Automated transformation of algorithms

MODELS OF THE PROCESS OF AN ANALYSIS OF XML-FORMATTED FORMULAE OF ALGORITHMS

Volodymyr Ovsyak, Krzystof Latawiec, Aleksandr Ovsyak

Abstract: *This paper describes two analytical models of the process of an analysis of XML-formatted formulae of algorithms represented in a special editor created for the algebra of algorithms. For identification and storage of types and orientation of selected operations and uniterms, two XML-formatted model algorithms are developed. The ability is shown for transformation of formulae of algorithms that results in 5-time reduction of a number of uniterms while maintaining the functionality of the algorithm.*

Keywords: Algebra of algorithms, algorithm formulae editor, transformation of algorithms.

Introduction

Algebra of algorithms [Ovsyak et all, 2011] provides means to describe algorithms as mathematical formulae. Identity transformations are performed over formulae of algorithms just like those for mathematical expressions. The aim of these transformations is to reduce a number of uniterms and algorithm formulae, thus reducing both a time consumed to build and execute the code and a memory to store the code. An *xml* format is used to describe operations in the algebra of algorithms. Applications of the algebra of algorithms in synthesis and transformation of formulae of algorithms are illustrated by examples of two XML-formatted model algorithms, developed in a formulae editor whose GUI is shown.

Construction of the two model algorithms is supported with four theorems, including the one that establishes the functional equivalence of the two algorithms.

Elements of Editor of Algorithm Formulae

2.1. Main window of editor

Fig.1 shows the main window of the editor of formulae of algorithms, including the menu of commands, operation menu and editing area.

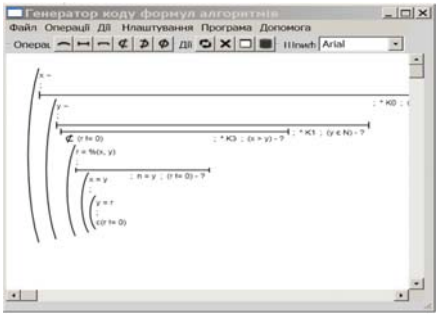


Fig.1. Main window of editor.

2.2. The encoding xml format for formulae of algorithms

Synthesized by the editor of the algorithm's formulas stored in computer memory as files with XML - similar format. For example, sequence of uniterms ***F(x, y)*** and ***S(z)***, to separate them using a semicolon and arrangement - sign of operation of horizontal sequence has the following entries

$$\overbrace{F(x, y); S(z)}$$

and is recorded into computer's memory as follows:

```
<s sep="sem" ori="hor">
    <u>F(x, y)</u>
    <u>S(z)</u>
</s>
```

where the third line is an identifier of description of the sequence operation (*s*), uniterm's separator (*sep* =), a separator (***"sem"***), orientation identifier (*ori* =) and value (***"hor"***), into the following are recorded identifier of the uniterms (*u*) and their significance (***F(x, y)*** oraz ***S(z)***) sequence operation with horizontal orientation of the sign sequence's operation

$$\begin{pmatrix} F(x, y), \\ S(z) \end{pmatrix}$$

and uniterms separator by comma has the following description:

```
<s sep="com" ori="ver">
    <u>F(x, y)</u>
    <u>S(z)</u>
</s>
```

XML - formats of the description of elimination operation (*e*) and parallelization (*p*) has a similar description. Beyond this operation of elimination has three uniterms elimination, separated by semicolons.

Description operation of cyclic sequence (*cs*) with horizontal (***hor***) and vertical (***ver***) orientation and condition of the cycle (***g-?***) and uniterm ***H***, bound with operation of cycle is as follows:

```
<cs ori="hor">
    <u>g-?</u>
    <u>H</u>
</cs>
```

amd

```
<cs ori="ver">
    <u>g-?</u>
    <u>H</u>
<cs>
```

Operation of cycle elimination (*ce*) and parallelization (*cp*) have similar description.

Formulas of Analysis Xml - Format of Algorithms Formulas

For receiving in a computer's editor with XML - format formula algorithms necessary is to identify types of transactions, which are sequence, elimination, parallelization, cyclic sequence, elimination and parallelization and uniterms, data separator character and orientation of operations signs. With a view to optimizing the data from

XML - files formulas algorithms was synthesized algorithms formula (1), where *pu st (w∈@T) Txml (t∈@T, n∈@Nod)* - header of formula algorithm.

```
pu st (w∈@T)Txml(t∈@T, n∈@Nod)=
|
|-- u∈@U;
|   ;
|   u.par=t
|   ;
|   u.val=n.IT
|   ;
|   *
|   ;
|   (n.IT.Le>0)-?
|   ;
|   w=u.
|
|-- s∈@S; A; (n.Na.Eq("s"))-?; (n.Na.Eq("u")-?; Exc("$_xml").; (n≠$)-?
    ;
    s.par=t
    ;
    s.sep=@S.Sep.Sem
    ;
    s.sep=@S.Sep.Com
    ;
    Exc(„Błąd_s xml").
    ;
    n.Attri["sep"].Val.Eq("com")-?                    (1)
    ;
    n.Attri["sep"].Val.Eq("sem")-?
    ;
    s.ori=@S.Ori.Hor
    ;
    s.ori=@S.Ori.Ver
    ;
    Exc(„Błąd_s_O xml")
    ;
    n.Attri["ori"].Val.Eq("ver")-?
    ;
    n.Attri["ori"].Val.Eq("hor")-?
    ;
    s.tA = Txml(s, n.CN[0])
    ;
    s.tB = Txml(s, n.CN[1])
    ;
    w= s.
```

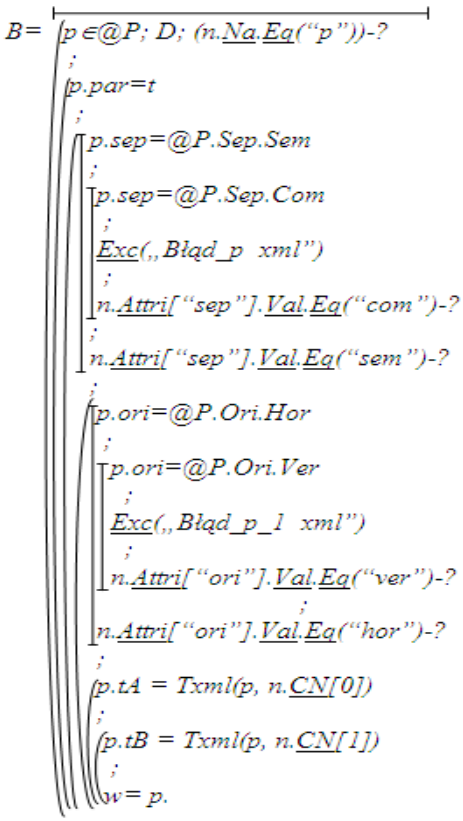
where *pu* - identifier of access, *st* - static property, (*w∈@T*) - is the input parameter w belongs (*∈*) to type *T* subsystem (@) for uniterms processing , *Txml* - name of algorithm's formula, *n∈@Nod*) - input parameters *t* and *n* - type subsystems *T* and *Nod* (abbreviated name of standard subsystem XmlNode [Petzold, 2002, MacDonald, 2008]); *u∈@U* - type variable *U* - subsystem "Uniterm " intended for uniterm processing ; *u.par = t* - attributing to variable par value of input variable *t*; *u.val = n.IT* - attributing variable *val* value of uniterm *n.IT*, selected by standard uniterm InnerText [MacDonald, 2008] from the input variable *n*; *w = u.* - attribution to input variable *w* variable value *u* subsystems *U* and the end (.) of the algorithm implementation; *((n.IT.Le> 0) -?)* - calculation (*n.IT.Le*), using standard uniterm Length [3], the number of uniterm characters chosen by standard uniterm InnerText [MacDonald, 2008] to *xml* – file and verification this number of marks for the majority from zero; *(n≠$)-?* – verification wether *xml* – file is not empty (*$*); *Exc("$_xml").* – view by standart uniterm Exception()[[Petzold, 2002, MacDonald, 2008] report *$_xml* about empty (*$*) *xml* – file and the end (.) of the algorithm implementation; *(N.Na.Eq ("u") -?)* - in input variable *n* search of the name (*Na*) keyword "*u*" string *xml* - file with using standard uniterm Equals() [MacDonald, 2008] compared *Eq("u")* ; (*n.Na.Eq("s "))*-? - compared with the keyword *s*; *s∈@S* - a variable of the subsystem *S*, appointed for working on sequence process; *s.par = t* - attributing value par to value of input variable *t*; *s.sep = @S.Sep.Sem* - attributing variable sep separator *Sem*; *s.sep = @S.Sep.Com* - attributing variable *sep* separator *Com*; *Exc("Błąd_s xml").* - view of error message ("*Błąd_s xml*") in the uniterm separator; *n.Attri["sep"]. (Val.Eq("com") -?)* - in *n* search (*Attri["sep"]*) using standard unitermu Attributes [""] [Ptzold, 2002, MacDonald, 2008] keyword (*sep*) and comparison (*Eq("com")*), using

standard uniterm Equals(""), its value is recorded in *Val* - standard variable Value [Petzold, 2002, MacDonald, 2008], with "*com*"; *(n.Attri ["sep"].Val.Eq("sem") -?)* - comparison with *"sem"*; *s.ori = @S.Ori.Hor* - attributing to variable ori identifier of horisontal orientation *Hor* sign of sequence operation, which is in division *Ori* subsystem *S*; *s.ori = @S.Ori.Ver* - attributing vertical (*Ver*) to orientation sign operation; *Exc("Błąd_s_O xml")* - view of error message in describing of orientation of the sign operation; *(n.Attri["ori"].Val.Eq("ver") -?)* - comparing the value orientation with *ver*; *(n.Attri["ori"].Val.Eq ("hor") -?)* - comparing the value orientation with *hor*; *s.tA = Txml(s, n.CN[0])* - choice (*CN*) uniterm value from position [0] *xml* - description of the algorithm formula using standard uniterm ChildNodes[] [Petzold, 2002, MacDonald, 2008] and the algorithm *Txml()* attribution to variable *tA* first uniterm value; *s.tB = Txml(s, n.CN[1])* - the choice from *xml* - description of algorithm of second uniterm and attributing its to variable *tB*; *w = s.* - attributing to input variable w value of variable s and the end (.) of the algorithm implementation.

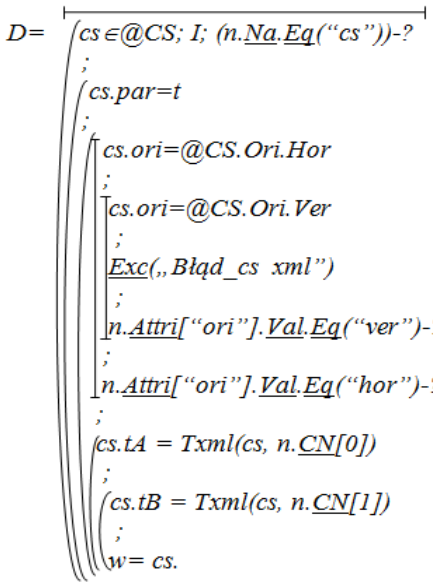
Formula *A* differs from elimination by condition *(n.Na.Eq("s ")-?)* the elimination by condition *(n.Na.Eq ("e ")-?)* contains an identifier *e*; *e∈@E* - the creating variable *e* subsystem *E* working on elimination operation and using variable *e* instead of variable *s* and with missing uniterm processing separator and presence uniterm *cond = Txml(e, n.CN [2])* - intended for attributing variable *cond* value of third uniterm, which is in *xml* - description of the algorithm formula.

```
A= e∈@E; B ; (n.Na.Eq("e"))-?
   ;
   e.par=t
   ;
   | e.ori=@E.Ori.Hor
   | ;
   | e.ori=@E.Ori.Ver
   | ;
   | Exc(,,Błąd_e xml")
   | ;
   | n.Attri["ori"].Val.Eq("ver")-?
   | ;
   | n.Attri["ori"].Val.Eq("hor")-?
   ;
   e.tA = Txml(e, n.CN[0])
   ;
   e.tB = Txml(e, n.CN[1])
   ;
   cond=Txml(e, n.CN[2])
   ;
   w= e.
```

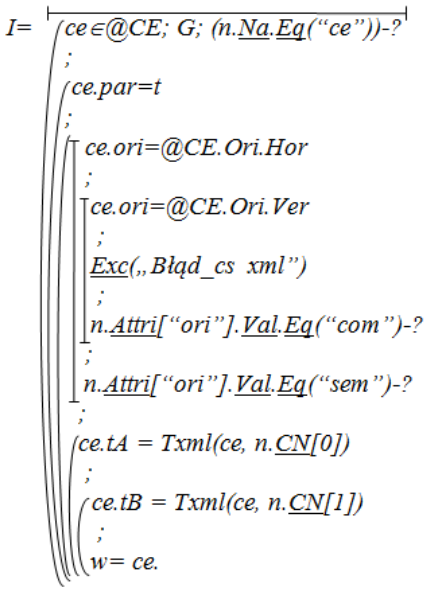
Formula *B* from elimination by condition (*n.Na.Eq("s")-?*) difference is in: in comparison by condition (*n.Na.Eq("p")-?*) identifier *p* is used; *p∈@P* - creating of variable *P* subsystem for working on paralellization and using variable *p* instead of the variable *s*.



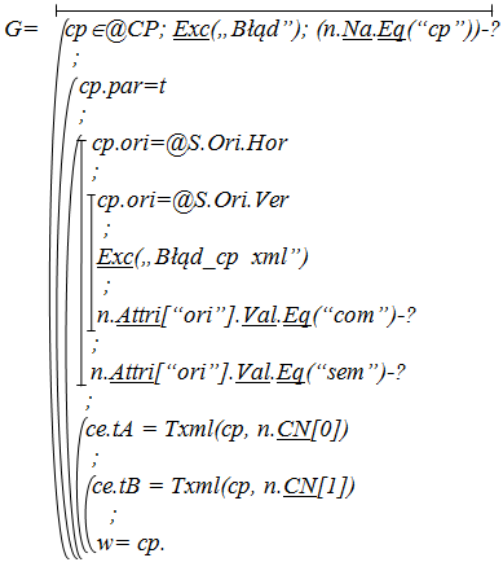
Formula *D* from elimination by condition (*n.Na.Eq("e")-?*) differs that: the elimination by condition (*n.Na.Eq ("cs")-?*) identifier *cs* is used; *cs*∈*@CS* - the creating variable *cs* subsystem *CS*, appointed for working on process of cyclic sequence and using variable *cs* instead variable *e*.



Formula *I* from elimination by condition *(n.Na.Eq("cs"))-?* differs that: in comparison by condition *(n.Na.Eq("ce"))-?* identifier *ce* is used; *ce*∈*@CE* – the creating variable *ce* subsystem *CE*, appointed for workion on cyclic elimination operation and using variable *ce* instead variable *cs*.



formula *G* from *I* differs by identifier and variable *cp subsystem CP*, appointed for working on operation of cyclic parallelization.



Theorem 1. If *F* is XML - formula of algorithm described by format, then formula of algorithm (1) is described the identification in *F* operation algebra algorithms, their orientation, uniterms separators and uniterms selection and detection of errors in XML - formulas describing algorithms.

Proof. Elimination by condition *(n ≠ $) -?* checking whether the input variable n is not empty *xml* - description. If the variable is empty, the elimination by this condition is obtained by uniterm *Exc("$ _xml")*. with information (*$ _xml*) about error in the input variable *n*. Otherwise elimination by condition *n.Na.Eq("u") -?* checking whether the line of the variable n contains uniterm identifier ("*u*"). If yes, then after the creating of variable u type subsystem for uniterms processing *U* and attributing for variable par value of abstract input variable t, checking whether the variable n is not empty (*(n.IT.Le> 0)-?*). Not empty value of the input variable is chosen (*n.IT*) and is attributed to variable *val* (*u.val = n.IT*). Then the output variable (*w*) is attributing (*w = u*) value of variable *u*, which is the value of uniterm (abstract or recorded by *xml* - format). This is ending the algorithm implementation. In case the

condition is not executed *n.Na.Eq("u") -?* is the checking (*n.Na.Eq ("s")*-?) wether string of variable n contains the name of identifier sequence transaction (*s*).

If the identifier xml (the description of sequence operation) is recognized - is creating variable sequence operations ($s \in @S$). Uniterm ($s.par = t$) is attributing variable *par* value of input variable t. Elimination by condition *n.Attri["sep"].Val.Eq("sem") -?* compared (*Eq("sem")*) wether value of attribute sep (*n.Attri["sep"]*) concordant with the name of the uniterms separator (*sem*). If convergence than variable sep is attributed ($s.sep = @S.Sep.Sem$) *Sem* value. When it is not the convergence , then is checking (*n.Attri["sep"].Val.Eq("com")-?*) wether com is separator. If yes, then the variable sep is attributing ($s.sep = @S.Sep.Com$) *Com* value. Otherwise, there is a view (*Exc("Błąd_s xml")*) of error in the description of the separator.

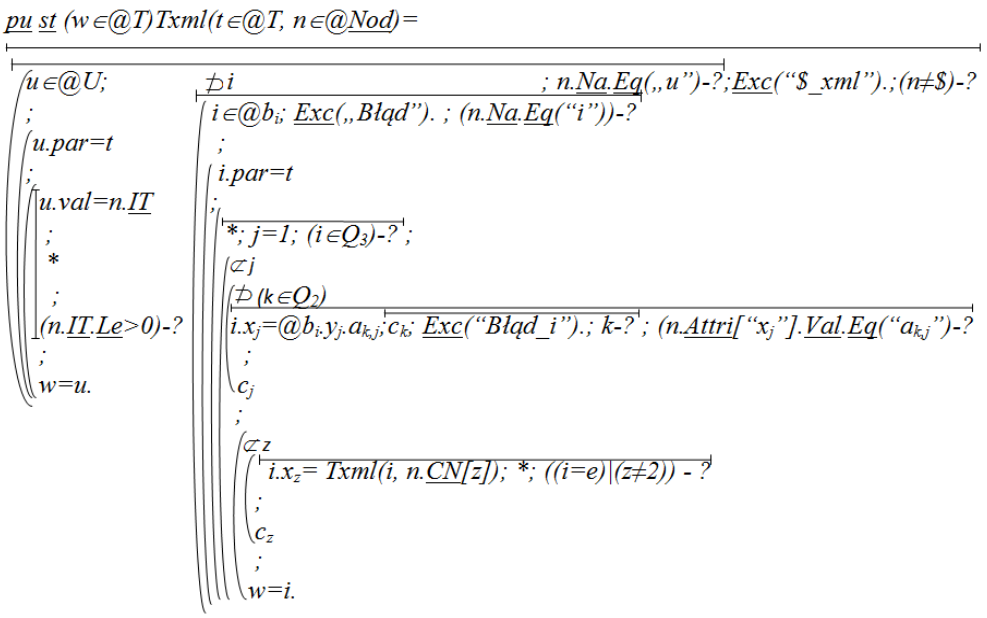
After identification and recording separator in elimination by conditions *n.Attri["ori"]. Val.Eq("hor") -?* and *n.Attri["ori"].Val.Eq("ver") -?* performed the same identification and recording *(s.ori = @S.Ori.Hor* and *s.ori = @S.Ori.Ver)* orientation for sequence operation *s.ori=@S.Ori.Ver*.

Next from *xml* - description is chosen line with uniterm (*n.CN[0]*), getting the uniterm using algorithm (*Txml()*) and attributing it to variable *tA*. Similarly is received second uniterm (*tB*) from *xml* - description. The output variable w is attributed value of variable *s* by the last uniterm ($w = s.$) .

Thus it is proved that the algorithm (1) describes the identification in the XML - description sequence operation, its orientation, uniterm separator and also choice of uniterms and attributing this data to output variable.

Similarly we can prove description by the algorithm (1) identification and read data from XML - description data of elimination operation, parallelization, cyclic sequence, elimination and parallelization. Theorem proved

Theorem 2. If *F* is a description of the algorithm in XML - format, the formula of the algorithm (2)



, (2)

where

$$i \in Q_0 = \overbrace{s;\ e;\ p;\ cs;\ ce;cp}\ ;\ b_i \in Q_1 = \overbrace{S;\ E;\ P;\ CS;\ CE;\ CP}\ ;\quad j,\ k \in Q_2 = \overbrace{0;\ 1}\ ;$$

$$Q_3 = \overbrace{s;\ p}\ ;\ x_j \in Q_4 = \overbrace{sep;\ ori}\ ;\ y_j \in Q_5 = \overbrace{Sep;\ Ori}\ ;$$

$$a_{k,j} \in Q_6 = \begin{cases} \overbrace{Hor;\ Ver}\ ; \\ ; \\ \overbrace{Sem;\ Com}\ , \end{cases}\ t_i = \begin{cases} tA, \text{ jeśli } i \in Q_7 = \overbrace{s,\ e,\ p}\ , \\ cond, \text{ jeśli } i \in Q_8 = \overbrace{cs,\ ce,\ cp;} \end{cases}\ r_i = \begin{cases} tB, \text{ jeśli } i \in Q_7\ , \\ t, \text{ jeśli } i \in Q_8, \end{cases}$$

$$x_z = \begin{cases} t_i, \text{ jeśli } z=0; \quad z \in Q_9 = \overbrace{0;1;2} \\ r_i, \text{ jeśli } z=1; \\ cond, \text{ jeśli } z=2, \end{cases}$$

is describing an identification in *F* operations algebra algorithms, their orientation, uniterm separators and uniterm choice and view of errors in *xml* – description of algorithm formulas.

Proof. Elimination by condition *(n ≠ $) -?* as well as formula (1) contains an elimination by condition *n.Na.Eq ("u") -?* and uniterm about error (*Exc ("$ _xml ")*.). Execution of condition goes to attributing to the output variables, as well as in formula (1), an abstract value or the readout uniterm XML - format.

Let the variable *i* has value *s*. Then bi has the value of *S*. Uniterms *i*∈@ b_i, *Exc("Błąd").* and *(n.Na.Eq("i ")-?),* elimination by condition *(n.Na.Eq (" i ")-?),* are as follows: *s*∈@*S, Exc("Błąd").* and *(n.Na.Eq("s ")-?).* All of them coincide with uniterm formula (1) for operation sequence.

Uniterm *i.par* = *t* i after replacement i to s has the form *(s.par* = *t)* which is the same as in formula (1). Elimination by condition *(i*∈Q_3*) -?* gives empty uniterm (*), which can be left out because it cannot change the formula.

The variables *j* and *k* get the value 0. Variables x_0, y_0, $a_{0,0}$ are getting values of *sep, Sep, Sem,* which gives elimination by condition *n.Attri["x_j"].Val.Eq("$a_{k,j}$")-?* and formula:

$$\begin{array}{l} s.sep=@S.Sep.Sem \\ \quad ; \\ c_k \\ \quad ; \\ Exc(\text{“Błąd\_s xml”}). \\ \quad ; \\ ((0+1) \in Q_2)\text{-?} \\ \quad ; \\ (n.Attri[\text{“sep”}].Val.Eq(\text{“Sem”})\text{-?} \end{array} \qquad (3)$$

Condition *(0*∈Q_2*)* -? cyclic elimination is performed, thats why the value of variable increases by 1 and becomes an returning to the cycle by variable *k*. We get expressions for variables uniterms x_0, y_0, $a_{1,0}$ operation sequence *sep, Sep, Com*. Substituting them to formula (3) we obtain the following expression:

$$
\begin{array}{l}
s.sep=@S.Sep.Sem \\
\quad ; \\
\quad s.sep=@S.Sep.Com \\
\quad\quad ; \\
\quad\quad c_k \\
\quad\quad\quad ; \\
\quad\quad \underline{Exc}(\text{“Błąd\_s”}). \\
\quad\quad\quad ; \\
\quad\quad ((1+1) \in Q_2)\text{-?} \\
\quad ; \\
\quad (n.\underline{Attri}[\text{“sep”}].\underline{Val}.\underline{Eq}(\text{“Com”})\text{-?} \\
; \\
(n.\underline{Attri}[\text{“sep”}].\underline{Val}.\underline{Eq}(\text{“Sem”})\text{-?}
\end{array}
$$

In the last expression condition *(1 +1)* $\in Q_2$ *-?* not performed because elimination by this condition can be replaced on uniterm *Exc("Błąd_s").*, which gives the formula:

$$
\begin{array}{l}
s.sep=@S.Sep.Sem \\
\quad ; \\
\quad s.sep=@S.Sep.Com \\
\quad\quad ; \\
\quad\quad \underline{Exc}(\text{“Błąd\_s”}). \\
\quad\quad\quad ; \\
\quad\quad (n.\underline{Attri}[\text{“sep”}].\underline{Val}.\underline{Eq}(\text{“Com”})\text{-?} \\
\quad ; \\
(n.\underline{Attri}[\text{“sep”}].\underline{Val}.\underline{Eq}(\text{“Sem”})\text{-?}
\end{array}
\qquad (4)
$$

Now variable of cyclical sequence *j* is increased by 1, and variable operation of cyclical elimination *k* becomes to the initial value 0. Then for variables x_1, y_1, a_0, we obtain such a value *ori, Ori, Hor* and for $a_{1,1}$ we have a *Ver*. Similarly, as we get the expression (4), we get the formula:

$$
\begin{array}{l}
s.ori=@S.Ori.Hor \\
\quad ; \\
\quad s.ori=@S.Ori.Ver \\
\quad\quad ; \\
\quad\quad \underline{Exc}(\text{“Błąd\_s”}). \\
\quad\quad\quad ; \\
\quad\quad (n.\underline{Attri}[\text{“ori”}].\underline{Val}.\underline{Eq}(\text{“Ver”})\text{-?} \\
\quad ; \\
(n.\underline{Attri}[\text{“ori”}].\underline{Val}.\underline{Eq}(\text{“Hor”})\text{-?}
\end{array}
$$

Now the variable *z* cycle takes the initial value of 0 and $x_z = t_i = tA$. Then condition $((s = e) \mid (0 \neq 2))$ -? elimination is performed, which allows elimination from uniterm *s.tA = Txml(i, n.CN [0])* and makes returning to the cycle for the variable *z*, which takes the value 1. The second iteration $x_z = r_i = tB$, and the elimination condition $((s = e) \mid (1 \neq 2))$ -? performed, giving the uniterm *s.tB = Txml (s, n.CN[1]*). On the third iteration *z* = 2. Condition $((s = e) \mid (2 \neq 2))$ -? of elimination is not performed. Therefore there is receiving an empty uniterm, which can be left. From the last line of formula (2) we get *w = s.*. Thus obtained formula is concordant with the corresponding fragment of the formula (1):

$$
\begin{array}{l}
(s.tA = Txml(s,\ n.\underline{CN}[0]) \\
\quad ; \\
\quad (s.tB = Txml(s,\ n.\underline{CN}[1]) \\
\quad\quad ; \\
\quad w = s.
\end{array}
$$

Thus it is shown that formula (2) in XML - description identifies data of sequence operation. Similarly, it can be proved also for the operation of elimination , and for operations of description cycles and parallelization. Theorem is proved.

Comparison Formulas Of Algorithms

As it was proven formulas (1) and (2) describe the same process of analysis XML - the description formulas of algorithms. Lets compare formulas of the algorithms (1) and (2) due tue the number of uniterms. Formula (1) contains 90 uniterms, while formula (2) has only 26 uniterms. Thus formula (2) has 3.5 times uniterms less.

Theorem 3. From formula (2) is derived formula (1).

The proof of the theorem is similar to the proof of Theorem 2.

Theorem 4. From formula (1) is derived formula (2).

The proof is based on the axiom operations of elimination descriptions cycles of algebra algorithms.

5. Conclusion

Description of algorithms as formulas algorithms provides performance identical transformations of algorithms, which reduced expenses needed for implement algorithms.

Bibliography

[Ovsyak et all, 2011] V. Ovsyak. Computation models and algebra of algorithms. Submitted to the Conference.

[Petzold, 2002] C. Petzold. Programming Microsoft Windows with C#., 2002.

[MacDonald, 2008] M. MacDonald. Pro WPF in C# 2008 Windows Presentetion Foundation with .NET 3.5.

Authors' Information

Volodymyr Ovsyak *– Full Professor, Department of Electrical, Control and Computer Engineering, University of Technology, Box: 31, Sosnkowskiego, Opole 45-272, Poland, e-mail: ovsyak@rambler.ru*

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling

Krzysztof Latawiec *– Full Professor, Department of Electrical, Control and Computer Engineering, University of Technology, Box: 31, Sosnkowskiego, Opole 45-272, Poland, e-mail: lata@po.opole.pl*

His research interests concentrate on system identification, multivariable control, adaptive and robust control (also in networks) and fractional systems.

Aleksandr Ovsyak *– Phd. National University of Culture and Arts, The L'viv Campus, Box: 5, Kuszewicza, L'viv, Ukraine; e-mail: : ovsjak@ukr.net*

He specializes in theoretical and applied computer science, theory of algorithms, programming, information systems, mathematical modeling of systems, computer simulation and mathematical modeling.