УДК 004.4'232

V. Ovsyak[1,2], A. Ovsyak[3]
[1]Department of Computer Science, Ukrainian University of Printing
[2]Department of Electrical, Control and Computer Engineering,
Opole University of Technology, Poland
[3]Kyiv National University of Culture and Arts, The L'vov Campus

THE EXTENDED ALGEBRA OF ALGORITHMS WITH MULTICONDITIONAL ELIMINATION

The existing, intuitive computation models, that is the virtual machines of Turing, Post, Kolmogorov, Schönhage, Aho-Ullman-Hopcroft as well as the algorithms of Markov and Krinitski, and the recursive functions, all lack precise, mathematical formulation. Consequently, an algebra of algorithms is defined using the axiomatic method. The algebra is based on the operations of sequencing, elimination, paralleling and reversing as well as cyclic sequencing, cyclic elimination and cyclic paralleling, all of them performed on the so-called uniterms. A useful extension is offered in terms of multiconditional elimination. An example illustrates the usefulness of the algebra of algorithms.

Keywords: model, operation, algorithm, definition.

Вказано відомі методи інтуїтивного опису алгоритмів, якими є віртуальні машини Т'юрінга, Поста, Колмогорова, Шонгаге, Ахо-Ульмана-Хопкрофта, а також алгоритми Маркова і Кріницкого та рекурсивні функції, засобами яких алгоритми описуються не формалізовано. Дефініцію розширеної алгебри алгоритмів подано аксіоматичним методом. Алгебра базується на операціях секвентування, багатозначного елімінування, паралелення і реверсування, а також циклічного секвентування, циклічного елімінування та циклічного паралелення, які виконуються над унітермами. Розширення торкається введення операції багатозначного елімінування. Прикладом проілюстрована ефективність розширеної алгебри алгоритмів.

Ключові слова: модель, операція, алгоритм, визначення.

1. Introduction

The algorithms or models of computations are the essence of computer systems. In particular, it is important for designers to be able to precisely and formally describe algorithms for the developed operating systems, platforms, programming languages and specialized computer systems, including monitoring, control and diagnostic systems. The importance results from the fact that the cost of location and correction of algorithmic errors when implementing or operating the algorithms is very high. The well-known existing models of computations are the λ-calculus [1], Turing machine [2,3], Post machine [4,5], Kolmogorov machine [6-8], Schönhage machine [9,10], Markov algorithms [11,12], recursive functions [1,13-15], the Aho, Ullman and Hopcroft machine (with random access to RAM memory) [16] and the analytical algorithm theory of Krinitski [17]. These are specific tools for algorithm presentation that occurred between the thirties and nineties of the 20th century. It is well known [18] that the above models are equivalent in that they compute 'the same functions'. It is also known [18] that the algorithms were initially described in those models in an intuitive way. Later, the algorithms have been given a precise mathematical rigor [3,5,12,19,20], which has had a tremendous impact on the development of functional and structural programming languages. In the existing computational models, an algorithm appears as a specifically described sequence of specific operations. However, some algorithmic figures like sequencing of operations, realized by means of e.g. 'go to' instructions or multiple bracketing, are still described in an informal, intuitive way. Offering such formal descriptions would contribute to the treatment of algorithms

as mathematical formulae, which could in turn lead to the introduction of algebra of algorithms. This could give means for transformation of the algorithms and their optimization in the sense of reduction of a number of operations and memory occupation, thus contributing to more effective implementation of the algorithms. Even if transformation of algorithms is available for some of the existing computation models, there exist no tools for formal description of some relations between their operations. A precise, formal tool for description, transformation and optimization of algorithms is offered in this paper. An extended version of the classical algebra of algorithms [21,22] is presented, in that multiconditional elimination is substituted for the elementary, conditonal elimination.

The alphabet and operations of the extended algebra of algorithms are presented in Section 2. Section 3 gives intuitive explanations of operations in the algebra of algorithms and Section 4 provides an illustrative example of application of the algebra. New results of the paper are summarized in conclusions of Section 5.

2. Algebra of algorithms

Here we present the (extended) fundamentals of the algebra of abstract algorithms originally introduced in Refs. [21,22]. Constructed with the axiomatic method, the algebra enables to describe algorithms by means of mathematical formulae.

2.1. Alphabet

Definition 1. Any symbols to be subject to ordering will be called **uniterms**.

For instance, the following uniterms can be exemplified: *-5,-4, -3.1,0, 1,1.25, 7, a, A, c, C, x,*…. Uniterms are divided into the **terminal** (or concrete), e.g. $p=q+2$, $S_1=2x-3$ and **abstract** ones, e.g. $F(x)$, $R(x,y)$. The abstract uniterms are denoted by capital Latin letters (with or without indices).

Definition 2. The **alphabet** of the theory of algorithms consists of the following:

1) symbols of operations: $\frown$ – sequencing; $\vdash\!\!\dashv$ – elimination; $\sqcap$ – paralleling; $-$ – reversing; $\circlearrowleft$ – cyclic sequencing; $\not\supset$ – cyclic elimination; $\varnothing$ – cyclic paralleling; $=$ – equalizing. (Note: Vertical arrangement of the denotations for sequencing, elimination and paralleling can also be used.)

2) uniterms not related with the cyclic operations, that is variables, coefficients and constants, with or without indices, denoted with lower-case letters from the beginning of the Latin alphabet: $a, b, c_o, c_1, \ldots, c_j, c^0, c^1, \ldots, c^I, c^i_j, \ldots$

3) uniterms related with the cyclic operations, that is variables, with or without indices, denoted with lower-case letters from the end of the Latin alphabet: $x, x_o, \ldots, x_j, x^0, \ldots, z^i_j, \ldots$

4) uniterms dependent on one or more variables, with or without indices, denoted with capital letters from the Latin alphabet: *P(a), P(a,b),* …

5) conditional uniterms assuming two values (0 or 1): $u, u_o, u_1, \ldots, u_j, u^i_j$.

6) ***** – empty uniterm;

7) coma (,), semicolon (;), colon (:) (Note: Coma and semicolon are used to separate commutative and noncommutative uniterms, respectively, whereas colon is used to denote either coma or colon.)

2.2. Operations

Definitions of the operations in the algebra of algorithms are given below.

2.2.1. **Equalizing** is an operation on uniterms having the following properties:

1) identity: $A = A$,

2) symmetry: if $A = B$, then $B = A$,

3) transitivity: if $A = B$ and $B = S$, then $A = S$.

2.2.2. **Sequencing** is an operation having the following properties:

1) commutativity:

$$\overset{\frown}{R, S} = \overset{\frown}{S, R}$$

Note: Two noncommutative operations on uniterms will be separated by semicolon (rather than coma), so that

$$\overset{\frown}{R;S} \neq \overset{\frown}{S;R}$$

2) associativity:

$$\overbrace{R, S}, T = R, \overbrace{S, T}$$

3) idempotency:

$$\overbrace{S, S} = S$$

4) absorption of the empty uniterm:

$$\overbrace{*: S} = S ,$$

where the symbol ":" means comma or semicolon;

5) extracting a common uniterm:

$$\overbrace{A; B}, \overbrace{A; C} = A; \overbrace{B, C} ,$$

$$\overbrace{A; B}, \overbrace{C; B} = \overbrace{A, C}; B .$$

2.2.3. **Elimination** is an operation having the following properties:

1) selection of a conditional uniterm:

$$\vdash\!\!\!\!-\!\!\!\!\dashv_{R;\, S;\, u\,-\,?} = \begin{cases} R, \text{ if } u = 1; \\ S, \text{ if } u = 0; \end{cases}$$

2) selection of a multiconditional uniterm:

$$\vdash\!\!\!\!-\!\!\!\!\dashv_{R;\, S;\, \ldots ;Z;\, w\,-\,?} = \begin{cases} R, \text{ if } w = v_0; \\ S, \text{ if } w = v_1; \\ . \\ . \\ . \\ S, \text{ if } w = v_{n-1}; \end{cases}$$

where $v_0, v_1, \ldots, v_{n-1}$ are the values of the multiconditional uniterm w.

3) selection of empty multiconditional uniterm(s)

$$\vdash\!\!\!\!-\!\!\!\!\dashv_{*;\, S;\, \ldots ;Z;\, w\,-\,?} = \begin{cases} *, \text{ if } w = v_0; \\ S, \text{ if } w = v_1; \\ . \\ . \\ . \\ S, \text{ if } w = v_{n-1}; \end{cases} .$$

4) idempotency (absorption of a conditional uniterm):

$$\vdash\!\!\!\!-\!\!\!\!\dashv_{A\,;\,A\,;\,u\,-\,?} = A$$

5) absorption of a multiconditional uniterm (multiconditional idempotency):

$$\vdash\!\!\!\!-\!\!\!\!\dashv_{S\,;\,S\,;\,\ldots\,;\,S;\;w\,-\,?} = S$$

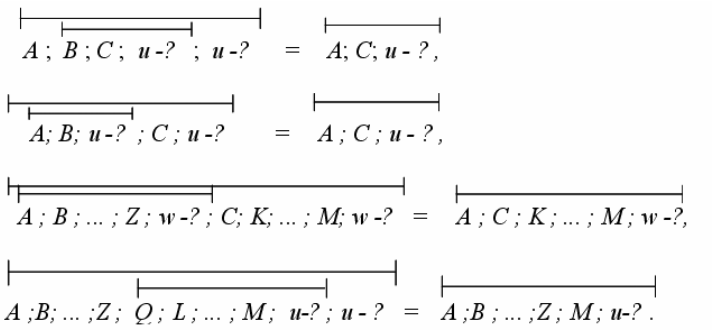
6) selection of a condition:

$$\vdash\!\!\!\!-\!\!\!\!\dashv_{R;\, S\,;\,u_1\,-\,?} \vdash\!\!\!\!-\!\!\!\!\dashv_{R;\, S\,;\,u_2\,-\,?} ;\, u_3 - ? = \vdash\!\!\!\!-\!\!\!\!\dashv_{R;\, S;\; \vdash\!\!\!\!-\!\!\!\!\dashv_{u_1\,-\,?;\, u_2\,-\,?;\, u_3\,-\,?}}$$

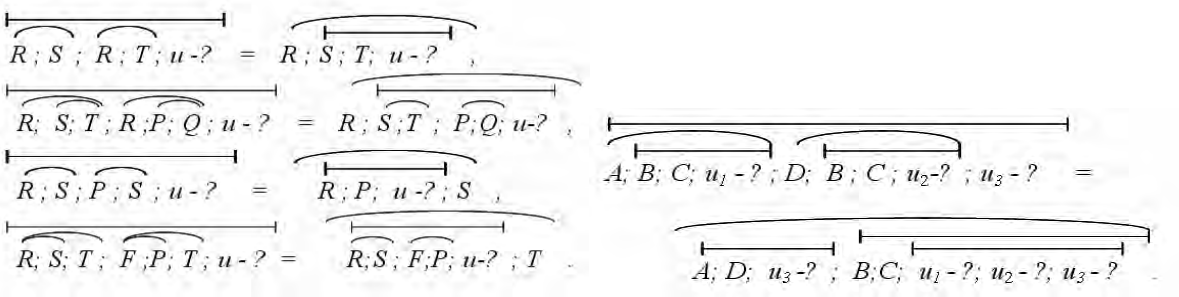
$$\vdash\!\!\!\!-\!\!\!\!\dashv_{R\,;\,S\,;\,\ldots\,;\,Z\,;\,w_1\,-\,?} ;\; \vdash\!\!\!\!-\!\!\!\!\dashv_{R\,;\,S\,;\,\ldots\,;Z\,;\,w_2\,-\,?} ;\, u_3 - ? =$$

$$= \vdash\!\!\!\!-\!\!\!\!\dashv_{R;\, S\,;\,\ldots\,;\,Z\,;\;w_1\,-\,?} = \vdash\!\!\!\!-\!\!\!\!\dashv_{R;\, S\,;\,\ldots\,;\,Z\,;\;w_2\,-\,?} .$$

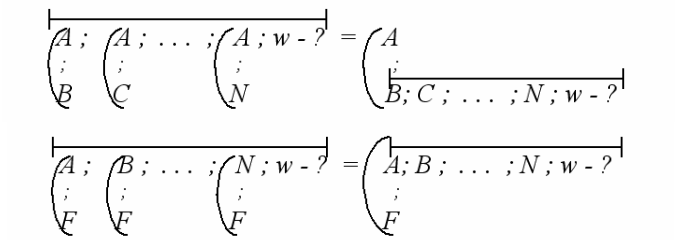
7) absorption of a uniterm (under two eliminations):



8) distributivity:



9) extraction of a uniterm outside the multiconditional elimination operation:

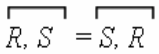


2.2.4. **Paralleling** is an operation having the following properties:

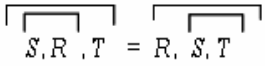
1) idempotency:

2) absorption of the empty uniterm:

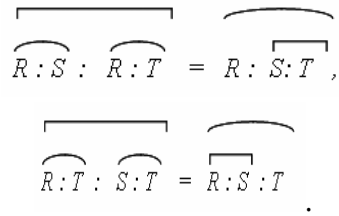
3) commutativity:



4) associativity:

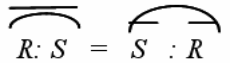


5) extracting a common uniterm:

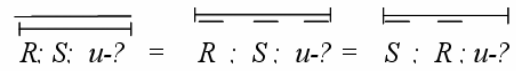


2.2.5. **Reversing** is an operation having the following properties:

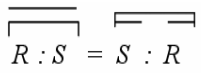
1) reversing of sequencing:



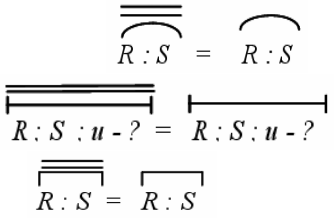
2) reversing of elimination:



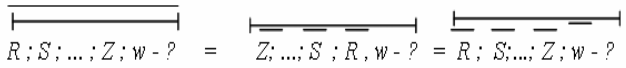
3) reversing of paralleling:



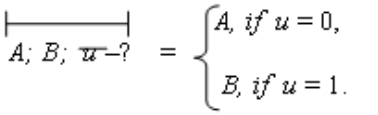
4) double reversing of uniterms:

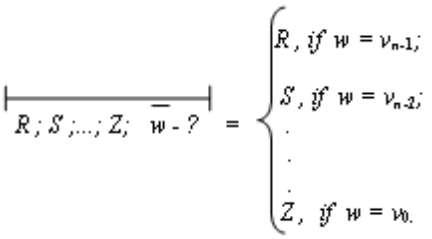


5) reversing of uniterms in multiconditional elimination:



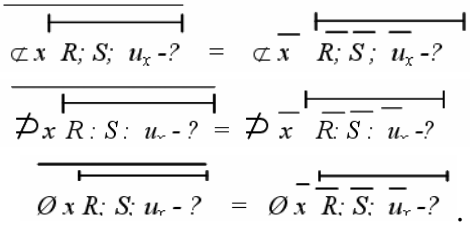
6) selection of a uniterm based on a reversive condition:





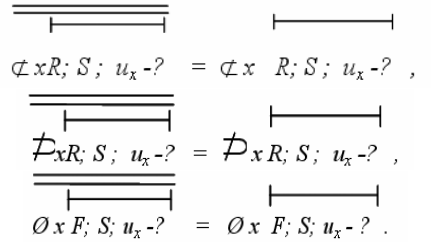
2.2.6. **Cyclic sequencing (⊂), cyclic elimination ( ) and cyclic paralleling (∅)** are cyclic operations having the following properties:

1) reversing of a variable related with a cyclic operation:

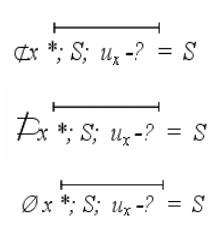


where *x* is the cycle variable and *R* and *S* are the uniterms (repeated in each cycle);

2) double reversing:



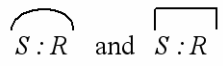
3) empty loop:



2.3. Abstract algorithm

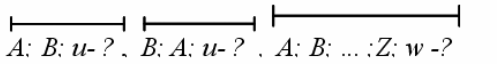
An **algorithm** is defined **abstract** or **formal** if it is composed of one or more (possibly all) expressions specified below, executed in a finite number of iterations (in particular, once).

1. If *S* and *R* are uniterms, then



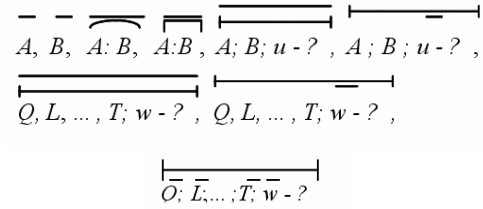
are algorithms.

2. If *A , B, …, Z* are uniterms or algorithms and *u, w* are conditional uniterms with the number of values *w* equal to the number of elimination uniterms under the condition *w,* then



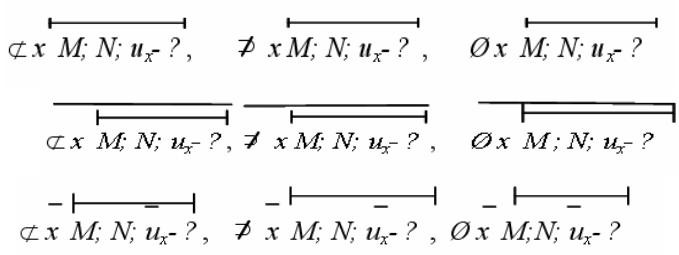
are algorithms.

3. If *A, B, Q, L, … , T* are the uniterms or algorithms and *u, w* are the conditional uniterms with the number of values *w* equal to the number of elimination uniterms under the condition *w,* then



are algorithms.

3. If *M* and *N* are the uniterms or algorithms, *x* is the variable related to the cycle operation and u_x is the condition related to the cycle operation, then



are algorithms.

5. If *F, S, T* are the uniterms or algorithms such that $F = S$ and $S = T$, then $F = T$ is the algorithm.

6. An algorithm can be only such an expression that can be presented in form of a finite number of operations specified in the above items 1 to 5.

The algebra of algorithms is defined as a system of the above specified operations on the uniterms. It provides means for formal transformation of algorithms and their minimization in terms of a minimum number of uniterms.

3. Intuitive explanation of operations

Here we give an intuitive rationale for explanation of the meaning of the operations for our algebra of algorithms, in terms of the classical block-diagram framework.

a. Equalizing is a classical operation not requiring additional explanation.

b. Sequencing is executed according to the block diagram as in Fig. 1.

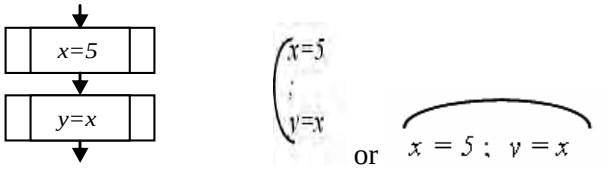


Fig.1. Diagram of a two-block algorithm (a) and its (horizontal and vertical) representations in terms of noncommutative sequencing (b)

c. Elimination is executed according to the block diagram as in Fig. 2.

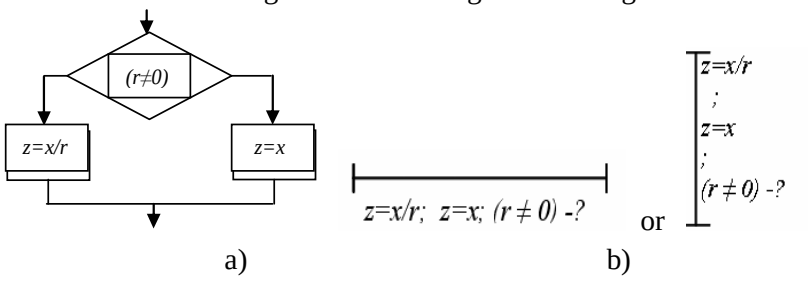


Fig.2. Diagram of an algorithm with a conditional block (a) and its (horizontal and vertical) descriptions in terms of the elimination operation (b)

d. Paralleling is intuitively illustrated in Fig. 3.

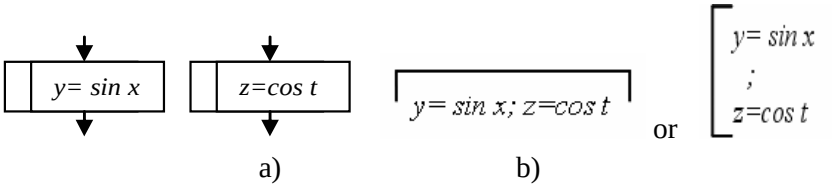


Fig.3. Diagram of two parallel algorithms (a) and their (horizontal and vertical) representations in terms of noncommutative paralleling (b)

e. Reversing is intuitively illustrated for reversing of sequencing in Fig. 4.

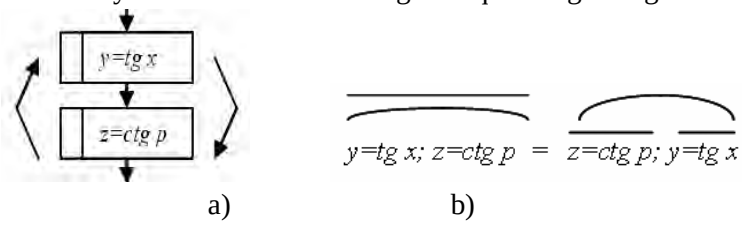


Fig.4. Diagram of a reverse-order algorithm of Fig. 2 (a) and its representation in terms of reversing of sequencing (b)

f. Operations of reversing of elimination and reversing of paralleling can be intuitively explained in a similar way.

g. Cyclic sequencing, cyclic elimination and cyclic paralleling are the operations executed in a classical loop.

4. Example

Algorithm for solving the quadratic equation $ax^2+bx+c=0$, consisting of the sequencing and elimination stages as below.

A) Synthesis of sequences:

- sequence describing calculation of two roots:

$$S_0 = \overbrace{a=w_0, b=w_1, c=w_2}\ ;\ \overbrace{\Delta=w_1^2-4w_0w_2\ ;\ x_1=(-w_1+\sqrt{\Delta})/(2w_0)}\ ;\ x_2=(-w_1-\sqrt{\Delta})/(2w_0)\ ,$$

- sequence describing calculation of a single root (for $\Delta=0$):

$$S_1 = \overbrace{a=w_0, b=w_1, c=w_2\ ;\ x=-w_1/(2w_0)}\ ,$$

- sequence valid for the case $\Delta<0$:

$$S_2 = \overbrace{a=w_0,\ b=w_1\ ,\ c=w_2}\ ;\ M_1\ ,$$

where M_1 is the message "($\Delta<0$) – no real solutions",

- sequence describing calculation of a solution for the case $w_2=0$:

$$S_3 = \overbrace{a=w_0, b=w_1, c=w_2;\ x=-w_1/w_0}\ ,$$

- sequence describing calculation of a solution for the case $w_1=0$:

$$S_4 = \overbrace{a=w_0, b=w_1, c=w_2;\ x=\sqrt{(-w_2/w_0)}}\ ,$$

- sequence valid for the case $(-w_2/w_0)<0$:

$$S_5 = \overbrace{a=w_0, b=w_1, c=w_2}\ ;\ M_2\ ,$$

where M_2 is the message "($w_1=0$ and $(-w_2/w_0) < 0$) – no solution",

- sequence describing calculation of a solution for the case $w_0=0$ and $w_1\neq0$:

$$S_6 = \overbrace{a=w_0, b=w_1\ ,\ c=w_2}\ ;\ x= -w_2/w_1,$$

- sequence valid for the case $w_0=0$ and $w_1=0$:

$$S_7 = \overbrace{a=w_0, b=w_1\ ,\ c=w_2}\ ;\ M_3\ ,$$

where M_3 is the message "($w_0=0$ and ($w_1= 0$)) – no quadratic equation".

B) Synthesis of elimination.

We eliminate all the sequences (S_0, S_1,..., S_7) under a general condition for m:

$$\underbrace{S_0\ ;\ S_1\ ;\ S_2\ ;\ S_3\ ;\ S_4\ ;\ S_5\ ;\ S_6\ ;\ S_7\ ;\ m\ -\ ?}\ . \qquad (1)$$

Firstly, we set a value m_1 for the condition m, for which the sequence S_0 (describing the calculation of two roots) will not be eliminated. Obviously, this concerns the case when the parameters a, b and c are nonzero ($a\neq0$, $b\neq0$ i $c\neq0$) and the discriminant Δ is positive. Fulfillment of the condition ($a\neq0$) will be written as ($a\neq0$)=1. Fulfillment of all other conditions will be denoted in a similar way, e.g. ($b\neq0$)=1 and ($c\neq0$)=1, whereas failing to fulfill a condition will be denoted as e.g. ($b\neq0$)=0. Thus, we have the following sequences:

S_0, if ($a\neq0$)=1 and ($b\neq0$)=1 and ($c\neq0$)=1 and ($\Delta>0$)=1;

S_1, if ($a\neq0$)=1 and ($b\neq0$)=1 and ($c\neq0$)=1 and ($\Delta>0$)=0 and ($\Delta=0$)=1;

S_2, if ($a\neq0$)=1 and ($b\neq0$)=1 and ($c\neq0$)=1 and ($\Delta>0$)=0 and ($\Delta=0$)=0;

S_3, if ($a\neq0$)=1 and ($b\neq0$)=1 and ($c\neq0$)=0;

S_4, if ($a\neq0$)=1 and ($b\neq0$)=1 and ($c\neq0$)=0 and (($-w_2/w_0$) ≥ 0)=1;

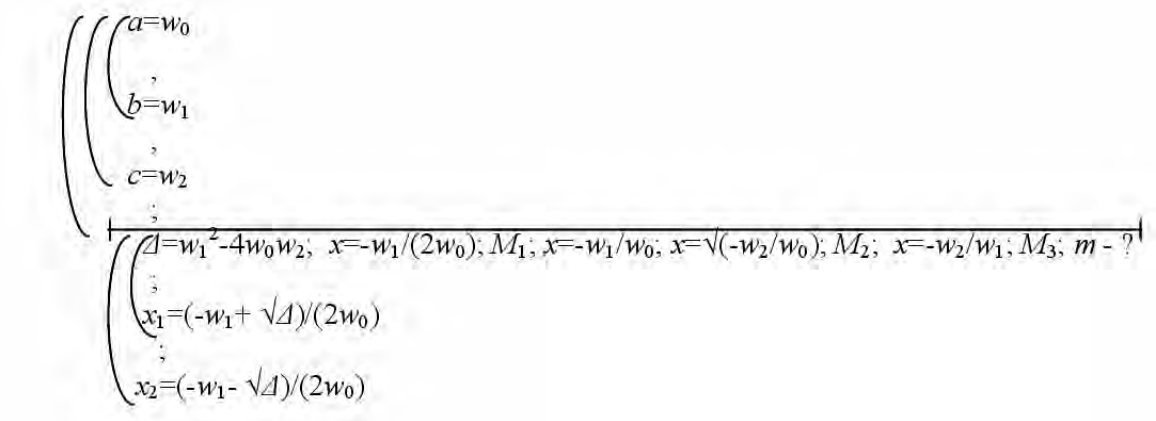
S_5, if $(a\neq 0)=1$ and $(b\neq 0)=1$ and $(c\neq 0)=0$ and $((-w_2/w_0)\geq 0)=0$;

S_6, if $(a\neq 0)=0$ and $(b\neq 0)=1$;

S_7, if $(a\neq 0)=0$ and $(b\neq 0)=0$.

C) Transformation of the algorithm

Allowing for the above specified expressions of the sequences and three-time applying the property of extraction of a uniterm outside the multiconditional elimination operation, the algorithm (1) can be minimized to obtain



Thanks to the introduced multiconditional elimination operation, the algorithm minimization process is essentially simplified as compared to the original one employing the conditional elimination only [21,22]. Specifically, a single multiconditional elimination is used here instead of seven conditional eliminations applied before.

5. Conclusions

In this paper, a number of the existing, intuitive computation models have been recalled, including the virtual machines of Turing, Post, Kolmogorov, Schönhage, Aho-Ullman-Hopcroft as well as the Markov and Krinitski algorithms and the recursive functions, all of them lacking a formal, mathematical presentaion. In pursue of precise mathematical formulation, a universal notion of a uniterm has firstly been introduced. Using the axiomatic method, various operations on the uniterms have been defined, thus contributing to the definition of an abstract or formal algorithm and the introduction of a new idea of the algebra of algorithms. The algebra enables to treat algorithms as mathematical formulae and it provides tools for formal transformation and possible minimization of the algorithms.

An important extension of the elimination operation has been introduced in the paper, namely a multiconditional elimination has been offered. The extension contributes to the reduction of a number of elimination operations and it simplifies the algorithm minimization process, while improving the readability of algorithms.

An example illustrates the efficiency of the proposed theory and the underlying methodology for processing of the algorithms. Some other application examples are presented/reported in a complementary paper [23].

1. Kleene S.C.: Origins of recursive function theory. Annals of the Theory of Computing, vol. 3, No. 1, Jan. 1981, pp. 52-67. 2. Turing A. M.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of London Mathematical Society, series 2, vol. 42 (1936-1937), pp. 230-265; correction, ibidem, vol. 43, pp. 544-546. Reprinted in [13 Davis M., pp. 155-222] and available online at http://www.abelard.org/turpap2/tp2-ie.asp. 3. Kleene S.C.: Turing's analysis of computability, and major applications of it. In Rolf Herken, Editor, The universal Turing machine: A half-century story, Oxford University Press, 1988, pp. 17-54. 4. Post E. L.: Finite Combinatory Processes – Formulation 1.

Journal of Symbolic Logic, 1, pp. 103-105, 1936. Reprinted in The Undecidable, pp. 289ff. 5. De Mol L.: Closing the circle: an analysis of Emil Post's early work. The Bulletin of Symbolic Logic, Vol. 12, Issue 02, June 2006, pp. 267 — 289. 6. Kolmogorov A.N.: On the concept of algorithm (in Russian). Uspekhi Mat. Nauk 8:4 (1953), pp. 175-176; translated into English in Uspensky V.A., Semenov A.L.: Algorithms: Main Ideas and Applications, Kluwer, 1993. 7. Kolmogorov A.N., Uspensky V.A.: On the definition of algorithm (in Russian). Uspekhi Mat. Nauk 13:4 (1958), pp. 3-28; translated into English in AMS Translations 29 (1963), pp. 217-245. 8.Gurevich Y.: Kolmogorov machines and related issues. In G. Rozenberg and A. Salomaa, Editors, Current Trends in Theoretical Computer Science, World Scientific, 1993, pp. 225-234; originally in Bull. EATCS 35 (1988). 9. Schönhage A.: Universelle Turing Speicherung. In J. Dörr and G. Hotz, Editors, Automatentheorie und Formale Sprachen, Bibliogr. Institut, Mannheim, 1970, pp. 369-383. 10.Schönhage A.: Storage modification machines. SIAM Journal on Computing, 9 (1980), pp. 490-508. 11.Markov A.A.: Theory of algorithms (in Russian). Editions of Academy of Sciences of the USSR, vol. 38, 1951, pp. 176-189; translated into English in American Mathematical Society Translations, 1960, series 2, 15, pp. 1-14. 12.Markov A.A., Nagorny N.M.: The Theory of Algorithms (Mathematics and its Applications). Springer, 2001. 13. Church A.: An unsolvable problem of elementary number theory. American Journal of Mathematics, vol. 58 (1936), pp. 345-363. 14. Constable R.L., Smith S.F.. Computational foundations of basic recursive function theory. Theoretical Computer Science, 121, pp. 89-112, Dec. 1993. 15. Sieg W.: Step by recursive step: Church's analysis of effective calculability. The Bulletin of Symbolic Logic, 3:2 (1997), pp. 154-180. 16. Aho A.V, Hopcroft J.E, Ullman J.D.: The design and analysis of computer algorithms. Addison-Wesley Publishing Company, 1974. 17. Krinitski N.A.: Algorithms around us (in Russian). Mir, Moscow, 1988; also translated to Spanish (Algoritmos a nuestro alrededor). 18. Uspensky V.A., Semenov A.L.: Algorithms: Main Ideas and Applications. Kluwer, 1993. 19. Barendregt H.P., The Lambda Calculus: Its Syntax and Semantics, North Holland, Amsterdam, 1984. 20. Davis M., Computability and Unsolvability, Mcgraw-Hill, 1985. 21. Ovsyak V.K., Ovsyak O.V, Ovsyak J.V.: Theory of Abstract Algorithms and Mathematical Modelling of Information Systems (in Polish), Opole University of Technology Press, Opole, Poland, 2005. 22. Ovsyak V.K.: Computation Models and Algebra of Algorithms. http://www.nbuv.gov.ua/Portal/natural/VNULP/ISM/ 2008_621/01.pdf 23. Ovsyak V.K., Latawiec K.J., Ovsyak O.V.: Applications of the algebra of algorithms – a sequential method for the synthesis of formulae of algorithms. Submitted for the CiE'2010 Conference.