ANNALES SOCIETATIS MATHEMATICAE POLONAE
Series IV: FUNDAMENTA INFORMATICAE I. 1 (1977), 19-32

Effectivity problems of algorithmic logic

ANTONI KRECZMAR

University of Warsaw

Received November 13, 1975

AMS Categories: 68A10

Abstract. In the paper we solve some effectivity problems of program schemas. Such properties of programs as, for example, the strong and the weak equivalence, the correctness and the partial correctness of a program, the halting problem are classified in Kleene–Mostowski hierarchy. A basic tool used in the paper is algorithmic logic.

Introduction

The subject we consider in the present paper is now very fashionable. We shall deal with effectivity problems such as recursiveness, degrees of unsolvability and arithmetical classes of notions investigated in the theory of programming. In opposition to many previous publications we shall try to show that these problems can be solved in a uniform way owing to an appropriate choice of formal language and the applications of metamathematical methods.

In Part I we present in brief the basis of our considerations, namely the algorithmic logic (AL). This language is an extension of the class of expressions which may be treated as programs, into the class of formulas describing properties of programs. At first it will not be clear why we adopt such a definition of formal language, in many points rather uncommon, as we have at our disposal several already known formal approaches. Nevertheless, it will turn out that AL is sufficiently strong and universal to comprise almost all previously introduced theories of programming.

During the last few years the metatheory of AL, called *algorithmic logic*, has been developed at Warsaw University. Now we know many facts about the algorithmic logic. Some of them are of syntactical and model-theoretical character, others can be expressed in terms of recursion theory. The first aspect of the whole theory will be very helpful in studying the latter.

In Part II we shall consider those properties of AL which concern its syntax and semantics. The fact, which is of great importance in that part, is the possibility of a complete syntactic characterization of semantics: in algorithmic logic every semantic proof can be replaced by a syntactic one. The above statement, the basic tool we shall use later, is called the *completeness theorem*. Its meaning has been very often overlooked in papers on the axiomatic approach to the theory of programming. It should be pointed out that this syntactic characterization of semantics is essentially infinitistic, so that the formal proofs are infinite and cannot be replaced by finite ones.

In Part III we shall answer questions mentioned in the title by means of previously developed methods. We shall examine from the point of view of recursion theory such notions as, for example, the strong and weak equivalence of programs, the correctness of a program and also the very much exploited halting problem. These properties will be examined in different classes of models. All of them lie at the bottom of arithmetical hierarchy, strictly speaking in class Π_2^0. The last and the most important theorem proves that the elementary theory of programming has the same degree of unsolvability as the notion of truth in first-order arithmetic. Hence, there are elementary properties of programs on arbitrarily high level of arithmetical hierarchy.

PART I

1. Definition of AL and its realization ([15])

It is a well-known fact that in every program we can avoid the 'go to' statement ([6], [16]). Thus, in the definition of AL we shall introduce only the following three program constructions:

'compoud statement': **begin** K; M **end**
'conditional statement': **if** α **then** K **else** M
'while statement': **while** α **do** K

We admit the following brief notation for these constructions:

$$[KM], \quad \vee[\alpha KM], \quad *[\alpha K].$$

In this notation we also replace the substitution sign $:=$ by $/$. We proceed to describe the alphabet of AL. It consists of an infinite sequence: $x_1, x_2, \ldots$ of individual variables, an infinite sequence: $p_1, p_2, \ldots$ of propositional variables, logical signs, functional symbols, predicates, symbols occurring in programs: $/$, $\vee$, $*$, $[$, $]$, and iteration quantifiers: $\bigcup, \bigcap$.

We also assume that AL includes the symbol of equality $=$, always in the sense of identity.

We distinguish in AL (see [15]) the following classes of expressions: the set T of terms, the set F of open formulas, the set S of parallel substitutions

$$[z_1/w_1, \ldots, z_n/w_n]$$

(where $z_1, \ldots, z_n$ are distinct variables and if z_i is a propositional variable, then w_i is an open formula, otherwise w_i is a term), the set FS of programs defined as the least set of expressions containing S and closed under program constructions.

The notion of input and output variable is introduced in a natural way. We admit also abbreviations:

$$\varDelta[\alpha K] \quad \text{instead of} \quad \varDelta[\alpha K[\,]],$$

where [ ] is a dummy substitution,

$$[K_1 K_2 \ldots K_n] \quad \text{instead of} \quad [\ldots[K_1 K_2] \ldots K_n]$$

and

$$K^i \quad \text{instead of} \quad [KK \ldots\ldots K],$$

where a program K is taken i times.

The set of formulas of AL is defined as the least set containing F, closed under propositional connectives and the following rules:

if α is a formula and $K \in FS$, then

$$K\alpha, \quad \bigcup K\alpha, \quad \bigcap K\alpha$$

are formulas;

if $\alpha(x)$ is a formula and x is an individual variable in $\alpha(x)$, then

$$(\exists x)\alpha(x), \quad (\forall x)\alpha(x)$$

are formulas;

Examples of formulas will be given in I.2.

We assume that the notion of the realization of a language is known. The notation is adopted from [13]. The realization of functional symbols and predicates will be denoted by R, the valuation of variables by v. We define values $\tau_R(v)$, $\alpha_R(v)$, $s_R(v)$, $K_R(v)$ of terms, open formulas, substitutions and programs in the usual way ([15]). Note that $\tau_R(v)$ is an element of the universe, $\alpha_R(v)$ is a Boolean value, $s_R(v)$ is a valuation and $K_R(v)$ either is a defined valuation or is undefined.

Finally we define the realization of formulas:

$$(K\alpha)_R(v) = \begin{cases} \alpha_R(K_R(v)) & \text{if } K_R(v) \text{ is defined,} \\ \text{false} & \text{otherwise;} \end{cases}$$

$$\left(\bigcup K\alpha\right)_R(v) = \sup_{i \in N} (K^i_\alpha)_R(v),$$

$$\left(\bigcap K\alpha\right)_R(v) = \inf_{i \in N} (K^i_\alpha)_R(v),$$

$$((\exists x)\alpha(x))_R(v) = \sup_{a\in|R|} \alpha_R(v'),$$
$$((\forall x)\alpha(x))_R(v) = \inf_{a\in|R|} \alpha_R(v'),$$

where $v'(x) = a$, $v'(z) = z$ for $z \neq x$ and $|R|$ denotes the universe.

The notions of satisfaction and model are the usual ones. If X is a set of formulas, α a formula, then

$$X \models \alpha$$

indicates that every model of X is a model of α. In particular,

$$\models \alpha$$

indicates that α is a tautology.

If $\mathfrak{A}$ is a relational system, then

$$\mathfrak{A} \models \alpha$$

denotes that $\mathfrak{A}$ is a model for the formula α.

2. Algorithmic properties

Let us consider the following formulas of AL:

Stop (K):	$K\mathbf{1}$,
Stop$'$ (K):	$\sim K\mathbf{1}$,
Corr (α, K, β):	$\alpha \to K\beta$,
PCorr (α, K, β):	$\alpha \to (K\mathbf{1} \to K\beta)$,
$K \simeq M$:	$[y_1/x_1, \ldots, y_n/x_n]K \quad M] \bigwedge_{i\leqslant n} (x_i = y_i)$ (where $x_1, \ldots, x_n$ are all variables in programs K and M and all occurrences of x_i in M are replaced by y_i),
$K \equiv M$:	$(K\mathbf{1} \leftrightarrow M\mathbf{1}) \wedge (K \simeq M)$,
Natural:	$(\forall y)[x/0] \bigcup [x/x+1](x = y)$,
Archimedean:	$(\forall y)[x/0] \bigcup [x/x+1](y < x)$,
Characteristic 0:	$[x/1] \bigcap [x/x+1] \sim (x = 0)$,
Torsion:	$(\forall x)[z/x] \bigcup [z/z \cdot x](z = 1)$,
Cyclic:	$(\forall x)[z/a] \bigcup [z/z \cdot a](x = z)$.

Now, let us examine their semantics.

Stop (K) is true iff K is defined. So, $K\mathbf{1}$ is the halting formula for a program K (see [1]).

Stop$'$ (K) is true iff K is undefined. So, this formula expresses the undefiniteness of the program K.

Corr and PCorr define the correctness and partial correctness of K with respect to the input condition α and the output condition β (see [4]).

$K \simeq M$ and $K \equiv M$ define the weak and strong equivalence of programs K and M (see [5]).

These formulas show that the theories of Engeler, Hoare, Igarashi are proper parts of AL. In a similar but more sophisticated way one can prove that the theories of Floyd, Manna, Scott, and others are definable in certain simple extensions of AL. Even the theory of recursive procedures may be investigated on that basis. In this paper we disregard these new modifications of AL and concentrate our attention on the starting point of its definition.

In continuing the analysis of algorithmic properties let us return to the remaining examples of formulas. They have algebraical rather than algorithmical meanings. The iteration quantifiers enable us to characterize non-elementary classes of models, i.e. those which do not have the first order characterization. We shall call a class of models defined by a recursive set of algorithmic formulas an *algorithmic class.*

Thus, the class of models isomorphic with the system of natural numbers, the Archimedean ordered fields, the fields of characteristic zero, the torsion free groups, the cyclic groups are algorithmic. To this list one can add the following: the ring of integers, the field of recursive numbers ([10]), the field of algebraic numbers, the field of geometrically constructible numbers ([3]), the field of rationals etc.

This great variety of algorithmic properties is astonishing. It is natural to inquire at the very beginning whether our definition of language is not too large. We shall show that the removal of iteration quantifiers and program constructions would lead us to a blind alley. We should not be able to prove many interesting facts about programs and their properties. It will turn out that these algorithmic classes of models may be very useful in locating the properties of programs in the arithmetical hierarchy.

3. Normal form theorem

DEFINITION 1. Program K is in a *normal form* iff K: $[s*[\alpha M]]$, where s is a substitution and M is a loop-free program.

LEMMA 1. *Let* K: $\big[[K_1*[\alpha M_1]]\,[K_2*[\beta M_2]]\big]$ *and let* p, q, r *be propositional variables not occurring in* K; *then*:

$$\models K \equiv \big[K_1[p/\mathbf{1}, q/\mathbf{1}, r/\mathbf{1}]\big] * \Big[p \vee q\Big[[p/\alpha \wedge p] \veebar \Big[p M_1 \big[\veebar [r K_2][q/\beta, r/\mathbf{0}] \veebar [q/M_2]\big]\big]\Big]\Big].$$

LEMMA 2. *Let* K: $\veebar\big[\beta\,[K_1*[\alpha_1 M_1]]\,[K_2*[\alpha_2 M_2]]\big]$ *and let* p, q, r *be propositional variables not occurring in* K; *then*:

$$\models K \equiv \big[[p/\beta] \veebar [p K_1 K_2][q/\alpha_1, r/\alpha_2]\big] * \Big[(q \wedge p) \vee (r \wedge \sim p) \veebar \Big[p\big[M_1[q/\alpha_1]\big]\,\big[M_2[r/\alpha_2]\big]\Big]\Big].$$

LEMMA 3. *Let* K: $*\big[\alpha\big[M*[\beta N]\big]\big]$ *and let* p, q *be propositional variables not occurring in* K; *then*:

$$\vDash K \equiv [p/\alpha, q/\mathbf{0}]*\Big[(p\vee q)\leq\Big[q\Big[N[q/\beta]\leq\big[\sim q[p/\alpha]\big]\big][M[p/\alpha, q/\beta]]\Big]\Big].$$

LEMMA 4. *Let* K: $\big[M*[\alpha N]\big]$ *and let* p *be propositional variable not occurring in* K; *then*:

$$\vDash K \equiv [p/\mathbf{1}]*\Big[(p\vee\alpha)\leq\Big[p\big[M[p/\sim p]\big]\,N\Big]\Big].$$

THEOREM 1 (due to Mirkowska [12]). *Every program can be effectively transformed into an equivalent program in a normal form.*

Proof: It follows from Lemmas 1–3 that every program can be transformed into a form $\big[K_1*[\alpha K_2]\big]$, where K_1, K_2 are loop-free programs. From Lemma 4 we obtain a normal form of a program. ■

4. Elimination of program symbols from formulas

In the sequel, s will denote a finite string of substitutions. If α is an open formula, then $\overline{s\alpha}$ is a result of the application of all substitutions s to α, beginning from the innermost one. Thus, for instance, if α: $(x = y)\vee p$ and s: $[x/x+y, y/x]\,[p/p\wedge q]\,[x/x\cdot y]$, then $\overline{s\alpha}$: $\big((x+y)\cdot x = x\big)\vee(p\wedge q)$.

LEMMA 5.

(i) $\vDash s\alpha\leftrightarrow\overline{s\alpha}$ *for an open formula* α;

(ii) $\vDash s[KM]\beta\leftrightarrow s\big(K(M\beta)\big)$;

(iii) $\vDash s\leq[\alpha KM]\beta\leftrightarrow s\big((\alpha\wedge K\beta)\vee(\sim\alpha\wedge M\beta)\big)$;

(iv) $\vDash s*[\alpha K]\beta\leftrightarrow s[p/\mathbf{1}]\cup\big[[p/p\wedge\alpha]K\big](p\wedge\sim\alpha\wedge\beta)$

provided p *is a propositional variable not occurring in a formula* $s*[\alpha K]\beta$.

For the proof see [12].

Remark: From now on the formula obtained from $K\beta$ by one of the equivalences (i)–(iv) will be denoted by $\overline{K\beta}$.

From Lemma 5 and Theorem 1 we obtain an analogue of Engeler's well-known theorem.

THEOREM 2. *For every program* K *one can construct in the effective way a formula* α *of the form* $s\bigcup M\beta$, *where* β *is open and* M *is a loop-free program such that* $\vDash K\mathbf{1}\leftrightarrow\alpha$.

Proof: By Theorem 1, $\vDash K \equiv \big[s*[\gamma M]\big]$. Thus, $\vDash K\mathbf{1}\leftrightarrow\big[s*[\gamma M]\big]\mathbf{1}$ and, by Lemma 4,

$$\vDash \big[s*[\gamma M]\big]\mathbf{1}\leftrightarrow\overline{\big[s*[\gamma M]\big]\mathbf{1}}.$$

Let α: $\overline{\big[s*[\gamma M]\big]\mathbf{1}}$; then $\vDash\alpha\leftrightarrow K\mathbf{1}$ and α has the required properties. ■

PART II

1. Deduction theorem

THEOREM 3. *Suppose that X is a set of formulas of* AL, *α is a closed formula of* AL, *β is a formula of* AL; *then*

$$X \cup \{\alpha\} \models \beta \quad \textit{iff} \quad X \models \alpha \rightarrow \beta .$$

Proof is carried out as in [13].

2. Gentzen style axiomatization

What we wish to do in this section, is to give a syntactic characterization of the relation $\models$. Obviously, there is another way of achieving, the same aim. For instance, we could adopt a Hilbert style axiomatization, as had already been done by Mirkowska [11]. The advantage of our decision will appear when we come to the analysis of effectivity problems. The reason is quite simple. Proofs in the Gentzen style can be carried out in a deterministic way, which is impossible in the Hilbert style.

An exact exposition of this axiomatization would be unbearably cumbersome and it not intended here. Being aware of it, we shall try to shorten all definitions and proofs as far as possible. Unfamiliar readers are referred to [8], [13].

We start from the following definitions.

DEFINITION 2. Let X be a set of equations $\tau_i = \tau_j$ where τ_i, τ_j are terms. We shall say that the terms τ, μ are X *equivalent* ($\tau = \mu[X]$) iff there exists a sequence of terms $\tau_1, \ldots, \tau_n$ such that τ_1 is τ, τ_n is μ and for all $i \leqslant n$ either τ_i is τ_{i+1} or the equation $\tau_i = \tau_{i+1}$ is in X.

DEFINITION 3. $\tau \asymp \mu[X]$ will denote the closure of relation $\tau = \mu[X]$ with respect to extensionality of functional symbols, i.e.

$$\text{if } \tau_i \asymp \mu_i[X],\ i \leqslant n, \text{ then } f(\tau_1, \ldots, \tau_n) \asymp f(\mu_1, \ldots, \mu_n)[X].$$

The relation $\asymp$ is of course a congruence in the algebra of terms.

DEFINITION 4. A *sequent* is a pair of finite sequences of formulas of AL.

We shall write sequents in the form $\Gamma \Rightarrow \Delta$, where Γ is the antecedent and Δ is the consequent in a sequent. As in every Gentzen style axiomatization, we shall speak about sequents and their proofs, and not about formulas and their proofs.

DEFINITION 5. A sequent $\Gamma \Rightarrow \Delta$ is said to be an *axiom* iff one of the following conditions is satisfied: (1) $\tau = \tau \in \Delta$; (2) $\mathbf{1} \in \Delta$; (3) $\mathbf{0} \in \Gamma$; (4) $\Gamma \cap \Delta \neq \emptyset$; (5) There exists a set of equations $X \subset \Gamma$ and a set of terms τ_i, μ_i, $i \leqslant n$, such that $\tau_i \asymp \mu_i[X]$, $i \leqslant n$, and, for a certain predicate ϱ,

$$\varrho(\tau_1, \ldots, \tau_n) \in \Gamma \quad \text{and} \quad \varrho(\mu_1, \ldots, \mu_n) \in \Delta .$$

Schemas of axioms (1)–(4) are quite natural. Schema (5) assures extensionality of predicates.

Rules of inference will be written in the form $\frac{\{S_i\}_{i\in I}}{S}$, where $\{S_i\}_{i\in I}$ is an enumerable sequence of premises, S is the conclusion. The rules are as follows:

(R1a) $\dfrac{\Gamma, sa \Rightarrow \Delta}{\Gamma \Rightarrow s\sim a, \Delta}$, (R1b) $\dfrac{\Gamma \Rightarrow \Delta, sa}{s\sim a, \Gamma \Rightarrow \Delta}$,

(R2a) $\dfrac{\{\Gamma \Rightarrow \Delta, sa;\ \Gamma \Rightarrow \Delta, s\beta\}}{\Gamma \Rightarrow s(a\wedge\beta), \Delta}$, (R2b) $\dfrac{\Gamma, sa, s\beta \Rightarrow \Delta}{s(a\wedge\beta), \Gamma \Rightarrow \Delta}$,

(R3a) $\dfrac{\Gamma \Rightarrow \Delta, \overline{sKa}}{\Gamma \Rightarrow sKa, \Delta}$, (R3b) $\dfrac{\Gamma, \overline{sKa} \Rightarrow \Delta}{sKa, \Gamma \Rightarrow \Delta}$,

(R4a) $\dfrac{\Gamma \Rightarrow sa, \Delta, s\bigcup K(Ka)}{\Gamma \Rightarrow s\bigcup Ka, \Delta}$, (R4b) $\dfrac{\{\Gamma, sK^i a \Rightarrow \Delta\}_{i\in N}}{s\bigcup Ka, \Gamma \Rightarrow \Delta}$,

(R5a) $\dfrac{\Gamma \Rightarrow \Delta, sa(x)}{\Gamma \Rightarrow s(\forall x)a(x), \Delta}$, (R5b) $\dfrac{s[x/\tau]a(x), \Gamma, s(\forall x)a(x) \Rightarrow \Delta}{s(\forall x)a(x), \Gamma \Rightarrow \Delta}$.

DEFINITION 6. A *diagram* of a sequent $\Gamma \Rightarrow \Delta$ is a tree whose nodes are sequents. Two nodes are *coincident* iff one belongs to a set of premises, the other is a conclusion, and moreover, a corresponding inference rule is applied to the leftmost decomposable formula in a sequent, on even levels in an antecedent and on odd levels in a consequent.

Note that a diagram need not to be of a finite order, since according to rule (R4b) one node has an infinite number of coincidents.

DEFINITION 7. $\vdash \Gamma \Rightarrow \Delta$ iff the diagram of $\Gamma \Rightarrow \Delta$ has no infinite path and all ending nodes are axioms of algorithmic logic.

THEOREM 4. $\models a$ *iff* $\vdash \Rightarrow a$.

Sketch of the proof: The implication from right to left follows solely from the correctness of rules of inference and the validity of axioms. To demonstrate the converse, note that every conclusion is equivalent to the conjunction of its premises. Thus, if an ending node is not an axiom, then a cannot be a tautology. Finally, let us consider the case where the diagram of $\Rightarrow a$ has an infinite path. From this path we build a model, in which a is not valid. The universe of this model is a set of classes $[\tau] = \{\mu\colon \tau \asymp \mu[X]\}$, where X is the set of equalities lying on antecedents of this path. Functional symbols are realized as corresponding classes:

$$f_R([\tau_1], \ldots, [\tau_n]) = [f(\tau_1, \ldots, \tau_n)].$$

Relational symbols are realized in the following way:

$$\varrho_R([\tau_1], \ldots, [\tau_n]) = \begin{cases} \text{true} & \text{if } \varrho(\mu_1, \ldots, \mu_n) \text{ is in the antecedent} \\ & \text{for } \mu_i \in [\tau_i], \\ \text{false} & \text{otherwise.} \end{cases}$$

It follows from the axioms that this realization is well defined. Let v be a valuation defined by

$$v(x) = \begin{cases} [x] & \text{if } x \text{ is an individual variable,} \\ \text{true} & \text{if } x \text{ is a propositional variable occurring} \\ & \text{in the antecedent,} \\ \text{false} & \text{otherwise.} \end{cases}$$

Let us suppose that $\alpha_R(v)$ is true. When analysing the inference rules, we can prove by induction on the complexity of formulas that there is an atomic formula in the consequent such that $\varrho_R(\tau_1, \ldots, \tau_n)(v)$ ie true. This contradicts the definition of realization. ■

PART III

1. Some elementary properties of programs

Let $\mathscr{A}$ be a class of models of the same type. We shall consider in this section the following properties of programs:

$$\text{Stop}_{\mathscr{A}} = \{K \in FS\text{: for every model } \mathfrak{A} \in A \ \mathfrak{A} \models K\mathbf{1}\},$$

$$\text{Stop}'_{\mathscr{A}} = \{K \in FS\text{: for every model } \mathfrak{A} \in A \ \mathfrak{A} \models \sim K\mathbf{1}\},$$

$$\text{Eq}_{\mathscr{A}} = \{\langle K, M\rangle \in FS \times FS\text{: for every model } \mathfrak{A} \in A \ \mathfrak{A} \models K \equiv M\}.$$

If $\mathscr{A}$ is the class of all models of the same type, we shall omit the symbol $\mathscr{A}$. Thus, for instance, $\text{Stop} = \{K \in FS\text{: } \models K\mathbf{1}\}$.

In what follows we try to place these properties for various classes of models in the arithmetical hierarchy. Arithmetical classes in Kleene–Mostowski hierarchy are usually denoted by Σ_n^0, Π_n^0. As we do not speak about analytical properties, we can use symbols Σ_n, Π_n.

If $\mathscr{N}$ is the class of models isomorphic to the model of natural numbers $\langle N, 0, S, =\rangle$, then the above properties are those of partial recursive functions, since programs in this model and partial recursive functions are recursively isomorphic. Thus, we have the following results:

$$\text{Stop}_{\mathscr{N}}, \ \text{Eq}_{\mathscr{N}} \in \Pi_2 - \Sigma_2,$$

$$\text{Stop}'_{\mathscr{N}} \in \Pi_1 - \Sigma_1.$$

It is worth while to point out that PCorr is recursively isomorphic to Stop′ and Corr is recursively isomorphic to Stop in any class of models. Moreover, the set of all weakly equivalent programs is recursively isomorphic to Stop in any class of models.

This follows from the equivalences:

(1) $\models K\mathbf{1}\leftrightarrow(\mathbf{1}\rightarrow K\mathbf{1})$,

(2) $\models(\alpha\rightarrow K\beta)\leftrightarrow\big[*\big[\sim\alpha[\ ]\big]K*\big[\sim\beta[\ ]\big]\big]\mathbf{1}$,

(3) $\models\sim K\mathbf{1}\leftrightarrow\big(\mathbf{1}\rightarrow(K\mathbf{1}\rightarrow K\mathbf{0})\big)$,

(4) $\models\big(\alpha\rightarrow(K\mathbf{1}\rightarrow K\beta)\big)\leftrightarrow\sim\big[*\big[\sim\alpha[\ ]\big]K*\big[\beta[\ ]\big]\big]\mathbf{1}$,

(5) $\models K\mathbf{1}\leftrightarrow(K\simeq[\ ])$,

(6) $\models(K\simeq M)\leftrightarrow(\mathbf{1}\rightarrow[sKM]\beta)$ where s: $[y_1/x_1,\ldots,y_n/x_n]$ and β: $\bigwedge\limits_{i\leqslant n}(x_i=y_i)$.

2. Properties of programs valid in all models

THEOREM 5. $\mathrm{Stop}\in\Sigma_1-\Pi_1$; $\mathrm{Stop}'\in\Pi_1-\Sigma_1$; $\mathrm{Eq}\in\Pi_2-\Sigma_2$.

Proof: $\mathrm{Stop}\in\Sigma_1$ follows from the axiomatization, since finite rules are used only in the proof of the sequent $\Rightarrow K\mathbf{1}$.

$\mathrm{Stop}\notin\Pi_1$: see [9].

$\mathrm{Stop}'\in\Pi_1$. In fact, $\models\sim K\mathbf{1}$ iff $\vdash K\mathbf{1}\Rightarrow$ iff $\vdash s\bigcup M\beta\Rightarrow$ by Theorem 2. After the application of (R4b) the last assertion is equivalent to the infinite sequence of assertions: for every i $\vdash sM^i\beta\Rightarrow$. But the relation $\vdash sM^i\beta\Rightarrow$ is recursive since M is a loop-free program. Hence, $\mathrm{Stop}'\in\Pi_1$.

$\mathrm{Stop}'\notin\Sigma_1$: see [9].

$\mathrm{Eq}\in\Pi_2$. $K\equiv M$ iff $\vdash K\mathbf{1}\Rightarrow M\mathbf{1}$ and $\vdash M\mathbf{1}\Rightarrow K\mathbf{1}$ and $\vdash\Rightarrow K\simeq M$. Let us consider a diagram of $K\mathbf{1}\Rightarrow M\mathbf{1}$. As in the case of Stop′, (R4b) will be used only once. The remaining relation is recursively enumerable as in the case of Stop. Since Π_2 is closed under intersection, $\mathrm{Eq}\in\Pi_2$.

$\mathrm{Eq}\notin\Sigma_2$. Let us suppose the contrary. At first we limit the set of non-logical constants to the zero-argument 0 and the one-argument S functional symbols. Every partial recursive function φ_i is of course programmable in this language.

Let the program $K_i(x)$ compute $\varphi_i(x)$. Consider the two programs:

$$M\colon\ \big[[y/0]*\big[y\neq x[y/S(y)]\big][y/0]\big],$$
$$P_i\colon\ \big[M\big[K_i[y/0]\big]\big].$$

Then $P_i\equiv M$ iff $P_i\mathbf{1}\leftrightarrow M\mathbf{1}$. By assumption, $\{i\colon\models P_i\mathbf{1}\leftrightarrow M\mathbf{1}\}\in\Sigma_2$. From the completeness theorem we get

$$\{i\colon\vdash P_i\mathbf{1}\Rightarrow M\mathbf{1}\text{ and }\vdash M\mathbf{1}\Rightarrow P_i\mathbf{1}\}\in\Sigma_2.$$

Arithmetical class will not change if the set of logical axioms is increased by the following one: $\Gamma\Rightarrow\Delta$ where (1) $\big(S(x)=0\big)\in\Gamma$ or (2) $\big(S(x)=S(y)\big)\in\Gamma$ and $(x=y)\in\Delta$.

Thus, by the completeness theorem, the set $U = \{i\colon \alpha \vdash P_i\mathbf{1} \leftrightarrow M\mathbf{1}\}$ is in Σ_2, where α: $\big(S(x) \neq 0\big) \wedge \Big(\big(S(x) = S(y)\big) \rightarrow (x = y)\Big)$. If $\mathfrak{A} = \langle A, o, s\rangle$ is a model of α, then a subsystem $\mathfrak{B} = \langle B, o, s\rangle$, where $B = \{s^i(o)\colon i \in N\}$, is isomorphic to $\langle N, O, S\rangle$. Thus, for every model $\mathfrak{A}$ of α, we have

for every $a \in A$, $M(a)$ *is defined iff* $P_i(a)$ *is defined.*

But $M(a)$ is defined iff an element a is in B.

We have obtained the following chain of equivalent sentences:

(i) $i \in U$;

(ii) for every model $\mathfrak{A}$ of α and for every $a \in A$ $M(a)$ is defined iff $P_i(a)$ is defined;

(iii) for every model $\mathfrak{A}$ of α and for every $a \in A$, $a \in B$ iff $P_i(a)$ is defined;

(iv) for the model $\langle N, O, S\rangle$, for every $a \in N$ $P_i(a)$ is defined;

(v) φ_i is total.

But the last condition is Π_2-complete (see [14]); hence, U cannot be Σ_2. ■

3. Properties of programs in the field of real numbers

We shall denote by $\mathscr{R}$ the class of models isomorphic to the model of real numbers $\langle R, +, -, \cdot, ^{-1}, 0, 1\rangle$. In this section we shall need the following lemma due to Engeler [3]:

LEMMA 6. *If* φ *is a Boolean combination of formulas* $K\alpha$ *where* α *is open, then*

$$\mathscr{R} \vdash \varphi \quad \textit{iff} \quad \text{Formally real} \vdash \varphi$$

where Formally real *is the recursive set of axioms of formally real fields, i.e. fields in which every finite sum of squares is* $\neq -1$.

THEOREM 6. $\mathrm{Stop}_{\mathscr{R}} \in \Sigma_1 - \Pi_1$; $\mathrm{Stop}'_{\mathscr{R}} \in \Pi_1 - \Sigma_1$; $\mathrm{Eq}_{\mathscr{R}} \in \Pi_2 - \Sigma_2$.

Proof: $\mathrm{Stop}_{\mathscr{R}} \in \Sigma_1$. By Lemma 6, we can add to the set of logical axioms the recursive set of open formulas, since the axioms of formally real fields are open.

Now we can proceed as in the case of $\mathrm{Stop} \in \Sigma_1$.

$\mathrm{Stop}_{\mathscr{R}} \notin \Pi_1$. Let K_i have the same meaning as in the proof of the case $\mathrm{Eq} \notin \Sigma_2$ in Theorem 5. Note that K_i may be expressed in the language under consideration. Then a program N_i: $\big[[x/1 + \ldots + 1]K_i\big]$, where 1 is taken i times, may be also expressed in this language. Consequently, $\varphi_i(i)$ is defined iff $\mathscr{R} \vdash N_i\mathbf{1}$.

$\mathrm{Stop}'_{\mathscr{R}} \in \Pi_1$. By the same argument as $\mathrm{Stop}_{\mathscr{R}} \in \Sigma_1$.

$\mathrm{Stop}'_{\mathscr{R}} \notin \Sigma_1$ because $\varphi_i(i)$ is undefined iff $N_i \in \mathrm{Stop}'_{\mathscr{R}}$.

$\mathrm{Eq}_{\mathscr{R}} \in \Pi_2$. As in the case of $\mathrm{Eq} \in \Pi_2$.

$\mathrm{Eq}_{\mathscr{R}} \notin \Sigma_2$. Let M and P_i be such as in the case $\mathrm{Eq} \notin \Sigma_2$.

Thus, $\langle M, P_i\rangle \in \mathrm{Eq}_{\mathscr{R}}$ iff φ_i is total. ■

4. Properties of programs in the ordered field of real numbers

Let $\mathscr{R}_<$ denote the class of models isomorphic to the ordered field of real numbers $\langle R, +, -, \cdot, ^{-1}, 0, 1, < \rangle$. In this section we shall use the following Engeler's lemma ([3]):

LEMMA 7. *If φ is a Boolean combination of formulas Ka where a is open, then*

$$\mathscr{R}_< \models \varphi \quad \textit{iff} \quad \text{Archimedean} \models \varphi.$$

THEOREM 7. $\mathrm{Stop}_{\mathscr{R}_<}, \mathrm{Eq}_{\mathscr{R}_<} \in \Pi_2 - \Sigma_2$; $\mathrm{Stop}'_{\mathscr{R}_<} \in \Pi_1 - \Sigma_1$.

Proof: $\mathrm{Stop}_{\mathscr{R}_<} \in \Pi_2$. By Theorems 2 and 3, $K \in \mathrm{Stop}_{\mathscr{R}_<}$ iff $\models (\forall y)[x/0] \cup \cup [x/x+1](y < x) \to s \cup Ma$. From Theorem 4 it follows that

$$\models (\forall y)[x/0] \bigcup [x/x+1](y < x) \Rightarrow s \bigcup Ma.$$

From axioms of logic we obtain

$$\models \Rightarrow (\exists y)[x/0] \bigcup [x/x+1] \sim (y < x), \quad s \bigcup Ma.$$

This sequent has a proof iff $s \bigcup Ma$ follows solely from axioms or there is such a term τ that for every i, $(\tau < 1 + \ldots + 1) \Rightarrow s \bigcup Ma$ (with 1 taken i times) has a proof. But there are only a finite number of terms τ which occur in more than one formula $sM^i a$. Thus, this case is reduced to the case of the sequent:

$$[x/0] \bigcup [x/x+1](\tau < x) \Rightarrow s \bigcup Ma,$$

for some finite number of possible τ.

$\mathrm{Stop}_{\mathscr{R}_<} \notin \Sigma_2$. Let us consider the program

$$M_i(y)\colon \big[[x/0] * [x < y\,[x/x+1]] K_i(x)\big],$$

where $K_i(x)$ computes $\varphi_i(x)$. Then $M_i \in \mathrm{Stop}_{\mathscr{R}_<}$ iff $K_i(n)$ is defined for every natural n. Hence, $M_i \in \mathrm{Stop}_{\mathscr{R}_<}$ iff φ_i is total.

$\mathrm{Stop}'_{\mathscr{R}} \in \Pi_1$. We have the following chain of equivalent sentences:

(i) $K \in \mathrm{Stop}'_{\mathscr{R}_<}$;

(ii) $\mathscr{R}_< \models \sim s \bigcup Ma$, where M is a loop-free program;

(iii) for every $i \in N$ $\mathscr{R}_< \models (\forall x_1, \ldots, x_n) sM^i a$;

(iv) for every $i \in N$ $\mathscr{R}_< \models (\forall x_1, \ldots, x_n) \beta_i(x_1, \ldots, x_n)$, where β_i is an open formula obtained effectively from $sM^i a$.

From Tarski's theorem ([17]) on decidability of elementary algebra it follows that

$$\mathscr{R}_< \models (\forall x_1, \ldots, x_n) \beta_i(x_1, \ldots, x_n)$$

is the recursive assertion.

$\mathrm{Stop}'_{\mathscr{R}_<} \notin \Sigma_1$. Proof is carried out as in the case of $\mathrm{Stop}_{\mathscr{R}}$.

$\mathrm{Eq}_{\mathscr{R}_<} \in \Pi_2$. By the same argument as for $\mathrm{Stop}_{\mathscr{R}_<}$.

$\mathrm{Eq}_{\mathscr{R}_<} \notin \Sigma_2$. As in the case of $\mathrm{Eq}_{\mathscr{R}}$. ■

Remark: The idea used in the case of $\mathrm{Stop}_{\mathscr{R}_<} \in \Sigma_2$ cannot be imitated in the case of $\mathrm{Stop}_{\mathscr{R}}$. For a program

$$M_i(y)\colon \big[[x/0]*\big[x = y\,[x/x+1]\big]\,K_i(x)\big]$$

we obtain $\mathscr{R}$ non $\vDash M_i\mathbf{1}$ for every i.

COROLLARY. *The relation $<$ is not programmable in the model $\mathscr{R}$.*

This follows immediately from the facts that $\mathrm{Stop}_{\mathscr{R}_<} \notin \Sigma_2$ and $\mathrm{Stop}_{\mathscr{R}} \in \Sigma_1$.

5. Degree of recursive unsolvability of AL

Let V denote the set of all sentences of the first order arithmetic valid in the standard model, and let W denote the set of all tautologies of AL.

THEOREM 8. *The sets V and W are recursively isomorphic.*

For the proof see [7].

COROLLARY. (i) *W is not an arithmetical set*;

(ii) *the infinitistic rules of inference cannot be replaced by finitistic ones.*

Both these facts follow immediately from Theorem 8.

Acknowledgments

The author is indebted to Professor H. Rasiowa and Professor A. Mostowski for their advice and suggestions, and to Dr A. Salwicki who inspired him to take up this work.

References

[1] Engeler, E., *Algorithmic properties of structures*, Math. Sys. Theory 1 (1967), 183–195.

[2] —, *Remarks on the theory of geometrical constructions*, in: *Syntax and Semantics of infinitary languages*, Lecture Notes 72 (1968).

[3] —, *On the solvability of algorithmic problems*, Reports of Eidg. Techn. Hochschule Zürich, Inst. für Informatik, 1974.

[4] Hoare, C. A., *An axiomatic basis for computer programming*, Com. ACM 12 (1969), 576–583.

[5] Igarashi, S., *An axiomatic approach to the equivalence problems of algorithms with application*, Rep. Comp. Cent. Univ. Tokyo 1 (1968).

[6] Knuth, D. E., Floyd, R. W., *Notes on avoiding 'goto' statements*, Inf. Proc. Let. 1 (1971), 23–31.

[7] Kreczmar, A., *Degree of recursive unsolvability of algorithmic logic*, Bull. Acad. Pol. Sci., Sér. Math. Astr. Phys. 20 (1972), 615–618.

[8] Lopez-Escobar, E. G., *The interpolation theorem for denumerably long formulas*, Fund. Math. 57 (1968), 253–272.

[9] Luckham, D. C., Park, D. M., Paterson, M. S.: *On formalized computer programs*, J. Comput. System. Sci. 4 (1970), 220–249.

[10] Mazur, S., *Computable analysis*, Dissertationes Math. 33 (1963).

[11] Mirkowska, G., *On formalized systems of algorithmic logic*, Bull. Acad. Polon. Sci., Sér. Math. Astr. Phys. 19 (1971), 421–428.
[12] –, *Algorithmic logic and its applications*, Doct. diss., Warszawa 1972 (in Polish).
[13] Rasiowa, H., Sikorski, R., *The mathematics of metamathematics*, Monografie Mat. vol. 41, Polish Scient. Publ., Warszawa 1963.
[14] Rogers, H., Jr, *Theory of recursive functions and effective computability*, McGraw Hill, New York 1967.
[15] Salwicki, A., *Formalized algorithmic languages*, Bull. Acad. Pol. Sci., Sér. Math. Astr. Phys. 18 (1970), 227–232.
[16] –, *On the equivalence of FS-expressions and programs*, ibid. 18 (1970), 275–278.
[17] Tarski, A., *A decision method for elementary algebra and geometry*, Berkeley 1951.