Theoretical Computer Science 135 (1994) 267–288
Elsevier

An algebraic approach to semantics of programming languages

Alexey L. Lastovetsky

Computer Centre for Scientific Research, Moscow State University, Leninskiye Gory, Moscow, 119899, Russian Federation

Sergey S. Gaissaryan

Institute for System Programming, Russian Academy of Sciences, 25 Bolshaya Kommunisticheskaya Street, Moscow, 109004, Russian Federation

Communicated by M. Nivat
Received May 1993
Revised January 1994

Abstract

Lastovetsky, A.L. and S.S. Gaissaryan, An algebraic approach to semantics of programming languages, Theoretical Computer Science 135 (1994) 267–288.

An abstract language for a computer of von Neumann type is presented. This language is considered not only as a programming language, but as an algebraic one, whose semantics is defined by methods of model theory. Calculus of equivalencies of the abstract programs and techniques for solving equations within limits of this calculus are presented. An algebraic technique is described which allows to define the propositional semantics of programs. To construct such techniques it was necessary to use the data type representation by continuous lattices and the continuity of type and intertype operations and elementary relations. It is demonstrated how the proposed algebraic technique may be used.

1. Introduction

An abstract language for a computer of von Neumann type was constructed in [7, 8]. This language can be treated not only as a programming language, but as an algebraic one whose semantics was determined by methods from model theory.

The type system of this language is represented by a multisorted algebraic system S in which different sorts of objects correspond to different data types. The operations and elementary relations of system S are functional abstractions of effective procedures

Correspondence to: S.S. Gaissaryan, Institute for System Programming, Russian Academy of Sciences, 25 Bolshaya Kommunisticheskaya Street, Moscow, 109004, Russian Federation. Email addresses of the authors: lastov@ivann.delta.msk.su and ssg@vann.delta.msk.su.

0304-3975/94/$07.00 © 1994—Elsevier Science B.V. All rights reserved
SSDI 0304-3975(94)00022-B

for transforming data. The set M_S of objects in this system contains three nonproper objects $\omega_S, \Theta, \Omega_S$. These objects are separated from proper ones by providing M_S with a complete lattice structure $\leqslant_S$ such that $\omega_S <_S \Theta <_S m <_S \Omega_S$ for all proper $m \in M_S$. The object ω_S is treated as the value of an abnormal terminating computation, Ω_S is treated as the value of a nonterminating computation, while Θ is treated as the value of a noninitialized variable. Such interpretation requires the monotony of operations and elementary relations with respect to $\leqslant_S$ and forbids Θ to be a result of any operations.

The memory state space is simulated by the cartesian product $\prod_{i=1}^{N} M_{T_i}$ of lattices of typed objects comprising memory cells (T_i is the type of the object in ith memory cell). The lattice thus obtained (with partial ordering denoted by $\leqslant_\Pi$) is complete, since it is the cartesian product of a finite number of complete lattices [8]. However, this lattice contains elements that cannot be distinguished from the informal point of view: an element of the form $\langle \ldots, \Omega_S, \ldots \rangle \in \prod_{i=1}^{N} (M_{T_i} \setminus \{\omega_S\})$ represents a terminal memory state of a nonterminating program, while $\langle, \ldots, \omega_S, \ldots \rangle \in \prod_{i=1}^{N} M_{T_i}$ represents a terminal memory state for any abnormally terminated program. We therefore construct an equivalence relation $\equiv$ in $\prod_{i=1}^{N} M_{T_i}$ by identifying indistinguishable elements:

$$\langle m_1, \ldots, m_N \rangle \equiv \langle \hat{m}_1, \ldots, \hat{m}_N \rangle$$

if and only if

$$(\forall i \leqslant N\colon\ m_i =_S \hat{m}_i) \vee (\exists i,j \leqslant N\colon m_i =_S \hat{m}_j =_S \omega_S)$$
$$\vee ((\forall i \leqslant N\colon\ m_i \neq_S \omega_S \,\&\, \hat{m}_i \neq_S \omega_S) \,\&\, (\exists i,j \leqslant N\colon m_i =_S \hat{m}_j =_S \Omega_S)) \ .$$

We denote by M the set of factor sets of $\prod_{i=1}^{N} M_{T_i}$ obtained from the relation $\equiv$, and by $[m]$ we denote the factor set containing the element $m \in \prod_{i=1}^{N} M_{T_i}$.

We set $[m] \leqslant [m']$ if and only if $m \leqslant_\Pi m'$ or $m \equiv m'$. The lattice $\langle M, \leqslant \rangle$ is a complete lattice (by the completeness of $\prod_{i=1}^{N} M_{T_i}$) that contains no indistinguishable or meaningless elements. We therefore use it for the universe of memory states. We denote by ω and Ω the zero and identity of the lattice M, respectively.

The syntax of the language [7, 8] contains three sorts of expressions: logical, assignment, and program expressions.

The set **RP** of logical expressions is defined [7] to contain atomic formulas of **S** whose notation does not use $\leqslant_S$ or symbols for operations that associate Ω_S with proper operands, and if $A, B \in$ **RP**, then $A \& B, A \vee B, \neg A \in$ **RP**. This definition ensures decidability for logical expressions treated as branch conditions in a computation.

The set of assignment expressions of type T includes terms of type T in the language of **S**. Intuitively, an assignment expression is treated as right part of the assignment statement.

The set **PP** of program expressions is defined so that $\phi, \lambda, \infty \in$ **PP**, where λ has interpretation of the null program (the "doing-nothing" program), and ϕ is a program that terminates abnormally for any initial proper memory state, while ∞ is a program that does not terminate for any initial proper memory state.

If a variable x_i in the language of system **S** and an assignment expression t_i $(i=1,\ldots,k)$ are of the same type, then $[x_1:t_1,\ldots,x_k:t_k]\in$ **PP** and is taken to mean that the cells named $x_1,\ldots,x_k$ are simultaneously assigned the values of the expressions $t_1,\ldots,t_k$, respectively.

If $p,q\in$ **PP**, $A\in$ **RP**, then $A\rightarrow p, p\circ q, p+q, p^*\in$ **PP** and have the following interpretation:

- $A\rightarrow p$ indicates execution of program p if A is true, and abnormal termination otherwise;
- $p\circ q$ denotes successive execution of programs p and q;
- $p+q$ denotes execution of p if q abnormally terminates, execution of q if p abnormally terminates, or nontermination in the case of nontermination of p or q. In the case of the normal termination of p and q we admit different interpretations: the execution of one of the programs p or q (for example, of that one which takes more time of processor), or a parallel execution of programs p and q;
- p^* denotes iterated execution of p in which p is repeatedly executed until an attempt does not fail. The result of a loop is the result of the last successfully terminated execution of p. If the first attempt at execution leads to failure, then p^* indicates execution of the null program λ.

The operation $*$ has the highest priority. It is followed by the operation $\rightarrow$. The operation $+$ has the lowest priority. So, $A\rightarrow p^*+q\circ r$ is equal to $(A\rightarrow(p^*))+(q\circ r)$.

The mathematical semantics of logical expressions are given by relations on the memory state space with universe M_{RP}. M_{RP} consists of all monotonic mappings $M\rightarrow B$ (here B is the two-element Boolean lattice $\langle\{\mathrm{f},\mathrm{t}\},\Rightarrow\rangle$), which map Ω_S onto t and ω_S onto f. The set M_{RP} with partial ordering $\Rightarrow$ induced by the partial ordering of the Boolean lattice B forms a Boolean lattice, whose zero and identity we denote by F and T, respectively. The algebraic operations of union $\vee$, intersection &, complementation $'$, and equality $\sim$, which make M_{RP} a Boolean algebra, are defined as usual. The interpretational mapping ξ: **RP** $\rightarrow M_{\mathrm{RP}}$ is defined so that

$$\xi[A\&B] \sim \xi[A]\&\xi[B], \qquad \xi[A\vee B] \sim \xi[A]\vee\xi[B].$$

If $A\in$ **RP** is an atomic formula in the language of **S**, then in this system there exists a corresponding mapping of the form $\prod_{j=1}^{n} M_{T'_j}\rightarrow B$, whose monotonic continuation to M gives the interpretation $\xi[A]\in M_{\mathrm{RP}}$ for the logical expression A.

If $\langle m_1,\ldots,m_N\rangle\in M\backslash\{\omega,\Omega\}$, then

$$\xi[\neg A](m_1,\ldots,m_N) \sim$$

$$\textbf{if } \exists j\leqslant N\text{: } (m_j =_S \Theta\,\&(\forall\langle\hat m_1,\ldots,\hat m_{j-1},\Theta,\ldots,\hat m_N\rangle\in M\backslash\{\omega,\Omega\}:$$

$$\xi[\neg A](\hat m_1,\ldots,\hat m_{j-1},\Theta,\ldots,\hat m_N) \sim \mathrm{f}))$$

then f
else $\xi[A'](m_1,\ldots,m_N)$.

As result of these semantics, the operation $\neg$ assures that branch conditions will evaluate to false if their evaluation requires the use of a variable that has been assigned the undefined value Θ.

An equivalence calculus for $\sim$ was constructed in [8] so that for any $A, B, C \in \mathbf{RP}$, as well as $D(x) \in \mathbf{RP}$ with isolated appearance of x, whose construction does not use symbols $\vee$ and Ω_S, the following formulas are axioms:

$$A \,\&\, A \sim A \vee A \sim A,$$

$$A \,\&\, B \sim B \,\&\, A,$$

$$A \vee B \sim B \vee A,$$

$$(A \,\&\, B) \,\&\, C \sim A \,\&\, (B \,\&\, C),$$

$$(A \vee B) \vee C \sim A \vee (B \vee C),$$

$$A \,\&\, (B \vee C) \sim (A \,\&\, B) \vee (A \,\&\, C),$$

$$A \,\&\, (A \vee B) \sim A \vee (A \,\&\, B) \sim A,$$

$$\mathrm{F} \,\&\, A \sim \mathrm{F},$$

$$\mathrm{T} \,\&\, A \sim A,$$

$$\mathrm{T} \vee A \sim \mathrm{T},$$

$$\mathrm{F} \vee A \sim A,$$

$$A \,\&\, \neg A \sim \mathrm{F},$$

$$\neg(\neg A) \sim A,$$

$$D(\Theta) \sim \mathrm{F}.$$

The mathematical semantics $\psi[p]$ of p is given by a pair of mappings $\langle \psi_1[p] : M \to M, \psi_2[p] : M \to M_D^N \rangle$, which we call a procedure. Here, $\psi_1[p]$ provides the functional semantics of p, which associate a terminal memory state with every initial state, while $\psi_2[p]$ provides the operational semantics of p, which associate with each initial memory state a series of computations (elements of M_D^N) that lead to the terminal state. The universe M_{PP} of procedures is organized so that the set of first components of the procedures is comprised of all monotonic maps $M \to M$ that leave elements ω_S and Ω_S stationary. On M_{PP} we define the operations $+$ and $\cdot$ so that $M_{\mathrm{PP}} \times M_{\mathrm{PP}} \to M_{\mathrm{PP}}$, the operation $*$ so that $M_{\mathrm{PP}} \to M_{\mathrm{PP}}$, and the operation $\to$ so that $M_{\mathrm{RP}} \times M_{\mathrm{PP}} \to M_{\mathrm{PP}}$. These operations provide mathematical abstractions of the control structures listed above. Besides, the relation of strong $(=)$ and weak $(\approx)$ equivalence for procedures are defined. Informally, two procedures are said to be weakly equivalent if for any given initial memory state both procedures either normally terminate with the same memory state, or they both abnormally terminate or fail to terminate. Two weakly equivalent procedures are said to be strongly equivalent if for any given initial memory state they both (in the case of the normal termination) execute the same computations.

2. Calculus of strong and weak equivalencies

The calculus of strong and weak equivalencies and techniques for solving equations within the limits of this calculus are constructed in [8]. In particular, the following well-formed formulas

(SA1) $p\circ(q\circ r)=(p\circ q)\circ r,$
(SA2) $\lambda\circ p=p\circ\lambda=p,$
(SA3) $\phi\circ p=\phi,$
(SA4) $\infty\circ p=\infty,$
(SA5) $p\circ\phi=\phi\vee p=\infty,$
(SA6) $p+q=q+p,$
(SA7) $p+(q+r)=(p+q)+r,$
(SA8) $\phi+p=p,$
(SA9) $\infty+p=\infty,$
(SA10) $\lambda+p=p\vee p=\phi,$
(SA11) $r\circ A\to p+r\circ\neg A\to q=r\circ A\to p$
$\vee\, r\circ A\to p+r\circ\neg A\to q=r\circ\neg A\to q,$
(SA12) $r\circ(p+q)=r\circ p+r\circ q,$
(SA13) $(p+q)\circ\phi=p\circ\phi+q\circ\phi,$
(SA14) $(p+q)\circ\infty=p\circ\infty+q\circ\infty,$
(SA15) $\lambda+p\circ p^*=p^*,$
(SA16) $\mathrm{F}\to p=\phi,$
(SA17) $\mathrm{T}\to p=p,$
(SA18) $(A\vee B)\to p=A\to p+B\to p,$
(SA19) $(A\&B)\to p=A\to(B\to p),$
(SA20) $A\to(p+q)=A\to p+A\to q,$
(SA21) $A\to(p\circ q)=A\to p\circ q$

are axioms for any $p,q,r\in\mathbf{PP}$ and $A,B\in\mathbf{RP}$. Here a formula $F\vee G$ is valid iff for every $m\in M$ either F or G is valid.

Besides, if all occurrences of x in formula $F(x)$ and in program expression $t(x)$ are marked then the following rules of inference

(I1) $$\frac{\lambda+p=p}{p^*=\infty\ \vee\ \lambda=p}$$

(I2) $$\frac{\lambda+r=r,\quad r\circ\phi=\phi}{(p+q)\circ r=p\circ r+q\circ r}$$

(I3) $$\frac{\forall n\geqslant 0\colon\ F(\lambda+p+\cdots+p^n),\quad F(\infty)}{F(p^*)}$$

(I4) $$\frac{\forall n\geqslant 0\colon\ F(\lambda+p+\cdots+p^n),\quad p^*\circ\phi=\phi}{F(p^*)}$$

(I5) $$\frac{\forall n\geqslant 0\colon\ \tau(\lambda+p+\cdots+p^n)=r,\quad \tau(\infty)=\infty,\quad p^*=\infty}{r=\infty}$$

(16) $$\frac{A \to (\lambda + p) = A \to p}{(A \to p)^* \circ \neg A \to \lambda = (A \to p)^* \ \vee \ A \to p = A \to \lambda}$$

(17) $$\frac{p = q \vdash \lambda = \phi \ \vee \ \lambda = \infty \ \vee \ \phi = \infty}{p \neq q}$$

(18) $$\frac{G \ \vee \ p = q, \quad p \neq q}{G}$$

(19) $$\frac{G \vee H, \ \ G \vdash J, \ \ H \vdash J}{J}$$

are valid for every $p, q, r \in \mathbf{PP}$, $A \in \mathbf{RP}$ and for every well-formed formulas G, H, J. Here $F \vdash G$ means that G can be deduced from F for fixed initial memory state. Formally, such inference forbids to use the steps $p = q \vdash p^* = q^*$ and $p = q \vdash r \circ p = r \circ q$ iff $F \vdash p = q$. The semantics of relation $\neq$ is as follows: $p \neq q$ iff $p = q$ is invalid for all proper $m \in M$.

Formulas of the calculus allow to express many important operational properties of programs. For example, the formula $\lambda + p = p$ means that the program p cannot abnormally terminate. The formula $A \to (\lambda + p) = A \to p$ means that the program p cannot abnormally terminate if initial memory state satisfies the condition A. The formula $p \circ \phi = \phi$ means that the program p always terminates. The formula $A \to p \circ \phi = \phi$ means that the program p terminates if initial memory state satisfies the condition A. The formula $A \to p = \phi$ means that the program p abnormally terminates if initial memory state satisfies the condition A. Correspondingly, the formula $A \to p = \infty$ means that the program p does not terminate if initial memory state satisfies the condition A.

Any iterative program can be expressed easy in limits of the calculus.

Example 1. Consider the Fortran program

```
10 FORMAT (I10)
11 FORMAT (1X, I10)
   INTEGER X, Y
   READ 10, X
 1 IF (X.GT.Y) GO TO 4
 2 IF (X.LT.Y) GO TO 6
   PRINT 11, Y
 3 STOP
 4 X=X-Y
 5 GO TO 1
 6 Y=Y-X
 7 GO TO 1
   END
```

Semantics of the program are given by the following system of equations:

$$\alpha = [Y : \Theta] \circ \alpha 1,$$

$$\alpha 1 = X > Y \to \alpha 4 + \neg(X > Y) \to \alpha 2,$$

$$\alpha 2 = X < Y \to \alpha 6 + \neg(X < Y) \to \alpha 3\,,$$

$$\alpha 3 = \lambda,$$

$$\alpha 4 = [X : X - Y] \circ \alpha 5,$$

$$\alpha 5 = \alpha 1,$$

$$\alpha 6 = [Y : Y - X] \circ \alpha 7,$$

$$\alpha 7 = \alpha 1.$$

Here α denotes a procedure corresponding to the entire program as a whole, and $\alpha 1, \alpha 2, \alpha 3, \alpha 4, \alpha 5, \alpha 6, \alpha 7$ denote procedures corresponding to program entries with the labels 1, 2, 3, 4, 5, 6, and 7. This Fortran program, which implements the Euclidean algorithm of finding the greatest common divisor of two numbers, has a semantic error (the variable Y has not been initialized) which in machine testing may not be detected. We shall now demonstrate how this error is revealed by means of the proposed formal tools. We have $\alpha = [Y : \Theta] \circ \alpha 1 = [Y : \Theta] \circ (X > Y \to \alpha 4 + \neg(X > Y) \to \alpha 2) = [Y : \Theta] \circ X > Y \to \alpha 4 + [Y : \Theta] \circ \neg(X > Y \to \alpha 2 = X > \Theta \to \alpha 4 + \neg(X > \Theta) \to \alpha 2 = \mathrm{F} \to \alpha 4 + \mathrm{F} \to \alpha 2 = \phi + \phi = \phi$.

Note that since the operation $+$ is an abstract generalization of selection statements, some program expressions of the calculus may not have direct images in corresponding programming language (for example, the expression $[X : X - Y] + [Y : Y - X]$ for Fortran 77). On the other hand, since the operation $+$ is a total operation having nice algebraic properties, one can make many difficult equivalent transformations of programs easy enough. It is not important if some intermediate expressions have no direct images in given programming languages, because we need some image only for the final expression of transformations.

Lemma 1. *The formula* $(A \to p + \neg A \to q) \circ r = A \to p \circ r + \neg A \to q \circ r$ *is valid for any* $p, q, r \in \mathbf{PP}$, $A \in \mathbf{RP}$.

Proof. According to (SA11) we have $A \to p + \neg A \to q = A \to p \vee A \to p + \neg A \to q = \neg A \to q$. If $A \to p + \neg A \to q = A \to p$, then $\neg A \to (A \to p + \neg A \to q) = \neg A \to (A \to p)$, whence $\neg A \to q = \phi$. Consequently, $(\neg A \to q) \circ r = \phi$ and $(A \to p + \neg A \to q) \circ r = (A \to p) \circ r = A \to p \circ r + \neg A \to q \circ r$. Similarly, assuming $A \to p + \neg A \to q = \neg A \to q$ we derive $(A \to p + \neg A \to q) \circ r = (\neg A \to q) \circ r = A \to p \circ r + \neg A \to q \circ r$. Thus, according to rule (19) we conclude that $(A \to p + \neg A \to q) \circ r = A \to p \circ r + \neg A \to q \circ r$. $\square$

Lemma 2. *The formula* $(\lambda + A \to q) \circ \neg A \to r = \neg A \to r + A \to q \circ \neg A \to r$ *is valid for any* $q, r \in \mathbf{PP}$, $A \in \mathbf{RP}$.

Proof. According to (SA10) we have $\lambda + A \to q = A \to q \vee A \to q = \phi$. If $\lambda + A \to q = A \to q$, then $\neg A \to \lambda + \neg A \to A \to q = \neg A \to A \to q$, that is $\neg A \to \lambda = \phi$.

Consequently, $(\lambda + A\to q \circ \neg A\to r = A\to q \circ \neg A\to r = A\to q \circ \neg A\to r + \phi \circ r = A\to q \circ \neg A\to r + (\neg A\to\lambda)\circ r = \neg A\to r + A\to q \circ \neg A\to r$. If $A\to q=\phi$, then $(\lambda + A\to q) \circ \neg A\to r = \neg A\to r = \neg A\to r + \phi \circ \neg A\to r = \neg A\to r + A\to q \circ \neg A\to r$. Thus, according to rule (I9) we conclude that $(\lambda + A\to q) \circ \neg A\to r = \neg A\to r + A\to q \quad \neg A\to r$. □

Semantics of an iterative program are given by a system of equations of the calculus. A solution of the system gives a structured form of the initial program more suitable for an analysis.

To solve equations of the form $\alpha = t(\alpha)$, where α denotes an unknown program expression, on the set **PP** let us introduce the partial ordering $\leqslant$ ($q \leqslant p$ iff $q+p=p$) which makes **PP** an upper semilattice with zero ϕ and unity ∞. This semilattice is not complete semilattice but the following theorem is valid.

Theorem 1. *Let the program expression* $\rho(a,b)$ *is monotonic with respect to* b, $q=r \vdash \rho(q,b)=\rho(r,b)$ *and* $\rho(\infty,b)=\infty$. *Let us denote* $\tau(a)=\rho(a,a)$. *Then if sequence* $\{\tau(\lambda+p+\ldots+p^n)\}$ $(n=0,1,\ldots)$ *is an increasing chain then it has a least upper bound* (lub), *defined as follows*:

$$\text{lub}\,\{\tau(\lambda+p+\cdots+p^n)\}=\tau(p^*)\ .$$

Note: The program expression $t(a,b)$ is said to be monotonic with respect to a if from $p\leqslant q$ it follows that $t(p,r)\leqslant t(q,r)$ for any $p,q,r \in$ **PP**.

Proof. Let us fix an arbitrary $m \geqslant 0$. Then for any $n \geqslant 0$ we have

$$\tau(\lambda+\cdots+p^m)\leqslant\tau(\lambda+\cdots+p^m\circ(\lambda+\cdots+p^n))$$

since $\{\tau(\lambda+\cdots+p^n)\}$ $(n=0,1,\ldots)$ forms a chain. Let us show that

$$\tau(\lambda+\cdots+p^m)\leqslant\tau(\lambda+\cdots+p^m\circ\infty)\,.$$

According to (SA5), $(p^m\circ\infty)\circ\phi=\phi \;\vee\; p^m\circ\infty=\infty$ whence $p^m\circ\infty=\phi \;\vee\; p^m\circ\infty=\infty$. If $p^m\circ\infty=\infty$ then $\rho(\lambda+\cdots+p^m\circ\infty,\beta)=\rho(\infty,\beta)=\infty$. Thus,

$$\rho(\lambda+\cdots+p^m,\beta)\leqslant\rho(\lambda+\cdots+p^m\circ\infty,\beta)\quad \text{if } p^m\circ\infty=\infty\ .$$

If $p^m\circ\infty=\phi$ then $p^m=\phi$, so that

$$\rho(\lambda+\cdots+p^m,\beta)=\rho(\lambda+\cdots+p^{m-1},\beta)=\rho(\lambda+\cdots+p^{m-1}+p^m\circ\infty,\beta)\ .$$

Thus, also in this case $\rho(\lambda+\cdots+p^m,\beta)\leqslant\rho(\lambda+\cdots+p^m\circ\infty,\beta)$. By virtue of monotony of the expression $\rho(a,b)$ with respect to b, we have

$$\rho(\lambda+\cdots+p^m\circ\infty,\lambda+\cdots+p^m)\leqslant\rho(\lambda+\cdots+p^m\circ\infty,\lambda+\cdots+p^m\circ\infty)\ .$$

Whence, finally,

$$\begin{aligned}\tau(\lambda+\cdots+p^m)&=\rho(\lambda+\cdots+p^m,\lambda+\cdots+p^m)\\&\leqslant\rho(\lambda+\cdots+p^m\circ\infty,\lambda+\cdots+p^m)\\&\leqslant\rho(\lambda+\cdots+p^m\circ\infty,\lambda+\cdots+p^m\circ\infty)\\&=\tau(\lambda+\cdots+p^m\circ\infty).\end{aligned}$$

Applying the rule (I3), we get

$$\tau(\lambda+\cdots+p^m)+\tau(\lambda+\cdots+p^m\circ p^*)=\tau(\lambda+\cdots+p^m\circ p^*).$$

Since $\lambda+\cdots+p^m\circ p^*=p^*$ for any finite $m\geqslant 0$, then $\tau(\lambda+\cdots+p^m)\leqslant\tau(p^*)$, meaning that $\tau(p^*)$ is an upper bound of the sequence $\{\tau(\lambda+\cdots+p^n)\}$. Let us now assume that q also is an upper bound of this sequence. This means that $\tau(\lambda+\cdots+p^n)+q=q$ for any $n\geqslant 0$. From axiom (SA5) follows that $p^*\circ\phi=\phi \ \vee \ p^*=\infty$. If $p^*=\infty$, then using $\tau(\infty)=\infty$, by rule (I5) we get $q=\infty$, whence $\tau(\infty)+q=q$. Thus, if $p^*=\infty$, then $\tau(p^*)+q=q$ according to (I3). If $p^*\circ\phi=\phi$, then according to (I4) we have $\tau(p^*)+q=q$. Applying rule (I9), we derive $\tau(p^*)\leqslant q$. This means that $\tau(p^*)=\text{lub}\{\tau(\lambda+p+\cdots+p^n)\}$ $(n=0,1,\ldots)$. □

Corollary 1. *If sequence* $\{\tau(\lambda+p+\cdots+p^n)\}$ $(n=0,1,\ldots)$ *of program expressions is an increasing chain and* $p^*\circ\phi=\phi$ *then*

$$\tau(p^*)=\text{lub}\{\tau(\lambda+p+\cdots+p^n)\}.$$

Theorem 2. *If* $t^{(n)}(\phi)\leqslant t^{(n)}(p)$ $(t^{(n)}(p)\leqslant t^{(n)}(\infty))$ *is valid for all* $p\in$ **PP** *and for all* $n>0$ *then any solution of the equation*

$$\alpha=t(\alpha) \tag{1}$$

is an upper bound of $\{t^{(i)}(\phi)\}$ (lower bound of $\{t^{(i)}(\infty)\}$) $(i=0,1,\ldots)$.

Note: Here $t^{(i+1)}(r)=t(t^{(i)}(r))$ where $t^{(0)}(r)=r$.

Proof. Let $p\in$ **PP** be a solution of equation (1). Obviously, $\phi\leqslant p$ $(p\leqslant\infty)$. From the condition follows that $t^{(i)}(\phi)\leqslant t^{(i)}(p)$ $(t^{(i)}(p)\leqslant t^{(i)}(\infty))$ for any $i\geqslant 0$. Since p is a solution of equation (1), then $t^{(i)}(p)=p$ so that $t^{(i)}(\phi)\leqslant p(p\leqslant t^{(i)}(\infty))$ for all $i\geqslant 0$. □

Corollary 2. *If program expression* $t(\alpha)$ *is monotonic with respect to* α, *then any solution of equation* (1) *is an upper bound of* $\{t^{(i)}(\phi)\}$ *and a lower bound of* $\{t^{(i)}(\infty)\}$ $(i=0,1,\ldots)$.

Proof. Obviously, $\phi\leqslant p\leqslant\infty$ for any $p\in$**PP**. Using the monotony of $t(\alpha)$, we conclude by induction that $t^{(i)}(\phi)\leqslant t^{(i)}(p)\leqslant t^{(i)}(\infty)$ for all $i\geqslant 0$. The assertion is thus proved. □

Theorem 3. *If $t^{(i)}(\phi) \leqslant t(\infty)$, $t^{(i)}(\phi) = \tau(\lambda + q + \cdots + q^{i-1})$ for any $i > 0$, $\tau(\alpha) = \rho(\alpha, \alpha)$, $\rho(\infty, \beta) = \infty$, and $r = p \vdash \rho(r, \beta) = \rho(p, \beta)$, then $\tau(q^*)$ is a solution of equation* (1) .

Proof. Obviously, $t(\tau(\lambda + q + \cdots + q^n)) = \tau(\lambda + q \circ (\lambda + q + \cdots + q^n))$ for all $n \geqslant 0$. According to axiom (SA5), $q^* \circ \phi = \phi \vee q^* = \infty$. If $q^* \circ \phi = \phi$, then, applying rule (I4), we conclude that $t(\tau(q^*)) = \tau(\lambda + q \circ q^*) = \tau(q^*)$. Let $q^* = \infty$. Then, according to (SA10), $\lambda + q = q \vee q = \phi$. If $q = \phi$, then $q^* = \lambda + q \circ q^* = \lambda + \phi \circ q^* = \lambda$. Hence we infer that $\lambda = \infty$ (since the assumption $q^* = \infty$ is in force) and by the rule (I8) we conclude that $\lambda + q = q$. Hence, $\tau(\lambda + q \circ \infty) = \rho(\lambda + q \circ \infty, \lambda + q \circ \infty) = \rho(\lambda + (\lambda + q) \circ \infty, \lambda + q \circ \infty) = \rho(\lambda + \infty + q \circ \infty, \lambda + q \circ \infty) = \rho(\infty, \lambda + q \circ \infty) = \infty$. Since $t^{(n)}(\phi) \leqslant t(\infty)$, then $t(\infty) = t(\infty) + t^{(n+1)}(\infty) = t(\infty) + \tau(\lambda + q + \cdots + q^n)$ for all $n \geqslant 0$. Hence, in accordance with rule (I5) we conclude that $t(\infty) = \infty$ assuming that $q^* = \infty$. Thus, if $q^* = \infty$, then $t(\tau(\infty)) = t(\rho(\infty, \infty)) = t(\infty) = \infty = \tau(\infty) = \tau(\lambda + q \circ \infty)$ and by rule (I3) $t(\tau(q^*)) = \tau(\lambda + q \circ q^*) = \tau(q^*)$. Applying rule (I9), we finally have $t(\tau(q^*)) = \tau(q^*)$. This proves the theorem. □

Corollary 3. *If under the hypothesis of Theorem 3, $t^{(n)}(\phi) \leqslant t^{(n)}(p)$ for any $n \geqslant 0$, $p \in$* **PP**, *and $\rho(\alpha, \beta)$ is monotonic with respect to β then $\tau(q^*)$ is the least solution of equation* (1) .

Corollary 4. *If $t^{(i)}(\phi) = \tau(\lambda + q + \cdots + q^{i-1})$ for all $i > 0$, and $q^* \circ \phi = \phi$, then $\tau(q^*)$ is a solution of equation* (1). *If, in addition, $\{\tau(\lambda + q + \cdots + q^n)\}, (n = 0, 1, \ldots)$ is an increasing chain, then $\tau(q^*)$ is the least solution of equation* (1) .

Theorem 4. *Let us denote $\sum_{i=0}^{n-1} (\neg A \to q)^i \circ A \to p \circ r^i$ by $\sum(n)$, assuming that $\sum(0) = \phi$. Then any solution of equation*

$$\alpha = A \to p \ + \ \neg A \to q \circ \alpha \circ r \tag{2}$$

is an upper bound of the sequence $\{\sum(n) + (\neg A \to q)^n \circ \phi\}$ and a lower bound of the sequence $\{\sum(n) + (\neg A \to q)^n \circ \infty\}$ $(n = 0, 1, 2, \ldots)$.

Proof. Let $t(\alpha) = A \to p \ + \ \neg A \to q \circ \alpha \circ r$. It is easy to prove by induction that $t^{(n)}(\phi) = \sum(n) + (\neg A \to q)^n \circ \phi$, $t^{(n)}(\infty) = \sum(n) + (\neg A \to q)^n \circ \infty$ so that for any $d \in$ **PP** we have $t^{(n)}(\phi) \leqslant t^{(n)}(d) \leqslant t^{(n)}(\infty)$ $(n = 0, 1, 2, \ldots)$. Consequently, the assertion of Theorem 4 follows directly from Theorem 2. □

Remark. In the general case it is impossible to find a solution of equation (2) in the form of a program expression of finite length composed of the expressions p, q, r and predicate A with the aid of basic operations $+$, $\circ$, $\to$, $*$, $\&$, $\vee$, $\neg$. This is due to the fact that in the calculus are fixed properties that are common for all programming languages including those in which all data types have finite sets of objects. At the same time (as follows from a connotative interpretation of Theorem 4), the existen
a general solution of equation (2) presupposes the presence of a data type "counte
loop turns" of infinite power that possesses completely defined properties. Despite this

it is possible for the expressions p, q, r and predicate A of a special kind to construct a general solution of equation (2) without imposing constraints on data types.

Theorem 5. *If* $(\neg A\to q)^n \circ A\to p\circ r^n = (\neg A\to q)^n \circ A\to p\circ r^*$ for any $n>0$, then $(\neg A\to q)^* \circ A\to p\circ r^*$ is the least solution of equation (2).

Proof. Let $t(\alpha)=A\to p \;+\; \neg A\to q\circ\alpha\circ r$. It is easy to show that under the conditions of the theorem

$$t^{(n)}(\phi)=\left(\sum_{i=0}^{n-1}(\neg A\to q)^i\right)\circ A\to p\circ r^* + (\lambda+\neg A\to q\circ\left(\sum_{i=0}^{n-1}(\neg A\to q)^i)\right)\circ\phi$$

for all $n>0$. This is followed by the application of Corollary 3 of Theorem 3. □

Theorem 6. *If* $r\circ\phi=\phi$, $p\circ A\to\lambda = A\to p$, $r\circ A\to\lambda=A\to r$, $r\circ\neg A\to\lambda=\neg A\to r$, $p\circ\neg A\to\lambda=\neg A\to p$, $\neg A\to q\circ r=\neg A\to r\circ q$, $\neg A\to q\circ p=\neg A\to p\circ q$, $(\neg A\to q\circ r)^*\circ\phi=\phi$, *then* $p\circ(\neg A\to q\circ r)^*\circ A\to\lambda$ *is a solution of equation* (2). *If, in addition,* $q\circ\phi=\phi$, *then* $p\circ(\neg A\to q\circ r)^*\circ A\to\lambda$ *is the least solution of equation* (2) .

Proof. Let $t(\alpha)=A\to p+\neg A\to q\circ\alpha\circ r$. Then, $t^{(n)}(\phi)=\sum_{i=0}^{n-1}(\neg A\to q)^i\circ A\to p\circ r^i+(\neg A\to q)^n\circ\phi$ for any $n>0$. We will show by induction that $(\neg A\to q)^n\circ\phi=(\sum_{i=0}^{n}(\neg A\to q\circ r)^i)\circ\phi$ for all $n>0$. Let $n=1$. Then $\neg A\to q\circ\phi=\lambda\circ\phi+\neg A\to q\circ(r\circ\phi)=(\lambda+\neg A\to q\circ r)\circ\phi$. Let us assume that $(\neg A\to q)^n\circ\phi=(\lambda+\neg A\to q\circ r+\cdots+(\neg A\to q\circ r)^n)\circ\phi$. It is easy to show that $(\neg A\to q)^n\circ\phi=(\neg A\to q\circ r)^n\circ\phi$ for any $n>0$. Thus, $(\neg A\to q)^{n+1}\circ\phi=(\neg A\to q\circ r)^{n+1}\circ\phi=\neg A\to q\circ r\circ(\neg A\to q\circ r)^n\circ\phi=\neg A\to q\circ r\circ(\lambda+\cdots+(\neg A\to q\circ r)^n)\circ\phi=\lambda\circ\phi+(\neg A\to q\circ r)\circ\phi+\cdots+(\neg A\to q)^{n+1}\circ\phi$ $=(\sum_{i=0}^{n+1}(\neg A\to q\circ r)^i)\circ\phi$. Let us now show that $\sum_{i=0}^{n-1}(\neg A\to q)^i\circ A\to p\circ r^i=p\circ(\sum_{i=0}^{n-1}(\neg A\to q\circ r)^i)\circ A\to\lambda$ for all $n>0$. We have $(\neg A\to q)^n\circ A\to p\circ r^n=p\circ(\neg A\to q\circ r)^n\circ A\to\lambda$ for any $n\geqslant 0$. In fact, if $n=0$ then $A\to p=p\circ A\to\lambda=p\circ\lambda\circ A\to\lambda$. Let $(\neg A\to q)^{n-1}\circ A\to p\circ r^{n-1}=p\circ(\neg A\to q\circ r)^{n-1}\circ A\to\lambda$. Then, $(\neg A\to q)^n\circ A\to p\circ r^n=\neg A\to q\circ((\neg A\to q)^{n-1}\circ A\to p\circ r^{n-1})\circ r = \neg A\to q\circ p\circ(\neg A\to q\circ r)^{n-1}\circ A\to\lambda\circ r = \neg A\to p\circ q\circ(\neg A\to q\circ r)^{n-1}\circ r\circ A\to\lambda$. It can be easily shown by induction that $(\neg A\to q\circ r)^{n-1}\circ r=r\circ(\neg A\to q\circ r)^{n-1}$ for all $n>0$. Thus, $\neg A\to p\circ q\circ(\neg A\to q\circ r)^{n-1}\circ r\circ A\to\lambda=p\circ\delta\neg A\to\lambda\circ q\circ r\circ(\neg A\to q\circ r)^{n-1}\circ A\to\lambda=p\circ(\neg A\to q\circ r)^n\circ A\to\lambda$. Then,

$$\begin{aligned}
&A\to p+\neg A\to q\circ A\to p\circ r+\cdots+(\neg A\to q)^{n-1}\circ A\to p\circ r^{n-1}\\
&\quad=p\circ\lambda\circ A\to\lambda+p\circ\neg A\to q\circ r\circ A\to\lambda+\cdots+p\circ(\neg A\to q\circ r)^{n-1}\circ A\to\lambda\\
&\quad=p\circ(\lambda\circ A\to\lambda+\neg A\to q\circ r\circ A\to\lambda+\cdots+(\neg A\to q\circ r)^{n-1}\circ A\to\lambda)\\
&\quad=p\circ(\lambda+\neg A\to q\circ r+\cdots+(\neg A\to q\circ r)^{n-1})\;\circ A\to\lambda
\end{aligned}$$

for all $n>0$. Thus,

$$t^{(n)}(\phi)=p\circ\left(\sum_{i=0}^{n-1}(\neg A\to q\circ r)^i\right)\circ A\to\lambda+\left(\sum_{i=0}^{n}(\neg A\to q\circ r)^i\right)\circ\phi$$

for all $n>0$. Since $(\neg A\to q\circ r)^*\circ\phi=\phi$, then according to Corollary 4 of Theorem 3 the program expression

$$p\circ(\neg A\to q\circ r)^*\circ A\to\lambda+(\lambda+\neg A\to q\circ r\circ(\neg A\to q\circ r)^*)\circ\phi$$
$$=p\circ(\neg A\to q\circ r)^*\circ A\to\lambda$$

is a solution of equation (2). Moreover, if $q\circ\phi=\phi$, then $(\sum_{i=0}^{n}(\neg A\to q\circ r)^i)\circ\phi=\phi$. Consequently, $t^{(n)}(\phi)=p\circ(\sum_{i=0}^{n-1}(\neg A\to q\circ r)^i)\circ A\to\lambda$. Since the sequence $\{p\circ(\sum_{i=0}^{n}(\neg A\to q\circ r)^i)\circ A\to\lambda\}$ $(n=0,1,\ldots)$ is an increasing chain, then according to Corollary 4 of Theorem 3 the program expression $p\circ(\neg A\to q\circ r)^*\circ A\to\lambda$ is the least solution of equation (2). □

Example 2. Let us find the least solution of the system from Example 1. The initial system can be transformed easy to the following equivalent system

$$\alpha=[Y:\Theta]\circ\alpha1$$
$$\alpha1=X>Y\to[X:X-Y]\circ\alpha1+X<Y\to[Y:Y-X]\circ\alpha1+(X=Y)\to\lambda\,.$$

Since $X<Y\sim\neg(X>Y)\,\&\,X<Y, X<Y\sim\neg(X=Y)\,\&\,X<Y, X>Y\sim\neg(X=Y)$ $\&\,X>Y$, then

$$X>Y\to[X:X-Y]\circ\alpha1+X<Y\to[Y:Y-X]\circ\alpha1+(X=Y)\to\lambda$$
$$=X>Y\to[X:X-Y]\circ\alpha1+\neg(X>Y)\to X<Y\to[Y:Y-X]\circ\alpha1$$
$$+(X=Y)\to\lambda$$
$$=(X>Y\to[X:X-Y]+\neg(X>Y)\to X<Y\to[Y:Y-X])\circ\alpha1+(X=Y)\to\lambda$$
$$=(X>Y\to[X:X-Y]+X<Y\to[Y:Y-X])\circ\alpha1+(X=Y)\to\lambda$$
$$=(\neg(X=Y)\to X>Y\to[X:X-Y]+\neg(X=Y)\to X<Y\to[Y:Y-X])$$
$$\circ\alpha1+(X=Y)\to\lambda$$
$$=\neg(X=Y)\to(X>Y\to[X:X-Y]+X<Y\to[Y:Y-X])\circ\alpha1+(X=Y)\to\lambda\,.$$

Thus,

$$\alpha1=(X=Y)\to\lambda+\neg(X=Y)\to(X>Y\to[X:X-Y]$$
$$+X<Y\to[Y:Y-X])\circ\alpha1\,.$$

Since $\lambda^n=\lambda$ for any $n>0$, and $\lambda^*=\lambda$, then according to Theorem 5 the expression

$$(\neg(X=Y)\to(X>Y\to[X:X-Y]+X<Y\to[Y:Y-X]))^*\circ(X=Y)\to\lambda$$

is the least solution of the equation

$$\alpha1=X>Y\to[X:X-Y]\circ\alpha1+X<Y\to[Y:Y-X]\circ\alpha1+(X=Y)\to\lambda<Y.$$

Thus, the least solution of the initial system with respect to α is the expression

$$[Y:\Theta]\circ(X>Y\to[X:X-Y]+X<Y\to[Y:Y-X])^*\circ(X=Y)\to\lambda.$$

Remark. The formula $p \leqslant q$ may informally mean that the program p is faster than the program q. This makes a formal optimization of programs possible. For example, if $p \approx q$ and $p \leqslant q$, then the program p is an optimized version of the program q. Few examples of using this technique for an optimization of real programs can be found in [13].

3. Algebraic statement

The present article extends the language of [7, 8] so that it is possible to provide an algebraic statement of the propositional semantics of program expressions.

Definition. The set **RPG** of generalized logical expressions is defined thus:

(1) $\mathbf{RP} \subset \mathbf{RPG}$;

(2) if $A, B \in \mathbf{RPG}$ then $A \,\&\, B, A \vee B, \neg A, A', \mathrm{wlp}(p, A) \in \mathbf{RPG}$ for all $p \in \mathbf{PP}$.

Definition. The interpretational mapping $\hat{\xi} : \mathbf{RPG} \to M_{\mathbf{RP}}$ is defined thus:

(1) if $A \in \mathbf{RP}$ then $\hat{\xi}[A] \sim \xi[A]$;

(2) if $p \in \mathbf{PP}$, $A \in \mathbf{RPG}$ and $m \in M$ then

$$\hat{\xi}[\mathrm{wlp}(p, A)](m) \;\sim\; \hat{\xi}[A](\psi_1[p](m)).$$

Remark. Informally, $\mathrm{wlp}(p, A)$ denotes a relation that extracts all of the initial memory states for which p either terminates normally in a state satisfying A or does not terminate. In other words, $\mathrm{wlp}(p, A)$ is the weakest loose pre-condition for post-condition A in the terminology of [4].

Proposition 7. *The definition of* wlp *is correct in the sense that if* $\xi[A] \in M_{\mathbf{RP}}$, $\psi[p] \in M$, *then* $\xi[\mathrm{wlp}(p, A)] \in M_{\mathbf{RP}}$.

Proof. The map $\hat{\xi}[\mathrm{wlp}(p, A)] : M \to B$ is monotonic because it is the composite of monotonic maps. Moreover,

$$\hat{\xi}[\mathrm{wlp}(p, A)](\omega_S) \;\sim\; \hat{\xi}[A](\psi_1[p](\omega_S)) \;\sim\; \hat{\xi}[A](\omega_S)) \;\sim\; \mathrm{f},$$

$$\hat{\xi}[\mathrm{wlp}(p, A)]\,(\Omega_S) \;\sim\; \hat{\xi}[A](\psi_1[p](\Omega_S)) \;\sim\; \hat{\xi}[A](\Omega_S)) \;\sim\; \mathrm{t}. \quad \square$$

Operation′ is characterized by the following properties:

$$(A')' \;\sim\; A,$$

$$(A \,\&\, B)' \;\sim\; A' \vee B',$$

$$(A \vee B)' \;\sim\; A' \,\&\, B',$$

$$A \vee A' \;\sim\; \mathrm{T},$$

$$A \,\&\, A' \;\sim\; \mathrm{F},$$

where $A, B \in \mathbf{RPG}$.

Computing the propositional semantics of program expressions comes to perfoming reductions of the form **RPG→RP**. Let us state the properties of the operation wlp that are necessary for performing these reductions.

Let $p, q \in \mathbf{PP}$, and $A, B \in \mathbf{RPG}$, and let all occurrences of the variable x in the expression $C(x) \in \mathbf{RP}$ be marked, and let t be an assignment expression of the same type as x and its notation not using the symbols $\rightarrow$ and $+$. Then the following formulas are valid:

(A1)	$\mathrm{wlp}([x{:}\ t], C(x)) \sim C(t)$
(A2)	$\mathrm{wlp}(\lambda, A) \sim A$
(A3)	$\mathrm{wlp}(\infty, A) \sim \mathrm{T}$
(A4)	$\mathrm{wlp}(p \circ q, A) \sim \mathrm{wlp}(p, \mathrm{wlp}(q, A))$
(A5)	$\mathrm{wlp}(B \rightarrow p + \neg B \rightarrow q, A) \sim B \,\&\, \mathrm{wlp}(p, A) \vee \neg B \,\&\, \mathrm{wlp}(q, A)$
(A6)	$\mathrm{wlp}(B \rightarrow p, A) \sim B \,\&\, \mathrm{wlp}(p, A)$
(A7)	$\mathrm{wlp}(\lambda + p, A) \sim \mathrm{wlp}(p, A) \vee A \,\&\, (\mathrm{wlp}(p, \mathrm{T}))'$
(A8)	$\mathrm{wlp}(p, A \vee B) \sim \mathrm{wlp}(p, A) \vee \mathrm{wlp}(p, B)$
(A9)	$\mathrm{wlp}(p, A \,\&\, B) \sim \mathrm{wlp}(p, A) \,\&\, \mathrm{wlp}(p, B)$

In addition, we have the rule

(B1) $$\frac{p \approx q}{\mathrm{wlp}(p, A) \sim \mathrm{wlp}(q, A)}$$

In particular, it follows from (A6) and (SA16) that $\mathrm{wlp}(\phi, A) \sim \mathrm{F}$.

The properties given above make it possible to compute the propositional semantics of program expressions that do not contain the operation *. The following analysis of the extended language is directed toward construction of a rule for computing the propositional semantics of the program expressions using *.

Theorem 7. *Let* $A, B, D \in \mathbf{RPG}$, *and* $p \in \mathbf{PP}$. *If* $\mathrm{wlp}(B \rightarrow p, \mathrm{T}) \sim D$, *then* $\mathrm{wlp}((B \rightarrow p)^*, A)$ *is a solution of the equation*

$$X \sim D' \,\&\, A \vee \mathrm{wlp}(B \rightarrow p, X). \qquad (3)$$

Proof. Let $X \sim \mathrm{wlp}((B \rightarrow p)^*, A)$. Then

$$\begin{aligned} X &\sim \mathrm{wlp}(\lambda + (B \rightarrow p) \circ (B \rightarrow p)^*, A) \\ &\sim \mathrm{wlp}(B \rightarrow p \circ (B \rightarrow p)^*, A) \vee A \,\&\, (\mathrm{wlp}(B \rightarrow p \circ (B \rightarrow p)^*, \mathrm{T}))' \\ &\sim \mathrm{wlp}(B \rightarrow p, \mathrm{wlp}((B \rightarrow p))^*, A)) \vee A \,\&\, (\mathrm{wlp}(B \rightarrow p, \mathrm{wlp}((B \rightarrow p)^*, \mathrm{T})))'. \end{aligned}$$

Here we have used the axiom (SA15). It follows from (A7) that $\mathrm{wlp}(\lambda + q, \mathrm{T}) \sim \mathrm{T}$ for all $q \in \mathbf{PP}$. Thus, $\mathrm{wlp}(B \rightarrow p)^*, \mathrm{T})) \sim \mathrm{T}$. As a result,

$$X \sim \mathrm{wlp}(B \rightarrow p, X) \vee A \,\&\, (\mathrm{wlp}(B \rightarrow p, \mathrm{T}))'. \quad \square$$

Remark. Search for solutions of equation (3) requires completeness of the Boolean lattice M_{RP}. Since in this case there exists at least one solution of equation (3), a stationary point of the monotonic map $B' \,\&\, A \vee \mathrm{wlp}(B \rightarrow p, X)$ of the lattice M_{RP} into itself [3], and it is meaningful to speak of the least upper and the greatest lower bounds (lub and glb, respectively) of infinite chains in M_{RP}.

Theorem 8. *Assume that the lattice* M_{RP} *is complete. Then, under the hypothesis of Theorem* 7, *if* $\mathrm{wlp}(B \rightarrow p, \mathrm{F}) \sim \mathrm{F}$ *and* X *is a solution of equation* (3), *we have*

$$\mathrm{lub}\left\{ \bigvee_{i=0}^{n} \mathrm{wlp}((B \rightarrow p)^i, D' \,\&\, A) \right\} \Rightarrow X$$

$$\Rightarrow \mathrm{glb}\left\{ \bigvee_{i=0}^{n} \mathrm{wlp}((B \rightarrow p)^i, D' \,\&\, A) \vee \mathrm{wlp}((B \rightarrow p)^n, D) \right\} \quad (n=0, 1, \ldots) \tag{4}$$

Proof. We write $G(X) \sim D' \,\&\, A \vee \mathrm{wlp}(B \Rightarrow p, X)$, setting $G^{(0)}(X) \sim X$. By (A8) and (A9), the map $G: M_{\mathrm{RP}} \rightarrow M_{\mathrm{RP}}$ is monotonic. It is clear that $G^{(0)}(\mathrm{F}) \sim \mathrm{F} \Rightarrow X \Rightarrow T \sim G^{(0)}(\mathrm{T})$. Let $G^{(m)}(\mathrm{F}) \Rightarrow X \Rightarrow G^{(m)}(\mathrm{T})$. Then $G^{(m+1)}(\mathrm{F}) \sim G(G^{(m)}(\mathrm{F})) \Rightarrow G(X) \Rightarrow G(G^{(m)}(\mathrm{T})) \sim G^{(m+1)}(\mathrm{T})$ because G is monotonic. Since X is a solution of equation (3), we have $G(X) \sim X$. Thus, $G^{(m+1)}(\mathrm{F}) \Rightarrow X \Rightarrow G^{(m+1)}(\mathrm{T})$. By induction, $G^{(n)}(\mathrm{F}) \Rightarrow X \Rightarrow G^{(n)}(\mathrm{T})$ for all $n \geqslant 0$. This means that X is an upper bound for $\{G^{(n)}(\mathrm{F})\}$, which is increasing, and a lower bound for $\{G^{(n)}(\mathrm{T})\}(n=0, 1, \ldots)$, which is decreasing. Thus $\mathrm{lub}\{G^{(n)}(\mathrm{F})\} \Rightarrow X \Rightarrow \mathrm{glb}\{G^{(n)}(\mathrm{T})\}$. Since

$$G^{(n)}(\mathrm{F}) \sim \bigvee_{i=0}^{n-1} \mathrm{wlp}((B \rightarrow p)^i, D' \,\&\, A),$$

$$G^{(n)}(\mathrm{T}) \sim \bigvee_{i=0}^{n-1} \mathrm{wlp}((B \rightarrow p)^i, D' \,\&\, A) \vee \mathrm{wlp}((B \rightarrow p)^{n-1}, D) \text{ for all } n>0,$$

formula (4) is valid. $\square$

The following two theorems explain the semantics of lub and glb of infinite chains in M_{RP}, which is necessary for interpretation of formula (4).

Definition. Let $\{r_n\}$ $(n=0, 1, \ldots)$ be a chain of M_{RP}. The relations $\bigvee_{n=0}^{\infty} r_n$ and $\&_{n=0}^{\infty} r_n$ are defined on the memory state space as follows:

(1) a state $m \in M$ satisfies $\bigvee_{n=0}^{\infty} r_n$ iff there exists $k>0$ such that m satisfies $\bigvee_{n=0}^{k} r_n$;

(2) a state $m \in M$ satisfies $\&_{n=0}^{\infty} r_n$ iff there exists k such that state for any $n \geqslant km$ satisfies r_n.

Theorem 9. *If the lattice* M_{RP} *is complete, then*

$$\mathrm{lub}\{r_n\} \sim \bigvee_{n=0}^{\infty} r_n,$$

$$\text{glb}\{r_n\} \sim \mathop{\&}_{n=0}^{\infty} r_n$$

for any chain $\{r_n\}$ $(n=0,1,\ldots)$ *of* M_{RP}.

Proof. It is known that in the class of Boolean lattices the properties of completeness and continuity are equivalent [3]. By definition [10], a complete lattice L is continuous iff for any $a\in L$ and arbitrary chain $C\subset L$ we have $a\,\&\,\text{lub}\,C=\text{lub}\{a\,\&\,c: c\in C\}$. Thus, $(\bigvee_{n=0}^{\infty} r_n)\,\&\,\text{lub}\{r_n\} \sim \text{lub}\{(\bigvee_{n=0}^{\infty} r_n)\,\&\,r_n\}$ for any chain $\{r_n\}$ $(n=0,1,\ldots)$ of M_{RP}. Clearly $(\bigvee_{n=0}^{\infty} r_n)\,\&\,r_n \sim r_n$, $(\bigvee_{n=0}^{\infty} r_n)\,\&\,\text{lub}\{r_n\} \sim \bigvee_{n=0}^{\infty} r_n$ $(n=0,1,\ldots)$. Thus, finally, $\text{lub}\{r_n\} \sim \bigvee_{n=0}^{\infty} r_n$. Similarly, $\text{glb}\{r_n\} \sim \&_{n=0}^{\infty} r_n$. □

Theorem 10. *Under the hypothesis of Theorem* 8

$$\text{glb}\left\{\bigvee_{i=0}^{n} \text{wlp}((B\to p)^i, D'\,\&\,A) \vee \text{wlp}((B\to p)^n, D)\right\}$$
$$\sim \left(\bigvee_{n=0}^{\infty} \text{wlp}((B\to p)^n, D'\,\&\,A)\right) \vee \left(\mathop{\&}_{n=0}^{\infty} \text{wlp}((B\to p)^n, D)\right).$$

Proof. We write $h_n \sim \text{wlp}((B\to p)^n, D)$, $r_n \sim \bigvee_{i=0}^{n}\text{wlp}((B\to p)^i, D'\,\&\,A)$. It is clear that $\{r_n\}$ is an increasing chain, while $\{h_n\}$ and $\{r_n \vee h_n\}$ are decreasing chains $(n=0,1,\ldots)$. It follows from the continuity of M_{RP} that $\text{lub}\{r_n\} \vee \text{glb}\{r_n \vee h_n\} \sim \text{glb}\{\text{lub}\{r_n\} \vee r_n \vee h_n\}$. By Theorem 8, we have $\text{lub}\{r_n\} \vee \text{glb}\{r_n \vee h_n\} \sim \text{glb}\{r_n \vee h_n\}$. Obviously, $\text{lub}\{r_n\} \vee r_n \sim \text{lub}\{r_n\}$ $(n=0,1,\ldots)$. Thus, $\text{glb}\{r_n \vee h_n\} \sim \text{glb}\{\text{lub}\{r_n\} \vee h_n\}$ $(n=0,1,\ldots)$. Once again appealing to the continuity of M_{RP}, we obtain, finally, $\text{glb}\{\text{lub}\{r_n\} \vee h_n\} \sim \text{lub}\{r_n\} \vee \text{glb}\{h_n\}$. Thus, $\text{glb}\{r_n \vee h_n\} \sim \text{lub}\{r_n\} \vee \text{glb}\{h_n\}$. □

Theorem 11. *Let* $B\to p \neq B\to\lambda$. *Then, under the hypothesis of Theorem* 8,

$$\text{wlp}((B\to p)^*, A) \sim \left(\bigvee_{n=0}^{\infty} \text{wlp}((B\to p)^n, D'\,\&\,A)\right) \vee \left(\mathop{\&}_{n=0}^{\infty} \text{wlp}((B\to p)^n, D)\right).$$

Proof. It is sufficient to prove that in the notation of Theorem 10, $(\bigvee_{n=0}^{\infty} r_n) \vee (\&_{n=0}^{\infty} h_n) \Rightarrow \text{wlp}((B\to p)^*, A)$. Let $m\in M$ satisfy $(\bigvee_{n=0}^{\infty} r_n) \vee (\&_{n=0}^{\infty} h_n)$. If m satisfies $(\bigvee_{n=0}^{\infty} r_n)$, then $(B\to p)^*$ terminates normally in a state satisfying A, so m satisfies $\text{wlp}((B\to p)^*, A)$. If, however, m satisfies $(\&_{n=0}^{\infty} h_n)$, then $(B\to p)^*$ does not terminate (which follows from $B\to p \neq B\to\lambda$), so in this case m also satisfies $\text{wlp}((B\to p)^*, A)$. □

Thus, under the assumption that the lattice M_{RP} is complete we have

$$\text{(B2)}\quad \frac{\text{wlp}(B\to p, \text{T}) \sim D, \qquad B\to p \neq B\to\lambda, \qquad \text{wlp}(B\to p, \text{F}) \sim \text{F}}{\text{wlp}((B\to p)^*, A) \sim \left(\bigvee_{n=0}^{\infty} \text{wlp}((B\to p)^n, D'\,\&\,A)\right) \vee \left(\mathop{\&}_{n=0}^{\infty} \text{wlp}((B\to p)^n, D)\right)}$$

Remark. If the expression $\text{wlp}(q^n, A)$ is represented by the formula $Q(n, m)$ in weak second-order logic, where $m \in M$, then

$$\bigvee_{n=0}^{\infty} \text{wlp}(q^n, A) \sim \bigvee_{n=0}^{\infty} Q(n, m) \sim \exists n : Q(n, m),$$

$$\mathop{\&}_{n=0}^{\infty} \text{wlp}(q^n, A) \sim \mathop{\&}_{n=0}^{\infty} Q(n, m) \sim \forall n : Q(n, m) \ .$$

Theorem 12. *If the lattice $\langle M_S, \leqslant_S \rangle$ is continuous, and, moreover, the operations and elementary relations of* **S** *are continuous, then $\langle M_{\text{RP}}, \Rightarrow \rangle$ is complete.*

Proof. The lattice $\prod_{i=1}^{N} M_{T_i}$ is continuous, since it is the cartesian product of a finite number of continuous lattices. It follows that $\langle M, \leqslant \rangle$ is also continuous. The Boolean lattice B is continuous because it is a lattice of finite length. It is known [15] that if L and L' are continuous lattices, then the family $[L \rightarrow L']$ of continuous mappings from L into L' is also a continuous lattice. Thus, the lattice $[M \rightarrow B]$ is continuous. By construction, the lattice M_{RP} consists of precisely all continious mappings $M \rightarrow B$ except two ones: r_{F} and r_{T}, where $r_{\text{F}}(m) \sim \text{f}$, $r_{\text{T}}(m) \sim \text{t}$, for all $m \in M$. The mappings r_{F} and r_{T} are the zero and identity of the lattice $[M \rightarrow B]$, respectively. Continuity of $[M \rightarrow B]$ ensures continuity of M_{RP}. Indeed, let $r \in M_{\text{RP}}$ and $C \subset M_{\text{RP}}$. Then $r \in [M \rightarrow B]$, $C \subset [M \rightarrow B]$, $r \,\&\, \text{lub}\, C \Leftrightarrow \text{lub}\{r \,\&\, c : c \in C\}$, where the operations $\&$ and lub are performed in the lattice $[M \rightarrow B]$, while $\Leftrightarrow$ denotes extension of equality $\sim$ of the lattice M_{RP} to the lattice $[M \rightarrow B]$. We have $\text{F} \Rightarrow c \Rightarrow \text{T}$, $\text{F} \Rightarrow r \,\&\, c \Rightarrow \text{T}$ for all $c \in C$. Thus, $\text{F} \Rightarrow \text{lub}\, C \Rightarrow \text{T}$, $\text{F} \Rightarrow r \,\&\, \text{lub}\, C \Rightarrow \text{T}$, $\text{F} \Rightarrow \text{lub}\{r \,\&\, c : c \in C\} \Rightarrow \text{T}$, so that $\text{lub}\{r \,\&\, c : c \in C\} \in M_{\text{RP}}$, $r \,\&\, \text{lub}\, C \in M_{\text{RP}}$. As a result, $r \,\&\, \text{lub}\, C \sim \text{lub}\{r \,\&\, c : c \in C\}$, where $\&$ and lub are evaluated in M_{RP}, which means that M_{RP} is continuous. Since continuity and completeness are equivalent in the class of Boolean lattices, this proves the theorem. □

Corollary 5. *If different proper objects in the lattice $\langle M_S, \leqslant_S \rangle$ are not comparable, then the lattice $\langle M_{\text{RP}}, \Rightarrow \rangle$ is complete.*

Proof. Incomparability of different proper objects assures that the length of $\langle M_S, \leqslant_S \rangle$ is finite, from which it follows that it is continuous. Since monotonic mappings between lattices of finite length are continuous, so are the type and intertype operations, as well as the elementary relations of **S**. The rest follows from Theorem 12. □

Remark. If we remove the object Θ from M_S, all of the results we have obtained remain valid for this interpretation of our formal language. Moreover, in this case the operations $\neg$ and 'coincide. This makes it possible to transform generalized logical expressions according to the rules of Boolean algebra, and ensures elimination of the symbol during the process of performing reductions of the form **RPG→RP**.

Example 3. In optimizing compilers automatic program transformations are performed under the assumption that semantics of the transformed program will be not changed. Let us show just how this assumption may be verified. For example, consider the Fortran loop

```
      DO 1 I=1, N
      IF(X.LT.1) X=1./Y
    1 X=X/2
```

Its semantics is established by the system of equations

$$\alpha=[I:1]\circ\alpha1,$$
$$\alpha1=I\leqslant N\to\alpha2+I>N\to\lambda,$$
$$\alpha2=(X<1\to[X:1./Y]+X\geqslant1\to\lambda)\circ\alpha3,$$
$$\alpha3=[X:X/2]\circ\alpha4,$$
$$\alpha4=[I:I+1]\circ\alpha1\,.$$

This system can be transformed to the following equivalent system:

$$\alpha=[I:1]\circ\alpha1$$
$$\alpha1=I>N\to\lambda+I\leqslant N\to(X<1\to[X:1./Y]+X\geqslant1\to\lambda)\circ[X:X/2]$$
$$\circ[I:I+1]\circ\alpha1\,.$$

Applying Theorem 5 with $r=\lambda$ to the second equation of the system we have

$$\alpha1=(I\leqslant N\to(X<1\to[X:1./Y]+X\geqslant1\to\lambda)\circ[X:X/2]\circ[I:I+1])^*\circ I$$
$$>N\to\lambda\,.$$

Thus, the semantics of the initial loop is established by the program expression

$$p=[I:1]\circ(I\leqslant N\to q)^*\circ I>N\to\lambda\,,$$

where

$$q=(X<1\to[X:1./Y]+X\geqslant1\to\lambda)\circ[X:X/2]\circ[I:I+1]\,.$$

Assume that an optimizing compiler recognizes the expression $1./Y$ as an invariant one [1] and moves it outside the loop. As a result, the loop is transformed to

```
      Z=1./Y
      DO 1 I=1, N
      IF(X.LT.1) X=Z
    1 X=X/2
```

The semantics of this fragment are given by the program expression

$$\hat{p}=[Z:1./Y]\circ[I:1]\circ(I\leqslant N\to\hat{q})^*\circ I>N\to\lambda\,,$$

where

$$\hat{q}=(X<1\to[X:Z]+X\geqslant 1\to\lambda)\ \circ[X:X/2]\circ[I:I+1]\,.$$

To simplify the analysis, we use the field of real numbers to simulate the type REAL, and the ring of integers to simulate the type INTEGER. Also, we assume that the variables X and Y that appear in the initial fragment have been initialized. This makes it possible to identify$'$ and $\neg$ and to operate on logical expressions by the rules of Boolean algebra. It is not difficult to show that the domains of nonterminating wlp(p, F) and wlp($\hat{p}$, F) of both the initial and transformed fragments are empty. Indeed,

$$\begin{aligned}\mathrm{wlp}(\hat{p},F)\ &\sim\ \mathrm{wlp}([I:1,Z:1./Y]\circ(I\leqslant N\to\hat{q})^*\circ I>N\to\lambda,\mathrm{F})\\ &\sim\ \mathrm{wlp}([I:1,Z:1./Y],\mathrm{wlp}((I\leqslant N\to\hat{q})^*,\mathrm{wlp}(I>N\to\lambda,\mathrm{F})))\\ &\sim\ \mathrm{wlp}([I:1,Z:1./Y],\mathrm{wlp}((I\leqslant N\to\hat{q})^*,\mathrm{F}))\ .\end{aligned}$$

By rule (B2),

$$\mathrm{wlp}((I\leqslant N\to\hat{q})^*,\mathrm{F})\ \sim\ \mathop{\&}_{n=0}^{\infty}\ \mathrm{wlp}((I\neq N\to\hat{q})^n,I\leqslant N)\ \sim\ \mathop{\&}_{n=0}^{\infty}\ (I+n<N)\ .$$

Thus,

$$\begin{aligned}\mathrm{wlp}(\hat{p},\mathrm{F})\ &\sim\ \mathrm{wlp}([I:1,Z:1./Y],\ \mathop{\&}_{n=0}^{\infty}(I+n<N))\\ &\sim\ \mathop{\&}_{n=0}^{\infty}\ (1+n<N)\ \sim\ \mathrm{F}.\end{aligned}$$

Similarly, wlp(p, F) $\sim$ F. We now calculate the domains wlp(p, T) and wlp($\hat{p}$, T) of normal termination of both the initial and transformated fragments. We have

$$\begin{aligned}\mathrm{wlp}(p,\mathrm{T})\ &\sim\ \mathrm{wlp}([I:1]\circ(I\leqslant N\to q)^*\circ I>N\to\lambda,\mathrm{T})\\ &\sim\ \mathrm{wlp}([I:1],\mathrm{wlp}((I\leqslant N\to q)^*,\mathrm{wlp}(I>N\to\lambda,\mathrm{T})))\\ &\sim\ \mathrm{wlp}([I:1],\mathrm{wlp}((I\leqslant N\to q)^*,I>N))\ .\end{aligned}$$

Since

$$\mathrm{wlp}(I\leqslant N\to q,\mathrm{T})\ \sim\ I\leqslant N\,\&\,(X<1\,\&\,Y\neq 0\vee X\geqslant 1)\ ,$$

it follows from (B2) that

$$\begin{aligned}\mathrm{wlp}((I\leqslant N\to q)^*,I>N)\ &\sim\ \bigvee_{n=0}^{\infty}\ \mathrm{wlp}((I\neq N\to q)^n,I>N)\\ &\sim\ I>N\vee I\leqslant N\,\&\,I+1>N\,\&\,(X<1\,\&\,Y\neq 0\vee x\geqslant 1)\end{aligned}$$

$$\vee\, I+1\leqslant N\,\&\,I+2>N\,\&\,(X/2<1\,\&\,Y\neq 0\vee X/2\geqslant 1)\ \vee \dots$$

$$\sim\ I>N\vee\left(\bigvee_{n=1}^{\infty}(I=N-n+1\,\&\,(X<2^{n-1}\,\&\,Y\neq 0\vee X\geqslant 2^{n-1}))\right).$$

Thus,

$$\begin{aligned}\mathrm{wlp}(p,\mathrm{T})\ &\sim\ N<1\vee\left(\bigvee_{n=1}^{\infty}(n=N\,\&\,(X<2^{n-1}\,\&\,Y\neq 0\vee X\geqslant 2^{n-1}))\right)\\ &\sim\ N<1\vee\exists n\geqslant 1\colon (n=N\,\&\,(X<2^{n-1}\,\&\,Y\neq 0\vee X\geqslant 2^{n-1}))\\ &\sim\ N<1\vee N\geqslant 1\,\&\,(X<2^{N-1}\,\&\,Y\neq 0\vee X\geqslant 2^{N-1}).\end{aligned}$$

On the other hand,

$$\mathrm{wlp}(\hat{p},\mathrm{T})\ \sim\ \mathrm{wlp}([I:1,Z:1./Y],\mathrm{wlp}((I\leqslant N\to\hat{q})^{*},I>N))\ .$$

By induction on n,

$$\mathrm{wlp}((I\leqslant N\to\hat{q})^{n},I>N)\ \sim\ I=N-n+1$$

for every $n\geqslant 1$. Consequently,

$$\begin{aligned}\mathrm{wlp}((I\leqslant N\to\hat{q})^{*},I>N)\ &\sim\ \bigvee_{n=0}^{\infty}\mathrm{wlp}((I\leqslant N\to q)^{n},I>N)\\ &\sim\ I>N\vee\left(\bigvee_{n=0}^{\infty}I=N-n+1\right).\end{aligned}$$

Thus,

$$\begin{aligned}\mathrm{wlp}(\hat{p},\mathrm{T})\ &\sim\ \mathrm{wlp}([Z:1./Y],\mathrm{wlp}([I:1],I>N\vee\left(\bigvee_{n=0}^{\infty}I=N-n+1)\right)\\ &\sim\ \mathrm{wlp}([Z:1./Y],1>N\vee\left(\bigvee_{n=0}^{\infty}N=n)\right)\\ &\sim\ \mathrm{wlp}([Z:1./Y],N<1\vee N\geqslant 1)\\ &\sim\ \mathrm{wlp}([Z:1./Y],\mathrm{T})\\ &\sim\ Y\neq 0\,.\end{aligned}$$

Thus, as a result of the optimizing transformation, the domain of definition of the program fragment under discussion has been restricted. For example, for $N=5$, $X=16$, $Y=0$, the initial loop terminates normally, while its optimized version abnormally terminates at the first operation.

4. Conclusion

In conclusion, we would like to say that our article presents in brief the mathematical basis of the approach to definition of semantics of programming languages. The detailed description of the approach as well as applying the approach to the definition of the semantics of the Fortran 77 language can be found in [13]. In contrast to denotational, propositional, or operational approaches, the presented approach is based on an abstraction of the real program in which its computational, functional and propositional properties as well as data types, storage allocation, and data initialization are taken into account.

The point is the different approaches to definition of semantics take into account different properties of real programs and are oriented to a decision of different problems. So, denotational and propositional approaches completely, while the operational approach partially, do not take into account the computational properties of programs that are connected with their execution. Of course, they provide techniques for a formal analysis of functional [4, 16, 14] and propositional [6, 11] properties of programs, they provide techniques for the formal transformations of programs that save their operational [17, 18] or functional [2, 12] properties. The integration of the different approaches that makes it possible to define semantics of a programming language completely enough, is very difficult because of the problem of the consistency of the language's descriptions provided by the different methods [5].

We hope that nice algebraic properties together with the model of a programming language that is rich enough, make the presented method convenient for formal analysis of real programming languages.

References

[1] A. Aho and J. Ullman, *The Theory of Parsing, Translation and Compiling* (Prentice-Hall, Englewood Cliffs., NJ, 1973).

[2] J. Backus, Can programming be liberated from the von Neumann Style? A functional style and its algebra of programs, *Comm. ACM* **21** (8) (1978) 613–641.

[3] G. Birkhoff, *Lattice Theory* (AMS, Providence, RI, 1967).

[4] A. Dijkstra, *A Discipline of Programming* (Prentice-Hall, Englewood Cliffs., NJ, 1976).

[5] J.E. Donahue, Complementary Definitions of Programming Language Semantics, in: G. Goos and J. Hartmanis, eds., Lecture Notes in Computer Science, Vol. 42 (Springer, Berlin, 1976).

[6] R.W. Floyd, Assigning meaning to programs, in: *Proc. of a Symposium in Applied Mathematics*, Vol. 19, Providence, 1967.

[7] S.S. Gaissaryan and A.L. Lastovetsky, An algebraic model of von Neumann programming languages, *Programmirovanie* **10** (6) (1984) 12–22 (in Russian); translated as *Programming & Comput. Software* **10** (1984) 241–249.

[8] S.S. Gaissaryan and A.L. Lastovetsky, Calculus of equivalencies of abstract programs, *Programmirovanie* **11** (5) (1985) 20–31 (in Russian); translated as *Programming & Comput. Software* **11** (1985) 265–273.

[9] S.S. Gaissaryan and A.L. Lastovetsky, Calculus of propositional properties of programs, *Programmirovanie* **16** (3) (1990) 36–44 (in Russian).

[10] G. Gratzer, *General Lattice Theory* (Akademie Verlag, Berlin, 1978).

[11] C.A.R. Hoar, An axiomatic approach to computer programming, *Comm. ACM* **12** (10) (1969) 322–329.

[12] V.P. Kutepov and V.N. Falk, Functional systems: theoretical and practical aspects, *Cybernetics* **15** (3) (1979) 46–58 (in Russian).

[13] A.L. Lastovetsky, Algebraic approach to semantics of programming languages, Ph.D. Thesis, Department of Numerical Mathematics and Computer Science, Moscow Aviation Institute, 1985 (in Russian).

[14] Z. Manna, *Mathematical Theory of Computation* (McGraw-Hill, New York, 1974).

[15] D. Scott, Lattice theory, data types and semantics, in : R. Rustin, ed., *Formal Semantics of Programming Languages* (Prentice-Hall, Englewood Cliffs., NJ, 1972).

[16] D. Scott and C. Strachey, Towards a mathematical semantics for computer languages, in: J. Fox, ed., *Computers and Automata* (Wiley, New York 1972).

[17] Yu.I. Yanov, On logical schemes of algorithms. *Problems of Cybernetics*, Vol. 1 (Fizmatgiz, Moscow, 1958) (in Russian).

[18] A.P. Yershov, On Yanov's schemes, *Problems of Cybernetics*, Vol. 20 (Fizmatgiz, Moscow, 1968) (in Russian).