\infty} \vec{F}^n(\vec{\phi}) \;=\; \vec{\phi} \;\sqcup\; \vec{F}(\vec{\phi}) \;\sqcup\; \vec{F}^2(\vec{\phi}) \;\sqcup\; \vec{F}^3(\vec{\phi}) \;\sqcup\; \cdots$$

$$=\; \vec{\mathcal{M}}(C, \Phi_{\vec{Y}\mapsto\vec{\phi}}) \;\sqcup\; \vec{\mathcal{M}}(C, \Phi_{\vec{Y}\mapsto\vec{F}(\vec{\phi})}) \;\sqcup\; \vec{\mathcal{M}}(C, \Phi_{\vec{Y}\mapsto\vec{F}(\vec{F}(\vec{\phi}))}) \;\sqcup\; \cdots$$

$$=\; \vec{\mathcal{M}}(\vec{C}, \Phi) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{C},\Phi)}) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{C},\Phi_{\vec{Y}\mapsto\vec{\mathcal{M}}(\vec{C},\Phi)})}) \;\sqcup\; \cdots$$

$$=\; \vec{m}(\vec{C}) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}_{\vec{C}/\vec{Y}}, \Phi) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\mathcal{M}(C_{\vec{C}/\vec{Y}},\Phi)}) \;\sqcup\; \cdots$$

$$=\; \vec{m}(\vec{C}) \;\sqcup\; \vec{\mathcal{M}}(\overrightarrow{C/Y}^{<2>}, \Phi) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}, \Phi_{\vec{Y}\mapsto\vec{\mathcal{M}}(\overrightarrow{C/Y}^{<2>},\Phi)}) \;\sqcup\; \cdots$$

$$=\; \vec{m}(\vec{C}) \;\sqcup\; \vec{m}(\overrightarrow{C/Y}^{<2>}) \;\sqcup\; \vec{\mathcal{M}}(\vec{C}_{(\overrightarrow{C/Y}^{<2>})/\vec{Y}}, \Phi) \;\sqcup\; \cdots$$

$$=\; \vec{m}(\vec{C}) \;\sqcup\; \vec{m}(\overrightarrow{C/Y}^{<2>}) \;\sqcup\; \vec{\mathcal{M}}(\overrightarrow{C/Y}^{<3>}, \Phi) \;\sqcup\; \cdots$$

$$=\; \vec{m}(\overrightarrow{C/Y}^{<1>}) \sqcup \vec{m}(\overrightarrow{C/Y}^{<2>}) \sqcup \vec{m}(\overrightarrow{C/Y}^{<3>}) \sqcup \cdots \;=\; \bigsqcup_{N} \vec{m}(\overrightarrow{C/Y}^{<N>}) \,.$$

The 3rd equality uses the alternative formula $(***)$ for the least fixed point. It also uses **Theorem 1** since F must be cts for that formula to be valid. The 7th and 9th equalities use **Theorem 2**. (The fussy mathematician will do this without the dots.)

I think the most interesting thing here is the connection between that beautiful, simple formula for *lix* and the desire to show that operational and fixpoint semantics coincide. The book [**deBa**] avoids dealing with the connection between the operational and fixpoint semantics uniquely for this type of language with recursive commands (or even giving the operational semantics), and its author is a serious follower of Dana Scott. And I've never seen our **Theorem 2** before. So I may be shocked to later learn that this connection had gone unnoticed.

Remark. The correct generalization of the theorem to 'relative' semantics, with essentially the same proof, is

$$\vec{\mathcal{M}}(\overrightarrow{\nabla C/\ Y}, \Theta) \ = \ \bigsqcup_{N \geq 0} \vec{\mathcal{M}}(\overrightarrow{C/Y}^{<N>}, \Theta_{\vec{Y} \mapsto \vec{\phi}}) \ .$$

Afterthought. One has the unsurprising result that $\mathcal{R}ec_{\infty +}$ with its semantics is a *deterministic* command language. That is, all $m(C)$ are graphs of (partial) functions—i.e. of functions from a subset of $\mathcal{S}te$ to $\mathcal{S}te$. One proves inductively on C that, if $\Theta(X)$ is a graph for all $X \in \mathcal{P}cv$, then $\mathcal{M}(C, \Theta)$ is also a graph. And then take $\Theta = \Phi$. We'll leave this as an exercise.

Appendix : Proofs of Section 0 Results.

Proof of Proposition DFX1. Assuming that $\alpha \subset \beta$, **Lemma P3** immediately gives $\alpha \circ m_{DFX}(C) \subset \beta \circ m_{DFX}(C)$, from which it is clear that $f_{H,C}(\alpha) \subset f_{H,C}(\beta)$, as required.

Proof of Theorem DFX2. Here we need only push through the part of the induction on commands that shows m_{GSO} and m_{DFX} to agree on while-do commands. Fix C and H. Abbreviate $f_{H,C}$ to f, and $m_{GSO}(whdo(H)(C))$ to α. We'll show directly from the definition of the least fixed point that $\alpha = lix(f)$ as required. The first two paragraphs below prove the fixpoint property, and the third proves it to be 'least'. Abbreviate $m_{DFX}(C)$ to $m(C)$, which agrees with $m_{GSO}(C)$ by the inductive assumption.

To show that $f(\alpha) \subset \alpha$: Let $(s, s') \in f(\alpha)$. Then either $s' = s$ and $H\mathsf{ff}@s$, or else $(s, s') \in \alpha \circ m(C)$ and $H\mathsf{tt}@s$. In the former case, take $N = 0$ in the definition of α to see that $(s, s) \in \alpha$. In the latter case, there is an s'' such that $(s, s'') \in m(C)$ and $(s'', s') \in \alpha$. The second of those guarantees a sequence $s'' = s_1, s_2, \cdots, s_N = s'$ for some $N \geq 1$ for which $H\mathsf{ff}@s_N$, but for $1 \leq i < N$ we have $H\mathsf{tt}@s_i$ and $(s_i, s_{i+1}) \in m(C)$. Now just append $s_0 := s$ to the left end of that sequence to get the corresponding condition showing $(s, s') \in \alpha$.

To show that $\alpha \subset f(\alpha)$: Let $(s, s') \in \alpha$. That guarantees a sequence $s = s_0, s_1, \cdots, s_N = s'$ for some $N \geq 0$ for which $H\mathsf{ff}@s_N$, but for $0 \leq i < N$ we have $H\mathsf{tt}@s_i$ and $(s_i, s_{i+1}) \in m(C)$. If $N = 0$, we get $s' = s$, and $(s, s) \in f(\alpha)$—in fact, it's in $f(\beta)$ for *any* β. If $N > 0$, note that $(s_1, s') \in \alpha$ also, by merely dropping s_0 from the above data, and effectively decreasing N by 1. But $(s, s_1) \in m(C)$, and so $(s, s') \in \alpha \circ m(C)$. Since $H\mathsf{tt}@s$ here, we get $(s, s') \in f(\alpha)$, as required.

To show that $f(\beta) = \beta$ implies $\alpha \subset \beta$: We actually prove the superficially stronger fact that $f(\beta) \subset \beta$ implies $\alpha \subset f(\beta)$. Let $(s, s') \in \alpha$ with a corresponding N and $s = s_0, s_1, \cdots, s_N = s'$ as in the previous paragraph. Such a pair will be said to be in α_N, so that α is the union of the α_N. We'll prove by induction on N that $\alpha_N \subset f(\beta)$. When $N = 0$, we get $s' = s$, and, as noted above, $(s, s) \in f(\beta)$. For the inductive step on N, we see that $(s_1, s') \in \alpha_{N-1}$, so it's in β. Combined with $(s, s_1) \in m(C)$, we get $(s, s') \in \beta \circ m(C)$. Since $H\mathsf{tt}@s$ here, we get $(s, s') \in f(\beta)$, as required.

Proof of Theorem BSO2. As mentioned in the remarks ending Section 0, we shall show by induction on commands that $m_{_{GSO}}$ and $m_{_{BSO}}$ agree on all commands, and suppress the assignment command case to avoid the risk that it might anesthetize us.

To show that $m_{_{BSO}}(C) \subset m_{_{GSO}}(C)$: For the inductive case $C = (D; E)$, if $(s, s') \in m_{_{BSO}}(C)$, a BS-derivation to show this must appeal to rule (III) for that last line. (Note that this does depend on unique readability of the strings in $\mathcal{A}ten$. In future, dependence on that will be allowed to slide by silently.) By the required appearances of certain earlier lines for the rule to be applicable, we have some s'' such that both $(s, s'') \in m_{_{BSO}}(D)$ and $(s'', s') \in m_{_{BSO}}(E)$. But the subscript on m can be changed to GSO in both cases by the inductive hypothesis. Thus $(s, s') \in m_{_{GSO}}(E) \circ m_{_{GSO}}(D) = m_{_{GSO}}(C)$, as required.

For the inductive case $C = whdo(H)(D)$, if $(s, s') \in m_{_{BSO}}(C)$, a BS-derivation to show this must appeal to rule (II) or rule (IV) for that last line, depending respectively on whether $H\mathsf{ff}@s$ or $H\mathsf{tt}@s$. In the first case, $s' = s$ and the result is immediate. In the other case, by the required earlier lines for the rule to be applicable, we have some s'' such that both $(s, s'') \in m_{_{BSO}}(D)$ and $(s'', s') \in m_{_{BSO}}(C)$. But the subscript on m can be changed to GSO in the first of these two underlined by the inductive hypothesis. Suppose for a contradiction that $(s, s') \notin m_{_{GSO}}(C)$. We may take the derivation referred to as the shortest one, varying over all D and all (s, s') for which our supposition holds. By that "shortest", it now follows that the subscript on m can be changed to GSO also in the second of the underlined. But that gives a sequence $s'' = s_1, s_2, \cdots, s_N = s'$ as in the definition verifying $(s'', s') \in m_{_{GSO}}(C)$. Appending $s_0 := s$ to this gives $(s, s') \in m_{_{GSO}}(C)$, as required.

To show that $m_{_{GSO}}(C) \subset m_{_{BSO}}(C)$: For the inductive case $C = (D; E)$, if $(s, s') \in m_{_{GSO}}(C)$, we have some s'' such that both $(s, s'') \in m_{_{GSO}}(D)$ and $(s'', s') \in m_{_{GSO}}(E)$. But the subscript on m can be changed to BSO in both cases by the inductive hypothesis. For each of these assertions, choose a BS-derivation which 'witnesses' it, and append the second derivation to the first, then write down a final line $(s\ ,\ C\ ,\ s')$, appealing to rule (III) for that last line. The resulting derivation shows $(s, s') \in m_{_{BSO}}(C)$, as required.

For the inductive case $C = whdo(H)(D)$, if $(s, s') \in m_{_{GSO}}(C)$, there is a sequence verifying that fact, $s = s_0, s_1, \cdots, s_N = s'$, as in the definition. If $N = 0$, then $s' = s$, and there is an obvious 1-line BS-derivation to witness $(s, s) \in m_{_{BSO}}(C)$, as required, using rule (II), since here, $H\mathsf{ff}@s$. If

$N > 0$, then $\underline{H\mathtt{tt}@s}$, and we can extract $s = s_0$ from the sequence and below use the fact that $\underline{(s, s_1) \in m_{_{BSO}}(D)}$ by the induction on commands. Using the remaining sequence $s_1, \cdots, s_N$ showing $(s_1, s') \in m_{_{GSO}}(C)$, we realize that this subproof should be proceeding by induction on N. That gets us $\underline{(s_1, s') \in m_{_{BSO}}(C)}$. The three underlined just above allow us to concoct a BS-derivation for the required statement, namely $(s, s') \in m_{_{BSO}}(C)$, with final line justified by rule (IV).

Proof of Theorem TSO. Following the remarks ending Section 0, we'll show $m_{_{GSO}}$ and $m_{_{TSO}}$ to agree on all commands, proceeding by induction on commands, and suppressing the assignment command case on the grounds that it may catatonify us.

$\underline{\text{To show that } m_{_{GSO}}(C) \subset m_{_{TSO}}(C) :}$ For the inductive case $C = (D; E)$, if $(s, s') \in m_{_{GSO}}(C)$, we have some s'' such that both $(s, s'') \in m_{_{GSO}}(D)$ and $(s'', s') \in m_{_{GSO}}(E)$. But the subscript on m can be changed to TSO in both cases by the inductive hypothesis. So there are (unique by the lemma at the end of Section 0) tiny step reductions beginning as

$$(D, s) = (D_0, s_0) \ \stackrel{1}{\to}\ (D_1, s_1) \ \stackrel{1}{\to}\ \cdots \ \stackrel{1}{\to}\ (D_n, s_n) = (bugga, s'')$$

$$(E, s'') = (E_0, s_0) \ \stackrel{1}{\to}\ (E_1, s_1) \ \stackrel{1}{\to}\ \cdots \ \stackrel{1}{\to}\ (E_k, s_k) = (bugga, s')$$

reaching *bugga* for the first time after "n" and "k" steps, as indicated. Possibly $n = 0$ or $k = 0$, i.e. respectively $D = bugga$ or $E = bugga$, in which case the corresponding sequence disappears. By that lemma, we have that $D_i \neq D_{i+1}$ for each i in the upper of the displays, so one of the rules of construction for tiny steps yields a sequence

$$((D; E), s) \ \stackrel{1}{\to}\ ((D_1; E), s_1) \ \stackrel{1}{\to}\ \cdots \ \stackrel{1}{\to}\ ((D_{n-1}; E), s_{n-1}) \ \stackrel{1}{\to}\ ((bugga; E), s'')$$

Attaching $((bugga; E), s'') \ \stackrel{1}{\to}\ (E, s'')$ to the right-hand end of the latter, and then attaching the lower of the displays, we get a sequence showing $(s, s') \in m_{_{TSO}}((D; E))$, as required.

For the inductive case $C = whdo(H)(D)$, if $(s, s') \in m_{_{GSO}}(C)$, there is a sequence verifying that fact, $s = r_0, r_1, \cdots, r_N = s'$, as in the definition, but changing s_i there to r_i. If $N = 0$, then $s' = s$, and there is the obvious single tiny step $(whdo(H)(D), s) \ \stackrel{1}{\to}\ (bugga, s)$ showing $(s, s) \in m_{_{TSO}}(C)$, as required, since here, $H\mathtt{ff}@s$. If $N > 0$, then $\underline{H\mathtt{tt}@s}$, and we can extract

$s = r_0$ from the sequence and below use the fact that $\underline{(s, r_1) \in m_{TSO}(D)}$ by the induction on commands. Using the remaining sequence $r_1, \cdots, r_N$ showing $(r_1, s') \in m_{GSO}(C)$, we realize that this subproof should be proceeding by induction on N. That gets us $\underline{(r_1, s') \in m_{TSO}(C)}$. The three underlined just above allow us to concoct a sequence of tiny step reductions for the required statement, namely $(s, s') \in m_{TSO}(C)$, by beginning with $(whdo(H)(D), s) \xrightarrow{1} ((D; whdo(H)(D)), s)$, followed by the sequence

$$((D; whdo(H)(D)), s) \xrightarrow{1} ((D_1; whdo(H)(D)), s_1) \xrightarrow{1} \cdots$$

$$((D_{n-1}; whdo(H)(D)), s_{n-1}) \xrightarrow{1} ((bugga; whdo(H)(D)), r_1)$$

coming, as in the previous paragraph, from

$$(D, s) = (D_0, s_0) \xrightarrow{1} (D_1, s_1) \xrightarrow{1} \cdots \xrightarrow{1} (D_n, s_n) = (bugga, r_1)$$

followed by

$$((bugga; whdo(H)(D)), r_1) \xrightarrow{1} (whdo(H)(D), r_1)$$

and then finishing with a sequence reducing $(whdo(H)(D), r_1)$ to $(bugga, s')$.

To show that $\underline{m_{TSO}(C) \subset m_{GSO}(C)}$: For the inductive case $C = (D; E)$, if $\underline{(s, s') \in m_{TSO}(C)}$, write down the unique sequence of tiny step reductions showing this. For some unique s'', it must begin with a (possibly empty if $D = bugga$) sequence of the form

$$((D; E), s) \xrightarrow{1} ((D_1; E), s_1) \xrightarrow{1} \cdots \xrightarrow{1} ((D_{n-1}; E), s_{n-1}) \xrightarrow{1} ((bugga; E), s'')$$

coming from

$$(D, s) = (D_0, s_0) \xrightarrow{1} (D_1, s_1) \xrightarrow{1} \cdots \xrightarrow{1} (D_n, s_n) = (bugga, s'')$$

followed by a single step $((bugga; E), s'') \xrightarrow{1} (E, s'')$, finishing with

$$(E, s'') = (E_0, s_0) \xrightarrow{1} (E_1, s_1) \xrightarrow{1} \cdots \xrightarrow{1} (E_k, s_k) = (bugga, s')$$

So we have some s'' such that both $(s, s'') \in m_{TSO}(D)$ and $(s'', s') \in m_{TSO}(E)$. But the subscript on m can be changed to GSO in both cases by the inductive hypothesis. Thus $(s, s') \in m_{GSO}(E) \circ m_{GSO}(D) = m_{GSO}(C)$, as required.

For the inductive case $C = whdo(H)(D)$, let $(s, s') \in m_{TSO}(C)$, and write down the unique sequence of tiny step reductions showing this. Proceed by induction on the length of that sequence. If that length is 1, the tiny step must be $(whdo(H)(D), s) \xrightarrow{1} (bugga, s)$ with $s' = s$, and $H\mathtt{ff}@s$. But $(s, s) \in m_{GSO}(C)$. For the inductive step on the length, the latter being larger than 1 forces $H\mathtt{tt}@s$, and the sequence must take the form by beginning with $(whdo(H)(D), s) \xrightarrow{1} ((D; whdo(H)(D)), s)$, followed by a sequence

$$((D; whdo(H)(D)), s) \xrightarrow{1} ((D_1; whdo(H)(D)), s_1) \xrightarrow{1} \cdots$$

$$((D_{n-1}; whdo(H)(D)), s_{n-1}) \xrightarrow{1} ((bugga; whdo(H)(D)), r_1)$$

coming from

$$(D, s) = (D_0, s_0) \xrightarrow{1} (D_1, s_1) \xrightarrow{1} \cdots \xrightarrow{1} (D_n, s_n) = (bugga, r_1)$$

followed by

$$((bugga; whdo(H)(D)), r_1) \xrightarrow{1} (whdo(H)(D), r_1)$$

and then finishing with a sequence reducing $(whdo(H)(D), r_1)$ to $(bugga, s')$. (We are relying on the earlier lemma, asserting uniqueness of 1-step reductions, in several places above, as well as on unique readability, of course.) This very last subsequence is shorter than the entire sequence, so by the inner induction this verification that $(r_1, s') \in m_{TSO}(C)$ allows us to change the subscript just written to GSO. But that gives a sequence $r_1, r_2, \cdots, r_N = s'$ as in the definition verifying $(r_1, s') \in m_{GSO}(C)$. Appending $r_0 := s$ to this gives $(s, s') \in m_{GSO}(C)$, as required, since $(s, r_1) \in m_{GSO}(D)$ follows from the penultimate display above combined with the inductive (on commands) hypothesis.

Proof of Theorem SGR1. We prove the final line of the theorem, all that's needed. Inspect that obese definition and proceed by induction on commands to see that Q_1 only takes states as its values.

To prove the other statement, proceed by induction on k, and for fixed k, by induction on C. For $k = 1$, there is nothing to prove, as we just observed.

For the inductive step on k, let $i + 1 = k$ in the definition. The result is then immediate for assignment commands by the definition.

For sequencing, suppose that $Q_k((C;D),s) = \not\exists$. We clearly must have the second clause of the inductive definition, that is, $Q_{k-\ell}(D, Q_\ell(C,s)) = \not\exists$ where $\ell < k$ and $Q_{\ell+1}(C,s) = \not\exists \neq Q_\ell(C,s)$. But now that 2nd clause applies for the calculation of $Q_{k+1}((C;D),s)$, which is $Q_{k+1-\ell}(D, Q_\ell(C,s))$, which itself must be $\not\exists$, by the inductive hypothesis on commands (applied to D).

For 'while-do', the middle clause clearly cannot apply. If it is the first clause, the result is clear, since that clause also gives the calculation of $Q_{k+1}(whdo(H)(C),s)$. Now the condition which dictates that 3rd clause, namely

$$H\mathbf{tt}@s \quad \text{and} \quad \exists \ell < k \ \text{ with } \ Q_{\ell+1}(C,s) = \not\exists \neq Q_\ell(C,s)$$

certainly also holds with k changed to $k+1$. So that clause also gives the calculation of $Q_{k+1}(whdo(H)(C),s)$, which is therefore $Q_{k+1-\ell}(whdo(H)(C), Q_\ell(C,s))$, which itself must be $\not\exists$, by the inductive hypothesis on k.

Proof of Theorem SGR2.

Following the remarks ending Section 0, we'll show m_{GSO} and m_{SGR} to agree on all commands, proceeding by induction on commands, and suppressing the assignment command case since, though Neitsche said: "Against boredom, the gods themselves fight in vain", we mortals can at least try.

To show that $m_{GSO}(C) \subset m_{SGR}(C)$: For the inductive case $C = (D;E)$, if $(s,s') \in m_{GSO}(C)$, we have some s'' such that both $(s,s'') \in m_{GSO}(D)$ and $(s'',s') \in m_{GSO}(E)$. But the subscript on m can be changed to SGR for both, by the inductive hypothesis. Thus we have, for some positive k and ℓ, that

$$s'' = Q_k(D,s) \neq \not\exists = Q_{k+1}(D,s) \ \text{ and } \ s' = Q_\ell(D,s'') \neq \not\exists = Q_{\ell+1}(D,s'') \,.$$

Now the definition of $Q_*((D;E),s)$ gives its initial "$k+\ell$" terms as the concatenation of the non-$\not\exists$ beginnings of the two sequences $Q_*(D,s)$ and then $Q_*(E,s'')$ (followed by nothing but $\not\exists$'s). That yields the required fact that $(s,s') \in m_{SGR}(C)$.

For the inductive case $C = whdo(H)(D)$, if $(s,s') \in m_{GSO}(C)$, there is a sequence verifying that fact, $s = r_0, r_1, \cdots, r_N = s'$, as in the definition, but changing s_i there to r_i. If $N = 0$, then $s' = s$ and $H\mathbf{ff}@s$. But then $Q_*(C,s)$ is s followed by nothing but $\not\exists$'s, yielding the required fact that

$(s,s) \in m_{SGR}(C)$. If $N > 0$, then $H\mathsf{tt}@s$. Because, using the inductive assumption, $(r_i, r_{i+1}) \in m_{SGR}(C)$ for $0 \leq i < N$, the sequences $Q_*(C, r_i)$ each get to r_{i+1} then become nothing but $\nexists$'s. But then the definition of $Q_*(C,s)$ gives it as the concatenation of the non-$\nexists$ beginnings of those sequences (in order of increasing i, of course), followed by nothing but $\nexists$'s. Then we can just read off from that sequence the required fact that $(s,s') \in m_{SGR}(C)$.

To show that $m_{SGR}(C) \subset m_{GSO}(C)$: For the inductive case $C = (D;E)$, if $(s,s') \in m_{SGR}(C)$, the sequence $Q_*(C,s)$ necessarily does end with nothing but $\nexists$'s, immediately preceded by the term s'. But by definition of $Q_*((D;E),s)$, that can only happen if both $Q_*(D,s)$ and $Q_*(E,s'')$ are similarly 'finite', with $s'' := OUT(D,s)$. It follows that $OUT(D,s'') = s'$ and that $(s,s'') \in m_{SGR}(D)$ and $(s'',s') \in m_{SGR}(E)$. But the subscript on m can be changed to GSO for both, by the inductive hypothesis. Thus $(s,s') \in m_{GSO}(E) \circ m_{GSO}(D) = m_{GSO}(C)$, as required.

For the inductive case $C = whdo(H)(D)$, let $(s,s') \in m_{SGR}(C)$. If $H\mathsf{ff}@s$, the sequence $Q_*(C,s)$ has the single term s followed by nothing but $\nexists$'s. Thus $s' = s$ and $(s,s') \in m_{GSO}(C)$, as required. If $H\mathsf{tt}@s$, since the sequence $Q_*(C,s)$ 'terminates', it follows from its definition that the 'finite part' of that sequence is, for some states t_i and some $N > 0$, the concatenation of the 'finite parts' of sequences

$$Q_*(D,t_0)\ ,\ Q_*(D,t_1)\ ,\ \cdots\ ,\ Q_*(D,t_{N-1})\ ,$$

the part of $Q_*(D,t_i)$ ending with the term t_{i+1}, and where $t_0 = s$ and $t_N = s'$. And furthermore, $H\mathsf{ff}@t_N$, but $H\mathsf{tt}@t_i$ for $i < N$. It is immediate that $(t_i, t_{i+1}) \in m_{SGR}(D)$ for $i < N$. But induction tells us that we can change that subscript to GSO. With that, we have all the data needed from its definition to conclude that, for each i, the set $m_{GSO}(C)$ contains the pair (s,s'), as required.

Conjecture. tiny steps=grundge steps

This can be an exercise for the reader now, and maybe a project for me later, when this paper gets updated.

References

[**Apt**] Apt, K.R. *Ten Years of Hoare's Logic: A Survey—Part 1.* ACM Trans. Prog. Lang. Syst. 3(4), Oct. 1981, 431-483.

[**AdB**] America, Pierre and de Boer, Frank *Proving Total Correctness of Recursive Procedures.* Information and Computation 84(2), 1990, 129-162.

[**book**] Hoffman, P. *Logic for the Mathematical.* (this web page).

[**deBa**] deBaaker, Jaco *Mathematical Theory of Program Correctness.* Prentice/Hall International, 1980.

[**depth**] Hoffman, P. *Depth-Measure Semantics and Partial-Correctness Rules for Imperative Recursion.* (this website).(Part 2 of the present paper!)

[**Harel**] Harel, David *First-Order Dynamic Logic.* Lecture Notes in CS # 68, Springer, 1979. See also *Correctness of regular deterministic programs.* Theor. Comp. Sci. 12(1980) 61-81.

[**Musk**] Muskens, Reinhard *Program Semantics and Classical Logic.* website : http://let.uvt.nl/general/people/rmuskens/ .

[**mutu**] Hoffman, P. *Imperative Mutual Simple Recursion Proof Systems.* (this website).

[**Nip**] Nipkow, Tobias *Hoare Logics for Recursive Procedures and Unbounded Nondeterminism.* pp. 103-119 of : *Computer Science Logic (CSL 2002),* Lecture Notes in CS # 2471, Springer, 2002.

[**Nip2**] Nipkow, Tobias *Winskel is (almost) right.* pp. 180-192 of : *Foundations of Software Technology and Theoretical Computer Science.* eds. V. Chandru and V. Vinay, Lecture Notes in CS # 1180, Springer, 1996.

[**Old**] Olderog, E-R. *Sound and Complete Hoare-like Calculi Based on Copy Rules.* Acta Inf. 16, 1981, 161-197.

[**sing**] Hoffman, P. *Deterministic Dynamic Logic Imperative Recursive Programming Proof Systems.* (this website).

[**vOh**] van Oheimb, David *Hoare Logic for Mutual Recursion and Local Variables.* in : *Foundations of Software Technology and Theoretical Computer Science* (eds. C. Pandu Rangan, V. Raman and R. Ramanujam) pp.168-180 of Lecture Notes in CS # 1783, Springer, 1999.

[**W**] Winskel, Glynn *The Formal Semantics of Programming Languages.* MIT Press, 1993.