9th STOC, 1977

Computability and Completeness in Logics of Programs
Preliminary Report

D. Harel, A. R. Meyer and V. R. Pratt
Massachusetts Institute of Technology
Cambridge, Mass. 02139

Abstract

Dynamic logic is a generalization of first order logic in which quantifiers of the form "for all X..." are replaced by phrases of the form "after executing program α...". This logic subsumes most existing first-order logics of programs that manipulate their environment, including Floyd's and Hoare's logics of partial correctness and Manna and Waldinger's logic of total correctness, yet is more closely related to classical first-order logic than any other proposed logic of programs. We consider two issues: how hard is the validity problem for the formulae of dynamic logic, and how might one axiomatize dynamic logic? We give bounds on the validity problem for some special cases, including a Π^0_2-completeness result for the partial correctness theories of uninterpreted flowchart programs. We also demonstrate the completeness of an axiomatization of dynamic logic relative to arithmetic.

Introduction

In this paper we continue the development of a proof theory and semantics of a formal logical language for making assertions about programs. This language, which was introduced in [13], will be called "dynamic logic" in this paper. It is based on the language of first order logic, with the quantifier "for all X ..." generalized to "after executing the program α ... ." Just as "for all X ..." is abbreviated to "$\forall X$...," so shall we abbreviate "after executing α ..." to "$[\alpha]$...". Just as $\exists X$ is the dual of $\forall X$, so is $\langle\alpha\rangle$ the dual of $[\alpha]$, and indeed $\langle\alpha\rangle P$ is equivalent to $\neg[\alpha]\neg P$. Informally, the truth of $[\alpha]P$ in a state (interpretation) $\mathfrak{I}$ amounts to the claim that, if α is started in state $\mathfrak{I}$, P will be true if and when α halts, no matter which state α halted in (α may be non-deterministic). Likewise, the truth of $\langle\alpha\rangle P$ in a state $\mathfrak{I}$ affirms that, starting from state $\mathfrak{I}$, some computation of α leads to a state in which P is true. A formal definition of $[\alpha]$ and $\langle\alpha\rangle$ is outlined below in terms of the notion of a modality in formal logic. Using this language, $\forall X$ could equivalently be expressed as $[X:=?]$ where $X:=?$ is a nondeterministic program which sets X to a random individual. This is the sense in which $[\alpha]$ generalizes ordinary logical quantification. (Our notation is motivated by the box and diamond notations of modal logic [10]; in conversation we have found it convenient to pronounce $[\alpha]$ as "box α" and $\langle\alpha\rangle$ as "diamond α.")

This research was supported by the National Science Foundation under contracts GJ43634X, DCR74-12997 A01 and MCS76-18461.

Many of the basic assertions one might like to make about programs are directly expressible as formulae involving $[\alpha]$ and $\langle\alpha\rangle$. For example, to assert that program α never halts on any input, one can write $[\alpha]$*false* (or $[\alpha]X\neq X$). Literally, this asserts that *false* holds in any final state reached by α, which of course is only possible if α never reaches such a final state. Dually, to assert that program α halts on all inputs, one need only write $\langle\alpha\rangle$*true*. Hoare's partial correctness assertion $P\{\alpha\}Q$ [8] may be expressed as $\models(P\supset[\alpha]Q)$, which suggests the analogous $\models(P\supset\langle\alpha\rangle Q)$ for a general termination assertion. Dijkstra's predicate transformer wp(α,P) [5] may be expressed as $[\alpha]P\wedge\langle\alpha\rangle P$ or equivalently as $[\alpha]P\wedge\langle\alpha\rangle$*true*. Two programs each using a single variable for output (respectively Y and Z) may be asserted to be equivalent with $\forall X((\langle\alpha\rangle Y{=}X)\equiv(\langle\beta\rangle Z{=}X))$, where X does not appear in α or β. This asserts that if X is a possible output of the program α, then X is also a possible output value of β, and conversely. We may also assert that α is deterministic by saying that the value it yields in some computation is the value it yields in every computation, using $\forall X(\langle\alpha\rangle Y{=}X\supset[\alpha]Y{=}X)$.

For convenience we refer to classical first-order logic as *first-order logic*. By *loop-free logic* we mean first-order logic augmented with modalities restricted to programs in the closure, under union and composition, of tests and assignments. By *regular logic* we mean loop-free logic together with transitive closure, denoted by *. By *dynamic logic* we permit $[\alpha]$ and $\langle\alpha\rangle$ modalities where α may be any r.e. program (defined below).

The first part of the paper deals with the problem of how hard it is to decide validity of formulae of dynamic logic. Validity is a recursively enumerable predicate on formulae of first-order logic. It was shown in [13] that validity is no harder for loop-free logic than for first-order logic, but harder for regular logic. In this paper we further widen this gap between loop-free and regular logic; in particular, we strengthen an incompleteness result obtained in [13] for the partial correctness theory of a trivial one-loop program with one instruction in its body. On the other hand, we show that at least for partial correctness assertions (formulae $P\supset[\alpha]Q$ for first-order P,Q), the validity problem is no harder for dynamic logic than it is for regular logic; thus if one considers enriching one's programming language with progressively more powerful control structures, the only change in the difficulty of the validity problem for partial correctness comes at the point where loops are introduced.

The results of the second part of the paper strike a more positive note by generalizing a relative completeness theorem of Cook [4]. We show that an economical set of Hoare-like axioms (first described in [13]), taken together with those formulae of logic that are valid in the "natural number universe," form a complete set of axioms for the formulae of regular logic valid in this universe.

Dynamic Logic

To keep this paper self-contained, we now give a brief account of dynamic logic as developed in [13]. The central concept here is the *symbol*. We envisage four kinds of symbols: *function symbols*, *predicate symbols*, symbols called *logical connectives*, and symbols called *modalities*, all of various arities ≥ 0 except that modalities are required to be unary. Modalities fall into two classes, "boxes" and "diamonds;" the distinction, discussed informally above, is explained formally in the paragraph below on expressions. We refer to a set of such symbols as a language; logicians may prefer the term "similarity type." Although we do not have any particular set of symbols in mind here, the reader will not go far astray if he assumes that the symbol set is fixed and consists of a countable supply of function and predicate symbols of each arity (including such standard symbols as +, x, >, and =), all standard unary and binary boolean logical connectives, and whatever modalities we permit explicitly in the sequel.

The four concepts dependent on the concept of symbol are state, universe, expression, and evaluation.

A *state* (interpretation, world, environment) of a language specifies a non-empty *domain* D (carrier, underlying set) and assigns a *value* to every symbol in the language except the modalities; it assigns as values k-ary functions on D to k-ary function symbols, k-ary predicates on D to k-ary predicate symbols, and k-ary boolean functions to k-ary logical connectives. All logical connectives receive their standard values, as does equality.

A *universe* (Kripke structure [10]) specifies a set U of states having a common domain D, subject to the foregoing constraint that = and all logical connectives have their standard values. A universe assigns to every modality in the language a value which is a binary relation on U. (Note that modalities differ from the other symbols in that their values are assigned per universe rather than per state.) A binary relation α on a universe U determines the net behavior of a (nondeterministic) program, i.e. a multi-valued function from start states to final states. For our purposes it is appropriate to regard programs as *being* such binary relations [1]. We consider only modalities whose value is "standard", in the sense that for each modality considered explicitly or implicitly in this paper and for each universe we have in mind a specific value for that modality in that universe. We shall write $[\alpha]$ (resp. $\langle\alpha\rangle$) to denote a box (resp. diamond) modality whose value in the universe is determined by α, where α is a syntactic object such as the deterministic assignment "X:=X+1", the non-deterministic assignment "X:=?", or the test "X>0?". Informally, the test P? in the universe U is the restriction of the identity relation on U to those states of U that satisfy P, while the assignment X:=T in universe U is the function (i.e. deterministic relation) that maps state $\mathfrak{s}$ of U to a state differing from $\mathfrak{s}$ only in the value it assigns to the zeroary function symbol X; this value is that of the term T in $\mathfrak{s}$. The equation (T) (resp. (A)) of [13] gives the precise rule for determining the binary relation from the test (resp. assignment) and the universe. While we do not explicitly consider array assignment in this paper, the results of [13] indicate that none of the results of this paper depend on whether array assignments are permitted.

An *expression* over a language L consists of an ordered pair whose first component (its *operator*) is a k-ary symbol of L and whose second component (its *operand*) is a k-tuple consisting of certain expressions (called the *arguments* of the expression). (This is equivalent to the definition given in [13] in terms of trees.) The simplest expressions have a 0-ary operator and the (unique) 0-tuple for an operand. Expressions are classified according to their operators as either *terms* (if the operator is a function symbol) or *formulae* (the rest). Formulae with predicate-symbol operators are called *atomic* while formulae with modal operators are called *modal*. The arguments of function and predicate symbols must be terms; arguments of modal symbols and logical connectives must be formulae. These remarks suffice to characterize the expressions of dynamic logic.

Like symbols, expressions are assigned values by states; the values of terms are individuals in D and the values of formulae are truth values. Expressions other than modal formulae are assigned values, or *evaluated*, by the standard Tarskian method [15]: the value in state $\mathfrak{s}$ of an expression is the result of applying the value in $\mathfrak{s}$ of the operator to the values in $\mathfrak{s}$ of the arguments. Given a universe U (which assigns a value to α) and a state $\mathfrak{s}$ in U, the value in $\mathfrak{s}$ of the modal formula $[\alpha]P$ is *true* when P is true in every $\mathfrak{f}$ of U satisfying $\mathfrak{s}\alpha\mathfrak{f}$, and *false* otherwise. The value of $\langle\alpha\rangle P$ in $\mathfrak{s}$ is *true* when P is true in some $\mathfrak{f}$ of U satisfying $\mathfrak{s}\alpha\mathfrak{f}$, and *false* otherwise.

Our previous remarks about first-order quantifiers can now be formalized as follows. Define a k-ary function (resp. predicate) symbol S to be *uninterpreted* in U if for every k-ary function (resp. predicate) V on D and for every state $\mathfrak{s}$ of U, there is a state $\mathfrak{s}'$ in U such that $\mathfrak{s}'$ is identical to $\mathfrak{s}$ except that $\mathfrak{s}'$ assigns the value V to S. Let X be a zeroary function symbol uninterpreted in U and let X:=? denote that binary relation on U that relates pairs of states differing at most in their value of X. (This is an equivalence relation. As we noted, this program may be regarded as the non-deterministic assignment statement that assigns an arbitrary element of the domain D to X; the ? can be taken in the spirit of APL's symbol for a random number.) Then it should be evident that we may take $\forall X$ to be the modality $[X:=?]$ and $\exists X$ to be $\langle X:=?\rangle$.

In [13] attention was focused on loop-free and regular programs. In the first part of this paper, on computability, we shall consider an even larger class of programs called r.e. programs. An *execution sequence* is a string over an alphabet whose elements denote tests and assignments of the kind considered in [13]. An execution sequence denotes the composition of the binary relations on states denoted by successive elements of the sequence. A set of execution sequences denotes the union of the denotations of the elements of the set. Then the above class of regular programs is just the class of programs denoted by regular sets (in the sense of automata theory) of execution sequences. The r.e. programs are precisely those denoted by recursively enumerable execution sequence sets. (Note that execution sequence sets are precisely the level (ii) programs of section 3.1 of [13].)

The only modalities we shall consider in this paper are first-order quantifiers and *program modalities* (defined by execution sequence sets, excluding the instruction X:=?).

We classify formulae of dynamic logic into four successively larger categories according to the kinds of program modalities which appear:

first-order	no program modalities
loop-free	loop-free program modalities permitted
regular	regular program modalities permitted
dynamic	r.e. program modalities permitted

We independently classify formulae according to the kinds of non-zeroary predicate and function symbols which appear:

arithmetic	only +,x,=
logic	any except +,x
augmented arithmetic	no restriction

A formula P is *valid in a universe* U, or *U-valid*, (notation: $\models_U P$) when it is true in every state of U. P is *valid* (notation: $\models P$) when it is valid in every non-empty universe in which function and predicate symbols (except =) are uninterpreted and modalities and logical connectives are interpreted as described in the above paragraph on universes. (For first-order formulae this usage coincides with the standard definition of validity.) The *natural number universe* $\mathcal{N}$ has as domain the natural numbers, and in every state assigns the standard values to + and x (and any other symbols the reader recognizes as standard symbols of arithmetic). P is *$\mathcal{N}$-valid* (notation: $\models_{\mathcal{N}} P$) when it is valid in the natural number universe.

We use the word *theory* to refer to valid formulae. Thus the valid formulae of dynamic logic constitute *dynamic theory*. The $\mathcal{N}$-valid formulae of first-order first-order augmented arithmetic will be called *augmented number theory*. The $\mathcal{N}$-valid formulae of first-order arithmetic will be called first-order number theory, etc. Deciding $\mathcal{N}$-validity of augmented arithmetic is known to be much harder than for first-order arithmetic because predicates not in the arithmetic hierarchy are implicitly definable in first-order augmented arithmetic. Similar remarks apply to dynamic augmented number theory versus dynamic number theory.

Two observations of [13] are relevant to this paper.

(1) The partial correctness theory of X:=FX* (i.e. the set of valid formulae $P \supset [X:=FX^*]Q$ where P,Q are formulae of first-order logic) is not r.e. (This is Theorem 16 of [13], and was proved in essence by showing that any set in the class Π_1^0 is reducible to this partial correctness theory. We henceforth use Σ_i and Π_i to denote classes of the arithmetic hierarchy, omitting the usual superscript 0 denoting first-order logic, cf [14])

(2) The axiom system of section 3.2 [13] for loop-free theory is sound, complete and effective.

In this paper we improve the Π_1 reduction in (1) to Π_2, still for the partial correctness theory of the same simple program. This implies the incompleteness of any axiom system in which theoremhood is not at least as hard to decide as membership in Π_2-complete sets. However, we also show that the partial correctness theory of all r.e. programs (the set of valid formulae $P \supset [\alpha]Q$ where α may be any r.e. program) is in Π_2, so no r.e. program has a more intractable partial correctness theory than X:=FX*.

It follows from (1) that a sound, complete, effective axiom system for the valid formulae of dynamic (or even regular) logic is impossible. However, as we show in this paper, by taking all of first-order number theory as axioms, the extension of the axiom system to handle loops (also given in [13]) is sound and complete for regular number theory. The result also holds for regular augmented number theory when we take first-order augmented number theory as axioms. Along with [6], this is the first time a completeness result has been obtained for systems that treat termination, let alone for one with the generality of dynamic logic.

Computability

We can abbreviate the four theorems of this section as follows. The notation should be self-explanatory when read in conjunction with the following expanded statements of the theorems.

(1) $\{\models \langle\alpha\rangle P\} \equiv \Sigma_1$	($\{\alpha\} \leqslant$ r.e.)
(2) $\{\models [\alpha]U\} \equiv \Pi_1$	(regular $\leqslant \{\alpha\} \leqslant$ r.e.)
(3) $\{\models [\alpha]P\} \equiv \Pi_2$	(regular $\leqslant \{\alpha\} \leqslant$ r.e.)
(4) $\{\models \exists Z[\alpha]P\} \geqslant \varphi^{(\omega)}$	($\{\alpha\} \geqslant$ X:=FX*)

Theorem 1. The valid formulae of dynamic logic of the form $\langle\alpha\rangle P$, where α is any r.e. program and P is a formula of first-order logic, form a complete r.e. set.

Proof. Note that just the valid formulae of first-order logic already form a complete r.e. set, that is, $\{\models P\} \equiv \Sigma_1$. So to prove the Σ_1-completeness of the set of valid formulae of the more general form $\langle\alpha\rangle P$, we need only prove that this set of valid formulae is r.e.

The validity of $\langle\alpha\rangle P$ amounts to the validity of an infinite disjunction of formulae $\langle\beta\rangle P$ where the β's are the denotations of the individual execution sequences of α. Each of these formulae may be expanded by Theorems 3 and 4 of [13] as formulae of first-order logic. Then the infinite disjunction is valid if and only if some finite subset of the disjunction is valid, by compactness of first-order logic. Since the disjunction is of an r.e. set of formulae, validity can be decided by enumerating elements of the disjunction until sufficiently many elements are present that their disjunction is valid. ∎

Theorem 2. The valid formulae of dynamic logic of the form $[\alpha]U$, where U is any universally quantified formula of first-order logic, form a complete co-r.e. set.

Proof. The validity of $[\alpha]U$ amounts to the validity of an infinite conjunction of formulae $[\beta]U$, which as in Theorem 1 may be expanded as universally quantified formulae of first-order logic. Then to check their validity it suffices to check the validity of each of the conjuncts, a decidable question since the conjuncts are universally quantified. The set of conjuncts being r.e., this problem is in Π_1 and so the valid formulae form a co-r.e. set.

To see that the set is complete in Π_1, it suffices to choose U to be *false*, and allow α to range merely over regular programs. $[\alpha]$*false* is valid iff the uninterpreted flowchart scheme corresponding to α never halts. This problem for flowchart schemes is known to be Π_1-complete [11]. ∎

Theorem 3. The valid formulae of dynamic logic of the form $[\alpha]P$, where α is any r.e. program and P is a formula of first-order logic, form a Π_2-complete set. This result holds even if the class of programs permitted is taken to be as small as regular programs; in fact, just the single regular program $X:=Y;X:=FX^*$ will suffice to obtain the result.

Proof. (Sketch). The upper bound is proved exactly as for Theorem 2, with the remark that the validity of each conjunct is now only partially decidable since each conjunct is an arbitrary formula of first-order logic including existential quantifiers. This boosts the problem from Π_1 to Π_2.

For the lower bound, our strategy will be to reduce the totality problem for Turing machines whose inputs are given in unary notation to the validity problem for sentences $[\alpha](P \supset Q)$ where α is the fixed program $X:=Y;X:=FX^*$. P will force models of the sentence to represent a computation of the Turing machine, while Q will assert that the machine halts.

Clearly $[\alpha]$ amounts to a universal quantifier "for all X in the set S = {Y,FY,F(FY),...}," which if thought of as natural numbers has Y for 0 and F for successor. We then supply a formula $W = \forall Z(G(FZ)=Z \wedge FZ \neq Y)$; it should be evident that in every model of W, S is infinite. This enables us to reason about Q in $[\alpha](W \supset Q)$ as though S were always infinite. Now consider the following more or less standard approach to showing that the validity problem for first-order logic is complete in Σ_1. Let R be a binary predicate symbol. Confine attention to the values of R on SxS, which we may think of as the upper quadrant of the two dimensional integer lattice. Take the rows of the lattice to be Turing machine i.d.'s (instantaneous descriptions) coded in binary in some way. (For definiteness, take R(i,j) to assert that cell (i,j) contains a 1.) It is tedious but straightforward to give a formula of first-order logic which forces adjacent rows of the lattice to describe i.d.'s the second of which is the result of running a given Turing machine for one step on the first. We can also say that a halting state appears on some row. Similarly we can give a formula that says that the first row contains just one state symbol (representing the Turing machine's head) positioned at the beginning of the tape (perhaps occupying the first hundred or thousand bits of the tape). And we can say that everywhere outside that head description, consecutive pairs of bits in the first row have only the configurations 11, 00 and 10, and furthermore that 10 occurs exactly once, namely at position X. If we could force X to occur within the S portion of the row, we would then have forced the row (outside the state descriptor) to encode a unary number with an initial finite segment of 1's and then the rest 0's. This is where $[\alpha]$ is used; we simply say $[\alpha](R(Y,X) \wedge \neg R(Y,FX))$. (To avoid "overwriting" the state, we really need to replace X by F(F(F...FX)) in R(Y,X) and R(Y,FX) for sufficiently many F's.) Let Q denote the statement that a halting state occurs on some row, and let P denote all the other statements we discussed (which will be a function of which Turing machine we had in mind). The $[\alpha]$ can be moved up (since X does not appear outside the original scope of $[\alpha]$) to yield the final formula $[\alpha](P \supset Q)$. Then we claim that $[\alpha](P \supset Q)$ is valid if and only if the Turing machine we had in mind halts on all inputs. If it is valid then it is true in the model in which S exhausts the domain and R represents a computation of that machine, which implies that the machine always halts. Now suppose that the machine always halts. Then there always exists a final state in every model of P for which X is in S, whence $[\alpha](P \supset Q)$ is always true, i.e. valid. ∎

We regard Theorem 3 as important, since it indicates the extent of the difficulty of supplying complete axiomatizations for the true partial correctness assertions of the form $P\{\alpha\}Q$. (The Π_2 upper bound generalizes to arbitrary P by the remark that $P \supset [\alpha]Q$ is equivalent to $[P?;\alpha]Q$.)

Theorem 4. The set of valid formulae of dynamic logic of the form $\exists Z[\alpha]P$, where α may simply be the fixed program $X:=Y;X:=FX^*$ and P is any formula of first-order logic, lies outside the arithmetic hierarchy [14].

(This sudden jump in the complexity of validity is attributable to being able to implicitly define [14] arithmetic truth given enough quantifiers.)

Proof. (Sketch). Our strategy in this proof is to reduce to the given validity problem the problem of deciding whether a natural number n is the Godel number of an N-valid, i.e. true, sentence of first-order arithmetic. We use a technique like that in the previous proof to construct from n a formula $\exists Z[\alpha](P \supset Q)$, to be valid if and only if n encodes a true sentence. P will constrain the models of the formula so that Q, which will simply be T(F(F...FY)) where there are n F's and T is a predicate symbol implicitly defined by P to test arithmetic truth, will be true if and only if n encodes a true sentence (cf [14], p344).

Given binary function symbols A and M, we can in first-order logic force A and M to behave like addition and multiplication on S (where S is as in the previous proof). Given a unary predicate symbol T, we can proceed to define T to act correctly on S, using auxiliary function and predicate symbols where necessary (mainly in primitive recursive definitions of functions such as SUB(X, 'VJ(E)') which given X and the Godel number of VJ(E) yields the Godel number of the result of substituting a constant equal to the value of X for J in E). In these definitions, almost all quantifiers may be ordinary first-order quantifiers. Only one quantifier requires care, and that is the quantifier that must appear in the definition of T(n) to cope with the case when n equals 'VJ(E)'. The definition of T at this point will look something like $T(\text{'}\forall J(E)\text{'}) \equiv \forall K(T(SUB(K, \text{'}\forall J(E)\text{'})))$, which ought to assert that T applied to the Godel number of VJ(E) should be true just when substituting any integer constant for J in E results in a true formula. Of course the VK quantifies over the whole domain, including elements on which all our defined functions may break down. This is where the $[\alpha]$ modality from the previous proof comes to the rescue, allowing us to quantify over just those elements on which we know our defined functions and predicates give the intended results. Thus the right hand side of the above equivalence could be phrased $[\alpha](T(SUB(X, \text{'}\forall J(E)\text{'})))$. This would complete the proof (modulo many details) except for the fact that $[\alpha]$ is more deeply embedded in our final formula than we would like. The technique for moving it out is to revert to the use of ordinary quantification (VK) for simulating "VJ", but at the "top level" a statement is made that if T(SUB(X,Z)) holds for all X in S, then it holds for all K. Now let us account for the quantifiers. Let R denote the definitions of F, A, M, T, etc, and let Q denote T(F(F...FY)) with n F's. Then the final formula is

$$(\forall Z([\alpha](T(SUB(X,Z))) \supset \forall K(T(SUB(K,Z)))) \wedge R) \supset Q.$$

This is logically equivalent to $\exists Z[\alpha](P \supset Q)$ where P,Q are first-order formulae. ∎

Note that $\langle Z:=Y;Z:=FZ^*\rangle$ can be used in place of $\exists Z$ in the above proof, so that we may also conclude

$$\{\models \langle\beta\rangle[\alpha]P\} \not\geq \varphi^{(\omega)} \text{ (regular } \not\leq \{\alpha\},\{\beta\}).$$

Completeness

In this section we prove that an axiomatization of regular number theory (that is, an axiom system whose theorems are among the N-valid formulae of dynamic logic with no non-zeroary function or predicate symbols save +, x, and =, and restricted to modalities with regular programs) that was given in [13] can be made complete simply by taking the formulae of number theory as further axioms. The same proof shows that the same axiom system completely axiomatizes regular augmented number theory (permitting other function and predicate symbols besides +, x and =) provided the formulae of augmented number theory are taken as axioms.

Cook [4] has used the notion of expressiveness to prove the completeness of a good approximation to Hoare's axiom system, and not surprisingly our proof does so too. We say that a language L is as *U-expressive* as a language M when for every formula P of M there exists a formula Q of L such that $\models_U (P \equiv Q)$. We argue briefly here that (augmented) number theory is as N-expressive as regular (augmented) number theory. (We could of course replace "regular" by "r.e.", or even more, but since we only exhibit axioms for regular programs there is little point in our so doing.) For the purposes of this section, we take α^N to be the program that maps state ϑ to the states that $\alpha\circ\alpha\circ\ldots\circ\alpha$ would map ϑ to, where the number of α's is given by the value of N in ϑ and N is not changed by α, nor does α depend on N. The main point is that from Cook's expressiveness proof we can infer that if P is N-expressible in (augmented) arithmetic and α is a regular program then $[\alpha^N]P$ is N-expressible in (augmented) arithmetic, say as Q. Hence $\forall NQ$ N-expresses $[\alpha^*]P$ provided N does not occur free in P. Further, if Q' N-expresses $\langle\alpha^N\rangle P$ then $\exists NQ'$ N-expresses $\langle\alpha^*\rangle P$.

We reproduce here the axiom system that appears in section 3.2 of [13] and refer to it henceforth as P. It is of interest inasmuch as it is the appropriate generalization of conventional axiom systems for pure first-order logic.

Logical Axioms

All tautologies of Propositional Calculus.

$[\alpha](P \supset Q) \supset ([\alpha]P \supset [\alpha]Q)$.

Logical Inference Rules

$P, P \supset Q \vdash Q$.

$P \vdash [\alpha]P$ (subsumes $P \vdash \forall xP$).

Non-logical Axioms

$\forall XP \supset P^t_X$ any formula t	∀ Performance Axiom.
$P \supset \forall XP$ (P has no free occ. of X)	∀ Invariance Axiom.
$[P]Q \equiv P \supset Q$	Test Axiom.
$[F(\underline{S}):=T]P \equiv P'$ (See [13] for details)	Assignment Axiom.
$[\alpha\cup\beta]P \equiv [\alpha]P \wedge [\beta]P$	Union Axiom.
$[\alpha\circ\beta]P \equiv [\alpha][\beta]P$	Composition Axiom.
$P \supset [\alpha]P \vdash P \supset [\alpha^*]P$	Rule of Invariance.
$P^{N+1}_N \supset \langle\alpha\rangle P \vdash P \supset \langle\alpha^*\rangle P^0_N$	Rule of Convergence.

A detailed discussion of these axioms appears in [13]. Here it suffices to observe that the first six axioms and rules, down to $P \supset \forall XP$, constitute a complete axiom system for classical pure predicate calculus. Together with the next four equivalences, they constitute a complete axiomatization of loop-free theory. Our objective now is to show that the whole system above, together with all the N-valid formulae of arithmetic as axioms, is a complete axiom system for regular number theory. The same proof will serve to show that when the N-valid formulae of augmented arithmetic are taken as axioms, the axiom system is complete for regular augmented number theory. We will not further consider the augmented case; however we note here that the augmented case falls much higher in the hierarchy of degrees of unsolvability and so it is interesting that the same proof applies.

The main result of this section is proved by a variant of the Star Interpolation Theorem (Theorem 24 of [13]). That theorem stated in essence that $[\alpha^*]P$ and $\langle\alpha^{*-}\rangle P$ (where for a program β, β^- is the converse relation: $\vartheta\beta\vartheta \equiv \vartheta\beta^-\vartheta$) were both invariants of α, which is obvious when one writes $[\alpha^*]P \supset [\alpha][\alpha^*]P$, and (not quite so obviously)

$$\begin{aligned}\langle\alpha^{*-}\rangle P &\supset [\alpha]\langle\alpha^-\rangle\langle\alpha^{*-}\rangle P \\ &\supset [\alpha]\langle\alpha^{*-}\rangle P\end{aligned}$$

We prove another Star Interpolation Theorem in this paper which interpolates, not invariants, but rather what we call convergents, which are to termination (and in the case of deterministic programs, to total correctness) what invariants are to partial correctness.

In the following we redefine some concepts from [13] in such a way as to make clear the relationship between Cook's completeness result for partial correctness alone and our completeness result for regular number theory, of which partial completeness and termination assertions are very special cases.

Duality Lemma $P \wedge \langle\alpha\rangle Q$ and $\langle\alpha^-\rangle P \wedge Q$ are equally satisfiable.

Proof. $\exists\vartheta\vartheta(\vartheta \vDash P \wedge \vartheta\alpha\vartheta \wedge \vartheta \vDash Q)$ asserts the satisfiability of each of the two formulae. ∎

Duality Principle. $\vDash(P \vee [\alpha]Q) \equiv \vDash([\alpha^-]P \vee Q)$.

Proof. Take the Boolean dual of satisfiability, $\wedge$, and $\langle\alpha\rangle$ in the duality lemma. ∎

Corollary. $\vDash(P \supset [\alpha]Q) \equiv \vDash(\langle\alpha^-\rangle P \supset Q)$.

Corollary. $\vDash(P \supset [\alpha]\langle\alpha^-\rangle P)$ (as remarked earlier).

Note that $\vDash(P \supset [\alpha]Q)$ expresses the same thing as Hoare's $P\{\alpha\}Q$, whence so does $\vDash(\langle\alpha^-\rangle P \supset Q)$. Whenever $P\{\alpha\}Q$ holds, we may call P a *box antecedent* of Q via α, and Q a *box consequent* of P via α. Since $\vDash([\alpha]Q \supset [\alpha]Q)$ and $\vDash(\langle\alpha^-\rangle P \supset \langle\alpha^-\rangle P)$, it follows that $[\alpha]Q$ must be the weakest box antecedent of Q via α (since for any antecedent P , $P \supset [\alpha]Q$ is valid) and similarly $\langle\alpha^-\rangle P$ must be the strongest box consequent of P via α.

Analogous to the partial correctness assertion $P \supset [\alpha]Q$ is the formula $P \supset \langle\alpha\rangle Q$, which asserts that if P holds α can terminate and satisfy Q (if α is deterministic, i.e. is a function, this asserts the total correctness of α). We can call $\langle\alpha\rangle Q$ a *weakest diamond antecedent* of Q via α, which leads us to ask for a strongest diamond consequent. Unfortunately the Duality Principle does not hold for $\langle\alpha\rangle$ in place of $[\alpha]$, as can be checked with $P = \neg Q = true$, $\alpha = \varphi$ (the empty program), for which $\vDash(true \vee \langle\varphi\rangle false)$ holds but $\vDash(\langle\varphi^-\rangle true \vee false)$ does not. That is, $[\alpha^-]P$ is not even a diamond consequent of P via α (since the above is a counterexample to $P \supset \langle\alpha\rangle[\alpha^-]P$), let alone a strongest diamond consequent. The conclusion is that termination is not exactly the dual of partial correctness: weakest diamond antecedents are given by $\langle\alpha\rangle Q$ but strongest diamond consequents are not given by $[\alpha^-]P$.

We define an *invariant* of α to be any formula P such that $P \supset [\alpha]P$ is N-valid.

Weakest Invariant Lemma. $[\alpha^*]P$ (the weakest box antecedent of P via α^*) is the weakest invariant of α that implies P.

Proof. Since $\alpha \circ \alpha^* \subseteq \alpha^*$, $[\alpha^*]P \supset [\alpha][\alpha^*]P$ is valid, so $[\alpha^*]P$ is an invariant of α. Further, since $I \subseteq \alpha^*$, $[\alpha^*]P \supset P$ (I is the identity relation) so it implies P. Finally, suppose $Q \supset [\alpha]Q$ and $Q \supset P$. Then $Q \supset [\alpha^*]Q \supset [\alpha^*]P$, so $[\alpha^*]P$ is the weakest such. ∎

Strongest Invariant Lemma. $\langle\alpha^{*-}\rangle P$ (the strongest box consequent of P via α^*) is the strongest invariant of α implied by P.

Proof. (Dual of the previous proof.) Since $\alpha^- \circ \alpha^{*-} \subseteq \alpha^{*-}$, $\langle\alpha^-\rangle\langle\alpha^{*-}\rangle P \supset \langle\alpha^{*-}\rangle P$ is valid, so $\langle\alpha^{*-}\rangle P \supset [\alpha]\langle\alpha^{*-}\rangle P$ by the Duality Principle, so $\langle\alpha^{*-}\rangle P$ is an invariant of α . Further, $P \supset \langle\alpha^{*-}\rangle P$. Finally, if $Q \supset [\alpha]Q$ and $P \supset Q$ then $Q \supset [\alpha^*]Q$, so $\langle\alpha^{*-}\rangle Q \supset Q$, so $\langle\alpha^{*-}\rangle P \supset \langle\alpha^{*-}\rangle Q \supset Q$. ∎

Note that weakest and strongest invariants are constructed analogously to weakest box antecedents and strongest box consequents, the only difference being the use of * for invariants.

We say that an invariant Q of α is an *invariant interpolate* of two formulae P and R via α, when $P \supset Q \supset R$ is valid.

Invariant Interpolation Lemma. If $P \supset [\alpha^*]R$ then $[\alpha^*]R$ and $\langle\alpha^{*-}\rangle P$ are both invariant interpolates of P and R via α.

Proof. Both are invariants of α by the above lemmas. Further, $P \supset [\alpha^*]R$ (hypothesis), $[\alpha^*]R \supset R$ ($I \subseteq \alpha^*$), $P \supset \langle\alpha^{*-}\rangle P$ ($I \subseteq \alpha^*$) and $\langle\alpha^{*-}\rangle P \supset R$ (dual of hypothesis). ∎

Just as the diamond antecedent was the analogue of the box antecedent, so do we have an analogue of the notion of invariant. We call Q a *convergent* of α when $Q' \supset \langle\alpha\rangle Q$ is valid, (where Q' is Q_N^{N+1}, which substitutes N+1 for all free occurrences of N in Q) and say that the convergent Q of α is a *convergent interpolate* of P and R when $P \supset \exists N(Q)$ and $Q_N^0 \supset R$. The interest in convergents is that they allow us to prove that a loop can eventually terminate with the right answer, just as invariants allow us to prove that a terminating loop always yields the right answer. In the case of deterministic programs, convergents subsume invariants, since for deterministic α, $\langle\alpha\rangle P \supset [\alpha]P$. Note that convergents and convergent interpolates are defined differently from invariants and invariant interpolates to permit the following lemmas, though when Q has no free occurrences of N these differences vanish except for the use of $\langle\rangle$ for $[]$.

Convergent Lemma. $\langle\alpha^N\rangle P$ is a convergent of α.

Proof. $\langle\alpha^{N+1}\rangle P \supset \langle\alpha\rangle\langle\alpha^N\rangle P$. ∎

Convergent Interpolation Lemma. If $P \supset \langle\alpha^*\rangle R$ then $\langle\alpha^N\rangle R$ is a convergent interpolate of P and R.

Proof. $P \supset \exists N \langle\alpha^N\rangle R$ and $\langle\alpha^0\rangle R \supset R$. ∎

We now prove that the axiom system P for regular number theory given at the beginning of this section is sound and complete when number theory is taken as additional axioms. We leave to the reader the task of showing that P is sound. The following three theorems deal with the completeness of P. They depend on our notion of expressiveness discussed at the beginning of this section.

Write P without the Rule of Invariance as $P\langle\rangle$, and without the Rule of Convergence as $P[]$.

Box Completeness Theorem. For any first-order formulae P and R, and for any α, $\models_N P \supset [\alpha]R$ iff $\vdash_{P[]} P \supset [\alpha]R$.

Proof. The result follows by induction on the number of *'s in α together with the fact that for $\alpha = \beta^*$, $[\beta^*]R$ is an invariant interpolate of P and R via β (by the Invariant Interpolation Lemma), which implies that the Rule of Invariance can be applied. Also in this case $P \supset [\beta^*]R$ and $[\beta^*]R \supset R$ are formulae of arithmetic, by the expressiveness of arithmetic, and therefore if valid are axioms of P. ∎

Diamond Completeness Theorem. For any first-order formulae P and R, and for any α, $\models_N P \supset \langle\alpha\rangle R$ iff $\vdash_{P\langle\rangle} P \supset \langle\alpha\rangle R$.

Proof. Again induction may be used on the number of *'s in α together with the fact that for $\alpha = \beta^*$, $\langle\beta^N\rangle R$ is a convergent interpolate of P and R via β (by the Convergent Interpolation Lemma), which implies that the Rule of Convergence can be applied. Also, in this case $P \supset \exists N \langle\beta^N\rangle R$ and $\langle\beta^0\rangle R \supset R$ are formulae of arithmetic, by the expressiveness of arithmetic, and therefore if valid are axioms of P. ∎

Main Completeness Theorem. For any formula P of regular number theory, $\vdash_P P$.

Proof. Again we appeal to the expressiveness of arithmetic, this time with respect to formulae of dynamic logic by a trivial argument on the depth of nesting of modalities. This implies that for any formula P there exists a formula L(P) of arithmetic such that $\models_N P \equiv L(P)$. We say that P is in conjunctive normal form when the argument of each $\neg$ is an atomic formula and the arguments of each $\vee$ are not conjuncts. Appealing to the evident completeness of our system for Propositional Calculus, we may assume that P is given in conjunctive normal form with n modalities, such that $\models_N P$. We proceed by induction on the number n of modalities in P. The case n = 1 can easily be seen to follow from the previous two theorems. Now assume the theorem holds for any formula with n-1 or less modalities. Observing that if $\models_N P_1 \wedge P_2$ then $\models_N P_1$ and $\models_N P_2$, we can restrict our discussion to a single disjunction. Without loss of generality we can assume P to be of the form $P_1 \vee m(\alpha)P_2$ where $m(\alpha)$ is $[\alpha]$ or $\langle\alpha\rangle$. We have $\models_N P_1 \vee m(\alpha)P_2$ and therefore $\models_N \neg L(P_1) \supset m(\alpha)L(P_2)$. Applying the appropriate of the two previous theorems we obtain $\vdash_P \neg L(P_1) \supset m(\alpha)L(P_2)$. Obviously by the definition of L(P) we have $\models_N \neg P_1 \supset \neg L(P_1)$ and $\models_N L(P_2) \supset P_2$. Both these last formula have less than n modalities, hence by the inductive hypothesis $\vdash_P \neg P_1 \supset \neg L(P_1)$ and $\vdash_P L(P_2) \supset P_2$. Applying the rule $P \vdash [\alpha]P$ we obtain $\vdash_P [\alpha](L(P_2) \supset P_2)$. We now apply modus ponens and either the axiom $[\alpha](P \supset Q) \supset ([\alpha]P \supset [\alpha]Q)$ or the theorem $[\alpha](P \supset Q) \supset (\langle\alpha\rangle P \supset \langle\alpha\rangle Q)$ to obtain $\vdash_P m(\alpha)L(P_2) \supset m(\alpha)P_2$. Easy applications of modus ponens now give $\vdash_P \neg P_1 \supset m(\alpha)P_2$ or equivalently $\vdash_P P_1 \vee m(\alpha)P_2$. ∎

The Diamond Completeness Theorem can be regarded as establishing the completeness of a system for proving total correctness of programs, if α is restricted to be deterministic. This provides a completeness proof for the Burstall-Manna-Waldinger technique [3,11], which essentially is an informal description of the method of proving $P \supset \langle\alpha\rangle R$, which is incorporated in P. Basu and Yeh [2] have the same notion for convergents; however they do not envisage the application for it that we have presented here. In a future paper, we hope to clarify in more detail the relationships between these and other techniques for proving total correctness of programs.

Acknowledgments

J. Schwarz pointed out the absence of any induction axioms in an earlier axiomatization of regular augmented number theory, prompting the inclusion of the Rule of Convergence, without which half of this paper would not have been written. S. Purcell supplied us with the formula $C(FX) = X \wedge FX \neq Y$ for Theorems 3 and 4. We benefited from discussions of dynamic logic with R. Burstall, S. Litvintchouk, R. Milner, G. Plotkin, M. Fischer, R. Ladner and R. Rivest.

References

[1] de Bakker, J.W., and W.P. de Roever. A calculus for recursive program schemes. in *Automata, Languages and Programming* (ed. Nivat), 167-196. North Holland, 1972.

[2] Basu, S. K. and R. T. Yeh. Strong Verification of Programs. IEEE Trans. Software Engineering, SE-1, 3, 339-345. Sept. 75.

[3] Burstall, R.M. Program Proving as Hand Simulation with a Little Induction. IFIP 1974, Stockholm.

[4] Cook, S.A. Axiomatic and Interpretive Semantics for an Algol Fragment. TR-79, Toronto, Feb. 1975.

[5] Dijkstra, E. *A Discipline of Programming.* Prentice-Hall, Englewood Cliffs, N.J. 1976.

[6] Harel, D., A. Pnueli and J. Stavi. A complete axiomatic system for proving deductions about recursive programs. Proc. Ninth Ann. ACM Symp. on Theory of Computing, Boulder, Col., May 1977.

[7] Hitchcock, P. and D. Park. Induction Rules and Termination Proofs. In *Automata, Languages and Programming* (ed. Nivat, M.), IRIA. North-Holland, 1973.

[8] Hoare, C.A.R. An Axiomatic Basis for Computer Programming. CACM *12*, 576-580, 1969.

[9] Hughes, G.E. and M.J. Cresswell. *An Introduction to Modal Logic.* London: Methuen and Co Ltd. 1972.

[10] Kripke, S. Semantical considerations on Modal Logic. Acta Philosophica Fennica, 83-94, 1963.

[11] Luckham, D., D. Park and M. Paterson. On Formalized Computer Programs. J.CSS *3*, 2, 119-127. May 1970.

[12] Manna, Z. and R. Waldinger. Is "sometime" sometimes better than "always"? Intermittent assertions in proving program correctness. Proc. 2nd Int. Conf. on Software Engineering, Oct. 1976.

[13] Pratt, V.R. Semantical Considerations on Floyd-Hoare Logic. 17th IEEE Symposium on Foundations of Computer Science, Oct. 1976.

[14] Rogers, H. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967.

[15] Tarski, A. The semantic conception of truth and the foundations of semantics. Philos. and Phenom. Res, *4*, 341-376, 1944.