Krassimir Markov, Vitalii Velychko, Oleksy Voloshin
(editors)

Information Models of Knowledge

I T H E A®
K I E V – S O F I A
2010

Krassimir Markov, Vitalii Velychko, Oleksy Voloshin (ed.)

Information Models of Knowledge

ITHEA®

Kiev, Ukraine – Sofia, Bulgaria, 2010

ISBN 978-954-16-0048-1

First edition

Recommended for publication by The Scientific Concil of the Institute of Information Theories and Applications FOI ITHEA

ITHEA IBS ISC: 19.

This book maintains articles on actual problems of research and application of information technologies, especially the new approaches, models, algorithms and methods fot information modeling of knowledge in: Intelligence metasynthesis and knowledge processing in intelligent systems; Formalisms and methods of knowledge representation; Connectionism and neural nets; System analysis and sintesis; Modelling of the complex artificial systems; Image Processing and Computer Vision; Computer virtual reality; Virtual laboratories for computer-aided design; Decision support systems; Information models of knowledge of and for education; Open social info-educational platforms; Web-based educational information systems; Semantic Web Technologies; Mathematical foundations for information modeling of knowledge; Discrete mathematics; Mathematical methods for research of complex systems.

It is represented that book articles will be interesting for experts in the field of information technologies as well as for practical users.

General Sponsor: Consortium FOI Bulgaria (www.foibg.com).

Printed in Ukraine

ISBN 978-954-16-0048-1

C\o Jusautor, Sofia, 2010

SOME PROPERTIES OF ACYCLIC COMPOSITIONAL PROGRAMS

Tetiana Parfirova, Vadim Vinnik

Abstract: *Structure, behaviour and properties of acyclic programs are investigated in methodological and mahematical framework of compositional programming. Relations of acyclic programs to sequantial-parallel and chain programs (defined in the previous authors' papers) are investigated: any acyclic program can be transformed to an equivalent sequential-parallel program with extra usages of basic subroutines but without extra names (informally, enlarging the executable code but not enlarging data); acyclic program can be transformed to an equivalent chain program with extra names but without extra basic sunroutine usages (informally, this enlarges data but does not not enlarge the executable code); sequential composition of two acyclic programs can be thansformed to an equivalent acyclic program with either complex internal interconnections (though without extra subroutines) or with one additional subroutine (though with simple interconnections).*

Keywords: *sequential composition, parallel composition, compositional programming, nominal function, acyclic program, sequential and parallel compositions.*

ACM Classification Keywords: *D.1.4 Sequential Programming, D.2.4 Software/Program Verification, F.3.2 Semantics of Programming Languages.*

Introduction

Previous papers [Parfirova 2010a, Parfirova 2010b] cover structure, properties and behavior of a certain simple form of programs where subroutines work sequentially, data flow in one direction from predecessors to successors, without loopback. It was noticed that this principle is intrinsic to the large-scale structural level of information processing systems, in particular to software systems that support business activity and electronic document processing, organizational activity, education etc. Except this, it was shown [Parfirova 2009] that sequential type of structure and its corresponding principle of operation is the most transparent implementation of the reference model of interaction in terms of entity platform [Redko 2008a, Redko *et al.* 2008]. Two particular cases, sequential-parallel and chain systems were described and investigated in [Parfirova 2010b]; it was shown that systems of the first kind can be transformed to the second kind, and the transformation doesn't introduce extra usages of subroutines to the compositional term that represents the system but requires extra names (used for temporary copies of data). The general model of acyclic programs is described in [Parfirova and Vinnik 2010]. The goal of this paper is to reveal some properties of acyclic programs and their relations to the previously investigated classes.

Basic Notions: Compositional Programming

According to compositional programming (CP) and entity platform created on its basis [Redko 1978, Redko 1998, Redko 2008, Redko *et al.* 2008], data objects are modelled by means of nominal sets (NS), whereas programs and their parts as data transformers are modelled as nominal functions (NF).

Sets of all *names* and all *detonates* are designated as $\mathbf{V}$ and $\mathbf{D}$ respectively. V-NS is a finite functional[1] binary relation $\alpha \in \mathbf{D}^V$ where $V \subseteq \mathbf{V}$. Note that, when α is a V-NS, $v \in V$ is some name, $\alpha(v)$ is the value of that name in the NS. Let us denote the set of all NSs as $\mathbf{N}$.

By definition, NF is a unary partial function of the form $\mathbf{N} \tilde{\to} \mathbf{N}$. NF is called V-ary if $\operatorname{dom} f \subseteq \mathbf{D}^V$, and (V,W)-ary if, except this, $\operatorname{rang} f \subseteq \mathbf{D}^W$, for some $V,W \subseteq \mathbf{V}$. Not all NFs have an arity. Function that has some arity is called polyary. Here and further we investigate polyary NFs only.

[1]There is also a branch of CP called *compositional-nominative approach* where functionality of NS is not required [Nikitchenko, 1999; Nikitchenko, 2009].

Restriction is a $(V, U \cap V)$-ary NF $\upharpoonright_U^V$ (where $U, V \subseteq \mathbf{V}$, vertical bar means restriction of a relation by its 1st component on a given set):

$$\upharpoonright_U^V(\alpha) = \alpha \mid U = \{(u,d) \mid (u,d) \in \alpha \wedge u \in U\},\ \alpha \in \mathbf{D}^V .$$

For some mapping $\xi : U \to V$, renaming is a (V,U)-ary NF $\updownarrow^\xi$ (where $U, V \subseteq \mathbf{V}$):

$$\updownarrow^\xi(\alpha) = \{(u, \alpha(\xi(u))) \mid u \in U\},\ \alpha \in \mathbf{D}^V .$$

Substantially, this operation replaces names $v \in V$ in NS α with new names $u \in U$ such that $\xi(u) = v$. It should be noted that ξ does not have to be a one-to-one mapping. It means that one name from α can turn after renaming into several new names. From the definition, it directly follows that if $\mathrm{pr}_1\, \alpha = V \subseteq \mathrm{pr}_2\, \xi = W$ (where $\mathrm{pr}_i\, \rho$ means projection of a relation ρ by its i-th component) then $(\upharpoonright_W^V \circ \updownarrow^\xi)(\alpha) = \xi \circ \alpha$ where $\circ$ means multiplication of unary functions (taking into account that NS α can be ragarded as a function that maps names to their denotates).

The identical mapping of some set X into itself is designated as $\mathrm{id}_X = \{(x,x) \mid x \in X\}$.

Overlapping is a binary operation such that, for any NSs α and β,

$$\alpha \nabla \beta = \beta \cup \{(u,d) \mid (u,d) \in \alpha, u \notin \mathrm{pr}_1(\beta)\}.$$

Operations on nominal functions are called *compositions*. Substantially, compositions model composing a complex program from simpler programs, or subroutines. Two fundamental compositions will be important in the next sections: multiplication (model of sequential execution) and overlapping (models parallel execution) defined as follows:

$$(f \circ g)(\alpha) \cong g(f(\alpha)),$$
$$(f \nabla g)(\alpha) \cong f(\alpha) \nabla g(\alpha).$$

In the second formula, ∇ in the left side denotes a composition (an operation on nominal functions) whereas in the right side it means operation on nominal sets; this should not confuse the reader because the context of this operation is always clear. Let NFs f and g be (U_1, V_1)-ary and (U_2, V_2)-ary respectively. Then $f \circ g$ is a (U_1, V_2)-ary NF if $V_1 = U_2$ and totally undefined NF otherwise; $f \nabla g$ is $(U, V_1 \cup V_2)$-ary NF if $U_1 = U_2 = U$ and totally undefined otherwise.

Sequential-Parallel and Chain Functions

Notions of sequential-parallel and chain functions were first introduced in [Parfirova 2010a, Parfirova 2010b]. Let us briefly remind the main definitions and theorems that will be important int this work.

Let Φ be some set of NFs. NF h is called *sequantial-parallel* in basis Φ if it can be obtained from functions of the set Φ and functions $\upharpoonright_U^V$, $\updownarrow^\xi$, $\mathrm{id}_{\mathbf{D}^V}$ by means of compositions $\circ$ and ∇.

Link with the kernel f is NF of the form

$$h = \mathrm{L}_\zeta^\xi(f) = \updownarrow^\xi \nabla \left(\upharpoonright_{\mathrm{pr}_2(\zeta)}^{\mathrm{pr}_2(\xi)} \circ \updownarrow^\zeta \circ f\right),$$

where ξ, ζ are renaming mappings, $\xi : U \to V$, $\zeta : U' \to V'$, where $U, V, U', V' \subseteq \mathbf{V}$. Instantially, the definition of the link means that some part of the input data, after selecting an appropriate part and renaming ζ, is passed to the kernel function f, and the rest of input data is passed to the output through a renaming ξ. The overlapping of these two parallel branches gives the link's output. Note that ranaming ζ is applied to the whole input NS without excluding any names; except this, renaming can transform any input name to two or more output names. This means that renaming ζ can be used to create "backup copies" of the input data, see fig. 1.

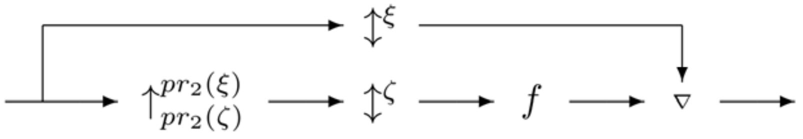

Fig. 1. Link

Chain function in basis Φ (where Φ is a set of NFs) is a function of the form $h = h_1 \circ h_2 \circ \ldots \circ h_n$ where every h_i ($i = 1, \ldots, n$) is a link with some kernel $f \in \Phi$.

It is proved [Parfirova 2010a, Parfirova 2010b] that any, say (U,V)-ary, sequential-parallel NF f in basis Φ can be transformed into a chain function $\hat{f}$ in the same basis with some arity $(U,\hat{V})$ where $V \subseteq \hat{V}$, that is equivalent to f up to extra names, i.e. $f = \hat{f} \circ \uparrow_V^{\hat{V}}$.

Acyclic Functions

Let us remind definitions [Parfirova and Vinnik 2010]. For an acyclic program consisting of n subroutines, let us consider the input of the whole program as the output of its imaginary 0-th subroutine; in a dual way, the output of the whole program is considered as the input of its imaginary $(n+1)$-th subroutine. The determinative feature of acyclic programs is that relation *act after* (or *use outputs of*) is a partial ordering on the set of all its subroutines. This order can be considered as linear: for each i, j where $0 \le i < j \le n+1$, data transfer channel from the output of the i-th subroutine to the input of the j-th subroutine exists if and only if $i < j$.

Suppose we have (U_i,V_i)-ary NFs, $i = \overline{1,n}$, a set of names V that will be also designated V_0, a set of names U that will be also designated U_{n+1}, and renaming mappings ξ_{ij} for all i,j where $0 \le i < j \le n+1$. The mappings ξ_{ij} can be regarded as an upper-right triangle matrix Ξ (this matrix has $n+1$ rows and as many columns; row indices start with 0, and column indices start with 1). The following restrictions are imposed:

$$V_i \supseteq \bigcup_{j=i+1}^{n+1} \mathrm{pr}_2(\xi_{ij}),\ i = \overline{0,n}.$$

$$U_j = \bigcup_{i=0}^{j-1} \mathrm{pr}_1(\xi_{ij}),\ j = \overline{1,n+1}.$$

By definition, *acyclic NF* is NF $f = \mathrm{T}_{V,U}^{\Xi}(f_1, \ldots f_n)$ such that for any V-ary NS α its value $\beta = f(\alpha)$ is defined by the following equations (fig. 2) where $\alpha_0 = \alpha$, $\beta = \beta_{n+1}$:

$$\beta_k = \mathop{\nabla}_{i=0}^{k-1} (\xi_{ik} \circ \alpha_i), \text{ for } k = \overline{1,n+1},$$

$$\alpha_k = f_k(\beta_k), \text{ for } k = \overline{1,n}.$$

Note that α_i is the value of f_i (for $0 \le i \le n$), and β_i (for $1 \le i \le n+1$) is the argument of f_i.

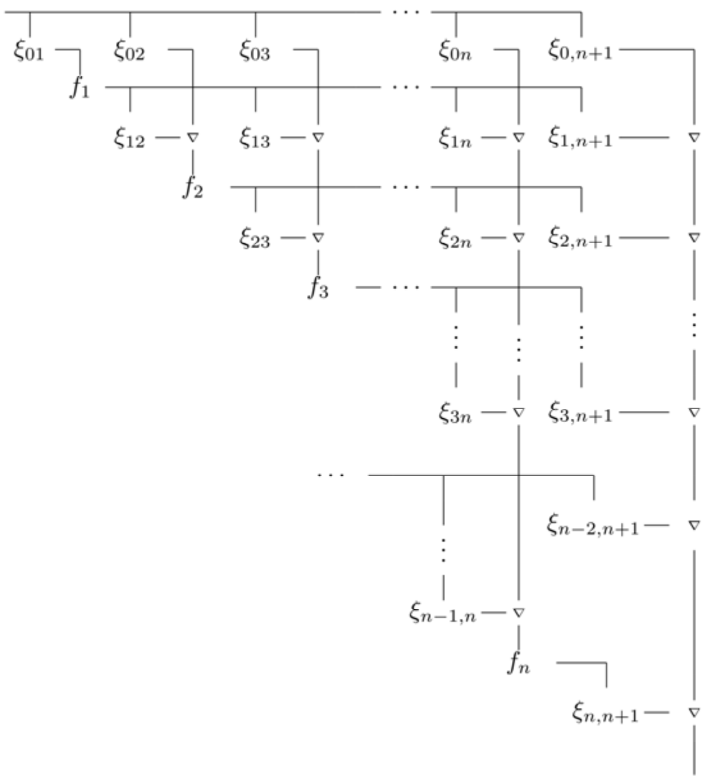

Fig. 2. Structure of an acyclic program, general case

Relation of Acyclic NFs to Sequentially-Parallel NFs

Considering features of acyclic NFs in general would require lots of technical details, so we will demonstrate these properties on some substantial examples.

Statement 1. Not every acyclic NF is sequential-parallel — see fig. 3 with parameters listed in table 1.

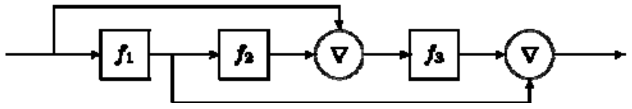

Fig 3. Example of an acyclic NF, that is not sequential-parallel

Table 1: Arities U_i, V_i and renamings ξ_{ij}.

$i \backslash j$	1	2	3	4	V_i
0	$id_{\{x,y\}}$	$\varnothing$	$id_{\{y\}}$	$\varnothing$	$\{x,y\}$
1	—	$id_{\{x\}}$	$\varnothing$	$id_{\{x\}}$	$\{x\}$
2	—	—	$id_{\{x\}}$	$\varnothing$	$\{x\}$
3	—	—	—	$id_{\{y\}}$	$\{y\}$
U_i	$\{x,y\}$	$\{x\}$	$\{x,y\}$	$\{x,y\}$	—

Statement 2. Every acyclic NF can be transformed into an equivalent sequential-parallel NF. This transformation increases number of basic functions usages in the compositional term but does not introduce extra names. For the acyclic NF from the previous example, we have the following equivalent sequentially-parallel NF (see fig. 4):

$$f = f_1 \nabla \left(\left(\uparrow^{\{x,y\}}_{\{x\}} \nabla \left(f_1 \circ f_2\right)\right) \circ f_3\right)$$

Apparently this sequentially-parallel NF has one extra usage of NF f_1.

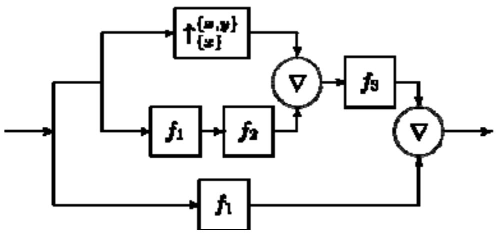

Fig 4. Sequentially-parallel NF equivalent to the acyclic NF

Statement 3. Every acyclic NF could be transformed, without introducing extra usages of basic functions to the compositional term but possibly introducing extra names, into an equivalent (up to extra names) chain NF.

Note that a weaker statement is trivial: indeed, every acyclic NF can be transformed to an equivalent sequential-parallel NF due to statement 2 (with extra usages of basic functions), and the later can be transformed to an equivalent chain NF due to the main theorem from [Parfirova 2010b]. To avoid extra usages of basic functions, however, we shoud transform acyclic NF to chain NF directly. For the given example, the equivalent chain NF (where parameters ξ_i and ζ_i are shown in the table 2) is:

$$g = \mathrm{L}_{\zeta_1}^{\xi_1}(f_1) \circ \mathrm{L}_{\zeta_2}^{\xi_2}(f_2) \circ \mathrm{L}_{\zeta_3}^{\xi_3}(f_3) \circ \uparrow_{\{x,y\}}^{\{x,x',y\}}.$$

Table 2. Transformation to the chain NF

i	ξ_i	ζ_i
1	$\{x \mapsto x, y \mapsto y\}$	$\{x \mapsto x, y \mapsto y\}$
2	$\{x' \mapsto x, y \mapsto y\}$	$\{x \mapsto x\}$
3	$\{x' \mapsto x, x \mapsto x', y \mapsto y\}$	$\{x \mapsto x, y \mapsto y\}$

This example is notable because it uses only one auxiliary name x'. On the other hand, we are interested in a general method of transformation acyclic NFs to chain NFs. Let us explain the main idea of the method. Suppose some f_i (including special case $i = 0$ for the very input) has an output name, say x. Then the $(i+1)$-th link of the chain NF should rename x to x_i. Except this, every (say, k-th) link must preserve all names of the form x_j, $0 \le j < k$, from its input. Therefore, the output of the k-th link consists of copies x_j of outputs of all previous subroutines and own outputs of f_k. Thus, if the j-th subroutine uses output x of the i-th subroutine when $i < j-1$ (i.e. if $\xi_{ij}(y) = x$ for some name y), the j-th link of the chain should rename x_i into y before applying f_k. The only remaining special issue is the last link that restores names from the form x_i and assembles the final result.

For the example above, this principle gives the following chain function with parameters shown in the table 3.

$$g = \mathrm{L}_{\zeta_1}^{\xi_1}(f_1) \circ \mathrm{L}_{\zeta_2}^{\xi_2}(f_2) \circ \mathrm{L}_{\zeta_3}^{\xi_3}(f_3) \circ \mathrm{L}_{\zeta_4}^{\xi_4}(\mathrm{id}_{\mathbf{D}^{\varnothing}}) \circ \uparrow_{\{x,y\}}^{\{x_0,y_0,x_1,x_2,x,y\}}.$$

Table 3. Transformation to the chain NF

i	ξ_i	ζ_i
1	$\{x_0 \mapsto x, y_0 \mapsto y\}$	$\{x \mapsto x, y \mapsto y\}$
2	$\{x_0 \mapsto x_0, y_0 \mapsto y_0, x_1 \mapsto x\}$	$\{x \mapsto x\}$
3	$\{x_0 \mapsto x_0, y_0 \mapsto y_0, x_1 \mapsto x_1, x_2 \mapsto x\}$	$\{x \mapsto x, y \mapsto y_0\}$
4	$\{x_0 \mapsto x_0, y_0 \mapsto y_0, x_1 \mapsto x_1, x_2 \mapsto x_2, x \mapsto x_1, y \mapsto y\}$	$\varnothing$

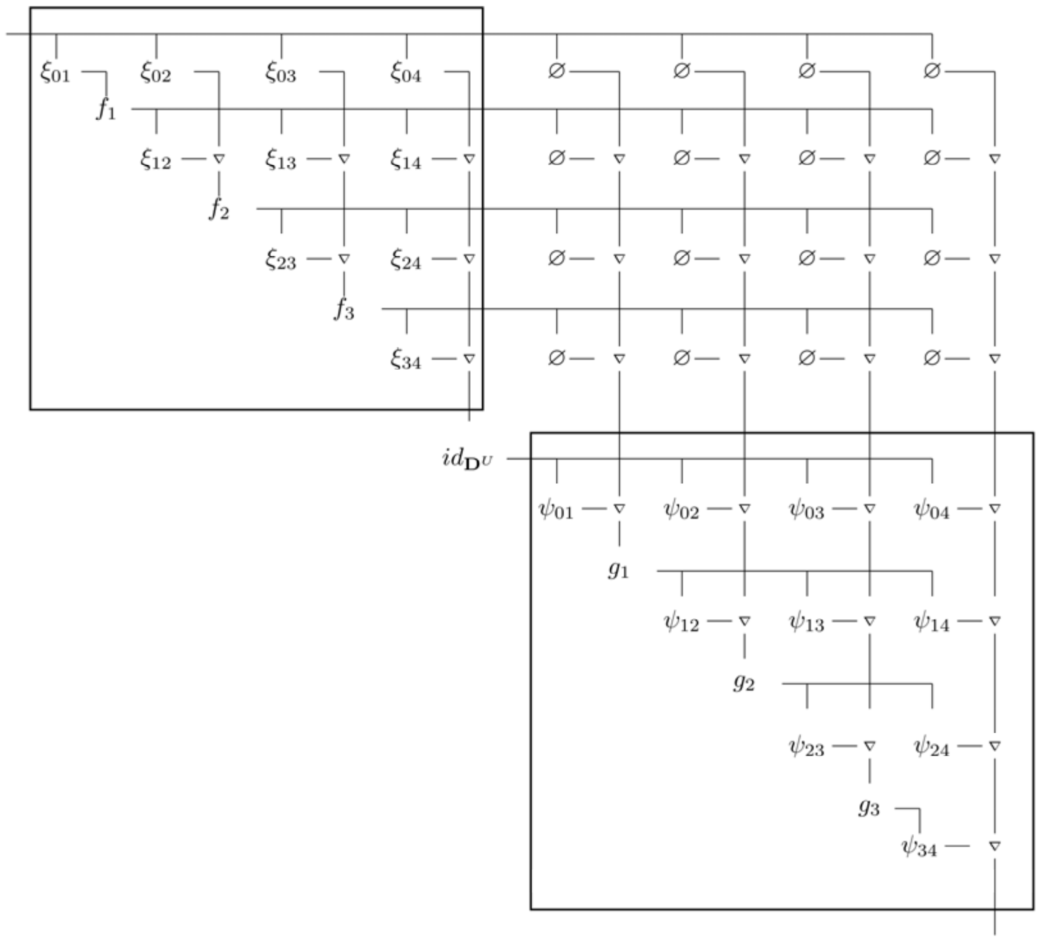

Fig. 5. Sequential composition of acyclic programs as an acyclic program

Statement 4. If f and g are both acyclic functions then their sequential composition $f \circ g$ could be presented as an acyclic function as well. Let $f = \mathrm{T}^{\Xi}_{V,U}(f_1,\dots,f_m)$, $g = \mathrm{T}^{\Psi}_{U,W}(g_1,\dots,g_n)$; then (see fig. 5) $f \circ g = h = \mathrm{T}^{\mathrm{Z}}_{V,W}(f_1,\dots,f_m,\mathrm{id}_{\mathbf{D}^U},g_1,\dots g_n)$ with the following matrix of renamings:

$$Z = \begin{bmatrix} \xi_{01} & \cdots & \xi_{0,m+1} & \varnothing & \varnothing & \varnothing \\ & \ddots & \vdots & \varnothing & \varnothing & \varnothing \\ & & \xi_{m,m+1} & \varnothing & \varnothing & \varnothing \\ & & & \psi_{01} & \cdots & \psi_{0,n+1} \\ & & & & \ddots & \vdots \\ & & & & & \psi_{n,n+1} \end{bmatrix} = \begin{bmatrix} \Xi & \varnothing \\ & \Psi \end{bmatrix}.$$

In other words,

$$\zeta_{ij} = \begin{cases} \xi_{ij}, & \text{if} \quad 0 \le i < j \le m+1, \\ \varnothing, & \text{if} \quad 0 \le i \le m, m+2 \le j \le m+n+2, \\ \psi_{i-m-1,\, j-m-1}, & \text{if} \quad m+1 \le i < j \le m+n+2. \end{cases}$$

From this example one can see that $f \circ g$ can be transformed to an equivalent acyclic function h in such a way that the renaming matrix of h is very simple but one additional (though trivial) subroutine is introduced to h. It can be shown that there is another option: no extra subroutines are introduced to h but the renaming matrix is not so simple. This should be regarded as an example of a common trade-off between simplicity of internal structure of a system vs. simplicity of interfaces between its subsystems.

Conclusion

The composition $\mathrm{T}_{V,U}^{\equiv}$ described above models a sequential data and control flow between parts of a program that does not contain loops. Its particular cases are sequantial programs, entirely parallel programs and various intermediate program structures that combine both parallel and sequential execution of statements. Finally, the model allows equivalent transformations by clear algebraic laws known from CP.

Bibliography

[Parfirova 2010a] T.S. Parfirova. Compositional model of sequential-parallel connections in information systems. Proceedings of the 10th international seminar "Discrete mathematics and its applications", Moscow State University, 198–201, 2010 (in Russian).

[Parfirova 2010b] T.S. Parfirova. On reduction of sequential-parallel compositional programs to entirely sequential form. Bulletin of Kyiv National Taras Shevchenko University, series: Physical and mathematical sciences, 1, 132–137, 2010 (in Ukrainian).

[Parfirova 2009] T.S. Parfirova. The notion of interaction from the perspective of the entity platform. Proceedings of the 6th international conference "Theoretical and applied aspects of program system development", Kyiv, 78–80, 2009 (in Russian).

[Parfirova and Vinnik 2010] T. Parfirova, V. Vinnik. Copmositional Model of Acyclic Programs. Proceedings of the International Conference on Computer Science and Engineering CSE'2010, Košice – Stará Ľubovňa, Slovakia (accepted for publication).

[Redko 1978] V.N. Redko. Compositions of programs and compositional programming. Programming and Computer Software. 5, 3–24, 1978 (in Russian).

[Redko 1998] V.N. Redko. Compositional structure of programmology. Cybernetics and Systems Analysis. 4, 47–66, 1998 (in Russian).

[Redko 2008] V.N. Redko. Existential foundations of compositional paradigm. Cybernetics and Systems Analysis. 2, 3–12, 2008 (in Russian).

[Redko et al. 2008] V.N. Redko, I.V. Redko, N.V. Gryshko. Programmological foundations of entity platform. Problems of Programming. 2–3, 75–83, 2008 (in Russian).

[Nikitchenko 1999] N.S. Nikitchenko. Compositional-nominative approach to clarifying the notion of program. Problems of Programming. 1, 16–31, 1999 (in Russian).

[Nikitchenko 2009] N.S. Nikitchenko. Compositional-nominative aspects of address programming. Cybernetics and Systems Analysis. 6, 24–35, 2009 (in Russian).

Authors' Information

Tetiana Parfirova *– Kiev National Taras Shevchenko University, faculty of Cybernetics, 01601, Volodymyrska str, 64, Kiev, Ukraine; e-mail tetiana.parfirova@gmail.com*

Major Fields of Scientific Research: Theory of programming, formal semantics of programming languages, educational environments, e-learning systems.

Vadim Vinnik *– Kiev National Taras Shevchenko University, faculty of Cybernetics, 01601, Volodymyrska str, 64, Kiev, Ukraine; e-mail vadim.vinnik@gmail.com*

Major Fields of Scientific Research: Theory of programming, formal semantics of programming languages, program specification and verificaztion, complex systems modelling and simulatio