ANNALES SOCIETATIS MATHEMATICAE POLONAE
Series IV: FUNDAMENTA INFORMATICAE I. 1 (1977), 1-17

Algorithmic logic and its applications in the theory of programs I

GRAŻYNA MIRKOWSKA

University of Warsaw

Received November 5, 1975

AMS Categories: 68A05. 02C93

Abstract. The paper presents tools for formalizing and proving properties of programs. The language of algorithmic logic constitutes an extension of a programming language by formulas that describe algorithmic properties. The paper contains two axiomatizations of algorithmic logic, which are complete. It can be proved that every valid algorithmic property possesses a formal proof. An analogue of Herbrand theorem and a theorem on the normal form of a program are proved. Results of metamathematical character are applied to theory of programs, e.g. Paterson's theorem is an immediate corollary to Herbrand's theorem.

Introduction

The aim of this paper is to discover general laws connected with programs, their properties and calculations.

A program is an expression of a formal language constructed with use of variables, signs of functions and relations, by means of logical and program connectives. The variables may be interpreted as names of cells in a computer's memory. The definition of an algorithmic language given here allows one (by an alteration of the set of fundamental signs) to obtain different fixed programming languages. Thus it is in fact a definition of a class of languages. The interpretation of the language consists in the definition of the sense of all functional and relational signs. This is in accordance with the programers intuition that they are connecting with the word "implementation" of the language $\mathscr{L}$ in the machine M. For any fixed language we can consider its different interpretations in different (but similar) relational systems. The notion of valuation as a function that assigns values to variables is a formal equivalent to the notion of "memory state". Expressions of the language are interpreted as functions (possibly partial) of the set of valuations into an appropriate set of values that depends on the type of the expression. A program is interpreted as a partial function from the set of valuations W into W. The properties of a program are expressed by formulas of the form $K\alpha$. The value of such a formula is false if the computation of K does not end or if the valuation we have obtained does not satisfy the condition α. The formulas of that form allow us to express at most all elementary properties of the program: halting problem $K\mathbf{1}$, correctness $(\alpha \Rightarrow K\beta)$, equivalence $(K\alpha \Leftrightarrow M\alpha)$.

In this work two axiomatizations of algorithmic logic are given: standard and Gentzen-style axiomatization. Completeness theorem for the respective deductive system is proved. A theorem analogous to the Herbrand theorem in classical logic is obtained. This theorem is useful in investigations connected with classification of problems in the program theory.

Problems presented in this paper have been explicated by many authors in different directions. There have been discussed questions of effectivity [3], many-valued algorithmic logic [8], [9], modular structure of programs and correctness [1], [2], applications of algorithmic logic to procedures [12] and others.

1. Algorithmic languages

The language of algorithmic logic is a formalized language; see [10]. To construct it, we have to distinguish a set of signs called the alphabet and to give rules of creating admissible expressions in that language. The alphabet fixes a language of algorithmic logic (algorithmic language) as well as a programming language. In this section we shall define a class of languages; each of them is uniquely determined by the alphabet. A language of that class is an extension of the language $\mathscr{L}^s$ introduced in [11].

Definition 1. By an *alphabet of algorithmic language* we shall mean a set A which is the union of disjoint and at most enumerable sets

$$V_i,\ V_0,\ \bigcup_{m\in N}\Phi_m,\ \bigcup_{m\in N}P_m,\ L_0,\ L_1,\ L_2,\ Q,\ \Pi,\ U,$$

where

(1) V_i denotes the infinite set of individual variables;

(2) V_0 the infinite set of propositional variables; we assume that the set $V_0 \cup V_i$ is linearly ordered by a certain ordering relation;

(3) $\mathcal{N}$ the set of non-negative integers;

(4) Φ_m the set of m-argument functors;

(5) P_m the set of m-argument predicates;

(6) L_0 the two-element set of logical constants denoted by $\mathbf{1}$ and $\mathbf{0}$;

(7) L_1 the one-element set of one-element logical functor $\sim$, called *negation*;

(8) L_2 the set of two-argument logical functors $\cap$, $\cup$, $\Rightarrow$ called *conjunction*, *disjunction* and *implication*;

(9) Q the set of signs $\bigcap$, $\bigcup$ called the *existential iteration quantifier* and the *universal iteration quantifier*;

(10) Π the set of program connectives $\circ$, $\veebar$, $*$ called *composition*, *branching* and *iteration sign*, respectively;

(11) U the set of auxiliary signs $/$, $[$, $]$, $($, $)$.

We now recall definitions of terms and formulas that are used in classical logic of the first order, see [10].

Definition 2. By the *set of all classical terms* we shall understand the least of expressions T, closed under the following two rules:

(1) if x belongs to the set of individual variables V_i, then x belongs to T;

(2) if φ is an m-argument functor of the alphabet A and if $\tau_1, \ldots, \tau_m$ are terms in T, then the expression $\varphi(\tau_1, \ldots, \tau_m)$ belongs to T.

Definition 3. By the *set of all open classical formulas* we shall understand the least set of expressions F closed under the rules:

(1) if $a \in V_0$, then $a \in F$, and $\mathbf{1} \in F$, $\mathbf{0} \in F$;

(2) if ϱ is an m-argument predicate and if $\tau_1, \ldots, \tau_m$ are terms, then $\varrho(\tau_1, \ldots, \tau_m) \in F$;

(3) if α and β are formulas, i.e. $\alpha, \beta \in F$, then the expressions $\sim \alpha$, $(\alpha \cup \beta)$, $(\alpha \cap \beta)$, $(\alpha \Rightarrow \beta)$ belong to F.

These two notions are used to the construction of other expressions of algorithmic language.

Definition 4. The *set of substitutions* S is the least set containing all sequences of signs from A of the form: $[x_1/\tau_1, \ldots, x_n/\tau_n, a_1/\alpha_1, \ldots, a_m/\alpha_m]$, where $x_1, \ldots, x_n$ denote different individual variables, $a_1, \ldots, a_m$ denote different propositional variables and τ_i for $i = 1, \ldots, n$, α_j for $j = 1, \ldots, m$ denote elements of the set T or F, respectively.

Elements of the set S will be called *substitutions*. Numbers n, m can be equal to zero, i.e. the substitution can have propositional variables $(n = 0, m \neq 0)$ or individual variables only $(m = 0, n \neq 0)$, or the substitution can be empty $(n = 0, m = 0)$.

Definition 5. By the *language of algorithmic logic* we shall mean the system

$$\langle A, T, F, S, FS, FST, FSF\rangle,$$

where A is the alphabet of the language, T is the set of classical terms, F is the set of classical formulas, S is the set of substitutions; FS, FST, FSF are sets of programs, terms, formulas in the algorithmic language defined as follows:

The *set of programs* FS is the least set containing all elements of S and closed under the rule

p. if $\alpha \in F$ and $K, M \in FS$, then the expressions $\circ[KM]$, $\veebar[\alpha KM]$, $*[\alpha K]$ belong to the set FS.

The *set of terms* FST is the least set containing all classical terms T and closed under the rules:

t_1. if $\tau_1, \ldots, \tau_n$ are in FST and if φ is an n-argument functor, then $\varphi(\tau_1, \ldots, \tau_n)$ is in FST;

t_2. if K is in FS and if τ belongs to FST, then $(K\tau)$ is in FST.

The *set of formulas* FSF is the least set containing the set F of all open classical formulas and closed under the rules:

f_1. if $\tau_1, \ldots, \tau_n$ are any terms, $\tau_i \in FST$, $i = 1, \ldots, n$, and if ϱ is an n-argument predicate, $\varrho \in P_n$, then the expression $\varrho(\tau_1, \ldots, \tau_n)$ belongs to FSF;

f_2. if α and β are in FSF, then $\sim\alpha$, $(\alpha\cap\beta)$, $(\alpha\cup\beta)$, $(\alpha\Rightarrow\beta)$ are in FSF;

f_3. if K is a program, i.e. $K \in FS$, and if α is a formula ($\alpha \in FSF$), then the expressions $(K\alpha)$, $\bigcup K\alpha$, $\bigcap K\alpha$ are in FSF.

To conclude this section we introduce other notation frequently used in the sequel.

Let s denote an element of the set S such that

$$s = [x_1/\tau_1, \ldots, x_n/\tau_n, a_1/\alpha_1, \ldots, a_m/\alpha_m]$$

and let ω be any proper expression of the algorithmic language $\mathcal{L}$, i.e. let $\omega \in (FS \cup FSF \cup FST)$. By $\overline{s\omega}$ we shall denote the expression obtained from ω by the simultaneous replacement of all occurences of variables $x_1, \ldots, x_n$ by terms $\tau_1, \ldots, \tau_n$ and variables $a_1, \ldots, a_m$ by formulas: $\alpha_1, \ldots, \alpha_m$.

We shall frequently make use of the following two simple remarks:

(1) If α is an open classical formula and if τ is a classical term, then for every $s \in S$, $\overline{s\alpha}$ is an open classical formula and $\overline{s\tau}$ is a classical term.

(2) If s is of the form $[x_1/y_1, \ldots, x_n/y_n, a_1/b_1, \ldots, a_m/b_m]$, where $x_1, \ldots, x_n, y_1, \ldots, y_n$ are individual variables and $a_1, \ldots, a_m, b_1, \ldots, b_m$ are propositional variables, then for every program K, $\overline{sK}$ is in the set FS.

The set of all variables occuring in the expression ω will be denoted by $V(\omega)$.

Definition 6. By an *elementary formula* in the algorithmic language $\mathcal{L}$ we shall understand any formula of the form $\varrho(\tau_1, \ldots, \tau_n)$, where $\varrho \in P_n$ and $\tau_1, \ldots, \tau_n$ are in FST.

2. Realization of algorithmic languages

Let J be a non-empty set and let B_0 be a complete Boolean algebra with operations $\wedge, \vee, \rightarrow, -$. The unit element of B_0 will be denoted by $\mathbf{1}$ and the zero element by $\mathbf{0}$.

DEFINITION 1. By a *valuation* v in the set J and the algebra B_0 we shall understand any pair of mappings (v^i, v^0) such that

$$v^i\colon V_i \rightarrow J, \qquad v^0\colon V_0 \rightarrow B_0.$$

The set of all valuations will be denoted by W, so that $W = J^{V_i} \times B_0^{V_0}$.

DEFINITION 2. By a *realization* of the language $\mathscr{L}$ in a non-empty set J and in a Boolean algebra B_0 we shall understand any mapping R assigning to every m-argument functor $\varphi \in \Phi_n$ an n-argument operation φ_R in J and to every m-argument predicate $\varrho \in P_m$ an m-argument relation φ_R in J. Any realization R of the language J induces partial mappings: a_R from the set W into J, K_R from the set W into W, and a mapping a_R, a_R: $W \rightarrow B_0$.

We now give precise inducitive definitions of these functions. Let v be an element of the set W; then

rp0. For every substitution $s = [x_1/\tau_1, \ldots, x_n/\tau_n, a_1/a_1, \ldots, a_m/a_m]$,

$$s_R(v) = \hat{v}$$

where $\hat{v} = \{\hat{v}(z)\}_{z \in V_i \cup V_0}$ and $\hat{v}(z) = v(z)$ for $z \notin \{x_1, \ldots, x_n, a_1, \ldots, a_m\}$, $\hat{v}(x_i) = \tau_{iR}(v)$ for $i = 1, \ldots, n$, $\hat{v}(a_i) = \tau_{iR}(v)$ for $i = 1, \ldots, m$;

rp1. Assume that mappings K_R, M_R, a_R are defined; then

$$\circ[KM]_R(v) = \begin{cases} M_R(K_R(v)) & \text{if the mapping } K_R \text{ is defined at the valuation } v \text{ and } M_R \text{ is defined at the valuation } K_R(v), \\ \text{undefined} & \text{otherwise;} \end{cases}$$

$$\veebar[aKM]_R(v) = \begin{cases} K_R(v) & \text{if } a_R(v) = \mathbf{1} \text{ and } K_R(v) \text{ is defined,} \\ M_R(v) & \text{if } a_R(v) \neq \mathbf{1} \text{ and } M_R(v) \text{ is defined,} \\ \text{undefined} & \text{in the opposite case;} \end{cases}$$

$$*[aK]_R(v) = \begin{cases} K^i_R(v)\,(^1) & \text{where } i \text{ is the least natural number such that } (K^i a)_R(v) = \mathbf{0},\ K^i_R(v) \text{ is defined and } (K^j a)_R(v) = \mathbf{1} \text{ for } j < i, \\ \text{undefined} & \text{otherwise;} \end{cases}$$

rt0. For every $x \in V_i$, $x_R(v) = v^i(x)$;

rt1. Given mappings $\tau_{1R}, \ldots, \tau_{nR}$, then for every functor $\varphi \in \Phi_n$, the

---

(1) K^i denotes the program $\underbrace{\circ[K \circ [K \ldots \circ[KK] \ldots]}_{i \text{ times}}$.

realization of the term $\varphi(\tau_1, \ldots, \tau_n)$ defines the following mapping:

$$\varphi(\tau_1, \ldots, \tau_n)_R(v) = \begin{cases} \varphi_R(\tau_{1R}(v), \ldots, \tau_{nR}(v)) & \text{if all mappings } \tau_{1R}, \ldots \\ & \ldots, \tau_{nR} \text{ are defined at} \\ & \text{the valuation } v, \\ \text{undefined} & \text{otherwise;} \end{cases}$$

rt2. Let us assume that mappings K_R, τ_R are defined; then

$$(K\tau)_R(v) = \begin{cases} \tau_R(K_R(v)) & \text{if } K_R(v) \text{ is defined,} \\ \text{undefined} & \text{otherwise;} \end{cases}$$

The mapping a_R will be defined in an analogous way:

rf0. If a is in V_0, then $a_R(v) = v^0(a)$.

Given $\tau_{1R}, \ldots, \tau_{nR}, \alpha_R, \beta_R$ and K_R, we can define:

rf1.

$$\varrho(\tau_1, \ldots, \tau_n)_R(v) = \begin{cases} \varrho_R(\tau_{1R}(v), \ldots, \tau_{nR}(v)) & \text{when all values } \tau_{iR}(v), i \leqslant n, \\ & \text{are defined,} \\ \mathbf{0} & \text{in the opposite case;} \end{cases}$$

rf2.

$$(\alpha \cup \beta)_R(v) = \alpha_R(v) \vee \beta_R(v),$$
$$(\alpha \cap \beta)_R(v) = \alpha_R(v) \wedge \beta_R(v),$$
$$(\alpha \Rightarrow \beta)_R(v) = \alpha_R(v) \rightarrow \beta_R(v),$$
$$(\sim \alpha)_R(v) = -\alpha_R(v);$$

rf3.

$$(K\alpha)_R(v) = \begin{cases} \alpha_R(K_R(v)) & \text{if } K_R\ (v) \text{ is defined,} \\ \mathbf{0} & \text{otherwise,} \end{cases}$$

$$\left(\bigcup K\alpha\right)_R(v) = \text{l.u.b. } \{(K^i\alpha)_R(v)\}_{i \in \mathcal{N}},$$
$$\left(\bigcap K\alpha\right)_R(v) = \text{g.l.b. } \{(K^i\alpha)_R(v)\}_{i \in \mathcal{N}}.$$

DEFINITION 2. We shall say that the *valuation v satisfies the formula α in the realization R* if and only if $\alpha_R(v) = \mathbf{1}$.

The formula α is *valid in the realization R* if $\alpha_R(v) = \mathbf{1}$ for every valuation v.

By a *tautology* we shall understand any formula α that is valid in every realization of the algorithmic language.

DEFINITION 3. We shall say that α is a *closed formula* if in every realization R the value of α does not depend on the choice of the valuation.

3. Properties of realizations

In the sequel, equalities of the form $K_R(v) = M_R(v)$, $\tau_R(v) = \tau'_R(v)$, where $K, M \in FS$ and $\tau, \tau' \in FST$, are to be understood as follows: the lefthand side of the equality is defined if and only if the right-hand side is defined; if both are defined, then they are identical.

Let us observe that

(1) the value of formula, term or program depends only on values of variables that occur in it;

(2) if a program is written without the symbol $*$, then its value is defined in every realization and for every valuation.

Consequently, we have

LEMMA 1. *For every program $K \in FS$, formula $\alpha \in FSF$, term $\tau \in FST$ and for every realization R of the language $\mathcal{L}$ and every valuation v, if $V(K) \cap V(\alpha) = \emptyset$ and $V(K) \cap V(\tau) = \emptyset$, then*

(i) *if $K_R(v)$ is defined then*

$$\alpha_R(v) = (K\alpha)_R(v) = \big(\bigcap K\alpha\big)_R(v),$$

$$\tau_R(v) = (K\tau)_R(v),$$

(ii) $\big(\bigcup K\alpha\big)_R(v) = \alpha_R(v)$.

LEMMA 2. *For every realization R and for every valuation v*

(i) $(s\tau)_R(v) = \overline{s\tau}_R(v)$,

(ii) $(s\alpha)_R(v) = \overline{s\alpha}_R(v)$,

where $\tau \in T$, $s \in S$ and $\alpha \in F$.

Proof: The proof is by induction on the length of the term and of the formula. Let $s = [x_1/\tau_1, \ldots, x_n/\tau_n, a_1/\alpha_1, \ldots, a_m/\alpha_m]$. If $\tau \in V_i$ and $\alpha \in V_0$, i.e. $\tau = x$ and $\alpha = a$, then of course $(sx)_R(v) = \overline{sx}_R(v)$ and $(sa)_R(v) = \overline{sa}_R(v)$ for every R and v. Suppose that the lemma holds for terms $\tau_1, \ldots, \tau_n$ and let us consider a term $\tau = \varphi(\tau_1, \ldots, \tau_n)$ and a formula $\alpha = \varrho(\tau_1, \ldots, \tau_n)$, where $\varphi \in \Phi_n$ and $\varrho \in P_n$. By definition of mappings α_R and τ_R, we have

$$\big(s\varphi(\tau_1, \ldots, \tau_n)\big)_R(v) = \varphi_R\big((s\tau_1)_R(v), \ldots, (s\tau_n)_R(v)\big) = \varphi_R\big(\overline{s\tau}_{1R}(v), \ldots, \overline{s\tau}_{nR}(v)\big)$$
$$= \varphi(\overline{s\tau}_1, \ldots, \overline{s\tau}_n)_R(v) = \overline{s\tau}_R(v)$$

and

$$\big(s\varrho(\tau_1, \ldots, \tau_n)\big)_R(v) = \varrho_R\big((s\tau_1)_R(v), \ldots, (s\tau_n)_R(v)\big) = \varrho_R\big(\overline{s\tau}_{1R}(v), \ldots, \overline{s\tau}_{nR}(v)\big)$$
$$= \varrho(\overline{s\tau}_1, \ldots, \overline{s\tau}_n)_R(v) = \overline{s\alpha}_R(v).$$

Now suppose that $(s\alpha)_R(v) = \overline{s\alpha}_R(v)$ and $(s\beta)_R(v) = \overline{s\beta}_R(v)$. Let γ be of the form $\alpha \otimes \beta$ where $\otimes$ denotes any of the two-argument functors $\cup$, $\cap$ or $\Rightarrow$. Consider the formula $s\gamma$: we have $(s\gamma)_R(v) = \gamma_R\big(s_R(v)\big) = \alpha_R\big(s_R(v)\big) \otimes \beta_R\big(s_R(v)\big) = \overline{s\alpha}_R(v) \otimes \overline{s\beta}_R(v) = \overline{s\gamma}_R(v)$. ■

Let us observe that in the above lemma the fact that we have considered only classical terms and classical open formulas is essential. The replacement of variables by terms and formulas does not always lead from proper expression to proper expression in algorithmic logic. However, if we consider substitutions of a special form, then Lemma 2 can be formulated more generally.

LEMMA 3. *Let s be a substitution of the form $[x_1/y_1, \ldots, x_n/y_n, a_1/b_1, \ldots \ldots, a_m/b_m]$, where $x_1, \ldots, x_n, y_1, \ldots, y_m \in V_i$, $a_1, \ldots, a_m, b_1, \ldots, b_m \in V_0$ and $y_i \neq y_j, b_i \neq b_j$ for $i \neq j$. For every realization R of algorithmic language $\mathcal{L}$ and for every valuation v, the following properties hold:*

(1) *for every $\tau \in FST$, if $\{y_1, \ldots, y_n, b_1, \ldots, b_m\} \cap V(\tau) = \emptyset$, then*

$$(s\tau)_R(v) = \overline{s\tau}_R(v);$$

(2) *for every formula a, if $\{y_1, \ldots, y_n, b_1, \ldots, b_m\} \cap V(a) = \emptyset$, then*

$$(sa)_R(v) = \overline{sa}_R(v);$$

(3) *for every program $K \in FS$, if $\{y_1, \ldots, y_n, b_1, \ldots, b_m\} \cap V(K) = \emptyset$, then*

$(\circ[sK]x_i)_R(v) = (\overline{sK}y_i)_R(v)$ *for $i = 1, \ldots, n$, and*

$(\circ[sK]a_j)_R(v) = (\overline{sK}b_j)_R(v)$ *for $j = 1, \ldots, m$ and*

for $z \in \{y_1, \ldots, y_n, b_1, \ldots, b_m\}$, $z_R\left(s_R^{-1}\left(K_R(v)\right)\right) = z_R\left(\overline{sK}_R\left(s_R^{-1}(v)\right)\right)$.

Lemma 3 implies the important fact that every term can be represented in the form $K\tau$, where τ is a classical term. Consequently, every elementary formula $\varrho(\tau_1, \ldots, \tau_n)$ can be written in the form $K\varrho(\tau_1', \ldots, \tau_n')$, where $\tau_1', \ldots, \tau_n'$ are classical terms. To prove this we must introduce some auxiliary notions.

DEFINITION 1. By a *subterm* of term τ we shall understand every term τ' such that either

(1) τ' is identical with τ, or

(2) if τ is of the form $K\tau''$, then τ' is a subterm of τ'', or

(3) if τ is of the form $\varphi(\tau_1, \ldots, \tau_n)$, then τ' is a subterm of one of the terms $\tau_1, \ldots, \tau_n$.

DEFINITION 2. For every term and for every elementary formula we define by induction on the length of the term the operation χ in the following way:

(1) $\chi(\tau) = \tau$ for every term τ from the set T.

Let us assume that χ is defined for all terms shorter than η.

(2) If η is of the form $K\tau$, then $\chi(K\tau) = K\chi(\tau)$;

(3) If η is of the form $\varphi(\tau_1, \ldots, \tau_n)$, $\varphi \in \Phi_n$, ϱ is an n-argument predicate and i is the smallest number such that $i \leqslant n$, $\tau_i \notin T$ and $K\tau$ is the first on the left subterm of τ_i, then we put

$$\chi\left(\varphi(\tau_1, \ldots, \tau_i, \ldots, \tau_n)\right) = \circ[s^{-1}\overline{sK}]\chi\left(\varphi(\tau_1, \ldots, \tau_{i-1}, \tau', \tau_{i+1}, \ldots, \tau_n)\right),$$

$$\chi\left(\varrho(\tau_1, \ldots, \tau_i, \ldots, \tau_n)\right) = \circ[s^{-1}\overline{sK}]\chi\left(\varrho(\tau_1, \ldots, \tau_{i-1}, \tau', \tau_{i+1}, \ldots, \tau_n)\right),$$

where s is the substitution which to every variable of the set $V(K\tau)$ assigns a variable of the set $V_i \cup V_0 - \bigcup_{i=1}^{n} V(\tau_i)$ according to the ordering relation in the set $V_i \cup V_0$. s^{-1} is the inverse substitution to s and τ' arises from the term τ_i by exchanging $\overline{s\tau}$ into $K\tau$.

LEMMA 4. *For every realization R and valuation v we have: for every term $\tau \in FST$, $\tau_R(v) = \chi(\tau)_R(v)$ and for every elementary formula a, $a_R(v) = \chi(a)_R(v)$.*

Proof: We shall prove the first equality by induction on the length of the term. If $x \in V_i$, then by Definition 2, $x_R(v) = \chi(x)_R(v)$ for every realization R and every valuation v. Let us assume that the lemma holds for all realizations, all valuations and all terms that have less signs than τ^*. We consider three cases:

— if τ^* is a classical term, then obviously

$$\chi(\tau^*)_R(v) = \tau^*_R(v),$$

— if τ^* is of the form $(K\tau)$, then by inductive assumption we obtain

$$\begin{aligned}\chi(K\tau)_R(v) &= \big(K\chi(\tau)\big)_R(v)\\ &= \begin{cases}\tau_R\big(K_R(v)\big) & \text{if } K_R(v) \text{ is defined}\\ \text{undefined} & \text{otherwise}\end{cases}\\ &= (K\tau)_R(v);\end{aligned}$$

— if τ^* is of the form $\varphi(\tau_1, \ldots, \tau_n)$, where τ_i is the first term in the sequence $\tau_1, \ldots, \tau_n$ such that $\tau_i \notin T$ and $K\tau$ is the first subterm of τ_i in that form, then

$$\chi(\tau^*) = \circ[s^{-1}\overline{sK}]\chi\big(\varphi(\tau_1, \ldots, \tau', \tau_{i+1}, \ldots, \tau_n)\big),$$

where s and τ' are as in Definition 2. Now, observe that the term $\varphi(\tau_1, \ldots, \tau', \tau_{i+1}, \ldots, \tau_n)$ has less signs than $\varphi(\tau_1, \ldots, \tau_i, \ldots, \tau_n)$, and so by inductive assumption

$$\varphi(\tau_1, \ldots, \tau', \tau_{i+1}, \ldots, \tau_n)_R(v) = \chi\big(\varphi(\tau_1, \ldots, \tau', \ldots, \tau_n)\big)_R(v).$$

Let us consider a certain realization R and a valuation v. If $K_R(v)$ is defined, then $\circ[s^{-1}\overline{sK}]_R(v)$ is defined, and conversely. So, if $K_R(v)$ is undefined, then $\tau^*_R(v)$ is undefined and $\chi(\tau^*)_R(v)$ is undefined. If $K_R(v)$ is defined, then we have

$$\begin{aligned}&\chi\big(\varphi(\tau_1, \ldots, \tau_n)\big)_R(v)\\ &= \varphi_R\Big(\tau_{1R}\big(\circ[s^{-1}\overline{sK}]_R(v)\big), \ldots, \tau'_R\big(\circ[s^{-1}\overline{sK}_R](v)\big), \ldots, \tau_{nR}\big(\circ[s^{-1}sK]_R(v)\big)\Big).\end{aligned}$$

But $V(\overline{sK}) \cap V(\tau_j) = \emptyset$, so by Lemma 1

$$(\overline{sK}\tau_j)_R\big(s_R^{-1}(v)\big) = \tau_{jR}\big(s_R^{-1}(v)\big) = \tau_{jR}(v) \quad \text{for} \quad j \neq i,$$

and by Lemma 3

$$(\circ[s^{-1}\overline{sK}]\overline{s\tau})_R(v) = \overline{s\tau}_R\big(\overline{sK}_R\big(s_R^{-1}(v)\big)\big) = (s\tau)_R\big(s_R^{-1}\big(K_R(v)\big)\big)$$
$$= \big(s^{-1}(s\tau)\big)_R\big(K_R(v)\big) = (\circ[s^{-1}s]\tau)_R\big(K_R(v)\big)$$
$$= \tau_R\big(\circ[s^{-1}s]_R\big(K_R(v)\big)\big) = (K\tau)_R(v).$$

Hence

$$\chi(\tau^*)_R(v) = \tau_R^*(v). \blacksquare$$

The proof for elementary formulas is similar.

LEMMA 5. *For every term $\tau\in FST$ and for every elementary formula a there exist a program K, a term τ^* and an elementary formula a^* such that for every realization R and for every valuation v,*

$$\tau_R(v) = (K\tau^*)_R(v), \quad a_R(v) = (Ka^*)_R(v). \blacksquare$$

LEMMA 6. *Let R be any realization of the algorithmic language, v any valuation, K, L, M — programs, a, β, γ, δ — formulas, $a, \beta\in F$, $\delta, \gamma\in FSF$, and $\tau, \tau_1, \ldots, \tau_n$ — terms. Then the following equalities hold:*

1p. $\circ\big[K\circ[ML]\big]_R(v) = \circ\big[\circ[KM]L\big]_R(v)$,

2p. $\circ\big[\veebar[aKM]L\big]_R(v) = \veebar\big[a\circ[KL]\circ[ML]\big]_R(v)$,

3p. $\veebar[\mathbf{1}KM]_R(v) = K_R(v) = \veebar[aKK]_R(v)$,

4p. $\veebar[\neg aKM]_R(v) = \veebar[aMK]_R(v)$,

5p. $\veebar[(a\cap\beta)KM]_R(v) = \veebar\big[a\veebar[\beta KM]M\big]_R(v)$,

6p. $\veebar[(a\cup\beta)KM]_R(v) = \veebar\big[aK\veebar[\beta KM]\big]_R(v)$,

7p. $*[aK]_R(v) = \veebar\big[a\circ[K*[aK]]\,[\,]\big]_R(v)$,

1t. $\big(K\varphi(\tau_1, \ldots, \tau_n)\big)_R(v) = \varphi(K\tau_1, \ldots, K\tau_n)_R(v)$ *where* $\varphi\in\Phi_n$,

2t. $(\circ[KM]\tau)_R(v) = \big(K(M\tau)\big)_R(v)$,

1f. $\big(K\varrho(\tau_1, \ldots, \tau_n)\big)_R(v) = \varrho(K\tau_1, \ldots, K\tau_n)_R(v)$ *where* $\varrho\in P_n$,

2f. $\big(K(\gamma\cap\delta)\big)_R(v) = (K\gamma\cap K\delta)_R(v)$,

3f. $\big(K(\gamma\cup\delta)\big)_R(v) = (K\gamma\cup K\delta)_R(v)$,

4f. $(\circ[KM]\gamma)_R(v) = \big(K(M\gamma)\big)_R(v)$,

5f. $(\veebar[aKM]\gamma)_R(v) = \big((a\cap K\gamma)\cup(\neg a\cap M\gamma)\big)_R(v)$,

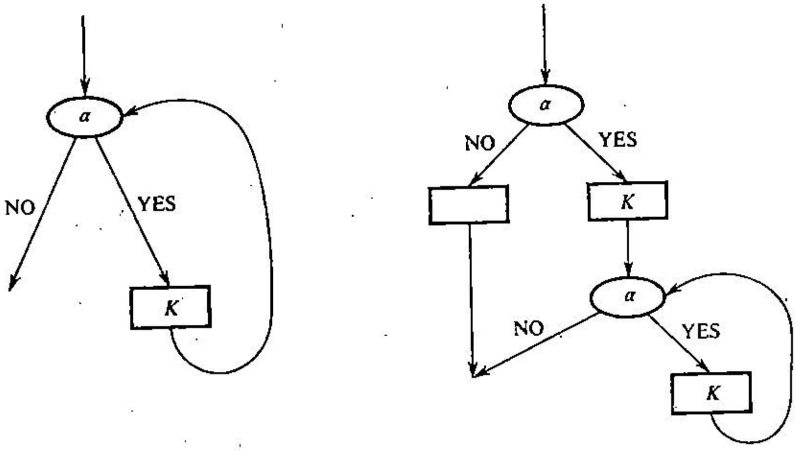
6f. $(*[aK]\gamma)_R(v) = \big(\bigcup\veebar[aK[\,]](\gamma\cap\neg a)\big)_R(v)$,

7f. $\big(\bigcup K\gamma\big)_R(v) = \big(\gamma\cup\bigcup K(K\gamma)\big)_R(v)$,

8f. $\big(\bigcap K\gamma\big)_R(v) = \big(\gamma\cap\bigcap K(K\gamma)\big)_R(v)$.

The proofs of the above equalities are, in general, very simple. We shall prove only two of them.

7p. Let us illustrate the equality by the following diagrams:



Let R and v be any realization and any valuation. By the definition of realization we have

$$*[aK]_R(v) = \begin{cases} K^i_R\ (v) & \text{where } i \text{ is the least natural number such that } (K^i a)_R(v) = \mathbf{0} \text{ and } K^i_R(v) \text{ is defined,} \\ \text{undefined} & \text{if such } i \text{ does not exist;} \end{cases}$$

$$= \begin{cases} v & \text{if } a_R(v) = \mathbf{0}, \\ K^{i-1}_R\big(K_R(v)\big) & \text{if there exists a natural number } i \text{ such that } (K^i a)_R(v) = \mathbf{0} = (K^{i-1} a)_R\big(K_R(v)\big) \text{ and for all } j < i,\ (K^j a)_R(v) = \mathbf{1}, \\ \text{undefined} & \text{otherwise;} \end{cases}$$

$$= \begin{cases} v & \text{if } a_R(v) = \mathbf{0} \\ \circ\big[K * [aK]\big]_R(v) & \text{in the opposite case} \end{cases}$$

$$= \veebar\Big[a \circ \big[K * [aK]\big][\ ]\Big]_R(v).$$

1t. $\big(K\varphi(\tau_1, \ldots, \tau_n)\big)_R(v)$

$$= \begin{cases} \varphi(\tau_1, \ldots, \tau_n)_R\big(K_R(v)\big) & \text{if } K_R(v) \text{ is defined,} \\ \text{undefined} & \text{otherwise;} \end{cases}$$

$$= \begin{cases} \varphi_R\Big(\tau_{1R}\big(K_R(v)\big), \ldots, \tau_{nR}\big(K_R(v)\big)\Big) & \text{if } K_R(v) \text{ is defined and for } i = 1, \ldots, n \text{ terms } \tau_{iR}\big(K_R(v)\big) \text{ are defined,} \\ \text{undefined} & \text{otherwise;} \end{cases}$$

$$= \varphi\big((K\tau_1), \ldots, (K\tau_n)\big)_R(v).$$

6f. $\bigcup \veebar [aK[\ ]](\neg a \cap \gamma)_R(v) = \mathbf{1}$ if and only if there exists a natural number i such that $(\veebar [aK[\ ]]^i(\neg a \cap \gamma))_R(v) = \mathbf{1}$. So there exists the smallest i with this property; let i_0 be this number. Then

$$(1)\qquad \begin{aligned} &(\veebar [aK[\ ]]^{i_0}(\neg a \cap \gamma))_R(v) = \mathbf{1}, \\ &(\veebar [aK[\ ]]^{j}(\neg a \cap \gamma))_R(v) = \mathbf{0}, \quad j < i_0. \end{aligned}$$

Hence $\veebar [aK[\ ]]^{i_0}_R(v) = K^p_R(v)$, where p is a natural number not greater then i_0. Suppose that $p < i_0$. Then there exists a natural number $j_0 < i_0$ such that $a_R(\veebar [aK[\ ]]^{j_0}_R(v)) = \mathbf{0}$. But we have $\veebar [aK[\ ]]^{j_0+1}_R(v) = \veebar [aK[\ ]]_R(\veebar [aK[\ ]]^{j_0}_R(v)) = \veebar [aK[\ ]]^{j_0}_R(v)$ and then for every $j > j_0$ $\veebar [aK[\ ]]^{j_0}_R(v) = \veebar [aK[\ ]]^{j}_R(v)$. This implies that for $j = i_0$,

$$(\neg a \cap \gamma)_R(\veebar [aK[\ ]]^{i_0}_R(v)) = (\neg a \cap \gamma)_R(\veebar [aK[\ ]]^{j_0}_R(v)),$$

which contradicts (1). So $p = i_0$ and $a_R(\veebar [aK[\ ]]^{j}_R(v)) = \mathbf{1}$ for $j < i_0$. As a consequence we have $\veebar [aK[\ ]]^{j}_R(v) = K^j_R(v)$ for $j \leqslant i_0$, and $a_R(K^j_R(v)) = \mathbf{1}$ for $j < i_0$. By (1), $\gamma_R(K^{i_0}_R(v)) = \mathbf{1}$ and $a_R(K^j_R(v)) = \mathbf{0}$ for $j = i_0$. Hence $(*[aK]\gamma)_R(v) = \mathbf{1}$. ∎

As a simple consequence of Lemmas 3–6 we have

LEMMA 7. *For every formula a without symbols $*$, $\bigcap$, $\bigcup$ we can find in an effective way an open formula $a_0 \in F$ such that for every realization R and every valuation v,*

$$a_R(v) = a_{0R}(v)\ . \blacksquare$$

4. The semantic consequence operation

Let R be a realization of the algorithmic language $\mathscr{L}$ as in §2, and let Z be any subset of the set of formulas in the language $\mathscr{L}$.

DEFINITION 1. A realization R is said to be a *model for the set Z* if for every valuation v and for every formula a in Z, $a_R(v) = \mathbf{1}$.

DEFINITION 2. A formula a is said to be a *semantic consequence of the set Z* (in symbols $Z \models a$) if and only if for every realization R of the language $\mathscr{L}$ the following condition holds: if R is a model for the set Z, then for every valuation v, $a_R(v) = \mathbf{1}$. The set of all formulas a such that $Z \models a$ will be denoted by $\mathrm{Cn}(Z)$ and will be called the set of semantic consequences of Z.

DEFINITION 3. The operation which to every set of formulas Z assigns the set $\mathrm{Cn}(Z)$ of all semantic consequences of Z is called the *consequence operation*. If the set Z is empty, then instead of $\emptyset \models a$ we will write $\models a$ and the formula a will be called a *tautology*.

LEMMA 1. *For every set $Z \subset FSF$ and every formula $a \in FSF$, if $Z \models a$, then the set $Z \cup \{\neg a\}$ has no model.* ∎

In the case when the formula α is closed, the above lemma can be strengthened:

$Z \models \alpha$ if and only if the set $Z \cup \{\neg \alpha\}$ has no model.

Now we formulate the following theorem of deduction:

LEMMA 2. *For every set $Z \subset FSF$ and any formulas $\alpha, \beta \in FSF$, if $Z \models (\alpha \Rightarrow \beta)$, then $(Z \cup \{\alpha\}) \models \beta$.* ■

The proof is omitted.

Let us note that the inverse theorem is, in general, not true. Consider the following example: $Z = \emptyset$ and $\beta = (s\alpha)$. Certainly $\{\alpha\} \models (s\alpha)$ but there exist a realization R and a valuation v such that $\alpha_R(v) = \mathbf{0}$ and $\alpha_R(s_R(v)) = \mathbf{0}$ for certain formula α and certain substitution s.

The semantic consequence operation in algorithmic logic has some simple properties, just as the consequence operation in classical logic. These are:

LEMMA 3. *For every sets of formulas X, Y*

(i) $X \subseteq \mathrm{Cn}(X)$;

(ii) *if* $X \subseteq Y$, *then* $\mathrm{Cn}(X) \subseteq \mathrm{Cn}(Y)$;

(iii) $\mathrm{Cn}(\mathrm{Cn}(X)) = \mathrm{Cn}(X)$.

The simple proof of this lemma is omitted. ■

Contrary to classical logic, the consequence operation in algorithmic logic does not satisfy the condition asserting the finiteness of the process of deduction.

THEOREM 1. *The consequence operation* Cn *has not the following property: if $Z \models \alpha$, then there exists a finite subset Z_0 of Z such that $Z_0 \models \alpha$.*

Proof: We shall give an example of the set Z and formula α such that $Z_0 \models \alpha$, but for every finite set $Z_0 \subset Z$, there exists a model for Z_0 which is not a model for the formula α. Let

$$Z = \{([x/0]([x/sx]^i\ 0 \leqslant x))\}_{i \in \mathcal{N}}, \qquad \alpha = ([x/0]\bigcap[x/sx]\ 0 \leqslant x),$$

where 0 is a constant ($0 \in \Phi_0$), s is a one-argument operation and $0 \leqslant$ is a one-argument relation. Let R be a model for the set Z. Then we have $([x/0]\bigcap[x/sx]\ 0 \leqslant x)_R(v) = \mathbf{1}$ for every valuation v. So R is a model for the formula α, and $Z \models \alpha$. Consider any finite subset Z_0 of the set Z. Z_0 is of the form $\{([x/0]([x/sx]^i 0 \leqslant x))\}_{i \in I}$, where I is a finite sequence of natural numbers. Now we define a realization $\bar{R}$ in the set of natural numbers $\mathcal{N}$ as follows: the constant 0 is zero in the set $\mathcal{N}$, the operation s is that of taking the consequent in $\mathcal{N}$, the relation $0 \leqslant$ is the characteristic function of the set I, i.e.

$$0 \leqslant n = \begin{cases} \mathbf{1} & \text{if} \quad n \in I, \\ \mathbf{0} & \text{if} \quad n \notin I. \end{cases}$$

The realization $\bar{R}$ defined in such a way is a model for the set Z_0 because for every $i \in I$ and for every v we have:

$$\big([x/0]\big([x/sx]^i(0\leqslant x)\big)\big)_{\bar R}(v)=(0\leqslant x)_{\bar R}\big([x/sx]^i_{\bar R}([x/0]_{\bar R}(v))\big)$$
$$=0\leqslant \underbrace{s(s\ \dots\ (s0)\ \dots)}_{i\text{ times}}=\mathbf{1}.$$

Nevertheless, if $i\notin I$, then the formula $\big([x/0]([x/sx]^i 0\leqslant x)\big)$ has value $\mathbf{0}$ for every valuation v in the realization $\bar R$. So $a_{\bar R}(v)=\mathbf{0}$. ■

LEMMA 4. *For every formulas a,β and every programs K, M the following conditions hold:*

(i) *If* $\models a$ *and* $\models K\mathbf{1}$, *then* $\models Ka$;

(ii) *If* $\models (a\Rightarrow\beta)$, *then* $\models \big((Ka)\Rightarrow(K\beta)\big)$;

(iii) *If* $\models (a\Rightarrow\beta)$, *then* $\models \big(\bigcup Ma\Rightarrow\bigcup M\beta\big)$ *and* $\models \big(\bigcap Ma\Rightarrow\bigcap M\beta\big)$;

(iv) *If for every natural number* $i\in\mathcal{N}$, $\models \big(a\Rightarrow(M(K^i\beta))\big)$, *then* $\models \big(a\Rightarrow(M\bigcap K\beta)\big)$;

(v) *If for every natural number* $i\in\mathcal{N}$, $\models \big((M(K^i a))\Rightarrow\beta\big)$, *then* $\models \big((M\bigcup Ka)\Rightarrow\beta\big)$.

Proof: (i) By assumption, for every realization R of the language $\mathscr{L}$ and for every valuation v, $a_R(v)=\mathbf{1}$ and $(K\mathbf{1})_R(v)=\mathbf{1}$. So, for every realization R and every valuation v, $K_R(v)$ is defined and $(Ka)_R(v)=a_R\big(K_R(v)\big)=\mathbf{1}$. Consequently $\models (Ka)$.

(ii) Let R be a realization of the language $\mathscr{L}$ and v a valuation.

Let us consider two cases:

1. $K_R(v)$ is defined; then $\big((Ka)\Rightarrow(K\beta)\big)_R(v)=(a\Rightarrow\beta)_R\big(K_R(v)\big)=\mathbf{1}$.

2. $K_R(v)$ is undefined; then $(Ka)_R(v)=(K\beta)_R(v)=\mathbf{0}$ and $\big((Ka)\Rightarrow(K\beta)\big)_R(v)=\mathbf{1}$.

In both cases $\big((Ka)\Rightarrow(K\beta)\big)_R(v)=\mathbf{1}$, and so $\models\big((Ka)\Rightarrow(K\beta)\big)$.

(iii) If $\big(\bigcup Ma\big)_R(v)=\mathbf{1}$, then there exists a natural number $i\in\mathcal{N}$ such that $(M^i a)_R(v)=\mathbf{1}$; but then $(M^i\beta)_R(v)=\mathbf{1}$. So $\big(\bigcup M\beta\big)_R(v)=\mathbf{1}$. If $\big(\bigcap Ma\big)_R(v)=\mathbf{1}$, then for every natural number $i\in\mathcal{N}$, $(M^i a)_R(v)=\mathbf{1}$. By (ii), for every $i\in\mathcal{N}$ we have $(M^i\beta)_R(v)=\mathbf{1}$. So $\big(\bigcap M\beta\big)_R(v)=\mathbf{1}$.

(iv) By assumption, for every $i\in\mathcal{N}$ and for every realization R and valuation v we have the inequality $\big(M(K^i a)\big)_R(v)\leqslant\beta_R(v)$ so that l.u.b. $\big((M(K^i a))_R(v)\big)_{i\in\mathcal{N}}\leqslant\beta_R(v)$ and consequently $(M\bigcup Ka)_R(v)\leqslant\beta_R(v)$. Therefore $\models\big((M\bigcup Ka)\Rightarrow\beta\big)$. ■

The above lemma can be formulated more generally in the following way:

LEMMA 5. *Let Z be any set of formulas, $Z\subset FSF$; then for every formulas a,β and every programs K, M*

(i) *if* $Z\models a$ *and* $Z\models (K\mathbf{1})$, *then* $Z\models (Ka)$;

(ii) *if* $Z\models (a\Rightarrow\beta)$, *then* $Z\models (Ka\Rightarrow K\beta)$;

(iii) *if* $Z\models (a\Rightarrow\beta)$, *then* $Z\models \big(\bigcup Ka\Rightarrow\bigcup K\beta\big)$ *and* $Z\models\big(\bigcap Ka\Rightarrow\bigcap K\beta\big)$;

(iv) *if for every* $i\in\mathcal{N}$, $Z\models\big((M(K^i a))\Rightarrow\beta\big)$, *then* $Z\models\big((M\bigcup Ka)\Rightarrow\beta\big)$;

(v) *if for every* $i\in\mathcal{N}_0$, $Z\models\big(a\Rightarrow(M(K^i\beta))\big)$, *then* $Z\models\big(a\Rightarrow(M\bigcap K\beta)\big)$. ■

5. A formalized consequence operation

Let $\mathscr{L}$ be a formalized algorithmic language as in §1. By α, β, γ we shall denote any formulas in FSF, δ — an open classical formula, S — a substitution, K, M — any programs and τ_i — any term. The formula $\alpha \Leftrightarrow \beta$ will be used as an abbreviation for $\big((\alpha \Rightarrow \beta) \cap (\beta \Rightarrow \alpha)\big)$.

By an *axiom of algorithmic logic* we shall understand any formula of one of the following forms:

T1. $\big((\alpha \Rightarrow \beta) \Rightarrow ((\beta \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow \gamma))\big)$,

T2. $(\alpha \Rightarrow (\alpha \cup \beta))$,

T3. $(\beta \Rightarrow (\alpha \cup \beta))$,

T4. $\big((\alpha \Rightarrow \gamma) \Rightarrow \big((\beta \Rightarrow \gamma) \Rightarrow ((\alpha \cup \beta) \Rightarrow \gamma)\big)\big)$,

T5. $((\alpha \cap \beta) \Rightarrow \beta)$,

T6. $((\alpha \cap \beta) \Rightarrow \alpha)$,

T7. $\big((\gamma \Rightarrow \alpha) \Rightarrow \big((\gamma \Rightarrow \beta) \Rightarrow (\gamma \Rightarrow (\alpha \cap \beta))\big)\big)$,

T8. $\big((\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \cap \beta) \Rightarrow \gamma)\big)$,

T9. $\big(((\alpha \cap \beta) \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow (\beta \Rightarrow \gamma))\big)$,

T10. $((\alpha \cap \neg \alpha) \Rightarrow \beta)$,

T11. $((\alpha \Rightarrow (\alpha \cap \neg \alpha)) \Rightarrow \neg \alpha)$,

T12. $(\alpha \cup \neg \alpha)$,

T13. $\neg(\mathbf{1} \Rightarrow \mathbf{0})$,

T14. $((s\alpha) \Leftrightarrow \overline{s\alpha})$, for every open formula α,

T15. $\big((K\varrho(\tau_1, \ldots, \tau_n)) \Leftrightarrow \varrho((K\tau_1), \ldots, (K\tau_n))\big)$, } for every n-argument

T16. $\big(\varrho(\tau_1, \ldots, \tau_n) \Leftrightarrow \chi(\varrho(\tau_1, \ldots, \tau_n))\big)$, } predicate $\varrho \in P_n$,

T17. $\big((K(\alpha \cup \beta)) \Leftrightarrow ((K\alpha) \cup (K\beta))\big)$,

T18. $\big((K(\alpha \cap \beta)) \Leftrightarrow ((K\alpha) \cap (K\beta))\big)$,

T19. $((K \neg \alpha) \Rightarrow (\neg K\alpha))$,

T20. $\big((K\mathbf{1}) \Rightarrow ((\neg K\alpha) \Rightarrow (K \neg \alpha))\big)$,

T21. $\big((K(\alpha \Rightarrow \beta)) \Rightarrow ((K\alpha) \Rightarrow (K\beta))\big)$,

T22. $\big((K\mathbf{1}) \Rightarrow \big(((K\alpha) \Rightarrow (K\beta)) \Rightarrow (K(\alpha \Rightarrow \beta))\big)\big)$,

T23. $\big((M \bigcup K\alpha) \Leftrightarrow \big((M\alpha) \cup (M \bigcup K(K\alpha))\big)\big)$,

T24. $\big((M \bigcap K\alpha) \Leftrightarrow \big((M\alpha) \cap (M \bigcap K(K\alpha))\big)\big)$,

T25. $((\circ[KM]\alpha) \Leftrightarrow (K(M\alpha)))$,

T26. $\big((\veebar[\delta KM]\alpha) \Leftrightarrow \big((\delta \cap (K\alpha)) \cup (\neg \delta \cap (M\alpha))\big)\big)$,

T27. $((*[\delta K]\alpha) \Leftrightarrow \bigcup \veebar[\delta K[\ ]] \neg(\delta \cap \alpha))$.

We shall admit four rules of inference:

r1. $\dfrac{\alpha, (\alpha \Rightarrow \beta)}{\beta}$;

r2. $\dfrac{\alpha, (K1)}{(K\alpha)}$;

r3. $\dfrac{\{(\gamma \Rightarrow (M(K^i \alpha)))\}_{i \in \mathcal{N}}}{(\gamma \Rightarrow (M \bigcap K\alpha))}$;

r4. $\dfrac{\{((M(K^i \alpha)) \Rightarrow \gamma)\}_{i \in \mathcal{N}}}{((M \bigcup K\alpha) \Rightarrow \gamma)}$.

The formulas over line in rules r1, r2, r3, r4 are called premises and the formulas underneath — conclusions.

DEFINITION 1. The *consequence operation* C is the mapping which to every set X of formulas in $\mathcal{L}$, $X \subset FSF$, assigns the least set $C(X)$ of formulas satisfying (a), (b) and closed under the rules r1–r4 (i.e., if premises of a fixed rule of inference belong to $C(X)$, then the conclusion belongs to $C(X)$):

(a) $X \subset C(X)$,

(b) all axioms are in $C(X)$.

DEFINITION 2. A deductive system $\langle \mathcal{L}, C \rangle$ will be called an *algorithmic logic*, and a system $\langle \mathcal{L}, C, \mathcal{A} \rangle$, where $\mathcal{A} \subset FSF$, *an algorithmic theory.*

As one could expect, the process of deducing is more complicated than in classical logic, on account of the rules r3, r4. For that reason the notion of formal proof in algorithmic logic has been changed as follows.

DEFINITION 3. By a *tree* we shall mean a set $\mathcal{D}$ of finite sequences of natural numbers such that if any sequence $c = (i_1, \ldots, i_n)$ is an element of $\mathcal{D}$, then every initial segment c_k of c, $c_k = (i_1, \ldots, i_k)$, is also an element of $\mathcal{D}$. The empty sequence denoted by $\emptyset$ belongs to every tree.

For any $c = (i_1, \ldots, i_n)$, the number n is called the *level of the element c in the tree $\mathcal{D}$*. By a *level of the tree $\mathcal{D}$* we shall mean the set of all elements that have the same level.

A subset of $\mathcal{D}$ such that its elements are linearly ordered with respect to the relation "to be an initial segment" is called a *branch of the tree $\mathcal{D}$*.

The tree is *finite* if all its branches are finite sets.

DEFINITION 4. By a *proof of a formula α from the set X* we shall understand the ordered pair $(\mathcal{D}, d)$, where $\mathcal{D}$ is a finite tree and d is a mapping, $d\colon \mathcal{D} \to FSF$ which to every $c \in \mathcal{D}$ assigns a formula $d(c)$ defined in the following way:

1. $d(c)$ is an axiom or $d(c) \in X$ for every maximal element c in $\mathcal{D}$;
2. for any other element $c = (i_1, \ldots, i_n)$, $d(c)$ is a conclusion in a rule of inference from all formulas $d(i_1, \ldots, i_n, j)$ such that $(i_1, \ldots, i_n, j) \in \mathcal{D}$.

By a *theorem in the theory* $\mathcal{T} = \langle \mathcal{L}, C, \mathcal{A} \rangle$ we shall understand any formula that has a proof from the set of specific axioms $\mathcal{A}$.

LEMMA 1. *For every formula* α, α *is a theorem in* $\langle \mathcal{L}, C, \mathcal{A} \rangle$ *if and only if* $\alpha \in C(\mathcal{A})$. ■

LEMMA 2. *If* α *is a theorem in algorithmic logic, then* α *is a tautology*, i.e. $C(\emptyset) \subset \mathrm{Cn}(\emptyset)$. ■

The proofs immediately follow from lemmas 3.4, 3.7, 4.4.

The second part of this paper will be found in the next issue of this journal.

References

[1] Banachowski, L., *Data structures*, Reports of IMMUW 46.

[2] —, *Investigations of properties of programs by means of the extended algorithmic logic*, doctoral dissertation, Warsaw University 1975.

[3] Kreczmar, A., *The set of all tautologies of algorithmic logic is hyperarithmetical*, Bull. Acad. Polon. Sci., Sér. Math. 21 (1971), 781–783.

[4] Kuratowski, K., and Mostowski, A., *Set theory*, PWN, Warsaw 1966.

[5] Mirkowska, G., *On formalized systems of algorithmic logic*, Bull. Acad. Polon. Sci. Sér. Math. 19 (1971), 421–428.

[6] —, *Algorithmic logic and its applications in the theory of programs*, doctoral dissertation, Warsaw University 1972 (in Polish).

[7] —, *Herbrand theorem in the algorithmic logic*, Bull. Acad. Polon. Sci., Sér. Math. 22 (1974), 539–543.

[8] Rasiowa, H., *On logical structure of programs*, ibid. 20 (1972), 319–324.

[9] —, *Extended* ω^+*-valued algorithmic logic*, ibid. 22 (1974), 605–610.

[10] Rasiowa, H., and Sikorski, R., *Mathematics of metamathematics*, PWN, Warsaw 1963.

[11] Salwicki, A., *Formalized algorithmic languages*, Bull. Acad. Pol. Sci., Sér. Math. 18 (1970), 227–232.

[12] —, *Programmability and recursiveness*, to appear.