Challenge Examples and a Conjecture re: Correctness of Imperative Recursive Programs

(both Partial and Total)

Peter Hoffman

Pure Mathematics, University of Waterloo

Abstact. *See intro.*

Removal is desired of the hypothesis of recursive command *simpleness*, apparently needed in [**infocom**] for the proof of Cook-completeness [**C**] of a (partial and total) correctness proof system. The present paper is a stop-gap whose purpose is to present a modified proof system that we conjecture to 'work' in complete generality. See the definition of the command language $\mathcal{R}ec$ in [**infocom**], pp. 7-8, and of the system there. That system is proved Cook-complete only for the sublanguage $\mathcal{R}ec_{-}$ in which all the non-simple recursions have been removed from $\mathcal{R}ec$. Here we'll remain in the entire $\mathcal{R}ec$.

It is the author's opinion that there is slightly more than just entertainment value in the following analogy : Passing from (simple recursion on a single procedure variable) to (simple recursion on several variables—i.e. simple mutual recursion) is analogous to passing from (real-valued functions of a single real variable) to (vector-valued functions of a single real variable). More 'ominously', passing from (simple recursion on a single procedure variable) to (general [non-simple] recursion on a single variable) is analogous to passing from (real-valued functions of a single real variable) to (real-valued functions of several real variables). The main point is that the latter two are rather thornier questions, whereas the earlier two are perhaps more matters of messier notation than of any need for really new ideas.

In [**dpth**] we give a fairly thorough treatment of a second set of difficulties associated with the passage from simple to non-simple recursion. These difficulties require a careful treatment of substitution (so what else is new?!!), to avoid 'procedure-variable clashes', and consequently require a somewhat more general way of defining command semantics, using a recursion-depth-measure with variable depths for different recursions within a command (plus proof that we have re-invented the wheel—no, I meant: re-invented fixed-point semantics, so lusted after by the denotationalizers). The conjecture referred to in the title here has to do with a modification of the system in [**infocom**], mainly by introducing a more general rule for partial correctness of recursive commands. That conjecture would claim the new system to be Cook-complete. If true, any thorough proof of it will need to make use of something like the technicalities on substitution and recursion-depth just referred to. However, here we have suppressed everything on that, in order to concentrate on the crucial aspects of the new rule.

The bulk of this paper has to do with two specific examples of non-simple recursion. We give a fairly detailed description of how to use the new rule to derive the appropriate partial correctness for one example, and to use the old total correctness rule in a new way to deal with a slightly simpler but related example in that more difficult context. We also challenge the reader to attempt the same, but using his/her favourite system (and we cheekily suggest that this attempt will land the reader in excessive difficulty in the partial case, and in a complete lack of success in the total case).

Included here also is a discussion of how to prove soundness of the new partial correctness rule. But we deal only with a special case, and, as mentioned above, suppress some (in my view, subtle) technicalities related to substitution. Clearly, if and when a proof of Cook-completeness surfaces, a convincing write-up will need to exhibit a great deal of care on several matters along these lines.

But the rough discussion just above, and its expansion just below, ought to generate confidence in there being a *sound* system along the lines suggested above. And the examples referred to two paragraphs above, when fleshed out below, will hopefully have isolated most of the extra ideas needed to concoct a proof for Cook-completeness.

Everything in this paper concentrates on recursion with a single procedure variable, as opposed to mutual recursion. We have faith in our analogy above, and expect that extension to the mutual case will be straightforward, if possibly messy. But we make no *claims* in that direction, in contradistinction to one tradition in the subject.

1. Some useful examples of breathtaking banality.

We are counting on the reader to get almost all the needed technical definitions from [**infocom**]. The final line of this paper gives a generalization of the rule for partial correctness of recursive commands, called $(\mathsf{RCN})_{[\ ]}$ there. Here is essentially a special case of the old rule, before the generalization begins to be discussed in Section 2:

$$\frac{\vdash \mathcal{C}\ \forall\mathcal{C}\in\Omega \quad , \quad \Omega\cup\{[\nabla XC]G\}\ \vdash'\ [|C|^{X\mapsto\nabla XC}]G}{\vdash [\nabla XC]G} \quad .$$

In this, $|C|^{X\mapsto\nabla XC}$ denotes substituting command ∇XC into command C for free occurrences of procedure variable X. And Ω is a finite set of wfd's $\mathcal{C}$ which don't have ∇XC in them, nor does G since it's just a 1st order formula. Actually G can more generally be a wfd (not merely a 1st order formula), again as long as it doesn't have ∇XC in it, but that stronger rule is overkill from the completeness point-of-view in that paper. The command ∇XC is 'recursion on X, using a body which is the command C'. This would be banal if C had no subcommand $callX$ which could potentially cause the recursion to 'kick-in'.

Our first example is exactly that banality, and is hardly surprising:

Suppose that C has no free X. Then $\vdash [C]G \implies \vdash [\nabla XC]G$.

To prove this using the above-displayed rule, the (usually crucial) premiss, namely $[\nabla XC]G$, in the main (the 2nd) antecedent of the rule can be utterly ignored, and the conclusion in that main antecedent of the rule, which reduces to just $[C]G$ because there are no free X for which to substitute ∇XC, can simply be taken as the unique member of what will be a singleton set Ω. But now the main antecedent is utterly trivial, with no need to know anything at all about the 'auxiliary' proof system $\vdash'$. And the first antecedent is our assumption in the example.

So this example gives evidence that the occurrence of potentially non-empty Ω in the rule cannot be dispensed with (though [**sing**], 2nd section appendix, has plenty of other such examples with slightly less banality).

Now consider $\nabla Y\nabla X callY$. This hardly exciting nested recursive command of course diverges on any input. Because of the free Y in the body of the ∇X, it is non-simple, and about as "simple" a non-simple as we could get for which the command itself has no free procedure variables. Because of non-simpleness, the completeness theorem in [**infocom**] strictly speaking does not apply directly, and we shall see that applying the rules there gets us into a bit of a vicious circle in trying to derive $[\nabla Y \nabla X callY]1 \approx 0$ without any new rule.

The main antecedent of the rule above would have the form

$$\Omega \cup \{[\nabla Y \nabla X callY]1 \approx 0\} \ \vdash' \ [\nabla X \nabla Y \nabla X callY]1 \approx 0 \ .$$

For this, we could again ignore the usually crucial half $\{\nabla Y \nabla X callY]1 \approx 0\}$ and simply take Ω to be the singleton $\{[\nabla X \nabla Y \nabla X callY]1 \approx 0\}$. The 1st antecedent then needs to be derived. For this, the general example in the previous paragraph, since the body, $\nabla Y \nabla X callY$, of $\nabla X \nabla Y \nabla X callY$ has no free X, can be considered. But it requires that we derive the original one, $[\nabla Y \nabla X callY]1 \approx 0$, first.

So all we seem to be able to establish is an equivalence

$$\vdash \ [\nabla Y \nabla X callY]1 \approx 0 \quad \Longleftrightarrow \quad \vdash \ [\nabla X \nabla Y \nabla X callY]1 \approx 0$$

Very soon below, we'll see how to trivially derive both these from a very special case of a new rule.

Since the examples are so banal, it is unsurprising that the formal derivations also are. But these examples at least deter one from attempting to eliminate the uneliminatable.

2. New rules.

The fully general rule, of which we wrote down a special case in the previous section, simply has "$F \rightarrow$ " in front of each "$[-]G$" in the rule. We will delay here to the end of the paper writing down the general form of our new, generalized rule. Here are two special cases:

$$\frac{\vdash \mathcal{C} \ \forall \mathcal{C} \in \Omega \ , \quad \Omega \cup \{F_1 \rightarrow [\nabla X_1 C_1]G_1, F_2 \rightarrow [\nabla X_2 C_2]G_2\} \vdash' (F_1 \rightarrow [|C_1|^{X_1 \mapsto \nabla X_1 C_1}]G_1) \wedge (F_2 \rightarrow [|C_2|^{X_2 \mapsto \nabla X_2 C_2}]G_2)}{\vdash \ (F_1 \rightarrow [\nabla X_1 C_1]G_1) \ \wedge \ (F_2 \rightarrow [\nabla X_2 C_2]G_2)} \ ;$$

$$\frac{\vdash \mathcal{C} \ \forall \mathcal{C} \in \Omega \ , \quad \Omega \cup \{ \ [\nabla X_i C_i]G_i \mid i \in \text{index set} \ \} \ \vdash' \ \wedge_i \ [|C_i|^{X_i \mapsto \nabla X_i C_i}]G_i}{\vdash \ \wedge_i \ [\nabla X_i C_i]G_i} \ .$$

In the latter one, $\wedge_i$ is just 'iterated anding' over some finite index set of "i"s. So the general form of the new rule is pretty obvious, with one caveat: we need to be more careful and expansive about those substitutions of commands for free occurrences of procedure variables, since it is exactly in the non-simple case that variable clashes can occur.

All we shall do in this section is to indicate how one would prove soundness, by treating the intersection of the above two special cases, and with Ω empty, namely, soundness of

$$\frac{\{[\nabla X_1 C_1]G_1\ ,\ [\nabla X_2 C_2]G_2\}\ \vdash'\ [|C_1|^{X_1\mapsto\nabla X_1 C_1}]G_1\ \wedge\ [|C_2|^{X_2\mapsto\nabla X_2 C_2}]G_2}{\vdash\ [\nabla X_1 C_1]G_1\ \wedge\ [\nabla X_2 C_2]G_2}\ .$$

Before doing that, let's return briefly to the latter example in the previous section. The rule displayed above shows

$$\vdash\ \ [\nabla Y \nabla X call Y]1 \approx 0\ \wedge\ [\nabla X \nabla Y \nabla X call Y]1 \approx 0$$

very trivially. The antecedent becomes virtually tautologous, since the two wfds to be $\vdash'$-derived are exactly the two in the premiss set in reverse order!

Returning to the question of soundness for this rule, having suppressed the subtle technicalities concerning substitutions, we will then also assume the semantics can be given by a 'fixed-depth'-measure. So what's here is really only convincing, and easy to generalize to the general form of the new rule, in the case of simple recursion. Filling in the necessary fully general details can wait till a proof surfaces of Cook-completeness for the system using the new rules.

For this soundness sketch, as in [**sem**], we'll use purported semantic functions, $\mathcal{M}_n$, one for each natural number n, of commands C and of 'assignments of procedure variables Θ, which produces a set, $\mathcal{M}_n(C,\Theta)$, of ordered pairs of states, (input, output) if you like. But here n is just a natural number (or 'infinity') measuring depth, not a function from the set of procedure variables to the set $\mathbf{N}\cup\{\infty\}$. We wish to show the consequent true in $\mathbf{N}$, given that a derivation as in the antecedent actually exists. Fundamental to this will be the fact that $\vdash'$ preserves $\mathsf{tt}n$. It suffices to show $[\nabla X_i C_i]G_i\mathsf{tt}n\mathbf{N}$ for all 'finite' n. It will be of the essence that we prove this simultaneously for both values of i.

So we need to establish that, if $(s,s')\in\mathcal{M}_n(\nabla X_i C_i,\Phi)$, then $G_i\mathsf{tt}@s'$, which will be done by induction on n.

For $n = 0$, the set $\mathcal{M}_0(\nabla X_i C_i, \Phi)$ is empty by definition, so the induction starts without needing booster cables.

For the inductive step, the inductive assumption allows that $[\nabla X_j C_j] G_j \mathtt{tt} n \mathbf{N}$ for both values of j. From the truth preservation property of $\vdash'$, and the truth of the antecedent, we get $[|C_j|^{X_j \mapsto \nabla X_j C_j}] G_j \mathtt{tt} n \mathbf{N}$ for both values of j. Suppose, fixing one of the two values of i, that $(s, s') \in \mathcal{M}_{n+1}(\nabla X_i C_i, \Phi)$, from which we'll deduce, as required, that $G_i \mathtt{tt}@s'$. The crucial identity needed (and the one which seems to force a more subtle depth-measure when we are truly considering non-simple recursion) will give

$$\mathcal{M}_n([|C_i|^{X_i \mapsto \nabla X_i C_i}] G_i, \Phi) \ = \ \mathcal{M}_n(C_i, \Phi_{X_i \mapsto \mathcal{M}_n(\nabla X_i C_i, \Phi)}) \ .$$

Now (s, s') is in the right-hand side of that identity, by the inductive (on n) definition of $\mathcal{M}_{n+1}(\nabla X_i C_i, \Phi)$ as being precisely that right-hand side. Thus it is also in the left-hand side, which from the above mentioned fact that $[|C_i|^{X_i \mapsto \nabla X_i C_i}] G_i \mathtt{tt} n \mathbf{N}$ gives the required result that $G_i \mathtt{tt}@s'$.

3. The Partial Correctness Example.

Here we shall use the first new rule stated in the previous section to do a program verification for a mildly non-banal example of non-simple recursion.

First we describe the example non-technically, and issue a couple of challenges to program-verifiers, including any who may have taken out patents on their systems, [**Hom**], with claims of vast generality. We also predict a somewhat negative outcome to these challenges (so there may be some issue of "truth in advertising" here which needs to be settled—it's not one of mathematical error as far as I know—the deficiency most obvious is that the command language in [**Hom**] won't even permit nesting of recursions, much less non-simple recursions).

We consider a command ∇XC with nested non-simple recursion. Its more-or-less 'fine-grained' behaviour is to increase by 1 the value of 1st order variable (program variable) x on each 'loop' of the outer recursion (on procedure variable X), and, after that, within the same loop, do two loops of the inner recursion (on Y), each of which reduces the value of program variable y by 1. Then a call to X within the inner recursion starts another loop of the outer recursion. This continues until the value of y becomes 0. This latter fact, plus the fact that x increases by roughly half the input value of y, constitute the 'postcondition' to be 'verified'. (There is a minor glitch in this rough description when y reaches the value 1, if its initial value is odd, but there turns out to be no genuine problem with this.)

Two challenges to experts on previous systems for correctness of imperative recursive programs are the following:

(P) As done below using our system, use your favourite system for partial correctness to derive the partial correctness related to this command (after formulating such a command in your own command language, but with the above fine-grained semantics—in particular, keep the call to X inside the body of the recursion on Y, notation as below—I realize it can be moved out, but if you insist on that, show me how you'd be able to do the same to 'simplicate' *any* non-simple recursion in a cogent and efficient manner—of course, appealing to computability theory, there's no need for any recursions at all, which is hardly the point here). My claim is that, if you give details at about the same level of thoroughness that we do below, those details will be at least twice as 'involved', complicated', ... as ours. And the same would be true for a completely formal version; that is, derivations and line justifications with no gaps left, including whatever of the various adhoceries, such as counting variables, auxiliary variables, adaptation rules, contexts, correctness phrases, etc., which might be required to get your system off the ground.

(T) Try to use your own system (or any other, such as Harel's [**H**]) for total correctness, to derive that stronger correctness statement for the same example, or even for the simpler example whose complete details are given in the next section. My claim here is that you will not be able to even get started (much less finished) on the job. The reason is that, IMHO, no system for total correctness of recursive commands exists, where nesting and non-simpleness are allowed, at least not one which is Cook-complete. In fact, except for my recent work [**infocom**] and Harel's (slightly incomplete, but easily fixable, unfortunately non-self-contained) system, none exists even when only simple recursion is allowed, but nesting can happen.

Now we'll write down the command in a semi-technical manner, just so the challenges above cannot be said to be vague, program variables being understood to take natural number values:

```
Do a recursion on X :
    if y is positive
    thendo (increment x by 1;
              set z to 0;
              Do a recursion on Y :
                  if z is at least 2
                  thendo call X
                  elsedo (increment z by 1;
                              properly decrement y by 1;
                              call Y))
    elsedo quit.
```

The fact that the call of X is within the body of the recursion on Y is what makes this *non-simple*, by definition. The verification to be done is that, at termination, the value of y is 0 and the value of x has been increased by u from its value before execution, where y is either $u+u$ or $u+u+1$ before execution.

Let's get on with careful details of the example, using the precise command language $\mathcal{R}ec$. Let

$$C \ := \ ite(0<y)(x \leftarrow: x+1 \ ; \ z \leftarrow: 0 \ ; \ \nabla Y D)(bugga) \ ,$$

where

$$D \ := \ ite(1<z)(callX)(z \leftarrow: z+1 \ ; \ `y \leftarrow: y-1' \ ; \ callY) \ ,$$

using the 'proper subtraction' command defined on page 29 of [**sing**]. There is an extra program variable in that command, and we won't mention again that this variable must be distinct from all others in any discussion involving the proper subtraction command. We wish to find a derivation for the wfd $F \rightarrow [\nabla XC]G$, where F is $J \wedge x \approx v$ with $J := \ y \approx u+u \vee y \approx u+u+1$, and where G is $y \approx 0 \wedge x \approx u+v$.

Of course, we are taking x, y, z, u and v to be names of distinct 1st order variables, and assuming that the procedure variables X and Y are distinct.

Now let

$$E \ := \ |D|^{X \mapsto \nabla XC} \ = \ ite(1<z)(\nabla XC)(z \leftarrow: z+1 \ ; \ `y \leftarrow: y-1' \ ; \ callY) \ .$$

Thus

$$|C|^{X\mapsto\nabla XC} \;=\; ite(0<y)(x \leftarrow\!\!: x+1 \;;\; z \leftarrow\!\!: 0 \;;\; \nabla YE)(bugga)\ ,$$

and

$$|E|^{Y\mapsto\nabla YE} \;=\; ite(1<z)(\nabla XC)(z \leftarrow\!\!: z+1 \;;\; \text{‘}y \leftarrow\!\!: y-1\text{’} \;;\; \nabla YE)\ .$$

We will find a 1$^{\underline{\text{st}}}$order formula H such that $(H \rightarrow [\nabla YE]G)$ **ttN**, and then use the new rule with consequent

$$(F \rightarrow [\nabla XC]G) \;\wedge\; (H \rightarrow [\nabla YE]G)\ ,$$

thereby deriving double what was asked. We shall therefore check the truth of a main antecedent of the form

$$\Gamma \;\vdash'\; (F \rightarrow [|C|^{X\mapsto\nabla XC}]G) \;\wedge\; (H \rightarrow [|E|^{Y\mapsto\nabla YE}]G)\ , \quad \text{with}$$

$$\Gamma \;:=\; \Omega \;\cup\; \{F \rightarrow [\nabla XC]G\ ,\ H \rightarrow [\nabla YE]G\}\ .$$

We will take the set Ω to be the singleton containing a wfd

$$H_1 \wedge 0 < u \;\rightarrow\; [\text{‘}u \leftarrow\!\!: u-1\text{’}]|H_1|^{u\mapsto u+1}\ .$$

All such wfd's have been shown to be derivable as Example 2, p.46 in [**sing**]. So the other antecedent will need no further comment. We shall delay specifying H_1 till when it is needed near the end of the verification.

After a moderate amount of blood, sweat, tears and calculation, it turns out that the following is the correct choice for H. We won't bother to check directly that $H \rightarrow [\nabla YE]G$ is **ttN**, since that follows from soundness plus one fact we are aiming for—that it is derivable :

$$H \;:=\; J \wedge K \quad \text{with} \quad K \;:=\; L \vee (1 < z \wedge x \approx v)$$

where

$$L \;:=\; (z \approx 0 \wedge y \approx 0 \wedge x \approx v) \vee (z \approx 0 \wedge 0 < y \wedge x \approx v{+}1) \vee (z \approx 1 \wedge y\ od \wedge x \approx v{+}1) \vee (z \approx 1 \wedge y\ ev \wedge x \approx v)\ .$$

Here $y\ od$ and $y\ ev$ are abbreviations for 1$^{\underline{\text{st}}}$order formulas saying that y is odd and even, respectively.

The earlier expansions of $|C|^{X\mapsto\nabla XC}$ and $|E|^{Y\mapsto\nabla YE}$ as 'if-then-else'-commands, together with the axioms $(\mathsf{AX})_{ite}$ and $(\mathsf{AX})_{;}$ of the system $\vdash'$, reduce our job to demonstrating the existence of four derivations, as follows.

$(\text{A})_+ \quad \Gamma \vdash' H\wedge 1<z \;\rightarrow\; [\nabla XC]G$;

$(\text{A})_- \quad \Gamma \vdash' H\wedge\neg 1<z \;\rightarrow\; [z \leftarrow\!\!: z{+}1][\text{`}y \leftarrow\!\!: y{-}1\text{'}][\nabla YE]G$;

$(\text{B})_+ \quad \Gamma \vdash' F\wedge 0<y \;\rightarrow\; [x \leftarrow\!\!: x{+}1][z \leftarrow\!\!: 0][\nabla YE]G$;

$(\text{B})_- \quad \Gamma \vdash' F\wedge\neg 0<y \;\rightarrow\; [bugga]G$.

This last one is quite a trivial application of the oracle, hypothetical syllogism and $(\mathsf{AX})_{\leftarrow:}$, at the end applying the fact that the "do-nothing-command", *bugga*, has the form $x_0 \leftarrow\!\!: x_0$. All but the last of the following 'arrows' is clearly **ttN** :

$$F\wedge\neg 0<y \;\rightarrow\; J\wedge x\approx v\wedge y\approx 0 \;\rightarrow\; y\approx u{+}u\wedge x\approx v\wedge u\approx 0$$
$$\rightarrow\; y\approx 0\wedge x\approx u{+}v\wedge u\approx 0 \;\rightarrow\; G \;\rightarrow\; [bugga]G \;.$$

Proving the existence of a derivation for the other three is done in the following paragraphs.

For $(\text{A})_+$, again here are two 'arrows' which are **ttN** :

$$H\wedge 1<z \;\rightarrow\; J\wedge 1<z\wedge x\approx v \;\rightarrow J\wedge x\approx v \;=\; F \;\;.$$

But $F \rightarrow [\nabla XC]G$ is in Γ, so that does it.

For $(\text{B})_+$, here is a sequence of 'arrows' doing the trick, as explained after the display:

$$F\wedge 0<y \;\rightarrow\; J\wedge x\approx v\wedge 0<y\wedge 0\approx 0 \;\rightarrow\; |J\wedge x\approx v{+}1\wedge 0<y\wedge z\approx 0|^{x\mapsto x+1\;,\;z\mapsto 0}$$
$$\rightarrow\; [x \leftarrow\!\!: x+1][z \leftarrow\!\!: 0](J\wedge x\approx v+1\wedge 0<y\wedge z\approx 0)$$
$$\rightarrow\; [x \leftarrow\!\!: x{+}1][z \leftarrow\!\!: 0]H \;\rightarrow\; [x \leftarrow\!\!: x{+}1][z \leftarrow\!\!: 0][\nabla YE]G \;.$$

The first two 'arrows' are true 1st order number theory formulas. The 'arrow' on the 2nd line is a double application of $(\mathsf{AX})_{\leftarrow:}$. The next one is a double application of $(\mathsf{UNAR})_{[\;\;]}$ together with the oracle. The final 'arrow' is another double application of $(\mathsf{UNAR})_{[\;\;]}$ together with the fact that $H \rightarrow [\nabla YE]G$ is in Γ.

As for (A)$_-$, rather more work is needed, but, except for using an analogue of the lemma at the end of the following section, it's much like what we've been doing. Firstly, we now define the H_1 which occurs as an ingredient in Γ, and also a messy formula L', which agrees with L except for shifting the 'values' of z up by 1, and with 2 being of course an abbreviation for $1+1$:

$$H_1 \ := \ |H|^{y \to y+1+1 \ , \ v \to v+1} \ ;$$

$$L' \ := \ (z \approx 1 \wedge y \approx 0 \wedge x \approx v) \vee (z \approx 1 \wedge 0 < y \wedge x \approx v{+}1) \vee (z \approx 2 \wedge y \; od \wedge x \approx v{+}1) \vee (z \approx 2 \wedge y \; ev \wedge x \approx v) \ .$$

Here come the arrows for this case:

$$H \wedge \neg 1 < z \ \rightarrow \ J \wedge L \ \rightarrow |J \wedge L'|^{z \to z+1} \ \rightarrow \ [z \leftarrow\!\!:\, z+1](J \wedge L')$$

$$\rightarrow \ [z \leftarrow\!\!:\, z+1]((J \wedge L' \wedge 0 < y) \ \vee \ (J \wedge L' \wedge y \approx 0))$$

$$\rightarrow \ [z \leftarrow\!\!:\, z{+}1][\text{`}y \leftarrow\!\!:\, y{-}1\text{'}](|J \wedge L'|^{y \to y+1}) \vee (y \approx 0 \wedge u \approx 0 \wedge z \approx 1 \wedge x \approx v) \vee (y \approx 0 \wedge u \approx 0 \wedge z \approx 2 \wedge x \approx v) \ .$$

For the pair of 1st order components "$\vee$ed" together on the right, we have used the fact that, quite generally, $(K \wedge y \approx 0 \ \rightarrow \ [\text{`}y \leftarrow\!\!:\, y-1\text{'}](K \wedge y \approx 0))$ is $\vdash'$-derivable.

Clearly, if we could do one more 'arrow' so that the arrow's target agrees with its source except for changing "$|J \wedge L'|^{y \to y+1}$" to a '$\vee$ing' of four 1st order formulas, and so that all six of the resulting formulas followed by "$\rightarrow H$" was true in $\mathbf{N}$, then we'd be finished, as with the last sentence of the paragraph analysing (B)$_+$. This holds for both of the formulas at the right-hand end of the display above, and also for two of the four formulas alluded to above, namely

$$(y + 1 \approx u + u \wedge z \approx 1 \wedge x \approx v + 1) \ \vee \ (y + 1 \approx u + u \wedge z \approx 2 \wedge x \approx v + 1) \ .$$

But for the other two of the four, namely

$$(y{+}1{+}1 \approx u{+}u \wedge z \approx 1 \wedge x \approx v{+}1) \vee (y{+}1{+}1 \approx u{+}u \wedge z \approx 2 \wedge x \approx v{+}1) \ ,$$

we have instead that each of those formulas followed by "$\rightarrow H_1$" is true in $\mathbf{N}$. Recall that H_1 is just obtained from H by substitution of $y + 1 + 1$ for y and $v + 1$ for v. Therefore it is perfectly obvious that $H_1 \rightarrow [\nabla YE]G$ is also true in $\mathbf{N}$, since G depends only on $u + v$, and the effect of the substitutions is to increment v but to decrement u (and also because the command ∇YE

affects only x, y and z, not u and v). And so a derivation of the wfd should exist as a result of one for $H \rightarrow [\nabla Y E]G$, which of course we have in the premiss set. So the example will have been thoroughly verified as soon as this is proved. That result is an exact analogue of the lemma which appears at the end of the following section. We have even included the case of $\vdash'$ there to be convincing with this. But the result we need here is not quite a special case of the lemma, because the 'H' in this section is a bit more complicated than the 'H' in the next section. In any case, we'll leave the needed modification to the reader, who may also enjoy the following.

Exercise. Simplify the above example in such a way that y only increases by 1 each time that x increases, and do the partial correctness as above. The main creative effort will be to find the appropriate formula H. We will do the total correctness of this example in the next section. So, on the one hand, don't read there till doing the exercise; and on the other hand, the technical definitions just below will be used in the next section.

To make the example specific, define

$$C \ := \ ite(0 < y)(x \leftarrow x + 1 \ ; \ z \leftarrow 0 \ ; \ \nabla Y D)(bugga) \ ,$$

where

$$D \ := \ ite(0 < z)(callX)(\text{`}y \leftarrow y - 1\text{'} \ ; \ z \leftarrow 1 \ ; \ callY) \ ,$$

Find a derivation for the wfd $F \rightarrow [\nabla X C]G$, where F is $y \approx u \wedge x \approx v$, and where G is $y \approx 0 \wedge x \approx u + v$.

4. The Total Correctness Example.

Here we shall show how to derive the stronger total correctness results for the example in the exercise just above. The basic aim is to derive the wfd $F \rightarrow< \nabla XC > G$ for the objects defined there (not the earlier ones), using old rules from **[infocomp]**, mainly that for total correctness. The 'new' thing here will be that the latter rule, definitely in the form needed for *mutual* recursion in *that* reference, will be employed in an essential way *here* only for *single* recursion (but of course with simpleness not true for the command in the example).

Now let us continue with the technical definitions.

$$E \;:=\; |D|^{X \mapsto \nabla XC} \;=\; ite(0 < z)(\nabla XC)(\text{`}y \leftarrow\!\!: y - 1\text{'}\;;\; z \leftarrow\!\!: 1\;;\; callY)\;.$$

Thus

$$|C|^{X \mapsto \nabla XC} \;=\; ite(0 < y)(x \leftarrow\!\!: x + 1\;;\; z \leftarrow\!\!: 0\;;\; \nabla YE)(bugga)\;,$$

and

$$|E|^{Y \mapsto \nabla YE} \;=\; ite(0 < z)(\nabla XC)(\text{`}y \leftarrow\!\!: y - 1\text{'}\;;\; z \leftarrow\!\!: 1\;;\; \nabla YE)\;.$$

The magic formula to be used (also in that exercise) is the following.

$$H \;:=\; (z \approx 0 \wedge y \approx 0 \wedge u \approx 0 \wedge x \approx v) \vee (z \approx 0 \wedge 0 < y \wedge y \approx u \wedge x \approx v{+}1) \vee (0 < z \wedge y \approx u \wedge x \approx v)\;.$$

We will use the old rule more-or-less to derive

$$(F \rightarrow< \nabla XC > G) \;\wedge\; (H \rightarrow< \nabla YE > G)\;,$$

again doubling our money.

The overall structure of the example is moderately complex, so I'll first write down the rules to be used, and then outline the proof before providing all details. The rule for total correctness is the following, where heretofore, only the case $k = 1$ was needed for single recursion:

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

for all variables w, all wfs's $\mathcal{A}_j$ with w not 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.

This rule looks much more abstract than the one for partial correctness, and is certainly more general than needed. A main point is that one has the luxury of an extra rule (below) for $\vdash^w$, which is not allowed for $\vdash'$. (It would make the partial correctness rule trivially unsound.) This extra rule is the following, always used by having all the $\mathcal{B}_i$ above have the form $<\nabla XC>\mathcal{C}$, whose occurrence on the 'right-hand side' of the rule above can, using the rule below, be replaced by $<|C|^{X\mapsto\nabla XC}>\mathcal{C}$:

$$\emptyset\ /\ (<|C|^{X\mapsto\nabla XC}>\mathcal{A}\ \ \longleftrightarrow\ \ \ <\nabla XC>\mathcal{A})\ .$$

Now here is how we will proceed. Let

$$\Omega_0 := \emptyset \quad ; \quad \Omega_1 := \{y\approx 0\wedge x\approx v\to <\nabla XC>(y\approx 0\wedge x\approx v)\} \quad ;$$

$$\Omega_2\ :=\ \Omega'\cup\{y\approx 0\wedge x\approx v\to <\nabla YE>(y\approx 0\wedge x\approx v)\}\ .$$

for C, E and G as defined just before and after the start of this section; and with Ω' containing any wfds of the form

$$0<u\ \wedge J\ \to\ [\text{`}u \leftarrow\!\!: u-1\text{'}]|J|^{u\mapsto u+1}\ .$$

needed later (All these have been derived in [**sing**] using the original system, so verifying that part of the antecedent can be passed over in silence below.)

Then we shall establish (1) to (6) below. Below in (1) to (3), there is no need for the "$0<w$, $0<w+1$, $0<w+2$", but we've included them as a security blanket for the author. In all cases, the antecedent $\vdash\bigwedge_{1\leq j\leq k}\neg|\mathcal{A}_j|^{w\mapsto 0}$ is trivial to check.

(1)

$$\Omega_0\cup\{y\approx 0\wedge x\approx v\wedge 0<w\to <\nabla XC>(y\approx 0\wedge x\approx v)\}\ \vdash^w$$
$$y\approx 0\wedge x\approx v\wedge 0<w+1\to <|C|^{X\mapsto\nabla XC}>(y\approx 0\wedge x\approx v)$$

(2) So combining the two rules as indicated above (the case $k = 1$ of the first rule being relevant here), we get

$$\vdash \; y \approx 0 \wedge x \approx v \rightarrow < \nabla XC > (y \approx 0 \wedge x \approx v) \; .$$

This also uses the truth in $\mathbf{N}$ of

$$y \approx 0 \wedge x \approx v \;\; \rightarrow \;\; \exists w \; (y \approx 0 \wedge x \approx v \wedge 0 < w) \; ,$$

as well as the (PRE)-rule, which allows us to 'put the $\exists w$ in there'.

In particular, Ω_1 is now 'useable'.

(3)

$$\Omega_1 \cup \{ y \approx 0 \wedge x \approx v \wedge 0 < w{+}1 \;\rightarrow\;\; < \nabla YE > (y \approx 0 \wedge x \approx v) \} \;\vdash^w$$
$$y \approx 0 \wedge x \approx v \wedge 0 < w{+}1{+}1 \rightarrow < |E|^{Y \mapsto \nabla YE} > (y \approx 0 \wedge x \approx v) \, .$$

(4) So indeed, just as in (2), this establishes

$$\vdash \; y \approx 0 \wedge x \approx v \rightarrow < \nabla YE > (y \approx 0 \wedge x \approx v) \; ,$$

and 'makes Ω_2 useable'.

(5) Define

$$F_w \;:=\; 0 < y \;\wedge\; y + y \approx w \quad ,$$

and

$$H_w \;:=\; 0 < y \;\wedge\; ((0 \approx z \;\wedge\; y + y \approx w + 1) \;\vee\; (0 < z \;\wedge\; y + y \approx w)) \quad .$$

Then, with

$$\Gamma \;:=\; \Omega_2 \cup \{ F \wedge F_w \;\rightarrow\;\; < \nabla XC > G \; , \; H \wedge H_w \;\rightarrow\;\; < \nabla YE > G \}$$

we have

$$\Gamma \;\vdash^w \;\; (F \wedge F_{w+1} \;\rightarrow\;\; < |C|^{X \mapsto \nabla XC} > G) \wedge (H \wedge H_{w+1} \;\rightarrow\;\; < |E|^{Y \mapsto \nabla YE} > G) \, .$$

(6) So combining the two rules as indicated above (but this time with the case $k = 2$ despite the fact that we're not doing *mutual* recursion), we get

$$\vdash \ (F \wedge 0 < y \rightarrow < \nabla XC > G) \ \wedge \ (H \wedge 0 < y \rightarrow < \nabla YE > G) \ .$$

This also uses the truth in $\mathbf{N}$ of both

$$F \wedge 0 < y \ \rightarrow \ \exists w \ (F \wedge F_w) \quad \text{and} \quad H \wedge 0 < y \ \rightarrow \ \exists w \ (H \wedge H_w)$$

and the (PRE)-rule, as in (2) and (4). (But $k > 1$ for non-mutual recursion is the novelty, unheard of in the annals of phoffman@mind, but there's no accounting for thickheadedness.)

To actually finish the example, that is, to derive $F \rightarrow < \nabla XC > G$, recall from (1) that deriving

$$F \wedge 0 \approx y \rightarrow < \nabla XC > G$$

has basically already been done.

All that remains is to show why derivations exist as in (1), (3) *and* (5).

For (1), because of the *ite*-nature of $|C|^{X \mapsto \nabla XC}$, we must $\vdash^w$-derive from the premiss both

$$0 < y \wedge 0 \approx y \wedge whatever \quad \rightarrow whateverelse$$

which clearly presents no problem, and

$$\neg 0 < y \wedge 0 \approx y \wedge x \approx v \wedge 0 < w + 1 \ \rightarrow \quad < bugga > (y \approx 0 \wedge x \approx v) \ ,$$

which is also trivial.

For (3), because of the *ite*-nature of $|E|^{Y \mapsto \nabla YE}$, we must $\vdash^w$-derive from the two premisses both

$$0 < z \wedge 0 \approx y \wedge x \approx v \wedge 0 < w + 1 + 1 \ \rightarrow \quad < \nabla XC > (y \approx 0 \wedge x \approx v) \ ,$$

which is almost one of the premisses, and

$$\neg 0 < z \wedge 0 \approx y \wedge x \approx v \wedge 0 < w{+}1{+}1 \ \rightarrow \quad < \text{`}y \leftarrow\!\!\div y{-}1\text{'} \ ; \ z \leftarrow\!\!\div 1 \ ; \ \nabla YE > (y \approx 0 \wedge x \approx v) \ .$$

Look at the arrows

$$\neg 0 < z \wedge 0 \approx y \wedge x \approx v \wedge 0 < w{+}1{+}1 \;\rightarrow\; 0 \approx y \wedge x \approx v \;\rightarrow\; <\text{`}y \leftarrow y{-}1\text{'}> (0 \approx y \wedge x \approx v) \;\rightarrow$$

$$<\text{`}y \leftarrow y{-}1\text{'}> (0 \approx y \wedge x \approx v \wedge 1 \approx 1) \;=\; <\text{`}y \leftarrow y{-}1\text{'}> |0 \approx y \wedge x \approx v \wedge z \approx 1|^{z\rightarrow 1} \;\rightarrow$$

$$<\text{`}y \leftarrow y-1\text{'}>< z \leftarrow 1 > (0 \approx y \wedge x \approx v \wedge z \approx 1) \;\rightarrow$$

$$<\text{`}y \leftarrow y-1\text{'}>< z \leftarrow 1 > (0 \approx y \wedge x \approx v \wedge 0 < w+1) \;\rightarrow$$

$$<\text{`}y \leftarrow y-1\text{'}>< z \leftarrow 1 >< \nabla Y E > (0 \approx y \wedge x \approx v)\;.$$

Each is clearly $\vdash^w$-derivable, the last by the other premiss and two applications of the rule $(\mathsf{UNAR})_{<\;>}$. This is much more trivial than it looks, but details are of the essence in this notoriously error-prone subject.

Finally, establishing the derivation in (5) is more interesting.

An important new wrinkle is that first we must establish the $\vdash^w$-derivability from the three premisses of the $F \wedge F_{w+1} \;\rightarrow\; < |C|^{X\rightarrow\nabla XC} > G$ half, in order to use it for the derivation of the other half. As usual, the "*ite*" nature of $|C|^{X\rightarrow\nabla XC}$ splits the task in two. Firstly we need to $\vdash^w$-derive from the three premisses the wfd

$$F \wedge F_{w+1} \wedge 0 < y \;\rightarrow\; < x \leftarrow x+1\;;\; z \leftarrow 0\;;\; \nabla Y E > G\;.$$

Consider the arrows

$$F \wedge F_{w+1} \wedge 0 < y \;\rightarrow\; y \approx u \wedge x \approx v \wedge 0 < y \wedge y + y \approx w + 1 \;\rightarrow$$

$$< x \leftarrow x{+}1\;;\; z \leftarrow 0 > (z \approx 0 \wedge y \approx u \wedge x \approx v{+}1 \wedge 0 < y \wedge y{+}y \approx w{+}1) \;\rightarrow$$

$$< x \leftarrow x{+}1\;;\; z \leftarrow 0 > (H \wedge H_w) \;\rightarrow\; < x \leftarrow x+1 >< z \leftarrow 0 >< \nabla Y E > G\;.$$

The required derivability follows from that of each of the arrows. The rightmost arrow on the top line comes from $x \approx v$ being logically equivalent to substituting $x+1$ for x in $x \approx v+1$, and the usual stuff. The last arrow comes from a premiss. The rest are easy from the definitions of the ingredients F, F_w, H and H_w.

The other needed derivation is quite trivial from those definitions, with the occurrence of F_{w+1} below being redundant. It is to $\vdash^w$-derive

$$F \wedge F_{w+1} \wedge \neg 0 < y \;\rightarrow\; < bugga > G\;.$$

It is tempting to think that we're really finished, since this half gives the one (re ∇XC) that we're really after, rather than the subsidiary one (re ∇YE). But "rules is rules", and of course the latter is essential. Again the *ite* leaves us with $\vdash^w$-deriving the following pair of wfds, using the three premisses, *BUT USING ALSO WHAT WE JUST DERIVED* !!

$$\textsf{(A)} \quad H \wedge H_{w+1} \wedge 0 < z \;\rightarrow\; < \nabla XC > G \;;$$

$$\textsf{(B)} \quad H \wedge H_{w+1} \wedge \neg 0 < z \;\rightarrow\; < \text{`}y \leftarrow\!\!: y - 1\text{'} \;;\; z \leftarrow\!\!: 1 \;;\; \nabla YE > G \;.$$

As for (A), consider the arrows

$$H \wedge H_{w+1} \wedge 0 < z \;\rightarrow\; y \approx u \wedge x \approx v \wedge 0 < y \wedge y + y \approx w + 1 \;\rightarrow$$

$$F \wedge F_{w+1} \;\rightarrow\; < |C|^{X \leftarrow \nabla XC} > G \;\rightarrow\; < \nabla XC > G \;.$$

The final arrow uses the 'other' rule from above. The $\vdash^w$ derivability from the premisses of the penultimate arrow is exactly what we did in the previous paragraph.

Finally and unfortunately, (B) presents the most fussiness, not unsurprising for those who did the exercise at the end of Section 3, or didn't, but did read that section carefully. Consider the arrows

$$H \wedge H_{w+1} \wedge \neg 0 < z \;\rightarrow\; 0 \approx z \wedge y \approx u \wedge x \approx v{+}1 \wedge 0 < y \wedge y{+}y \approx w{+}1{+}1 \;\rightarrow$$

$$< \text{`}y \leftarrow\!\!: y - 1\text{'} > (y + 1 \approx u \wedge x \approx v + 1 \wedge y + 1 + y + 1 \approx w + 1 + 1) \;\rightarrow$$

$$< \text{`}y \leftarrow\!\!: y - 1\text{'} >< z \leftarrow\!\!: 1 > (z \approx 1 \wedge y + 1 \approx u \wedge x \approx v + 1 \wedge y + y \approx w) \;.$$

The derivability of these is straightforward.

The next step is to take the 1st order formula in that last wfd, '$\wedge$' it separately with $y \approx 0$ and with $0 < y$, and show that each is $\vdash^w$-derivable, when preceded as above by $< \text{`}y \leftarrow\!\!: y - 1\text{'} >< z \leftarrow\!\!: 1 >$ and followed by "$\ldots \rightarrow < \text{`}y \leftarrow\!\!: y - 1\text{'} >< z \leftarrow\!\!: 1 >< \nabla YE > G$". Using the (UNAR)-rule, we can drop the $< \text{`}y \leftarrow\!\!: y - 1\text{'} >< z \leftarrow\!\!: 1 >$.

The first of these is easy, *using the premiss in* Ω_2. It down comes to

$$z \approx 1 \wedge y{+}1 \approx u \wedge x \approx v{+}1 \wedge y{+}y \approx w \wedge y \approx 0 \;\rightarrow\; y \approx 0 \wedge 1 \approx u \wedge x \approx v{+}1 \;\rightarrow$$

$$< \nabla YE > (y \approx 0 \wedge x \approx v{+}1 \wedge 1 \approx u) \;\rightarrow\; < \nabla YE > (y \approx 0 \wedge x \approx v{+}u) \;\rightarrow\; < \nabla YE > G \,.$$

That premiss is used for the right-most arrow on the top line. Because $v + u \neq u + v$ (really!), the last arrow is not actually equality.

The other is done to begin with by observing the truth in $\mathbf{N}$ of

$$z \approx 1 \wedge y+1 \approx u \wedge x \approx v+1 \wedge y+y \approx w \wedge 0 < y \;\rightarrow\; |H|^{y \mapsto y+1, v \mapsto v+1} \wedge H_w \;.$$

It remains only to use our premiss $(H \wedge H_w \rightarrow < \nabla Y E > G)$ to $\vdash^w$-derive

$$|H|^{y \mapsto y+1, v \mapsto v+1} \wedge H_w \;\rightarrow\; < \nabla Y E > G \;,$$

or equivalently

$$|H|^{y \mapsto y+1, v \mapsto v+1} \rightarrow (H_w \;\rightarrow\; < \nabla Y E > G) \;.$$

This is not surprising, given that G depends only on $u + v$, and adding 1 to y is 'sort of subtracting 1 from u'. It is immediate from the following lemma, whose proof will finally complete the analysis of the example.

Lemma. *Let $\vdash^*$ be either $\vdash'$ or $\vdash^w$. Let $\mathcal{B}$ be a wfd, with u and v variables not in any subcommand of $\mathcal{B}$, and $|\mathcal{B}|^{v \mapsto v+1} \;=\; |\mathcal{B}|^{u \mapsto u+1}$. Then*

$$\Gamma \;\vdash^*\; (H \rightarrow \mathcal{B}) \quad \Longrightarrow \quad \Gamma \;\vdash^*\; (|H|^{y \mapsto y+1, v \mapsto v+1} \rightarrow \mathcal{B}) \;.$$

Proof. Taking

$$A \;:=\; (z \approx 0 \wedge y+1 \approx u \wedge x \approx v+1+1) \vee (0 < z \wedge y+1 \approx u \wedge x \approx v+1) \;,$$

we have $(|H|^{y \mapsto y+1, v \mapsto v+1} \rightarrow A)$ is $\mathbf{ttN}$; and, of course, so is $A \rightarrow A \wedge 0 < u$. Thus it suffices to establish

$$\Gamma \;\vdash^*\; A \wedge 0 < u \;\rightarrow\; \mathcal{B} \;.$$

Consider the sequence of 'arrows'

$$A \wedge 0 < u \;\rightarrow\; < \text{`}u \leftarrow\!\!\!\!\!\div u - 1\text{'} > |A|^{u \mapsto u+1}$$

$$\rightarrow\; < \text{`}u \leftarrow\!\!\!\!\!\div u-1\text{'} > (z \approx 0 \wedge y \approx u \wedge x \approx v+1+1) \vee (0 < z \wedge y \approx u \wedge x \approx v+1)$$

$$\rightarrow\; < \text{`}u \leftarrow\!\!\!\!\!\div u-1\text{'} > (|H|^{v \mapsto v+1} \vee \neg 0 < u) \;\rightarrow\; < \text{`}u \leftarrow\!\!\!\!\!\div u-1\text{'} > (|\mathcal{B}|^{v \mapsto v+1} \vee \neg 0 < u) \;=$$

$$< \text{`}u \leftarrow\!\!\!\!\!\div u - 1\text{'} > (|\mathcal{B}|^{u \mapsto u+1} \vee \neg 0 < u) \;\rightarrow\; \mathcal{B} \vee \neg 0 < u \;.$$

The usual arguments easily give the required $\vdash^*$-derivability of each—that of $(H \rightarrow \mathcal{B})$ being used for the penultimate line—and the very last arrow being very nearly the contrapositive of

$$0 < u \ \wedge \mathcal{B} \ \rightarrow \ [`u \leftarrow\!\!\div u - 1'] |\mathcal{B}|^{u \rightarrow u+1} \ .$$

The unwanted $\neg 0 < u$ at the end is easily eliminated in view of the start of all these 'arrows'. So the lemma is proved.

5. The Conjecture.

An oddity of my progress on this business, of extending the Cook-completeness proof so that simplicity need not be assumed, is that it had seemed to me to be total correctness which presented all the problems; and yet now it seems likely that my rule for total correctness of recursive commands was already general enough (but I didn't realize what was the best way to use it!), whereas the one for partial correctness is in need of improvement.

So here we go: *I suspect to be Cook-complete, for the entire command language $\mathcal{R}ec$ (specialized for now to single, not mutual, recursion), the system obtained merely by replacing the rule* $(\mathsf{RCN})_{[\ \ ]}$ *by the displayed rule below* (but leaving all other rules the same in the single recursion specialization of the system in [**infocom**]).

The whole discussion, when simpleness is abandoned, needs to be formulated more carefully, as indicated earlier, to avoid variable-clash problems when substituting for procedure variables. And that seems to require some work on the semantics. But all the machinery for doing that is in [**dpth**]. So we'll just write the general form of the new rule in a state of 'variable-clash innocence' :

$$\frac{\vdash \mathcal{C} \ \forall \mathcal{C} \in \Omega \quad , \quad \Omega \cup \{F_i \rightarrow [\nabla X_i C_i] G_i \mid i \in I\} \ \vdash' \ \ \wedge_{i \in I} \ (F_i \rightarrow [|C_i|^{X_i \rightarrow \nabla X_i C_i}] G_i)}{\vdash \ \ \wedge_{i \in I} \ (F_i \rightarrow [\nabla X_i C_i] G_i)} \ .$$

The sets I, of indices, and Ω, of wfd's, are both finite, and none of the wfd's in Ω contains any subcommand of the form $\nabla X_i C_i$.

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