Systems for 'while'-Total Correctness with Quantifiers and Connectives

Peter Hoffman

Pure Mathematics, University of Waterloo

Abstract. *The literature includes only a few instances of a proof system for total and partial correctness of programs in the* while *command language, where the correctness-assertion language includes all the usual* $1^{\underline{st}}$*order constructions. This system was unsound due to an oversight, easily corrected. Once corrected, the system then turns out in its original incarnation to be incomplete (in the sense of Cook), due to another small slip-up. In another incarnation with a restricted language, the unsoundness problem is there but not the incompleteness. We exhibit an example and a proof concerning these facts, after giving a slightly simpler system, and discuss the verification of the main 'tricky' points concerning soundness and completeness. To finish, there are brief comments about the analogue in which 'while-do' is replaced by recursion with a single variable.* [1] [2]

The system of the title is for a familiar deterministic specialization of (perhaps—'a poor cousin of') dynamic logic [**H**]. We begin just below by describing the formal languages involved, and giving a proof system. The paper then gives details concerning the following facts: firstly, a sketch of the validity of the main rule of inference for proving termination of the 'while-do' command and an example showing how earlier versions of that rule need a slight correction to be valid; then a theorem to show that a slightly weakened system, as appearing in earlier literature, is not in fact complete (in the sense of Stephen Cook); finally, the proof, whose basic strategy is entirely derived from [**H**], that the system presented here is complete. This subject evidently should be interesting to many mathematicians and logicians (despite dealing with a command language of impractical austerity). It deserves a careful and (hopefully) attractive treatment.

[1] AMS Subject Codes (2000): 03D80, 03D99, 68N30.

[2] Keywords: Floyd-Hoare logic, Cook completeness, $1^{\underline{st}}$order dynamic logic.

The official language here will have, as its atomic strings, the atomic formulae from the language of $1^{\underline{st}}$order number theory, and be closed under negation ($\neg$), conjunction ($\wedge$), implication ($\rightarrow$), universal quantification and under $\mathcal{B} \mapsto < C > \mathcal{B}$, for any C in the while command language arising from that $1^{\underline{st}}$order language. (The construction and semantics for that command language and for $< C >$ are explained below.) Other connectives and existential quantification are given their usual definitions in terms of the constructs above. We'll refer to a string produced this way as a *well-formed string*, abbreviated *wfs*.

As to the semantics of the "official language", the string $< C > \mathcal{B}$ is true at input $\underline{v}$ iff both $||C||(\underline{v}) \neq err$ and $\mathcal{B}$ is true at $||C||(\underline{v})$. Here $\underline{v}$ denotes an assignment of the variables, which would normally be conceived in this context as a state of the machine implementing commands from while, the latter explained below. So we prefer the term *input* to the multisyllabic *assignment of variables*. And $||C||$ is the 'function' which the command C computes, with the value *err* indicating an 'infinite loop'.

Note that then, the universal closure of $\mathcal{A} \rightarrow < C > \mathcal{B}$ becomes a typical Floyd-Hoare *total* correctness statement, in that it is true in a given interpretation of the language exactly when both $||C||(\underline{v}) \neq err$ and $\mathcal{B}$ is true at $||C||(\underline{v})$, for all inputs $\underline{v}$ where $\mathcal{A}$ is true. In ordinary language, it's true if, whenever the machine implementing the command C begins in a state $\underline{v}$ at which $\mathcal{A}$ is true, it doesn't loop ('totalness'), and that implementation produces an output state where $\mathcal{B}$ is true.

As is well known from dynamic logic, one can introduce a syntactical construct which deals with partial correctness analogously. Define

$$[C]\mathcal{B} \ =: \ \neg < C > \neg \mathcal{B} \ .$$

Then $[C]\mathcal{B}$ is true at $\underline{v}$ iff either $||C||(\underline{v}) = err$ or $\mathcal{B}$ is true at $||C||(\underline{v})$; and the universal closure of $\mathcal{A} \rightarrow [C]\mathcal{B}$ has exactly the semantics of the usual Floyd-Hoare *partial* correctness statement, often denoted in undergraduate CS courses by either $F\{C\}G$ or $\{F\}C\{G\}$ when $\mathcal{A}$ and $\mathcal{B}$ are specialized to $1^{\underline{st}}$order number theory formulae F and G respectively.

For us, the while command language is built by structural induction from atomic strings $x \doteqdot t$, the assignment command, which replaces the old value of the variable x with the value of the term t, by two command constructions:

$(C; D)$ is the sequencing of two commands, doing C first;

$\mathsf{whdo}(H)(C)$ is the command "while H, do C".

We will omit $\mathsf{ite}(H)(C)(D)$, the command "if H, then do C, else do D". As explained in many places, and likely known 200 years ago to Charles Babbage, ite is redundant. If it is preferred to include it as part of the basic language rather than as an abbreviation, just add one axiom to the system below, namely

$$[\mathsf{ite}(H)(C_1)(C_2)]G \longleftrightarrow (H \to [C_1]G) \wedge (\neg H \to [C_2]G) \quad ,$$

(or else the same with [—] repaced by <—>).

In both the 'while-do' and the 'if-then-else' commands above, the $1^{\underline{\text{st}}}$order formula H is quantifier-free.

The semantics of this command language is very obvious from its informal description just above. It is written out formally in many places. It makes sense abstractly for any interpretation of the $1^{\underline{\text{st}}}$order number theory language, such as a non-standard model for '$1^{\underline{\text{st}}}$order natural numbers'. But it will be easier to proceed as follows.

Take the interpretation to be $\mathbf{N}$, the natural numbers, with the usual addition, multiplication, ordering and 0 and 1 as the structure interpreting the language. This may be generalized to an *arithmetical universe* as in [**H**]. Doing so would do no more than divert attention away from the points being made here. Apparently, it is still not known how to get a relatively complete proof system in the situation of an arbitrary (expressive) language/interpretation without extending the language and interpretation to include numbers.

Just below is a sound, and relatively complete in Cook's sense, proof system for the language/semantics given above. This is for deriving strings which are universally true, i.e. true 'in $\mathbf{N}$', not just 'at $\underline{v}$'. In particular, validity of a rule means that when all antecedents are true in $\mathbf{N}$, then so is the consequent. Relative completeness in Cook's sense means that, by taking as premisses all $1^{\underline{\text{st}}}$order formulas true in $\mathbf{N}$ (i.e. one has an oracle for truth of $1^{\underline{\text{st}}}$order formulas about the natural numbers), the system permits derivations of *all* strings from the language that are true in $\mathbf{N}$. But we have built the oracle into the system as the first collection of axioms (or *premisses* if you like), making this "system" non-axiomatic in the technical sense.

Here is some (not entirely standard) notation we use below:

$F^{[x \to t]}$ is the result of substituting the term t for the variable x in the $1^{\underline{\text{st}}}$order formula F—(we need not agonize over generalizing this to the whole language—fortunately, since it produces some sweat when applied, for example to assignment commands) ;

Perhaps overly fussy, we use $\approx$ for the equality symbol in the formal languages, to distinguish it from the semantic relation of equality.

It is salient to note that, in the proof system below, symbols F, G and H denote $1^{\underline{\text{st}}}$order number theory formulae (sometimes quantifier-free as indicated above), not general wfs's. The latter are denoted $\mathcal{A}, \mathcal{B}, \mathcal{C}, \cdots$.

The following system (only a little different from one in [**H**]) has a few small advantageous properties, discussed below, over earlier ones.

The System.

(ORACLE) All $1^{\underline{\text{st}}}$order formulas which are true in **N**.

(TAUT) All propositional tautologies (with wfs's substituted for their propositional variables).

$(\text{AX})_{\div}$ $\quad < x \div t > F \iff F^{[x \to t]}$

$(\text{AX})_{;}$ $\quad < (C_1; C_2) > F \iff < C_1 >< C_2 > F$

(MP) $\quad \dfrac{\mathcal{A}\ ,\ \mathcal{A} \to \mathcal{B}}{\mathcal{B}}$, i.e. modus ponens

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

$(\text{UNAR})_{\forall}$ $\quad \dfrac{\mathcal{A} \to \mathcal{B}}{\forall x\ \mathcal{A} \to \forall x\ \mathcal{B}}$

$(\text{LOOP})_{[\ ]}$ $\quad \dfrac{F \wedge G \to [C]F}{F \ \to\ [\mathsf{whdo}(G)(C)](F \wedge \neg G)}$

$(\text{LOOP})_{<>}$ $\quad \dfrac{F^{[x \to 0]} \to \neg G \quad , \quad F^{[x \to x+1]} \ \to\ (G \wedge < C > F)}{\exists x F \to < \mathsf{whdo}(G)(C) > F^{[x \to 0]}}$

for $x \notin C \cup G$.

The last line is just saying that the variable x should not appear in either string, C or G.

Note that F in both $(\text{AX})_{;}$ and $(\text{LOOP})_{[\ ]}$ is a $1^{\underline{\text{st}}}$order formula, not a completely general wfs, giving a minor advantage to this system over the

earlier one derived directly from dynamic logic, though the questions about soundness and completeness discussed in the remainder of this paper are likely of more interest.

The claim for a system like this to be 'Cook-complete' is a result known to the experts since at least sometime in the late 1970's. But previous versions seem all to have one or another minor defect, such as *not* being sound and/or complete. Although while is a skeleton for, but is not itself, a practical 'Algol-like' command language, it seems worthwhile, since while is in fact 'Turing-equivalent' or 'universal' (and also in order to satisfy the sensitivities of at least one pure mathematician) to have a system for while which is defect-free. And we claim that this one is. Generally speaking, CSers seem happy to come close, as long as the defects are ones which can be immediately seen to be needing correction and to be correctible in an obvious way, at least by those with deep knowledge of the subject. (One trusts that these corrections do take effect before attempted conversion into practical software verification products to check software which lands airliners. However, one mustn't be too hypercritical here; the subject seems to be very tricky—even for a very basic system for commands with recursive procedures, a much referred-to paper which fixed an incompleteness from 5 years previous stood itself for 10 years, and with probably at least 100 contributions to the literature referring to it, before the fact that its system had now become unsound was discovered. See the postscript at the end.) Sometimes it is quite a tedious chore to provide a rigorous proof that some defect really does need correction; e.g. omitting the existential quantifier in $(\mathrm{LOOP})_{<>}$ results in an incomplete system (see **Theorem 1** below), despite some (who remember their rules for prenexing from high school logic) thinking otherwise.

Example. Fix a pair of distinct variables x and y. Any command, which decreases y by 1 when it is positive, and doesn't affect x, will be denoted "$y \dot{\leftarrow} y - 1$". The assertion $< C > J$ is clearly true in $\mathbf{N}$, where

$$C \text{ is } \mathsf{whdo}(0 < y)(\text{“}y \dot{\leftarrow} y - 1\text{”}) \ , \text{ and } J \text{ is } y \approx 0 \ .$$

A derivation of it is fairly easy to come by: The rule $(\mathrm{LOOP})_{<>}$ yields a derivation of

$$\exists x \ y \approx x \ \rightarrow \ \ < C > J \ ,$$

since J is $(y \approx x)^{[x \mapsto 0]}$. The rule is applicable since the corresponding

assertions in the 'numerator' of that rule, namely

$$y \approx 0 \to \neg 0 < y \ , \ y \approx x{+}1 \to 0 < y \ , \text{ and } \ y \approx x{+}1 \to < \text{“}y \dot{-} y{-}1\text{”} > y \approx x$$

are 'clearly' all derivable. (The last one would use $(\text{UNAR})_{<>}$ after writing out "$y \dot{-} y-1$" explicitly.) Now apply modus ponens and the logical validity of $\exists x \ y \approx x$. (Note for later purposes that the existential quantifier seems to be crucial here.)

Soundness. The main points are as follows in the next three paragraphs.

A version of the last rule, $(\text{LOOP})_{<>}$, is invalidly stated in [**A**], Rule 6, p.441, and also in [**H**] , bottom of p.37, both failing to have the restriction that the variable x does not occur in G. Here is a simple counterexample to the validity of that stronger rule (whether "$\exists x$" is in the 'denominator' or not).

$$\frac{(x \approx y)^{[x \mapsto 0]} \to \neg x + 1 \approx y \ , \ (x \approx y)^{[x \mapsto x+1]} \to x + 1 \approx y \ , \ x + 1 \approx y \to < \text{“}y \dot{-} y - 1\text{”} > x \approx y}{\exists x \ x \approx y \to < \mathsf{whdo}(x + 1 \approx y)(\text{“}y \dot{-} y - 1\text{”}) > (x \approx y)^{[x \mapsto 0]}}$$

It is easily seen that the 'numerator' assertions are true in **N**, but the one in the 'denominator' is not.

On the positive side, here is a sketch of the proof, summarized into a diagram, that $(\text{LOOP})_{<>}$, as we state it, is valid. (Since it is stronger than the version in the literature which fails to give completeness [as established in **Theorem 1** below], it seems worthwhile including this. But alternatively, we could appeal to the validity of that too-weak rule plus that of the "high-school prenex rule" mentioned in the remark just after **Theorem 1**'s statement.) With the notation from the rule, assume that the two antecedents in its numerator are true in **N**. Let $\underline{v}$ be such that $\exists x \ F$ is true at $\underline{v}$. Choose some $\underline{w}$ with $\underline{v} \ \overset{\not x}{\sim} \ \underline{w}$, [i.e. so that $\underline{v}$ agrees with $\underline{w}$ except possibly at x] and with F itself true at $\underline{w}$. Let $n := \underline{w}(x)$. For $0 \leq i \leq n$, define $\underline{w}^{(i)} := \underline{w}^{(x \mapsto n-i)}$, which maps x to $n - i$, but any other y to $\underline{w}(y)$. Thus $\underline{w}^{(0)} = \underline{w}$; $\underline{w}^{(n)}(x) = 0$; and $\underline{w}^{(i)} \ \overset{\not x}{\sim} \ \underline{v}$ for all i .

Now we get a diagram as follows, each detail of which has an elementary verification. The symbol string $H\mathbf{t}\underline{u}$ at all sites in the diagram abbreviates the statement "The formula H is true at input $\underline{u}$".

$$
\begin{array}{cccccc}
F\mathbf{t}\underline{w}^{(0)} \Rightarrow F^{[x\to x+1]}\mathbf{t}\underline{w}^{(1)} \Rightarrow F\mathbf{t}||C||(\underline{w}^{(1)}) \Rightarrow F^{[x\to x+1]}\mathbf{t}||C||(\underline{w}^{(2)}) \Rightarrow F\mathbf{t}||C||^2(\underline{w}^{(2)}) \Rightarrow \cdots F^{[x\to x+1]}\mathbf{t}||C||^{n-1}(\underline{w}^{(n)}) \Rightarrow F\mathbf{t}||C||^n(\underline{w}^{(n)}) \\
\Downarrow & \Downarrow & & \Downarrow & \Downarrow \\
G\mathbf{t}\underline{w}^{(1)} & G\mathbf{t}||C||(\underline{w}^{(2)}) & \cdots & G\mathbf{t}||C||^{n-1}(\underline{w}^{(n)}) & F^{[x\to 0]}\mathbf{t}||C||^n(\underline{w}^{(n)}) \\
\Downarrow & \Downarrow & & \Downarrow & \Downarrow \\
G\mathbf{t}\underline{v} & G\mathbf{t}||C||(\underline{v}) & \cdots & G\mathbf{t}||C||^{n-1}(\underline{v}) & \neg G\mathbf{t}||C||^n(\underline{w}^{(n)}) \\
 & & & & \Downarrow \\
 & & & & \neg G\mathbf{t}||C||^n(\underline{v})
\end{array}
$$

The statements at the bottoms of the columns together with the second statement in the rightmost column tell us what we need to know, namely that $< \mathsf{whdo}(G)(C) > F^{[x\to 0]}$ is true at $\underline{v}$.

Completeness. Now we consider the question of completeness of the present system, and lack thereof for an earlier version. This is completeness in the sense of Cook, as explained earlier.

Theorem 1. (Cook incompleteness of similar system) *If " $\exists x$ " is erased in the consequent of the last rule,* $(\mathrm{LOOP})_{<>}$*, then the resulting system is incapable of deriving any wfs $\mathcal{W}$ of the form $< \mathsf{whdo}(G)(C) > H$.*

Remarks. We shall use $\mathcal{W}$ as the generic name for any wfs of the form $< \mathsf{whdo}(G)(C) > H$, as in the theorem. If $\mathcal{W}$ is true in $\mathbf{N}$, this weakened system will derive plenty of wfs's of the form $\exists z\mathcal{W}$ and $< D > \mathcal{W}$ for variables z and commands D which 'have nothing to do' with $\mathcal{W}$; and so it derives wfs's which are more-or-less semantically as strong as $\mathcal{W}$. Thus the theorem's proof is necessarily somewhat arcane. Furthermore, one additional axiom, looking like a high-school prenexing rule, namely: for $\mathcal{B}$ having no occurrence of z,

$$\forall z(\mathcal{A} \to \mathcal{B}) \ \to \ ((\exists z\mathcal{A}) \to \mathcal{B}) \ ,$$

would suffice to fix the system for anyone loathe to re-introduce that existential quantifier in $(\mathrm{LOOP})_{<>}$. The instances of that 'axiom' in which $\mathcal{A}$ and $\mathcal{B}$ are merely $1^{\underline{\mathrm{st}}}$order formulas are trivially derivable, but not the general case without the correct version of $(\mathrm{LOOP})_{<>}$. The main reason why it is hard to 'believe in' the weakened system is the following. The examples further up make one expect that the usual use of $(\mathrm{LOOP})_{<>}$ is to find an F for which

$\exists x F$ is true in **N**, and for which the antecedents in that rule are derivable, and then to just apply that rule, and then (MP), to derive the while-do statement in the consequent of that rule. But if the existential quantifier is dropped, this quickly leads to the contradiction of having both G and its negation true in **N**, as we see in **Lemma A**. In particular, the argument for the lemma breaks down at an obvious place when the consequent is allowed to again have that existential quantifier.

The proof of the theorem is after lemmas **A, B** and **C**.

Lemma A. *If a line $F \to \mathcal{W}$ justifiable by the weakened (LOOP)$_{<>}$ appears in a derivation, then the 1$^{\underline{\text{st}}}$order formula F is not true in* **N**.

Proof. Because of its justification by (LOOP)$_{<>}$, the line $F \to \mathcal{W}$ must be preceded in the derivation by two lines $F^{[x \to x+1]} \to (G \wedge < C > F)$ and $F^{[x \to 0]} \to \neg G$. In particular, $F^{[x \to 0]} \to \neg G$ and $F^{[x \to x+1]} \to G$ are both true in **N**. But if F were true in **N**, so are $F^{[x \to 0]}$ and $F^{[x \to x+1]}$. Combining with the above, we see that both $\neg G$ and G are true in **N**, a contradiction.

Lemma A$_+$. *If a derivation contains lines $F_1 \to \mathcal{W}, \cdots, F_k \to \mathcal{W}$, all justifiable by the weakened (LOOP)$_{<>}$, then there is an input where all the F_i are false.*

Proof. (This becomes **A** when $k = 1$ of course. The proof is similar, but a bit more involved.) Firstly we may have several distinct variables, say x_i occurring free in F_i, but none occurring free in $\mathcal{W}$, which are the ones appealed to in using (LOOP)$_{<>}$ to justify the appropriate lines. Suppose **A**$_+$ to be false, so for all $\underline{v}$ there is an i with F_i true there. We'll show G and $\neg G$ both true at some particular input, giving the contradiction. First let $\underline{v}$ be any input where all the x_i are zero. Choose i with F_i true at $\underline{v}$. Then $F_i^{[x_i \to 0]}$ is true at $\underline{v}$. Since $F_i^{[x_i \to 0]} \to \neg G$ is true in **N**, we see that $\neg G$ is true at $\underline{v}$. But $\neg G$ has no free variables x_j for any j, so in fact $\neg G$ is true at every input. Now let $\underline{v}$ be any input in which all x_i are positive. Choose j with F_j true at $\underline{v}$. Then $F_j^{[x_j \to x_j+1]}$ is true at the input which agrees with $\underline{v}$ except that x_j has a value one smaller. Since $F_j^{[x_j \to x_j+1]} \to G$ is true in **N**, we see that G is true at that input, giving the promised contradiction. (It is interesting that the proof seems to need the extra condition included to avoid the soundness problem of previous systems. Lack of soundness does tend to allow the possibility of deriving anything, so might be troublesome

in the proof of the theorem.)

Lemma B. *If the wfs $\mathcal{A} \to \mathcal{W}$ is a tautology and $\mathcal{A}$ is true in* **N**, *then $\mathcal{A}$ necessarily has $\mathcal{W}$ as a sub-wfs. More precisely, $\mathcal{W}$ occurs as a* propositional sub-wfs *("pswfs"), in the sense that, for some propositional formula J in which the variable P appears, $\mathcal{A}$ is produced by a substitution of wfs's for occurrences in J of variables, including substituting $\mathcal{W}$ for P.*

Proof. This follows because a propositional tautology of the form $J \to P$, where P is a variable and J is not a 'contradiction', necessarily must include P as a variable appearing in J. Otherwise we could just set P to false, and all the other variables in some way which makes J true, contradicting tautologicality.

Lemma C. *The following wfs's do not have $\mathcal{W}$ as a pswfs : Any* $1^{\underline{\text{st}}}$*order formula; any instance of either* (AX)*; any instance of the consequent of either rule* (UNAR).

Proof. Syntactic inspection and unique readability.

Main Lemma D. *For any $n \geq 1$, any $\mathcal{W}$ and any "n" distinct wfs's $\mathcal{A}_1, \cdots, \mathcal{A}_n$ such that $\mathcal{A}_1 \wedge \cdots \wedge \mathcal{A}_n \to \mathcal{W}$ is a tautology, there is no derivation* (in the weakened system of course) *containing all the lines $\mathcal{A}_i$.*

Remark. Taking $n = 1$ and $\mathcal{A}_1 = \mathcal{W}$, this reduces to **Theorem 1**. (On the other hand, that theorem easily implies **D**, since a derivation forbidden by **D** quickly yields a derivation of $\mathcal{W}$.)

Proof of D. Assume, for a contradiction, that such a derivation exists. Among such derivations, consider one for which the set of the line numbers of the $\mathcal{A}_i$ is as small as possible, with respect to the following ordering on the set of finite subsets of positive integers: Write such a set as a decreasing sequence, and compare it to other sets by the lexicographic order on the corresponding sequences.

From now on, let $\mathcal{W}$ denote the fixed wfs $<$ whdo *etc.*... to which the theorem statement refers in this 'minimal' derivation. By **B**, *there is at least one i for which $\mathcal{A}_i$ has $\mathcal{W}$ as a pswfs.*

First we'll show that *any such line $\mathcal{A}_i$ can be justified only by* (MP) *or by* $(\text{LOOP})_{< >}$. By **C**, the only other possibilities are (TAUT) and $(\text{LOOP})_{[\ ]}$. But if $\mathcal{A}_i$ were a tautology, and since $\mathcal{A}_i \to ((\wedge_{j \neq i} \mathcal{A}_j) \to \mathcal{W})$ also is, we get

that $(\wedge_{j\neq i}\mathcal{A}_j) \rightarrow \mathcal{W}$ is a tautology, contradicting the minimality of the line number set. As for $(\text{LOOP})_{[\ ]}$, then $\mathcal{A}_i$ would have the form $F \rightarrow \neg\mathcal{W}'$ for some $\mathcal{W}'$. But $\mathcal{W}$ can only occur as a sub-wfs of this if it coincides with $\mathcal{W}'$. Now we have that

$$(F \rightarrow \neg\mathcal{W}) \rightarrow ((\wedge_{j\neq i}\mathcal{A}_j) \rightarrow \mathcal{W})$$

is a tautology. Thus there is a propositional tautology of the form

$$(K \rightarrow \neg P) \rightarrow (J \rightarrow P)$$

for a variable P, where $\wedge_{j\neq i}\mathcal{A}_j$ is obtained by substituting wfs's for the variable occurrences in J, including substituting $\mathcal{W}$ for P. It is almost immediate that $J \rightarrow P$ is also a propositional tautology, and so $(\wedge_{j\neq i}\mathcal{A}_j) \rightarrow \mathcal{W}$ is a tautology. This again contradicts the minimality of the line number set. (The above even works for the case $n = 1$, though the reader might prefer to give a direct argument for that case.)

Next, we'll show that *at least one such $\mathcal{A}_i$ can be justified only by* (MP). To see this, suppose instead that all such $\mathcal{A}_i$ can be justified by $(\text{LOOP})_{<\ >}$. We can re-index these $\mathcal{A}_i$ to be the first few,

$$\mathcal{A}_1 = F_1 \rightarrow \mathcal{W}_1, \quad \cdots \quad , \mathcal{A}_k = F_k \rightarrow \mathcal{W}_k \ ,$$

where $k \geq 1$, and each F_i is a $1^{\underline{\text{st}}}$order formula. As in the previous paragraph, each $\mathcal{W}_i$ must be $\mathcal{W}$ itself. So we see that

$$(F_1 \rightarrow \mathcal{W}) \wedge \cdots \wedge (F_k \rightarrow \mathcal{W}) \wedge \mathcal{A}_{k+1} \wedge \cdots \wedge \mathcal{A}_n \rightarrow \mathcal{W}$$

is a tautology, where $\mathcal{W}$ does not occur as a pswfs in $\mathcal{A}_i$ for $i > k$. Thus there is a propositional tautology

$$(K_1 \rightarrow P) \wedge \cdots \wedge (K_k \rightarrow P) \wedge J \rightarrow P \ ,$$

where the variable P does not occur in J and $\wedge_{j>k}\mathcal{A}_j$ is obtained by substituting wfs's for J's variable occurrences, including $\mathcal{W}$ for P. It follows that any truth assignment which makes J true must make at least one K_i true, since we can take P to be false, and at least one $K_i \rightarrow P$ must be false. Now consider any input $\underline{v}$. Produce a truth assignment by first replacing each propositional variable by its substituted wfs, and then seeing whether or not

that wfs is true at $\underline{v}$. Pick an i such that K_i is true at that assignment. So for each $\underline{v}$ there is an i with F_i true at $\underline{v}$. But this contradicts $\mathbf{A}_+$ since all $F_i \to \mathcal{W}$ are lines which can be justified by $(\text{LOOP})_{< >}$ in the derivation.

Finally, we shall get our overall contradiction, proving **D**, by looking at the justification for $\mathcal{B} \to \mathcal{A}_1$, where, after re-indexing, the wfs $\mathcal{A}_1$ is an $\mathcal{A}_i$ which can only be justified by (MP), and so lines $\mathcal{B}$ and $\mathcal{B} \to \mathcal{A}_1$ occur above the line containing $\mathcal{A}_1$. Since $\mathcal{W}$ is a pswfs of $\mathcal{B} \to \mathcal{A}_1$, the only possible such justifications, by **C**, are (TAUT), (MP) and the two (LOOP)'s. Now $(\text{LOOP})_{[\ ]}$ is out of the question, since otherwise $\mathcal{B} \to \mathcal{A}_1$ has the form $F \to \neg\mathcal{W}$, and $\neg\mathcal{W}$ occurs as a line. But we cannot have both $\mathcal{W}$ and $\neg\mathcal{W}$ true in **N**. Also $(\text{LOOP})_{< >}$ is eliminated by an application of **A**, since $\mathcal{B} \to \mathcal{A}_1 \ = \ F \to \mathcal{W}$ and $\mathcal{B} = F$ both would occur as lines. Furthermore, (TAUT) is impossible, since then $\mathcal{B} \to \mathcal{A}_1$ is a tautology, so

$$\mathcal{B} \wedge \mathcal{A}_2 \wedge \cdots \wedge \mathcal{A}_n \to \mathcal{A}_1 \wedge \mathcal{A}_2 \wedge \cdots \mathcal{A}_n \ \text{ and therefore } \ \mathcal{B} \wedge \mathcal{A}_2 \wedge \cdots \wedge \mathcal{A}_n \to \mathcal{W}$$

also are. But then we may replace $\mathcal{A}_1$ by $\mathcal{B}$ (or drop $\mathcal{A}_1$ if $\mathcal{B} = \mathcal{A}_j$ for some $j > 1$) and obtain another contradiction to the minimality of the line number set.

So we are left with (MP) as the only possibility. Thus lines $\mathcal{B}_1$ and $\mathcal{B}_1 \to (\mathcal{B} \to \mathcal{A}_1)$ must occur before the line $\mathcal{B} \to \mathcal{A}_1$. On very simple syntax grounds, the line $\mathcal{B}_1 \to (\mathcal{B} \to \mathcal{A}_1)$ is only justifiable by (TAUT) or (MP). If the former, then $\mathcal{B}_1 \wedge \mathcal{B} \to \mathcal{A}_1$ is a tautology, so, as above,

$$\mathcal{B}_1 \wedge \mathcal{B} \wedge \mathcal{A}_2 \wedge \cdots \wedge \mathcal{A}_n \to \mathcal{W}$$

is also, which once again contradicts the minimality of the line number set. So we are again forced to (MP), and lines $\mathcal{B}_2$ and $\mathcal{B}_2 \to (\mathcal{B}_1 \to (\mathcal{B} \to \mathcal{A}_1))$ would occur. The same argument again applies, forcing the existence of lines $\mathcal{B}_3$ and $\mathcal{B}_3 \to (\mathcal{B}_2 \to (\mathcal{B}_1 \to (\mathcal{B} \to \mathcal{A}_1)))$, etc. This gives an infinite regress, thereby proving **Lemma D**, and so, completing the proof of **Theorem 1**.

In [**A**], the existential quantifier is correctly included in rule $(\text{LOOP})_{< >}$ (though validity is a problem, as mentioned earlier). The list of rules there is much briefer, because the language restricts itself to the traditional Floyd-Hoare assertions, but interpreted to mean total, not partial, correctness. See the interruption in the middle of the proof of **Theorem 2** below for more detail. Also the suggestion in [**A**] is essentially to replace $(\text{LOOP})_{[\ ]}$ with

$(\text{LOOP})_{< >}$, rather than just add the latter as an extra rule, to pass from a system for partial correctness to one for total correctness. This suggests several more questions about the completeness of systems which have some but not all the ingredients of the system we have given in this paper. The guess would be that no reasonable such weakened system is actually Cook-complete; but the degree of interest anyone has in actually proving this rigorously (as in **Theorem 1**) is possibly not too high. At least for some of those cases, a more semantic argument might be possible. For the theorem above, it seems not to be.

The earliest appearance of a rule which is almost identical to $(\text{LOOP})_{< >}$ seems to be **[W]**, p.98, rule T7. Questions of completeness are not explicitly addressed there. Some terms are not formally defined, such as "auxiliary variable". And the existential quantifier does not appear explicitly. Also, whdo-commands are not regarded as basic constructs, but rather defined in terms of 'GOTO-commands' [to be more precise, ... in terms of 'GO if (quantifier-free formula) TO (label)']. So the above rule is regarded as being derived from basic rules for these GOTO-style commands. It is therefore difficult to comment accurately on how that rule compares with $(\text{LOOP})_{< >}$ with respect to the matters we have been discussing. There are some difficulties dealing with GOTO-commands which seem not to have yet been completely resolved. See **[O'D]**. Although deterministic command languages are somewhat out-of-style in present-day practical CS program verification research, such programs with FORTRANish labels are even more out-of-style, apparently.

Before proving adequacy for the (unweakened) system, we need an **expressiveness result**, namely :

For any wfs $\mathcal{A}$, there is a $1^{\underline{\text{st}}}$order formula A with $A \leftrightarrow \mathcal{A}$ true in $\mathbf{N}$.

All cases except the last inductive case (of the proof by structural induction on $\mathcal{A}$) are very easy. The last case, when $\mathcal{A}$ is $< C > \mathcal{B}$, can be done readily with the help of Gödel's definability result for semi-decidable relations, as follows:

Let all the variables in C and the free ones in $\mathcal{B}$ be from among the distinct variables $y_1, \cdots, y_n$. Let $x_1, \cdots, x_n$ be distinct variables, disjoint from the y_i's. For the fixed command C, define a $2n$-ary relation, R_C , on

natural numbers by

$$R_C(a_1, \cdots, a_n, b_1, \cdots, b_n) \iff ||C||(\vec{a}) = \vec{b}\ .$$

This is semi-decidable, so, by Gödel, let H be a $1^{\underline{\text{st}}}$order formula with free variables from among the x_i and y_i, such that

$$R_C(\vec{a}, \vec{b}) \iff H^{[\vec{x}\to\vec{a}\ ,\ \vec{y}\to\vec{b}]} \text{ is true in } \mathbf{N}\ .$$

By the inductive hypothesis, choose a $1^{\underline{\text{st}}}$order formula B with free variables from among the y_i, such that $B \leftrightarrow \mathcal{B}$ is true in $\mathbf{N}$.

Now define A to be $\exists y_1 \cdots \exists y_n\ (H \wedge B)$.
Then

$$A \text{ is true at } \vec{a} \iff \text{for some } \vec{b}\ ,\ (H \wedge B)^{[\vec{y}\to\vec{b}]} \text{ true at } \vec{a} \iff$$

$$\text{for some } \vec{b}\ ,\ \text{both } H^{[\vec{x}\to\vec{a}\ ,\ \vec{y}\to\vec{b}]} \text{ and } B^{[\vec{y}\to\vec{b}]} \text{ are true in } \mathbf{N} \iff$$

$$\text{for some } \vec{b}\ ,\ \text{both } R_C(\vec{a}, \vec{b}) \text{ holds and } B \text{ is true at } \vec{b} \iff$$

$$\text{for some } \vec{b}\ ,\ ||C||(\vec{a}) = \vec{b} \text{ and } B \text{ is true at } ||C||(\vec{a}) \iff$$

$$||C||(\vec{a}) \neq \mathit{err} \text{ and } \mathcal{B} \text{ is true at } ||C||(\vec{a}) \iff < C > \mathcal{B} \text{ is true at } \vec{a}\ .$$

Thus, as required, $A \leftrightarrow < C > \mathcal{B}$ is true in $\mathbf{N}$.

Theorem 2. *The system given is capable of deriving any wfs which is true in* $\mathbf{N}$*, i.e. it is Cook-complete.*

Remark. As mentioned, the overall ideas for this proof are mostly due to Harel [**H**], in his sketch proof for a dynamic logic system, whose specialization to the deterministic case is presumably almost the same as our system above, if done carefully. The subset of those ideas below in dealing with $F \to [C]G$ (as opposed to $F \to < C > G$) go right back to Cook's seminal paper [**C**].

Proof. Accuse an occurrence of $\forall x$ in $\mathcal{A}$ of being dull when its scope is a $1^{\underline{\text{st}}}$order formula. Define, for each $\mathcal{A}$,

$$M_{\mathcal{A}} := \#\text{non-dull occurrences in } \mathcal{A} \text{ of } \forall x \text{ for various } x\ +$$

$$\#\text{occurrences in } \mathcal{A} \text{ of } < C > \text{ for various } C\ .$$

Then $\mathcal{A}$ is a $1^{\underline{\text{st}}}$order formula if and only if $M_{\mathcal{A}} = 0$.

We shall prove

$$(*) \quad \text{if } \mathcal{A} \text{ is true in } \mathbf{N}, \text{ then it is derivable}$$

by induction on $M_{\mathcal{A}}$. Since all (true in $\mathbf{N}$) $1^{\underline{\text{st}}}$order formulas are premisses, the start of the induction is trivial.

Here is the overall sketch of the lengthy inductive step, in reverse order :

First reduce it to proving $(*)$ for wfs's $\mathcal{C} \to \mathsf{sym}\mathcal{B}$, in the four cases of the symbol string

$$\mathsf{sym} \;=\; < C > \;\text{ or }\; \neg < C > \;\text{ or }\; \forall x \;\text{ or }\; \neg\forall x \;\;,$$

where the latter two cases involve non-dull occurrences (that is, $\mathcal{B}$ is not a $1^{\underline{\text{st}}}$order formula). Further reduce it to proving $(*)$ for assertions of the form $F \longrightarrow < C > G$ and $F \longrightarrow \neg < C > G$ for $1^{\underline{\text{st}}}$order formulas F and G. (So we are down to two particular cases of when $M_{\mathcal{A}} = 1$.) Finally give separate, somewhat parallel, lengthy arguments for those two cases.

Now we proceed to do these in reverse order.

Proof of $(*)$ for $\mathcal{A}$ of the form $F \longrightarrow \neg < C > G$.

Since $[C]G := \neg < C > \neg G$, derivability of $F \longrightarrow \neg < C > G_1$ (for all F and G_1 where it's true) is equivalent to derivability of $F \longrightarrow [C]G$ (for all F and G where *it* is true), which is what we'll do. This proceeds by structural induction on C, simultaneously for all $1^{\underline{\text{st}}}$order formulas F and G.

When C is $x \dot{\leftarrow} t$: By $(\text{AX})_{\dot{\leftarrow}}$, $G^{[x \to t]} \longrightarrow [x \dot{\leftarrow} t]G$ is derivable, so it remains to show $F \longrightarrow G^{[x \to t]}$ is derivable, and then to apply hypothetical syllogism from propositional completeness. But the $1^{\underline{\text{st}}}$order formula $F \longrightarrow G^{[x \to t]}$ is is true in $\mathbf{N}$, because both $F \longrightarrow [x \dot{\leftarrow} t]G$ [by assumption] and $[x \dot{\leftarrow} t]G \longrightarrow G^{[x \to t]}$ [validity of half-$(\text{AX})_{\dot{\leftarrow}}$] are.

When C is $(C_1; C_2)$: Here $(\text{AX})_;$ is of course crucial, but expressivity is also involved. By expressivity, choose a $1^{\underline{\text{st}}}$order formula H so that the wfs $H \longleftrightarrow [C_2]G$ is true in $\mathbf{N}$. By the structural induction on C, the assertion $H \longrightarrow [C_2]G$ is derivable. So, by $(\text{UNAR})_{<>}$ and propositional completeness, the assertion $[C_1]H \longrightarrow [C_1][C_2]G$ is derivable. By hypothetical syllogism, and since $[C_1][C_2]G \longrightarrow [C]G$ is derivable, [by use of $(\text{AX})_;$], it remains only to show that the assertion $F \longrightarrow [C_1]H$ is derivable. This follows from the inductive hypothesis as long as that assertion is true. But in

$$F \;\longrightarrow\; [(C_1; C_2)]G \;\longrightarrow\; [C_1][C_2]G \;\longrightarrow\; [C_1]H$$

we have that 'each $\longrightarrow$' is true, respectively, by assumption, by validity of half-(AX),, and essentially by validity of half-$(\text{UNAR})_{<>}$, knowing the wfs $[C_2]G \longrightarrow H$ to be true.

When C is $\mathsf{whdo}(H)(D)$: Using expressivity, choose a 1st order formula J with $J \longleftrightarrow [C]G$ true in $\mathbf{N}$.

I claim that each of the following is derivable :

$$(1)\ F \to J\ ; \quad (2)\ J \to [C](J \wedge \neg H)\ ; \quad (3)\ J \wedge \neg H \to G\ .$$

If so, then (3), propositional completeness and $(\text{UNAR})_{<>}$ give the wfs $[C](J \wedge \neg H) \to [C]G$ to be derivable. So the double application of hypothetical syllogism to that assertion, (2) and (1) yields the desired derivation of $F \to [C]G$.

Verifying (1) : Both '$\to$'s in $F \to [C]G \to J$ are true in $\mathbf{N}$ (by assumption and choice of J), and therefore so is $F \to J$.

Verifying (2) : The rule $(\text{LOOP})_{[\ ]}$:

$$\frac{J \wedge H \to [D]J}{J \to [\mathsf{whdo}(H)(D)](J \wedge \neg H)}\ ,$$

leaves us only to derive $J \wedge H \to [D]J$. That follows from its truth in $\mathbf{N}$ by the overall induction on C of this part of the proof. To see its truth : for a contradiction, suppose $\underline{v}$ is such that $J \wedge H$ is true at $\underline{v}$, $||D||(\underline{v}) \neq err$, and J is false at $||D||(\underline{v})$. From 'while-do' semantics and the truth of H at $\underline{v}$, we get

$$||C||(\underline{v}) \ = \ ||C||(||D||(\underline{v}))\ .$$

Thus J being false at $||D||(\underline{v})$ gives, by the specification of J, that $[C]G$ is false at $||D||(\underline{v})$. Thus the right-hand side of the display is $\neq err$, and G is false there.

So the left-hand side of the display is $\neq err$. But then, since J, therefore $[C]G$, is true at $\underline{v}$, we see that G is true at the left-hand side of the display.

The underlined statements contradict the display.

Verifying (3) : That 1st order formula is a premiss, since its truth in $\mathbf{N}$ is easy to see : If $J \wedge \neg H$ is true at $\underline{v}$, then so are :

$\neg H$, giving $||C||(\underline{v}) = \underline{v}$, and

J , and so $[C]G$, giving either $||C||(\underline{v}) = err$ or G true at $||C||(\underline{v})$.

So, indeed, G is true at $\underline{v}$, as required.

Proof of $(*)$ for $\mathcal{A}$ of the form $F \longrightarrow\!< C > G$. This proof also proceeds by structural induction on C, simultaneously for all F and G, quite analogously to the previous proof, though the last case, of a 'while-do' command, is somewhat harder. Indeed, the first two cases may be done almost word-for-word as before, changing [ ] to <> everywhere. Actually they are more straightforward, since the earlier proof relied on observing that both the axioms (AX), modified with <> changed to [ ], are derivable.

This leaves only one case in the structural inductive proof of the derivability of $F \rightarrow\!< C > G$ when that assertion is true, namely the case

When C is $\mathsf{whdo}(H)(D)$:

Pick a $1^{\underline{\text{st}}}$order formula J and a variable $x \notin D \cup G \cup H$ such that

J is true at $\underline{v}$ if and only if : with $n := \underline{v}(x)$ we have

(α) $||D||^n(\underline{v}) \neq err$ and H is true at $||D||^i(\underline{v})$ for $0 \leq i < n$;
and

(β) $G \wedge \neg H$ is true at $||D||^n(\underline{v})$.

To prove that J can be found, first define a command

$$D^{\#} \ := \ \mathsf{whdo}(0 < x)(D \ ; \text{“}x \leftarrow x - 1\text{”}) \ .$$

Let $\vec{y} = (y_1, y_2, \cdots, y_r)$ be a list of distinct variables including all those in $D \cup G \cup H$. Here, as earlier, the command “$x \leftarrow x - 1$” has the effect, when the value of x is positive, of decreasing it by 1 without altering any y_i (but it can behave arbitrarily otherwise.) We'll 'contract' $\underline{v}$ down to just writing the relevant variables. Then

$$||D^{\#}||(n, \vec{a}) \ = \ (0, ||D||^n(\vec{a})) \ ,$$

where the first coordinate is the value of x and the vector coordinate is the value of $\vec{y}$ [and $(0, err)$ on the right-hand side would mean just err of course].

Now use expressivity for the assertion $<D^{\#}> H$ to find a $1^{\underline{\text{st}}}$order formula K such that

$$K^{[x \mapsto i \ , \ \vec{y} \mapsto \vec{a}]} \text{ is true in } \mathbf{N} \iff ||D^{\#}||(i, \vec{a}) \neq err \text{ and } H \text{ is true there .}$$

Use expressivity again, this time for the assertion $<D^{\#}> (G \wedge \neg H)$ to find a $1^{\underline{\text{st}}}$order formula L such that

$$L^{[x \mapsto n \ , \ \vec{y} \mapsto \vec{a}]} \text{ is true in } \mathbf{N} \iff ||D^{\#}||(n, \vec{a}) \neq err \text{ and } (G \wedge \neg H) \text{ is true there .}$$

Then let z be a 'new' variable, and define the required formula by

$$J \;:=\; \forall z\; ((z < x \to K^{[x \mapsto z]}) \wedge (z \approx x \to L^{[x \mapsto z]}))\;.$$

Then

$$J \text{ is true at } (n, \vec{a}) \iff J^{[x \mapsto n\ ,\ \vec{y} \mapsto \vec{a}]} \text{ is true in } \mathbf{N} \iff$$

$$\forall z\; ((z < n \to K^{[x \mapsto z\ ,\ \vec{y} \mapsto \vec{a}]}) \wedge (z \approx n \to L^{[x \mapsto z\ ,\ \vec{y} \mapsto \vec{a}]})) \text{ is true in } \mathbf{N} \iff$$

$$\text{true in } \mathbf{N} \text{ are : } K^{[x \mapsto i\ ,\ \vec{y} \mapsto \vec{a}]} \text{ for } 0 \leq i < n, \text{ and } L^{[x \mapsto n\ ,\ \vec{y} \mapsto \vec{a}]} \iff$$

$$||D^{\#}||(i, \vec{a}) \neq err \text{ and } H \text{ is true there for } 0 \leq i < n \text{ and}$$

$$||D^{\#}||(n, \vec{a}) \neq err \text{ and } G \wedge \neg H \text{ is true there} \iff$$

$$||D||^n(\vec{a}) \neq err, \text{ the formula } G \wedge \neg H \text{ is true there, and } H \text{ is true at } ||D||^i(\vec{a}) \text{ for } 0 \leq i < n\,,$$

as required.

I claim that the following are all true in $\mathbf{N}$.

$$(1)\ J^{[x \mapsto 0]} \to G \wedge \neg H\ ; \quad (2)\ J^{[x \mapsto x+1]} \to H \wedge < D > J\ ; \quad (3)\ F \to \exists x\ J\ .$$

Suspending disbelief in this for the moment, the following are then derivable :

$$J^{[x \mapsto 0]} \to \neg H \quad ; \quad J^{[x \mapsto x+1]} \to H \quad ; \quad J^{[x \mapsto x+1]} \to < D > J\ ;$$

the first two by (ORACLE), and the latter because it is true and therefore derivable by the induction on C.

By rule $(\text{LOOP})_{<>}$ and since $x \notin D \cup H$, we get a derivation of

$$\exists x\ J \;\to\; < C > J^{[x \mapsto 0]}\ .$$

But the $1^{\underline{\text{st}}}$order formula $J^{[x \mapsto 0]} \to G$ is a premiss, and therefore the assertion $< C > J^{[x \mapsto 0]} \to < C > G$ is derivable by rule $(\text{UNAR})_{<>}$. Finally $F \to \exists x\ J$ is true, so derivable. A double dose of hypothetical syllogism then shows $F \to < C > G$ to be derivable, as required.

It remains to check the truth of (1), (2) and (3), using the assumed truth of $F \to < \mathsf{whdo}(H)(D) > G$.

(1) Assume $J^{[x \mapsto 0]}$ is true at $\underline{w}$. Then J itself is true at $\underline{v} := \underline{w}^{(x \mapsto 0)}$. So $n = 0$ in the specification defining J. From (β) we get that $G \wedge \neg H$ is true

at $\underline{v}$. But since $\underline{v} \ \not\sim_x \ \underline{w}$ and $x \notin G \cup H$, we get, as required, that $G \wedge \neg H$ is true at $\underline{w}$.

(2) Assume that $J^{[x \mapsto x+1]}$ is true at $\underline{w}$.

Then J itself is true at the state $\underline{v} := \underline{w}^{(x \mapsto \underline{w}(x)+1)}$. In the specification of J, we have $n = \underline{v}(x) = \underline{w}(x) + 1 > 0$. So $i = 0$ occurs in $(\alpha)_{\underline{v},n}$, and we get that H is true at $||D||^0(\underline{v}) = \underline{v}$. So H is true also at $\underline{w}$ (which is half the required), since $\underline{v} \ \not\sim_x \ \underline{w}$ and $x \notin H$.

But, since $n \geq 1$, condition $(\alpha)_{\underline{v},n}$ also gives $||D||(\underline{v}) \neq err$. And so $||D||(\underline{w}) \neq err$, since $\underline{v} \ \not\sim_x \ \underline{w}$ and $x \notin D$. Furthermore, J is true at $||D||(\underline{w})$: We check this using the definition of J as follows. Here

$$||D||(\underline{w})(x) = \underline{w}(x) = n - 1 \ ,$$

so we must verify conditions $(\alpha)_{||D||(\underline{w}),n-1}$ and $(\beta)_{||D||(\underline{w}),n-1}$.

For the first, note that $||D||^{n-1}(||D||(\underline{w})) = ||D||^n(\underline{w}) \neq err$, since we have $||D||^n(\underline{v}) \neq err$. Also, for $0 \leq i < n-1$, the formula H is true at $||D||^i||D||(\underline{w})) = ||D||^{i+1}(\underline{w})$, since again, $\underline{w}$ may be replaced by $\underline{v}$, and because $i + 1 < n$, making use of $(\alpha)_{\underline{v},n}$.

For $(\beta)_{||D||(\underline{w}),n-1}$, we again use the fact that $||D||^{n-1}||D||(\underline{w})) = ||D||^n(\underline{w})$, and $G \wedge \neg H$ is indeed true there, because it is true at $||D||^n(\underline{v})$.

The underlined above show that $J^{[x \mapsto x+1]} \rightarrow\!< D > J$ is true in $\mathbf{N}$, completing the discussion of (2).

(3) Assume that F is true at $\underline{v}$. Our basic assumption that the assertion $F \rightarrow\!< \mathsf{whdo}(H)(D) > G$ is true in $\mathbf{N}$ yields that

$$||\mathsf{whdo}(H)(D)||(\underline{v}) \ = \ ||C||(\underline{v}) \ \neq \ err \ ,$$

and that G is true at $||C||(\underline{v})$. Thus, there is a (unique of course) $n \geq 0$ such that $(\alpha)_{\underline{v},n}$ and $(\beta)_{\underline{v},n}$ hold, by the semantics of 'whdo'. Now define $\underline{w}$ by $\underline{w} \ \not\sim_x \ \underline{v}$ and $\underline{w}(x) = n$. By the definition of J, the latter formula is true at $\underline{w}$. Since $\underline{w} \ \not\sim_x \ \underline{v}$, it is immediate from the basic (Tarski) definition of truth for an existential $1^{\underline{\text{st}}}$order formula that $\exists x \ J$ is true at $\underline{v}$, as required. (But not necessarily J itself, so this argument for adequacy fails, with the existential quantifier missing in the last rule, as it must, by Theorem 1.)

This finally completes the (structural induction on C) proof of $(*)$ for $\mathcal{A}$ of the form $F \longrightarrow\!< C > G$.

Interruption. At this point it becomes easy to explain in more detail the part of [**A**] which relates to this paper. Suitably translated, the language there contains a wfs only if it is either

(i) a $1^{\underline{\text{st}}}$order formula ; or

(ii) of the form $F \rightarrow < C > G$ where each of F and G is a $1^{\underline{\text{st}}}$order formula (our standard notation).

Once corrected as above for soundness, the proof system there can be interpreted to consist of (ORACLE), the $\rightarrow$-half of $(\text{AX})_{\doteq}$, $(\text{LOOP})_{< >}$, and just two more rules which are ubiquitous in the Floyd-Hoare literature, as follows: (Actually there are *three* more rules in [**A**], but we are suppressing everything about the [redundant] if-then-else command.)

$$\frac{F \rightarrow < C_1 > G \ , \ G \rightarrow < C_2 > H}{F \rightarrow < (C_1 ; C_2) > H} \qquad \frac{F \rightarrow F_1 \ , \ F_1 \rightarrow < C > G_1 \ , \ G_1 \rightarrow G}{F \rightarrow < C > G}$$

It is dead easy to derive these two rules within the system here, since hypothetical syllogism is part of propositional completeness. Use $(\text{UNAR}_{< >}$ as well, and nothing else. But the main thing is that this system is (Cook-) complete for that restricted language, i.e. every wfs in that language which is true in **N** is derivable using that shorter system. Writing out the case we dealt with just before this interruption, one sees that a few lines can actually be omitted to prove completeness for the above system, though the main point, the existence of J, is no shorter. A very brief sketch of this, omitting J's existence, and culled also from [**H**], is presented in [**A**].

It does seem much more natural however to deal with the entire language, though the above is of interest in isolating just how much of the proof system is needed to get one's hands on the classical Floyd-Hoare statements. On the other hand, surely there is some interest in the possibility of, say, post-conditions which say something about the behaviour of some other program (i.e. command), not just the usual ones which say something static about numbers.

Not all interesting derivable rules are quite as easy to capture from the system as the two above. For example, rules approximating the following, which is a kind of generalization of $(\text{LOOP})_{< >}$, have appeared in the literature:

$$\frac{t \approx x \wedge G \wedge H \ \rightarrow \ < C > ((t < x \vee \neg G) \wedge H)}{H \ \rightarrow \ < \mathsf{whdo}(G)(C) > (H \wedge \neg G)} \quad \text{for} \ \ x \notin C \cup G \cup H \cup t \ .$$

And, once one has the Cook completeness, the following are clearly derivable since true, but elegantly?

$$< C > \mathcal{A} \ \rightarrow \ [C]\mathcal{A} \ \ ;$$

$$< C > \mathcal{A} \wedge < C > \mathcal{B} \ \longleftrightarrow \ \ < C > (\mathcal{A} \wedge \mathcal{B}) \ \ .$$

(If we were using a non-deterministic command language, these likely wouldn't be true—see the postscript at the end.)

To continue with the proof of Theorem 2,

Lemma E. *For any wfs $\mathcal{C}$, there is a propositional derivation of a wfs of the form*

$$\mathcal{C} \ \longleftrightarrow \wedge_\alpha \mathcal{D}_\alpha \ \ ,$$

where the finite conjunction on the right-hand side has each $\mathcal{D}_\alpha$ as a finite disjunction of wfs's, each of which has the following form : either each is a $1^{\underline{\text{st}}}$order formula , or else at least one of them has the form sym$\mathcal{B}$ *for our four possible* sym*'s, where $\mathcal{B}$ is not a $1^{\underline{\text{st}}}$order formula when* sym *is either $\forall x$ or $\neg\forall x$.*

The proof of this proceeds by structural induction on $\mathcal{C}$, and is simplified by simultaneously proving the dual fact, where the prefixes "con" and "dis" trade places. The initial case is trivial. In two inductive cases, namely $\mathcal{A} \mapsto \forall x \ \mathcal{A}$; and $\mathcal{A} \mapsto < C > \mathcal{A}$, one simply has a single conjunct (resp. disjunct), which itself is a single disjunct (resp. conjunct). The inductive step for negations is easy because of proving the duals simultaneously : up to cancellation of double negations, deMorganization converts each expression from the right-hand side of the lemma to its dual. The symmetry breaks for the final case of conjuncting two assertions. For the actual lemma statement, the result is numbingly obvious. For the dual statement, it is run-of-the-mill obvious by using distributivity of $\wedge$ over $\vee$. (All we are really talking about here is conjunctive and disjunctive form.)

Since a conjunction is derivable (resp. true in $\mathbf{N}$) if and only if each conjunct has the same property (using propositionality of our system for the derivability), and since every true $1^{\underline{\text{st}}}$order formula is a premiss, the lemma reduces the question to proving $(*)$ only for assertions of the form $\mathcal{E} \vee$ sym$\mathcal{B}$. (One may use $\mathcal{E} = \ 0 \approx 1$ for any conjunct consisting of a single disjunct sym$\mathcal{B}$.) Taking $\mathcal{A}$ as $\neg\mathcal{E}$,

we need only prove $(*)$ **for assertions of the form** $\mathcal{A} \rightarrow \mathsf{sym}\mathcal{B}$, where, since $\mathcal{B}$ is not a $1^{\underline{\text{st}}}$order formula in two of four relevant cases, the inductive assumption applies to all assertions whose 'M-function' does not exceed that of one or the other of $\mathcal{A}$ or $\mathcal{B}$. All we are saying here is that

$$M_{\mathcal{A}\rightarrow\mathsf{sym}\mathcal{B}} \;=\; M_{\mathcal{A}} + M_{\mathcal{B}} + 1 \;.$$

By expressivity, choose A and B, each a $1^{\underline{\text{st}}}$order formula , such that both $A \longleftrightarrow \mathcal{A}$ and $B \longleftrightarrow \mathcal{B}$ are true in $\mathbf{N}$.

The proof will now be completed by considering each of the four possibilities for sym.

$\underline{\mathsf{sym} \text{ is } < C >}$: Use hypothetical syllogism after establishing the derivability of the three '$\rightarrow$' below :

$$\mathcal{A} \;\rightarrow\; A \;\rightarrow\; < C > B \;\rightarrow\; < C > \mathcal{B} \;.$$

The wfs $\mathcal{A} \rightarrow A$ is true, therefore derivable by the induction on the number M.
The wfs $B \rightarrow \mathcal{B}$ is true, therefore derivable by induction, and then so is the wfs $< C > B \rightarrow < C > \mathcal{B}$ derivable, using $(\text{UNAR})_{<>}$.
Derivability of $A \rightarrow < C > B$ follows from its truth and the second of the earlier agonizingly established $(*)$'s. Its truth is clear, since it 'factors into true wfs's' :

$$A \;\rightarrow\; \mathcal{A} \;\rightarrow\; < C > \mathcal{B} \;\rightarrow\; < C > B \;.$$

$\underline{\mathsf{sym} \text{ is } \forall x}$: Just replace $< C >$ by $\forall x$ everywhere in the previous paragraph, and appeal to $(\text{UNAR})_{\forall}$ rather than $(\text{UNAR})_{<>}$, and get derivability of $A \rightarrow \forall x\ B$ much more easily, as a premiss.

$\underline{\mathsf{sym} \text{ is } \neg < C >}$: Just replace $< C >$ by $\neg < C >$ everywhere in the second previous paragraph, and make a few tiny wording changes :

Use $\mathcal{B} \rightarrow B$ true, so derivable by induction on the number M, to get $< C > \mathcal{B} \rightarrow < C > B$, and thence $\neg < C > B \rightarrow \neg < C > \mathcal{B}$ both derivable.

This case of course uses also the somewhat less agonizingly established version of $(*)$ giving the derivability of $A \rightarrow \neg < C > B$.

sym is $\neg\forall x$: Write out the previous paragraph properly, then replace each $\neg < C >$ by $\neg\forall x$, again appealing simply to the 'premissiveness' of $A \rightarrow \neg\forall x\ B$, rather than needing the proof of the derivability of the wfs $A \rightarrow \neg < C >\ B$.

So we're done : the proof system is indeed Cook-complete.

It is surprising that the book [**H-K-T**] on dynamic logic seems to have no information at all on specializing and converting to assertions about deterministic command languages whose grand-daddy is the 'while' language.

Postscript. The minor problem isolated by Theorem 1 apparently has precursors. The underlying dynamic logic system, from which the system in [**H**] for the (deterministic) language 'while' was derived, has a rule which is the progenitor of (LOOP$_{<>}$ (see [**H**], p. 32, rule (C^*); and also the more recent book [**H-K-T**], p.336, "convergence" rule). Either this rule may need an existential quantifier, or else the "high school prenexing rule" mentioned earlier may need to be added to those systems, to be sure of completeness. An analogue of Theorem 1 to establish this could be worth the effort, however unappealing. In any case, deriving the correct deterministic rule from the non-deterministic rule does seem to require some addition to these systems.

Also of interest in this context is the intriguing system for dynamic logic using a command language with recursive procedures in [**H**], pp. 44-55. Any neophyte like the author who has tried for a 'while' program, and then realized how much easier it is to write a procedure for calculating, say, Ackermann's function using a language *with* recursion, will appreciate the major 'step-up' having this feature is, despite it not adding any abstract computability. A hopefully elegant, completely deterministic, system for a basic language with recursion, and for which the 'verification' language includes all $1^{\underline{st}}$order constructions (and in particular both total and partial correctness) seems not to exist, except implicitly, in the literature, and would appear to be of at least academic interest.[3] This would be another poor cousin of dynamic logic, such as the one exposited in this paper for 'while'. To do this is presumably not particularly hard or original, more a matter of interpreting

[3]This is out-of-date—the three later papers than this on my web page do more than that.

and compiling from some papers of the last 15 years, such as **[A-dB], [N]** and **[Sc]**. These build on the somewhat delayed realization that the fix in **[A]** of the lack of completeness, despite its claim, in **[So]**, unfortunately itself then introduced an unsoundness (not entirely dissimilar to the one pointed out earlier in this paper, though maybe more related to the 'meta'ness of the main rule related to the command calling the recursively defined procedure).

This system in **[H]** referred to above is apparently inescapably non-deterministic as is. The rules (displayed below) for the procedure call command, both for the $< C >$- and the $[C]$- construction, involve a non-deterministic 'achieve'-command which seems hard to purge from the system, if we wished to specialize to a self-contained system for a deterministic language with recursion. (However **[A-dB]** has a briefly-discussed claim concerning how to do this.) As is, its completeness also seems doubtful, in the same way that we have discussed above, related to Theorem 1 : without adding that prenexing rule, deriving (within that non-deterministic system above) the most basic example (below) which involves a genuine recursion does not seem to make itself obvious, though something simple might have escaped notice.

A sketch of this in notation closer to the rest of this paper follows. Alter the 'while' command language by including call X as a new atomic command, for just *one* 'procedure variable', X; and also include ∇XC as a 'declaration-and-call' command, whose semantics would be the same as that of the 'block': begin declare X to 'be' C ; call X end , in a language closer to ALGOL. (The ∇ behaves a bit like a quantifier.) Also include if-then-else; and throw away the while-do command as now being redundant. (At this point we have the deterministic command language inside the non-deterministic one of Harel.) The dynamic logic language of verification is constructed exactly as before, except that commands C are from the new command language. Then alter the system of the main part of this paper by

(1) adding that prenexing rule and the earlier mentioned if-then-else rule;

(2) adding the axiom

$$(\mathcal{A} \to [\nabla XC]\mathcal{B}) \;\to\; (\mathcal{A} \wedge \mathcal{C} \to [\nabla XC](\mathcal{B} \wedge \mathcal{C}))$$

for $\mathcal{C}$ and C having no common variables;

(3) replacing the two (LOOP)-rules by the following two (RCRS)-rules of Harel, explained below:

(RCRS)$_{[\,]}$:

$$\frac{y \approx z \ \rightarrow \ [C^{[X \rightarrow F^{y \leftarrow\!\div\ z}\ ]}]F}{y \approx z \ \rightarrow \ [\nabla XC]F}$$

where all variables in C are from y ;

(RCRS)$_{<>}$:

$$\frac{F^{[x \rightarrow x+1]} \rightarrow < C^{[X \rightarrow F^{y \leftarrow\!\div\ z}\ ]} > y \approx z \ , \ \neg F^{[x \rightarrow 0]}}{F \ \rightarrow \ \ < \nabla XC > y \approx z}$$

where x is in neither y nor z, and all variables in C are from y; and finally

(4) adding the axiom

$$[F^{y \leftarrow\!\div\ z}\ ]G \ \leftrightarrow \ \forall u \ (F^{[z \rightarrow u]} \rightarrow G^{[y \rightarrow u]}) \ .$$

In (3) and (4), the non-deterministic 'achieve'-command, $F^{y \leftarrow\!\div\ z}$, for any $1^{\underline{\text{st}}}$order formula F, has two finite equal-length sequences of variables, y and z, all different from each other. But, for our purposes below, take y and z to simply be a pair of distinct variables. The semantics of $F^{y \leftarrow\!\div\ z}$ is that it sends any input to a (non-deterministically selected) output which agrees with the input except (possibly) at y, that value being altered by $F^{y \leftarrow\!\div\ z}$ to any value which, if used to replace z's value in the original input, would make F true.

Except for adding the prenex rule (and using the ite-deterministic command/rule in place of that for $\cup$), this is Harel's system for the dynamic logic language based on the extension of the command language for recursive commands to include those "achieve" commands as well.

Now the analogue here, of that 'simplest possible genuine' while-do command in the example much earlier, is the 'simplest possible' recursive command ∇XC where

$$C \ = \ \mathsf{ite}(0 < y)(y \leftarrow\!\div\ y - 1 \ ; \mathsf{call}X)(bugga)$$

where *bugga* is any null command; i.e. it computes the identity function in very close to zero steps. The statement to be derived is $< \nabla XC > y \approx 0$.

Allowing the prenex rule as part of the system, one could derive this as follows. Take F to be $\ 0 \approx z \ \wedge \ y < x \ $ in (RCRS)$_{<>}$. Checking the truth in $\mathbf{N}$ of the antecedants in that rule is easy, and so one would have

first derived the non-1$^{\underline{\text{st}}}$order of these, the rule then yielding a derivation of $F \rightarrow < \nabla XC > y \approx z$. From the prenex rule, one then has a derivation of $\exists x F \rightarrow < \nabla XC > y \approx z$. But $0 \approx z \rightarrow \exists x F$ is logically valid, so $0 \approx z \rightarrow < \nabla XC > y \approx z$ is derivable. Logical validity also gives

$$0 \approx z \rightarrow 0 \approx z \wedge 0 \approx z \quad \text{and} \quad y \approx z \wedge 0 \approx z \rightarrow 0 \approx y \ ;$$

and so, using $(\text{UNAR})_{<>}$ one derives

$$< \nabla XC > (y \approx z \wedge 0 \approx z) \ \rightarrow \ < \nabla XC > 0 \approx y \ .$$

Two hypothetical syllogisms applied to

$$0 \approx z \rightarrow 0 \approx z \wedge 0 \approx z \rightarrow < \nabla XC > (y \approx z \wedge 0 \approx z) \rightarrow < \nabla XC > 0 \approx y \ ,$$

the middle arrow of which uses the earlier new axiom and (MP), now gives a derivation of $0 \approx z \rightarrow < \nabla XC > 0 \approx y$. To finish, use the prenex rule again, logical validity of $\exists z \ 0 \approx z$ and (MP).

Details in [**H**] of the completeness proof are too sketchy to know for sure whether one really cannot get round taking the prenex rule as part of the system. A derivation as in the above paragraph without it would be interesting to see. Perhaps an analogue of Theorem 1 here shows this to be impossible.

References

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

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

[**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.

[**H**] Harel, David, *First-Order Dynamic Logic.* Lecture Notes in CS # 68, Springer, 1979. See also the same (correctible for validity and adequacy) version of $(\mathrm{LOOP})_{<>}$ as (4.2), p. 69, in: *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.

[**N**] Nipkow, Tobias, *Hoare Logics for Recursive Procedures and Unbounded Nondeterminism.* XXXXXXS IAM. J. C OMPUT. 10(3), 2003+, n+1-n+16.

[**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.

[**Sc**] 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.

[**So**] 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.

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