Variable Depth Recursion Semantics and Correctness

Peter Hoffman

Pure Mathematics, University of Waterloo

Abstact. *We consider a 'completely general' manifestation of parameterless imperative recursion on a single procedure variable. The principal new technical method is a recursion-depth-measure semantics where the depth bound can be different for different recursions within a given command. The more severe technicalities relate to substitution for procedure variables, where variable clash avoidance via re-naming bound variables is the motivation for the form of semantics above. The other novelty is a new rule for partial correctness of recursive commands which seems to be needed for completeness when the assumption is dropped from earlier work of simpleness for recursions. The main results are a validity theorem for this rule, plus proof that both the partial and the total correctness halves of the statement below are true. That statement claims this stronger new rule, together with ones already used to get a Cook-complete dynamic logic system for simple recursion, will succeed similarly with the simplicity assumption removed. By using the language of dynamic logic, both partial and total correctness assertions (and others as well) are expressible.*

Apparently, specifying a sound and Cook-complete proof system for *total* correctness of recursion, where declarations are local (and so one can, in particular, have nested recursions), is a matter that somehow got 'left behind' some years back. (Perhaps the needed technicalities, some appearing below, acted as a disincentive.) By restricting to so-called *simple* recursion, we had found such a system in [**infocomp**]. It uses the language of dynamic logic, and so is able to deal with both total and partial correctness, and to formalize and derive other statements, such as semantic equivalence of commands, as well. Here we initially present some rather technical results, dealing with a modification of the system above to make it work correctly for the *partial* correctness aspects, after one drops the restriction of simpleness. Somewhat surprising is that, concerning *total* correctness of recursive commands, no modification is needed, in fact even a simplification is possible, to the old system; and completeness follows, with a less tortuous proof, now without simpleness being assumed. (As exposited in [**Harel**], once those two aspects are dealt with so far as proving Cook-completeness, the job is finished for the entire language. The obstruction on the total side before was only a mental block—basically a form of the rule

which was thought to be needed only for strictly mutual recursion now gets used in a different way to deal with the non-simple case of the total correctness, needed even in the strictly single (i.e. 'non-mutual') case dealt with in this paper.)

More precisely, we first point out how one may evade both fixed point considerations and syntactic unfoldings, and define the semantics of (parameterless, but nestable) recursive commands using only a formalization of the idea of recursion depth. And then, motivated by that formalization, we see how to 'totally unfold' a command to give an unfolding version of the semantics. A (likely) novelty here is that, in order to treat recursion in a sufficiently general form, we use a depth measure which can vary from one recursion to another within a given command. In the largely pedagogical [**sem**] and in [**dpth**], considerably more detail is given on this material and several other related matters. In particular, all those proof details left out here are given in excruciating detail in [**dpth**]. (Here we have avoided *mutual* recursion. Past experience does indicate that messiness, rather than the need for new ideas, will likely be the only impediment to extending this work to cover the mutual case.)

The usual 'Floyd-Hoare' rule for partial correctness of a recursive command (essentially due to Dana Scott) takes several forms. We prove valid a more general version of the rule from [**infocom**] for nested recursion, the old version having assumed simpleness, which may now be dropped. This part of the work was stimulated by [**Nip**], where no nesting of recursion is allowed. The indicated proof there of the (classical, fixed-body, only global declaration) recursion rule's validity apparently (being machine generated, it's not manifestly accessible) uses nothing but recursion depth. Because nested recursion is allowed here (more generally, calls are local, not necessarily global), our treatment of depth is more complicated than that in [**Nip**]. And depth analysis becomes an order of magnitude even more involved as a result of allowing **non-simpleness** [e.g. *one can have a recursive command on some variable, the body of which has a recursive subcommand on a different variable, the latter of which has a body which includes a free occurrence of (i.e. a call to) the earlier variable*].

Having proved the rule valid, we then give the appropriate (partial part) of the completeness result, when that rule is used in conjunction with the other straightforward classical rules in [**infocom**]. Even though it refers only to the case of partial correctness with its vast literature, the last result is possibly new even independently of the language and system used, as I am still unsure whether earlier work on imperative recursion partial correctness

(e.g. [**Old**]) is able to deal with non-simpleness.

Finally, we use the old total correctness rule for mutual simple recursion in a different way here for single non-simple recursion to get the remainder of the completeness proof.

In Section 1, we recall the syntactic definition of $\mathcal{R}ec_{1+}$ from [**sem**]. That set of strings is one form of the simplest deterministic Turing equivalent imperative language which includes recursion on one procedure variable (at a time), and for which no 'non-nesting' nor 'simpleness' restrictions are imposed. At the same time, a sublanguage important for Section 3, and called $\mathcal{R}ec_{1.01}$, is delineated. It consists of those commands where the 'bound' procedure variables have all been 're-named' so as to be distinct from one another. Thus, no computing power has been lost, just elegance. (Recall that much of the complexity in [**Old**] revolves around the need for re-naming identifiers because of the ubiquity of substitution operations in 'doing unfoldings'.) Then, back in the slightly more general $\mathcal{R}ec_{1+}$, we use the notational conventions from [**sem**], and define the 'depth $\underline{n}$ semantics'. Finally, a series of totally elementary proofs are indicated for its most basic properties, leading up to showing that, when $\underline{n} = \underline{\infty}$, the usual fixpoint semantics is recovered. The notation $\underline{n}$ refers to a depth bound for various recursions within a command, with $\underline{\infty}$ indicating no bound. We can, of course, as well have $\underline{n}$ with $\underline{n}(X) = \infty$ for some procedure variables X, but not all.

In Section 2, we give the total unfolding definition which parallels the depth measure above. The overall semantics of the $\underline{n}$th unfolding of a command C is easily seen to coincide with the $\underline{n}$-depth approximation semantics for C itself. Then it follows quickly that these unfoldings are all recursion-free commands, the union of whose semantic (i.e. input-output pairs of states) sets is the semantic set for the given command being unfolded. The reader may be interested in consulting [**Musk**], not in the mainstream of CS literature, where a quite different version of the same result is given, with everything based on $2^{\underline{\text{nd}}}$order type-logic.

In the rather technical Section 3, we first discuss substitution for free occurrences of a procedure variable, say X. The usual complication (caused by *not wanting free occurrences of any procedure variable in the command which is to be substituted for X to become bound after the substitution*) has its usual solution via re-naming bound variables. Since a decent theory here seems to require a separate depth measure for each procedure variable, this messes things up a bit unless all the bound variables are distinct, so we stick

to $\mathcal{R}ec_{1.01}$ here. And we make claims, concerning a command produced by substitution, only for those $\underline{n}$ taking the same value on a variable and all its re-namings; that is, all these must have the same depth bound. Since a separate depth bound seems most natural not just for each variable which occurs bound but actually for each separate recursion within the command, the use of the sublanguage $\mathcal{R}ec_{1.01}$ now has a second motivation. The main formula here, concerning the depth-semantics of a command resulting from a substitution for a procedure variable, is the central step in the validity result described just below.

In Section 4 the partial correctness rule for recursion from [**infocom**] is recalled, and souped up to deal with the possibility of non-simpleness. This souping up not only deals with the 'variable-clash-difficulties' which the earlier sections are largely directed towards, but also generalizes the rule in a way which seems necessary to actually have completeness in the non-simple case. I have not found any earlier completeness theorem which explicitly allows non-simpleness, despite the extensive literature on partial (as opposed to total) correctness. The validity of the rule is then proved using a notion of $\underline{n}$-*truth*, with $\underline{n}$ as before: a 'supernatural-number'-valued function on procedure variables. The only tedium in the proof (not actually 'ploughed through' here) is checking, for all the more innocuous rules, that a certain relation from [**infocom**] preserves $\underline{n}$-truth. Note that the style of system we use here does have the usual so-called 'metarule' sort of rule for recursive command correctness, yet doesn't suffer from the danger of changing the meaning of the rule by introducing new rules for other features, thereby possibly inadvertantly converting a sound system into an unsound one.

In Section 5, we prove the promised partial completeness theorem for the system containing the new rule.

Finally, in Section 6, we prove the total completeness theorem for the system, and thereby, immediately from an old theorem of Harel, completeness in general has been established. That is, Cook-completeness is now known for the dynamic logic system consisting of the new rule (to replace the corresponding rule in [**infocomp**]), plus all the others there. This generalizes the main result there, it being mildly surprising that neither had been done decades ago. In particular, both the 4th and the 5th sections there, essentially proving completeness for the partial correctness part (respectively total correctness part), are somewhat simplified here (as well as generalized to the

non-simple case). The only concession needed for obtaining this generalization and simplification is to include the stronger rule for partial correctness.

The double-entendre on the words "partial" and "total" beginning the two previous paragraphs is not unintended.

1. **Depth-measure semantics.**

See [**sem**] for all the basic notation. We have procedure variables such as X below, forming a countable set $\mathcal{P}cv$; plus the language of $1^{\underline{\text{st}}}$order number theory and its standard interpretation $\mathbf{N}$ (in particular, variables such as y and terms such as t below). For a command C, we shall want to refer to its sets, $fr(C)$, of *free procedure variables*, and $bd(C)$, of *bound procedure variables*. Here is the definition by structural induction of triples of objects $[C, fr(C), bd(C)]$ defining the triple

$$[\ \mathcal{R}ec_{1+}\ ,\ fr : \mathcal{R}ec_{1+} \to 2^{\mathcal{P}cv}\ ,\ bd : \mathcal{R}ec_{1+} \to 2^{\mathcal{P}cv}\ ].$$

(1) Assignment : $[\ y \leftarrow\!\!\div t\ \ ,\ \emptyset\ ,\ \emptyset\ ]$

(2) Call : $[\ callX\ ,\ \{X\}\ ,\ \emptyset\ ]$

(3) Sequencing : $[\ (C';C'')\ ,\ fr(C') \cup fr(C'')\ ,\ bd(C') \cup bd(C'')\ ]$

(4) if-then-else : $[\ ite(H)(C')(C'')\ ,\ fr(C') \cup fr(C'')\ ,\ bd(C') \cup bd(C'')\ ]$

(5) Recursive : $[\ \nabla X D\ ,\ fr(D) \setminus \{X\}\ ,\ bd(D) \cup \{X\}\ ]$

The sublanguage $\mathcal{R}ec_{1.01}$ (ignored till the 3rd section) is defined in the same way, with the added side conditions :
in (3) and (4), that $bd(C') \cap bd(C'') = \emptyset$;
in (5), that $X \notin bd(D)$.
Rather trivial induction on $C \in \mathcal{R}ec_{1+}$ shows that

$C \in \mathcal{R}ec_{1.01} \iff$

for all X, the consecutive pair ∇X appears at most once in the string C .

Of course, first one will have shown

$X \in bd(C) \iff$ the consecutive pair ∇X appears in the string C .

'Clearly' it is also true that

$X \in fr(C) \iff$

the substring $callX$ appears at least once not within the scope of a ∇X ;

but that requires a definition of *scope*, which isn't needed in the paper, so will be left as an exercise. Note that, for X to be a fully fledged member of $bd(C)$, the procedure variable X need not be actually "called" in any C-subcommand $\nabla X D$, just that such a subcommand should exist, with D possibly being rather uninteresting in that it contains no 'free' $callX$.

Now let $\underline{n}$ denote a function $\mathcal{P}cv \to \mathbf{N} \cup \{\infty\}$. **Intuitively**, this will be used to measure recursion depth, as follows. An '(input,output) pair' (s, s') of states will be in the set $\mathcal{M}_{\underline{n}}(C, \Theta)$ when and only when, implementing the command C with the machine initially in state s *can* (actually, *will*, by determinisiticality), assuming the computation converges, produce the state s', where the implementation 'allows' , for all X and each subcommand $\nabla X D$ of C, the latter recursion to 'cycle' at most "$\underline{n}(X)$" times [and where, from [**sem**], the component Θ gives the 'phantom semantics' for ("is the assignment of") the free occurrences of procedure variables in C]. This description plays no role whatsoever in the mathematics that follows, only in its motivation.

The really crucial part of the exact definition below is the middle case of phrase (5). It tells us what recursion depth really means here. There are several other possibilities for that definition. But none that I know will do what's needed in Section 3 of this paper, though several (e.g. making the depth bound the same for all procedure variables) can be used to give results analogous to the rest of this section.

Recall that Θ denotes a function from $\mathcal{P}cv$ to the set $\mathcal{B}sm$ of "basic semantic values", which is $2^{\mathcal{S}te \times \mathcal{S}te}$, the set of subsets of pairs of states. Also $\Theta_{X \mapsto \beta}$ agrees with Θ except for mapping X to β. Similarly $s_{y \mapsto k}$ differs from the state s on only one variable, a $1^{\underline{\text{st}}}$order variable y, and for natural numbers k (which form the only interpretation considered in this paper).

Definition. By induction on C, the five clauses just below define $\mathcal{M}_{\underline{n}}(C, \Theta)$:

(1) $\mathcal{M}_{\underline{n}}(y \mathrel{=\!\!\!\div} t, \Theta) \;:=\; \{(s, s_{y \mapsto s(t)}) \mid s \in \mathcal{S}te\}$ (independent of Θ and $\underline{n}$) .

(2) $\mathcal{M}_{\underline{n}}(callX, \Theta) \;:=\; \Theta(X)$ (independent of $\underline{n}$) .

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

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

(5) $\mathcal{M}_{\underline{n}}(\nabla XD,\Theta)$ is defined by (minimally transfinite) induction on $\underline{n}(X)$ via :

if $\underline{n}(X) = 0$, it is $\emptyset$;

if $0 < \underline{n}(X) < \infty$, it is $\mathcal{M}_{\underline{n}_{X-}}(D, \Theta_{X \mapsto \mathcal{M}_{\underline{n}_{X-}}(\nabla XD,\Theta)})$,
where $\underline{n}_{X-}$ agrees with $\underline{n}$ except that $\underline{n}_{X-}(X) := \underline{n}(X) - 1$;

if $\underline{n}(X) = \infty$, it is $\bigcup_{\underline{\ell}}\{\ \mathcal{M}_{\underline{\ell}}(\nabla XD,\Theta) \mid \underline{\ell}$ agrees with $\underline{n},$ except $\underline{\ell}(X) < \infty\ \}$.

The first six of the following seven results are proved by induction on C; and the final case of those inductions, where $C = \nabla XD$, is always proved by induction on $\underline{n}(X)$. These proofs need no other ideas, so parts of the proofs are suppressed. All details are given in **[dpth]**.

Note that $\subset$ means *non-strict* inclusion here.

Proposition 1.1 *If $\theta \sqsubset \Psi$* [meaning that, for all X, $\Theta(X) \subset \Psi(X)$], *then, for all $\underline{n}$ and C, we have*

$$\mathcal{M}_{\underline{n}}(C,\Theta) \;\subset\; \mathcal{M}_{\underline{n}}(C,\Psi)\ .$$

Proof. The required inclusion becomes $\Theta(X) \subset \Psi(X)$ when C is $callX$, and we get equality when C is an assignment command. When $C = (D;E)$,

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

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

To finish the structural induction and the proof, for $C = \nabla XD$ we proceed by induction on $\underline{n}(X)$. When 0, both sides are $\emptyset$. When $0 < \underline{n}(X) < \infty$,

$$\begin{aligned}\mathcal{M}_{\underline{n}}(C,\Theta) \;&=\; \mathcal{M}_{\underline{n}_{X-}}(D, \Theta_{X\mapsto \mathcal{M}_{\underline{n}_{X-}}(\nabla XD,\Theta)}) \\ &\subset\; \mathcal{M}_{\underline{n}_{X-}}(D, \Theta_{X\mapsto \mathcal{M}_{\underline{n}_{X-}}(\nabla XD,\Psi)}) \\ &\subset\; \mathcal{M}_{\underline{n}_{X-}}(D, \Psi_{X\mapsto \mathcal{M}_{\underline{n}_{X-}}(\nabla XD,\Psi)}) \;=\; \mathcal{M}_{\underline{n}}(C,\Psi)\,.\end{aligned}$$

The second inclusion uses only the structural induction, but the first uses both it and the induction on $\underline{n}(X)$. Finally, when $\underline{n}(X) = \infty$,

$$\begin{aligned}\mathcal{M}_{\underline{n}}(\nabla XD,\Theta) \;&=\; \bigcup\{\, \mathcal{M}_{\underline{\ell}}(\nabla XD,\Theta) \mid \underline{\ell} \text{ agrees with } \underline{n}, \text{ except } \underline{\ell}(X) < \infty \,\} \\ &\subset\; \bigcup\{\, \mathcal{M}_{\underline{\ell}}(\nabla XD,\Psi) \mid \text{ as above } \,\} \;=\; \mathcal{M}_{\underline{n}}(\nabla XD,\Psi)\,.\end{aligned}$$

So the proof is complete.

Proposition 1.2 *If* $\underline{n} \sqsubset \underline{m}$ [meaning that, for all X, $\underline{n}(X) \le \underline{m}(X)$], *then, for all* Θ *and* C, *we have*

$$\mathcal{M}_{\underline{n}}(C,\Theta) \;\subset\; \mathcal{M}_{\underline{m}}(C,\Theta)\,.$$

Proof. Here we'll suppress all cases of the structural induction except that when $C = \nabla XD$, where we proceed by induction on $\underline{n}(X)$. When 0, the left side is $\emptyset$. When $0 < \underline{n}(X) < \infty$,

$$\begin{aligned}\mathcal{M}_{\underline{n}}(C,\Theta) \;&=\; \mathcal{M}_{\underline{n}_{X-}}(D, \Theta_{X\mapsto \mathcal{M}_{\underline{n}_{X-}}(\nabla XD,\Theta)}) \\ &\subset\; \mathcal{M}_{\underline{n}_{X-}}(D, \Theta_{X\mapsto \mathcal{M}_{\underline{m}_{X-}}(\nabla XD,\Theta)}) \\ &\subset\; \mathcal{M}_{\underline{m}_{X-}}(D, \Theta_{X\mapsto \mathcal{M}_{\underline{m}_{X-}}(\nabla XD,\Theta)}) \;=\; \mathcal{M}_{\underline{m}}(C,\Theta)\,.\end{aligned}$$

The second inclusion uses the structural induction, and the first uses both **1.1** and the induction on $\underline{n}(X)$. Finally, when $\underline{n}(X) = \infty$ [and so is $\underline{m}(X)$],

$$\begin{aligned}\mathcal{M}_{\underline{n}}(\nabla XD,\Theta) \;&=\; \bigcup\{\, \mathcal{M}_{\underline{\ell}}(\nabla XD,\Theta) \mid \underline{\ell} \text{ agrees with } \underline{n}, \text{ except } \underline{\ell}(X) < \infty \,\} \\ &\subset\; \bigcup\{\, \mathcal{M}_{\underline{k}}(\nabla XD,\Theta) \mid \underline{k} \text{ agrees with } \underline{m}, \text{ except } \underline{k}(X) < \infty \,\} \;=\; \mathcal{M}_{\underline{m}}(\nabla XD,\Theta)\,.\end{aligned}$$

Proposition 1.3 *If C has no free occurrences of X* [i.e. $X \notin fr(C)$] *and $\Theta(Y) = \Psi(Y)$ for all $Y \neq X$, then we have*

$$\mathcal{M}_{\underline{n}}(C, \Theta) \;=\; \mathcal{M}_{\underline{n}}(C, \Psi)$$

for all $\underline{n}$.

It follows from an innocuous generalization of this (i.e. assume the sets $fr(C)$ and $\{X \mid \Theta(X) \neq \Psi(X)\}$ are disjoint) that Θ-dependence of $\mathcal{M}_{\underline{n}}(C, \Theta)$ is only on Θ's restriction to $fr(C)$. Its proof is another rather mechanical induction on structure which will be left to the reader.

Proposition 1.4 *If C has no bound occurrences of X [i.e. $X \notin bd(C)$] and $\underline{n}(Y) = \underline{m}(Y)$ for all $Y \neq X$, then we have*

$$\mathcal{M}_{\underline{n}}(C, \Theta) \;=\; \mathcal{M}_{\underline{m}}(C, \Theta)$$

for all Θ.

An easy generalization (i.e. assume $bd(C) \cap \{X \mid \underline{m}(X) \neq \underline{n}(X)\} = \emptyset$) implies that, for each C, the dependence of $\mathcal{M}_{\underline{n}}(C, \Theta)$ on $\underline{n}$ is only on its restriction to $bd(C)$.

Proof. The structural induction is again mechanical. Here is the final case, for $C = \nabla Y D$, where we proceed by induction on $\underline{n}(Y)$. Note that $Y \neq X$, so $\underline{n}(Y) = \underline{m}(Y)$. When 0, both sides are $\emptyset$. When $0 < \underline{n}(Y) < \infty$,

$$\begin{aligned}\mathcal{M}_{\underline{n}}(C, \Theta) \;&=\; \mathcal{M}_{\underline{n}_{Y-}}(D, \Theta_{Y \mapsto \mathcal{M}_{\underline{n}_{Y-}}(\nabla Y D, \Theta)}) \\ &=\; \mathcal{M}_{\underline{n}_{Y-}}(D, \Theta_{Y \mapsto \mathcal{M}_{\underline{m}_{Y-}}(\nabla Y D, \Theta)}) \\ &=\; \mathcal{M}_{\underline{m}_{Y-}}(D, \Theta_{Y \mapsto \mathcal{M}_{\underline{m}_{Y-}}(\nabla Y D, \Theta)}) \;=\; \mathcal{M}_{\underline{m}}(C, \Theta)\,.\end{aligned}$$

The second equality uses the structural induction, and the first uses the induction on $\underline{n}(Y)$. Finally, when $\underline{n}(Y) = \infty$ (and so is $\underline{m}(Y)$),

$$\begin{aligned}\mathcal{M}_{\underline{n}}(\nabla Y D, \Theta) \;&=\; \bigcup\{\, \mathcal{M}_{\underline{\ell}}(\nabla Y D, \Theta) \mid \underline{\ell} \text{ agrees with } \underline{n}, \text{ except } \underline{\ell}(Y) < \infty \,\} \\ &=\; \bigcup\{\, \mathcal{M}_{\underline{k}}(\nabla Y D, \Theta) \mid \underline{k} \text{ agrees with } \underline{m}, \text{ except } \underline{k}(Y) < \infty \,\} \;=\; \mathcal{M}_{\underline{m}}(\nabla Y D, \Theta)\,.\end{aligned}$$

Proposition 1.5 *The function $[\beta \mapsto \mathcal{M}_{\underline{n}}(C, \Theta_{Y \mapsto \beta}\,]\;:\; \mathcal{B}sm \to \mathcal{B}sm$ is continuous, for each $\underline{n}, C, \Theta$, and Y. More generally, if $\Theta_0 \sqsubset \Theta_1 \sqsubset \Theta_2 \sqsubset \cdots$, then we have that, for any $\underline{n}, C$ and Θ,*

$$\mathcal{M}_{\underline{n}}(C, \bigsqcup_i \Theta_i) \;=\; \bigcup_i \mathcal{M}_{\underline{n}}(C, \Theta_i)\,.$$

By definition, $(\bigsqcup_i \Theta_i)(X) := \bigcup_i(\Theta_i(X))$.

Proof. That last definition proves it when $C = callX$. The result is clear for assignment commands, where $\mathcal{M}_{\underline{n}}(C, \Theta)$ is independent of Θ.

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

$$\mathcal{M}_{\underline{n}}((C'; C''), \bigsqcup_i \Theta_i) \;=\; \mathcal{M}_{\underline{n}}(C'', \bigsqcup_i \Theta_i) \circ \mathcal{M}_{\underline{n}}(C', \bigsqcup_i \Theta_i) \;=$$

$$\bigcup_i \mathcal{M}_{\underline{n}}(C'', \Theta_i) \circ \bigcup_i \mathcal{M}_{\underline{n}}(C', \Theta_i) \;=$$

$$\bigcup_i (\mathcal{M}_{\underline{n}}(C'', \Theta_i) \circ \mathcal{M}(C', \Theta_i)) \;=\; \bigcup_i \mathcal{M}_{\underline{n}}((C'; C''), \Theta_i) \; .$$

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

$$(s, s') \in \mathcal{M}_{\underline{n}}(ite(H)(C')(C''), \bigsqcup_i \Theta_i) \iff$$

$$[H\mathbf{tt}@s \text{ and } (s, s') \in \mathcal{M}_{\underline{n}}(C', \bigsqcup_i \Theta_i) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s, s') \in \mathcal{M}_{\underline{n}}(C'', \bigsqcup_i \Theta_i)] \iff$$

$$[H\mathbf{tt}@s \text{ and } (s, s') \in \bigcup_i \mathcal{M}_{\underline{n}}(C', \Theta_i) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s, s') \in \bigcup_i \mathcal{M}_{\underline{n}}(C'', \Theta_i)] \iff$$

$$\exists i \, [H\mathbf{tt}@s \text{ and } (s, s') \in \mathcal{M}_{\underline{n}}(C', \Theta_i) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s, s') \in \mathcal{M}_{\underline{n}}(C'', \Theta_i)] \iff$$

$$\exists i \, [(s, s') \in \mathcal{M}_{\underline{n}}(ite(H)(C')(C''), \Theta_i)] \; .$$

When $C = \nabla X D$, we proceed by induction on $\underline{n}(X)$. First note that

$$(\bigsqcup_i \Theta_i)_{X \mapsto \cup_i \beta_i} \;=\; \bigsqcup_i ((\Theta_i)_{X \mapsto \beta_i}) \; ,$$

since both map X to $\cup_i \beta_i$, and any other Y to $\cup_i \Theta_i(Y)$. Now when $\underline{n}(X)$ is 0, both sides of the identity to be proved are $\emptyset$. When $0 < \underline{n}(X) < \infty$, we get

$$\mathcal{M}_{\underline{n}}(\nabla XD, \bigsqcup_i \Theta_i) \;=\; \mathcal{M}_{\underline{n}_{X-}}(D, (\bigsqcup_i \Theta_i)_{X \mapsto \mathcal{M}_{\underline{n}_{X-}}(\nabla XD, \bigsqcup_i \Theta_i)})$$

$$= \; \mathcal{M}_{\underline{n}_{X-}}(D, (\bigsqcup_i \Theta_i)_{X \mapsto \cup_i \mathcal{M}_{\underline{n}_{X-}}(\nabla XD, \Theta_i)})$$

$$= \; \mathcal{M}_{\underline{n}_{X-}}(D, \bigsqcup_i (\Theta_i)_{X \mapsto \mathcal{M}_{\underline{n}_{X-}}(\nabla XD, \Theta_i)})$$

$$= \; \bigcup_i \mathcal{M}_{\underline{n}_{X-}}(D, (\Theta_i)_{X \mapsto \mathcal{M}_{\underline{n}_{X-}}(\nabla XD, \Theta_i)}) \;=\; \bigcup_i \mathcal{M}_{\underline{n}}(\nabla XD, \Theta_i) \; .$$

Successively, the equalities come from : definition; induction on $\underline{n}(X)$; the display just above this big one; induction on C; and definition. Note that the correct 'square-inclusions' hold for the penultimate equality, using **1.1**. Finally, for $\underline{n}(X) = \infty$, we get

$$\mathcal{M}_{\underline{n}}(\nabla XD, \bigsqcup_i \Theta_i) \;=\; \bigcup_{\underline{\ell}} \{ \mathcal{M}_{\underline{\ell}}(\nabla XD, \bigsqcup_i \Theta_i) \mid \underline{\ell} \text{ agrees with } \underline{n}, \text{ except } \underline{\ell}(X) < \infty \}$$

$$= \; \bigcup_{\underline{\ell}} \{ \bigcup_i \mathcal{M}_{\underline{\ell}}(\nabla XD, \Theta_i) \mid \underline{\ell} \text{ agrees with } \underline{n}, \text{ except } \underline{\ell}(X) < \infty \} \;=\; \bigcup_i \mathcal{M}_{\underline{n}}(\nabla XD, \Theta_i) \, ,$$

where we unionize in the other order for the last step.

Proposition 1.6 *If $\underline{\infty}$ means $\underline{\infty}(X) = \infty$ for all X, then, for all C, we have*

$$\mathcal{M}_{\underline{\infty}}(C, \Theta) \;=\; \bigcup_{\underline{n}} \{ \mathcal{M}_{\underline{n}}(C, \Theta) \mid \underline{n}(X) < \infty \text{ for all } X \}) \; .$$

More generally, for any $\underline{k}$,

$$\mathcal{M}_{\underline{k}}(C, \Theta) \;=\; \bigcup_{\underline{n}} \{ \; \mathcal{M}_{\underline{n}}(C, \Theta) \mid \text{ for all } X, \; \underline{n}(X) < \infty \text{ and } \; \underline{n}(X) \leq \underline{k}(X) \; \} \; .$$

Proof. Once again, because of independence from $\underline{n}$, the cases of C being atomic are trivial. Let "$COND(\underline{n})$" be short for the phrase
" for all X, $\underline{n}(X) < \infty$ and $\underline{n}(X) \leq \underline{k}(X)$".
When $C = (C'; C'')$, we get

$$\mathcal{M}_{\underline{k}}(C,\Theta) \;:=\; \mathcal{M}_{\underline{k}}(C'',\Theta)\circ\mathcal{M}_{\underline{k}}(C',\Theta)$$

$$= \;(\bigcup_{\underline{a}}\{\mathcal{M}_{\underline{a}}(C'',\Theta)\mid COND(\underline{a})\;\})\circ(\bigcup_{\underline{b}}\{\mathcal{M}_{\underline{b}}(C',\Theta)\mid COND(\underline{b})\;\})$$

$$= \;\bigcup_{\underline{a},\underline{b}}\{\mathcal{M}_{\underline{a}}(C'',\Theta)\circ\mathcal{M}_{\underline{b}}(C',\Theta)\mid COND(\underline{a}) \text{ and } COND(\underline{b})\}$$

$$= \;\bigcup_{\underline{n}}\{\mathcal{M}_{\underline{n}}(C'',\Theta)\circ\mathcal{M}_{\underline{n}}(C',\Theta)\mid COND(\underline{n})\;\}$$

$$= \;\bigcup_{\underline{n}}\{\mathcal{M}_{\underline{n}}(C,\Theta)\mid COND(\underline{n})\;\}\;.$$

When $C = ite(H)(C')(C'')$, we get

$$(s,s')\in\mathcal{M}_{\underline{k}}(ite(H)(C')(C''),\Theta) \iff$$

$$[H\mathbf{tt}@s \text{ and } (s,s')\in\mathcal{M}_{\underline{k}}(C',\Theta) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s,s')\in\mathcal{M}_{\underline{k}}(C'',\Theta)] \iff$$

$$[H\mathbf{tt}@s \text{ and } (s,s')\in\bigcup_{\underline{n}}\{\mathcal{M}_{\underline{n}}(C',\Theta)\mid COND(\underline{n})\;\}) \quad \text{or}$$

$$H\mathbf{ff}@s \text{ and } (s,s')\in\bigcup_{\underline{n}}\{\mathcal{M}_{\underline{n}}(C'',\Theta)\mid COND(\underline{n})\;\})] \iff$$

$\exists\underline{n}$ satisfying $COND(\underline{n})$ with

$$[H\mathbf{tt}@s \text{ and } (s,s')\in\mathcal{M}_{\underline{n}}(C',\Theta) \quad \text{or} \quad H\mathbf{ff}@s \text{ and } (s,s')\in\mathcal{M}_{\underline{n}}(C'',\Theta)] \iff$$

$$\exists\underline{n} \text{ satisfying } COND(\underline{n}) \text{ with } (s,s')\in\mathcal{M}_{\underline{n}}(ite(H)(C')(C''),\Theta)\;,$$

as required.

And finally, when $C = \nabla XD$, we proceed by induction on $\underline{k}(X)$. When it is 0, each $\underline{n}(X)$ on the right-hand side is also 0, and the result reduces to the empty set being a union of empty sets. When $0 < \underline{k}(X) < \infty$, we get, letting $COND_-$ be the $COND$ corresponding to $\underline{k}_{X-}$ rather than $\underline{k}$,

$$\mathcal{M}_{\underline{k}}(\nabla XD,\Theta) \;=\; \mathcal{M}_{\underline{k}_{X-}}(D,\Theta_{X\mapsto\mathcal{M}_{\underline{k}_{X-}}(\nabla XD,\Theta)})$$

$$=\; \mathcal{M}_{\underline{k}_{X-}}(D,\Theta_{X\mapsto\bigcup_{\underline{a}}\{\mathcal{M}_{\underline{a}}(\nabla XD,\Theta)\;\mid\; COND_-(\underline{a})\;\}})$$

$$=\; \bigcup_{\underline{a}}\{\; \mathcal{M}_{\underline{k}_{X-}}(D,\Theta_{X\mapsto\mathcal{M}_{\underline{a}}(\nabla XD,\Theta)}) \mid COND_-(\underline{a})\;\}))$$

$$=\; \bigcup_{\underline{a}}\bigcup_{\underline{b}}\{\; \mathcal{M}_{\underline{b}}(D,\Theta_{X\mapsto\mathcal{M}_{\underline{a}}(\nabla XD,\Theta)}) \mid COND_-(\underline{a}) \text{ and } COND_-(\underline{b})\;\}$$

$$=\; \bigcup_{\underline{e}}\{\; \mathcal{M}_{\underline{e}}(D,\Theta_{X\mapsto\mathcal{M}_{\underline{e}}(\nabla XD,\Theta)} \mid COND_-(\underline{e})\;\}$$

$$=\; \bigcup_{\underline{n}}\{\; \mathcal{M}_{\underline{n}}(\nabla XD,\Theta) \mid COND(\underline{n}) \text{ and } \underline{n}(X)\neq 0\}\;.$$

$$=\; \bigcup_{\underline{n}}\{\; \mathcal{M}_{\underline{n}}(\nabla XD,\Theta) \mid COND(\underline{n})\;\}\;.$$

The successive equalities come from : definition ; induction on $\underline{k}(X)$; **1.5** ; induction on C ; easy argument using **1.2** ; definition and taking $\underline{e}$ to be $\underline{n}_{X-}$; add a $\emptyset$ to the union. Finally, when $\underline{k}(X)=\infty$, we get, letting $COND_{\underline{\ell}}$ be the $COND$ corresponding to $\underline{\ell}$ rather than $\underline{k}$,

$$\mathcal{M}_{\underline{k}}(\nabla XD,\Theta) \;=\; \bigcup_{\underline{\ell}}\{\; \mathcal{M}_{\underline{\ell}}(\nabla XD,\Theta) \mid \underline{\ell} \text{ agrees with } \underline{k},\ \text{except } \underline{\ell}(X)<\infty\;\}$$

$$=\; \bigcup_{\underline{\ell}}\bigcup_{\underline{n}}\{\; \mathcal{M}_{\underline{n}}(\nabla XD,\Theta) \mid COND_{\underline{\ell}}(\underline{n}) \text{ holds and } \underline{\ell} \text{ agrees with } \underline{k},\ \text{except } \underline{\ell}(X)<\infty\;\}$$

$$=\; \bigcup_{\underline{n}}\{\; \mathcal{M}_{\underline{n}}(\nabla XD,\Theta) \mid COND(\underline{n}) \text{ holds}\;\}\;,$$

as it is easy to verify that $\underline{n}$ satisfies $COND(\underline{n})$ [which is $COND_{\underline{k}}(\underline{n})$ and, in this case, simply says that $\underline{n}(X)<\infty$] if and only if there is an $\underline{\ell}$ agreeing with $\underline{k}$, except $\underline{\ell}(X)<\infty$, and such that $COND_{\underline{\ell}}(\underline{n})$ holds .

Theorem 1.7 *The semantics $\mathcal{M}_{\underline{\infty}}$ reproduces fixpoint semantics* (as defined in Section 1+ of [**sem**], for example).

We'll just use $\mathcal{M}$ for the overall semantics after the following proof, dropping the subscript $\underline{\infty}$. The definition of fixpoint semantics is fairly explicit in the following proof, so the reader need not go back to [**sem**].

Proof. The two semantics are defined in the same way on all but '∇-commands', so we need only deal with that case.

To show that $\mathcal{M}_{\underline{\infty}}(\nabla Y D, \Theta)$ is a fixed point of $[\beta \mapsto \mathcal{M}_{\underline{\infty}}(D, \Theta_{Y \mapsto \beta})\ ]$, temporarily abbreviate $\mathcal{M}_{\underline{\underline{k}}}$ to $\mathcal{N}_k$ when $\underline{\underline{k}}(X) = \infty$ for all X except that $\underline{\underline{k}}(Y) = k \in \mathbf{N}$. [So the '$k$' within '$\underline{\underline{k}}$ actually means something here—it doesn't in the 'single-underlined $\underline{k}$'.] Calculate as follows.

$$\begin{aligned}
\mathcal{M}_{\underline{\infty}}(\nabla Y D, \Theta) \;&=\; \bigcup_{k<\infty} \mathcal{N}_k(\nabla Y D, \Theta) \;=\; \bigcup_{k<\infty} \mathcal{N}_{k+1}(\nabla Y D, \Theta) \\
&= \bigcup_{k<\infty} \mathcal{N}_k(D, \Theta_{Y \mapsto \mathcal{N}_k(\nabla Y D, \Theta)}) \;=\; \bigcup_{a,b<\infty} \mathcal{N}_a(D, \Theta_{Y \mapsto \mathcal{N}_b(\nabla Y D, \Theta)}) \\
&= \bigcup_{a<\infty} \mathcal{N}_a(D, \Theta_{Y \mapsto \bigcup_{b<\infty} \mathcal{N}_b(\nabla Y D, \Theta)}) \;=\; \bigcup_{a<\infty} \mathcal{N}_a(D, \Theta_{Y \mapsto \mathcal{M}_{\underline{\infty}}(\nabla Y D, \Theta)}) \\
&= \mathcal{M}_{\underline{\infty}}(D, \Theta_{Y \mapsto \mathcal{M}_{\underline{\infty}}(\nabla Y D, \Theta)})\ ,
\end{aligned}$$

as required. The equalities, respectively, use : $(**)$ that is, (**1.2** and **1.6**) ; **1.2** ; definition ; **1.2** and elementary considerations ; **1.5** ; $(**)$ again ; and $(**)$ yet again.

To show that $\mathcal{M}_{\underline{\infty}}(\nabla Y D, \Theta)$ is contained in every other fixed point, we show (superficially more generally) that

$$\mathcal{M}_{\underline{\infty}}(D, \Theta_{Y \mapsto \beta}) \subset \beta \implies \mathcal{M}_{\underline{\infty}}(\nabla Y D, \Theta) \subset \beta\ .$$

It suffices to deduce, by induction on k, that $\mathcal{N}_k(\nabla Y D, \Theta) \subset \beta$ for all k. When $k = 0$, on the left is $\emptyset$ by definition. For the inductive step,

$$\mathcal{N}_{k+1}(\nabla Y D, \Theta) \;=\; \mathcal{N}_k(D, \Theta_{Y \mapsto \mathcal{N}_k(\nabla Y D, \Theta)}) \;\subset\; \mathcal{N}_k(D, \Theta_{Y \mapsto \beta}) \;\subset\; \beta\ ,$$

as required, respectively by definition; by induction and by continuity (**1.5**) which implies monotony; and finally by the left-hand side of the penultimate display and **1.2**.

2. **'Total' unfolding—a variant on Muskens' Theorem.**

Motivated by the definition of $\mathcal{M}_{\underline{n}}$ in the previous section, but only for those $\underline{n}$ with $\underline{n}(X) < \infty$ for all X, define a command $C^{\underline{n}}$ by induction on commands C as follows, where $gigabug := \nabla Y_0 call Y_0$ for some particular Y_0, a command which 'always loops', i.e. its semantics set, $\mathcal{M}(gigabug, \Theta)$ is empty for all Θ . Recall that $\underline{n}_{X-}$ agrees with $\underline{n}$, except its value on X is one less.

Definition. If C is atomic, $C^{\underline{n}} := C$;
$(C'; C'')^{\underline{n}} := (C'^{\underline{n}} ; C''^{\underline{n}})$;
$ite(H)(C')(C'')^{\underline{n}} := ite(H)(C'^{\underline{n}})(C''^{\underline{n}})$;
when $C = \nabla XD$, proceed by induction on $\underline{n}(X)$:
when $\underline{n}(X) = 0$, let $(\nabla XD)^{\underline{n}} := gigabug$;
when $0 < \underline{n}(X) < \infty$, let $(\nabla XD)^{\underline{n}} := |D^{\underline{n}_{X-}}|^{X \mapsto (\nabla XD)^{\underline{n}_{X-}}}$.

The last line uses substitution notation from [**sem**]. We shall get somewhat more involved with this in the next section. But the definition and results in [**sem**] are sufficient for this short section.

Proposition 2.1. *If being " ∇-free" for a command means that any occurrence of " ∇" is immediately followed by "$Y_0 call Y_0$"* (i.e. the only 'recursions' in the command, if any, are *gigabug*s), *then $C^{\underline{n}}$ is always ∇-free.*

Proof. This is immediate by induction: firstly on commands; then on $\underline{n}(X)$, when the command is ∇XD.

Theorem 2.2. *For any C and $\underline{n}$ with $\underline{n}(X) < \infty$ for all X, we have*

$$\mathcal{M}_{\underline{n}}(C, \Theta) \;=\; \mathcal{M}(C^{\underline{n}}, \Theta) \; .$$

Proof. All but the final case of the induction on C are very straightforward from definitions. When the command is ∇XD, proceed by induction on $\underline{n}(X)$, with the initial case immediate. When $0 < \underline{n}(X) < \infty$, we have

$$\mathcal{M}((\nabla XD)^{\underline{n}}, \Theta) \;=\; \mathcal{M}(|D^{\underline{n}_{X-}}|^{X \mapsto (\nabla XD)^{\underline{n}_{X-}}}, \Theta)$$

$$= \; \mathcal{M}(D^{\underline{n}_{X-}}, \Theta_{X \mapsto \mathcal{M}((\nabla XD)^{\underline{n}_{X-}}, \Theta)}) \;=\; \mathcal{M}_{\underline{n}_{X-}}(D, \Theta_{X \mapsto \mathcal{M}((\nabla XD)^{\underline{n}_{X-}}, \Theta)})$$

$$= \; \mathcal{M}_{\underline{n}_{X-}}(D, \Theta_{X \mapsto \mathcal{M}_{\underline{n}_{X-}}(\nabla XD, \Theta)}) \;=\; \mathcal{M}_{\underline{n}}(\nabla XD, \Theta) \; ,$$

as required. The equalities, respectively, are justified by : definition ; substitution property for $\mathcal{M}$—see $\mathbf{1.2}_+$ in [**sem**], somewhat less difficult than what is to come in the next section for $\mathcal{M}_{\underline{n}}$; the induction on commands ; the induction on $\underline{n}(X)$; and finally, by definition.

Corollary 2.3. (to **2.2**, **1.6** and **1.7**) *For any C, we have*

$$\mathcal{M}(C,\Theta) \;=\; \bigcup_{\underline{n}}\{\mathcal{M}(C^{\underline{n}},\Theta) \mid \underline{n}(X) < \infty \text{ for all } X\ \}) \ .$$

The proof is immediate. So, for each C, we have produced a collection of ∇-free commands in a canonical inductive way, which might be called its *total unfoldings*, so that the semantics of C is the union of those of its total unfoldings. A quite different way of doing the same thing is the main result in [**Musk**]. His unfoldings are different, and obtained in an entirely different way, using 2nd order type-logic to specify the semantics. Some comparisons between this work and Muskens' appears in the final section of [**dpth**]. Generalizing what is here to *mutual* recursion, which Muskens deals with ab initio, is presumably worth the effort.

3. **Substitution formula.**

We wish to prove (a correct analogue for) the following (ill-defined and incorrect, but intuitively appealing) formula for the 'depth-semantics' of the command, denoted here as $|C|^{X\to E}$, which is what results from substituting the command E for each free occurrence of $callX$ in the command C :

$$\mathcal{M}_{\underline{n}}(|C|^{X\to E},\Theta) \stackrel{?}{=} \mathcal{M}_{\underline{n}}(C,\Theta_{X\mapsto\mathcal{M}_{\underline{n}}(E,\Theta)}) \ .$$

Besides wanting to check carefully that the depths are correctly correlated, the only really problematic thing here is that normally there is some 're-naming of bound variables' built into the definition of substitution, and usually (elsewhere) there is no need to be explicit about the names of the new bound variables. But, as yet here, no connection exists between the value of the depth measure $\underline{n}$ on such a bound variable and on any of its new 'versions', after re-naming. So that's basically the only thing which needs to be fixed. Aesthetically, the details leave something to be desired, but this topic always seems to have that failing.

As explained roughly in the introduction, we will work strictly in $\mathcal{R}ec_{1.01}$; i.e. commands contain no repeated bound variable. That will constitute one extra requirement for re-naming, convenient, harmless and possibly dispensible with. The official notation for substitution will have an extra subscript 'h' which carries along re-naming information which ensures that substituting for a free variable which occurs more than once will not result in repeating bound variables from what is substituted, when substituting in for these different occurrences. But the earlier 'non-capture of free variables' motivation for re-naming will also be operative in the definition of $|C|_h^{X\to E}$, but will not be part of the notation, so there will still be that indeterminacy in the meaning of such a notation. In any case, the basic result, **3.5** below, will be the formula above, except with (1) the extra subscript h; (2) the proviso that the formula only applies to commands in $\mathcal{R}ec_{1.01}$; (3) $bd(C)\cap bd(E)=\emptyset$, helping to ensure that $|C|_h^{X\to E}$ *is* in $\mathcal{R}ec_{1.01}$; and, most important, (4) the formula only being asserted for those $\underline{n}$ which are constant on each orbit consisting of all procedure variables related under the equivalence relation generated by re-naming.

Definition. The *disjoint union*, $A\sqcup B$, of two sets A and B will not be defined explicitly, but rather characterized by some of its properties. By

definition, $\sqcup$ is a binary operator on sets that comes with two additional functions,

$$\alpha : A \to A \sqcup B \quad \text{and} \quad \beta : B \to A \sqcup B \ ,$$

such that for any set C, there is a bijection between the set of functions from $A \sqcup B$ to C, and the set of pairs of functions, from A to C and from B to C, a bijection defined by the obvious composition using the fixed functions α and β.

This is the category-theoretic notion of *coproduct*, applied to the category of sets, and is all we'll need here.

Definition. Denoting by $foc(X, C)$ *the set of free occurrences of the procedure variable* X *in the command* C [again more handily characterized abstractly, rather than specified explicitly—no sane mathematician would doubt its existence] gives a collection of sets with the following properties [which are far more convenient and perspicuous for our purposes than any explicit choices for $foc(X, C)$] :

(i) $foc(X, y \leftarrow\!\!\div t)$ is empty ;
(ii) $foc(X, callX)$ is a singleton, but $foc(X, callY)$ is empty for $Y \neq X$;
(iii) $foc(X, (C'; C'')) \ = \ foc(X, C') \sqcup foc(X, C'')$;
(iv) $foc(X, ite(H)(C')(C'')) \ = \ foc(X, C') \sqcup foc(X, C'')$;
(v) $foc(X, \nabla X D) \ = \ \emptyset$, whereas $foc(X, \nabla Y D) = foc(X, D)$ if $Y \neq X$.

Proposition 3.1. *The set* $foc(X, C)$ *is non-empty if and only if* X *is in* $fr(C)$.

The proof, by induction on the structure of C, is very easy.

Definition. For a pair of procedure variables Y and Z, and D in $\mathcal{R}ec_{1.01}$, define $|D|^{Y \mapsto callZ} \in \mathcal{R}ec_{1.01}$ below by induction on *the length of* the string D. [This is a special case of the definition after next, with D here in place of C there, and with Z in place of E ; and ignoring h there, because its domain here is empty, so it is unique, simplifying considerably that complicated later definition. This definition is a needed stopgap, occurring as part of the next two definitions, the second of which generalizes it. Also we need to prove **3.2** below, and use it in establishing properties of the 're-naming' definition coming right after it.]

(1)′ $\quad |y \leftarrow\!\!\div t|^{Y \mapsto callZ} \ := \ y \leftarrow\!\!\div t$;

(2) $|callY|^{Y\mapsto callZ} := callZ$;
(2)′ When $X \neq Y$, $|callX|^{Y\mapsto callZ} := callX$;

(3) $|(C';C'')|^{Y\mapsto callZ} := (|C'|^{Y\mapsto callZ} ; |C''|^{Y\mapsto callZ})$;

(4) $|ite(H)(C')(C'')|^{Y\mapsto callZ} := ite(H)((|C'|^{Y\mapsto callZ})(|C''|^{Y\mapsto callZ})$;

(5) $|\nabla YC|^{Y\mapsto callZ} := \nabla YC$;
(5)′ When $X \neq Y$, $|\nabla XC|^{Y\mapsto callZ} := \nabla X'|C'|^{Y\mapsto callZ}$, where, for some choice of $X' \notin \{X\} \cup \{Y\} \cup \{Z\} \cup C$, we take $C' := |C|^{X\mapsto callX'}$.
As it happens, only the case $X = Z$ needs this last elaborate phrase, but, as written, it will be easier to see it as a special case of the even more elaborate definition just before **3.4**. The reason for not using induction on D itself is to make this last definition work, i.e. the $|C'|^{Y\mapsto callZ}$-part. The lengths of C and C' are both two less than that of ∇XC, of course.

Lemma 3.2. *For any $\underline{k}, D, \Psi, Y$ and α, and where Z is chosen outside $\{Y\} \cup D$, we have*

$$\mathcal{M}_{\underline{k}}(|D|^{Y\mapsto callZ}, \Psi) = \mathcal{M}_{\underline{k}}(D, \Psi_{Y\mapsto\Psi(Z)}) .$$

Note that this lemma is a particular case of the formula we are seeking (and given precisely in **3.5**), but it seems logically simpler to give a completely separate proof, even if we are duplicating a few things from **3.5**'s proof.

Proof. Proceed by induction on the *length* of the string D.

All but the last case are straightforward.

When $D = \nabla YC$, we have $|D|^{Y\mapsto callZ} = D$, so the left-hand side is $\mathcal{M}_{\underline{k}}(D, \Psi)$. But $Y \notin fr(\nabla YC)$, so by **1.3**, the right-hand side is also $\mathcal{M}_{\underline{k}}(\nabla YC, \Psi)$.

When $D = \nabla XC$ where $X \neq Y$, choose any $X' \notin \{X\} \cup \{Y\} \cup \{Z\} \cup D$, let $C' := |C|^{X\mapsto callX'}$. So $\nabla X'|C'|^{Y\mapsto callZ}$ is a typical choice for $|D|^{Y\mapsto callZ}$, and we must prove

$$\mathcal{M}_{\underline{k}}(\nabla X'|C'|^{Y\mapsto callZ}, \Psi) = \mathcal{M}_{\underline{k}}(\nabla XC, \Psi_{Y\mapsto\Psi(Z)}) .$$

Proceed by induction on $\underline{k}(X)$. We have $\underline{k}(X) = \underline{k}(X')$. Note that $Z \neq Y$, since $Z \notin \{Y\}$; and $Z \neq X$, since X is in $bd(D)$ but Z isn't.

When $\underline{k}(X) = 0$, both sides are the empty set.

When $0 < \underline{k}(X) < \infty$, let $\underline{\ell} := (\underline{k}_{X'-})_{X-} = (\underline{k}_{X-})_{X'-}$. The left-hand side is

$$\mathcal{M}_{\underline{k}_{X'-}}(|C'|^{Y\mapsto callZ}, \Psi_{X'\mapsto \mathcal{M}_{\underline{k}_{X'-}}(\nabla X'|C'|^{Y\mapsto callZ},\Psi)})$$

$$= \mathcal{M}_{\underline{\ell}}(|C'|^{Y\mapsto callZ}, \Psi_{X'\mapsto \mathcal{M}_{\underline{\ell}}(\nabla X'|C'|^{Y\mapsto callZ},\Psi)})$$

$$= \mathcal{M}_{\underline{\ell}}(|C'|^{Y\mapsto callZ}, \Psi_{X'\mapsto \mathcal{M}_{\underline{\ell}}(\nabla XC,\Psi_{Y\mapsto\Psi(Z)})})$$

$$= \mathcal{M}_{\underline{\ell}}(C', (\Psi_{X'\mapsto \mathcal{M}_{\underline{\ell}}(\nabla XC,\Psi_{Y\mapsto\Psi(Z)})})_{Y\mapsto\Psi(Z)})$$

$$= \mathcal{M}_{\underline{\ell}}(C', (\Psi_{Y\mapsto\Psi(Z)})_{X'\mapsto \mathcal{M}_{\underline{\ell}}(\nabla XC,\Psi_{Y\mapsto\Psi(Z)})})$$

$$= \mathcal{M}_{\underline{\ell}}(C, (\Psi_{Y\mapsto\Psi(Z)})_{X\mapsto \mathcal{M}_{\underline{\ell}}(\nabla XC,\Psi_{Y\mapsto\Psi(Z)})}) \; [= \gamma, \text{say}] \; .$$

The successive equalities are justified as follows : definition of $\mathcal{M}$; **1.4** using that $X \notin bd(|C'|^{Y\mapsto callZ}) \cup bd(\nabla X'(|C'|^{Y\mapsto callZ})$, which follows because $X \in bd(C')$ [which implies $X \in bd(C)$] would contradict the fact that ∇XC is in $\mathcal{R}ec_{1.01}$; the inductive hypothesis on $\underline{k}(X)$; the inductive hypothesis on the length of D ; fairly obvious, since $X' \neq Y$; and again the inductive hypothesis on the length of D, since $C' := |C|^{X\mapsto callX'}$ and $X' \notin C$; respectively.

On the other hand, the right-hand side, by definition of $\mathcal{M}$, is

$$\mathcal{M}_{\underline{k}_{X-}}(C, (\Psi_{Y\mapsto\Psi(Z)})_{X\mapsto \mathcal{M}_{\underline{k}_{X-}}(\nabla XC,\Psi_{Y\mapsto\Psi(Z)})}) \; ,$$

which reduces to γ, as required, by changing $\underline{k}_{X-}$ to $\underline{\ell}$ in both places using **1.4** since X' is in neither $bd(\nabla XC)$ nor in $bd(C)$.

When $\underline{k}(X) = \infty$, the result is immediate from the previous cases and the definition of $\mathcal{M}$ in this case as a union.

Let f be any bijection from $\mathcal{P}cv$ to itself. Below is a formal inductive definition of $]C[_f \in \mathcal{R}ec_{1+}$, which is supposed to denote the result of re-naming, within $C \in \mathcal{R}ec_{1+}$, each bound occurrence of Y to the 'new' variable $f(Y)$, for all $Y \in bd(C)$. [That is, the unique (if any) bound occurrence $\nabla Y \cdots callY \cdots$ in C is replaced by $\nabla f(Y) \cdots callf(Y) \cdots$.] We shall only define this when

$$[bd(C) \cup fr(C)] \cap f(bd(C)) = \emptyset \; ;$$

that is, none of the new bound variables already occurs in C, bound or free.

Definition.

(1) $[)y \leftarrow\!\!\div t(]_f \ := \ y \leftarrow\!\!\div t$;

(2) $[)callY(]_f \ := \ callY$;

(3) $[)(C';C'')(]_f \ := \ ([)C'(]_f \ ; \ [)C''(]_f)$;

(4) $[)ite(H)(C')(C'')(]_f \ := \ ite(H)(([)C'(]_f)([)C''(]_f)$;

(5) $[)\nabla X D(]_f \ := \ \nabla f(X)|[)D(]_f|^{X \mapsto callf(X)}$.

Here we use the previous definition for $|-|^{X \mapsto callf(X)}$.

Immediately from the definition, $C \in \mathcal{R}ec_{1.01} \implies [)C(]_f \in \mathcal{R}ec_{1.01}$

The following is easily proved by induction : *If f and f' agree on $bd(C)$, then $[)C(]_f = [)C(]_{f'}$.*

Applications.

(i) Given injective $g : bd(C) \to \mathcal{P}cv$ whose image is disjoint from $bd(C)$, extend it arbitrarily (bijectively) to f and define $[)C(]_g := [)C(]_f$.

(ii) By taking f' as the identity map, we get $[)C(]_f = C$ if $bd(C) = \emptyset$.

The formula $bd([)C(]_f) = f(bd(C))$ is also proved by induction, using $f(A \cup B) = f(A) \cup f(B)$ for the inductive cases of (3) and (4).

The most crucial property is again proved by an induction on commands E :

Proposition 3.3. *For f such that $[)E(]_f$ is defined, we have*

$$\mathcal{M}_{\underline{n}}([)E(]_f, \Theta) \ = \ \mathcal{M}_{\underline{n}}(E, \Theta) \ ,$$

as long as $\underline{n}$ takes the same values on each variable and its re-naming, i.e. $\underline{n}(X) = \underline{n}(f(X))$ for all $X \in bd(E)$.

A detailed proof may be found in **[dpth]**.

Definition. Assume given C, D and E in $\mathcal{R}ec_{1.01}$, and $X \in \mathcal{P}cv$. For each injective function

$$h \ : \ bd(E) \times foc(X, C) \to \mathcal{P}cv$$

whose image is disjoint from $bd(E)$, define $|C|_h^{X\mapsto E}$ by induction on C as follows : [The idea is that if γ is a free occurrence of X in C, that occurrence is to be replaced by E'', say, which is the result of first producing E' by changing, for each procedure variable Y, the unique bound occurrence $\nabla Y\cdots callY\cdots$ (if any) in E to $\nabla h(Y,\gamma)\cdots callh(Y,\gamma)\cdots$. So the E' produced for the different occurrences of X in C continue, as with E, to have all their bound variables distinct, but also entirely disjoint from one another, as we vary the free occurrence of $callX$. Then E'' is obtained from E' by changing, for each procedure variable Y, the unique bound occurrence $\nabla Y\cdots callY\cdots$ (if any) in E' to $\nabla Z\cdots callZ\cdots$ for Z's chosen different from each other and from every procedure variable within a million miles. This second step is the one which makes sure that none of the free variables in E get captured, so to speak, by ∇'s in C. It is the one where we don't need to get more specific about the choices of Z for each Y, at least not in the notation. But it should be clear by now that an inductive definition is absolutely necessary, so as to have a chance of giving (mechanical, but somewhat tortuous) proofs of properties.] The definition before **3.2**, as noted there, does agree with this definition, it being the considerably simpler case of this definition when E happens to be a procedure variable.

(1) $\quad |y \leftarrow\!\!\!\!\cdot\, t|_h^{X\mapsto E} \;:=\; y \leftarrow\!\!\!\!\cdot\, t$;

(2) $\quad$ for $Y \neq X$, $\; |callY|_h^{X\mapsto E} \;:=\; callY$;

whereas, $|callX|_h^{X\mapsto E} \;:=\; [\,)E(\,]_{Y\mapsto h[Y,elt]}$, where elt is the unique element in $foc(X, callX)$, and noting that $bd([\,)E(\,]_g) = g(bd(E))$;

(3) $\quad |(C';C'')|_h^{X\mapsto E} \;:=\; (|C'|_{h\mathrm{restr}}^{X\mapsto E} \;;\; |C''|_{h\mathrm{restr}}^{X\mapsto E})$,
where, in each case, "$h$restr" denotes h restricted to what the domain must be in that context, and where we are definitely working in $\mathcal{R}ec_{1.01}$, so this definition can only be 'legally' applied when the data is such that

$$bd(|C'|_{h\mathrm{restr}}^{X\mapsto E}) \cap bd(|C''|_{h\mathrm{restr}}^{X\mapsto E}) = \emptyset \;;$$

(4) $\quad |ite(H)(C')(C'')|_h^{X\mapsto E} \;:=\; ite(H)(|C'|_{h\mathrm{restr}}^{X\mapsto E})(|C''|_{h\mathrm{restr}}^{X\mapsto E})$,
where the same remarks as in (3) apply ;

(5) $\quad |\nabla XD|_h^{X\mapsto E} \;:=\; \nabla XD$;

whereas, for $Y \neq X$, picking some

$$Z \notin X \cup Y \cup D \cup E \cup h[bd(E) \times foc(X, \nabla Y D)] \ ,$$

define $|\nabla Y D|_h^{X \to E} := \nabla Z |B|_h^{X \to E}$, where, using the definition two paragraphs back *or* the induction, $B := |D|^{Y \to call Z}$. Note that the length of B is two less than that of $\nabla Y D$, so $|B|_h^{X \to E}$ has already been defined. In this definition, we can use the same h on both sides because $foc(X, \nabla Y D) = foc(X, B)$. This requires a minor inductive argument, to see the 'obvious' fact that $foc(X, |D|^{Y \to call Z})$ and $foc(X, D)$ are the same.

Note that, for any C' and C'' in (3) and (4), there are always choices of h so that the substituted formula (being defined) *is* actually defined. The extreme case might be where $C' = C''$, in which case the 'image under h' of the two halves would need to be disjoint from each other. Even though the two sets in the displayed part of the condition then look to be identical, they are not, because the two "$h$restr"s are quite different, to make them disjoint.

Proposition 3.4. *For data as in the definition, in particular, any choices of the 'Z' as in the last clause occurring as we build inductively, the string* $|C|_h^{X \to E}$ *is a command in* $\mathcal{R}ec_{1.01}$.

[The number of such choices is the size of $bd(C)$.]

Sketch Proof. Each right-hand side in the definition is certainly a command; it's the distinctness of the bound procedure variables that needs comment. For this proof, we'll use that $\mathcal{R}ec_{1.01}$ consists of exactly those commands in $\mathcal{R}ec_{1+}$ for which, for any X, the consecutive symbols ∇X occur at most once. There are no bound procedure variables in (1). The injectiveness of h plus the fact that E has mutually distinct bound variables are the relevant facts for (2). For (3) and (4), the disjointness of $foc(X, C')$ and $foc(X, C'')$ within $foc(X, C)$ is the crucial fact. For (5), the induction plus the fact that Z is chosen disjoint from $h[bd(E) \times foc(X, \nabla Y D)]$ and from C are what is used. In every case, note that $bd(|C|_h^{X \to E})$ consists of $h[bd(E) \times foc(X, C)]$ together with all the Z's in the 'Y-to-Z-choices' occurring in the course of building $|C|_h^{X \to E}$ inductively. Fortunately, we have no need to introduce a notation to specifically record those choices.

We shall say that $\underline{n}$ *is compatible with the re-namings in the substitution-definition of* $|C|_h^{X \to E}$ when

(i) for each Y, $\underline{n}(h(Y,*))$ is independent of $*$; and
(ii) for all Y-to-Z-renamings in building up that definition, $\underline{n}(Z) = \underline{n}(Y)$.

Theorem 3.5. *For any C and E in $\mathcal{R}ec_{1.01}$ with $bd(C) \cap bd(E) = \emptyset$, and for all Θ, X and h, and any $\underline{n}$ compatible with the re-namings in the substitution-definition of $|C|_h^{X\mapsto E}$, we have*

$$\mathcal{M}_{\underline{n}}(|C|_h^{X\mapsto E}, \Theta) \;=\; \mathcal{M}_{\underline{n}}(C, \Theta_{X\mapsto \mathcal{M}_{\underline{n}}(E,\Theta)}) \;.$$

Note that this result agrees with **3.2**, which is a special case of it when E is a procedure variable.

Proof. Proceed by induction on **the length of the string** C. Once again only the interesting inductive case will be done here.

When $C = \nabla X D$, we must show

$$\mathcal{M}_{\underline{n}}(\nabla X D, \Theta) \;=\; \mathcal{M}_{\underline{n}}(\nabla X D, \Theta_{X\mapsto \mathcal{M}_{\underline{n}}(E,\Theta)}) \;,$$

which is immediate from **1.3**, since $X \notin fr(\nabla X D)$.

When $C = \nabla Y D$ where $Y \neq X$, let $B := |D|^{Y\mapsto call Z}$, after choosing some $Z \notin \{X\} \cup \{Y\} \cup C \cup D \cup E \cup image(h)$. So $\nabla Z|B|_h^{X\mapsto E}$ is a typical choice for $|C|_h^{X\mapsto E}$, and we must prove

$$\mathcal{M}_{\underline{n}}(\nabla Z|B|_h^{X\mapsto E}, \Theta) \;=\; \mathcal{M}_{\underline{n}}(\nabla Y D, \Theta_{X\mapsto \mathcal{M}_{\underline{n}}(E,\Theta)}) \;.$$

Proceed by induction on $\underline{n}(Y)$, which equals $\underline{n}(Z)$.

When $\underline{n}(Y) = 0$, both sides are the empty set.

When $0 < \underline{n}(Y) < \infty$, let $\underline{\ell} := (\underline{n}_{Y-})_{Z-}$, recalling that compatibility of $\underline{n}$ implies that $\underline{n}(Y) = \underline{n}(Z)$, and therefore implies compatibility of $\underline{\ell}$ as well. By definition of the semantics of '∇-commands', the right-hand side is

$$\mathcal{M}_{\underline{n}_{Y-}}(D, (\Theta_{X\mapsto \mathcal{M}_{\underline{n}}(E,\Theta)})_{Y\mapsto \mathcal{M}_{\underline{n}_{Y-}}(\nabla Y D, \Theta_{X\mapsto \mathcal{M}_{\underline{n}}(E,\Theta)})})$$
$$= \; \mathcal{M}_{\underline{\ell}}(D, \Psi_{Y\mapsto \mathcal{M}_{\underline{\ell}}(\nabla Y D, \Theta_{X\mapsto \mathcal{M}_{\underline{\ell}}(E,\Theta)})})$$

where $\Psi := \Theta_{X\mapsto \mathcal{M}_{\underline{n}}(E,\Theta)}$. That last equality, inserting $\underline{\ell}$ in three places, follows from **1.4** because Z is a not a bound procedure variable in any of D or E or $\nabla Y D$, and also $Y \notin bd(E)$. Now, by the induction on $\underline{n}(Y)$,

$$\mathcal{M}_{\underline{\ell}}(\nabla Z|B|_h^{X\mapsto E}, \Theta) \;=\; \mathcal{M}_{\underline{\ell}}(\nabla Y D, \Theta_{X\mapsto \mathcal{M}_{\underline{\ell}}(E,\Theta)}) \;[=\; \alpha \;, \text{say}] \;,$$

so the right-hand side is $\mathcal{M}_{\underline{\ell}}(D, \Psi_{Y\mapsto\alpha})$.

The left-hand side is

$\mathcal{M}_{\underline{n}_{Z-}}(|B|_h^{X\to E}, \Theta_{Z\mapsto\mathcal{M}_{\underline{n}_{Z-}}(\nabla Z|B|_h^{Y\to E},\Theta)})$ [[definition of semantics of $\nabla-$commands]]

$= \mathcal{M}_{\underline{\ell}}(|B|_h^{X\to E}, \Theta_{Z\mapsto\alpha})$ [[use **1.4** to change both $\underline{n}_{Z-}$ to $\underline{\ell}: Y \notin bd(|B|_h^{X\to E})\cup bd(\nabla Z|B|_h^{X\to E})$]]
since it′s not bound in D because $\nabla Y D \in \mathcal{R}ec_{1.01}$]]

$= \mathcal{M}_{\underline{\ell}}(B, (\Theta_{Z\mapsto\alpha})_{X\mapsto\mathcal{M}_{\underline{\ell}}(E,\Theta)})$ [[induction on length of C]]

$= \mathcal{M}_{\underline{\ell}}(B, (\Theta_{X\mapsto\mathcal{M}_{\underline{n}}(E,\Theta)})_{Z\mapsto\alpha})$ [[by **1.4** since $\underline{Y \notin bd(E)}$ and $Z \notin bd(E)$]]

$= \mathcal{M}_{\underline{\ell}}(B, \Psi_{Z\mapsto\alpha}) = \mathcal{M}_{\underline{\ell}}(|D|^{Y\to callZ}, \Psi_{Z\mapsto\alpha}) = \mathcal{M}_{\underline{\ell}}(D, (\Psi_{Z\mapsto\alpha})_{Y\mapsto\Psi_{Z\mapsto\alpha}(Z)}) = \mathcal{M}_{\underline{\ell}}(D, \Psi_{Y\mapsto\alpha})$,

as required, using, for the last two equalities respectively: a previous lemma, namely **3.2**; and **1.3** combined with the fact that $Z \notin fr(D)$.

When $\underline{n}(Y) = \infty$, the result is immediate from **1.6**, using the previous cases.

4. Validity of a rule of inference.

The main reference for this is [**infocom**], particularly once we notationally simplify the latter from mutual to single recursion. There, a proof system for a dynamic logic form of Floyd-Hoare logic is presented. Germane to what follows is the restriction in [**infocom**] to *simple* recursion. Here we wish to show how to give a different proof of the validity of the rule there for partial correctness of a recursive command. The advantage here is that we can see now how to get this for *non-simple* recursion as well. The notion of simpleness needn't be defined here, just ignored. (But the introduction to the paper does indicate the crucial point about simpleness.) In any case, our rule here is a generalization in two independent senses: both to deal with the variable-clash question, and also to be strong enough to get completeness.

The system we are discussing deals with 'wfd's' ('well-formed dynamos') in dynamic logic. See [**sem**] for notation concerning wfd's.

We need to refine the notion of 'truth', called "tt", in [**infocom**] to a notion "$\mathsf{tt}\underline{n}$". The semantic assertion "$\mathcal{A}$ $\mathsf{tt}\underline{n}@s$" is defined by structural induction on the wfd $\mathcal{A}$, with all but the last clause identical with those in defining just "$\mathcal{A}$ $\mathsf{tt}@s$". Indeed those clauses coincide with the notion in $1^{\underline{\text{st}}}$order logic of truth at a state, i.e. at an assignment of $1^{\underline{\text{st}}}$order variables (standard Tarski $1^{\underline{\text{st}}}$order definition). The extra final clause is

$$< C > \mathcal{A} \;\; \mathsf{tt}\underline{n}@s \iff \exists (s, s') \in \mathcal{M}_{\underline{n}}(C, \Phi) \text{ with } \mathcal{A} \; \mathsf{tt}\underline{n}@s' \; .$$

Here Φ maps all procedure variables to the empty set (of pairs of states!).

Then "$\mathcal{A}$ $\mathsf{tt}\underline{n}\mathbf{N}$" means "$\mathcal{A}$ $\mathsf{tt}\underline{n}@s$ for all s".

It is immediate from **1.7** here and this definition that tt coincides with $\mathsf{tt}\underline{\infty}$. It is easily deduced that

$(*)$ *$F \to [C]G$ is $\mathsf{tt}@s$ if and only if it is $\mathsf{tt}\underline{\underline{n}X}@s$ for some X and arbitrarily large finite values of n (where $\underline{\underline{n}X}$ takes the value n on X, and the value ∞ on all other procedure variables).*

We wish to prove the following theorem, where $fr(\mathcal{A})$ is the union of the $fr(D)$ for which D is a subcommand of $\mathcal{A}$.

Theorem 4.1. *Assume that $\vdash'$ is a 'transitive' binary relation from sets of wfd's to wfd's such that $\vdash'$ preserves $\mathtt{tt}\underline{n}\mathbf{N}$ for all $\underline{n}$. Suppose given a finite set I of quintuples $(C, f, X, \mathcal{B}, \mathcal{A})$ with the properties*

$$\nabla XC \in \mathcal{R}ec_{1.01} \quad , \quad X \notin fr(\mathcal{A}) \cup fr(\mathcal{B}) \quad \text{and}$$

$$\{\ \mathcal{B} \to [\nabla XC]\mathcal{A} \mid (C, f, X, \mathcal{B}, \mathcal{A}) \in I\ \} \quad \vdash' \quad \wedge_I\ (\mathcal{B} \to [|C_1|_f^{X \mapsto \nabla XC}]\mathcal{A})\ ,$$

where C_1 denotes C with all its bound variables re-named to be distinct from each other and from those of ∇XC. Then $\wedge_I\ (\mathcal{B} \to [\nabla XC]\mathcal{A})\ \ \mathtt{tt}\mathbf{N}$.

Corollary 4.2. *We have validity for the following rule of inference,* simultaneously generalizing that 'in' $\mathcal{R}ec_{1.01}$ from **[infocom]** to the non-simple case and to the case where I can be larger than just a singleton, *namely*

$$\frac{\{\ \mathcal{B} \to [\nabla XC]\mathcal{A} \mid (C, f, X, \mathcal{B}, \mathcal{A}) \in I\ \} \quad \vdash' \quad \wedge_I\ (\mathcal{B} \to [|C_1|_f^{X \mapsto \nabla XC}]\mathcal{A})}{\wedge_I\ (\mathcal{B} \to [\nabla XC]\mathcal{A})}\ ,$$

as long as $X \notin fr(\mathcal{A}) \cup fr(\mathcal{B})$ for all $(C, f, X, \mathcal{B}, \mathcal{A}) \in I$.

The rule in this corollary is of a form which some refer to as a *metarule*. The corollary refers to the $\vdash'$ in **[infocom]**, which is given by a bunch of rules, not including the rule in the corollary itself. These rules aren't needed specifically below in the sketch of the proof. Two interesting points here are that :

(i) we are now working in $\mathcal{R}ec_{1+}$, whereas the reference above refers strictly to $\mathcal{R}ec_{1-}$, where non-simple recursions are excluded; and

(ii) the proof of the corollary uses the fact that we specialized Θ to Φ in the definition of 'truth'. The question unspecializing and adding "for all Θ" gives an interesting avenue for extensive comment in the final section of **[dpth]**.

The corollary's proof then consists in checking the truth preservation assumption in the theorem for the $\vdash'$ in **[infocom]**. This is tedious but elementary and unoriginal, being a wade through the rules defining $\vdash'$ in **[infocom]**, but one which is virtually identical with the corresponding proof given in excruciating detail in **[sing]** for just 'ordinary truth', $\mathtt{tt}$, itself. The point in (ii) above comes up in checking $\mathtt{tt}\underline{n}\mathbf{N}$-validity of the axiom
$\neg < callX > \mathcal{A}$ for all X and $\mathcal{A}$; equivalently, the $\mathtt{tt}\underline{n}\mathbf{N}$-truth of the wfd $[callX]\mathcal{B}$ for all X and $\mathcal{B}$.

Proof of 4.1. To show, as required, for one particular element of the index set I, that $\mathcal{B}_0 \to [\nabla X_0 C_0]\mathcal{A}_0\ \mathtt{tt}\underline{\infty}\mathbf{N}$, we use $(*)$ previous, and ordinary induction on n to demonstrate that, for each $n \in \mathbf{N}$, we have $\mathcal{B}_0 \to [\nabla X_0 C_0]\mathcal{A}_0\ \mathtt{tt}\underline{n(X_0)}\mathbf{N}$, simultaneously for all $(C_0, X_0, \mathcal{B}_0, \mathcal{A}_0)$. (Please excuse the subscripts 0 occurring often in this proof. It just seems easier to follow by thinking of one particular quintuple from I for much of the discussion, using that notation, but not having the subscripts (less often, unfortunately) when varying over I 'locally'.

Recall from dynamic logic that $[D]\mathcal{A}$ is defined to be an abbreviation for $\neg < D > \neg\mathcal{A}$. It follows easily that

$$[D]\mathcal{A}\ \mathtt{tt}\underline{k}@s \iff \text{for all } (s,s') \in \mathcal{M}_{\underline{k}}(D,\Phi) \text{ we have } \mathcal{A}\ \mathtt{tt}\underline{k}@s' \ .$$

Since $\mathcal{M}_{\underline{0(X)}}(\nabla XC, \Phi) = \emptyset$ [because $\underline{0(X)}$ maps X to 0] it follows that $[\nabla XC]\mathcal{A}\ \mathtt{tt}\underline{0}(X)@s$ for all s, X, C and $\mathcal{A}$, and therefore the same holds for $\mathcal{B} \to [\nabla XC]\mathcal{A}$. So the initial case of the induction is trivially true.

To 'get from n to $n+1$', the inductive assumption, the displayed hypothesis in **4.1**, plus the 'truth preservation' of $\vdash'$, taken together, give that $\mathcal{B} \to [|C_1|_f^{X \to \nabla XC}]\mathcal{A}\ \ \mathtt{tt}\underline{n}(X)\mathbf{N}$ for all $(C, f, X, \mathcal{B}, \mathcal{A})$ in I. From this, we shall deduce the required fact that $\mathcal{B}_0 \to [\nabla X_0 C_0]\mathcal{A}_0\ \ \mathtt{tt}\underline{(n+1)(X_0)}\mathbf{N}$.

So suppose that $\mathcal{B}_0\mathtt{tt}\underline{(n+1)(X_0)}@s$ and $(s,s') \in \mathcal{M}_{\underline{n+1(X_0)}}(\nabla X_0 C_0, \Phi)$. The latter set of ordered pairs of states is, by definition,

$$\mathcal{M}_{\underline{n(X_0)}}(C_0, \Phi_{X_0 \mapsto \mathcal{M}_{\underline{n(X_0)}}(\nabla X_0 C_0, \Phi)}) \ ,$$

that is, $\underline{(n+1)(X_0)}_{X_0 -} = \underline{n(X_0)}$, obviously.

We need to show that $\mathcal{A}_0\ \ \mathtt{tt}\underline{(n+1)(X_0)}@s'$. Because $X_0 \notin fr(\mathcal{A}_0) \cup fr(\mathcal{B}_0)$, and only on X_0 do $\underline{(n+1)(X_0)}$ and $\underline{n(X_0)}$ differ, we see that

$$\mathcal{A}_0\ \mathtt{tt}\underline{(n+1)(X_0)}@s' \iff \mathcal{A}_0\ \mathtt{tt}\underline{n(X_0)}@s' \ \text{ and } \ \mathcal{B}_0\ \mathtt{tt}\underline{(n+1)(X_0)}@s \iff \mathcal{B}_0\ \mathtt{tt}\underline{n(X_0)}@s \ .$$

[A very easy induction on wfd's proves the obvious general result here, dependent on **1.2**, and analogous to it.] Because $\mathcal{B}_0\ \mathtt{tt}\underline{n(X_0)}@s$, and because

$$\mathcal{B}_0 \to [|(C_0)_1|_f^{X_0 \to \nabla X_0 C_0}]\mathcal{A}_0\ \ \mathtt{tt}\underline{n(X_0)}\mathbf{N} \ ,$$

it now suffices to show that (s,s') is in the set $\mathcal{M}_{\underline{n(X_0)}}(|(C_0)_1|_f^{X_0 \to \nabla X_0 C_0}, \Phi)$. The latter set of pairs of states, by our hard-earned **3.5**, is the same as

$$\mathcal{M}_{\underline{n(X_0)}}((C_0)_1, \Phi_{X_0 \mapsto \mathcal{M}_{\underline{n(X_0)}}(\nabla X_0 C_0, \Phi)}) \ ,$$

since the assumptions in **3.5** hold. But, by **3.3**, this last set of pairs of states is unchanged if we change the "'$(C_0)_1$" to "C_0", since $\underline{n}(X_0)$ is constant (at the value ∞) on all variables other than X_0 [which is in $\overline{bd}(\nabla X_0 C_0)$ and so not in $bd((C_0)_1)$]. In particular, $\underline{n}(X_0)$ takes the same value on any two that are re-namings of one another to get $(C_0)_1$ from C_0. But (s, s') *is* in that last displayed set, as noted in the previous paragraph with its not-quite-identical display. That completes the proof.

5. Proof of Partial Correctness Completeness.

Here we shall revert to the ubiquitous cavalier attitude which is often phrased something like "variable clashes are avoided by re-naming of bound variables, without further explicit mention", and go back to the simpler notation from before these technicalities were dealt with in our Section 3. The information in the previous two sections should suffice to generate confidence that no confidence tricks are being played by the author in doing this suppression of detail.

In Section 4 of [**infocom**], the following result appears in the more general mutual recursion context of that paper. It refers to a certain $1^{\underline{\text{st}}}$order formula $A(C, F)$ there, which we need not re-define here for what we wish to do. (For cognoscenti, this stuff is a version of what ought to be called the *universal example method*, an ingenious invention of Gorelick [**Gor**] in his Masters thesis under Stephen Cook, called "most general formula" by him.) The reader will of course need to consult [**infocom**] for a full understanding of this section, but not of what it is presenting a proof for.

Lemma CP5. *Suppose given a command D_1, and $1^{\underline{\text{st}}}$order formulas F and G, such that $(F \to [D_1]G)$ ttN. For each subcommand ∇ZE of D_1, choose a pair of matching disjoint lists $\vec{x}$ and $\vec{z}$ of pairwise distinct variables, such that all the variables in E are in $\vec{x}$. Let Γ_{D_1} be the* (finite!) *set of all formulas (one for each such ∇ZE)*

$$\vec{x} \approx \vec{z} \ \to \ [\nabla ZE]A(\nabla ZE, \vec{x} \approx \vec{z}) \ .$$

Then $\Gamma_{D_1} \ \vdash' \ (F \to [D_1]G)$.

The proof refers entirely to the more primitive system $\vdash'$ which does not have the rules for partial and total correctness of recursive commands, so that proof applies here with no modification other than notationally specializing from mutual to single recursion

It is a happy circumstance that, with the generalized rule of the previous section here, we may skip the analogue of **Lemma CP6** in [**infocom**], and proceed fairly directly to a proof of the following completeness theorem. It refers to the proof system obtained by notationally specializing the system in [**infocom**] from mutual to single recursion and replacing the rule $(\mathsf{RCN})_{[\ ]}$ there for partial correctness of recursive procedures with the new rule in the previous section (and also the rules in [**infocom**] for total correctness of re-

cursive procedures may of course be ignored in this section, to be resurrected in the next).

Theorem 5. *We have* $\vdash (F \to [D_1]G)$ *for all* $1^{\underline{\text{st}}}$*order* F *and* G *for which that wfd is true in* **N**.

The following lemma may be proved most handily by writing ∇XC as $\nabla Y_1 \nabla Y_2 \cdots \nabla Y_n B$ for some $n \geq 1$, and of course $Y_1 = X$, and proceeding by induction on the longest sequence

$$\{\nabla X_i C_i \mid i \leq k \geq 1\,, \nabla X_i C_i \subset B\,,\ \nabla X_{i+1} C_{i+1} \subset \nabla X_i C_i\,,\ X_i \text{ free in } \nabla X_{i+1} C_{i+1}\}\,,$$

then by structural induction on B, and for given B, by induction on n. Details are tedious.

Lemma A. *For any* X *and* C*, there is a finite set* Σ *of commands of the form* ∇YD*, that set containing* ∇XC *and also having the property*

$$(\nabla YD \in \Sigma \ \text{ and } \ \nabla ZE \text{ a subcommand of } |D|^{Y \to \nabla YD}) \implies \nabla ZE \in \Sigma\ .$$

Such a set can be generated algorithmically.
The emphasis here is the word "finite", since otherwise, the existence of such a set is banal beyond belief.

Proof of Theorem 5. Since $\vdash'$ is stronger than $\vdash$, we have $\Gamma_{D_1} \vdash (F \to [D_1]G)$ for the set Γ_{D_1} in **CP5**. So it suffices to prove that $\vdash \mathcal{D}$ for each $\mathcal{D} \in \Gamma_{D_1}$. So we must derive $\mathcal{A}_{\nabla XC}$, using the following abbreviations:

$$\text{for any command } B,\ \ \mathcal{A}_B \ :=\ \vec{x} \approx \vec{z} \ \to\ [B]A(B, \vec{x} \approx \vec{z})\ ;$$

$$\text{for any set } \Sigma,\ \ \mathcal{A}_\Sigma \ :=\ \{\ \mathcal{A}_{\nabla YD} \mid \nabla YD \in \Sigma\ \}\ .$$

We now apply the new rule with consequent equal to, for some Σ as in the lemma just above, the conjunction of $\mathcal{A}_\Sigma$, that is, the conjunction of all the wfds in $\{\ \mathcal{A}_{\nabla YD} \mid \nabla YD \in \Sigma\ \}$. (Since $\nabla XC \in \Sigma$ and $\mathcal{A}_{\nabla XC}$ is what we must derive, this will complete the proof.) It remains only to verify the antecedent of that rule, namely

$$\mathcal{A}_\Sigma \ \vdash' \ \wedge \{\vec{x} \approx \vec{z} \ \to\ [|D|^{Y \to \nabla YD}]A(\nabla YD, \vec{x} \approx \vec{z}) \mid \nabla YD \in \Sigma\ \}.$$

Each of the wfds conjuncted on the right-hand side of the display is true in **N**, and so, to derive a fixed wfd from that right-hand side, by using **CP5** with $D_1 = |D|^{Y \mapsto \nabla Y D}$, we need only check that

$$\Gamma_{|D|^{Y \mapsto \nabla Y D}} \subset \mathcal{A}_\Sigma \ .$$

But that's exactly what the definition of Σ in the **Lemma A** tells us, completing the proof.

The two separate uses of **CP5** seem a bit curious, rather clever, but in essence go back at least to [**Apt**] in a sense. However, the sort of recursion dealt with there does not even allow nesting it seems, so the question of simpleness for it would be beside the point. Relating this to [**Old**], and determining whether his results *do* apply to non-simple recursion, both need some further work.

6. Proof of Total Correctness Completeness.

Again we adopt the attitude that variable clashes are avoided by renaming of bound variables, without further explicit mention. In Section 5 of [**infocom**], the following appears (in a more general form for mutual recursion) as a consequence of **CT1** and as parts of **CT3**.

Lemma B. *For all commands D and C, procedure variables X, $1^{\underline{\text{st}}}$order formulas G, and variables $w \notin C \cup D \cup G$, there is a $1^{\underline{\text{st}}}$order formula $B(w, C/Y, D, G)$ such that* (i), (ii) and (iii) *below hold, and satisfying*

$$B(w, C/Y, D, G)\ \mathsf{tt}@s \ \rightarrow\ \ < |D|^{Y \rightarrow \nabla YC} > G \quad \mathsf{tt}\mathbf{N}\ .$$

In each of (i) *to* (iii) *below, a C/Y, a variable w, and strings of variables $\vec{x}$, $\vec{z}$ are involved.* Let us just use $B(C/Y)$ to denote $B(w, C/Y, callY, \vec{x} \approx \vec{z})$. This notation is only used with $\vec{x}$ and $\vec{z}$ matching disjoint strings of distinct variables, with all variables in C from $\vec{x}$, and with one more new variable w. *Then each of the following three $1^{\underline{\text{st}}}$order formulas is true in* $\mathbf{N}$*:*

(i) $\quad F \ \rightarrow\ \exists \vec{z}((\exists w B(C/Y)) \wedge |G|^{\vec{x} \rightarrow \vec{z}})$;

(ii) $\quad \neg |B(C/Y)|^{w \rightarrow 0}$;

(iii) $\quad |B(C/Y)|^{w \rightarrow w+1} \ \rightarrow\ B(w, C/Y, C, \vec{x} \approx \vec{z})$;

where, in (i) *, assume $F \rightarrow < \nabla YC > G$ is true in* $\mathbf{N}$*, with no variables from $\vec{z}$ in the $1^{\underline{\text{st}}}$order formulas F and G.*
(Note that (i) is used only in proving **CT5** below, and so never re-appears in this paper.)

The key is to define $B(w, C/Y, D, G)$ to have the following property, appealing for existence to very old key results essentially due to Gödel :

$$B(w, C/Y, D, G)\ \mathsf{tt}@s \quad \Longleftrightarrow \quad < |D|^{Y \rightarrow \nabla YC} > G\ \mathsf{tt}\underline{\underline{s(w)Y}}@s.$$

This is different from [**infocom**] where a form of partial unfolding of ∇YC is used in the definition. Recall that $\underline{\underline{s(w)Y}}$ maps Y to the natural number $s(w)$, and all other procedure variables to ∞, so we are using w to syntactically measure depth, so-to-speak; whereas [**infocom**] 'unfolds "$s(w)$" times'. The latter likely only works in the case of simple recursion, and the present approach is actually simpler, as well as more general, once the technicalities

of this paper's first three sections are done (and partially ignored here in the notation). The proof of **Lemma B** using only the display characterizing $B(w, C/Y, D, G)$ just above is very straightforward.

Next is another result from [**infocom**] whose proof there depends only on **Lemma B** above and the 'preliminary' proof system $\vdash^w$ from [**infocom**], so we need not say more on its proof. The latter proof system uses one extra axiom beyond those for $\vdash'$, namely

$$(\mathsf{AX})_{<\nabla>} \qquad < (|C|^{X \to \nabla XC} > \mathcal{A} \quad \longleftrightarrow \quad < \nabla XC > \mathcal{A} \ .$$

And it requires the non-appearance of the variable w in three of the rules. But it has nothing to do with the two basic rules (partial and total) for correctness of recursions.

Lemma CT5. *Suppose given a command D_1, and $1^{\underline{\text{st}}}$order formulas F and G, such that $(F \to < D_1 > G)$ $\mathsf{tt}\mathbf{N}$. For each C/Y such that ∇YC is a sub-command of D_1, choose a pair of matching disjoint lists $\vec{x}$ and $\vec{z}$ of pairwise distinct variables, and variable $w \notin \vec{x} \cup \vec{z}$, such that all the variables in C are in $\vec{x}$. Let the set Λ_{D_1} consist of all formulas (one for each such C/Y)*

$$B(C/Y) \ \to \ < \nabla YC > \vec{x} \approx \vec{z} \ .$$

(Each is true in **N** by the display in **Lemma B**.)

Then $\Lambda_{D_1} \ \vdash^w \ (F \to < D_1 > G)$.

It is a particularly happy circumstance here that we shall be able to skip the analogue of **Lemma CT6** in [**infocom**], which had a rather lengthy proof, and proceed fairly directly to a proof of the following completeness theorem. Again it refers to the proof system obtained by notationally specializing the system in [**infocom**] from mutual to single recursion (and replacing the rule $(\mathsf{RCN})_{[\ ]}$ there for partial correctness of recursive procedures with the new rule in the previous section, though that is irrelevant to this half). Note though that the rule in [**infocom**] for total correctness of recursive procedures must be essentially left as is, just this

$(\mathsf{RCN})_{<\,>}$

$$\frac{\vdash \bigwedge_{1\le j\le k} \neg|\mathcal{A}_j|^{w\mapsto 0}\ ,\ \{\ \mathcal{A}_j \to \mathcal{B}_j \mid 1\le j\le k\}\ \vdash^w\ \bigwedge_{1\le j\le k}(|\mathcal{A}_j|^{w\mapsto w+1} \to \mathcal{B}_j\ )}{\vdash\ \bigwedge_{1\le i\le k}(\mathcal{A}_i \to \mathcal{B}_i)}\ ,$$

for all variables w, all wfd's $\mathcal{A}_j$ (because of the substitution, w cannot be a program *variable in any of $\mathcal{A}_j$'s subcommands), and all wfd's $\mathcal{B}_j$* in which *w* does not occur at all.

The point is that, for single *simple* recursion, the case $k = 1$ would suffice, whereas here we do need general k, despite not being in the mutual recursion situation.

On the other hand, we do not need the set Ω which occurs in the rule as presented in [**infocom**], so we have 'taken it to be empty' above, as that's all that's needed for the following proof.

Theorem 6. *We have* $\vdash (F \to\!< D_1 > G)$ *for all* $1^{\underline{\text{st}}}$*order F and G for which that wfd is true in* **N**.

Proof. Since $\vdash^w$ is stronger than $\vdash$, we have $\Lambda_{D_1} \vdash (F \to\!< D_1 > G)$ for the set Λ_{D_1} in **CT5**. So it suffices to prove that $\vdash \mathcal{D}$ for each $\mathcal{D} \in \Lambda_{D_1}$.

Thus it remains only to derive wfd's of the form $\mathcal{B}_{\nabla XC}$ using the following abbreviations:

$$\mathcal{B}_{\nabla XC}\ :=\ B(C/X)\ \to\ <\nabla XC>\vec{x} \approx \vec{z}\ ;$$

$$\text{for any set } \Sigma,\ \ \mathcal{B}_\Sigma\ :=\ \{\ \mathcal{B}_{\nabla YD}\ \mid\ \nabla YD \in \Sigma\ \}\ .$$

Fix C/X and apply $(\mathsf{RCN})_{<\,>}$ with consequent equal to, for some Σ as in the **Lemma A** of the previous section, the conjunction of $\mathcal{B}_\Sigma$, that is, the conjunction of all the wfds in $\{\ \mathcal{B}_{\nabla YD} \mid \nabla YD \in \Sigma\ \}$. (Since $\nabla XC \in \Sigma$ and $\mathcal{B}_{\nabla XC}$ is what we must derive, this will complete the proof.)

It is important not to slither past the question of how to choose the variables x_i etc. in all these formulas: Simply take all the $1^{\underline{\text{st}}}$order variables occurring in all the commands D with $\nabla YD \in \Sigma$ (or any bigger finite set of variables, if you have some peculiar desire to do so), make them into a list, pick any matching disjoint list of $z_i's$, pick w different from all the above, and

use this same collection of variables in every one of the $\mathcal{B}_{\nabla YD} \in \mathcal{B}_\Sigma$. Those wfd's are then completely explicit up to truth in **N**, which is all that matters.

It remains only to verify the antecedents of the instance being used of $(\mathsf{RCN})_{<\ >}$. In this instance of the rule, the natural number "k" is the cardinality of the set Σ, and the subscripts i between 1 and k are matched one-to-one with pairs D/Y for which $\nabla YD \in \Sigma$.

The first antecedent is the conjunction of the $\neg|B(D/Y)|^{w \rightarrowtail 0}$, which is derivable by the oracle, since it is true in **N** by (ii) of **Lemma B**.

The main antecedent then requires us to show

$$\mathcal{B}_\Sigma \ \vdash^w \ \wedge \{|B(D/Y)|^{w \rightarrowtail w+1} \ \rightarrow \ < \nabla YD > \vec{x} \approx \vec{z} \ \mid \ \nabla YD \in \Sigma \ \}.$$

First, using $(\mathsf{AX})_{<\nabla>}$ and $(\mathsf{UNAR})_{<\ >}$ of the system, the oracle and hypothetical syllogism, we may replace the $< \nabla YD >$ following $\vdash^w$ in the display by $< |D|^{Y \mapsto \nabla YD} >$.

Next, using **Lemma B**(iii), the oracle and hypothetical syllogism, we may replace $|B(D/Y)|^{w \rightarrowtail w+1}$ following $\vdash^w$ in the display by $B(w, D/Y, D, \vec{x} \approx \vec{z})$.

Each entire wfd in the conjunction following $\vdash^w$ in the display is true in **N**, and so the resulting wfd from these two replacements also is. And so we can apply **CT5** again to derive the latter wfd from $\mathcal{B}_\Sigma$, and complete the proof, as long as

$$\Lambda_{|D|^{Y \mapsto \nabla YD}} \ \subset \ \mathcal{B}_\Sigma \ .$$

But that's exactly what the definition of Σ in **Lemma A** tells us.

Several examples of specific verifications concerning non-simple recursive commands using the system here may be found in [**xmpl**].

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.*
papers #9 and #12 on : http://www.math.uwaterloo.ca/PM(underscore)
Dept/Homepages/Hoffman/hoffman.shtml .

[**C**] Cook, Stephen A. *Soundness and Completeness of an Axiom System for Program Verification.* SIAM. J. COMPUT. (7), 1978, 70-90.

[**C+**] Cook, Stephen A. *Corrigendum : etc.* SIAM. J. COMPUT. 10(3), 1981, 612.

[**Cst**] Cousot, Patrick *Methods and Logics for Proving Programs.* pp. 841-993 in: *Handbook of Theoretical Computer Science*, ed. J. van Leeuwen, Elsevier Science Publishers B. V., 1990.

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

[**dpth**] Hoffman, P. *Depth Measure Semantics and Partial Correctness Rules for Imperative Recursion.*
paper #18 on : http://www.math.uwaterloo.ca/PM(underscore)
Dept/Homepages/Hoffman/hoffman.shtml

[**End**] Enderton, Herbert A. *A Mathematical Introduction to Logic.* Harcourt/Academic Press, 2001/ 1972.

[**Gor**] Gorelick, Gerald Arthur *A Complete Axiomatic System for Proving Asser- tions about Recursive and Non-recursive Programs.* Tech. Report 75, Dept. of Computer Science, Univ. of Toronto, 1975.

[**H-K-T**] Harel, D., Kozen, D. and Tiuryn, J. *Dynamic Logic.* MIT Press, 2000.

[**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

[**Hom**] Homeier et al *USPatent #5,963,739* Slogan: Making Formal Methods into Normal Methods. (As yet, no record of SuperBowl TV commercial.)

[**infocom**] Hoffman, P. *A Proof System for Recursive Programming with Local Declarations.*
paper #16 on : http://www.math.uwaterloo.ca/PM(underscore)Dept/Homepages/Hoffman/hoffman.shtml

[**Kmn**] Kliemann, Thomas *Hoare Logic and Auxiliary Variables.* Formal Aspects of Computing, 11(1999) 541-566.

[**Men**] Mendelson, Elliott *Introduction to Mathematical Logic.* Chapman & Hall, 1964.

[**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.*
paper #15 on : http://www.math.uwaterloo.ca/PM(underscore)Dept/Homepages/Hoffman/hoffman.shtml

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

[**O'D**] O'Donnell, Michael J. *A Critique of the Foundations of Hoare-Style Programming Logics.* in: [Kozen-ed.] *Logics of Programs.* pp. 349-374 of Lecture Notes in CS # 131, Springer, 1982.

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

[**Sch**] Schreiber, Thomas *Auxiliary Variables and Recursive Procedures.* in: *TAPSOFT'97: Theory and Practice of Software Development* pp. 697-711 of Lecture Notes in CS # 1214, Springer, 1997.

[**sem**] Hoffman, P. *Detailed Semantics of Toy Command Languages for Iteration and Recursion.*
paper #17 on : http://www.math.uwaterloo.ca/PM(underscore)
Dept/Homepages/Hoffman/hoffman.shtml

[**sing**] Hoffman, P. *Deterministic Dynamic Logic Imperative Recursive Programming Proof Systems.*
paper #14 on : http://www.math.uwaterloo.ca/PM(underscore)
Dept/Homepages/Hoffman/hoffman.shtml

[**Ten**] Tennant, R.D. *Denotational Semantics.* Ch.2, pp.170-322 in Vol. 3 of *Handbook of Logic in Computer Science.* Eds. S Abramsky, Dov M. Gabbay and T.S.E. Maibaum, Oxford U. Press, 1994.

[**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.

[**while**] Hoffman, P. *Systems for 'while'-Total Correctness with Quantifiers and Connectives.*
paper #13 on : http://www.math.uwaterloo.ca/PM(underscore)
Dept/Homepages/Hoffman/hoffman.shtml

[**xmpl**] Hoffman, P. *Challenge Examples and a Conjecture re Correctness of Imperative Recursive Programs.*
paper #19 on : http://www.math.uwaterloo.ca/PM(underscore)
Dept/Homepages/Hoffman/hoffman.shtml