Theoretical Computer Science 79 (1991) 341–355
North-Holland

Note

On strictly arithmetical completeness in logics of programs*

Andrzej Szalas

Institute of Informatics, University of Warsaw, PKiN VIIIp., 00-901 Warsaw, Poland

Communicated by G. Mirkowska
Received December 1988
Revised February 1990

Abstract

Szalas, A., On strictly arithmetical completeness in logics of programs, Theoretical Computer Science 79 (1991) 341–355.

We introduce and discuss a notion of strictly arithmetical completeness related to relative completeness of Cook (1978) and arithmetical completeness of Harel (1978). We present a powerful technique of obtaining strictly arithmetical axiomatizations of logics of programs. Given a model-theoretic semantics of a logic and a set of formulae defining (in a metalanguage) its nonclassical connectives, we automatically derive strictly arithmetically complete and sound proof systems for this logic. As examples of application of the technique we obtain new axiomatizations of algorithmic logic, (concurrent) dynamic logic and temporal logic.

1. Introduction

The current paper is devoted to axiomatizing logics of programs. Such logics play a similar role in computer science to that of the classical logic in "pure" mathematics. Classical formulae, however, mirror the static nature of mathematic notions. On the other hand, dynamic behaviour of programs requires another approach. Currently, in general, researchers agree that the dynamic character of phenomena appearing in most areas of computer science have their counterparts in nonclassical logics.

* This work was supported by grant RP.I.09 of the Polish Ministry of National Education.

The history of development of logics of programs was initiated in the late sixties and may seem a rather short one. Nevertheless the research on logics of programs was very intensive. Many logics have been defined and investigated. They were strongly influenced by the development of new programming tools for expanding applications of computers, and by essential progress in programming methodology. The explosion of various applications of computers resulted in development of many new programming concepts. Over fifteen-hundred different programming languages have been defined. They were usually accompanied by less or more suitable axiom systems for proving correctness of programs. Thus one of the major trends in the area of logics of programs concerns proof systems that enable formal reasoning about program properties.

First-order logics of programs that are intended to express at least the most basic properties of programs as e.g. halting property, cannot be characterized completely (in the classical sense) by finitistic proof systems (cf. e.g. [1, 3, 5, 13, 15]). On the other hand, in order to stay within a finitary framework, one can try to weaken classical notions of completeness. Various nonclassical notions of completeness were defined and new completeness proving techniques were developed (cf. e.g. [2, 4, 5, 14]). The most widely accepted nonclassical notion of completeness is that of relative completeness defined by Cook [4]. He separated the reasoning about programs from reasoning about first-order properties of the underlying interpretation, and proved that the famous Hoare system for proving partial correctness of programs is complete relative to the class of expressive interpretations. Arithmetical completeness was derived from relative completeness by Harel [5] in his works on dynamic logic. Harel restricts the class of admissible interpretations to those containing the arithmetics of natural numbers. The notion of arithmetical completeness was adapted to the case of temporal logic in [14].

In the current paper we introduce a notion of strictly arithmetical (s-arithmetical, in short) completeness and show that the task of finding s-arithmetically complete axiomatizations for a wide class of logics of programs can be reduced to a few typical steps. S-arithmetical completeness differs from arithmetical completeness in its definition of the class of admissible interpretations. We discuss this more precisely in Section 3.

By logic we shall mean in this paper a set of well-formed formulae together with the class of admissible interpretations and a satisfiability relation. We demand that the logic is an extension of classical first-order logic, and has model-theoretic semantics. We consider a class of logics with nonclassical connectives definable at a metalevel by infinite disjunctions (thus, by duality, conjunctions) of formulae. Such a basic characterization is essential in our investigations. However, due to the nature of computations performed by computers, most reasonable constructs can be characterized by the infinite operations we deal with. Consequently, the method presented in our paper is applicable to a large class of logics of programs. For instance, as we show in our examples (Section 2), algorithmic logic, (concurrent) dynamic logic and temporal logic fall into this class.

2. Logics of programs

First let us establish the logical framework assumed in our paper. By M we shall denote an enumerable set of nonclassical connectives. We assume that the connectives are unary. We shall show, however, how to deal with nonclassical connectives that have more than one argument (cf. the **atnext** operator in temporal logic). In the sequel we shall always assume that a first-order signature is fixed. By L we shall then denote the set of many-sorted classical first-order formulae.

Definition 2.1. Let M be an enumerable set of nonclassical connectives. We form an *M-extension of classical first-order logic* (M-logic, in short) as the triple $\langle L(M), C, \models \rangle$, where

(a) $L(M)$ is the set of formulae obtained from L augmented with the following syntax rules:
- $L \subset L(M)$,
- for any $m \in M$ and $A \in L(M)$, $m(A) \in L(M)$;

(b) C is a class of admissible interpretations (we assume C is a subclass of classical first-order interpretations in relational structures);

(c) $\models$ is a satisfiability relation that agrees with the classical one for classical first-order formulae (for $I \in C$, $A \in L(M)$ and a valuation v of free variables, $I, v \models A$, means that A is satisfied by interpretation I and valuation v).

It should be remarked here that relational structures are the only admissible interpretations considered in the above definition. This follows from two basic reasons. First, data types used by programmers in practice can always be described by relational structures. Second, such an assumption allows us to treat different logics in a uniform framework.

Below we shall discuss case studies in logics of programs. The notion of proof systems for logics of programs will be discussed later (Section 3). Let us only remark here that searching for (arithmetically) complete proof systems for logics of programs was usually considered a difficult task, all the more since each of those logics seemed considerably different from other logics and thus it seemed necessary to develop new techniques of proving completeness theorems. The main motivation of the case studies discussed below is then to show that from a certain point of view logics of programs are very close to each other.

In definitions of logics of programs we consider only the most important parts of the logics. We also reformulate the original definitions of semantics, in order to stay within the framework described above.

Definition 2.2. By *algorithmic logic* (AL) we shall mean the M_{AL}-logic satisfying the following conditions (cf. e.g. [3, 11]):

(a) set M_{AL} contains connectives of the form $[P]$ and $[\bigcup P]$, where P is a program, i.e. an expression defined inductively as follows:
- $z := t$ is a program, where z is a variable, and t is a term,
- $P;Q$ and **if** G **then** P **fi** are programs, where P and Q are programs, and G is an open formula;

(b) the class of admissible interpretations is the class of classical first-order interpretations;

(c) the satisfiability relation of AL, $\models_{AL}$, is defined as follows:
- for classical connectives and first-order quantifiers $\models_{AL}$ agrees with the satisfiability relation of classical first-order logic,
- $I,v \models_{AL} [z := t]A$ iff $I,v \models_{AL} A(z/t)$, where $A \in L$ and $A(z/t)$ means the formula obtained from A by replacing z by t renaming the free variables of t which are bound in A if necessary.
- $I,v \models_{AL} [P;Q]A$ iff $I,v \models_{AL} [P]([Q]A)$,
- $I,v \models_{AL}$ [**if** G **then** P **fi**]A iff $I,v \models_{AL} (G \to [P]A) \wedge (\neg G \to A)$,
- $I,v \models_{AL} [\bigcup P]A$ iff there is $i \in \omega$ such that $I,v \models_{AL} [P^i]A$, where $[P^0]A = A$, and $[P^{i+1}]A = [P][P^i]A$.

We shall say that interpretation I is an AL-model for a formula $A \in L(M_{AL})$, and denote this by $I \models_{AL} A$, iff for any valuation v of free variables, $I,v \models_{AL} A$.

Definition 2.3. By *dynamic logic* (DL) we shall mean the M_{DL}-logic satisfying the following conditions (cf. e.g. [5]):

(a) set M_{DL} contains connectives of the form $\langle P \rangle$, where P is a program, i.e. an expression defined inductively as follows:
- $z := t$ is a program, where z is a variable and t is a term,
- $G?$ is a program, where G is an open formula,
- $P;Q$, $P \cup Q$, P^* are programs, where P and Q are programs;

(b) the class of admissible interpretations is the class of classical first-order interpretations;

(c) the satisfiability relation of DL, $\models_{DL}$, is defined as follows:
- for classical connectives and first-order quantifiers $\models_{DL}$ agrees with the satisfiability relation of classical first-order logic,
- $I,v \models_{DL} \langle z := t \rangle A$ iff $I,v \models_{AL} A(z/t)$, where $A \in L$ and $A(z/t)$ is defined in Definition 2.2,
- $I,v \models_{DL} \langle G? \rangle A$ iff $I,v \models_{DL} G \wedge A$,
- $I,v \models_{DL} \langle P;Q \rangle A$ iff $I,v \models_{DL} \langle P \rangle (\langle Q \rangle A)$,
- $I,v \models_{DL} \langle P \cup Q \rangle A$ iff $I,v \models_{DL} \langle P \rangle A$ or $I,v \models_{DL} \langle Q \rangle A$,
- $I,v \models_{DL} \langle P^* \rangle A$ iff there is $i \in \omega$ such that $I,v \models_{DL} \langle P^i \rangle A$.

We shall say that I is a DL-model for a formula $A \in L(M_{DL})$, and denote this by $I \models_{DL} A$, iff for any valuation v, $I,v \models_{DL} A$.

Note that the main difference between AL and DL is that AL concerns deterministic programs, whilst DL also contains nondeterministic ones (due to program connectives $\cup$ and *). Consequently, $\langle\ \rangle$ means modal possibility (cf. [5]).

Definition 2.4. By *concurrent dynamic logic* (CDL) we shall mean DL augmented with additional program connective $\cap$, and the following rule concerning its semantics (cf. e.g. [12]):

$$I,v \models_{\mathrm{CDL}} \langle P \cap Q \rangle A \text{ iff } I,v \models_{\mathrm{CDL}} \langle P \rangle A \text{ and } I,v \models_{\mathrm{CDL}} \langle Q \rangle A.$$

We shall say that I is a CDL-model for a formula $A \in L(M_{\mathrm{CDL}})$, and denote this by $I \models_{\mathrm{CDL}} A$, iff for any valuation v, $I,v \models_{\mathrm{CDL}} A$.

The last logic to be defined in this section is temporal logic (TL). The version of TL we consider deals with linear and discrete time, points of which correspond to elements of ω. Usually the semantics of TL is given by means of Kripke structures of the form $\langle I,S \rangle$, where I is a first-order interpretation, and S is a sequence of states, $S = (s_i)_{i \in \omega}$. Each state assigns values to so-called local variables (cf. e.g. [9, 10]). In order to stay within the framework of relational structures we shall "encode" the sequence of states by adding new constants to the first-order interpretation. Let us describe the encoding more precisely. Assume $Z = \{z_i\}_{i \in \omega}$ is a set of local variables. A Kripke structure $\langle I,S \rangle$ will be "encoded" by a relational structure which results from I by adding fresh constants $\{c_{ij}\}_{i,j \in \omega}$. Since the thus obtained relational structure has to keep information about the sequence of states, we shall assume the following interpretation of new constants: the value of c_{ij} in I is equal to $s_i(z_j)$. Obviously, local variables have to be eliminated from the language. In order to put the above encoding into good use, we shall assume that every local variable z_j is replaced in the language by a constant symbol c_{0j}.

Now we are ready to define TL.

Definition 2.5. By *temporal logic* (TL) we shall mean the M_{TL}-logic satisfying the following conditions (cf. e.g. [9]):

(a) set $M_{\mathrm{TL}} = \bigcup_{i \in \omega} M_{\mathrm{TL}}^{(i)}$ where $M_{\mathrm{TL}}^{(0)} = \{\mathbf{atnext}_B \mid B \text{ is a classical first-order formula}\}$ and $M_{\mathrm{TL}}^{(i+1)} = M_{\mathrm{TL}}^{(i)} \cup \{\mathbf{atnext}_B \mid B \in L(M_{\mathrm{TL}}^{(i)})\}$;

(b) the class of admissible interpretations is the class of classical first-order interpretations, constants of which contain $\{c_{ij}\}_{i,j \in \omega}$;

(c) the satisfiability relation of TL, $\models_{\mathrm{TL}}$ is defined as follows:

- for classical first-order formulae $\models_{\mathrm{TL}}$ agrees with the satisfiability relation of classical first-order logic,
- $I,v \models_{\mathrm{TL}} A \,\mathbf{atnext}_B$ iff there exists $i \in \omega - \{0\}$ such that $I,v \models_{\mathrm{TL}} (A \wedge B)|i$, and for all $0 < j < i$, $I,v \models_{\mathrm{TL}} \neg B|j$, where by $A|k$ we mean the formula obtained from A by replacing all occurrences of c_{ij} $(i,j \in \omega)$ by $c_{i+k,j}$.

We shall say that I is a TL-model for a formula $A \in L(M_{\mathrm{TL}})$, and denote this by $I \models_{\mathrm{TL}} A$, iff for any valuation v, and any $i \in \omega$, $I,v \models A|i$.

Note that operators $\mathbf{atnext}_B$ correspond to the strong **atnext** operator of Kröger (cf. [9]). In fact $A\,\mathbf{atnext}_B$ means A **atnext** B. We introduced infinitely many operators

A **atnext**$_B$ since the non-classical connectives we consider are one-argument ones. Note also, that we use so-called normal semantics of TL in the above definition.

From the above definitions we can easily prove the following fact.

Fact 2.6. *For any first-order interpretation I and valuation v of free variables the following conditions hold*:

(a) $I,v \models_{\mathrm{AL}} [\bigcup P]A \leftrightarrow A \vee [P][\bigcup P]A$,

(b) $I,v \models_{\mathrm{(C)DL}} \langle P^* \rangle A \leftrightarrow A \vee \langle P \rangle \langle P^* \rangle A$,

(c) $I,v \models_{\mathrm{TL}} A$ **atnext**$_B \leftrightarrow (A \wedge B) \mid 1 \vee (\neg B \wedge A$ **atnext**$_B) \mid 1$.

Fact 2.6 points out the most essential characterization of nonclassical connectives in the considered logics of programs. Namely, the equivalences given in Fact 2.6 have the following common form:

$$x \leftrightarrow \Gamma(x),$$

where $\Gamma(x)$ is defined as $\Gamma(x) = A \vee [P]x$, $\Gamma(x) = A \vee \langle P \rangle x$, $\Gamma(x) = (A \wedge B) \mid 1 \vee (\neg B \wedge x) \mid 1$ for AL, (C)DL and TL respectively. In fact, every formula is defined by its own functional Γ. To indicate this fact we shall denote the functional by Γ with its respective formula as index; e.g.,

$$\Gamma_{[\bigcup P]A}(x) = A \vee [P]x, \qquad \Gamma_{\langle P^* \rangle A}(x) = A \vee \langle P \rangle x,$$

$$\Gamma_{A\,\mathbf{atnext}_B}(x) = (A \wedge B) \mid 1 \vee (\neg B \wedge x) \mid 1.$$

Let us point out here that, from a syntactic point of view, functionals lead from respective sets of formulae to themselves. From a semantic point of view we shall consider them as operations on sets. We shall do this using the obvious correspondence between formulae (ordered by implication) and sets (ordered by inclusion). In the sequel we shall consider (schemes of) functionals as syntactic expressions as well as semantic ones. The context will always exclude confusion.

The following fact gives an alternative characterization of the connectives considered in Fact 2.6, where $\Gamma^0(x) = x$, and for $i \in \omega$, $\Gamma^{i+1}(x) = \Gamma(\Gamma^i(x))$.

Fact 2.7. *For any first-order interpretation I and valuation v of free variables the following conditions hold*:

(a) $I,v \models_{\mathrm{AL}} [\bigcup P]A$ *iff there is* $i \in \omega$ *such that* $I,v \models_{\mathrm{AL}} \Gamma^i_{[\bigcup P]A}$**(false)**,

(b) $I,v \models_{\mathrm{(C)DL}} \langle P^* \rangle A$ *iff there is* $i \in \omega$ *such that* $I,v \models_{\mathrm{(C)DL}} \Gamma^i_{\langle P^* \rangle A}$**(false)**,

(c) $I,v \models_{\mathrm{TL}} A$ **atnext**$_B$ *iff there is* $i \in \omega$ *such that* $I,v \models_{\mathrm{TL}} \Gamma^i_{A\,\mathbf{atnext}_B}$**(false)**.

Fact 2.7 indicates that the logics we consider can be characterized by sets of (schemes of) functionals defining their nonclassical connectives. Note also that connectives in logics of programs can usually be ordered by a well-founded relation. Such an ordering can usually be found in a natural way, e.g. can be given by the

syntax of programs. In what follows we shall then assume that the set M of nonclassical connectives is always suppiemented by a well-founded relation $<_M$.

The above discussion leads to the following definition.

Definition 2.8. We say that set $\Gamma(M) = \{\Gamma_{m(A)}(x) \mid m \in M, A \in L(M)\}$ *defines set* M of non-classical connectives of M-logic provided that the following conditions hold:

(a) for any first-order interpretation I and valuation v of free variables,
- $I,v \models m(A) \leftrightarrow \Gamma_{m(A)}(m(A))$,
- $I,v \models m(A)$ iff there is $i \in \omega$ such that $I,v \models \Gamma^i_{m(A)}(\mathbf{false})$;

(b) there is a well-founded relation $<_M$ on M such that right-hand sides of equalities defining functionals $\Gamma_{m(A)} \in \Gamma(M)$, contain (syntactically) only connectives less (w.r.t. $<_M$) than m.

Definition 2.8 is a basic one in further considerations. Let us note that the phenomenon that logics of programs are definable by sets of (schemes of) functionals we consider is rather general in logics of programs. The reader can notice that there are some similarities between fix-point (or denotational) semantics of programs and the characterization of nonclassical connectives given above. Namely, computation of, say, a loop consists of a sequence of smaller steps. The loop is then characterized as the supremum (or—in other words—the least upper bound) of finite sequences each of which is a better approximation of the loop. A similar situation appears in the characterization of nonclassical connectives given above. To make this more clear suppose $\Phi(P^*)$ is a formula specifying a property of loop P^*. The loop can be characterised as the least upper bound of its initial computation sequences, $P^* = \bigsqcup_{i \in \omega} P^i$. Thus $\Phi(P^*) = \Phi(\bigsqcup_{i \in \omega} P^i)$ and it is natural to demand that $\Phi(P^*) = \bigvee_{i \in \omega} \Phi_i(P^i)$, for some well-defined formulae Φ_i. Moreover, since the next approximation of P^* can be computed from the previous one, the connection between Φ_i and Φ_{i+1} should reflect this computation. In the case of our formulae, in order to mirror this phenomenon, we used $\Gamma^i(\mathbf{false})$ as Φ_i. Note also that the research on propositional μ-calculus (cf. e.g. [8]) shows other arguments that the class of logics definable by the considered sets of functionals is a large one.

Definition 2.9

(a) Given an M-logic, we shall say that set M of nonclassical connectives is *monotone* iff for any interpretation I, $m \in M$, and formulae A, B,

$$I \models A \to B \text{ implies } I \models m(A) \to m(B).$$

(b) Given an M-logic, we shall say that set Γ of functionals is *monotone* iff for any interpretation I, functional $G \in \Gamma$, and formulae A, B,

$$I \models A \to B \text{ implies } I \models G(A) \to G(B).$$

Fact 2.10. *Sets* M_{AL}, M_{DL}, *and* $M_{(\mathrm{C})\mathrm{DL}}$ (*defined in Definitions* 2.2(a), 2.3(a), (2.4) *and* 2.5(a)) *are monotone.*

Proof. The proof can be carried out by easy verification. □

Fact 2.11. *If* M *is a monotone set of nonclassical connectives, and* Γ *is a set of* (*schemes of*) *functionals defined by using connectives of* $M \cup \{\wedge, \vee\}$ *then* Γ *is monotone.*

Proof. The easy proof can be carried out by structural induction on formulae of Γ. □

Definition 2.12. (a) By Γ_{AL} we shall mean the following set of (schemes of) functionals:

$$\Gamma_{[z:=t]A}(x) = A(z/t), \qquad \Gamma_{[P;Q]A}(x) = [P][Q]A,$$
$$\Gamma_{[\textbf{if } G \textbf{ then } P \textbf{ fi}]A}(x) = (\neg G \to A) \wedge (G \to [P]A),$$
$$\Gamma_{[\bigcup P]A}(x) = A \vee [P]x.$$

By the relation $<_{M_{\mathrm{AL}}}$ we shall mean the transitive closure of the smallest relation in which minimal elements take the form $[z := t]$ and which contains all pairs of the form $([P], [P;Q])$, $([Q], [P;Q])$, $([P], [\textbf{if } G \textbf{ then } P \textbf{ fi}])$ and for all $i \in \omega$ pairs of the form $([P]^i, [\bigcup P])$.

(b) By Γ_{DL} we shall mean the following set of (schemes of) functionals:

$$\Gamma_{\langle z:=t\rangle A}(x) = A(z/t), \qquad \Gamma_{\langle G?\rangle A}(x) = G \wedge A,$$
$$\Gamma_{\langle P;Q\rangle A}(x) = \langle P\rangle\langle Q\rangle A, \qquad \Gamma_{\langle P \cup Q\rangle A}(x) = \langle P\rangle A \vee \langle Q\rangle A,$$
$$\Gamma_{\langle P^*\rangle A} = A \vee \langle P\rangle x.$$

By the relation $<_{M_{\mathrm{DL}}}$ we shall mean the transitive closure of the smallest relation in which minimal elements take form $\langle z := t\rangle$, $\langle G?\rangle$ and which contains all pairs of the form $(\langle P\rangle, \langle P;Q\rangle)$, $(\langle Q\rangle, \langle P;Q\rangle)$, $(\langle P\rangle, \langle P \cup Q\rangle)$, $(\langle Q\rangle, \langle P \cup Q\rangle)$ and for all $i \in \omega$ pairs of the form $(\langle P\rangle^i, \langle P^*\rangle)$.

(c) By Γ_{CDL} we shall mean set Γ_{DL} augmented with the following scheme of functionals:

$$\Gamma_{\langle P \cap Q\rangle A}(x) = \langle P\rangle A \wedge \langle Q\rangle A$$

By the relation $<_{M_{\mathrm{CDL}}}$ we shall mean the relation $<_{M_{\mathrm{DL}}}$ augmented with the rule stating that pairs of the form $(\langle P\rangle, \langle P \cap Q\rangle)$, $(\langle Q\rangle, \langle P \cap Q\rangle)$.

(d) By Γ_{TL} we shall mean the following set of (schemes of) functionals:

$$\Gamma_{A\,\mathbf{atnext}_B}(x) = (A \wedge B) \,|\, 1 \vee (\neg B \wedge x) \,|\, 1.$$

By the relation $<_{M_{\mathrm{TL}}}$ we shall mean the transitive closure of the smallest relation in which minimal elements take form $\mathbf{atnext}_B$, where B is a classical formula, and which contains pairs of the form $(\mathbf{atnext}_B, \mathbf{atnext}_C)$, whenever B contains less occurrences of **atnext** than C, where the respective nonclassical connectives are defined in Definitions 2.2, 2.3, 2.4 and 2.5.

Note that most of the functionals defined in Definition 2.12 are constant functionals.

Fact 2.13. Sets Γ_{AL}, $\Gamma_{(C)DL}$ and Γ_{TL} define AL, (C)DL and TL respectively.

Proof. The proof can be carried out by using standard techniques and Definitions 2.2, 2.3, 2.4, 2.5, 2.8 and 2.12. □

Fact 2.14. Sets Γ_{AL}, $\Gamma_{(C)DL}$ and Γ_{TL} are monotone.

Proof. The proof follows immediately from Facts 2.10 and 2.11. □

Let us note that another property common to the considered logics is that their sets of nonclassical connectives are monotone. Similarly, all of those logics can be characterized by monotone sets of (schemes of) functionals. Moreover, the sets of functionals exactly mirror the respective definitions of satisfiability relations. Let us conclude the discussion of this section with the following definition of logics considered in our paper.

Definition 2.15. We shall say that M-logic is *monotone* if M is monotone and there is a monotone set Γ defining connectives of M.

To indicate the fact that set M of non-classical connectives of monotone M-logic is definable by Γ we shall write (M, Γ)-logic instead of M-logic.

3. Strictly arithmetical interpretations

Let us now discuss the class of admissible interpretations we consider in this paper. First, we assume that the s-arithmetical interpretation contains sort ω of natural numbers together with constants 0, 1 and functions $+$, $*$. Next note that programmers deal with potentially infinite data types whose elements, however, are represented by finitistic means. Queues, stacks, arrays, trees, symbols, etc. are always finite. We shall formulate this condition as an assumption that for each sort there is a relation "encoding" its elements as natural numbers.

Let us note that we do not consider finite interpretations, for ω is infinite. On the other hand, finite interpretations are of great importance in the proof theory of logics of programs. The problem of logics of programs interpreted in finite relational structures has been discussed by many authors (cf. e.g. [6, 16]) thus we shall not follow this point here.

The following definition summarizes the discussion.

Definition 3.1. A first-order interpretation I is called *s-arithmetical* provided that

(a) I contains the sort ω of natural numbers together with constants 0, 1 and functions $+$, $*$ interpreted as usual;

(b) for each sort s of I there is an effective binary relation e_s such that for each x of sort s there is exactly one $i \in \omega$ with $e_s(x, i)$ true in I.

Let us remark here that because of the effectiveness of relation e_s all sorts of I are effective (i.e. some natural kind of Turing computability is defined on sorts of I). Note also that the class of s-arithmetical interpretations is a proper subclass of the arithmetical interpretations of Harel [5].

In order to simplify our considerations, in what follows we shall consider one-sorted s-arithmetical interpretations with sort ω, operations 0, 1, $+$, $*$ and additional functions having signature $\omega \to \omega$. In the presence of encoding relations this can be done without loss of generality. Namely, functions and relations on sorts other than ω can be represented by functions with signature $\omega \to \omega$ or $\omega \to \{0, 1\}$, respectively.

Let us now briefly discuss the notion of a partial recursive functional. Namely, by a partial recursive functional we shall mean any functional that, interpreted in an s-arithmetical interpretation, is partial recursive (perhaps relative to some oracle). That is to say, a functional Γ is partial recursive whenever for each formula α and x, given an oracle answering whether $\alpha(x)$ is true, the question whether $\Gamma(\alpha)(x)$ is true, is partial recursive. This notion of partial recursiveness is well known and its precise definition need not be quoted here. The definition of partial recursive functionals that best serves our purposes is to be found in the book of Hinman [7].

Lemma 3.2 (on expressiveness). *For any formula A of an (M, Γ)-logic, and s-arithmetical interpretation I, if functionals of Γ are partial recursive, then there is a classical first-order formula A' such that $I \models A \leftrightarrow A'$.*

Proof. We proceed by structural induction on formula A. If A is a classical first-order formula there is nothing to prove. What remains to show is that, for any $m \in M$ and $B \in L(M)$, there exists formula $C \in L$ such that $I \models C \leftrightarrow m(B)$. Consider $\Gamma_{m(B)}$. $\Gamma_{m(B)}$ is monotone and partial recursive (by assumption). By Definition 2.8(a), $m(B)$ is the least fixed point of $\Gamma_{m(B)}$. By Theorem 3.5 in Hinman [7, p. 92], $m(B)$ is partial recursive, thus first-order definable in the language we deal with. □

As a simple corollary of Lemma 3.2 and Facts 2.10, 2.13, 2.14 we obtain the fact that s-arithmetical interpretations are expressive for AL, (C)DL and TL.

4. Axiomatizing monotone M-logics

In this section we show a powerful technique of obtaining s-arithmetically complete and sound proof systems for monotone M-logics. The notion of proof systems

assumed in our paper is the standard one. As usual, the symbol $\vdash$ stands for the syntactic consequence relation.

Definition 4.1. We say that a proof system P for M-logic is s-arithmetically sound (complete) provided that, for any s-arithmetical interpretation I and any formula $A \in L(M)$,

$$Th_I \vdash A \text{ implies (is implied by) } I \models A$$

where Th_I denotes the set of all classical first-order formulae valid in I.

The above definition differs from the relative completeness of Cook [4] and the arithmetical completeness of Harel [5] in the class of admissible interpretations.

Below we define proof systems for (M, Γ)-logics and show their s-arithmetical soundness and completeness.

Definition 4.2. By proof system $P_{(M,\Gamma)}$ for (M, Γ)-logic we shall mean the proof system containing the following axioms and inference rules:

(1) all instances of classical propositional tautologies;

(2) for all $m \in M$ and formula A such that $\Gamma_{m(A)}(x)$ is a constant functional (syntactically, i.e. a functional containing no occurrences of x) we assume the following axiom:

(LR) $\vdash_{(M,\Gamma)} m(A) \leftrightarrow \Gamma_{m(A)}(\text{false})$;

(3) for all $m \in M$ other than those in (2) we assume the following inference rules:

(L) $\Gamma_{m(A)}(C) \to C, C \to B \vdash_{(M,\Gamma)} m(A) \to B$,

(R) $B \to \exists n C(n), C(n+1) \to \Gamma_{m(A)}(C(n)), \neg C(0) \vdash_{(M,\Gamma)} B \to m(A)$

where n is a variable not appearing in $m(A)$;

(4) rules:

(MP) $A, A \to B \vdash_{(M,\Gamma)} B$,

(M) $A \to B \vdash_{(M,\Gamma)} m(A) \to m(B)$.

Note that the distinction between cases (2) and (3) is not essential. In fact, we could consider case (3) only, a particular case of which is case (2). However, we make the distinction in order to obtain more elegant proof systems. Definition 2.12 shows that constant functionals appear frequently in logics of programs.

Theorem 4.3. *For any (M, Γ)-logic proof system $P_{(M,\Gamma)}$ is s-arithmetically sound.*

Proof. Cases (1) and (MP) are obvious. Soundness of (M) follows from monotonicity of M. What remains to be shown is that rules (L) and (R) are sound (axiom (LR), as a particular case of rules (L) and (R), needs no separate proof).

Assume the premises of rule (L) are true in interpretation I. We shall show that, for all $i \in \omega$, $I \models \Gamma^i_{m(A)}(\mathbf{false}) \to B$. We proceed by induction on i. The case of $i = 0$ is trivial. Assume that $I \models \Gamma^i_{m(A)}(\mathbf{false}) \to C$. Thus, by monotonicity of Γ, $I \models \Gamma_{m(A)}(\Gamma^i_{m(A)}) \to \Gamma_{m(A)}(C)$, and so, since $I \models \Gamma_{m(A)}(C) \to C$, by transitivity of $\to$ we obtain $I \models \Gamma^{i+1}_{m(A)}(\mathbf{false}) \to C$. Now to prove (L) it suffices to use Definition 2.8(a).

Assume the premises of rule (R) are true in interpretation I. We shall show that, for all $i \in \omega$, $I \models C(i) \to \Gamma^i_{m(A)}(\mathbf{false})$. We proceed by induction on i. The case $i = 0$ is trivial, for formula $C(0) \to \Gamma^0_{m(A)}(\mathbf{false})$ is just the last of premises. Assume $I \models C(i) \to \Gamma^i_{m(A)}(\mathbf{false})$. Thus, by monotonicity of Γ, $I \models \Gamma_{m(A)}(C(i)) \to \Gamma_{m(A)}(\Gamma^i_{m(A)}(\mathbf{false}))$. From the second premise of (R) we have $I \models C(i+1) \to \Gamma_{m(A)}(C(i))$, thus, by transitivity of $\to$, $I \models C(i+1) \to \Gamma^{i+1}_{m(A)}(\mathbf{false})$. From the first premise of (R), $I \models B \to \exists n C(n)$. Thus, by Definition 2.8(a), we obtain the desired result. □

Lemma 4.4. *For any (M, Γ)-logic, any formula A of the logic, any classical formula B and any s-arithmetical interpretation I, if functionals of Γ are partial recursive, then*

$$I \models A \leftrightarrow B \text{ implies } Th_I \vdash_{(M,\Gamma)} A \leftrightarrow B.$$

Proof. First we shall show, that $Th_I \vdash_{(M,\Gamma)} A \leftrightarrow B$, where A takes the form $m(D)$ for a classical first-order formula D. We proceed by induction on relation $<_M$. The case of minimal connectives (w.r.t. $<_M$) is similar to that of the induction step. Thus we present both proofs together.

By the proof of Theorem 3.5 in Hinman [7, p. 92] (with a slight adaptation to our formalism), there is a primitive recursive function f and a recursive relation T such that, for all $n \in \omega$,

$$I \models \Gamma^n_{m(D)}(\mathbf{false}) \leftrightarrow \exists u T(f(n+1), z, u)$$

(the vector z represents the free variables of $m(D)$). Denote by $C(n, z)$ the formula $\exists u T(f(n+1), z, u)$. Obviously, by Definition 2.8(a), $I \models m(D) \leftrightarrow \exists n C(n, z)$.

Now we show that $Th_I \vdash_{(M,\Gamma)} B \leftrightarrow m(D)$.

Observe that $I \models B \to m(D)$ implies $I \models B \to \exists n C(n, z)$. Moreover, since $I \models C(n+1, z) \leftrightarrow \Gamma^{n+1}_{m(D)}(\mathbf{false}) \leftrightarrow \Gamma_{m(D)}(\Gamma^n_{m(D)}(\mathbf{false})) \leftrightarrow \Gamma_{m(D)}(C(n, z))$, we have that $I \models C(n+1) \to \Gamma_{m(D)}(C(n, z))$. Clearly, $I \models \neg C(0, z)$. Since $B \to \exists n C(n, z)$ and $\neg C(0, z)$ are classical first-order formulae, $Th_I \vdash_{(M,\Gamma)} B \to \exists n C(n, z)$ and $Th_I \vdash_{(M,\Gamma)} \neg C(0, z)$. By Definition 2.8(b), $\Gamma_{m(D)}(C(n, z))$ contains neither the non-classical connective m nor those greater than m. Thus the case when m is minimal is proved, for $\Gamma_{m(D)}(C(n, z))$ is then simply a classical formula. In the case of the induction step we use here inductive assumption in order to obtain $Th_I \vdash_{(M,\Gamma)} C(n+1, z) \to \Gamma_{m(D)}(C(n, z))$. Note that usually $\Gamma_{m(D)}(C(n, z))$ is not of the form $m'(D')$. In such a case we use inductive assumption and the procedure of eliminating nonclassical connectives described at the end of this proof, in order to prove equivalence between $\Gamma_{m(D)}(C(n, z))$ and some classical formula. Now, applying rule (R), we obtain $Th_I \vdash_{(M,\Gamma)} B \to m(D)$, i.e. $Th_I \vdash_{(M,\Gamma)} B \to A$.

Next let us show that $Th_I \vdash_{(M,\Gamma)} m(D) \to B$. Consider formula $C(z)$ defined as $\exists n C(n, z)$, where $C(n, z)$ is defined above. Since, by Definition 2.8(a), $I \models \Gamma_{m(D)}(m(D)) \leftrightarrow m(D)$, and by monotonicity of Γ, we conclude $I \models \Gamma_{m(D)}(C(z)) \to C(z)$. By classical propositional reasoning we obtain $I \models C(z) \to B$. Since $C(z)$ is a classical formula, by Definition 2.8, $\Gamma_{m(D)}(C(z))$ contains only nonclassical connectives less than m. From the inductive assumption we thus have $Th_I \vdash_{(M,\Gamma)} \Gamma_{m(D)}(C(z)) \to C(z)$. Since $C(z) \to B$ is a classical formula, $Th_I \vdash_{(M,\Gamma)} C(z) \to B$. Now, applying rule (L) we obtain $Th_I \vdash_{(M,\Gamma)} m(D) \to B$, i.e. $Th_I \vdash_{(M,\Gamma)} A \to B$.

Summing up, we proved $Th_I \vdash_{(M,\Gamma)} A \leftrightarrow B$ for A of the form $m(D)$.

Now, if A is more complex than $m(D)$, we can eliminate all occurrences of nonclassical connectives, starting from innermost ones, by proving their equivalence to respective first-order formulae and then substituting the former by the latter. By Lemma 3.2 this can always be done. Such a procedure results in equivalent classical first-order formula. □

Theorem 4.5. *For any (M, Γ)-logic, if all functionals of Γ are partial recursive, then proof system $P_{(M,\Gamma)}$ is s-arithmetically complete.*

Proof. We need to show that $I \models A$ implies $Th_I \vdash_{(M,\Gamma)} A$, where I is an s-arithmetical interpretation. We proceed by structural induction on A. The case when A is a classical first-order formula is obvious.

Assume A is not classical. By Lemma 3.2 there is a classical first-order formula A' such that $I \models A \leftrightarrow A'$. From Lemma 4.4 we have $Th_I \vdash_{(M,\Gamma)} A \leftrightarrow A'$. Obviously, since $I \models A$, $I \models A'$ thus $A' \in Th_I$, and so $Th_I \vdash_{(M,\Gamma)} A'$. Now by classical propositional reasoning we obtain $Th_I \vdash_{(M,\Gamma)} A$. □

5. Examples of applications

In this section we show new axiomatizations of AL, (C)DL and TL. The theorems given below immediately follow from Theorems 4.3, 4.5, Facts 2.13, 2.14 and the simple observation that functionals defining nonclassical connectives are partial recursive (perhaps relatively to finite parts of first-order theories of respective s-arithmetical interpretations).

Theorem 5.1. *The following proof system for* AL *is s-arithmetically sound and complete*

(1) *all instances of classical propositional tautologies,*

(2) $\vdash_{\mathrm{AL}} [z := t]A \leftrightarrow A(z/t)$ *where A is an open formula,*

(3) $\vdash_{\mathrm{AL}} [P;Q]A \leftrightarrow [P][Q]A$,

(4) $\vdash_{\mathrm{AL}} [\textbf{if } G \textbf{ then} P \textbf{ fi}]A \leftrightarrow (\neg G \to A) \wedge (G \to [P]A)$,

(5) $(A \vee [P]C) \to C,\ C \to B \vdash_{\mathrm{AL}} [\bigcup P]A \to B$,

(6) $B \to \exists n C(n),\ C(n+1) \to (A \vee [P]C(n)),\ \neg C(0) \vdash_{\mathrm{AL}} B \to [\bigcup P]A$ *where* n *does not appear in* $[\bigcup P]A$,

(7) $A, A \to B \vdash_{\mathrm{AL}} B$,
$A \to B \vdash_{\mathrm{AL}} [P]A \to [P]B$,
$A \to B \vdash_{\mathrm{AL}} [\bigcup P]A \to [\bigcup P]B$.

Theorem 5.2. *The following proof system for* (C)DL *is s-arithmetically sound and complete*:

(1) *all instances of classical propositional tautologies*,

(2) $\vdash_{(\mathrm{C})\mathrm{DL}} \langle z := t\rangle A \leftrightarrow A(z/t)$ *where* A *is an open formula*,

(3) $\vdash_{(\mathrm{C})\mathrm{DL}} \langle G?\rangle A \leftrightarrow G \wedge A$,

(4) $\vdash_{(\mathrm{C})\mathrm{DL}} \langle P;Q\rangle A \leftrightarrow \langle P\rangle\langle Q\rangle A$,

(5) $\vdash_{(\mathrm{C})\mathrm{DL}} \langle P \cup Q\rangle A \leftrightarrow \langle P\rangle A \vee \langle Q\rangle A$,

(6) $\vdash_{\mathrm{CDL}} \langle P \cap Q\rangle A \leftrightarrow \langle P\rangle A \wedge \langle Q\rangle A$,

(7) $(A \vee \langle P\rangle C) \to C,\ C \to B \vdash_{(\mathrm{C})\mathrm{DL}} \langle P^*\rangle A \to B$,

(8) $B \to \exists n C(n),\ C(n+1) \to (A \vee \langle P\rangle C(n)),\ \neg C(0) \vdash_{(\mathrm{C})\mathrm{DL}} B \to \langle P^*\rangle A$ *where* n *does not appear in* $\langle P^*\rangle A$,

(9) $A, A \to B \vdash_{(\mathrm{C})\mathrm{DL}} B$,
$A \to B \vdash_{(\mathrm{C})\mathrm{DL}} \langle P\rangle A \to \langle P\rangle$.

Theorem 5.3. *The following proof system for* TL *is s-arithmetically sound and complete*:

(1) *all instances of classical propositional tautologies*,

(2) $((A \wedge B) \mid 1 \vee (\neg B \wedge C) \mid 1) \to C,\ C \to D \vdash_{\mathrm{TL}} A\ \mathbf{atnext}_B \to D$,

(3) $D \to \exists n C(n),\ C(n+1) \to ((A \wedge B) \mid 1 \vee (\neg B \wedge C(n)) \mid 1),\ \neg C(0) \vdash_{\mathrm{TL}} D \to A\ \mathbf{atnext}_B$ *where* n *does not appear in* $A\ \mathbf{atnext}_B$,

(4) $A, A \to B \vdash_{\mathrm{TL}} B$,
$A \to B \vdash_{\mathrm{TL}} A\ \mathbf{atnext}_D \to B\ \mathbf{atnext}_D$.

Note that rules 5.1(5), 5.2(7) and 5.3(2) can be reformulated (by classical propositional reasoning) as follows:

5.1(5′) $A \to C, [P]C \to C, C \to B \vdash_{\mathrm{AL}} [\bigcup P]A \to B$,

5.2(7′) $A \to C, \langle P\rangle C \to C, C \to B \vdash_{\mathrm{DL}} \langle P^*\rangle A \to B$,

5.3(2′) $(A \wedge B) \mid 1 \to C, (\neg B \wedge C) \mid 1 \to C, C \to D \vdash_{\mathrm{TL}} A\ \mathbf{atnext}_B \to D$.

Thus the above rules can be considered as rules for "backward" invariants.

Note that in the case of temporal logic we introduced some auxiliary symbols like $|1$, c_{ij}, in order to make our approach applicable to this logic. On the other hand, the auxiliary operator $|1$ means exactly the same as nexttime operator $\bigcirc$ of "pure" temporal logic (defined as $\mathbf{atnext}_{\mathbf{true}}$ in terms of the language we deal with). Constants c_{ij} correspond to temporal terms of the form $\bigcirc^i z_j$. Thus one can easily eliminate the auxiliary symbols from proof system obtained in Theorem 5.3 and formulate the system in terms of "pure" temporal logic.

Acknowledgment

I would like to thank the anonymous referees for contributing helpful comments and suggestions.

References

[1] H. Andréka, I. Németi and I. Sain, Completeness problems in verification of programs and program schemes, in : *Proc. MFCS '79*, Lecture Notes in Computer Science **74** (Springer, Berlin, 1979) 208–218.

[2] H. Andréka, I. Németi and I. Sain, A complete logic for reasoning about programs via nonstandard model theory (Parts I and II), *Theoret. Comput. Sci.* **17** (1982) 193–212 and 213–216.

[3] L. Banachowski, A. Kreczmar, G. Mirkowska, H. Rasiowa and A. Salwicki, An introduction to algorithmic logic, in : A. Mazurkiewicz and Z. Pawlak, eds., *Mathematical Foundations in Computer Science* (Banach Centre Publications, PWN, Warsaw 1977) 7–99.

[4] S.A. Cook, Soundness and completeness of axiom system for program verification, *SIAM J. Comput.* **7** (1) (1978) 70–90.

[5] D. Harel, *First-Order Dynamic Logic*, Lecture Notes in Computer Science **68** (Springer, Berlin, 1978).

[6] D. Harel and D. Peleg, On static logics, dynamic logic and complexity classes, *Inform. and Control* **60** (1984) 86–102.

[7] P.G. Hinman, *Recursion-Theoretic Hierarchies* (Springer, Berlin, 1978).

[8] D.C. Kozen, Results on propositional μ-calculus, in: *Proc. 9th ICALP*, Lecture Notes in Computer Science **140** (Springer, Berlin, 1982) 348–359.

[9] F. Kröger, *Temporal Logic of Programs*, EATCS Monographs in Computer Science **8** (Springer, Berlin, 1987).

[10] Z. Manna and A. Pnueli, Verification of concurrent programs: the temporal framework, in: R.S. Boyer and J.S. Moore, eds., *The Correctness Problem in Computer Science* (Academic Press, New York, 1981) 215–273.

[11] G. Mirkowska and A. Salwicki, *Algorithmic Logic* (Reidel, Dordrecht and PWN, Warsaw, 1987).

[12] D. Peleg, Concurrent dynamic logic, *J. ACM* **34** (2) (1987) 450–479.

[13] A. Szalas, Concerning the semantic consequence relation in first-order temporal logic, *Theoret. Comput. Sci.* **47** (1986) 329–334.

[14] A. Szalas, Arithmetical axiomatization of first-order temporal logic, *Inform. Process. Lett.* **26** (1987/1988) 111–116.

[15] A. Szalas and L. Holenderski, Incompleteness of first-order temporal logic with Until, *Theoret. Comput. Sci.* **57** (1988) 317–325.

[16] J. Tiuryn and P. Urzyczyn, Some Relationships between logics of programs and complexity theory, *Theoret. Comput. Sci.* **60** (1988) 83–108.