Arithmetical Completeness in Logics of Programs

David Harel
Laboratory for Computer Science
Massachusetts Institute of Technology, Cambridge, MA 02139

Abstract.

We consider the problem of designing *arithmetically complete* axiom systems for proving general properties of programs; i.e. axiom systems which are complete over *arithmetical universes*, when all first-order formulae which are valid in such universes are taken as axioms. We prove a general Theorem of Completeness which takes care of a major part of the responsibility when designing such systems. It is then shown that what is left to do in order to establish an arithmetical completeness result, such as those appearing in [12] and [14] for the logics DL and DL$^+$, can be described as a chain of reasoning which involves some simple utilizations of arithmetical induction. An immediate application of these observations is given in the form of an arithmetical completeness result for a new logic similar to that of Salwicki [22]. Finally, we contrast this discipline with Cook's [5] notion of *relative* completeness.

1. Introduction.

In the past ten years substantial progress has been made in the field of *logics of programs*, a field which concerns itself with the design of mathematical tools (formal and informal) for reasoning about programs. Such logics should provide means for expressing interesting properties of programs, and, hopefully, methods for proving them when true. In this paper we would like to focus on proof theory and to single out a specific trend in its development.

The logics we consider are the formal logics DL (*dynamic logic*) and DL$^+$. DL was suggested as a powerful logic of programs by Pratt [21] and was further developed in [12] and [9]. DL$^+$ was defined in [14]. These logics are extensions of first-order predicate calculus, and are capable of expressing a very wide variety of properties such as partial and total correctness, equivalence of programs, etc.

We are interested in the formulae of DL (DL-*wffs*) which are true in all *states* of a given *universe* U (U-*valid* DL-wffs). With each universe U there is associated a *domain* D, and in all states of U, function and predicate symbols are interpreted as functions and predicates over D. Our approach is to consider those universes A (*arithmetical* universes) in which the domain essentially includes the natural munbers, and in which $+$, $\cdot$ and 0 are given their standard interpretation. In this framework one can show that with each DL-wff P there is associated a first-order formula F_P which is equivalent to P in any state in A.

Our approach to proving the fact that a DL-wff P is A-valid is to design an axiom system with which P can be syntactically transformed into F_P. Then, for any arithmetical

universe A, all A–valid first–order formulae are taken as axioms, and thus, once this transformation has been carried out, the set of axioms should contain F_p *iff* P is A–valid, and thus the proof is complete. This approach we call *arithmetical axiomatization.* If the axiom system has the property that *any* A–valid DL–wff P can indeed be transformed into the A–equivalent first–order F_p, we say that it is an *arithmetically complete* axiom system, for then a *proof* of it is guaranteed to exist.

Section 2 contains rigorous definitions of the concepts of state and universe and of the logics DL and DL$^+$. In Section 3 we define the notion of an arithmetical universe, present axiom systems for DL and DL$^+$, and state the arithmetical completeness results we were able to prove for them in [12] and [14].

Sections 4 and 5, the main sections of the paper, are devoted to the establishment of a general framework in which the process of designing such axiom systems and proving their arithmetical completeness can be derived out of simple inductive considerations. In Section 4 we prove a very general theorem, the Theorem of Completeness, in which the main inductive step in the proofs of *all* the completeness results appearing in [12], [14], [10] and [9] is captured in an abstract way. Thus, it can be seen that for *any* "DL–like" logic, once arithmetical completeness can be established for very simple formulae which involve only one appearance of a program this theorem can be applied to obtain the completeness result for the *full* logic. In Section 5 we describe a way in which these "basic" results for one–program formulae can be developed for DL and DL$^+$ using straightforward reasoning. Thus, the intuition behind the inductive assertion method of Floyd [7] and Hoare [15], which is embodied in our axiomatization of DL, can be viewed as a special case of a rather broad observation.

Section 6 contains an immediate application of these ideas, by describing the rather effortless development of an arithmetically complete axiomatization of a new logic, which borrows the $\cap\alpha$ operator of [22]. In Section 7 we give a precise definition, based on the concept of a universe, of Cook's [5] notion of *relative completeness*, and then discuss some of the literature regarding this concept, and comment as to the relationship between it and arithmetical completeness.

2. *The Definition of DL and DL$^+$.*

We define regular first–order dynamic logic (DL) as follows: We are given sets of *function symbols* and *predicate symbols*, each symbol with a fixed nonnegative *arity*. We assume the inclusion of the special binary predicate symbol "=" (equality) in the latter set. We denote predicate symbols by p, q,... and k–ary function symbols for k>0 by f, g,... Zeroary function symbols are denoted by z,x,y,... and are called *variables*. A *term* is some k–ary function symbol followed by a k–tuple of terms, where we restrict ourselves to terms resulting from applying this formation rule finitely many times only. For a variable x we abbreviate x() to x, thus f(g(x),y) is a term provided f and g are binary and unary, respectively. An *atomic formula* is a k–ary predicate symbol followed by a k–tuple of terms.

We define by simultaneous induction the set RG of first–order regular programs and the set of DL–wffs:

(1) For any variable x and term e, $x \leftarrow e$ is in RC,
(2) For any program-free (see below) DL-wff P, P? is in RC,
(3) For any α and β in RC, $(\alpha;\beta)$, $(\alpha \cup \beta)$ and α^* are in RC,
(4) Any atomic formula is a DL-wff,
(5) For any DL-wffs P and Q, α in RC and variable x,
$\neg P$, $(P \vee Q)$, $\exists x P$ and $\langle\alpha\rangle P$ are DL-wffs.

A DL-wff which contains no occurrence of a program of RC is called *program free*, or simply a *first order* formula. Programs of the form indicated in (1) and (2) are called, respectively, (simple) *assignments* and (simple) *tests*. We use $\wedge$, $\supset$ and $\equiv$ for abbreviations in the standard way and, in addition, abbreviate $\neg\exists x \neg P$ to $\forall x P$, and $\neg\langle\alpha\rangle\neg P$ to $[\alpha]P$.

The semantics of DL is based on the concept of a state. A *state* $\mathcal{J}$ consists of a non-empty domain D and a mapping from the sets of function and predicate symbols to the sets of functions and predicates over D, such that to a k-ary function symbol f (resp. predicate symbol p) there corresponds a total k-ary function (resp. predicate) over D, denoted by $f_{\mathcal{J}}$ (resp. $p_{\mathcal{J}}$). In particular, to a variable there corresponds an element of the domain, and to a 0-ary predicate symbol (propositional letter) a truth value (*true* or *false*). The standard equality predicate over D is that which corresponds to the equality symbol (=). We will sometimes refer to the domain of $\mathcal{J}$ as $D_{\mathcal{J}}$. Observe that the way states are defined no distinction is made between what are normally called variables and constants. These will, however, be defined below for simple universes.

We denote by Γ the collection of all possible states, which we call the *grand universe*. Our semantics will assign to a program α a binary relation $m(\alpha)$ over Γ, and to a formula P, a subset of Γ consisting of those states which *satisfy* P. In the sequel, however, we will be interested in special subsets of Γ, namely, universes:

A *pseudo-universe* U is a set of states, all of which have a common domain D. A function symbol f (resp. predicate symbol p) is called *uninterpreted in* U if for every state $\mathcal{J} \in U$ and for every function F (resp. predicate P) over D there exists $\mathcal{G} \in U$ such that $\mathcal{J}$ and $\mathcal{G}$ differ at most in the value of f (resp. p), which in $\mathcal{G}$ is F (resp. P).

Notation: for any function C: $A \to B$, arbitrary element e, and $a \in A$, we define $[e/a]C$ to be the function with domain A and range $B \cup \{e\}$ giving the same values at points in $A-\{a\}$ as C, and such that $C(a)=e$. Thus, the situation described above for uninterpreted f is simply $\mathcal{G}=[F/f]\mathcal{J}$.

A symbol is called *fixed in* U if its value is the same in all states of U. Thus, "=" is fixed in any universe. A *universe* is a pseudo-universe in which every predicate symbol is fixed, and in which every function symbol is either fixed or uninterpreted. A universe is called *simple* if the only uninterpreted symbols in it are a designated set of variables. In a simple universe the fixed variables can be called *constants* following ordinary usage.

The value of a term $e = f(e1,\ldots,ek)$ in a state $\mathcal{J}$ is defined inductively by

$$e_{\mathcal{J}} = f_{\mathcal{J}}(e1_{\mathcal{J}},\ldots,ek_{\mathcal{J}}).$$

We now define by simultaneous induction the binary relation over Γ corresponding to a program α of RC, and those states $\mathcal{J}$ in Γ which satisfy a DL-wff P. The relation will be

denoted by $m(\alpha)$ and for the latter we write $\mathcal{I}\models P$. $(\mathcal{I},\mathcal{J})$ being an element of $m(\alpha)$ can be thought of as representing the fact that there exists a *computation sequence* (or *path*) of α starting in state $\mathcal{I}$ and terminating in $\mathcal{J}$. Thus, $\mathcal{I}\models[\alpha]P$ will be seen to be making an assertion about *all* terminating computations of α starting in state $\mathcal{I}$; namely the assertion that the final states of these computations satisfy P. Similarly, $\mathcal{I}\models\langle\alpha\rangle P$ asserts the *existence* of a terminating computation of α starting in state $\mathcal{I}$, which ends in a state satisfying P.

(1') For any variable x and term e,
$m(x\leftarrow e) = \{(\mathcal{I},\mathcal{J}) \mid \mathcal{J}=[e_{\mathcal{I}} / x]\mathcal{I}\}$,

(2') for any program-free DL-wff P,
$m(P?) = \{(\mathcal{I},\mathcal{I}) \mid \mathcal{I}\models P\}$,

(3') For any α and β in RG,
$m(\alpha;\beta) = m(\alpha) \circ m(\beta)$, (composition of relations),
$m(\alpha\cup\beta) = m(\alpha) \cup m(\beta)$, (union of relations),
$m(\alpha^*) = (m(\alpha))^*$, (reflexive transitive closure of a relation),

(4') For an atomic formula $p(e1,\ldots,ek)$,
$\mathcal{I}\models p(e1,\ldots,ek)$ whenever $p_{\mathcal{I}}(e1_{\mathcal{I}},\ldots,ek_{\mathcal{I}})$ is true,

(5') For any DL-wffs P and Q, α in RG and variable x,
$\mathcal{I}\models\neg P$ iff it is not the case that $\mathcal{I}\models P$,
$\mathcal{I}\models(P\vee Q)$ iff either $\mathcal{I}\models P$ or $\mathcal{I}\models Q$,
$\mathcal{I}\models\exists xP$ iff there exists an element d in $D_{\mathcal{I}}$ such that $[d / x]\mathcal{I}\models P$,
$\mathcal{I}\models\langle\alpha\rangle P$ iff there exists a state $\mathcal{J}$ such that $(\mathcal{I},\mathcal{J})\in m(\alpha)$ and $\mathcal{J}\models P$.

Note that the only kinds of formulae, whose truth in state $\mathcal{I}$ depends possibly upon states other than $\mathcal{I}$, are those containing subformulae of the form $\exists xP$ and $\langle\alpha\rangle P$.

In this paper we will primarily be interested in investigating the truth of formulae in a given simple universe U. However, one can see that for some $\mathcal{I}\in U$ and some assignment $x\leftarrow e$, the unique state $\mathcal{J}$ such that $(\mathcal{I},\mathcal{J})\in m(x\leftarrow e)$, i.e. the state $[e_{\mathcal{I}} / x]\mathcal{I}$, might not be in U at all. We outlaw this phenomenon by adopting, from now on, the convention that in the context of a given universe, the only programs we consider are those in which the variables assigned to (e.g. x in $x\leftarrow e$) and the quantified variables (e.g. x in $\exists xP$) are uninterpreted. Thus, for $\mathcal{I}\in U$ and for any DL-wff P, the truth of $\mathcal{I}$ in P can be seen to depend only on states in U.

We abbreviate $(\mathcal{I},\mathcal{J})\in m(\alpha)$ to $\mathcal{I}\alpha\mathcal{J}$, and also take the liberty of writing

$\mathcal{I}\models\langle\alpha\rangle P$ iff $\exists\mathcal{J}(\mathcal{I}\alpha\mathcal{J} \wedge \mathcal{J}\models P)$, and thus we have also
$\mathcal{I}\models[\alpha]P$ iff $\forall\mathcal{J}(\mathcal{I}\alpha\mathcal{J} \supset \mathcal{J}\models P)$.

Given a universe U, we say that a DL-wff P is U-*valid* (and write $\models_U P$) if for every $\mathcal{I}\in U$ we have $\mathcal{I}\models P$. We say P is *valid* ($\models P$) if it is U-valid for every universe U in which, in line with the above convention, the assigned and quantified variables of P are uninterpreted.

The following are examples of valid DL-wffs:

$[(x=z \wedge y=u)?;(x\leftarrow f(x) \cup y\leftarrow f(y))](x=z \vee y=u)$,
$x=y \supset [(x\leftarrow f(f(x)))^*]\langle(y\leftarrow f(y))^*\rangle x=y$,
$x=y \supset [(x\leftarrow f(x))^*](p(x) \supset (x=y \vee \langle y\leftarrow f(y);(y\leftarrow f(y))^*\rangle p(y)))$.

The first asserts that at most one of the components of ∪ is executed. The second states that the process of repeatedly applying a function composed with itself is a special case of that of repeatedly applying it. The third asserts essentially that the process of achieving a property of x by repeatedly applying f can be simulated in y.

Denote by ***N*** the *simple* universe of *pure arithmetic*; i.e., the domain D is the set of natural numbers, and +, · and 0 are fixed with their standard interpretations. We freely use standard arithmetical abbreviations such as ≥, *gcd*, etc. The following are ***N***–valid DL–wffs:

$\langle(x\leftarrow x-1)^*\rangle x=0$,
$y>0 \vee \langle y=0?\rangle true$ (We abbreviate x=x to *true* and ¬*true* to *false*),
$[(x=x' \wedge y=y' \wedge x'y>0)?]\langle(x\neq y?;(x>y?;x\leftarrow x-y \cup x<y?;y\leftarrow y-x))^*;x=y?\rangle x=gcd(x',y')$.

The last example asserts that the program inside the diamond, under the assumption that its two inputs are positive integers, terminates and computes the *gcd* of these inputs. This program can be written in more popular terms as:

while x≠y *do if* x>y *then* x←x−y *else* y←y−x *end.*

We adopt the standard definition of a *free occurrence* of a variable x in a first order formula Q to be an occurrence of x which is not in any subformulae of the form ∃xP. Any other occurrence is called *bound*. Also, we define Q^e_x for variable x and term e to be the formula which is obtained from Q by uniformly renaming all bound variables of Q which appear in e and replacing all free occurrences of x by e.

We now describe our extension of DL, namely DL$^+$, for dealing with infinite computations. The definitions of DL$^+$ given in [14] and [9] amount to exactly the same, but are presented differently; in [14] we use a "divergence state", and in [9] computation trees. However, once the present definition or any one of the others has been given, the concepts are well defined and the rest of the treatment is identical.

For any program α∈RG and state $\mathcal{I}\in\Gamma$, a Boolean constant $loop_\alpha$ is added to the vocabulary of DL, giving DL$^+$. The semantics of $loop_\alpha$ are given by induction on the structure of α as follows:

$loop_{x\leftarrow e} \equiv false$,
$loop_{P?} \equiv false$,
$loop_{\alpha\cup\beta} \equiv (loop_\alpha \vee loop_\beta)$,
$loop_{\alpha;\beta} \equiv (loop_\alpha \vee \langle\alpha\rangle loop_\beta)$.

For α^* we define $\mathcal{I}\models loop_{\alpha^*}$ iff either $\mathcal{I}\models\langle\alpha^*\rangle loop_\alpha$, or there exist states $\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2, \ldots\in\Gamma$ such that $\mathcal{I}_0=\mathcal{I}$ and $(\forall i\geq 0)(\mathcal{I}_i\alpha\mathcal{I}_{i+1})$.

(We remark that our original notation in [14] had $dv(\alpha)$ standing for $\neg loop_\alpha$. We now prefer the notation $loop_\alpha$ which is what is used in [19] and [9].)

Now abbreviate the DL$^+$–wff $(\langle\alpha\rangle P \vee loop_\alpha)$ to $\langle\alpha\rangle^+P$, and $\neg\langle\alpha\rangle^+\neg P$ to $[\alpha]^+P$. Thus we have

$\langle\alpha\rangle^+ false \equiv loop_\alpha$, and
$[\alpha]^+ true \equiv \neg loop_\alpha$.

The intuition is that $\langle\alpha\rangle^+false$ (i.e. $loop_\alpha$) is true in a state $\mathcal{J}$ iff α "can enter an infinite loop"; i.e. "α has a divergence". $[\alpha]^+true$ is true in $\mathcal{J}$ iff α is "divergence-free".

We refer the reader to [21], [12] and [9] for many more details and results concerning DL and DL$^+$.

3. *Arithmetical Axiomatization of DL and DL$^+$.*

We would like to supply a syntactic characterization of the U-valid DL-wffs and DL$^+$-wffs for specific universes U; namely those which "contain" the simple universe of arithmetic, $\boldsymbol{N}$. This characterization will be in the form of sound axiom systems for DL and DL$^+$ which make explicit use of variables that range over the natural numbers. For any such "arithmetical" universe A, we take all A-valid first order formulae as further axioms, and show that then the axiom systems are A-complete, i.e. that a proof in them does indeed exist for any A-valid DL-wff. This property we will term *arithmetical completeness.*

An *arithmetical universe* A is a universe in which the domain includes the set of natural numbers, the binary function symbols + and · are fixed and given their standard meanings (addition and multiplication respectively) when applied to the natural numbers in the domain, and 0 is a fixed zeroary-order function symbol interpreted as the natural number "zero". Furthermore there is a fixed unary predicate symbol *nat* with the interpretation "$nat_\mathcal{J}(d)$ is true iff d is a natural number", that is, for every state $\mathcal{J}$, $\{d\in D_\mathcal{J} \mid nat_\mathcal{J}(d)\}$ is the set of natural numbers. Thus, we are able to distinguish the natural numbers in the domain from the other elements, and we do not care, say, what the value of x+y is in state $\mathcal{J}$ when it is not the case that $nat_\mathcal{J}(x_\mathcal{J})$ holds. Note that one particular arithmetical universe is the simple universe $\boldsymbol{N}$ of "pure arithmetic" in which *nat* is identically true.

Throughout the rest of the paper, A stands for any arithmetical universe, and L for the set of first-order formulae. When talking about arithmetical universes we will often want to use n, m,... to stand for variables ranging only over the natural numbers. We do this by adopting the following convention: Any L-wff we will use, in which we have explicitly mentioned, say, the variable n as a free variable, is assumed to be preceeded by $nat(n)\supset$. Thus, for example, $\mathcal{J}\models(P(n)\supset Q)$ stands for $\mathcal{J}\models(nat(n)\supset(P(n)\supset Q))$, asserting that in state $\mathcal{J}$, $(P(n)\supset Q)$ is true if $n_\mathcal{J}$ happens to be a natural number. Furthermore, by convention, $\forall nP(n)$ stands for $\forall n(nat(n)\supset P(n))$, and hence $\exists nP(n)$ abbreviates $\exists n(nat(n)\wedge P(n))$.

For any $\alpha\in RG$, we let $var(\alpha)$ stand for the set of variables appearing in α.

Consider the following axiom system $\boldsymbol{P}$ for DL:

Axioms:

(A) All tautologies of propositional calculus.

(B) All A-valid L-wffs.

(C) $[x\leftarrow e]P \equiv P^e_x$, for an L-wff P.

(D) $[Q?]P \equiv (Q\supset P)$.

(E) $[\alpha;\beta]P \equiv [\alpha][\beta]P$.

(F) $[\alpha\cup\beta]P \equiv ([\alpha]P \wedge [\beta]P)$.

Inference rules:

(G) Modus Ponens $\dfrac{P \ , \ P \supset Q}{Q}$

(H) $\dfrac{P \supset Q}{[\alpha]P \supset [\alpha]Q}$

(I) Invariance $\dfrac{P \supset [\alpha]P}{P \supset [\alpha^*]P}$

(J) Convergence $\dfrac{P(n+1) \supset \langle\alpha\rangle P(n)}{P(n) \supset \langle\alpha^*\rangle P(0)}$ for an L–wff P with free n, s.t. $n \notin var(\alpha)$.

A DL–wff P is said to be *provable* in $\boldsymbol{P}$, written $\vdash_{\boldsymbol{P}} P$, if there exists a finite sequence S of DL–wffs, the last one being P, and such that each formula in S is an axiom (or instance of an axiom scheme), or is obtained from previous formulae of S by one of the rules of inference.

We now state four theorems which essentially appear in [12], and in more detail in [9].

Theorem 1 (A–expressiveness for DL): For any DL–wff P there exists an L–wff F_P such that $\models_A (P \equiv F_P)$.

Theorem 2 (A–soundness for DL): For any DL–wff P, if $\vdash_{\boldsymbol{P}} P$ then $\models_A P$.

Theorem 3 (Box–completeness): For every $\alpha \in RG$ and L–wffs R and Q, if $\models_A (R \supset [\alpha]Q)$ then $\vdash_{\boldsymbol{P}} (R \supset [\alpha]Q)$.

Theorem 4 (Diamond–completeness): For every $\alpha \in RG$ and L–wffs R and Q, if $\models_A (R \supset \langle\alpha\rangle Q)$ then $\vdash_{\boldsymbol{P}} (R \supset \langle\alpha\rangle Q)$.

Theorem 5 (A–completeness for DL): For any DL–wff P, if $\models_A P$ then $\vdash_{\boldsymbol{P}} P$.

Thus, for any arithmetical universe A, $\boldsymbol{P}$ characterizes the set of A–valid DL–wffs.

Turning to DL^+, we remark that one way of supplying an axiomatization of DL^+ can be derived from a recent theorem of Winklmann [24], who shows (contrary to our conjecture in [14]) that DL and DL^+ are *equal* in expressive power. His proof provides an algorithm for obtaining, for any α, a DL–wff Q_α such that $\models (Q_\alpha \equiv \langle\alpha\rangle^+ false)$. Thus, one could add a version of this algorithm to $\boldsymbol{P}$ as a rule for transforming $\langle\alpha\rangle^+ false$ into Q_α, and then the additional axiom,

$$\langle\alpha\rangle^+ P \equiv (\langle\alpha\rangle P \vee \langle\alpha\rangle^+ false)$$

would render this an arithmetically complete axiomatization of DL^+. However, Q_α is a rather

complicated formula and was designed in order to prove a "power of expression" result. The goals of axiomatization, and arithmetical axiomatization in particular, are different. Here we are interested in concise, elegant and well structured axiom systems for the purpose of proof. Consequently, we will concentrate on our original axiomatization , given in [14].

Augment $\boldsymbol{P}$ with the following, to obtain $\boldsymbol{P}^+$:

Axioms:

(K) $[x\leftarrow E]^+true$,
(L) $[Q?]^+true$,
(M) $[\alpha;\beta]^+true \equiv [\alpha]^+[\beta]^+true$,
(N) $[\alpha\cup\beta]^+true \equiv ([\alpha]^+true \wedge [\beta]^+true)$,
(O) $[\alpha]^+P \equiv ([\alpha]P \wedge [\alpha]^+true)$,

Inference rules:

(P) Finiteness $\dfrac{P(n+1)\supset[\alpha]^+P(n)\ ,\ \neg P(0)}{P(n)\supset[\alpha^*]^+true}$ for an L–wff P with free n, s.t. $n\notin var(\alpha)$,

(Q) Divergence $\dfrac{P\supset\langle\alpha\rangle^+P}{P\supset\langle\alpha^*\rangle^+false}$

Here too we have:

Theorem 6 (A–expressiveness for DL^+): For any DL^+–wff P there exists an L–wff F_P such that $\vdash_A(P\equiv F_P)$.

Theorem 7 (A–soundness for DL^+): For any DL^+–wff P, if $\vdash_{P^+} P$ then $\vdash_A P$.

Theorem 8 (Box^+–completeness): For every $\alpha\in RG$ and L–wffs R and Q, if $\vdash_A(R\supset[\alpha]^+Q)$ then $\vdash_{P^+}(R\supset[\alpha]^+Q)$.

Theorem 9 ($Diamond^+$–completeness): For every $\alpha\in RG$ and L–wffs R and Q, if $\vdash_A(R\supset\langle\alpha\rangle^+Q)$ then $\vdash_{P^+}(R\supset\langle\alpha\rangle^+Q)$.

Theorem 10 (A–completeness for DL^+): For any DL^+–wff P, if $\vdash_A P$ then $\vdash_{P^+} P$.

We now set ourselves out to find the thread connecting the previous results.

4. *The Theorem of Completeness.*

In this section we prove a general theorem which can be seen to provide the inductive step used in the proofs of all the specific completeness results of [12], [14] and [9], and in particular, of the present Theorems 5 and 10. Also, the proofs of arithmetical completeness results for any DL–like logics which fall under the general criterions of this theorem can be considerably shortened by appealing to it. Effort can then be devoted entirely to establishing the "base" of this induction, as illustrated in the next section.

As above, we denote the set of first-order formulae by L. Assume we are given a universe U, a set K, and a functional

$$M: \; Kx2^U \to 2^U.$$

The M-*extension of* L, L(M), is defined to be the following language which is L augmented with one formation-rule:

(1) Any atomic formula is in L(M),
(2) For any $k \in K$, variable x and L(M)-wffs P and Q,
$\neg P$, $(P \vee Q)$, $\exists x P$ and $(M_k)P$ are L(M)-wffs.

The semantics of L(M) are defined with $\mathcal{J} \models (M_k)P$ holding whenever $\mathcal{J} \in M(k, \{\mathcal{J} \mid \mathcal{J} \models P\})$, and with the other clauses receiving their standard meanings.

Some intuition might be gained at this point by noticing that if K is taken to be the class RC of regular programs over assignments and tests, and $(M_\alpha)P$ is interpreted as $\langle\alpha\rangle P$, then L(M) is in fact precisely DL.

We now define some important concepts to be used in the sequel:

We say that L is U-*expressive for* L(M) if for every L(M)-wff P there exists an L-wff Q such that $\models_U (P \equiv Q)$.

An *axiom system* $\boldsymbol{P}(M)$ for L(M) is any set of axioms (or axiom schemas) and inference rules over L(M). *Provability* of an L(M)-wff P in $\boldsymbol{P}(M)$ is defined in the standard way and is denoted by $\vdash_{\boldsymbol{P}(M)} P$. $\boldsymbol{P}(M)$ is said to be U-*sound* if all the axioms are U-valid and all the rules of inference preserve U-validity. Note then, that whenever $\vdash_{\boldsymbol{P}(M)} R$, we have $\models_U R$.

$\boldsymbol{P}(M)$ is said to be *propositionally complete* if all instances of tautologies of propositional calculus are theorems of $\boldsymbol{P}(M)$ and *modus ponens* is in the set of inference rules. It is said to be U-*complete* if for every L(M)-wff R, whenever $\models_U R$, we have $\vdash_{\boldsymbol{P}(M)} R$.

Theorem 11 (Theorem of Completeness): For any universe U, M-extension L(M) of L, a U-sound axiom system $\boldsymbol{P}(M)$ for L(M) is U-complete whenever the following hold:

(1) $\boldsymbol{P}(M)$ is propositionally complete,
(2) L is U-expressive for L(M),
(3) For any $k \in K$ and L(M)-wffs R and Q,
if $\vdash_{\boldsymbol{P}(M)} (R \supset Q)$ then $\vdash_{\boldsymbol{P}(M)} ((M_k)R \supset (M_k)Q)$,
(4) For any $k \in K$ and L-wffs R and Q,
a) if $\models_U R$ then $\vdash_{\boldsymbol{P}(M)} R$,
b) if $\models_U (R \supset (M_k)Q)$ then $\vdash_{\boldsymbol{P}(M)} (R \supset (M_k)Q)$, and
c) if $\models_U (R \supset \neg(M_k)Q)$ then $\vdash_{\boldsymbol{P}(M)} (R \supset \neg(M_k)Q)$.

Proof: We have to prove that if P is an L(M)-wff such that $\models_U P$, then $\vdash_{\boldsymbol{P}(M)} P$. By the propositional completeness of $\boldsymbol{P}(M)$ we can assume that P is given in conjunctive normal form, and we proceed by induction on the sum of the number of appearances of M and the number of quantifiers in P. Assume the theorem holds for any formula with n-1 or less appearances of M

and quantifiers. If P is of the form P1∧P2, then we have $\vdash_U$ P1 and $\vdash_U$ P2, so that we can restrict our attention to a single disjunction. Without loss of generality we can, therefore, assume that P is of one of the forms:

$$P1 \vee (M_k)P2, \quad P1 \vee \neg(M_k)P2, \quad P1 \vee \exists x P2 \quad \text{or} \quad P1 \vee \neg\exists x P2,$$

where $k \in K$ and P1 and P2 each have n−1 or less appearances of M and quantifiers. Let us use ρ to denote (M_k), $\neg(M_k)$, $\exists x$ or $\neg\exists x$ according to which is the case.

L is expressive for L(M), and so for any L(M)−wff Q there is some L−wff F_Q which is equivalent to Q. We have then $\vdash_U (\neg F_{P1} \supset \rho F_{P2})$. Now, using assumption (4) (since F_{P1} and F_{P2} are L−wffs), we also have

$$(^{+}) \qquad \vdash_{P(M)} (\neg F_{P1} \supset \rho F_{P2}).$$

Now surely, by the definition of F_{P1} and F_{P2}, we have $\vdash_U (\neg P1 \supset \neg F_{P1})$ and $\vdash_U (F_{P2} \supset P2)$. Both these last formulae have less than n appearances of M and quantifiers, and hence by the inductive hypothesis

$$(^{++}) \qquad \vdash_{P(M)} (\neg P1 \supset \neg F_{P1}) \quad \text{and}$$

$$\vdash_{P(M)} (F_{P2} \supset P2).$$

By assumption (3) or (4a) (depending on whether ρ is an appearance of M or a quantifier) together with the propositional completeness, we obtain from the latter

$$(^{+++}) \qquad \vdash_{P(M)} (\rho F_{P2} \supset \rho P2).$$

From $(^{+})$, $(^{++})$ and $(^{+++})$ we get, using propositional reasoning, $\vdash_{P(M)} (\neg P1 \supset \rho P2)$, or $\vdash_{P(M)} (P1 \vee \rho P2)$. ∎

Now, take A to be any arithmetical universe and K to be the set RC of regular programs over assignments and tests. Take M to be the "diamond" operator, or more precisely define $\mathcal{J} \models (M_\alpha)P$ iff $\mathcal{J} \models \langle\alpha\rangle P$. Certainly then, L(M) is simply DL. Furthermore, note that in this framework taking $P(M)$ to be the system P, we see that Theorem 2 establishes the U−soundness required in Theorem 11, axiom scheme (A) and rule (C) satisfy requirement (1) of the theorem, Theorem 1 satisfies requirement (2), rule (H) (or, more precisely, an easily derived version of it) satisfies requirement (3), and Theorems 3 and 4, together with axiom scheme (B), satisfy requirement (4). Consequently, Theorem 5 can be seen to be a corollary of Theorem 11, using Theorems 1−4.

Turning to DL^+, it is quite easy to see that Theorem 11 can be extended to deal with "double extensions" of L; i.e. for the case where there are two functionals M and M'. We omit the precise rephrasing of the theorem for this case, but note that taking $(M_\alpha)P$ to be $\langle\alpha\rangle P$ as above and $(M^+{}_\alpha)P$ to be $\langle\alpha\rangle^+ P$, the (M,M^+)−extension, $L(M,M^+)$, is simply DL^+. Here too, Theorem 10 can be seen to be a corollary of Theorem 11, using Theorems 6−9.

Theorem 11 captures the inductive step used in proving all the completeness results of [12], [14] and [9]. These include results similar to Theorems 5 and 10 for the extensions of DL

and DL^+ for dealing with recursive programs, CFDL and $CFDL^+$. In these cases the set K is taken to be the set CF of *context–free* programs over assignments and tests.

Note now that requirements (1), (3) and (4a) of Theorem 11 were satisfied in $\boldsymbol{P}$ and $\boldsymbol{P}^+$ by simply including, in a straightforward manner, axiom schemes (A), and (B) and rules (C) and (H). In the next section, however, we try to see how $\boldsymbol{P}$ and $\boldsymbol{P}^+$ were made to satisfy requirements (4b) and (4c).

5. One-program Completeness Results.

In this section we describe how $\boldsymbol{P}$ and $\boldsymbol{P}^+$ were designed so that the basic completeness results, i.e., Theorems 3,4,8 and 9, could be established in [12] and [14].

Let us first consider DL. Theorem 3 can be seen to be essentially Cook's [5] theorem concerning Hoare's [15] axiom system for proving the partial correctness of regular deterministic programs (see also [11], and Section 3.3 of [9]). Hoare's approach was to construct the system in such a way that the program α in the formula $R \supset [\alpha]Q$ (which he wrote as $R\{\alpha\}Q$) is "decomposed" one step at a time, based on the structure of α. The idea was to supply one rule of inference for every formation rule of the programming language, enabling one to conclude a property of a program by establishing similar properties of its *immediate components.* Completeness is then proved by induction on the structure of α, showing that at each step the appropriate rule could indeed be applied.

This appealing approach is imitated in our systems $\boldsymbol{P}$ and $\boldsymbol{P}^+$. One can show that axioms (C)–(F), with the addition of (K)–(O) for DL^+ together with the ability (made possible by propositional completeness) to reason about $\langle\alpha\rangle$ and $\langle\alpha\rangle^+$ using $[\alpha]$ and $[\alpha]^+$, serve to enable the carrying out of this decomposition for all the program constructs in RG but one: α^*.

Finding the appropriate axioms or rules for proving the properties of α^* is what we might now call the heart of the problem. Let us assume that we can find a sound rule of inference having (a) $R \supset [\alpha^*]Q$ as its conclusion, (b) programs of complexity "at most" α in its premises, and (c) the property that whenever $R \supset [\alpha^*]Q$ is A–valid, then one can establish the A–validity of its premises. Then the task of proving the A–validity of $R \supset [\alpha^*]Q$ can always be reduced to that of proving the "simpler" premises. This fact would then serve as the α^* part of the proof of Theorem 3 "in the spirit" of Hoare [15]. Similarly, such a rule for $R \supset \langle\alpha^*\rangle Q$ would establish Theorem 4, and these two would, as shown above, complete the proof of the general completeness result, Theorem 5. In the same way, having axiom (O) at hand, rules for $R \supset [\alpha^*]^+ \mathit{true}$ and $R \supset \langle\alpha^*\rangle^+ \mathit{false}$ can be seen to be sufficient for establishing Theorem 10.

We will show that these four tasks, resulting in rules (I), (J), (P) and (Q), are quite easy and are dual to one another in a rather interesting way.

We will work on $[\alpha^*]$ and $\langle\alpha^*\rangle$ simultaneously, and later exhibit the analogous process for $[\alpha^*]^+$ and $\langle\alpha^*\rangle^+$.

Let us take a look at the concept involved:

$[\alpha^*]Q$ $\qquad\qquad$ $\langle\alpha^*\rangle Q$.

We have the "arithmetical equivalent" of our concept:

$\forall n [\alpha^n]Q$ $\qquad\qquad$ $\exists n \langle\alpha^n\rangle Q$.

Note that for a given n and state $\mathcal{I} \in A$, we can prove

$$\mathcal{I} \models [\alpha^n]Q \qquad\qquad \mathcal{I} \models \langle\alpha^n\rangle Q,$$

by finding some formula P with a free variable n, $n \notin var(\alpha)$, such that the following hold:

$$\mathcal{I} \models P(n) \qquad\qquad \mathcal{I} \models P(n)$$
$$\models_A \forall m(P(m+1) \supset [\alpha]P(m)) \qquad\qquad \models_A \forall m(P(m+1) \supset \langle\alpha\rangle P(m))$$
$$\models_A (P(0) \supset Q) \qquad\qquad \models_A (P(0) \supset Q).$$

The reason is that the above establishes, by induction on n, that

$$\mathcal{I} \models [\alpha][\alpha]....[\alpha]Q \qquad\qquad \mathcal{I} \models \langle\alpha\rangle\langle\alpha\rangle....\langle\alpha\rangle Q,$$

with n occurencies of α, which, by

$$\models ([\alpha][\beta]Q \equiv [\alpha;\beta]Q) \qquad\qquad \models (\langle\alpha\rangle\langle\beta\rangle Q \equiv \langle\alpha;\beta\rangle Q),$$

gives

$$\mathcal{I} \models [\alpha^n]Q \qquad\qquad \mathcal{I} \models \langle\alpha^n\rangle Q.$$

Thus, if we can prove

$$\mathcal{I} \models (R \supset \forall n P(n)) \qquad\qquad \mathcal{I} \models (R \supset \exists n P(n))$$

instead of $\mathcal{I} \models P(n)$, we will have established

$$\mathcal{I} \models (R \supset \forall n [\alpha^n]Q) \qquad\qquad \mathcal{I} \models (R \supset \exists n \langle\alpha^n\rangle Q).$$

And so, being able to prove

$$\models_A (R \supset \forall n P(n)) \qquad\qquad \models_A (R \supset \exists n P(n))$$
$$\models_A (P(m+1) \supset [\alpha]P(m)) \qquad\qquad \models_A (P(m+1) \supset \langle\alpha\rangle P(m))$$
$$\models_A (P(0) \supset Q) \qquad\qquad \models_A (P(0) \supset Q)$$

will result in the establishment of

$$\models_A (R \supset [\alpha^*]Q) \qquad\qquad \models_A (R \supset \langle\alpha^*\rangle Q).$$

This gives rise to the A–sound rule of inference

$$(*)\ \frac{R \supset \forall n P \ ,\ P' \supset [\alpha]P \ ,\ P^0 \supset Q}{R \supset [\alpha^*]Q} \qquad \frac{R \supset \exists n P \ ,\ P' \supset \langle\alpha\rangle P \ ,\ P^0 \supset Q}{R \supset \langle\alpha^*\rangle Q}$$

where P' abbreviates P(n+1), and P^0 abbreviates P(0).

Now, in order for our required completeness property to hold, we must show that whenever

$$R \supset [\alpha^*]Q \qquad\qquad R \supset \langle\alpha^*\rangle Q$$

is A–valid, there exists a P(n) satisfying the premises of the rule. This, however, is true: take P(n) to be simply some first–order formula which for any n is A–equivalent to

$$[\alpha^n]Q \qquad\qquad \langle\alpha^n\rangle Q$$

itself. Such a formula exists as shown in [9]. In other words we have

$$\models_A \forall n (P(n) \equiv [\alpha^n]Q) \qquad\qquad \models_A \forall n (P(n) \equiv \langle\alpha^n\rangle Q).$$

Thus in fact, we have found sound and complete rules for α^* in DL, and we could have stopped here. However, we would like to go a step further. Using the observation that for the $[\alpha^*]$ case we have the duality principle $((R \supset [\beta]Q) \equiv (\langle\beta^-\rangle R \supset Q))$, where α^- is the *inverse* of α, defined such that $m(\alpha^-) = \{(\mathcal{I},\mathcal{J}) \mid \mathcal{J}\alpha\mathcal{I}\}$ (see [21]), we note that we can then write down an "ascending" induction rule, which is constructed symmetrically to the "descending rule" above:

$$\frac{R \supset P^0 \ , \ \langle \alpha^- \rangle P \supset P' \ , \ \exists n P \supset Q}{\exists n \langle (\alpha^-)^n \rangle R \supset Q}$$

which by the duality principle is really

$$(**) \quad \frac{R \supset P^0 \ , \ P \supset [\alpha] P' \ , \ \exists n P \supset Q}{R \supset [\alpha^*] Q}$$

and can be seen to be complete by taking P(n) to be $\langle (\alpha^-)^n \rangle R$. (The A–expressiveness of L is retained in the presence of the inverse operator $^-$.) Now, because of the universal character of $[\alpha^*]$ and the fact that $\forall n [\alpha^n] P \supset [\alpha] \forall n [\alpha^n] P$, or $[\alpha^*] P \supset [\alpha][\alpha^*] P$, we are able to *collapse* the descending and ascending rules (*) and (**) into *one* sound and complete rule by eliminating the n–indexing:

$$\frac{R \supset P \ , \ P \supset [\alpha] P \ , \ P \supset Q}{R \supset [\alpha^*] Q}$$

The premises of this rule are made A–valid when the conclusion is, by both $\forall n [\alpha^n] Q$ and $\exists n \langle (\alpha^-)^n \rangle R$, or essentially, by both $[\alpha^*] Q$ and $\langle (\alpha^-)^* \rangle R$. (Note that * replaces n in these satisfying formulae for (*) and (**).) Hence, we have arrived at the weakest box–antecedent and strongest box–consequent (see [12]), both of which can be used as the loop invariant P, stemming from the descending and ascending induction rules respectively.

Since there is no strongest diamond–consequent (see [12]), it turns out that there is, for the $\langle \alpha^* \rangle$ case, only the uninteresting ascending rule

$$\frac{R \supset P^0 \ , \ [\alpha^-] P \supset P' \ , \ \forall n P \supset Q}{\forall n [(\alpha^-)^n] R \supset Q}$$

the consequent of which is *not* A–equivalent to $R \supset \langle \alpha^* \rangle Q$.

We are left, then, with

$$\frac{R \supset P \ , \ P \supset [\alpha] P \ , \ P \supset Q}{R \supset [\alpha^*] Q} \qquad \frac{R \supset \exists n P \ , \ P' \supset \langle \alpha \rangle P \ , \ P^0 \supset Q}{R \supset \langle \alpha^* \rangle Q}$$

The final step is simply pruning the arms and legs of these rules in order to make them more concise, noting e.g. that from having proved $R \supset \exists n P$, $P^0 \supset Q$ and $\forall n (P \supset \langle \alpha^* \rangle P^0)$, we can deduce $R \supset \langle \alpha^* \rangle Q$ using validities of first order logic (included as axioms in (A)). We arrive, therefore, at our rules of invariance (I) and convergence (J)

$$\frac{P \supset [\alpha]P}{P \supset [\alpha^*]P} \qquad \frac{P' \supset \langle\alpha\rangle P}{P \supset \langle\alpha^*\rangle P^0}$$

and have in effect "developed" the invariant assertion method for partial correctness which is captured by rule (1) in parallel to the analogous method for its dual.

We would now like to present an *identical* chain of thought for DL^+, resulting in a suprisingly similar pair of rules. There we also see two concepts, one of universal and the other of existensial character, with the universal ($\langle\alpha^*\rangle^+ false$) giving rise to an index-free rule with strongest and weakest satisfying predicates. We will do this by summarizing the above discussion and giving the main points for $[\alpha^*]$ and $\langle\alpha^*\rangle$ again, together with the analogues for $[\alpha^*]^+$ and $\langle\alpha^*\rangle^+$. First, we restate here the results from [14]:

Theorem 12.

(1) $[\alpha^*]^+ true \equiv \exists n [\alpha^n] false \wedge [\alpha^*][\alpha]^+ true \equiv \exists n [\alpha^n]^+ false,$

(2) $\langle\alpha^*\rangle^+ false \equiv \forall n \langle\alpha^n\rangle true \vee \langle\alpha^*\rangle\langle\alpha\rangle^+ false \equiv \forall n \langle\alpha^n\rangle^+ true.$

The intuitive meaning of, say, (2) being that a divergence in α^* is due either to being able to run α for ever, or to being able to run α some number of times and then have α itself diverge.

The concept involved is

$[\alpha^*]Q$	$\langle\alpha^*\rangle Q$
$[\alpha^*]^+ true$	$\langle\alpha^*\rangle^+ false.$

The arithmetical equivalent is

$\forall n [\alpha^n] Q$	$\exists n \langle\alpha^n\rangle Q$
$\exists n [\alpha^n]^+ false$	$\forall n \langle\alpha^n\rangle^+ true.$

The descending rule is

$\dfrac{R \supset \forall n P \ , \ P' \supset [\alpha]P \ , \ P^0 \supset Q}{R \supset \forall n [\alpha^n] Q}$	$\dfrac{R \supset \exists n P \ , \ P' \supset \langle\alpha\rangle P \ , \ P^0 \supset Q}{R \supset \exists n \langle\alpha^n\rangle Q}$
$\dfrac{R \supset \exists n P \ , \ P' \supset [\alpha]^+ P \ , \ P^0 \supset false}{R \supset \exists n [\alpha^n]^+ false}$	$\dfrac{R \supset \forall n P \ , \ P' \supset \langle\alpha\rangle^+ P \ , \ P^0 \supset true}{R \supset \forall n \langle\alpha^n\rangle^+ true}$

The premises are satisfied (when the consequent is A-valid) by taking P to be A-equivalent to

$[\alpha^n]Q$	$\langle\alpha^n\rangle Q$
$[\alpha^n]^+ false$	$\langle\alpha^n\rangle^+ true.$

(We could have stopped here too; the above rules are sound and complete, and will enable the DL^+–completeness theorem to go through. We continue however, as we did above.)
The ascending rule is

$\dfrac{R \supset P^0 \ , \ P \supset [\alpha]P' \ , \ \exists n P \supset Q}{R \supset [\alpha^*]Q}$	*no rule*
no rule	$\dfrac{R \supset P^0 \ , \ P \supset \langle\alpha\rangle^+ P' \ , \ \exists n P \supset \mathit{true}}{R \supset \langle\alpha^*\rangle^+ \mathit{false}}$

The premises of this rule are satisfied by

$\langle(\alpha^-)^n\rangle R$	*no rule*
no rule	$\langle(\alpha^-)^n\rangle R \wedge \langle\alpha^*\rangle^+ \mathit{false}$.

The unified rule is

$\dfrac{R \supset P \ , \ P \supset [\alpha]P \ , \ P \supset Q}{R \supset [\alpha^*]Q}$	*no rule*
no rule	$\dfrac{R \supset P \ , \ P \supset \langle\alpha\rangle^+ P \ , \ P \supset \mathit{true}}{R \supset \langle\alpha^*\rangle^+ \mathit{false}}$

The premises of this rule are satisfied by both

$[\alpha^*]Q$ and $\langle(\alpha^-)^*\rangle R$	*no rule*
no rule	$\langle\alpha^*\rangle^+ \mathit{false}$ and $\langle(\alpha^-)^*\rangle R \wedge \langle\alpha^*\rangle^+ \mathit{false}$.

The final "pruned" rule is

$\dfrac{P \supset [\alpha]P}{P \supset [\alpha^*]P}$	$\dfrac{P' \supset \langle\alpha\rangle P}{P \supset \langle\alpha^*\rangle P^0}$
$\dfrac{P' \supset [\alpha]^+ P \ , \ \neg P^0}{P \supset [\alpha^*]^+ \mathit{true}}$	$\dfrac{P \supset \langle\alpha\rangle^+ P}{P \supset \langle\alpha^*\rangle^+ \mathit{false}}$

The name given to the construct used in a proof (i.e. the P involved) is

invariant	*convergent*
??	*divergent* (we suggest).

We would appreciate suggestions on names for the "??".

We would like the reader to consider the virtues of conducting this reasoning for the language of regular expressions over assignments and tests. Consider how much more obscure the observations of this section would have been if we were to reason about, say, the *while* statement, instead of about α^*. In our opinion α^* captures the raw essence of *iterating* in programming languages, just as $\alpha \cup \beta$ captures the essence of *branching* and $\alpha ; \beta$ the essence of *sequencing*. For the programming language designer who is interested in a deterministic language or in a more "disciplined" nondeterministic one, we can recommend means of restricting the generality of these constructs (e.g. *if-then-else-fi* and *while*, or Dijkstra's [6] *IF* and *DO*, etc.). Note how the *invariant assertion* method of Floyd [7], as described by Hoare's *while* rule [15] (see also Section 3.3 of [9]), has been shown to fall out of this general pattern of arithmetically complete rules as a special case.

A word about recursion. In [9] and [10], we have developed an arithmetically complete system for an augmented programming language which has $\mu X\tau(X)$ as an additional program construct (see, say, [2]). The basic idea there is to develop an analogy between $\mu X\tau(X)$ and $\tau^n(\mathit{false}?)$ on one hand, and α^* and α^n on the other. That is, α^* being $\cup_{n=0}^{\infty} \alpha^n$, enabled us to "count" how many times we *iterated* α, and to use this counting in the construction of our rules. Similarly, we are using the fact that (for continuous τ's) $\mu X\tau(X)$ is $\cup_{n=0}^{\infty} \tau^n(\mathit{false}?)$ to "count" how many times we *recured* with τ, and we have obtained similar results. Thus, it seems that working with α^* is showing its advantages in applying these ideas to a more complicated construct.

6. *An Application.*

We would now like to illustrate the usefulness of the step-by-step development of the α^* rules for DL and DL$^+$ in Section 5 by considering another extension of DL, namely ADL.

We add inductively, for any program α and ADL-wff P, the formula $(\cap\alpha)P$, defining $J \models (\cap\alpha)P$ iff for all n, $J \models \langle\alpha^n\rangle P$. Thus, we might write $\models_A((\cap\alpha)P \equiv \forall n\langle\alpha^n\rangle P)$. $(\cap\alpha)P$ is a construct used in the (deterministic) *algorithmic logic* of Salwicki [22]. Theorem 12 supplies an easy translation of DL$^+$ into ADL. In fact, Winklmann's [24] theorem shows that ADL, DL$^+$ and DL are all equivalent in expressive power. However, as was the case for DL$^+$, we are interested in an arithmetically complete axiom system for ADL, and therefore simply follow the lines of the previous section. First we must have

Theorem 13 (A-expressiveness for ADL): For any ADL-wff P, there exists an L-wff F_P, such that $\models_A(P \equiv F_P)$.

Proof. Easy inductive proof using the definition $(\cap\alpha)Q \equiv \forall n\langle\alpha^n\rangle Q$. ∎

Lemma 14. For any program α and L-wffs R and Q,

if $\models_A(R \supset Q)$ then $\models_A((\cap\alpha)R \supset (\cap\alpha)Q)$.

Proof. We omit this slightly tedious but nevertheless straightforward proof. ∎

Having Lemma 14 at hand, we add the following rule to $\mathcal{P}$:

$$(R)\qquad \frac{P \supset Q}{(\cap\alpha)P \supset (\cap\alpha)Q}$$

We also add the rules:

$$(S)\quad \frac{R \supset P(n)\ ,\ P(n+1) \supset \langle\alpha\rangle P(n)\ ,\ P(0) \supset Q}{R \supset (\cap\alpha)Q}$$

For an L–wff P with free n, s.t. $n \notin var(\alpha)$,

$$(T)\quad \frac{P(n+1) \supset [\alpha]P(n)}{P(n) \supset \neg(\cap\alpha)\neg P(0)}$$

For an L–wff P s.t. $n \notin var(\alpha)$,

(S) and (T) are obtained from the following rules, which in turn follow easily from considerations similar to those described in Section 5:

$$\frac{R \supset \forall n P\ ,\ P' \supset \langle\alpha\rangle P\ ,\ P^0 \supset Q}{R \supset \forall n \langle\alpha^n\rangle Q} \qquad \text{and} \qquad \frac{R \supset \exists n P\ ,\ P' \supset [\alpha]P\ ,\ P^0 \supset Q}{R \supset \exists n [\alpha^n] Q}$$

We do not know of a duality principle, or of any other way for doing away with the indices in rule (S). Denoting the resulting axiom system by $P(\cap)$, we have

Theorem 15 (A–soundness and completeness for ADL): For any ADL–wff P, $\models_A P$ iff $\vdash_{P(\cap)} P$.

Proof. Apply Theorem 11, proving assumptions (4b) and (4c) of that theorem using the above rules, by showing that their premises are satisfied respectively by $\langle\alpha^n\rangle Q$ and $[\alpha^n]Q$. ∎

7. *Relative Completeness.*

The approach to axiomatization taken in this paper is closely related to, and in fact is derived from, Cook's [5] notion of *relative completeness.* In this section we take up the task of comparing the two approaches.

As we indicated in the previous section, Hoare [15] introduced an axiom system for proving the partial correctness of deterministic programs. For the sake of this discussion we can in fact think of the subsystem of P consisting of axioms (A), (C)–(F) and rules (C) and (I) as Hoare's system, and can denote it by H (see Section 3.3 of [9]). Cook [5] investigated the question of completeness of Hoare's system and managed to formalize what seems to be the intuitive way in which people prove correctness (partial in this case) of programs in line with the method suggested by Floyd [7] and Naur [20]. Cook separated the reasoning about the program from the reasoning about the underlying language, making a distinction between proving say, $[x \leftarrow 1]x=1$, and proving $(x>0 \supset x \geq 0)$. The former still requires some program–oriented manipulation in order to turn it into a first–order formula, whereas the second does not. Thus, Cook's idea was to supply Hoare's system with a generous *oracle* which had the ability to answer questions concerning the truth of first order formulae. In this way he was able to shift concentration to Hoare's rules themselves, which were to serve as a tool for performing a step–by–step transformation of partial correctness assertions (of the form $P \supset [\alpha]Q$) into equivalent first–order formulae. The truth of the latter is then checked using the oracle.

The way we formalize this approach is, as we have done, to represent the oracle simply as the collection of the true first-order formulae that we are interested in, taken as further axioms. Thus we arrive at the following formal definition using the terminology we have developed: Assume we are given a language L' which includes all first-order formulae as wffs; thus L is part of L'. Assume AX is a sound axiom system for L', and denote by AX_U the system AX $\cup$ {P| P$\in$L and $\models_U$ P}. In other words, AX_U is AX augmented with all the U-valid first-order formulae as further axioms. AX is said to be *complete for* L' *relative to* L if for every universe U such that L is U-expressive for L', AX_U is U-complete for L' (every U-valid L'-wff is provable in AX_U).

Theorem 16 (Cook): Denote by L_H the language {R$\supset$[α]Q| R and Q are L-wffs}. Then, H is complete for L_H relative to L.

The proof is in fact identical to that of our Box-completeness theorem (Thm. 3).

Now, if we restrict ourselves to languages L' such that for any arithmetical universe A, L is A-expressive for L' (as is the case, for example, when L' is taken to be any one of DL, DL^+, CFDL, $CFDL^+$ etc.), we note that arithmetical completeness is a special case of relative completeness; we do not require that AX_U be U-complete for *all* universes U which make L U-expressive for L', but only that that be the case for any *arithmetical* universe. Consequently then, *in* AX itself we may use symbols in ways which take their standard interpretation for granted. This is the flavor of the usage of n, + and 0 in the rules (J) and (P), as well as in the rules for ($\cap\alpha$) in Section 6.

Let us look at some related work. Although coming earlier than [5] and mentioning neither expressiveness nor Hoare-like systems, Theorem 1 of [18], Theorem 2(1) of [3], and the part of [1] which considers regular programs, include the observations needed for obtaining relative completeness theorems for Floyd's inductive assertion method, and would carry over to suitable versions of Hoare's system.

Extensions of Cook's result to cover full mutually recursive procedures appear in [8] and in [13], and both utilize a method for "freezing" the input variables upon entry to a procedure. (These systems are subsumed by the axiomatization of context-free DL in [10] and [9].)

This flurry of "positive" research led inevitably to a counter-effort of "negative" research aimed at proving incompleteness results which indicate when Hoare-like systems are doomed to be incomplete even in the relative sense of Cook. The first notable result in this direction is that of Wand [23], who shows essentially that it is not the case that L is U-expressive for *every* universe U. Thus Wand shows that there exist universes U such that AX_U is *not* U-complete for L_H. More recently, Lipton [16] claims to have proved the following very interesting characterization of these "good" universes: L is U-expressive for L_H, *iff* U is an arithmetical universe or a universe with a finite domain (call the latter a *finite* universe). Thus according to this claim the only universes for which a Hoare-like system can be relatively complete are the arithmetical ones and the finite ones. So Cook's [5] requirement boils down to requiring that AX_U be U-complete for these two kinds of universes.

The finite universes, however, cause trouble: Clarke [4] has shown that introducing (into the programming language in which the programs of L_H are written) various programming

concepts such as procedures as parameters or coroutines, in the presence of recursion and other reasonable mechanisms, prevents the possibility of obtaining relatively complete axiom systems. The argument in [4] is based on the fact that the first order language L is U-expressive for L_H for any finite universe U. The incompleteness results are then established by showing that these complex programming languages have an undecidable halting problem over finite domains, and hence the set of diverging programs is not r.e., a fact which would contradict the existence of any relatively complete Hoare-like axiom system for such a language (the existence of one implying that, in particular, the set of valid formula of the form $true \supset [\alpha] false$ is r.e.). Hence, the essence of Clarke's results lies in the fact that *Cook's condition of expressiveness of* L *is satisfied by universes with finite domains.*

The research of Lipton and Snyder [17] and Lipton [16] culminates in a generalization and extension of Clarke's results, with a theorem (Theorem 1 in [16]) which seems to tie up as equivalent the two properties of a programming language: (1) having a decidable halting problem over finite universes, and (2) the set of formulae $P \supset [\alpha]Q$ over it being r.e. in the set of all U-valid L-wffs, for any U such that L is U-expressive for L_H.

We conclude that relaxing the requirement and requiring that AX_U be U-complete only for all *arithmetical* universes (i.e. playing our arithmetical-completeness game) seems a reasonable thing to do *even for the restricted language of partial correctness,* L_H.

In addition, it seems that in order for axiomatizations of much richer logics like, say, DL and DL^+ to be *relatively* complete (i.e that they work for finite universes too), the rules that involve arithmetic (i.e rules (J) and (P)) would have to be modified to deal with the finite-domain case, and would probably result in a system which is far less natural and elegant.

We are of the opinion, therefore, that the finite domains crept in because (1) the concept treated most extensively by researchers in the area was partial correctness ($[\alpha]P$ essentially), and (2) a weaker kind of expressiveness is needed to ensure the existence of an elegant relatively complete axiomatization of this particular concept on its own.

Thus we feel that it is natural and benificial to allow the integers into ones reasoning language, in order to make possible the kind of "counting" we carry out in $\boldsymbol{P}$ and $\boldsymbol{P}^+$.

Note that by adopting the "Hoare spirit" of structured, natural axiom systems, the remark in [23, pp. 90] "if the language is expressive it is trivial to write down a complete axiom system for partial correctness" becomes irrelevent. We are not interested in a one-rule system (as described in [23], or as shown to us in more detail by A.R. Meyer for the case of arithmetic) which has built into it essentially the full description of how to Godel-encode any wff and how to construct the equivalent formula of arithmetic. Rather, we want systems for composing our formulae step by step, using various kinds of assertions on the way. Of course the proof that these systems are *complete* might involve relying on the expressive power of arithmetic, and hence might call upon the use of Godel encoding, in turn making "the formulae .... be less than perspicuous" [23] (as is the case with our completeness results which at various points require finding the arithmetical equivalent to formulae). Nevertheless, we believe that the construction of these systems contributes considerably to the understanding of the concepts involved and provides the framework in which the natural and intuitive proofs one might have for one's programs can be formulated.

8. Acknowledgements.

The observations made in this paper would have been impossible without the research carried out jointly with V.R. Pratt and A.R. Meyer.

9. References.

[1] deBakker, J.W. and L.G.L.T. Meertens. On the Completeness of the Inductive Assertion Method. J. of Computer and System Sciences, 11, 323–357. 1975.

[2] deBakker, J.W. and W.P. deRoever. A Calculus for Recursive Program Schemes. in *Automata, Languages and Programming* (ed. Nivat), 167–196. North Holland. 1972.

[3] Banachowski, L. Modular Properties of Programs. Bull. Acad. Pol. Sci., Ser. Sci. Math. Astr. Phys. Vol. 23. No. 3. 1975.

[4] Clarke, E.M. Programming Language Constructs for which it is impossible to obtain good Hoare–like Axiom Systems. Proc. 4th ACM Symp. on Principles of Programming Languages. 10–20. Jan. 1977.

[5] Cook, S.A. Soundness and Completeness of an Axiom System for Program Verification, SIAM J. Comp. Vol. 7, no. 1. Feb. 1978. (A revision of: Axiomatic and Interpretive Semantics for an Algol Fragment, TR–79. Dept. of Comuter Science, U. of Toronto. 1975.)

[6] Dijkstra, E.W. Guarded Commands, Nondeterminacy and Formal Derivation of Programs. CACM Vol. 18, no. 8. 1975

[7] Floyd, R.W. Assigning Meaning to Programs. In J.T. Schwartz (ed.) *Mathematical Aspects of Computer Science.* Proc. Symp. in Applied Math. 19. Providence, R.I. American Math. Soc. 19–32. 1967.

[8] Gorelick, G.A. A Complete Axiomatic System for Proving Assertions about Recursive and Nonrecursive Programs. TR–75. Dept. of Computer Science, U. of Toronto. 1975.

[9] Harel, D. Logics of Programs: Axiomatics and Descriptive Power. Ph.D. Thesis. Dept. of EECS. MIT, Cambridge MA. June. 1978.

[10] Harel, D. Complete Axiomatization of Properties of Recursive Programs. Submitted for publication.

[11] Harel, D. On the Correctness of Regular Deterministic Programs; A Unified Survey. Submitted for publication.

[12] Harel, D., A.R. Meyer and V.R. Pratt. Computability and Completeness in Logics of Programs. Proc. 9th Ann. ACM Symp. on Theory of Computing, 261–268, Boulder, Col., May 1977.

[13] Harel, D., A. Pnueli and J. Stavi. Completeness Issues for Inductive Assertions and Hoare's Method. Technical Report, Dept of Appl. Math. Tel–Aviv U. Israel. Aug. 1976.

[14] Harel, D. and V.R. Pratt. Nondeterminism in Logics of Programs. Proc. 5th ACM Symp. on Principles of Programming Languages. Tucson, Ariz. Jan. 1978.

[15] Hoare, C.A.R. An Axiomatic Basis for Computer Programming. CACM 12, 576–580. 1969.

[16] Lipton, R.J. A Necessary and Sufficient Condition for the Existence of Hoare Logics. 18th IEEE Symposium on Foundations of Computer Science, Providence, R.I. Oct. 1977.

[17] Lipton, R.J. and L. Snyder. Completeness and Incompleteness of Hoare–like Axiom Systems. Manuscript. Dept. of Computer Science. Yale University. 1977.

[18] Manna, Z. The Correctness of Programs. JCSS 3. 119–127. 1969.

[19] Meyer, A.R. Equivalence of DL, DL$^+$ and ADL for Regular Programs with Array Assignments. Manuscript. Lab. for Computer Science. MIT, Cambridge MA. August 1977.

[20] Naur, P. Proof of Algorithms by General Snapshots. BIT 6. 310–316. 1966.

[21] Pratt, V.R. Semantical Considerations on Floyd–Hoare Logic. 17th IEEE Symposium on Foundations of Computer Science, 109–121, Oct. 1976.

[22] Salwicki, A. Formalized Algorithmic Languages. Bull. Acad. Pol. Sci., Ser. Sci. Math. Astr. Phys. Vol. 18. No. 5. 1970.

[23] Wand, M. A New Incompleteness Result for Hoare's System. Proc. 8th ACM Symp. on Theory of Computing, 87–91. Hershey, Penn. May 1976.

[24] Winklmann, K. Equivalence of DL and DL$^+$ for regular programs. Manuscript, Lab. for Computer Science. MIT, Cambridge, MA. March. 1978.

This report was prepared with the support of the National Science Foundation under NSF grant no. MCS76-18461.