Binary Dynamic Models of Structural Synthesis of Programs

G.A. Oparin, V.G. Bogdanova and A.A. Pashinin
Matrosov Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, Russia
prn51@icc.ru, bvg@icc.ru, apcrol@gmail.com

***Abstract* – In this paper, we propose a matrix-vector Boolean differential model for constructing plans of computational actions in solving non-procedural problems on a computational model of a modular software system. The conditions of the problem of structural synthesis of a program from pre-implemented modules are formulated as a system of Boolean differential equations, the solutions of which, under given initial conditions (consistent with the non-procedural formulation of the problem), determine the solvability of the problem and give constructive plans for its solution (including parallel ones). The proposed method of structural synthesis is focused on high-dimensional models. It allows highly efficient software implementation due to the high parallelism of performing vector-matrix operations on binary vectors at the level of machine instructions. The developed model is used for planning computations in packages of applied microservices based on the HPCSOMAS-MSC platform.**

Keywords – computational model; Boolean differential equations; abstract program; action planning

I. INTRODUCTION

There is a wide class of complex subject areas in which the process of computer solving of problems is represented as a step-by-step execution of modules-procedures from the set *M*, working on a common field of variables *Z*, which are the actual parameters of these procedures. Typical representatives of software systems with such an organization are computing software packages with functional content in the form of libraries of autonomously compiled modules (subroutines/functions written in the traditional sequential programming languages Fortran/C), as well as tool environments for organizing distributed computing, which allows integrating geographically remote heterogeneous autonomous computational resources into one resource-intensive multidisciplinary task.

A distinctive feature of this kind of applied computation is as follows: the variables from *Z* in these computations act as distinguished concepts of the theory and concurrent parameters of the problem statement and therefore have applied independent semantics. Many problems within some applied theories consist of the fact that it is required to find the required quantities from $B_0 \subset Z$, using other quantities $A_0 \subset Z$ whose values are considered known. The solution to such a computational problem $T = (A_0, B_0)$ (if it exists) is the composition of functions extracted from the computational model as a network of functional relations of the computability of quantities in the considered theory. It is not required to know how its constituent functions (i.e., modules from *M*) are calculated to prove the existence of a solution - it is enough to know only the names of these functions (the set of operation symbols *F*) and associated with these names the sets of input and output parameter names from *Z*. The proof of existence must be constructive. It means that in the course of the proof a problem solving plan must be constructed (an abstract program).

The transformation of a plan into a specific program depends on the specific capabilities of the hardware and software of the computational environment and is not considered in this article.

The article is organized in the following way. Section II provides a brief overview of the problem of structural program synthesis. Section III describes the knowledge base of the abstract program planner. Section IV is devoted to the static Boolean planning model. Section V contains the equations of the dynamic planning model. Section VI discloses the technology of applying the developed theoretical results on the example of a small computational Boolean network. Section VII describes the means of automating the management of computations based on a dynamic model. Section VIII contains a conclusion on the results achieved during the study.

II. RELEATED WORK

Automated generation of plans for solving problems in various subject areas is one of the main areas of research in artificial intelligence, which has been discussed for a long time in foreign and domestic literature [1]. The problem of obtaining abstract programs or, in other words, plans for computational actions on a computational domain model is known as the planning problem in the structural synthesis of programs.

A new wave of increased interest in the idea of program synthesis is mainly associated with the understanding of the possibility of representing a synthesis problem in the form of a Boolean satisfiability problem [2, 3], for which efficient algorithms for its solution are developed.

Structural program synthesis is a method of constructing programs from pre-implemented modules that are treated as functions. For the first time, the idea of structural synthesis of programs was presented in a report

[4] at a conference on algorithms in modern mathematics and computer science, organized by A. Ershov and D. Knuth in 1979. More recently, this idea stimulates the creation of new modern integrated environments for the development of domain-specific software based on the structural synthesis of programs. For example, CoCoVila [5] supports visual and simulation software development and uses structural program synthesis to translate declarative specifications of simulation tasks into executable code. The HPCSOMAS platform [6] provides automated development of applied microservice packages (AMP) for distributed scientific computations. The CLAVIRE framework [7] represents a set of high-level tools for solving e-Science problems within the data-driven approach. Based on iPSE (Intelligent Problem Solving Environment), it allows one to interact with the user based on domain-specific knowledge.

The traditional planning technique in the structural synthesis of programs is deductive means of proving theorems based on the methods of mathematical logic [8]. Our approach to the synthesis of abstract programs (both parallel and sequential) is based on the systematic use of binary dynamical systems as the main formalism of the internal representation for the computational domain model. The search for a plan under these conditions is reduced to constructing particular solutions for such dynamical systems. AI planning languages are not considered in this paper.

III. Planner Knowledge Base

As the knowledge base (KB) of the planner, the computational model $KB = (F, Z, In, Out)$ is used, where Z is a finite set of symbols (names) of parameters (attributes, values of the subject area), and *F* is a finite set of symbols of *k*+*p* arity operations (*k* and *p*, generally speaking, are different for different operation symbols). $In \subset F \times Z$ and $Out \subset F \times Z$ define relationships between operations and parameters at the input and output, respectively. Each symbol of the operation $F_i \in F$ of arity $k_i + p_i$ is connected to the set $in(F_i) \subset Z$ from k_i of input parameters and a set $out(F_i) \subset Z$ from p_i of output parameters associated with it. In terms of content, the operation $F_i \in F$ implies the possibility of calculating variables $out(F_i)$ from variables $in(F_i)$ using some program module M_i from the library of modules *M*.

It is assumed that the KB has a high level of internal parallelism and is redundant in the sense that only a part of the operations from *F* is used to solve the problem, and (or) the problem *T* has several alternative solution plans.

Relationships of both *In* and *Out* are conveniently specified as two Boolean matrices *A* and *B* with dimensions $n \times m$, whose elements are formed as follows: $a_{ij} = 1$ ($b_{ij} = 1$) if the parameter Z_j is input (output) for the operation F_i. The matrices *A* and *B* will be referred to as the input and output matrices, respectively. Rows of matrices A_i and B_i are the binary representation of the set of input parameters $in(F_i)$ and output ones $out(F_i)$ of the operation F_i, respectively. Similarly to the relations *In* and *Out*, it is also convenient to represent the required input-output relation of the problem statement $T = (A_0, B_0)$ in the form of *m*-dimensional Boolean strings a_0 and b_0, assuming that the *j*-th element of these strings takes the value equal to 1 if the parameter Z_j is input (output) in the problem statement *T*.

IV. Static Boolean Planning Model

Let us associate the parameter vectors *Z* and the operation *F* with the Boolean vectors *z* and *f*, the values of the components of which (z_i and f_i) will be determined as follows: $z_i = 1$, if the value of the parameter Z_i is known (given or calculated), and $z_i = 0$ - otherwise, $f_i = 1$, if the operation F_i is executed and $f_i = 0$ - otherwise.

Let us assume that over Boolean matrices and vectors, the elementwise operations of disjunction $\vee$, conjunction $\wedge$ (hereinafter, this sign is replaced by a dot or omitted completely), negation $\neg$, addition modulo two $\oplus$, and implication $\rightarrow$ are defined.

Let A^T be the matrix transposed to *A*. Let *E* be the column, all elements of which are equal to one. The symbols $\otimes$ and $\circ$ will also mean the $\vee-$ and $\wedge-$product of two matrices:

$$C = A \otimes B, \text{ where } c_{ik} = \bigvee_j a_{ij} b_{jk},$$
$$C = A \circ B, \text{ where } c_{ik} = \bigwedge_j (a_{ij} \vee b_{jk}).$$

Taking into account the introduced notation, the relations reflecting the relationship of operations with input and output parameters can be written as a system of two matrix Boolean equations

$$\begin{aligned} &\bar{A} \circ z \rightarrow f = E, \\ &B^T \otimes f \rightarrow z = E. \end{aligned} \tag{1}$$

A distinctive feature of system (1) is its large dimension and strong sparseness of the matrices *A* and *B*. Along with the original system we need the derivative from system (1), called the inverse:

$$\begin{aligned} &f \leftarrow B \otimes z = E, \\ &z \leftarrow A^T \otimes f = E. \end{aligned} \tag{2}$$

In the vector-matrix form, the statement of the problem $T = (A_0, B_0)$ corresponds to a Boolean equation of the form

$$\bar{a}_0 \circ z \rightarrow \bar{b}_0 \circ z = 1. \tag{3}$$

Due to the equivalence of the calculus of functional dependencies to a fragment of the propositional calculus, it is easy to show that the problem $T = (A_0, B_0)$ is solvable in the KB if and only if equation (3) is a logical consequence of the system of equations (1) in the sense that all solutions of system (1) are solutions of equation (3).

The vector-matrix Boolean formalism of representation of the computational model KB proposed in this section allows:

- Introduce a certain standardization in this class of models;
- Create on a regular basis methods for analyzing the structural features of models;
- Obtain (due to the introduction of two types of variables – Boolean vectors *z* and *f* into the model) a plan for solving the problem directly in terms of operation symbols;
- Create efficient planning algorithms due to the high parallelism of matrix-vector operations on binary vectors.

Systems of Boolean equations (1) and (2) can be used in logical derivation in implementing a strategy for moving from initial states to target states ("forward wave" - system (1)) and from target states to initial ones ("reverse wave" - system (2)). However, the representation of the problem conditions in the form of systems of Boolean equations (1, 2) allows the creation of a new approach to constructing plans for solving non-procedural problems based on dynamic Boolean planning models.

V. Dynamic Boolean Planning Model

The vector-matrix model of the subject area considered in Section IV accurately reflects the statics of the modular system (its structure and functional relationships between the input and output actual parameters of operations) but does not allow describing the dynamics of the system behavior in time in solving non-procedural problems.

A well-known mathematical apparatus for describing the dynamics of the behavior of logical systems is Boolean differential calculus [9, 10]. This section proposes new logical models of modular software complexes in the form of systems of nonlinear Boolean differential equations. Such systems of equations allow effective numerical methods for constructing the necessary particular solutions consistent with the problem formulation (3).

The transition from static Boolean equations (1, 2) to dynamic ones is carried out by reducing the implication operation to the time domain, namely: interpreting the implication $x \to y$ as a cause-and-effect dependence of *y* on *x* and considering the variables *x* and *y* as functions of discrete time $t \in \{0,1,2,...\}$, the logical equation $x \to y = 1$ is put in correspondence a pair of Boolean differential equations:

$$\begin{aligned} \dot{x}(t) &= 0, \\ \dot{y}(t) &= \bar{y}(t) \cdot x(t), \end{aligned} \tag{4}$$

where $\dot{y}(t)$ is the time derivative of the function $y(t)$ defined by the following relation:

$$\dot{y}(t) = y(t) \oplus y(t+1) .$$

For $\dot{x}(t) = \dot{y}(t) = 0$, we receive the original equation $x \to y = 1$, whose solutions determine the set of possible equilibrium states for (4). Replacing cause-and-effect relationships with temporal ones makes it possible to more clearly present the structural features of the functioning of a modular computational system and dispense with the use of inference tools in solving non-procedural problem formulations. In a certain sense, equations of type (4) combine the original logical relations and the means of logical inference for them.

Following the methodology just considered, we assign to the system of matrix Boolean equations (1) a system of matrix Boolean differential equations of the form:

$$\begin{aligned} \dot{f}(t) &= \bar{f}(t) \cdot (\bar{A} \circ z(t)), \\ \dot{z}(t) &= \bar{z}(t) \cdot (B^T \otimes f(t)). \end{aligned} \tag{5}$$

For system (2), the matrix Boolean differential equations will be as:

$$\begin{aligned} \dot{f}(t) &= \bar{f}(t) \cdot (B \otimes z(t)), \\ \dot{z}(t) &= \bar{z}(t) \cdot (A^T \otimes f(t)). \end{aligned} \tag{6}$$

Systems of equations (5) and (6) describe the dynamics of the main and auxiliary bipartite computational Boolean networks, respectively.

In (5, 6) the column ($f(t)$, $z(t)$) is the state vector of these Boolean dynamical systems at time *t*.

Taking into account the definition of the derivative, we construct partial solutions $z(t, a_0)$, $f(t, a_0)$ of the systems (5) and $z(t, b_0)$, $f(t, b_0)$ of the systems (6) under the initial conditions $z(0) = a_0^T$, $f(0) = 0$ and $z(0) = b_0^T$, $f(0) = 0$, respectively. With the passage of time ($t \le n + m$), these particular solutions, as nondecreasing functions of time, reach equilibrium states (that is, there are no cyclic attractors). Let us denote these equilibrium states as $z_b(a_0)$, $f_b(a_0)$ for system (5) and $z_a(b_0)$, $f_a(b_0)$ for system (6). Let us introduce the notation $z^* = z_b(a_0) \cdot z_a(b_0)$ and $f^* = f_b(a_0) \cdot f_a(b_0)$. Then the following assertions can be made:

1. If $z_b(a_0) \ge b_0$, then the problem $T = (A_0, B_0)$ is solvable on the computational model KB and the time derivatives $\dot{f}(t, a_0)$, $\dot{z}(t, a_0)$ of particular

solutions $f(t,a_0)$, $z(t,a_0)$ represent the so-called preliminary plan for solving problem T deployed in time. The components of the vector-functions $\dot{f}(t,a_0)$, $\dot{z}(t,a_0)$ are either identically equal to zero, or represent elementary functions of the form:

$$\delta_j(t)=\begin{cases}1, t=j\\ 0, t\neq j\end{cases}.$$

From a meaningful point of view, the equality $\dot{f}_i(t,a_0)=\delta_j(t)$ means that the operation F_i is activated at time $t=j$, and from equality $\dot{z}_i(t,a_0)=\delta_k(t)$ it follows that the parameter Z_i receives a value at time $t=k$.

Generally speaking, a vector-function $\dot{f}(t,a_0)$ defines a network of parallel computations, namely: if several vector $\dot{f}(t,a_0)$ components are determined by the same function $\delta_j(t)$, then the operations corresponding to these components can be activated and executed in parallel.

2. In the general case, the preliminary plan is redundant in the sense that it may contain operations, the execution of which in the end "gives nothing" to achieve the objective B_0. Their elimination is possible due to the formation of Boolean vector functions of the form:

$$\begin{aligned} y_z(t)&=\dot{z}(t,a_0)\cdot z^*,\\ y_f(t)&=\dot{f}(t,a_0)\cdot f^*, \end{aligned} \qquad (7)$$

which determine the so-called complete problem solving plan.

3. If the solution to problem *T* is not unique, then the complete plan is also redundant in the sense that it is a "union" of possible plans for solving problem *T*.

Ultimately, we are interested in non-alternative non-redundant plans, the plans for which the exclusion of any operation (included in the plan) from them leads to the loss of solvability of the original problem by this plan. The presence of the complete plan allows us to construct a set of non-alternative non-redundant plans according to the following algebraic method:

a) For each column of the matrix *B* with the number *j*, such that $z_j^*=1$, we write out an elementary disjunction from the variables f_i, for which $b_{ij}=1$;

b) Write down the conjunction of the obtained elementary disjunctions;

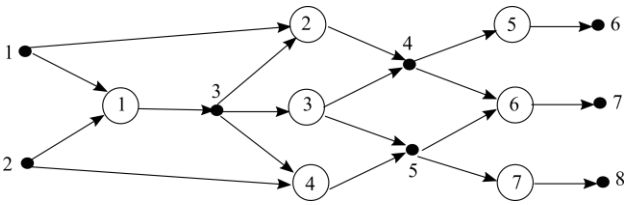

Figure 1. Main computational Boolean network. The dots denote the variables of the vector *z (m*=8*)*, the circles denote the variables of the vector *f (n*=7*).*

c) The resulting expression, taking into account the laws of idempotency and absorption of the algebra of logic, is transformed into a disjunctive normal form. The terms of this form determine the set of all alternative non-redundant plans for solving the problem $T=(A_0,B_0)$.

VI. ILLUSTRATIVE EXAMPLE

Let us consider the technology of synthesis of plans for solving the "given-to-find" problem on the example of a computational Boolean network shown in Fig. 1.

Matrices *A* and *B* corresponding to this network have the form:

$$A=\begin{bmatrix}1&1&0&0&0&0&0&0\\1&0&1&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&1&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&1&1&0&0&0\\0&0&0&0&1&0&0&0\end{bmatrix};$$

$$B=\begin{bmatrix}0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&1&1&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}.$$

In the scalar representation, matrix differential equations (5) and (6) have the form (8) and (9), respectively:

$$\begin{aligned}
\dot{f}_1&=\bar{f}_1 z_1 z_2 & \dot{z}_1&=0\\
\dot{f}_2&=\bar{f}_2 z_1 z_3 & \dot{z}_2&=0\\
\dot{f}_3&=\bar{f}_3 z_3 & \dot{z}_3&=\bar{z}_3 f_1\\
\dot{f}_4&=\bar{f}_4 z_2 z_3 & \dot{z}_4&=\bar{z}_4(f_2\vee f_3)\\
\dot{f}_5&=\bar{f}_5 z_4 & \dot{z}_5&=\bar{z}_5(f_3\vee f_4)\\
\dot{f}_6&=\bar{f}_6 z_4 z_5 & \dot{z}_6&=\bar{z}_6 f_5\\
\dot{f}_7&=\bar{f}_7 z_5 & \dot{z}_7&=\bar{z}_7 f_6\\
& & \dot{z}_8&=\bar{z}_8 f_7
\end{aligned} \qquad (8)$$

$$\begin{aligned}
\dot{f}_1 &= \bar{f}_1 z_3 & \dot{z}_1 &= \bar{z}_1(f_1 \vee f_2)\\
\dot{f}_2 &= \bar{f}_2 z_4 & \dot{z}_2 &= \bar{z}_2(f_1 \vee f_4)\\
\dot{f}_3 &= \bar{f}_3(z_4 \vee z_5) & \dot{z}_3 &= \bar{z}_3(f_2 \vee f_3 \vee f_4)\\
\dot{f}_4 &= \bar{f}_4 z_5 & \dot{z}_4 &= \bar{z}_4(f_5 \vee f_6)\\
\dot{f}_5 &= \bar{f}_5 z_6 & \dot{z}_5 &= \bar{z}_5(f_6 \vee f_7)\\
\dot{f}_6 &= \bar{f}_6 z_7 & \dot{z}_6 &= 0\\
\dot{f}_7 &= \bar{f}_7 z_8 & \dot{z}_7 &= 0\\
 & & \dot{z}_8 &= 0
\end{aligned} \quad . \qquad (9)$$

Let us assume that the problem statement T in the Boolean representation is given as follows: $a_0 = (11000000)$, $b_0 = (00000010)$.

After seven steps ($t = 7$) particular solutions $z(t, a_0)$, $f(t, a_0)$ of system (8) and particular solutions $z(t, b_0)$, $f(t, b_0)$ of system (9) reach equilibrium states

$$\begin{aligned}
z_b(a_0) &= col(11111111),\\
f_b(a_0) &= col(1111111),\\
z_a(b_0) &= col(11111010),\\
f_a(b_0) &= col(1111010).
\end{aligned}$$

So, we have

$$\begin{aligned}
z^* &= col(11111010),\\
f^* &= col(1111010).
\end{aligned}$$

Since $z_b(a_0) \geq b_0$, then the given problem is solvable on a Boolean computational network and the preliminary plan represents the derivative $\dot{f}(t, a_0)$ of a particular solution $f(t, a_0)$ of system (7), presented in Table 1.

In terms of elementary functions $\delta_j(t)$ will be:

$$\begin{aligned}
&\dot{f}_1(t, a_0) = \delta_0(t),\ \dot{f}_2(t, a_0) = \dot{f}_3(t, a_0) = \dot{f}_4(t, a_0) = \delta_2(t),\\
&\dot{f}_5(t, a_0) = \dot{f}_6(t, a_0) = \dot{f}_7(t, a_0) = \delta_4(t)
\end{aligned} .$$

A complete problem solving plan is formed according to equations (7) and is presented in Table 2.

To verify the resulting plan for uniqueness, we write out the logical product of elementary disjunctions

$$f_1(f_2 \vee f_3)(f_3 \vee f_4) f_6 = f_1 f_3 f_6 \vee f_1 f_2 f_4 f_6,$$

which means that the task has two solution plans, and in the second plan, operations F_2 and F_4 can be executed in parallel.

VII. Automation Tools for Computation Control Based on the Dynamic Planning Model

The HPCSOMAS-MSC platform provides tools for creating AMP agents and microservices, tools for automating this process, and tools for multi-agent control

TABLE I. Preliminary Plan

t	$\dot{f}$						
	1	**2**	**3**	**4**	**5**	**6**	**7**
0	1	0	0	0	0	0	0
2	0	1	1	1	0	0	0
4	0	0	0	0	1	1	1

of computations in a hybrid microservice computational infrastructure (HMCI). The basic version of HPCSOMAS-MSC [11] supports the following control schemes:

- Decentralized control of a group of distributed computational agents (DCA), self-organizing according to a non-procedural problem statement (NPS) on a distributed computational model KB (knowledge base) of the subject area;
- Hierarchical control of the composition of microservices formed according to the procedural problem statement (PPS), performed by the computational agents launching these microservices.

In the modified version of HPCSOMAS-MSC (Fig. 2), computation-planning tools are integrated into a separate subsystem that provides an additional possibility of centralized planning and control for the case of a centralized KB formed when creating an AMP based on a library of applied modules. To obtain a plan for solving the problem, a wizard for the abstract program creation uses the dynamic Boolean model proposed in section V. The wizard has WEB and API interfaces and works both alone and within AMP. In the latter case, the centralized planning and control agent (CPC) using this wizard receives a plan, according to which it deploys the computational microservice agents (CMA) required to solve the problem in the HMCI and launches them in the order specified by the plan as soon as the data is ready. When finished, the CMA returns and passes the output to the CPC. At this point, the CMA life cycle ends, and the occupied by this agent resource is released. With a decentralized approach, due to the presence of a decentralized KB, the life cycle of the DCA ends (the resource occupied by the agent is released) after receiving a message about the completion of the task solution from the last completed DCA. Thus, the CPC agent saves resources by distributing them dynamically, unlike the agent UDA (User Dew Agent, [12]), which manages decentralized computing (Fig. 2). So, for the example given in Section VI, the CPC will allocate three

TABLE II. Complete Problem Solving Plan

t	y_f						
	1	**2**	**3**	**4**	**5**	**6**	**7**
0	1	0	0	0	0	0	0
2	0	1	1	1	0	0	0
4	0	0	0	0	0	1	0

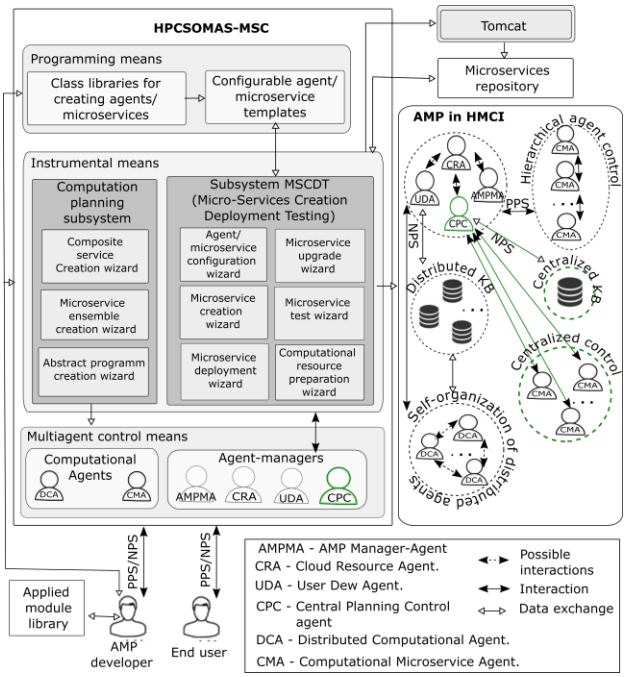

Figure 2. HPCSOMAS-MSC tools for working with AMP

computing resources for the first plan or four resources for the second one. The UDA will allocate seven resources. If necessary, the CPC agent is duplicated and receives the CMA launch monitoring protocol from the main CPC agent to increase the reliability of computations.

VIII. Conclusion

This article proposes new logical models of modular software complexes in the form of systems of nonlinear Boolean differential equations. Such systems allow both the creation of efficient methods for constructing the necessary particular solutions for high-dimensional models and the use of analytical and qualitative methods for studying the required dynamic properties of their solutions.

From a practical point of view, it is important to note that the logical models of modular software complexes in the form of systems of Boolean differential equations allow effective both software and hardware implementation. In this case, it becomes possible to use them to build high-speed programmable logic control devices. Another important area of application of the presented models is the construction of real-time intelligent control systems for moving objects, which provide dynamic reconfiguration of the structure and synthesis of control programs during the object functioning.

Acknowledgment

The study was supported by the Ministry of Science and Higher Education of the Russian Federation, project no. 121032400051-9. The authors would like to thank Irkutsk Supercomputer Center of SB RAS for providing access to HPC-cluster “Akademik V.M. Matrosov” [13].

References

[1] M. Ghallab, D. S. Nau, and P. Traverso “Automated planning: theory and practice,” Morgan Kaufmann, May 2004.

[2] A. Solar-Lezama, “Program synthesis by sketching,” Diss. UNIVERSITY OF CALIFORNIA, BERKELEY, 2008.

[3] G. A. Oparin, A. P. Novopashin, “Boolean models and planning methods for parallel abstract programs,” Autom. Remote Control, vol. 69, no. 8, 2008, pp. 1423–1432.

[4] E.H. Tyugu, “The structural synthesis of programs,” in Algorithms in Modern Mathematics and Computer Science, LNCS, vol. 122, A.P. Ershov and D.E. Knuth, Eds., Berlin, Heidelberg: Springer, 1981, pp. 290–303.

[5] V. Kotkas, A. Ojamaa, P. Grigorenko, R. Maigre, M. Harf, and E. Tyugu, “CoCoViLa as a multifunctional simulation platform,” in International ICST Conference on Simulation Tools and Techniques, Brussels: ICST, 2011, pp. 1–8.

[6] G.A. Oparin, V.G. Bogdanova, A.A. Pashinin, and S.A. Gorsky, “Microservice-oriented approach to automation of distributed scientific computations,” in 42nd Intern. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 2019, pp. 236-241.

[7] K.V. Knyazkov, S.V. Kovalchuk, T.N. Tchurov, S.V. Maryin, and A.V. Boukhanovsky, “CLAVIRE: e-science infrastructure for data-driven computing,” Journal of Computational Science, 2012, vol. 3, no. 6, pp. 504–510.

[8] G. Mints, E. Tyugu, “Justification of the structural synthesis of programs,” Science of Computer Programming, 1982, vol. 2, no. 3, pp. 215–240.

[9] D. Bohmann, R. S. Stankovič, Zh. Toshich, V. P. Shmerko, and S. N. Yanushkevich, “Logic differential calculus: achievements, trends, and applications,” Autom. Remote Control, 2000, vol. 61, no. 6, part 2, pp. 1033–1047, https://zbmath.org/pdf/1060.94051.pdf [accessed: March 16 2023].

[10] D. Bochmann, C. Posthoff, “Binäre dynamische systeme,” Berlin: Akademie-Verlag, 1981.

[11] G.A. Oparin, V.G. Bogdanova, and A.A. Pashinin, “Automated tools for the development of microservice compositions for hybrid scientific computations,” in 2nd Information, Computation, and Control Systems for Distributed Environments, 2020, pp. 201–213.

[12] A.A. Pashinin and V.G. Bogdanova, “Application of user dew agent in hybrid-computing environments,” in 1st International Workshop on Advanced Information and Computation Technologies and Systems (AICTS), 2020, pp. 135–145.

[13] “Irkutsk Supercomputer Centre of SB RAS,” http://hpc.icc.ru [online, accessed: January 31 2023].