A Proof System for Recursive Programming with Local Declarations

Peter Hoffman

Pure Mathematics, University of Waterloo

Waterloo, ON, Canada

Running Title: Local Recursion Proof System

Key words:
recursive programming, local declarations, verification system, dynamic logic

All correspondence re manuscript to address above or

phoffman@uwaterloo.ca

or

519-664-3670 (usually)

or

519-888-4567 X.35564 or X.33484

A Proof System for Recursive Programming with Local Declarations

Peter Hoffman

Pure Mathematics, University of Waterloo

Abstact. *For a reasonably general instance of a 'pure' recursion imperative command language, with* local *procedure declarations and mutual recursion in any number of procedure variables, we find a sound, Cook-complete program verification proof system, for both partial and total correctness. The restriction that procedure declarations be strictly global has been employed in that portion of earlier literature not strictly to do with* partial *correctness, except for a discussion in deBakker's book without completeness considerations, and Harel's system for a considerably extended command language including a vast array of non-deterministic commands to make the system work. So the new aspect here is dealing with* total *correctness while being able to have procedures declared whose bodies themselves contain procedure declarations (that is,* nested *recursions). The work is expressed in terms of deterministic dynamic logic, has all proofs presentable in humanly readable form, and emphasizes mathematical accuracy, leaving the practical aspects of program verification for future cogitation.*

Somewhat unexpectedly, for *total* correctness of imperative recursive programs with *nested* recursions, there seems to be no known sound and Cook-complete (self-contained) proof system (except for one idiosyncratic 'non-self-contained' system, p.56 in [**Harel**], discussed below). Slightly more precisely, this question is entirely about (parameterless) recursion itself: it is studied with a stripped-down 'toy' command language built from just assignments, sequencing, 'if-then-else', and recursive commands with *local* declarations of parameterless procedures. See Section 1 for the exact syntax.

Peripheral questions (e.g. concerning expressivity) will be crudely finessed away by basing the command language on the language of $1^{\underline{\text{st}}}$order number theory, and basing its semantics on the standard interpretation, the natural numbers. We shall work with mutual recursion. But that is not the central point, which comes up equally for recursion on a single procedure variable.

The central point to be made in this paper is that one can produce a proof system which 'almost' does the job: to do so we find it necessary to assume that, in the body associated to each of the procedure variables

involved in a block of declarations, no other procedure variable can occur freely; that is, only the procedure variables 'being recursed upon' can be called, except for calls within properly nested 'sub-recursions', where that call would be 'covered' by its own 'sub-declaration'. This is more easily specified technically (see Section 1), and is called *simple* recursion in [**Harel**]. He actually does give a system for this case, one which is sound and (with one minor correction—inserting an existential quantifier at the left end of the consequent in his rule for total correctness of a recursive command) is Cook-complete. But, as criticized for example in [**AdB**], p.131, this is not a "self-contained" system. Harel expands the language with a vast panoply of extra non-deterministic commands, which occur quite explicitly in his proof rules for both partial and total correctness of recursive commands. (Actually, because of the introduction of a second kind of $1^{\underline{\text{st}}}$order variable, the so-called *counting* variables, the language is also somewhat expanded in [**AdB**], where only global declarations of procedure variables occur in any case.) Our system stays 'within' the given command language, as well as our recursions being nested when required.

Note that the issue is not one concerning pure existence, in that the following expressivity result is relatively standard: the Floyd-Hoare total correctness assertions here (and more generally the dynamic logic assertions) are semantically equivalent to $1^{\underline{\text{st}}}$order formulae which can be found by an algorithm. And so those assertions that are true form a set which is recursively enumerable in $Th(\mathbf{N})$. See discussion of this matter for more realistic programming languages in [**C-G-H**], [**Gr1**], [**Gr2**] and [**Gr-Hu**]. Nor is the question of existence here to be waved away by some de-nesting or other syntactic transformation, since the pure recursion languages, where self-contained Cook-complete total correctness proof systems already occur in the literature (listed in the next paragraph and where such transformations would necessarily take their values), are not Turing-equivalent, i.e. are not strong enough to compute all (partial) recursive functions, once they are 'purified' by removing iteration.

Finding such a proof system for *non-nested* recursion was first done (in this journal) by America and deBoer in [**AdB**] (about a dozen years after Harel), patching up the unsound system from about ten years earlier in [**Apt**]. More recently there has been a spate of papers from the Isabelle/HOL specialists—e.g. [**Kln1**],[**Kln2**],[**Nip1**],[**Nip2**],[**NvO**],[**Sch**],[**vOh1**],[**vOh2**]), many entirely about partial correctness, but a few containing proof systems

for total correctness with non-nested recursion.

From looking at actual ('non-toy') programming languages, little convincing needs to be done that allowing nested recursion is desirable. That doing so may not constitute a completely trivial step-up from the 'fixed body' case seems a possibility when one reflects on the following: By Clarke's celebrated incompleteness theorem [**Clk**], and by strong results, e.g. in [**Old**], showing the more-or-less impossibility of strengthening that theorem, there is a programming language with nested recursion (and lots of other features) for which no sound Cook-complete partial correctness proof system can exist; and yet such a system does exist as soon as one forbids nesting ([**Old**], pp.189-190). These ALGOL-type languages are of course far richer than what we deal with here. And partial correctness is a much more developed theory than is total correctness. But the point should be clear that passing from disallowing to allowing nesting is not necessarily a triviality. It is possible that the systems above for "fixed-body recursion" could be adapted to the nested case by introducing a bunch of extra syntactic categories, 'Gentzenizing' so to speak, as done by Olderog [**Old**] to smooth the way to his quite definitive, but technically complicated, results for partial correctness. That will not be our approach.

Inspired by [**Harel**], we shall exploit the convenience of dynamic logic as a language. There will be no need for various (sometimes ad-hoc) devices such as *counting variables, auxiliary variables, adaptation rules, contexts, correctness phrases, conversion of Hilbert systems to Gentzen systems* to make sense of soundness, *environments*, ALGOL-like *programs*, *units*, *atomic Hoare formulas* of three distinct types, *sets of the latter*, *proof lines* etc. which often seem to crop up when dealing with the (perhaps unnatural) languages above where all the procedure variables have fixed meanings (i.e. only global declarations involved), and where one is denied the flexibility provided by the language of dynamic logic. (Some of those might be needed were we to extend the present paper by allowing local variables and procedures *with* parameters.) A byproduct of employing dynamic logic is that both partial and total correctness assertions, as well as many others, such as assertions concerning semantic equivalence of commands, are formalizable. And so the true ones become formally provable (modulo the oracle for $1^{\underline{\text{st}}}$order truth) within the system.

Another reason for being able to avoid some of the list of adhocisms above is the nature of our two proof rules for recursive commands, as described in

the next paragraph.

There has been a deluge of papers since the 1970's concerning partial correctness of recursive commands, many of which allow nesting. See particularly **[Old]**, where one is much closer to real programming languages than to toys for boys. So the completeness result for the *partial* correctness assertions in the present paper is of only slight interest, and only perhaps for the reason that the rule here for partial correctness of recursive commands is somewhat novel. In fact, the two rules in our system for recursive commands (essentially for partial and for total correctness) resemble what are often called *meta-rules*. They come from fundamental ideas of the 1960's (Dana Scott) and 70's (O'Donnell, Solokowski). But they differ from earlier rules in not making the system *self-referential*: that is, $\vdash$ is defined by rules none of which contain antecedents which require an auxiliary conditional $\vdash$-derivation. Our two recursion rules, in both the partial and the total correctness cases, each has as main antecedent the requirement for a conditional derivation which uses a more primitive system. This is a second reason for us not needing counting variables, etc., as listed in the previous paragraph. It also gets rid of a drawback pointed out in **[O'D]**: the fact that adding new valid rules to a self-referential system, when extending the language for example, can turn a sound system into an unsound one, by indirectly altering the meaning of the recursion rule(s).

It appears that, of the papers related to correctness of object-oriented programming, JAVA, etc., where recursion comes up in other guises, e.g. **[BP]**, **[vOh1]**, very few deal with total correctness and even fewer give consideration to completeness questions, with the intersection of those sets empty, as far as one can determine. So serious overlap with the present paper seems unlikely. Hopefully this paper may be found to be useful in this more 'populated' area.

There are a number of quite general and interesting studies regarding Floyd-Hoare-type systems of a more abstract kind (e.g. **[BT1]**, **[BT2]**,**[KT]**), but again, partial correctness seems always to be the emphasis. These stimulate interesting questions re total correctness, whose answers might eventually simplify and clarify the present paper.

As to the methods of proof: For soundness, as long as the technical intricacies of denotational semantics are avoided, as we do here, nothing of any real depth is needed, one always just follows one's nose (though the case of the proof rule here for partial correctness of recursive commands

should command a bit of interest). For completeness, with the exception of Harel's idiosyncratic system mentioned above, it seems that all the proofs for the various instances of systems mentioned above derive from basic ideas of Gorelick [**Gor**] (i.e. the "Most General Formula" approach), with the need for inventing some adjustments, of course. Our proof method for the new case of total correctness relative to nested recursion follows the same pattern. One might say it largely imitates America-deBoer's adjustments to Apt's (total and 'dualized') variations on Gorelick's (partial) theme. The adjustments and expansions needed to [**AdB**] are fairly extensive and are not entirely obvious, but there is no over-arching theme which can be explained without descending into the technicalities themselves.

To summarize, the new contribution of this paper rests almost entirely on the statements in the first few sentences at the beginning, with emphasis on the words *total* and *nesting*, and rests just a bit on the novelty of the two recursion proof rules.

Here is the plan:

In Section 1, we give the syntax, discuss several forms of substitution needed, and then give all the rules of inference, including the two rules mentioned above for recursive commands.

In Section 2, we prove the soundness of the system.

In the brief Section 3, we state a theorem of Harel which reduces the general question of *relative-* (or *Cook-*) completeness of such a dynamic logic system to essentially that of classical Floyd-Hoare partial and total correctness.

Sections 4 and 5 (respectively, for partial and total correctness) contain detail appropriate to a reasonable-length publication on proving the Cook-completeness of the system. The two sections are organized to make the informal duality between the two situations apparent. A sequence of technical results together with some proof descriptions are given. In two lengthy web-page-pdf references, we have provided a quite painstaking checking of details, on this and all the other aspects of this endeavor. That seems essential in view of past problems in the literature with many unsound and/or incomplete systems. In particular, though relatively recent papers from the Isabelle/HOL experts bring to light individuals who have more faith in a proof of soundness or completeness if it has been machine-generated/checked (see [**Kmn**] [**Nip1**] [**Sch**] [**vOh2**]), the previous makes clear, without us-

ing excessive numbers of journal pages, that these proofs can be given in a humanly readable and checkable form.

Another break with tradition in the subject is that we launch right into the general case of recursion on *many* procedure variables, allowing mutual recursion. One of the two web-page references mentioned above, [**sing**], does the case of recursion on a single procedure variable in lurid detail. But it turns out that the general case is no more difficult, at least as long as we stick to the case of simple recursion, roughly defined above. The other such reference, [**mutu**], gives all extra details needed for that case of recursion on many procedure variables. So the oft-used strategy (of simply claiming the straightforwardness of generalizing to mutual recursion without actually providing any details because they are so messy) is unnecessary here.

History of the subject; credit for the major ideas (none of which are mine), particularly to Stephen Cook, David Harel and Dana Scott (and of course Floyd and Hoare); more comparison to existing literature; etc.... occur briefly within the body of the paper and in much detail in the second section of [**sing**]. Note that [**AdB**] has been influential for this work. See [**sing**] for both the positive and the negative side of this. This work of America and deBoer has also been important for the material in the references to the Isabelle/HOL group above with respect to total correctness.

In [**sing**], Appendix Sect. 2, there is analysis, including explicit derivations, of the standard examples of recursion with a single procedure variable.

1. Definitions of Syntax, Semantics and Proof System.

The version here of the language of $1^{\underline{\text{st}}}$order number theory (with equality—the symbol $\approx$ being used in the formal language) has constant symbols 0 and 1, function symbols $+$ and $\times$, and relation symbol $<$. To avoid peripheral/orthogonal issues (and I don't mean *problems*!), we will never consider any interpretation other than $\mathbf{N}$ with its usual structure.

Command syntax. Now define, by structural induction, some strings of symbols: the sets $\mathcal{R}ec$ and $\mathcal{R}ec_-$ of *commands*, using $t, t_1, t_2, \ldots$ for *terms*, and $F, G, \ldots$ for *formulas*, strings already defined as usual in the $1^{\underline{\text{st}}}$order language above.

Additional symbols for $\mathcal{R}ec$ will be

$$ite \mid ; \mid \leftarrow\!\!\div \mid X_1, X_2, \cdots \mid \nabla_1, \nabla_2, \cdots \mid call \quad .$$

(The first two will be our notation for expressing "if-then-else" and sequencing commands, respectively. The rest will be explained below.)

Atomic commands are : callX_i, and $x \leftarrow\!\!\div t$ for every variable x and term t of the $1^{\underline{\text{st}}}$order language. The structural induction generates, for commands C, D, C_i and quantifier-free $1^{\underline{\text{st}}}$order formulas G, the new commands

$$(C; D) \quad | \quad ite(G)(C)(D) \quad | \quad \nabla_j C_1 Y_1 C_2 Y_2 \cdots C_k Y_k \quad ,$$

where $Y_1, Y_2, \cdots, Y_k$ are mutually distinct X_ℓ's and $1 \leq j \leq k$. This defines the command language $\mathcal{R}ec$.

The string $x \leftarrow\!\!\div t$ is our notation for the assignment command. The string $\nabla_j C_1 Y_1 C_2 Y_2 \cdots C_k Y_k$ is meant to 'convey' the mutual recursion

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

The subset $\mathcal{R}ec_-$ is defined by requiring that *no C_i in the '∇-command' has a free occurrence of a subcommand call$X$ for any X other than the Y_j's.* Following [**Harel**], this restriction is here called *simple* recursion. See [**mutu**] for a thorough, but predictable, discussion of the inductive definitions of *occurrences of subcommands* and *freeness of an occurrence of call$X$*, the latter paraphrasable as 'all occurrences except those of the *callY_i* in $\nabla_j C_1 Y_1 C_2 Y_2 \cdots C_k Y_k$'. The symbol ∇ behaves in many respects like a quantifier acting on the procedure variables X_i. Then $\mathcal{R}ec_-$ is essentially identical

with the deterministic part of [**Harel**]'s (p.45) language. Most things here work more generally for $\mathcal{R}ec$ with two important exceptions:

(1) the relatively simple semantics of the recursion command would need amplification, and much more care to avoid 'variable clashes' would be needed in the definitions of substitutions; and

(2) the final proof for total correctness at the end depends on being in the sublanguage $\mathcal{R}ec_{-}$.

Isolating these is done with a hope of later overcoming the restriction from $\mathcal{R}ec$ to $\mathcal{R}ec_{-}$. See [**sem**] for how to deal with (1).

Abbreviate the command $\nabla_j C_1 Y_1 C_2 Y_2 \cdots C_k Y_k$ to $\nabla_j \vec{C}/\vec{Y}$. The notation $\vec{C}/\vec{Y}$ will never be used except when $\vec{C} = (C_1, \cdots, C_k)$ is a sequence of commands of the same length as the sequence $\vec{Y} = (Y_1, \cdots, Y_k)$ of *distinct* procedure variables.

Command semantics. We are thinking of the usual meanings for the other constructs : $x \doteqdot t$ for *assignment*, $(C; D)$ for *sequencing*, *ite* for *if-then-else*.

Let $\mathcal{S}te$ be the set of states $s : \mathcal{V}ar \to \mathbf{N}$, where $\mathcal{V}ar$ is the set of variables in the $1^{\underline{\text{st}}}$order language, those variables usually denoted $x, y, z, \ldots$. Formally, the semantics is a function (defined in the following paragraphs)

$$m \ : \ \mathcal{R}ec_{-} \ \to \ 2^{\mathcal{S}te \times \mathcal{S}te} \ ,$$

where $2^{\mathcal{S}te \times \mathcal{S}te}$ is the set of all subsets of $\mathcal{S}te \times \mathcal{S}te$. It will make $\mathcal{R}ec_{-}$ into a deterministic imperative command language, proof of the deterministic aspect being entirely elementary and provided in painstaking detail in [**sing**].

We take $m(callX_i)$ to be empty for all i. Defining m on all the other non-recursive commands is elementary and classical—see [**sing**]. As for the ∇-command, first it is convenient to make two diversions.

Dynamic Logic Syntax and Semantics. We review [**Harel**]. Define by structural induction another set of strings (called here *wfs's—well-formed strings*) denoted $\mathcal{A}, \mathcal{B}, \ldots$:
An *atomic wfs* is just the same as an atomic formula in the $1^{\underline{\text{st}}}$order language, that is,

$$t_1 \approx t_2 \quad \text{or} \quad t_1 < t_2 \quad \text{for terms } t_1, t_2 \text{ of the } 1^{\underline{\text{st}}}\text{order language .}$$

The *inductively defined wfs*'s are given by

$$\neg\mathcal{A} \ \mid \ (\mathcal{A} \wedge \mathcal{B}) \ \mid \ \forall x \mathcal{A} \ \mid \ < C > \mathcal{A}$$

for wfs's $\mathcal{A}, \mathcal{B}$, with $\neg$, $\wedge$, $\forall$ the same symbols as in $1^{\underline{\text{st}}}$order, for x any variable in $1^{\underline{\text{st}}}$order, and for C any command.

Abbreviated wfs's are used as usual : Remove outside brackets and use the priority rule making $\rightarrow$ and $\leftrightarrow$ 'less sticky' than $\vee$ and $\wedge$;

$\mathcal{A} \vee \mathcal{B}$ is $\neg(\neg\mathcal{A} \wedge \neg\mathcal{B})$; $\mathcal{A} \rightarrow \mathcal{B}$ is $\neg(\mathcal{A} \wedge \neg\mathcal{B})$; $\mathcal{A} \leftrightarrow \mathcal{B}$ is $(\mathcal{A} \rightarrow \mathcal{B}) \wedge (\mathcal{B} \rightarrow \mathcal{A})$;

$\exists x \mathcal{A}$ is $\neg \forall x \neg \mathcal{A}$; $[C]\mathcal{A}$ is $\neg < C > \neg \mathcal{A}$.

Note that the $1^{\underline{\text{st}}}$order formulas all are wfs's.

Define $\mathcal{A}$ tt**N** to mean $\mathcal{A}$ tt@s for all states s (use the overused $\models$ here twice if preferred), where the latter is defined by induction on $\mathcal{A}$ as for $1^{\underline{\text{st}}}$order logic with one addition:

$$< C > \mathcal{A} \ \text{tt@}s \iff \text{there is some } s' \text{ with } (s, s') \in m(C) \text{ and } \mathcal{A} \ \text{tt@}s' \ .$$

By determinacy, the s' just above is unique. This last display seems to be the essence of dynamic logic. It follows immediately that

$$[C]\mathcal{A} \ \text{tt@}s \iff \text{for all } s' \text{ with } (s, s') \in m(C) \text{ we have } \mathcal{A} \ \text{tt@}s' \ .$$

Substitution. Three forms of this are important for this paper, and we shall use a convenient, if non-standard, notation.

Let the object O be a term, a formula or a wfs. Then $|O|^{y \rightarrow t}$ will always denote O with the term t substituted for all free occurrences of the variable y. For terms and formulas, there is no need to write down the familiar inductive definitions.

Substitution for y will never be applied to any wfs $\mathcal{A}$ in this paper if y is a program variable in any of its subcommands. A *program variable* is one which occurs to the left of an assignment symbol "$\leftarrow\!\!\!\cdot$ ", this definition being easy to formalize by induction on commands.

For substitutions in wfs's, first ignore the remark about program variables and imagine t is just a variable. Define $|C|^{y \rightarrow t}$ for commands C :

$$|x \leftarrow\!\!\!\cdot t_0|^{y \rightarrow t} := |x|^{y \rightarrow t} \leftarrow\!\!\!\cdot |t_0|^{y \rightarrow t}$$

$$|callX_i|^{y \rightarrow t} := callX_i$$

$$|(C; D)|^{y \rightarrow t} := (|C|^{y \rightarrow t}; |D|^{y \rightarrow t})$$

$$|ite(H)(C)(D)|^{y\to t} \;:=\; ite(|H|^{y\to t})(|C|^{y\to t})(|D|^{y\to t})$$
$$|\nabla_j \vec{C}/\vec{Y}|^{y\to t} \;:=\; \nabla_j |\vec{C}|^{y\to t}/\vec{Y} \quad ,$$

where

$$|\vec{C}|^{y\to t}/\vec{Y} \;:=\; |C_1|^{y\to t} Y_1 |C_2|^{y\to t} Y_2 \cdots \quad .$$

Then the only inductive part of the substitution definition for wfs's not identical to that for 1$^{\underline{\text{st}}}$order formulas is

$$|<C>\mathcal{A}|^{y\to t} \;:=\; <|C|^{y\to t}>|\mathcal{A}|^{y\to t} \; .$$

There is no problem with t being a general term because of our prohibition about y not being a program variable.

Next, replacing $\nabla_j \vec{C}/\vec{Y}$ being central to a rule-validity proof, and where D and $E_1, \cdots, E_k$ may be any commands, the command $|D|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j}$ is defined by induction on D as follows:

$$|x \leftarrow\!\!\div t|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j} \;:=\; x \leftarrow\!\!\div t \; ;$$

$$|callX|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j} \;:=\; callX \; ;$$

$$|(C;D)|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j} \;:=\; (|C|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j}; |D|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j}) \; ;$$

$$|ite(G)(C)(D)|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j} \;:=\; ite(G)(|C|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j})(|D|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j}) \; ;$$

$$|\nabla_i \vec{C}/\vec{Y}|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j} \;:=\; E_i \; ;$$

but when $\vec{B}/\vec{Z} \neq \vec{C}/\vec{Y}$,

$$|\nabla_i \vec{B}/\vec{Z}|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j} \;:=\; \nabla_i |B_1|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j} Z_1 |B_2|^{\forall j \nabla_j \vec{C}/\vec{Y}\to E_j} Z_2 \cdots\cdots \; .$$

All of this just says: "Replace $\nabla_j \vec{C}/\vec{Y}$ by E_j, everywhere in the command and for every j".

Finally, replacing 'free' occurrences of $callX_i$ is central to the completion of the definition for the semantics of commands. We shall define

$$E_{\vec{D}/\vec{Y}} \; , \text{ an abbreviation for } \quad |E|^{\forall i: callY_i \to D_i} \; ,$$

by induction on E as follows:

$$|x \leftarrow\!\!\div t|^{\forall i: callY_i \to D_i} \;:=\; x \leftarrow\!\!\div t \; ;$$

$$\text{if } X \neq Y_j \text{ for any } j \ , |callX|^{\forall i: callY_i \mapsto D_i} \ := \ callX \ ;$$
$$|callY_j|^{\forall i: callY_i \mapsto D_i} \ := \ D_j \ ;$$
$$|(C;C')|^{\forall i: callY_i \mapsto D_i} \ := \ (|C|^{\forall i: callY_i \mapsto D_i}; |C'|^{\forall i: callY_i \mapsto D_i}) \ ;$$
$$|ite(G)(C)(C')|^{\forall i: callY_i \mapsto D_i} \ := \ ite(G)(|C|^{\forall i: callY_i \mapsto D_i})(|C'|^{\forall i: callY_i \mapsto D_i}) \ ;$$
$$|\nabla_j \vec{B}/\vec{Z}|^{\forall i: callY_i \mapsto D_i} \ := \ \nabla_j B \quad .$$

This is the obvious definition of replacing only *free* occurrences. The symbol ∇ behaves somewhat like a quantifier acting on the procedure variables. (The last clause would be considerably more complicated for $\mathcal{R}ec$ as opposed to $\mathcal{R}ec_-$. If interested, see [**sem**] for one way to do that.)

Command semantics continued. Next define a sequence

$$\overrightarrow{\nabla_* C/Y} \ := \ (\nabla_1 \vec{C}/\vec{Y}, \cdots, \nabla_k \vec{C}/\vec{Y}) \ .$$

Now define a command $(\vec{C}/\vec{Y})_j^{<n>}$ for $1 \leq j \leq k$ by induction on $n \geq 0$:

$$(\vec{C}/\vec{Y})_j^{<0>} \ := \ callY_j \quad \text{and} \quad (\vec{C}/\vec{Y})_j^{<n+1>} \ := \ (C_j)_{\vec{D}/\vec{Y}} \ , \text{where}$$

$$\vec{D} = \overrightarrow{(\vec{C}/\vec{Y})^{<n>}} \ := \ ((\vec{C}/\vec{Y})_1^{<n>}, \cdots, (\vec{C}/\vec{Y})_k^{<n>}) \ .$$

We actually define $m(C)$ by induction on

$$\text{nab}(C) \ := \ \text{the number of distinct } \nabla_j \vec{C}/\vec{Y} \text{ which occur as subcommands of } C \ .$$

For fixed nab(C), it is then defined by structural induction on C. The only part of the definition needing clarification is

$$m(\nabla_j \vec{C}/\vec{Y}) \ := \ \bigcup_{n=0}^{\infty} m((\vec{C}/\vec{Y})_j^{<n>}) \ .$$

This is well-defined, but our verification of it depends on restricting from $\mathcal{R}ec$ to $\mathcal{R}ec_-$. What we need is

$$\text{nab}((\vec{C}/\vec{Y})_j^{<n>}) \ < \ \text{nab}(\nabla_j \vec{C}/\vec{Y}) \ .$$

This follows from defining NAB to be the actual *set* of such subcommands, and observing that

$$\text{NAB}(\nabla_j \vec{C}/\vec{Y}) \ = \ \{\nabla_j \vec{C}/\vec{Y}\} \cup \bigcup_{\ell} \text{NAB}(C_\ell)$$

[which is straightforward], and

$$\mathrm{NAB}(\vec{C}/\vec{Y})^{<n>}_j) \;\subset\; \bigcup_\ell \mathrm{NAB}(C_\ell)\ ,$$

[and, of course, $\nabla_j\vec{C}/\vec{Y} \;\notin\; \bigcup_\ell \mathrm{NAB}(C_\ell)$]. The last display follows by a straightforward induction on n from

$$\mathrm{NAB}(C_{\vec{D}/\vec{Y}}) \;=\; \mathrm{NAB}(C) \cup \bigcup \mathrm{NAB}(D_\ell)\ ,$$

where the latter union is over all ℓ for which $callY_\ell$ occurs freely as a subcommand of C. The proof of this is not difficult by induction on C, but the case of a '∇-command' definitely depends on the restriction to $\mathcal{R}ec\_$. (If interested, see **[sem]** for one way to do avoid this restriction, and to define the semantics in a way which uses a definition of *recursion depth* [avoiding substitution], where different recursions within a command can have different depth bounds. This leads to an entirely different notion of unfolding, in which none of the unfoldings themselves have subcommands which are recursive.)

The Proof System.

Just below are the names of all the rules, organized in a convenient fashion. Further down, each rule is written out explicitly, and the exact meaning of the basic derivability assertion symbol, namely $\vdash$, as well as those of three subsidiary ones $\vdash'$, $\vdash''$ and $\vdash^w$, will all be included as we proceed. The first of the following rules for $\vdash$ is a naive way of emphasizing that it is Cook-completeness (not absolute completeness because of Gödel's theorem) which is emphasized (so the system is only 'axiomatic' in the logician's technical sense after abandoning the oracle) :

(ORACLE)

(TAUT) (MP)

$(\mathsf{AX})_{<\leftrightarrows>}$ $(\mathsf{AX})_{<;>}$ $(\mathsf{AX})_{<ite>}$

$(\mathsf{UNAR})_{\forall}$ $(\mathsf{UNAR})_{<\ >}$ (PRE)

$(\mathsf{AX})_{\forall}$ $(\mathsf{AX})_{disj}$ $(\mathsf{AX})_{[AND]}$ $(\mathsf{AX})_{<call>}$ $(\mathsf{AX})_{<\rightarrow>}$

$(\mathsf{AX})_{<\nabla>}$

$(\mathsf{RCN})_{[\ ]}$ $(\mathsf{RCN})_{<\ >}$

All but the last two lines is the system defining $\vdash'$.
All but the last line is the system defining $\vdash''$.
The latter system, with added restrictions for the three non-axiom rules on the 4th line as to the non-occurrence of w, is the system defining $\vdash^w$.
$\vdash'$ and $\vdash^w$ are needed merely to state the rules $(\mathsf{RCN})_{[\ ]}$ and $(\mathsf{RCN})_{<\ >}$.
The second line gives 'propositionality'.
The third line consists of the 'Hoare's innocuous command rules'.
The last line consists of the 'not-so-innocuous command rules' for recursion.
The 4th line has all the non-axioms other than the two rules for recursion. They definitely don't preserve truth 'at a state', only 'in $\mathbf{N}$'.
The penultimate line is the first which has explicit reference to ∇, so must be excluded in defining $\vdash'$, in order to get validity for $(\mathsf{RCN})_{[\ ]}$.

Here are the rules stated in detail, intermixed with discussion for the last three. Note that x cannot be a program variable in a subcommand of $\mathcal{A}$ in $(\mathsf{AX})_{<\leftrightarrows>}$, nor in $(\mathsf{AX})_{\forall}$, as specified when we defined substitution.

(ORACLE) $\quad \emptyset$ / any $1^{\underline{\text{st}}}$ order formula that is true in $\mathbf{N}$

(TAUT) $\quad \emptyset$ / any propositional tautology

(MP) $\quad \mathcal{A}\ ,\ \mathcal{A} \to \mathcal{B}\ /\ \mathcal{B}$

$(\mathsf{AX})_{<\leftrightarrow>} \quad \emptyset\ /\ (< x \leftrightarrow t > \mathcal{A}\ \leftrightarrow\ |\mathcal{A}|^{x \to t})$

$(\mathsf{AX})_{<;>} \quad \emptyset\ /\ (< (C; D) > \mathcal{A}\ \leftrightarrow\ < C >< D > \mathcal{A})$

$(\mathsf{AX})_{<ite>} \quad \emptyset\ /\ (< ite(G)(C)(D) > \mathcal{A}\ \leftrightarrow\ (G \to < C > \mathcal{A}) \wedge (\neg G \to < D > \mathcal{A}))$

$(\mathsf{UNAR})_{\forall} \quad (\mathcal{A} \to \mathcal{B})\ /\ (\forall x\ \mathcal{A} \to \forall x\ \mathcal{B})$

$(\mathsf{UNAR})_{< >} \quad (\mathcal{A} \to \mathcal{B})\ /\ (< C > \mathcal{A} \to < C > \mathcal{B})$

(PRE) $\quad (\mathcal{A} \to \mathcal{B})\ /\ ((\exists z \mathcal{A}) \to \mathcal{B})$

where the variable z is not in $\mathcal{B}$.

$(\mathsf{AX})_{\forall} \quad \emptyset\ /\ \forall x \mathcal{A}\ \longrightarrow\ |\mathcal{A}|^{x \to t}$

$(\mathsf{AX})_{[AND]} \quad \emptyset\ /\ (\mathcal{A} \to [D]\mathcal{B}) \wedge (\mathcal{A}' \to [D]\mathcal{B}')\ \to\ (\mathcal{A} \wedge \mathcal{A}' \to [D](\mathcal{B} \wedge \mathcal{B}'))$

$(\mathsf{AX})_{disj} \quad \emptyset\ /\ (\mathcal{A} \to [D]\mathcal{A})$

if $\mathcal{A}$ and D have no common variable.

$(\mathsf{AX})_{<call>} \quad \emptyset\ /\ \neg < callX > \mathcal{A}$

$(\mathsf{AX})_{<\to>} \quad \emptyset\ /\ ((\mathcal{A} \to < C > \mathcal{B})\ \to\ (\mathcal{A} \wedge \mathcal{D} \to < C > (\mathcal{B} \wedge \mathcal{D})))$

when $\mathcal{D}$ and C have no variables in common.

Recall that $\vdash'$ is defined to mean derivability within the completely straightforward Hilbert-style proof system defined by the rules given up to now. With one more axiom immediately below, the relation $\vdash''$ is defined to mean derivability within the proof system defined by the same rules plus this extra one:

$$(\mathsf{AX})_{<\nabla>} \qquad \text{with } \vec{D} = \overrightarrow{\nabla_* C/Y} \,,$$

$$\emptyset \;/\; (< (C_j)_{\vec{D}/\vec{Y}} > \mathcal{A} \iff < \nabla_j \vec{C}/\vec{Y} > \mathcal{A}) \,.$$

Now define, for any variable w, any set Γ of wfs's, and any wfs $\mathcal{A}$, the relation

$$\Gamma \;\vdash^w\; \mathcal{A}$$

to mean that there is a derivation witnessing $\Gamma \;\vdash''\; \mathcal{A}$ in which justifications using the following three rules have extra restrictions as given. (Basically we just require the variable w to not be involved.)

$(\mathsf{UNAR})_\forall \quad x \neq w \quad ; \quad (\mathsf{UNAR})_{<>} \quad w \notin C \; ; \quad ; \quad (\mathsf{PRE}) \quad z \neq w \quad .$

The crucial property of $\vdash^w$ will be that it is halfway between 'Mendelsonian' and 'Endertonian' in the sense that it preserves 'truth at the set of states with a fixed value of w'.

Using these subsidiary systems, we can now present the two new rules, used for partial and for total correctness of recursive commands, where $\bigwedge$ denotes 'anding' together a finite set of wfs's;
e.g.

$$\bigwedge_{1 \leq j \leq 2} \mathcal{A}_j = \mathcal{A}_1 \wedge \mathcal{A}_2 \;\; :$$

$$(\mathsf{RCN})_{[\,]} \qquad \text{with } \vec{D} = \overrightarrow{\nabla_* C/Y} \,,$$

$$\frac{\vdash \bigwedge_{\mathcal{C} \in \Omega} \mathcal{C} \;\; , \;\; \Omega \;\cup\; \{\mathcal{A}_j \to [\nabla_j \vec{C}/\vec{Y}]\mathcal{B}_j \mid 1 \leq j \leq k\} \;\vdash'\; \bigwedge_{1 \leq j \leq k} (\mathcal{A}_j \to [(C_j)_{\vec{D}/\vec{Y}}]\mathcal{B}_j)}{\vdash \quad \bigwedge_{1 \leq i \leq k} (\mathcal{A}_i \to [\nabla_i \vec{C}/\vec{Y}]\mathcal{B}_i)} \;\; ,$$

where :
(1) *the set* Ω *is a finite set of wfs's* ;
(2) *the object* $\vec{C}/\vec{Y}$ *is such that no* $\nabla_j \vec{C}/\vec{Y}$ *is a subcommand of any command occurring within a wfs in* Ω ;
(3) *the wfs's* $\mathcal{A}_i$ *and* $\mathcal{B}_i$ *also have no occurrences of any* $\nabla_j \vec{C}/\vec{Y}$ *in them.*

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

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

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

It is likely easier to see what these two rules for recursive commands say by re-writing them with $k = 1$ at first, and perhaps even with Ω empty.

To repeat, the list above is the entire system for $\vdash$. Because of using $\vdash'$ and $\vdash^w$ (not $\vdash$ itself) in the main antecedents of these last two rules, and taking advantage of dynamic logic language, this system may be seen to be more straightforward than one sometimes sees, needing no definitions of concepts such as *counting variables, auxiliary variables, adaptation rules, contexts, correctness phrases, conversion of Hilbert systems to Gentzen systems to make sense of soundness.* In the next section we will write down quite explicitly what one means by a derivation within this system. Not claiming any independence facts about it, we have simply included above, and among the fairly obviously derivable rules below, every rule explicitly referred to in the proofs in Sections 4 and 5. All the following rules are easily $\vdash'$-derivable, the first one perhaps being the hardest.

$(\mathsf{AX})_{[ite]}$ $\quad \emptyset\ /\ ([ite(G)(C)(D)]\mathcal{A}\ \leftrightarrow\ (G\to[C]\mathcal{A})\wedge(\neg G\to[D]\mathcal{A}))$

(AND) $\quad \mathcal{A}\to[D]\mathcal{B}\ ,\ \mathcal{A}'\to[D]\mathcal{B}'\ /\ (\mathcal{A}\wedge\mathcal{A}'\to[D](\mathcal{B}\wedge\mathcal{B}'))$

$(\mathsf{SUB})_1$ $\quad \mathcal{A}\ /\ |\mathcal{A}|^{u\mapsto t}$

(Recall that, here and just below, the variable u does not occur as a program variable in any command within $\mathcal{A}$.)

.

$(\mathsf{SUB})_2$ $\quad (\mathcal{A}\to\mathcal{B})\ /\ (|\mathcal{A}|^{u\mapsto x}\to\mathcal{B})$

where the variable u is not in $\mathcal{B}$.

In the justifications for lines of derivations, a notation like $(\mathsf{SUB})_i^{\vec{y}\to\vec{x}}$ sometimes occurs. This just means several applications of $(\mathsf{SUB})_i$, using the variables indicated in the superscript. So actually there are several separate lines; and the order in which it is done is irrelevant. Similar remarks apply to (PRE). Not to be patronizing, but it may be worth noting that in a derivation to witness a $\vdash^w$-claim, with derived rules one must be careful that w is not the variable 'involved', if the derivation of the rule uses any of the three basic rules which are restricted by the definition of $\vdash^w$.

A rule $(\mathsf{AX})_{[\to]}$ (*use* [ ] *in place of* $<\quad>$ *in* $(\mathsf{AX})_{<\to>}$) is valid, and can be easily derived from $(\mathsf{AX})_{disj}$ and (AND). It seems to be a matter of indifference whether $(\mathsf{AX})_{<\to>}$ is made part of the $\vdash'$-system, or reserved for the $\vdash''$-system. But there is a break in the symmetry here, since the analogue of $(\mathsf{AX})_{disj}$, where we use $<\quad>$ in place of [ ], is clearly *invalid*.

Here are a few derivable 'propositional' rules [i.e. derivable simply from (MP) and (TAUT)] to which we shall have occasion to refer:

$$
\begin{array}{cc}
(\mathsf{P})_1 & \mathcal{A}\wedge\mathcal{B}\to\mathcal{C}\ /\ \mathcal{A}\to(\mathcal{B}\to\mathcal{C}) \\
(\mathsf{P})_2 & \mathcal{A}\to\mathcal{B}\ ,\ \mathcal{A}\to\mathcal{C}\ /\ \mathcal{A}\to\mathcal{B}\wedge\mathcal{C} \\
(\mathsf{P})_3 & \neg\mathcal{A}\ /\ \mathcal{A}\to\mathcal{B} \\
(\mathsf{P})_4 & \mathcal{B}\ /\ \mathcal{A}\to\mathcal{B} \\
(\leftrightarrow) & \mathcal{A}\leftrightarrow\mathcal{B}\ /\ \mathcal{A}\to\mathcal{B} \\
(\mathsf{HYSY}) & \mathcal{A}\to\mathcal{B}\ ,\ \mathcal{B}\to\mathcal{C}\ /\ \mathcal{A}\to\mathcal{C}
\end{array}
$$

The following are also easily derivable from the corresponding (UNAR)-rules and a few others:

$$
\begin{array}{cc}
(\mathsf{AX})_{[call]} & \emptyset\ /\ [callX]\mathcal{A} \\
(\mathsf{UNAR})_{\exists} & (\mathcal{A}\to\mathcal{B})\ /\ (\exists x\ \mathcal{A}\to\exists x\ \mathcal{B}) \\
(\mathsf{UNAR})_{[\ ]} & (\mathcal{A}\to\mathcal{B})\ /\ ([C]\mathcal{A}\to[C]\mathcal{B})
\end{array}
$$

Each of $(\mathsf{AX})_{<;>}$ and $(\mathsf{AX})_{<\leftrightarrow>}$ rather easily implies the analogous rule, $(\mathsf{AX})_{[;]}$ or $(\mathsf{AX})_{[\leftrightarrow]}$, which is obtained by changing the explicit occurrences of $<\quad>$ to [ ] .

Finally, there is again an exceptionally simple proof that another twin, namely $(\mathsf{AX})_{[\nabla]}$, is derivable.

Our perhaps over-thoroughness here has been stimulated by the checkered career of this subject, where presentations of unsound/incomplete systems in the literature seems to be the rule rather than the exception.

2. Soundness.

Theorem S′. *The $\vdash'$-proof system is sound. That is, if $\Gamma \vdash' \mathcal{A}$, and if $\mathcal{B}$ ttN for each $\mathcal{B} \in \Gamma$, then $\mathcal{A}$ ttN.*

The proofs of validity for its rules are well-known. See Theorem 14 in [**sing**].

Theorem S″. *The $\vdash''$-proof system is sound.*

It remains to discuss the validity of $(\mathsf{AX})_{<\nabla>}$ The required truth in **N** of

$$< (C_j)_{\vec{D}/\vec{Y}} > \mathcal{A} \quad \longleftrightarrow \quad < \nabla_j \vec{C}/\vec{Y} > \mathcal{A} \ ,$$

where $\vec{D} = \overrightarrow{\nabla_* C/Y}$, will clearly follow once we prove

$$m((C_j)_{\vec{D}/\vec{Y}}) = m(\nabla_j \vec{C}/\vec{Y}) \ .$$

More generally, we'll prove that

$$m(C_{\vec{D}/\vec{Y}}) = \bigcup_{k \geq 0} m(C_{\vec{D}(k)/\vec{Y}}) \ .$$

where $\vec{D}(k) = \overrightarrow{(\vec{C}/\vec{Y})^{<k>}}$. Since $(C_j)_{\vec{D}(k)/\vec{Y}} = (\vec{C}/\vec{Y})_j^{<k+1>}$, this specializes to the required result. To prove $\subset$ here is a straightforward induction on C: The first three cases give equality just as easily—when C is an assignment command, or *call*X for $X \neq Y_i$ for any i, or of the form $\nabla_j \vec{B}/\vec{Z}$, we have $C_{\vec{E}/\vec{Y}} = C$ for all $\vec{E}/\vec{Y}$, so the result is trivial; when $C =$ *call*Y_i, the result amounts to the definition of $m(\nabla_i \vec{C}/\vec{Y})$—and the two remaining inductive cases follow mechanically from definitions.

The proof of $\supset$ in the display above comes from:

Monotonicity Lemma. *If $m(E_i) \subset m(D_i)$ for all i, then, for any C,*

$$m(C_{\vec{E}/\vec{Y}}) \subset m(C_{\vec{D}/\vec{Y}}) \ .$$

Proof. Proceed by induction on C. As above, three cases are trivial, and the other two are entirely mechanical.

To finish the validity proof: By the definition of $m(\nabla_i \vec{C}/\vec{Y})$, we have $m((\vec{D}(k))_i) \subset m(D_i)$ for all i. By monotonicity, $m(C_{\vec{D}(k)/\vec{Y}}) \subset m(C_{\vec{D}/\vec{Y}})$. So that does it.

$\vdash^w$-Lemma . (Mendelsonian $\neq$ $\vdash^w$-validity $\neq$ Endertonian) *Let $k \in \mathbf{N}$. If $\Gamma \vdash^w \mathcal{A}$, and all $\mathcal{B}$ in Γ are $\mathsf{tt}@s_{w\mapsto k}$ for all s, then $\mathcal{A}$ is $\mathsf{tt}@s_{w\mapsto k}$ for all s.*

Remark. By definition, the state $s_{w\mapsto k}$ agrees with s except that it maps w to k. Saying "$\mathsf{tt}@s_{w\mapsto k}$ for all s" is just saying "true at all states with that fixed value, k, for w".

Proof. Once again go through, rule-by-rule, a proof of $\mathbf{S}''$. But this time, each rule must be valid in the stronger sense given in the lemma, using the fact that, for three of the rules, we have those restrictions re w not being essentially involved. (The axiom-style rules are trivial cases here.) Inspecting those proofs, all can be construed to have the form of showing the truth of antecedent(s) at some state $\bar{s}$ implying the truth of the consequent at some related state s which is 'arbitrary'. With the extra restrictions referred to above, in each case the value on w of the two states stays the same, and that's exactly what we need. (The case of a single procedure variable is thoroughly presented in [**sing**].)

Now let us be completely explicit about the overall system. The following can be taken as the definition of $\vdash$, via spelling out the form which derivations take, so no proof of it is offered or required.

The assertion $\Gamma \vdash \mathcal{D}$ holds if and only if there is a sequence

$$(\mathcal{D}_1, \mathsf{J}_1)\ ,\ (\mathcal{D}_2, \mathsf{J}_2)\ ,\ \cdots\cdots\cdots\ ,\ (\mathcal{D}_k, \mathsf{J}_k)$$

(J_i is the 'justification for appending $\mathcal{D}_i$ to $\mathcal{D}_1, \cdots, \mathcal{D}_{i-1}$ ')
with $\mathcal{D}_k = \mathcal{D}$, and such that each J_i is one of :

(0) *the observation that $\mathcal{D}_i \in \Gamma$; or*

(1) *a $\vdash''$-justification for $\mathcal{D}_i$, with antecedents being some from among the lines $\mathcal{D}_1, \cdots \mathcal{D}_{i-1}$ preceding it; or*

(2) *a justification based on $(\mathsf{RCN})_{[\ ]}$: that is, $\mathcal{D}_i$ is a wfs of the form $\bigwedge_j(\mathcal{A}_j \to [\nabla_j \vec{C}/\vec{Y}]\mathcal{B}_j)$ and J_i is a $\vdash'$-derivation witnessing, with $\vec{D} = \overrightarrow{\nabla_* C/Y}$,*

$$\Omega\ \cup\ \{\mathcal{A}_j \to [\nabla_j \vec{C}/\vec{Y}]\mathcal{B}_j \mid 1 \leq j \leq k\}\ \vdash'\ \bigwedge_j(\mathcal{A}_j \to [(C_j)_{\vec{D}/\vec{Y}}]\mathcal{B}_j)\ ,$$

with all wfs's in Ω occurring as lines above $\mathcal{D}_i$, and for no j is the command $\nabla_j \vec{C}/\vec{Y}$ occurring inside any $\mathcal{A}_n$ or $\mathcal{B}_n$ or any $\mathcal{C} \in \Omega$; or finally

(3) *a justification based on* $(\mathsf{RCN})_{<\ >}$*:* $\mathcal{D}_i$ *has the form* $\bigwedge_j(\mathcal{A}_j \to \mathcal{B}_j)$ *, with the variable* w *not occurring in any wfs* $\mathcal{B}_j$*, all wfs's in* Ω *and all* $\neg|\mathcal{A}_j|^{w\mapsto 0}$*'s occur as lines above* $\mathcal{D}_i$*, and* J_i *is a* $\vdash''$*-derivation witnessing*

$$\Omega \cup \{\ \mathcal{A}_j \to \mathcal{B}_j \mid 1 \le j \le k\} \quad \vdash^w \quad \bigwedge_j(|\mathcal{A}_j|^{w\mapsto w+1} \to \mathcal{B}_j)\ .$$

The above explicit description is not intended to patronize the reader, but rather is included because earlier works where 'self-referential rules' are used (that is, the 'full' $\vdash$ appears in the antecedent) can sometimes explain these rules in a semi-mystical fashion (e.g. [**Apt**], p.448—"It is... not clear... in what sense the recursion rule is to be proved sound"), or even in an incorrect fashion (e.g. [**Cst**], p.934).

The main result of this section is then the following.

Theorem S. *The* $\vdash$*-proof system is sound.*

It remains only to check validity of the two (RCN)-rules.

Here is the sequence of results for $(\mathsf{RCN})_{[\ ]}$:

Theorem SUB. *Suppose* $\Gamma \vdash' \mathcal{A}$ *. Then* $|\Gamma|^{\forall i:\nabla_i\vec{C}/\vec{Y}\to E_i} \vdash' |\mathcal{A}|^{\forall i:\nabla_i\vec{C}/\vec{Y}\to E_i}$ *for any* $\vec{C}/\vec{Y}$ *and* $\vec{E}$ *for which all the program variables in* $\vec{E}$ *already occur as program variables in* $\vec{C}$ *(that is, ... in some* C_i*).*

To prove this, one simply checks that the $|\ |^{\forall i:\nabla_i\vec{C}/\vec{Y}\to E_i}$ -operation takes any instance of a rule to another instance of the same rule. The crucial point is that none of the rules for $\vdash'$ explicitly involves ∇. See the proof of Theorem 11 in [**sing**] for excruciating details in the case of a single procedure variable.

In all five of the $\mathbf{S}[i]$ below, the objects $\vec{C}/\vec{Y}$, Ω , $\vec{\mathcal{A}}$, $\vec{\mathcal{B}}$ and $\vec{D}$ are as specified in the statement of $(\mathsf{RCN})_{[\ ]}$.

Corollary S[1]**.**

$$\Omega \cup \{\mathcal{A}_j \to [\nabla_j\vec{C}/\vec{Y}]\mathcal{B}_j \mid 1 \le j \le k\} \ \vdash' \bigwedge_{1\le j\le k}(\mathcal{A}_j \to [(C_j)_{\vec{D}/\vec{Y}}]\mathcal{B}_j) \ \textit{ implies that}$$

$$\Omega \ \cup \ \{\mathcal{A}_j \to [E_j]\mathcal{B}_j \mid 1 \le j \le k\} \ \vdash' \bigwedge_{1\le j\le k}(\mathcal{A}_j \to [(C_j)_{\vec{E}/\vec{Y}}]\mathcal{B}_j)$$

for all $\vec{E}$ *all of whose program variables are also program variables in* $\vec{C}$ *.*

Proof. The conditions imply that there is only the one (very visible) occurrence of $\nabla_j\vec{C}/\vec{Y}$ on each side of $\vdash'$ for which to substitute the command E_j, so this is immediate from **SUB**.

Corollary S[2]**.** *With assumption as in* **S**[1]*, for all* $n \geq 0$ *we have*

$$\Omega \cup \{\mathcal{A}_j \to [(\vec{C}/\vec{Y})_j^{<n>}]\mathcal{B}_j \mid 1 \leq j \leq k\} \vdash' \bigwedge_{1\leq j\leq k} (\mathcal{A}_j \to [(\vec{C}/\vec{Y})_j^{<n+1>}]\mathcal{B}_j) \ .$$

Proof. Take $E_j = (\vec{C}/\vec{Y})_j^{<n>}$ in **S**[1] , noting that $(C_j)_{\vec{E}/\vec{Y}} = (\vec{C}/\vec{Y})_j^{<n+1>}$ and that $(\vec{C}/\vec{Y})_j^{<n>}$ has all its program variables occurring in $\vec{C}$ also as program variables.

Corollary S[3]**.** *With the conditions on* $\vec{C}/\vec{Y}$*,* $\vec{\mathcal{A}}$*,* $\vec{\mathcal{B}}$*,* $\vec{D}$ *and* Ω *stated in* $(\mathsf{RCN})_{[\ ]}$*, if*

$$\Omega \cup \{\mathcal{A}_j \to [\nabla_j\vec{C}/\vec{Y}]\mathcal{B}_j \mid 1 \leq j \leq k\} \vdash' \bigwedge_{1\leq j\leq k} (\mathcal{A}_j \to [(C_j)_{\vec{D}/\vec{Y}}]\mathcal{B}_j)$$

where all wfs's in Ω *are* ttN*, then* $(\mathcal{A}_j \to [(\vec{C}/\vec{Y})_j^{<n>}]\mathcal{B}_j)$ ttN *for all* j *and all* $n \geq 0$ *.*

Proof. Proceed by induction on n. Since $(\vec{C}/\vec{Y})_j^{<0>} = \mathit{call}Y_j$ which never converges, clearly $(\mathcal{A} \to [(\vec{C}/\vec{Y})_j^{<0>}]\mathcal{B})$ ttN for *any* $\mathcal{A}$ and $\mathcal{B}$. For the inductive step, simply use **S**[2], and the soundness of $\vdash'$ from $\mathbf{S}'$.

Lemma S[4]**.** *If, for all* $n \geq 0$ *,* $(\mathcal{A}_j \to [(\vec{C}/\vec{Y})_j^{<n>}]\mathcal{B}_j)$ ttN*, then* $(\mathcal{A}_j \to [\nabla_j\vec{C}/\vec{Y}]\mathcal{B}_j)$ ttN (where j is irrelevant as a subscript on $\mathcal{A}$ and $\mathcal{B}$).

The proof is straightforward from the definitions of $m(\nabla_j\vec{C}/\vec{Y})$, of ttN, and of $[D]\mathcal{B}$.

Corollary S[5]**.** *With the conditions on* $\vec{C}/\vec{Y}$*,* $\vec{\mathcal{A}}$*,* $\vec{\mathcal{B}}$*,* $\vec{D}$ *and* Ω *stated in* $(\mathsf{RCN})_{[\ ]}$*, if*

$$\Omega \cup \{\mathcal{A}_j \to [\nabla_j\vec{C}/\vec{Y}]\mathcal{B}_j \mid 1 \leq j \leq k\} \vdash' \bigwedge_{1\leq j\leq k} (\mathcal{A}_j \to [(C_j)_{\vec{D}/\vec{Y}}]\mathcal{B}_j)$$

where all wfs's in Ω *are* ttN*, then* $(\mathcal{A}_j \to [\nabla_j\vec{C}/\vec{Y}]\mathcal{B}_j)$ ttN *for all* j *.*
The proof is immediate from **S**[3] and **S**[4], and gives the validity of $(\mathsf{RCN})_{[\ ]}$.

Finally, validity of $(\mathsf{RCN})_{<\ >}$ is proved as follows.

Assuming the truth in $\mathbf{N}$ of the antecedents, we prove by induction on ℓ, as required, that, for all ℓ,

$$(\mathcal{A}_i \to \mathcal{B}_i)\ \mathsf{tt}@s_{w\mapsto\ell} \quad \text{for all } s\ ,$$

where i might as well be fixed.

For $\ell = 0_{\mathbf{N}}$, we have $|\mathcal{A}_i|^{w\to 0}\ \mathsf{ff}@s$ for all s, and so $\mathcal{A}_i\ \mathsf{ff}@s_{w\mapsto 0_{\mathbf{N}}}$ for all s, and so $(\mathcal{A}_i \to \mathcal{B}_i)\ \mathsf{tt}@s_{w\mapsto 0_{\mathbf{N}}}$ for all s.

For the inductive step, both $\mathcal{A}_j \to \mathcal{B}_j$ for all j by the inductive assumption, and all wfs's in Ω are $\mathsf{tt}@s_{w\mapsto\ell}$ for all s. Thus by $\vdash^w$**-Lemma**, the truth of the $\vdash^w$-antecedent shows that

$$|\mathcal{A}_i|^{w\to w+1} \to \mathcal{B}_i\ , \text{ which equals } |\mathcal{A}_i \to \mathcal{B}_i|^{w\to w+1}\ ,$$

is also $\mathsf{tt}@s_{w\mapsto\ell}$ for all s. But that last statement amounts to $\mathcal{A}_i \to \mathcal{B}_i$ being $\mathsf{tt}@(s_{w\mapsto\ell})_{w\mapsto s_{w\mapsto\ell}(w+1)}$ for all s. But the latter state is simply $s_{w\mapsto\ell +_{\mathbf{N}} 1_{\mathbf{N}}}$, as required.

3. Main Results—Completeness.

Theorem C. (Cook-completeness of the $\vdash$-system)
If $\mathcal{A}\ \mathsf{ttN}$ *, then* $\vdash \mathcal{A}$ *.*

The proof is split into three pieces just below, the first due to Harel, proved as Theorem 21 in [**sing**].

Theorem H. (essentially in [**Harel**]) *To prove* **C***, it suffices to prove the following two special cases, for* $1^{\underline{\text{st}}}$*order* F *and* G*, and any command* D *:*

(i) $(F \to [D]G)\ \mathsf{ttN} \implies \vdash (F \to [D]G)$ *; and*

(ii) $(F \to\!< D >\! G)\ \mathsf{ttN} \implies \vdash (F \to\!< D >\! G)$.

Theorem CP. *For all* F, D *and* G*, statement* **H**(i) *holds.*

Theorem CT. *For all* F, D *and* G*, statement* **H**(ii) *holds.*

The **P** and **T** stand for 'partial' and 'total', and the **C** for 'completeness'.

Proofs of technical results leading up to these two are outlined in the next two sections, respectively, with the proofs of the two theorems just above coming at the ends of the sections. Note that all these proofs are given in complete detail in [**mutu**], except for its references to some complete proofs in [**sing**]. The parallels between the next two sections is not a new phenomenon, rather a discovery of [**Apt**].

4. Technical results and proof of CP.

Lemma CP1. (a) *For each command C, and $1^{\underline{\text{st}}}$order formula F, there is a $1^{\underline{\text{st}}}$order formula $A(C,F)$—("A" for "after" rather than "post")—such that*

$$A(C,F)\ \mathsf{tt}@s' \quad \Longleftrightarrow \quad \exists s \text{ with } F\ \mathsf{tt}@s \text{ and } (s,s') \in m(C)\ .$$

And so, $A(C,F)$ is the 'universal' A for which $(F \to [C]A)$ $\mathsf{tt}\mathbf{N}$. That is, for all $1^{\underline{\text{st}}}$order K,

$$F \to [C]K\ \ \mathsf{tt}\mathbf{N} \ \Longleftrightarrow\ \ A(C,F) \to K\ \ \mathsf{tt}\mathbf{N}\ .$$

(b) *For any wfs $\mathcal{H}$, there is a $1^{\underline{\text{st}}}$order formula H, whose free variables are from the variables occurring in $\mathcal{H}$, with $H \leftrightarrow \mathcal{H}$ true in $\mathbf{N}$.*

This is standard stuff; the proof is in [**sing**] i.e. Lemma 22.0, at least for the case of a single procedure variable. But it works quite generally, depending only on the effectiveness of the command language $\mathcal{R}ec$ and its semantics.

Lemma CP2. *If D is a command, and Γ is a set of wfs's, define the phrase "Γ der $[D]$" to mean*

$$\text{for all } 1^{\underline{\text{st}}} \text{ order formulas } F \text{ and } G, \quad (F \to [D]G)\,\mathsf{tt}\mathbf{N} \implies \Gamma \vdash' (F \to [D]G)\,.$$

Assume Γ_1 der $[D_1]$ and Γ_2 der $[D_2]$. Then $\Gamma_1 \cup \Gamma_2$ der $[(D_1;D_2)]$; and similarly, with H any quantifier-free $1^{\underline{\text{st}}}$order formula, we also have

$$\Gamma_1 \cup \Gamma_2\ \ \mathsf{der}\ \ [ite(H)(D_1)(D_2)]\ .$$

Half proof of CP2. For the sequencing command, given F and G such that $F \to [(D_1;D_2)]G$ $\mathsf{tt}\mathbf{N}$, by **CP1**(b), choose a $1^{\underline{\text{st}}}$order J such that $J \longleftrightarrow [D_2]G$ $\mathsf{tt}\mathbf{N}$. Then $\Gamma_2 \vdash' J \to [D_2]G$ since Γ_2 der $[D_2]$, so

$$\Gamma_2 \ \vdash'\ [D_1]J \to [D_1][D_2]G \qquad\qquad (I)$$

by $(\mathsf{UNAR})_{[\ ]}$. Since

$$(F \to [D_1]J) \ \longleftrightarrow\ (F \to [D_1][D_2]G)\quad \mathsf{tt}\mathbf{N}\ ,$$

and $(F \to [D_1][D_2]G)$ ttN, we get

$$\Gamma_1 \ \vdash' \ F \to [D_1]J \qquad\qquad (II)$$

since Γ_1 der $[D_1]$. Now $(I), (II)$ and (HYSY) give

$$\Gamma_1 \cup \Gamma_2 \ \vdash' \ F \to [D_1][D_2]G \ .$$

Then $(\mathsf{AX})_{[;]}$ plus (HYSY) allow one to change $[D_1][D_2]$ to $[(D_1; D_2)]$ in the last display, as required.

For 'if-then-else', the proof is essentially simpler, since **CP1** is unneeded. See the second half of 22.5 in [**sing**].

In many places in the rest of this paper,

$$\vec{x} \approx \vec{z} \quad \text{is short for} \quad x_1 \approx z_1 \ \wedge \ \cdots \ \wedge \ x_\ell \approx z_\ell \ ,$$

for various ℓ, where the two matching (i.e. same length) lists of distinct variables are always disjoint from each other.

Lemma CP3. *For any* $1^{\underline{\text{st}}}$*order* F *and* G*, command* C'*, and matching disjoint lists* $\vec{x}$ *and* $\vec{z}$ *of pairwise distinct variables such that all variables from* C' *are in* $\vec{x}$*, if* $(F \to [C']G)$ ttN *, then we have*

$$(A(C', \vec{x} \approx \vec{z}) \ \wedge \ |F|^{\vec{x} \to \vec{z}} \ \to \ G) \quad \mathsf{ttN} \ .$$

The proof consists of straightforward manipulations with semantic definitions. See 22.4 in [**sing**].

Lemma CP4. *With hypotheses as in* **CP3**,

$$\{\vec{x} \approx \vec{z} \ \to \ [C']A(C', \vec{x} \approx \vec{z})\} \quad \vdash' \quad (F \to [C']G) \ .$$

Proof that CP3 implies CP4. First we prove the special case when no $\vec{z}$-variables occur in F nor G. Here is a derivation, shortening $A(C', \vec{x} \approx \vec{z})$ to just A :

$\vec{x} \approx \vec{z} \ \rightarrow \ [C']A$	(premiss)
$\lvert F\rvert^{\vec{x}\to\vec{z}} \rightarrow [C']\lvert F\rvert^{\vec{x}\to\vec{z}}$	$(\mathsf{AX})_{disj}$
$\vec{x} \approx \vec{z} \wedge \lvert F\rvert^{\vec{x}\to\vec{z}} \rightarrow [C'](A \wedge \lvert F\rvert^{\vec{x}\to\vec{z}})$	(AND)
$A \wedge \lvert F\rvert^{\vec{x}\to\vec{z}} \rightarrow G$	(ORAC) and **CP3**
$[C'](A \wedge \lvert F\rvert^{\vec{x}\to\vec{z}}) \rightarrow [C']G$	$(\mathsf{UNAR})_{[\ ]}$
$\vec{x} \approx \vec{z} \wedge \lvert F\rvert^{\vec{x}\to\vec{z}} \rightarrow [C']G$	(HYSY)
$\vec{x} \approx \vec{x} \wedge F \rightarrow [C']G$	$(\mathsf{SUB})_2^{\vec{z}\to\vec{x}}$
$F \rightarrow \ \vec{x} \approx \vec{x} \wedge F$	(ORAC)
$F \rightarrow [C']G$	(HYSY)

Line 7 uses that no z_i is in G or F, giving $\lvert\lvert F\rvert^{\vec{x}\to\vec{z}}\rvert^{\vec{z}\to\vec{x}} = F$ and $\lvert G\rvert^{\vec{z}\to\vec{x}} = G$.

Now in the general case, let $\vec{u}$ be a matching list of pairwise distinct variables, disjoint from $\vec{x}$ and $\vec{z}$. Define F_1 to be $\lvert F\rvert^{\vec{z}\to\vec{u}}$ and let $G_1 := \lvert G\rvert^{\vec{z}\to\vec{u}}$, so F_1 and G_1 have no $\vec{z}$-variables. Since the validity of $(\mathsf{SUB})_1^{\vec{z}\to\vec{u}}$ entails that the truth of $F \rightarrow [C']G$ in $\mathbf{N}$ also guarantees $\mathsf{tt}\mathbf{N}$ for the wfs $F_1 \rightarrow [C']G_1$, the special case above yields

$$\{\vec{x} \approx \vec{z} \ \rightarrow \ [C']A(C', \vec{x} \approx \vec{z})\} \quad \vdash' \quad (F_1 \rightarrow [C']G_1) \ .$$

But now application of $(\mathsf{SUB})_1^{\vec{u}\to\vec{z}}$ allows us the erase the subscript on F and G in this last display, as required.

By the convenient phrase "bits of $(C_1, \cdots, C_k)$", we just mean the commands C_i. To give the inductive definition of the word *subcommand*:
The set of proper *subcommands is empty for atomic commands, is the union of the sets of* all *subcommands of C and of D when the command is obtained from them by sequencing or ite, and is the set of* all *subcommands of bits of $\vec{C}$ when the command is $\nabla_j \vec{C}/\vec{Y}$.*

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 $\nabla_j \vec{C}'/\vec{Y}'$ 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 $\vec{C}'$ are in $\vec{x}$. Let Γ_{D_1} be the* (finite!) *set of all formulas (one for each such $\nabla_j \vec{C}'/\vec{Y}'$)*

$$\vec{x} \approx \vec{z} \ \to \ [\nabla_j \vec{C}'/\vec{Y}']A(\nabla_j \vec{C}'/\vec{Y}', \vec{x} \approx \vec{z}) \ .$$

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

The proof proceeds by induction on D_1.

When it's atomic, the argument is direct and elementary. See M22.2 in [**mutu**].

When $D_1 = \nabla_i \vec{C}/\vec{Y}$: The result comes from **CP4** with $C' = \nabla_i \vec{C}/\vec{Y}$, since the set $\Gamma_{\nabla_i \vec{C}/\vec{Y}}$ here contains the element $(\vec{x} \approx \vec{z} \ \to \ [\nabla_i \vec{C}/\vec{Y}]A)$ occurring to the left of $\vdash'$ in that instance of **CP4**.

When $D_1 = (D; E)$ or $ite(H)(D)(E)$: Apply **CP2** with $\Gamma_1 = \Gamma_D$ and $\Gamma_2 = \Gamma_E$, noting that $\Gamma_{D_1} = \Gamma_D \cup \Gamma_E$ for both types of D_1 here. Use the inductive hypotheses for D and E as the hypotheses in **CP2**. The latter's conclusions are exactly the required results.

Lemma CP6. *For all $\vec{C}/\vec{Y}$ and matching disjoint lists $\vec{x}$ and $\vec{z}$ of pairwise distinct variables such that all variables from $\vec{C}$ are in $\vec{x}$, the finite set Ω below, of wfs's which are*

(i) all ttN*; and*

(ii) involve only commands which occur as subcommands of bits of $\vec{C}$;

is such that, with $\vec{D} = \overrightarrow{\nabla_* C/Y}$,

$$\Omega \ \cup \ \{ \ \vec{x} \approx \vec{z} \ \to \ [\nabla_j \vec{C}/\vec{Y}]A(\nabla_j \vec{C}/\vec{Y}, \vec{x} \approx \vec{z}) \mid 1 \leq j \leq k \ \} \ \ \vdash'$$
$$\bigwedge_{1 \leq j \leq k} (\vec{x} \approx \vec{z} \ \to \ [(C_j)_{\vec{D}/\vec{Y}}]A(\nabla_j \vec{C}/\vec{Y}, \vec{x} \approx \vec{z})) \ .$$

The useful Ω here is the set of all $(\vec{x} \approx \vec{z} \ \to \ [\nabla_i \vec{C}'/\vec{Y}']A(\nabla_i \vec{C}'/\vec{Y}', \vec{x} \approx \vec{z}))$ as $\nabla_i \vec{C}'/\vec{Y}'$ ranges over all subcommands, of bits of $\vec{C}$, which have that form.

To dramatize the contrast between $\vdash'$ and $\vdash''$, note that $(\mathsf{AX})_{[\nabla]}$ gives a completely trivial derivation to witness

$$\vec{x} \approx \vec{z} \ \to \ [\nabla_j \vec{C}/\vec{Y}]A(\nabla_j \vec{C}/\vec{Y}, \vec{x} \approx \vec{z}) \ \ \vdash'' \ \ (\vec{x} \approx \vec{z} \ \to \ [(C_j)_{\vec{D}/\vec{Y}}]A(\nabla_j \vec{C}/\vec{Y}, \vec{x} \approx \vec{z})) \ ,$$

and lots more.

Proof that CP5 implies CP6. We'll fix $j = \ell$ on the right-hand side of $\vdash'$ in **CP6**, and show

$$\Omega \ \cup \ \{ \ \vec{x} \approx \vec{z} \ \rightarrow \ [\nabla_j \vec{C}/\vec{Y}]A(\nabla_j \vec{C}/\vec{Y}, \vec{x} \approx \vec{z}) \mid 1 \leq j \leq k \ \} \quad \vdash'$$
$$(\vec{x} \approx \vec{z} \ \rightarrow \ [(C_\ell)_{\vec{D}/\vec{Y}}]A(\nabla_\ell \vec{C}/\vec{Y}, \vec{x} \approx \vec{z})) \ .$$

In **CP5**, take $F = \vec{x} \approx \vec{z}$; $G = A(\nabla_\ell \vec{C}/\vec{Y}, \vec{x} \approx \vec{z})$; and $D_1 = (C_\ell)_{\vec{D}/\vec{Y}}$. To see that $(F \rightarrow [D_1]G)$ is true in $\mathbf{N}$, the validity of $(\mathsf{AX})_{[\nabla]}$ reduces it to showing that $(F \rightarrow [\nabla_\ell \vec{C}/\vec{Y}]G)$ $\mathsf{tt}\mathbf{N}$. The latter is immediate from the definition of A in **CP1**(a) and the comment just after it. So we can apply **CP5** and get

$$\Gamma_{(C_\ell)_{\vec{D}/\vec{Y}}} \ \vdash' \ (F \rightarrow [D_1]G) \ .$$

Thus it remains only to show that

$$\Gamma_{(C_\ell)_{\vec{D}/\vec{Y}}} \ \subset \ \Omega \cup \{ \ \vec{x} \approx \vec{z} \ \rightarrow \ [\nabla_j \vec{C}/\vec{Y}]A(\nabla_j \vec{C}/\vec{Y}, \vec{x} \approx \vec{z}) \mid 1 \leq j \leq k \ \} \ .$$

This amounts to the fact that, for fixed $\vec{C}/\vec{Y}$, if $\nabla_i \vec{C}'/\vec{Y}'$ is a subcommand of $(C_\ell)_{\vec{D}/\vec{Y}}$ where $\vec{D} = \overrightarrow{\nabla_* C/Y}$, then either (i) the former is one of the $\nabla_j \vec{C}/\vec{Y}$; or (ii) it is a subcommand of a bit of $\vec{C}$.

For this, combine the definition of *subcommand* with the inductive definition of $C_{\vec{D}/\vec{Y}}$ for general $\vec{D}/\vec{Y}$ and C to prove that, *for any commands C and $\vec{D}$, a subcommand of $C_{\vec{D}/\vec{Y}}$, must be : either* (a) *a subcommand of a bit of $\vec{D}$; or else* (b) *of the form $(C_1)_{\vec{E}/\vec{Z}}$ for some subcommand C_1 of C and some 'subvector' $\vec{Z}$ of $\vec{Y}$ with corresponding subvector $\vec{E}$ of $\vec{D}$.* The inductive proof (on C) is entirely mechanical.

Then, using this for the case at hand, the case (a) would give $\nabla_i \vec{C}'/\vec{Y}'$ as a subcommand of $\nabla_j \vec{C}/\vec{Y}$ for some j, and so immediately gives (i) or (ii). The case (b) would force the C_1 to be either a call command or a ∇-command, in order to get a ∇-command after the substitution. In the first case, it must be a $callZ_i$, and so a $callY_j$, and so after the substitution we get case (i). In the second case, we use the fact that we are dealing with simple recursion (this is essential, the result fails in the non-simple case). So C_1 has no free procedure variables and substituting has no effect. Thus we get a subcommand of C_ℓ itself, and (ii) holds.

Deduction of Theorem CP from CP5 and CP6. Proceed by induction on D_1 to show $\vdash (F \to [D_1]G)$ for all $1^{\underline{\text{st}}}$order F and G for which that wfs is true in $\mathbf{N}$. 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 $\mathcal{D} \in \Gamma_{D_1}$.

Any such $\mathcal{D}$ has the form $F_2 \to [D_2]G_2$ for $1^{\underline{\text{st}}}$order F_2 and G_2, is $\mathsf{tt}\mathbf{N}$, and D_2 has the form $\nabla_j \vec{C}/\vec{Y}$ and is a subcommand of D_1. Thus in all but one case, the induction on D_1 gives $\vdash \mathcal{D}$ as required.

In that case, for some $\vec{C}, \vec{Y}$ and j, we have $D_1 = D_2 = \nabla_j \vec{C}/\vec{Y}$ and

$$\mathcal{D} = (\vec{x} \approx \vec{z} \ \to \ [\nabla_j \vec{C}/\vec{Y}]A(\nabla_j \vec{C}/\vec{Y}, \vec{x} \approx \vec{z})) = \mathcal{D}_j \ \ \text{(say)}.$$

Do all j simultaneously—i.e. we'll show $\vdash \bigwedge_j \mathcal{D}_j$. This is done by applying the instance of $(\mathsf{RCN})_{[\,]}$ for which $\mathcal{A}_j = \vec{x} \approx \vec{z}$, $\mathcal{B}_j = A(\nabla_j \vec{C}/\vec{Y}, \vec{x} \approx \vec{z})$ and

$$\Omega = \{\ \vec{x} \approx \vec{z} \ \to \ [\nabla_i \vec{C}'/\vec{Y}']A(\nabla_i \vec{C}'/\vec{Y}', \vec{x} \approx \vec{z}) \mid$$

$$\nabla_i \vec{C}'/\vec{Y}' \text{ is a subcommand of a bit of } \vec{C}\ \}\ .$$

Verifying the main antecedent in this instance of $(\mathsf{RCN})_{[\,]}$ consists merely of noting the conclusion of **CP6**.

As for the other antecedent, all $\mathcal{C} \in \Omega$ are true in $\mathbf{N}$ and have the form $F_3 \to [D_3]G_3$ in which D_3 is a *proper* subcommand of D_1 (because $D_3 = \nabla_i \vec{C}'/\vec{Y}'$, a subcommand of a bit of $\vec{C}$, and $D_1 = \nabla_j \vec{C}/\vec{Y}$). And so we have $\vdash \mathcal{C}$ for all $\mathcal{C} \in \Omega$ by the inductive hypothesis, giving the other antecedent, and completing the proof.

5. Technical results and proof of CT.

Lemma CT1. *For all $\vec{C}/\vec{Y}$, all commands D, $1^{\underline{\text{st}}}$order formulas G, and variables $w \notin \vec{C} \cup D \cup G$, there is a $1^{\underline{\text{st}}}$order formula $B(w, \vec{C}/\vec{Y}, D, G)$—("$B$" for "before" rather than "pre")—such that*

$$B(w, \vec{C}/\vec{Y}, D, G)\ \mathsf{tt}@s \iff < D_{\vec{E}/\vec{Y}} > G\ \mathsf{tt}@s \quad \text{with} \quad \vec{E} = \overrightarrow{(\vec{C}/\vec{Y})^{<s(w)>}}.$$

For the proof, see that of 23.0 in **[sing]**.

We don't here use the notation $F(t)$, elsewhere denoting F with all free occurrences of some-or-other variable replaced by the term t. The "w" in $B(w, \vec{C}/\vec{Y}, D, G)$ is there for a different reason: to indicate which variable is the 'special one', playing a 'counting role' in its semantics, as defined by the display in the lemma. See the remarks after 23.0 in **[mutu]** for information which may be helpful to orient oneself to the semantics here.

Lemma CT2. *This is identical to* **CP2** *except for replacing $[C]$ by $< C >$ for each command C, and replacing $\vdash'$ by $\vdash^w$.*

The proof is virtually identical also.

In each of **CT3** to **CT6** below, a $\vec{C}/\vec{Y}$, a variable w, and strings of variables $\vec{x}$, $\vec{z}$ are involved.

Let us just use $B(\vec{C}/\vec{Y}, j)$ to denote $B(w, \vec{C}/\vec{Y}, callY_j, \vec{x} \approx \vec{z})$.

This notation is only used with $\vec{x}$ and $\vec{z}$ disjoint strings of distinct variables, with all variables in $\vec{C}$ from $\vec{x}$, and with one more new variable w.

Lemma CT3. *Under the conditions listed below, each of the following four $1^{\underline{\text{st}}}$order formulas is true in* **N**.

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

(b) $B(w, \vec{C}/\vec{Y}, E, G) \rightarrow \exists \vec{z}(B(w, \vec{C}/\vec{Y}, E, \vec{x} \approx \vec{z}) \wedge |G|^{\vec{x} \to \vec{z}})$;

(ii) $\neg |B(\vec{C}/\vec{Y}, i)|^{w \to 0}$;

(iii) $|B(\vec{C}/\vec{Y}, i)|^{w \to w+1} \rightarrow B(w, \vec{C}/\vec{Y}, C_i, \vec{x} \approx \vec{z})$.

Assumptions: *We are given $\vec{C}/\vec{Y}$, and matching disjoint lists $\vec{x}$ and $\vec{z}$ of pairwise distinct variables such that all variables in $\vec{C}$ are from $\vec{x}$, plus a*

variable w not from $\vec{x} \cup \vec{z}$. In (i)(a), *assume $F \rightarrow\!< \nabla_i \vec{C}/\vec{Y} > G$ is true in* **N**, *with no variables from $\vec{z}$ in the* $1^{\underline{\text{st}}}$*order formulas F and G in both parts of* (i), *and E any command in* (i)(b).

The proof of all these semantic facts is somewhat lengthy (but entirely a question of persistence, not ingenuity). See the proof of 23.4 in [**mutu**].

Lemma CT4. *For any* $1^{\underline{\text{st}}}$*order F and G, any $\vec{C}/\vec{Y}$, and matching disjoint lists $\vec{x}$ and $\vec{z}$ of pairwise distinct variables such that all variables from $\vec{C}$ are in $\vec{x}$, if* $(F \rightarrow\!< \nabla_i \vec{C}/\vec{Y} > G)$ ttN , *then*

$$\{(\exists w\ B(\vec{C}/\vec{Y}, i)) \rightarrow\!< \nabla_i \vec{C}/\vec{Y} > \vec{x} \approx \vec{z}\} \quad \vdash^w \quad (F \rightarrow\!< \nabla_i \vec{C}/\vec{Y} > G)\ .$$

Proof that CT3(i) **implies CT4.** First we prove the special case when no $\vec{z}$-variables occur in F nor in G. Here is a slightly abbreviated $\vdash^w$-derivation for that :

$(\exists w\ B(\vec{C}/\vec{Y}, i)) \rightarrow\!< \nabla_i \vec{C}/\vec{Y} > \vec{x} \approx \vec{z}$ (premiss)

$\exists w\ B(\vec{C}/\vec{Y}, i) \wedge |G|^{\vec{x} \rightarrow \vec{z}} \ \rightarrow \ < \nabla_i \vec{C}/\vec{Y} > (\vec{x} \approx \vec{z} \wedge |G|^{\vec{x} \rightarrow \vec{z}})$
$\mathsf{(AX)}_{<\rightarrow>}$ and $\mathsf{(MP)}$

$\vec{x} \approx \vec{z} \wedge |G|^{\vec{x} \rightarrow \vec{z}} \ \rightarrow \ G$ $\mathsf{(ORAC)}$

$< \nabla_i \vec{C}/\vec{Y} > (\vec{x} \approx \vec{z} \wedge |G|^{\vec{x} \rightarrow \vec{z}}) \ \rightarrow \ < \nabla_i \vec{C}/\vec{Y} > G$ $\mathsf{(UNAR)}_{< >}$

$\exists w B(\vec{C}/\vec{Y}, i) \wedge |G|^{\vec{x} \rightarrow \vec{z}} \ \rightarrow \ < \nabla_i \vec{C}/\vec{Y} > G$ $\mathsf{(HYSY)}$

$\exists \vec{z}(\exists w B(\vec{C}/\vec{Y}, i) \wedge |G|^{\vec{x} \rightarrow \vec{z}}) \ \rightarrow \ < \nabla_i \vec{C}/\vec{Y} > G$ $\mathsf{(PRE)}^{\vec{z}}$

$F \rightarrow \ \exists \vec{z}(\exists w B(\vec{C}/\vec{Y}, i) \wedge |G|^{\vec{x} \rightarrow \vec{z}})$ $\mathsf{(ORAC)}$ and **CT3**(i)(a)

$F \rightarrow\!< \nabla_i \vec{C}/\vec{Y} > G$ $\mathsf{(HYSY)}$

Now the general case is deduced from this exactly as in **CP4**.

Lemma CT5. *Suppose given a command D_1, and $1^{\underline{\text{st}}}$order formulas F and G, such that $(F \rightarrow\!< D_1 >\! G)$ ttN. For each i and $\vec{C}/\vec{Y}$ such that $\nabla_i\vec{C}/\vec{Y}$ 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 $\vec{C}$ are in $\vec{x}$. Let the set Λ_{D_1} consist of all formulas (one for each such i and $\vec{C}/\vec{Y}$)*

$$(\exists w\ B(\vec{C}/\vec{Y}, i)) \ \rightarrow \ <\nabla_i\vec{C}/\vec{Y}>\vec{x} \approx \vec{z}\ .$$

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

Proof that (CT4 plus CT2) imply CT5. This proceeds by induction on commands D_1, which we shall re-name as just D, since there is no danger here (only in the proof of **CT!**) of confusing it with the D in **CT6**.

When $D = callX_j$: By validity of $(\mathsf{AX})_{<call>}$, the wfs $<callX_j> G$ $\mathsf{ff}@s$ for all s. So the truth in $\mathbf{N}$ of $F \rightarrow\!< callX_j >\! G$ gives us that F $\mathsf{ff}@s$, and so $\neg F$ $\mathsf{tt}@s$, for all s. Now (ORAC) and $(\mathsf{P})_3$ give us a 2-line derivation witnessing $\vdash^w (F \rightarrow\!< callX_j >\! G)$, as required.

When $D = y \leftarrow\!\!\div t$: By the soundness of $(\mathsf{AX})_{<\leftrightarrow>}$, we see that

$$(F \rightarrow\!< y \leftarrow\!\!\div t >\! G)\ \mathsf{ttN} \quad \Longrightarrow \quad (F \rightarrow |G|^{y \rightarrowtail t})\ \mathsf{ttN}\ .$$

So here's a little derivation doing the job :

$F \rightarrow \lvert G\rvert^{y \rightarrowtail t}$	(ORAC)
$\lvert G\rvert^{y \rightarrowtail t} \leftrightarrow \ <y \leftarrow\!\!\div t> G$	$(\mathsf{AX})_{<\leftrightarrow>}$
$\lvert G\rvert^{y \rightarrowtail t} \rightarrow \ <y \leftarrow\!\!\div t> G$	$(\leftrightarrow)$
$F \rightarrow\!< y \leftarrow\!\!\div t >\! G$	(HYSY)

When $D = \nabla_i\vec{C}/\vec{Y}$: The set Λ_D in this case contains the element

$$(\exists w\ B(\vec{C}/\vec{Y}, i)) \ \rightarrow \ <\nabla_i\vec{C}/\vec{Y}>\vec{x} \approx \vec{z}\ ,$$

so **CT4** gives the result immediately.

When $D = (D'; D'')$ or $ite(H)(D')(D'')$: Apply **CT2** with $\Gamma_1 = \Lambda_{D'}$ and $\Gamma_2 = \Lambda_{D''}$, noting that $\Lambda_D = \Lambda_{D'} \cup \Lambda_{D''}$ for both types of D here. Use the inductive hypotheses for D' and D'' as the hypotheses in **CT2**. The latter's conclusions are exactly the required results.

Lemma CT6. *For each $\vec{C}/\vec{Y}$, pair of matching disjoint lists $\vec{x}$ and $\vec{z}$ of pairwise distinct variables such that all variables from $\vec{C}$ are in $\vec{x}$, and variable $w \notin \vec{x} \cup \vec{z}$, the finite set Ω of wfs's below, which are all* ttN*, is such that, for all* $1^{\underline{\text{st}}}$*order formulas G and subcommands D of bits of $\vec{C}$, we have*

$$\Omega \ \cup \ \{ \ B(\vec{C}/\vec{Y}, i) \rightarrow < \nabla_i \vec{C}/\vec{Y} > \vec{x} \approx \vec{z} \mid 1 \leq i \leq k \ \} \ \ \vdash^w$$

$$B(w, \vec{C}/\vec{Y}, D, G) \rightarrow < D_{\vec{E}/\vec{Y}} > G \ , \ \ \text{where } \vec{E} = \overrightarrow{\nabla_* C/Y} \ .$$

The Ω which usefully does the trick is the set of all wfs's
$B(w, \vec{C}/\vec{Y}, \nabla_j \vec{C}'/\vec{Y}', \vec{x} \approx \vec{z}) \rightarrow \ \ < (\nabla_j \vec{C}'/\vec{Y}')_{\vec{E}/\vec{Y}} > \vec{x} \approx \vec{z}$ *, as j and $\vec{C}'/\vec{Y}'$ range over all pairs for which $\nabla_j \vec{C}'/\vec{Y}'$ is a subcommand of a bit of $\vec{C}$.*

The lemma above will only be used in the case $D = C_j$ and $G = \vec{x} \approx \vec{z}$; but the general case seems needed for seeing how to prove it—namely, by induction on D, and with one inductive case really needing general G. Note that the subscript in $(\nabla_j \vec{C}'/\vec{Y}')_{\vec{E}/\vec{Y}}$ isn't needed as long as we are working with simple recursion, since then, $\nabla_j \vec{C}'/\vec{Y}'$ has no free occurrences of $callX$ for any procedure variable X, much less any of the form $callY_i$ for which to substitute E_i. (They are all of the non-free form $callY_j'$.) But we'll leave the lemma as above, to emphasize that, along with everything else so far in this section, we could just as well be working in $\mathcal{R}ec$ as in $\mathcal{R}ec_-$.

Proof of CT6. This also proceeds by induction on D.

When $D = y \leftarrow\!\!\!\div t$: For any D with no free $callY_j$-subcommands, and twice using that $D_{\vec{E}/\vec{Y}} = D$ for such D and *any* $\vec{E}$, for a certain $\vec{E}_0$ we have

$$(I) \ \ B(w, \vec{C}/\vec{Y}, D, G)\, \mathsf{tt@}s \iff \ < D_{\vec{E}_0/\vec{Y}} > G\, \mathsf{tt@}s \iff \ < D > G\, \mathsf{tt@}s \ ;$$

and

$$(II) \ \ D_{\vec{E}/\vec{Y}} \ = \ D \ .$$

For D here, since $< D > G \longleftrightarrow |G|^{y \rightarrowtail t}$ is ttN, we can take $B(w, \vec{C}/\vec{Y}, D, G)$ to be $|G|^{y \rightarrowtail t}$ by (I). By (II), we only need to establish

$$\vdash^w \ \ (|G|^{y \rightarrowtail t} \rightarrow \ \ < y \leftarrow\!\!\!\div t > G) \ ,$$

which is immediate from $(\mathsf{AX})_{<\leftrightarrow>}$.

When $D = callX_i$ where $X_i \neq Y_j$ for any j: Here again $D_{\vec{E}/\vec{Y}} = D$, so what must be derived has the form $F \rightarrow\!< callX_i > G$. But $\neg < callX_i > G$ is immediately derivable from $(\mathsf{AX})_{<call>}$, so we then can propositionally derive $F \rightarrow\!< callX_i > G$.

This last case won't actually occur since we are allowing only *simple* recursion, but we have included it to show that, as mentioned above, so far we could be working in $\mathcal{R}ec$ rather than just $\mathcal{R}ec_{-}$.

The next two cases are done in a manner which is very nearly the same as the proof of **CT4**.

When $D = callY_j$: First prove the special case when no $\vec{z}$-variables occur in G by giving $\vdash^w$-derivation which, on the penultimate line, uses (ORAC) and **CT3**(i)(b) with $E = callY_j$, but otherwise is nearly the same as the one in **CT4**. Then do the general case as in **CT4**/**CP4**.

When $D = \nabla_j \vec{C}'/\vec{Y}'$: It suffices to prove

$$B(w, \vec{C}/\vec{Y}, \nabla_j \vec{C}'/\vec{Y}', \vec{x} \approx \vec{z}) \rightarrow \ < (\nabla_j \vec{C}'/\vec{Y}')_{\vec{E}/\vec{Y}} > \vec{x} \approx \vec{z} \vdash^w$$

$$B(w, \vec{C}/\vec{Y}, \nabla_j \vec{C}'/\vec{Y}', G) \rightarrow \ < (\nabla_j \vec{C}'/\vec{Y}')_{\vec{E}/\vec{Y}} > G \ ,$$

since on the left of the $\vdash^w$ we have an element of Ω. (This is where D being a subcommand of a bit of $\vec{C}$ is used.) First prove the special case when no $\vec{z}$-variables occur in G by giving $\vdash^w$-derivation which, on the penultimate line, uses (ORAC) and **CT3**(i)(b) with $E = \nabla_j \vec{C}'/\vec{Y}'$, but otherwise is nearly the same as the one in **CT4** and the one not spelled out just above. Then do the general case as in **CT4**/**CP4**.

When $D = (D_1; D_2)$ or $ite(H)(D_0)(D_1)$: For the first, letting

$$B_* \ := \ B(w, \vec{C}/\vec{Y}, D_1, B(w, \vec{C}/\vec{Y}, D_2, G)) \ ,$$

show that we may take

$$B(w, \vec{C}/\vec{Y}, (D_1; D_2), G) \ = \ B_* \ ,$$

by showing

$$B(w, \vec{C}/\vec{Y}, (D_1; D_2), G) \ \longleftrightarrow \ B_* \quad \mathsf{ttN} \ .$$

The proof then proceeds rather mechanically. All details for this and the next paragraph may be found in the proof of 23.1 in [**mutu**].

Make a parallel argument for the second, first letting

$$B_{**} \ := \ (H \to B_+) \wedge (\neg H \to B_-) \ ,$$

where

$$B_+ \ := \ B(w, \vec{C}/\vec{Y}, D_0, G) \ \text{ and } \ B_- \ := \ B(w, \vec{C}/\vec{Y}, D_1, G) \ ,$$

and showing that we may take

$$B(w, \vec{C}/\vec{Y}, ite(H)(D_0)(D_1), G) \ = \ B_{**} \ ,$$

by proving

$$B(w, \vec{C}/\vec{Y}, ite(H)(D_0)(D_1), G) \ \longleftrightarrow \ B_{**} \qquad \mathsf{ttN} \ .$$

Deduction of Theorem CT from CT5, CT6 and (ii)**,** (iii) **of CT3**. To show $\vdash (F \to < D_1 > G)$ from its truth in **N**, proceed by induction on the length of the string D_1.

By **CT5** it suffices to establish $\ \vdash \mathcal{D}\ $ for all $\mathcal{D} \in \Lambda_{D_1}$. Combined with (PRE), that says it suffices to prove

$$\vdash (B(\vec{C}/\vec{Y}, \ell) \ \to \ \ < \nabla_\ell \vec{C}/\vec{Y} > \vec{x} \approx \vec{z}) \qquad\qquad (*)$$

whenever $\nabla_\ell \vec{C}/\vec{Y}$ is a subcommand of D_1, with some $\vec{x}$ and $\vec{z}$ chosen as in **CT5**.

For D_1 of small length, no such subcommand exists, so the induction starts easily (vacuously!)

For the inductive step, we prove $(*)$ simultaneously for all ℓ, $\vec{C}/\vec{Y}$, $\vec{x}$ and $\vec{z}$ for which $1 \leq \ell \leq k$, with $\vec{C} = C_1, \cdots, C_k$ having all variables in $\vec{x}$, and for which $\nabla_m \vec{C}/\vec{Y}$ is a subcommand of D_1 for at least one m equal to one of those ℓ (and thus the sum of the lengths of the C_i is less than the length of D_1). Do this simultaneously for all ℓ, with fixed $\vec{C}/\vec{Y}$, $\vec{x}$ and $\vec{z}$, after picking any 'new' variable w, to be used in applying an instance of $(\mathsf{RCN})_{< >}$ in which, for all ℓ,

$$\mathcal{A}_\ell = B(\vec{C}/\vec{Y}, \ell) \ \text{ and } \ \mathcal{B}_\ell =< \nabla_\ell \vec{C}/\vec{Y} > \vec{x} \approx \vec{z} \ .$$

Now we verify that rule's antecedents in the order presented in its 'numerator', and specify which Ω will be employed.

(i) By **CT3**(ii), the oracle gives a derivation of $\neg|\mathcal{A}_j|^{w\mapsto 0}$, which is $\neg|B(\vec{C}/\vec{Y},j)|^{w\mapsto 0}$.

(ii) Taking

$$\Omega \;:=\; \{\, B(w,\vec{C}/\vec{Y},\nabla_j\vec{C}'/\vec{Y}',\vec{x}\approx\vec{z}) \to \;\; <(\nabla_j\vec{C}'/\vec{Y}')_{\vec{E}/\vec{Y}} > \vec{x}\approx\vec{z} \;\mid$$

$$\nabla_j\vec{C}'/\vec{Y}' \text{ is a subcommand of a bit of } \vec{C}\,; \vec{E} = \overrightarrow{\nabla_* C/Y}\,\}\,,$$

for this antecedent we need to show $\vdash \mathcal{C}$ for all $\mathcal{C}\in\Omega$. This comes from the inductive hypothesis. Each $\mathcal{C}$ has the form $F_0 \to\!< \nabla_j\vec{C}'/\vec{Y}' > G_0$, for some $1^{\underline{\text{st}}}$order F_0 and G_0. (Note that, for this to hold, we need our standing assumption of being in $\mathcal{R}ec_-$, not the whole $\mathcal{R}ec$.) The formulas F_0 and G_0 are such that $\mathcal{C}$ is true in $\mathbf{N}$ (by the general definition of B). Because $\nabla_j\vec{C}'/\vec{Y}'$ is a subcommand of a bit of $\vec{C}$, and $\nabla_m\vec{C}/\vec{Y}$ a subcommand of D_1, the first has length strictly less than the last. And so the induction applies as required.

(iii) Finally, for checking the main antecedent in this instance of the total correctness recursion rule, we prove, for each j, that

$$\Omega \;\cup\; \{\; B(\vec{C}/\vec{Y},i) \to\!< \nabla_i\vec{C}/\vec{Y} > \vec{x}\approx\vec{z} \mid 1\le i\le k \;\} \;\; \vdash^w$$

$$|B(\vec{C}/\vec{Y},j)|^{w\mapsto w+1} \to\!< \nabla_j\vec{C}/\vec{Y} > \vec{x}\approx\vec{z}\,.$$

Now Ω is the correct one for applying **CT6**, in which we take $D = C_j$ and $G = \vec{x}\approx\vec{z}$, so at least it gives a $\vdash^w$-derivation of

$$B(w,\vec{C}/\vec{Y},C_j,\vec{x}\approx\vec{z}) \;\to\; <(C_j)_{\vec{E}/\vec{Y}} > \vec{x}\approx\vec{z}\,, \;\; \text{where } \vec{E} = \overrightarrow{\nabla_* C/Y}\,.$$

By $(\mathsf{AX})_{<\nabla>}$, in this last display, we can replace $<(C_j)_{\vec{E}/\vec{Y}} > \vec{x}\approx\vec{z}$ by $<\nabla_j\vec{C}/\vec{Y} > \vec{x}\approx\vec{z}$, which is half the battle. The other half is immediate from (HYSY), the oracle, and **CT3**(iii). It replaces

$$B(w,\vec{C}/\vec{Y},C_j,\vec{x}\approx\vec{z}) \;\text{ by }\; |B(\vec{C}/\vec{Y},j)|^{w\mapsto w+1}\,.$$

This completes the proof of Theorem **CT**. Along with the proof of Theorem **CP** in the previous section, and in view of Theorem **H** from Section 3, the proof is finally finished of the completeness of the dynamic logic

system presented in Section 1. Extracting a more traditional 'Hoare logic' purely for total correctness (resp. partial correctness), and proving its Cook-completeness, should be a routine job, by inspection of Section 5 (resp. Section 4) details.

References.

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

[**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.

[**BP**] de Boer, Frank S. and Pierik, Cees *How to Cook a Complete Hoare Logic for Your Pet OO Language.* FMCO 2003: 111-133.

[**BT1**] Bergstra, Jan A. and Tucker, J. V. *Expressiveness and the Completeness of Hoare's Logic.* J. Comput. Syst. Sci. 25(3): 267-284 (1982).

[**BT2**] Bergstra, Jan A. and Tucker, J. V. *Two Theorems About the Completeness of Hoare's Logic.* Inf. Process. Lett. 15(4): 143-149 (1982).

[**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.

[**C-G-H**] Clarke, E.M. Jr., German, S.M. and Halpern, J.Y. *Effective Axiomatizations of Hoare Logics.* J. ACM 30(3), 1983, 612-636.

[**Clk**] Clarke, E.M. Jr. *Programming Language Constructs for which it is Impossible to Obtain Good Hoare-like Axioms.* J. ACM 26, 1979, 129-147.

[**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.

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

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

[**Gr1**] Grabowski, Michal *On Relative Completeness of Hoare Logics.* Inf. and Control (66), 1985, 29-44.

[**Gr2**] Grabowski, Michal *On the Status of Proving Program Properties in Effective Interpretations.* Theor. Comp. Sci. 120 (1993) 69-81.

[**Gr-Hu**] Grabowski, M. and Hungar, H. *On the Existence of Effective Hoare Logics.* Proc. 3rd Annual Symp. on Logic in Computer Science (LICS '88), IEEE, Edinburgh, 1988, 428-435.

[**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.

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

[**Kln1**] Kleymann, Thomas *Hoare Logic and VDM: Machine-Checked Soundness and Completeness Proofs.* PhD dissertation, University of Edinburgh, 1998. http://www.lfcs.inf.ed.ac.uk/reports/98/ECS-LFCS-98-392/

[**Kln2**] Kleymann, Thomas *Hoare Logic and Auxiliary Variables.* Formal Aspects of Computing 11 (1999), 541-566.

[**KT**] Kozen, Dexter and Tiuryn, Jerzy *On the completeness of propositional Hoare logic.* Information Sciences, 139 (2001), 187-195.

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

[**Musk**] Muskens, R. *Program Semantics and Classical Logic.* CLAUS report 86, Universitat des Saarlandes, January 1997.

[**mutu**] Hoffman, P. *Imperative Mutual Simple Recursion Proof Systems.* http://www.math.uwaterloo.ca/PM(underscore)Dept/Homepages/Hoffman/15-mutualrec-webpage-fix.pdf

[**Nip1**] 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 *Hoare Logics in Isabelle/HOL.* in: *Proof and System Reliability*, 2002.

[**NvO**] Nipkow, Tobias and von Oheimb, David *Hoare Logic for NanoJava: Auxiliary Variables, Side Effects and Virtual Methods Revisited.* in: *FME 2002* pp. 89-105 of Lecture Notes in CS # 2391, Springer, 2002.

[**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. *Depth-Measure Semantics and Partial-Correctness Rules for Imperative Recursion.*
http://www.math.uwaterloo.ca/PM(underscore)Dept/Homepages/Hoffman/new(underscore)18-Dept-Measure-Semantics.pdf

[**sing**] Hoffman, P. *Deterministic Dynamic Logic Imperative Recursive Programming Proof Systems.*
http://www.math.uwaterloo.ca/PM(underscore)Dept/Homepages/Hoffman/14-zrec-paper-webpage.pdf

[**Sok**] Sokolowski, Stefan *Total Correctness for Procedures.* in:
[J. Gruska-ed] *Sixth Mathematical Foundations of Computer Science (Tatranská Lomnica)*, pp. 475-483 of Lecture Notes in CS # 53, Springer, 1977.

[**vOh1**] von Oheimb, David *Hoare logic for Java in Isabelle/HOL.* Concurrency and Computation: Practice and Experience 13(13) (2001), 1173-1214.

[**vOh2**] von 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**] Wang, Arne *An Axiomatic Basis for Proving Total Correctness of GOTO-Programs.* Bit 16(1976), 88-102.