Volodymyr OVSYAK [1], Oleksandr OVSYAK [2]
[1] KIELCE UNIVERSITY OF TECHNOLOGY, Al. Tysiaclecia Panstwa Polskiego 7, 25-314, Kielce, Poland;
UKRAINIAN ACADEMY OF PRINTING, Pid Goloskom 19, 79-020, Lviv, Ukraine
[2] NATIONAL UNIVERSITY OF CULTURE AND ARTS, Shuchevitcha 5, 79-020 Lviv, Ukraine

A sequential method for the synthesis of formulae of algorithms

Abstract

New method for the synthesis of algorithm formulae based on the concept of algorithm algebra is presented. The concept of algorithm algebra there is shown in papers [1-4]. The synthesis is performed in two stages. First, all sequences are formulated, which is followed by the bonding the sequences by conditions. This operation is termed by elimination. If the algorithm contains a cyclic operations, then each cyclic operation is considered as a complex one, over which the synthesis of sequences and eliminations are successively performed. Additionally, the algorithm algebra tools for transformation and minimization of algorithms are illustrated by a simple but instructive example.

Keywords: Computation models, algebra of algorithms, synthesis of algorithms, minimization of algorithm.

1. Introduction

In the former papers [1-4], a new concept of the algebra of algorithms has been presented. The algebra defines specific operations executed over the so-called uniterms. Formal expressions describing operations are considered the algorithms. The simplest algorithm is created by the operation of sequencing of two simple uniterms. More complex formulae of algorithms can be synthesized over complex uniterms, being themselves some operations. It is essential that the algebra of algorithms enables to treat the algorithms as mathematical expressions or mathematical formulae which can be transformed and possibly minimized.

A new method for the synthesis of algorithms is introduced in this paper. Since the algebra of algorithms is a new, perhaps controversial approach, we start from an illustrative example which clarifies the essence of the method application, in particular in terms of possible transformation and minimization of algorithms. Then we proceed with a simple generalization procedure for the synthesis of algorithms.

2. Synthesis of algorithms – an example

We refer to [1–4] in which basics of the algebra of algorithms has been presented. To additionally illustrate the algorithm algebra application to the synthesis of algorithms we start from a simple synthesis example and then generalize the synthesis procedure.

Consider an example of finding the greatest common divisor (GCD) for two positive integers x and y, with $x>y$. The algorithm should also include entering the two numbers and checking if they are positive integers, and if the first one is greater than the second one. (Note: possible extension of the algorithm by reversing the numbers when the latter condition is not fulfilled is left for the readers.)

The synthesis is proposed to be performed in two stages. The first stage includes exclusively the construction of sequences, and the second stage encompasses eliminations of sequences. In general, such an approach enables to avoid possible errors in case an algorithm is very complicated.

2.1. Synthesis of sequences

We denote an operation of entering a number x by $x\leftarrow$. In case the entered symbol is not a number (but, for instance, a letter), a message K_1 is displayed. This can described by the sequence (or the algorithm)

$$S_1 = \begin{pmatrix} x\leftarrow \\ ; \\ K_1 \end{pmatrix}$$

In case the entered number x is not a positive integer, an error message K_2 is returned, which can be described by the sequence

$$S_2 = \begin{pmatrix} x\leftarrow \\ ; \\ K_2 \end{pmatrix}$$

The same is checked for the number y, starting from the inspection if the entered symbol is a number, and returning an error message K_3 when it is not, which can be sequenced as

$$S_3 = \begin{pmatrix} x\leftarrow \\ ; \\ \begin{pmatrix} y\leftarrow \\ ; \\ K_3 \end{pmatrix} \end{pmatrix}$$

and when y is not a positive integer, which is messaged by K_4, we have the sequence

$$S_4 = \begin{pmatrix} x\leftarrow \\ ; \\ \begin{pmatrix} y\leftarrow \\ ; \\ K_4 \end{pmatrix} \end{pmatrix}$$

When the numbers x and y are positive integers but $x\leq y$ we have the message K_5 and the sequence is

$$S_5 = \begin{pmatrix} x\leftarrow \\ ; \\ \begin{pmatrix} y\leftarrow \\ ; \\ K_5 \end{pmatrix} \end{pmatrix}$$

For correct numbers x and y, the content of Q cycle of GCD is calculated, which can be described as

$$S_6 = \begin{pmatrix} x\leftarrow \\ ; \\ \begin{pmatrix} y\leftarrow \\ ; \\ Q \end{pmatrix} \end{pmatrix}$$

In an algorithm, the above sequences are mutually eliminated when some conditions are fulfilled or failed. We will describe all the eliminations below.

2.2. Synthesis of eliminations

Sequences S_6 i S_5 are eliminated under the condition $x>y$, which can be described as

$$E_1 = \overline{S_6;\ S_5;\ (x>y) - ?} = \begin{pmatrix} x\leftarrow; \\ ; \\ y\leftarrow \\ ; \\ Q \end{pmatrix} \overline{\begin{pmatrix} x\leftarrow; (x>y) - ? \\ ; \\ y\leftarrow \\ ; \\ K_5 \end{pmatrix}}$$

The expression E_1 and the sequence S_4 are eliminated under the condition $y>0$, which yields the formula

$$E_2 = \overline{E_1;\ S_4;\ (y>0) - ?}$$

E_2 and S_3 are eliminated under the condition that y is a number, which is denoted as $y \epsilon N$, resulting in the formula

$$E_3 = \overline{E_2;\ S_3;\ (y \epsilon N) - ?}$$

Eliminating E_3 and S_2 under the condition $x>0$ we have

$$E_4 = \overline{E_3;\ S_2;\ (x>0) - ?}$$

Eliminating E_4 and S_1 under the condition $x \epsilon N$ finally yields

$$E_5 = \overline{E_4;\ S_1;\ (x \epsilon N) - ?} =$$

$$= \overline{\begin{pmatrix} x\leftarrow; \\ ; \\ y\leftarrow \\ ; \\ Q \end{pmatrix} \overline{\begin{pmatrix} x\leftarrow; (x>y)-? \\ ; \\ y\leftarrow \\ ; \\ K_5 \end{pmatrix} \overline{\begin{pmatrix} x\leftarrow; (y>0)-? \\ ; \\ y\leftarrow \\ ; \\ K_4 \end{pmatrix} \overline{\begin{pmatrix} x\leftarrow; (y \epsilon N)-? \\ ; \\ y\leftarrow \\ ; \\ K_3 \end{pmatrix} \overline{\begin{pmatrix} x\leftarrow; (x>0)-? \\ ; \\ K_2 \end{pmatrix} \begin{pmatrix} x\leftarrow; (x \epsilon N)-? \\ ; \\ K_1 \end{pmatrix}}}}}}$$

2.3. Synthesis of cyclic eliminations

Calculation of GCD n is performed by finding a residue r from the division x/y and comparing the residue to zero. When $r=0$ then y is the GCD. This is described by the sequence

$$S_7 = \begin{pmatrix} r=x/y \\ ; \\ n=y\ . \end{pmatrix}$$

Otherwise ($r\neq 0$) x is set to y and y is set to r, and the division is repeated, which is described by the sequence

$$S_8 = \begin{pmatrix} r=x/y \\ ; \\ x=y \\ ; \\ y=r \end{pmatrix}$$

Eliminating the sequences S_7 and S_8 under the condition $r=0$ we obtain the formulae

$$F = \overline{S_7;\ S_8;\ (r=0)-?}$$

which is performed in the cycle until GCD is obtained. The cycle is executed under the condition $r\neq 0$. Using the symbol $\subset$ for the operation of cyclic sequencing we can describe the cycle and the main part of the algorithm as $\subset(r\neq 0)\ F$, or, in the vertical arrangement, as before

$$Q = \subset(r\neq 0) \overline{\begin{pmatrix} r=x/y\ ; \\ ; \\ n=y \end{pmatrix} \begin{pmatrix} r=x/y\ ; (r=0) - ? \\ ; \\ x=y \\ ; \\ y=r \end{pmatrix}}$$

2.4. Minimization of the algorithm *Q*

Minimization of the algorithm formula Q is understood as a process of transformation of the algorithm, based on the properties of the algebra of algorithms [1], which results in a minimal number of uniterms used.

Recalling the sequences S_7 and S_8 in the subalgorithm F we observe that they both include the uniterm $r=x/y$, which can be extracted from the elimination operation [1]. Applying it to the algorithm formula Q we obtain the Euclidean algorithm in terms of the algebra of algorithms

$$Q_1 = \subset(r\neq 0) \begin{pmatrix} r=x/y \\ ; \\ n=y; \end{pmatrix} \overline{\begin{pmatrix} x=y;\ (r=0) - ? \\ ; \\ y=r \end{pmatrix}}$$

Of course, in terms of the final results of the algorithms we have $Q_1=Q$.

Let us now minimize the algorithm E_1, in which the elimination sequences include the uniterms $x\leftarrow$ and $y\leftarrow$ to be extracted by virtue of distributivity outside the elimination operation [1] yielding

$$E_{1\min} = \begin{pmatrix} x\leftarrow \\ ; \\ y\leftarrow \\ ; \\ \overline{Q_1\ ;\ K_5\ ;\ (x>y) - ?} \end{pmatrix}$$

Now, transforming E_5 in a similar way as for Q we obtain the minimized version of the whole algorithm for calculation of GCD, that is

$$E_{\min} = x\leftarrow \ ; \overline{y\leftarrow \ ; \overline{Q\ ;\ K_5\ ;\ (x>y) - ?\ ;\ K_4\ ;\ (y>0)-?\ ;\ K_3\ ;\ (y \epsilon N)-?}\ ; \overline{K_2\ ;\ (x>0)-?}\ ; K_1\ ;\ (x \epsilon N)-?} =$$

$$= x\leftarrow \ ; \overline{y\leftarrow \ ; \overline{\subset(r\neq 0);\ K_5\ ;\ (x>y) - ?\ ;\ K_4\ ;\ (y>0)-?\ ;\ K_3\ ;\ (y \epsilon N)-?}\ ; \overline{K_2\ ;\ (x>0)-?}\ ; K_1\ ;\ (x \epsilon N)-?}$$

$$r=x/y \ ; \ n=y; \overline{x=y;\ (r=0) - ? \ ; \ y=r}$$

The number of components of the algorithm can be reduced even more if we use the known [2] multi elimination operation. The application of multi-elimination operations gives the following formula of the algorithm finding the greatest common divisor of two integers

$$\left(\begin{array}{l} x\leftarrow \\ ; \\ \left(\begin{array}{l} y\leftarrow \\ ; \\ \not\subset(r\neq 0);\ K_1;\ K_2;\ K_3;\ K_4;\ K_5;\ u_1\text{-}?;\ u_2\text{-}?;\ u_3\text{-}?;\ u_4\text{-}?;\ u_5\text{-}? \\ \left(\begin{array}{l} r=x/y \\ ; \\ n=y;\ \left[\begin{array}{l} x=y;\ (r=0)\text{-}? \\ ; \\ y=r, \end{array}\right. \end{array}\right. \end{array}\right. \end{array}\right.$$

where u_1 is the condition of $x\epsilon N$, u_2 is the condition of $\overline{(x\epsilon N);\ (x>0)}$, u_3 is the condition of $\overline{(x\epsilon N);\ (x>0);\ (y\epsilon N)}$, u_4 is the condition of $\overline{(x\epsilon N);\ (x>0);\overline{(y\epsilon N);\ (y>0)}}$, u_5 is the condition of $\overline{(x\epsilon N);\ (x>0);\ \overline{(y\epsilon N);\ \overline{(y>0);\ (x>y)}}}$.

There is the condition of u_1 in the condition of u_2. Every next condition contains the previous conditions. Recurrence of conditions can be avoided if we conduct their transformations. In particular, these transformations can be carried out as the following

$$\overline{u_1\text{-}?;\ \overline{u_2\text{-}?;\ \overline{u_3\text{-}?;\ \overline{u_4\text{-}?;\ u_5\text{-}?}}}} =$$

$$\overline{\begin{array}{l} (x\epsilon N);\ \overline{(x\epsilon N);\ (x>0);\ (x\epsilon N);\ (x>0);\ (y\epsilon N)} \\ ; \\ (x\epsilon N);\ (x>0);\overline{(y\epsilon N);\ (y>0)};\ (x\epsilon N);\ (x>0);\ \overline{(y\epsilon N);\ \overline{(y>0);(x>y)}} \end{array}} =$$

$$\left(\begin{array}{l} (x\epsilon N) \\ ; \\ \left[\begin{array}{l} *; \left(\begin{array}{l} (x>0) \\ ; \\ \left[\begin{array}{l} *; \left(\begin{array}{l} (y\epsilon N) \\ ; \\ \left[\begin{array}{l} *; \left(\begin{array}{l} (y>0) \\ ; \\ \left[\begin{array}{l} *;\ (x>y) \end{array}\right. \end{array}\right) \end{array}\right. \end{array}\right. \end{array}\right. \end{array}\right. \end{array}\right. \end{array}\right).$$

3. Generalization of sequential method for algorithm synthesis

3.1. Algorithms not containing any cycle operation:

1. Perform the synthesis for all sequences of the algorithm.
2. Perform the synthesis for eliminations of all the sequences of the algorithm.
3. Desirably (but not necessarily), perform the minimization of the algorithm.

3.2. Algorithms containing cycle operation(s):

1. The contents of a cycle operation, which is not nested in another cycle operation, is denoted as a single complex uniterm.
2. Perform the synthesis for all possible sequences over all the uniterms, including the complex uniterm.
3. Perform the synthesis for all possible eliminations of all the sequences.
4. If the complex uniterm contains an unnested cycle operation, then repeat the above items 1 to 3 until the complex uniterm not containing any nested cycle operation is obtained.
5. If the complex uniterm does not contains any nested cycle operation, then repeat the above items 1 and 2.
6. Desirably (but not necessarily), perform the minimization of the algorithm.

4. Conclusions

Based on the concept of the algebra of algorithms [1-4], a new method for the synthesis of algorithms has been proposed. The general method covers algorithms both containing and not containing the cycle operations. The synthesis is two-stage and is first performed on sequencing operations, after which eliminations of sequences are conducted. The treatment of algorithms as mathematical formulae enables to transform and minimize the algorithms, which is of practical interest demonstrated on a simple example.

5. References

[1] Owsiak W., Owsiak A.: Rozszerzenie algebry algorytmów. Pomiary Automatyka Kontrola. No 2, 2010, pp. 184-188.
[2] Ovsyak A.V., Ovsyak V.K.: Modificirowana algebra algoritmov i instrumentalny sredstva obrabotki formul algebry algoritmiv. Upravlajuszczije systemy i maszyny. No 1, 2013. pp. 27–36.
[3] Ovsyak O.: Comparison of algebraic methods for algorithm transform. Pomiary Automatyka Kontrola. No 10, 2013, pp. 1046-1048.
[4] Ovsyak O.: Algebraic models of subsystems of abstract system with the user interface. Pomiary Automatyka Kontrola. No 11, 2013, pp. 1179-1182.

Received: 02.11.2014 *Paper reviewed* *Accepted: 01.12.2014*

Prof. PhD. eng. Volodymyr OVSYAK

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling of systems, computer simulation and mathematical modeling. He works as an professor in Kielce University of Technology.

e-mail: ovsyak@rambler.ru

PhD. eng. Oleksandr OVSYAK

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling of systems, computer simulation and mathematical modeling. He works as an associate professor in National University of Culture and Arts.

e-mail: ovsjak@ukr.net