Nexo Revista Científica
Vol. 34, No. 01, pp. 01-12/Marzo 2021

ISSN-E 1995-9516
Universidad Nacional de Ingeniería

http://revistas.uni.edu.ni/index.php/Nexo
https://doi.org/10.5377/nexo.v34i01.11278

Parallel decomposition of control algorithms for computational processes based on the use of nondeterministic automaton logic

Descomposición paralela de algoritmos de control para procesos computacionales basados en el uso de lógica autómata no determinista

Dmitry V. Pashchenko[1*], Alexey I. Martyshkin[2], Dmitry A. Trokoz[1], Tatyana Yu. Pashchenko[3], Mikhail Yu. Babich[4], Mikhail M. Butaev[4]

[1] Penza State Technological University, 440039, Russia, Penza, 1/11 Baydukova lane/Gagarina st., 1/11
[2] Department of Computational Automatons and Systems, Penza State Technological University, 440039, Russia, Penza, 1/11 Baydukova lane/Gagarina st., 1/11
[3] Penza State University, 440026, Russia, Penza, Krasnaya Street, 40
[4] JSC Research and Production Enterprise "Rubin", 440000, Russia, Penza, Baydukova st, 2
[*]dmitry.pashcenko@gmail.com

(recibido/received: 25-October-2020; aceptado/accepted: 01-December-2020)

ABSTRACT

The paper deals with the issues of decomposition of control algorithms for the processes in parallel computing systems and the use of automaton models. When designing parallel processing systems, an important task is the formal presentation of process control algorithms since they allow achieving a packaged solution to the problems of specification, development, implementation, verification, and analysis of complex control systems, including the control of interacting processes and resources in parallel computing systems. It is especially necessary to use formal methods to verify complex information processing systems by model testing. One of the methods for the formal description of control algorithms is based on the use for these purposes of the nondeterministic automaton (NDA) logic, which is a method that allows one to present control algorithms for information processing in the form of systems of canonical equations describing all particular events implemented in the algorithm. The advantage of such a language is that all transitions in the control system are described not in terms of system states, but in terms of particular events, the simultaneous existence of which determines all states and transitions in the system; this allows avoiding a "combinatorial explosion" in the state space to the possibilities of means verification. Purpose of the paper: research of control algorithms for parallel computing systems using the NDA apparatus. The development and research object is parallel decomposition of control algorithms for parallel computing systems using automatic models.

Keywords: control algorithm, verification, finite automaton, simulation, parallel system, formalization.

RESUMEN

El trabajo aborda los problemas de descomposición de algoritmos de control para los procesos en sistemas de computación paralelos y el uso de modelos de autómatas. A la hora de diseñar sistemas de procesamiento paralelo, una tarea importante es la presentación formal de los algoritmos de control de procesos, ya que permiten lograr una solución empaquetada a los problemas de especificación, desarrollo, implementación, verificación y análisis de sistemas de control complejos, incluyendo el control de los procesos que interactúan y recursos en sistemas informáticos paralelos. Es especialmente necesario utilizar métodos formales para verificar sistemas de procesamiento de información complejos mediante pruebas de modelos. Uno de los métodos para la descripción formal de los algoritmos de control se basa en el uso para estos fines de la lógica del autómata no determinista (NDA), que es un método que permite presentar algoritmos de control para el procesamiento de la información en forma de sistemas de ecuaciones canónicas. describiendo todos los eventos particulares implementados en el algoritmo. La ventaja de tal lenguaje es que todas las transiciones en el sistema de control se describen no en términos de estados del sistema, sino en términos de eventos particulares, cuya existencia simultánea determina todos los estados y transiciones en el sistema; esto permite evitar una "explosión combinatoria" en el espacio de estados a las posibilidades de verificación de medios. Objeto del trabajo: investigación de algoritmos de control para los procesos de sistemas informáticos paralelos utilizando el aparato NDA. El objeto de investigación y desarrollo es el proceso de descomposición en paralelo de algoritmos de control para sistemas de cómputo en paralelo utilizando modelos automáticos.

Palabras clave: algoritmo de control, verificación, autómata finito, simulación, sistema paralelo, formalización.

1. INTRODUCTION

To increase control systems' performance for various processes and objects, including information processing control, various methods of parallelization of control algorithms at different stages of information processing are used. To effectively solve the problems of parallelizing algorithms, they use various formal methods for representing control algorithms. One of the promising formal languages for the specification of control algorithms is the nondeterministic automaton (NDA) language presented in the form of a canonical equation system (CES) (Hoare, 1989) describing all particular events implemented in the control system. Depending on the algorithm's detail for controlling the computational process, the events described by the CES could be understood as the execution of certain acts of information processing, which may include: the execution of micro-operations and macro-operations functional operators and subroutines, etc (Baumgertner, 2012).

2. MATERIALS AND METHODS

In the works of well-known domestic and foreign scientists (Gorbatov et al.,1991; Vashkevich & Filatov, 1994; Vashkevich & Vashkevich, 1993; Hopcroft & Ullman, 1979; Hopcroft et al., 2002), (Martin, 2010; Allan et al., 2005; Saglam, Bilge & Mooney III, 2001; Baumgertner & Melnikov, 2012; Baumgertner, 2012) and (Salomaa, 1986; Melnikov, 2010; Melnikov & Vakhitova, 1998; Melnikov, 2010), great advantages of the automaton representation for logic control algorithms are noted, which include great convenience, compactness and simplicity of the formal representation for the CAs (control algorithms) in comparison with their representation based on the Petri net language. However, the authors attribute this advantage of the automaton CA representation only for representing sequence systems, taking into account the use of models of only deterministic automata (DA). For parallel action systems, the authors give preference to Petri nets, not taking into account the main advantage of the nondeterministic automaton (NDA) models, which is defined in the parallelism intrinsic for them (Vashkevich & Biktashev, 2016; Pashchenko et al., 2016; Pashchenko et al., 2015).

The formal language of CA representation based on the NDA logic can also be considered as a version of the basic language, which includes two components: a graphical form for the visual representation of algorithms in the form of graph-schemes with parallel branches (GSPB) and an analytical one in the form of a system of recurrent canonical equations (CES) describing all particular events implemented in the algorithm (Vashkevich & Biktashev, 2016; Vashkevich, 1973).

To increase control systems' performance for various processes and objects, including information processing control, various methods of parallelization of control algorithms at different stages of information processing are used. They use various formal methods for representing the CA for effectively solving the problems of parallelizing algorithms and quantitative assessing their complexity. One of the most promising formal languages for the specification of control algorithms is the NDA language presented in the form of a canonical equations system (CES) (Vashkevich, 1994).

When parallelizing the CA, we will take into account the dependence of events both in terms of information and control and in terms of data being shared variables or shared resources. The CA parallelization result should be the splitting of all particular events included in the original CES into groups of incompatible events when in each group, the events can only be executed sequentially (Pashchenko et al., 2020).

Parallelization of an original CA is based on the use of the results of the CES determination, which is the original algorithm. The CES determining process is performed, in turn, based on the use of a direct transition table (DTT) of the NDA for all particular events implemented in the CA (Pashchenko et al., 2020; Martyshkin et al., 2020; Nikolaevich et al., 2020; Pashchenko et al., 2019; Pashchenko et al., 2019). Before starting the NDA DTT construction, all possible pairs of particular events are found in the original CA, which will correspond to the following four values:

$$\left[S_i, S_{j,з}, S_j, S_{j,с}\right]$$

Where S_j is the event being formalized,

S_i - an event immediately preceding the event S_j;

$S_{j,з}$ - an event symbolizing the logical conditions (LC) for the origin of the event S_j (the condition of its initial appearance);

$S_{j,с}$ - an event symbolizing the LC of saving the event S_j after its initial appearance; this event determines the duration of the event S_j.

In the simplest case, events $S_{j,з}$ and $S_{j,с}$ can be represented as particular input signals $x_{i,j}$ that determine the transition from the event S_i to the event S_j. In the case if that $S_{j,с} \neq 0$, then the signal $x_{i,j}$ will take on the designation $x_{j,j}$, which means the transition from the event S_j to the event S_j again.

Among all the pairs of events presented in the CA, the events are first found that do not have information and control connections between themselves but have a common immediately preceding event, from which, in fact, parallelization of the algorithm begins.

The NDA direct transfer table (DTT) will have the following structure (Table 1)

Table 1. The structure of an NDA direct transition table representing the original CA

Algorithm step	The initial particular event at time **t** $S_i(t)$	Logic transition conditions (Particular input signal) $x_{i,j}(t)$	Transition event $S_j(t+1)$
1	2	3	4

The DTT should be constructed in such a way that unattainable events of the original algorithm can be identified. For this purpose, the construction of the DTT begins with marking the initial event in column 2, and for all subsequent steps of the algorithm, column 2 is marked only by the event that took place in the previous steps of the algorithm and was marked in column 4. Taking into account the introduced designations of events and the CES for the CA, the corresponding NDA DTT can be represented in the general case by the following system of quantifier-free recursive canonical equations for all events realized in the CA and by the corresponding Moore's NDA model:

$$S_j^{Y_j}(t+1) = \bigvee_{i,j} S_i(t)\, S_{j,3}(t) \vee \bigvee_{j} S_j(t)\, S_{j,3}(t), \tag{1}$$

Where $j = \overline{0,n}$, $S_0(0) = 1$; Y_j is a combination of particular output signals to mark the event S_j.

The determination algorithm is considered in detail in (Vashkevich & Biktashev, 2016; Pashchenko et al., 2020; Vashkevich, 2002) and is based on the search for events, the simultaneous existence of which in the original algorithm is possible. This means that there are values of logical variables for which several events are executed simultaneously with one implementation of the algorithm, i.e., such events are compatible. The determination algorithm execution result is a deterministic DTT, which is constructed in steps, starting with a combination of initial events, and has the following form (Table 2). For all subsequent steps in the construction of the transition table, filling in column 2 is carried out according to the same principle as for the DTT NDA. Based on the determination results, we get all possible combinations of particular events (Table 2, column 4) that can be performed simultaneously, i.e., they are compatible events.

Table 2. Direct transition table of a deterministic automaton representing the original CA

Algorithm step	Population of initial particular events	Combination of particular input signals at the transition	A set of particular events at the transition
	Complete event $a_m(t)$	Complete input signal $x(a_m, a_s)(t)$	Complete event $a_s(t+1)$
1	2	3	4

Using the results of determinisation, it is possible to construct an auxiliary compatibility matrix (Figure 1) of all particular events of the original CA.

	S_1	…	S_β	…	S_n
S_1	0	…	$\tau_{1,\beta}$	…	$\tau_{1,n}$
…	…	…	…	…	…
S_α	$\tau_{\alpha,1}$	…	$\tau_{\alpha,\beta}$	…	$\tau_{\alpha,n}$
…	…	…	…	…	…
S_n	$\tau_{n,1}$	…	$\tau_{n,\beta}$	…	0

Figure 1. Matrix of compatibility of particular events, taking into account their relationship in management and information

The cells of the presented compatibility matrix are filled on the basis of the following: $\tau_{\alpha,\beta}$ = 0, if S_α and S_β are not compatible, and $\tau_{\alpha,\beta}$ = 1, if S_α and S_β are compatible, i.e., jointly enter a complete event. It should be noted that when constructing the compatibility matrix based on the results of the determination algorithm, only the connections of events by control and information were taken into account, without taking into account the relationship of events by data. The connection by data will be taken into account when adjusting the event splitting into groups of incompatible events.

An event connection by information indicates that the output information obtained from the execution of the event S_i is input information for the execution of the event S_j, since the event S_i is the event immediately preceding the event S_j. Thus, such a relationship indicates that such events cannot under any circumstances be executed simultaneously.

The connection of events by control takes place in the case that based on the results of the execution of the event S_i, a LC $\left(X_{i,j}\right)$ are generated, which determines the transition to one of the events. Thus, the connection by control, which determines the sequence of events, is at the same time a connection by information.

The algorithm for constructing the inclusion matrix is based on the requirements for the satisfaction of the following conditions:

a) $m_i \cap m_j = \varnothing;\ i \neq j;\ i, j = \overline{1,H}$, $H \geq P$,

b) $m_1 \cup m_2 \cup \ldots \cup m_H = n$,

Where m_i and m_j are groups of incompatible events,

H is the number of groups of incompatible events,

n is the number of all particular events,

P is the maximum number of particular events included incomplete events.

Condition (a) means that each event can be included in only one single group of incompatible events. Condition (b) means that each event must be included in one of the groups of incompatible events. The last option will correspond to the serial port of the CA, which the main processor can carry out, and the first option will correspond to such a way of implementing the CA when the functions of the main processor will be assigned to the working processors, each of which implements one of the parallel branches of the CA. The main processor performs tasks that would be associated with the synchronization of working processors, including when there is a connection of particular events by the data due to contention. Usually, particular events that are incompatible with any of the original algorithm's events include events of two consecutive branches of a common CA.

The first initial sequential part of the CA serves to perform preparatory functions. In turn, the second part of the sequential composition of the CA is needed to perform functions related to the processing of information obtained as a result of executing parallel branches of the algorithm.

2.1. Taking into account the connection of events by data when parallelizing CA

Let us now consider the issues on correcting subsets' composition containing incompatible particular events, taking into account their dependence on the data. The need for such a correction arises in the case, when in the course of constructing a matrix for the inclusion of particular events determining the division of the entire set of events of the original CA into subsets of incompatible events, a situation may arise when

individual pairs of data-dependent events will be located in different subsets. This circumstance can cause critical deadlocks in implementing the information processing algorithm since such events require a shared resource.

There are two possible solutions to the problem of correcting the composition of subsets of incompatible particular events, which will not lead to deadlocks. For the first option, the solution to such a problem is considered without considering the correction of the original CA, and the second is based on the correction of the original CA by its equivalent transformations. We can use an auxiliary matrix of the dependence of particular events on the data; this matrix has the form (Figure 2).

	S_1		S_β		S_n
S_1	0	…	$D_{1,\beta}$	…	$D_{1,n}$
…					
S_α	$D_{\alpha,1}$	…	$D_{\alpha,\beta}$	…	$D_{\alpha,n}$
…					
S_n	$D_{n,1}$	…	$D_{n,\beta}$	…	0

Figure 2. Matrix of dependence of particular events on data

The matrix cells describing the dependence of particular events on data are filled based on the following: if a common resource is required to execute events S_α and S_β, i.e. they have data dependencies, so to avoid data contention, they must only be executed sequentially. $D_{\alpha,\beta} = 0$ if there is no data connection for events S_α and S_β, i.e. they can run at the same time.

Identification of particular events in different groups m_i that have a connection by data is carried out by sequentially performing operations of intersecting the row S_α of the data compatibility matrix (Figure 2) with the inclusion matrix row, starting with i = 1.

If $m_i \cap S_\alpha = S_\beta$, this means that the subset m_i contains some event S_β that has a data connection with the event S_α. In this case, the event S_β can belong either to a subset m_i or to some other subset m_j. In this case, the following options are possible:

a) If $m_i \cap S_\alpha = S_\beta$, and $S_\beta \cap m_i = S_\alpha$, then S_α and S_β also belong to the same subset m_i and there will be no contention between them by the data, accordingly.

b) If $m_i \cap S_\alpha = S_\beta$, then $S_\beta \cap m_i \neq S_\alpha$ and $S_\beta \cap m_j = S_\alpha$, then S_α and S_β both belong to different subsets m_i and m_j, accordingly; they should be placed in one subset of incompatible particular events to exclude data competition, if possible.

In the event that a connection by data was found for the events presented in different groups, then the composition of the particular events of these groups should be adjusted. Let us illustrate the technique of such a correction with an example.

Example. Let the following variants of the inclusion matrix (Figure 3) are obtained based on the results of CA determination, for which the event S_2 can be placed in the subset m_1 or m_2, and the event S_5 can

be placed in the subset $m2$ or $m3$. Thus, Figure 3 shows four possible options for placing particular events in 3 subsets $m1$, $m2$ and $m3$.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
m_1	1	1	1	0	0	0	0
m_2	0	0	0	1	1	0	0
m_3	0	0	0	0	0	1	1

а)

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
m_1	1	0	1	0	0	0	0
m_2	0	1	0	1	0	0	0
m_3	0	0	0	0	1	1	1

б)

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
m_1	1	1	1	0	0	0	0
m_2	0	0	0	1	0	0	0
m_3	0	0	0	0	1	1	1

в)

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
m_1	1	0	1	0	0	0	0
m_2	0	1	0	1	1	0	0
m_3	0	0	0	0	0	1	1

г)

Figure 3. Options for the inclusion matrix

Let the matrix of event dependence on data for all particular events have the form shown in Figure 4.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
S_1	0	0	0	0	0	0	0
S_2	0	0	1	0	1	0	0
S_3	0	1	0	0	0	0	0
S_4	0	0	0	0	0	0	0
S_5	0	1	0	0	0	0	0
S_6	0	0	0	0	0	0	0
S_7	0	0	0	0	0	0	0

Figure 4. The example of a matrix of particular event dependence on data

Using the possible options for the inclusion matrix, we sequentially execute the operations of the intersection between the row S_α of the compatibility matrix for particular events according to the data (Figure 4) and the rows of the inclusion matrix (Figure 3); as a result, we obtain the following options of such intersections, which are not equal to $\varnothing$:

Option (a)

$S_2 \cap m_1 = S_3$,
$S_3 \cap m_1 = S_2$,
$S_5 \cap m_1 = S_2$,
$S_2 \cap m_2 = S_5$,

Option (b)

$S_2 \cap m_1 = S_3$,
$S_3 \cap m_2 = S_2$,
$S_5 \cap m_2 = S_2$,
$S_2 \cap m_3 = S_5$,

Option (c)

$S_2 \cap m_1 = S_3$,
$S_3 \cap m_1 = S_2$,
$S_5 \cap m_1 = S_2$,
$S_2 \cap m_3 = S_5$,

Option (d)

$S_2 \cap m_1 = S_3$,
$S_3 \cap m_2 = S_2$,
$S_2 \cap m_2 = S_5$,
$S_5 \cap m_2 = S_2$.

Based on the obtained variants of intersections between the row S_α of the compatibility matrix of particular events by data and the rows of the inclusion matrix, the following conclusions can be drawn:
For option (a): data-dependent events S_2 and S_3 belong to the same subset m_1 without causing a critical deadlock; events S_2 and S_5, depending on data, belong to different subsets m_1 and m_2, i.e., they can cause a critical deadlock.

For option (b): events S_2 and S_3 depending on the data belong to different subsets m_1 and m_2, i.e., they can cause a critical deadlock; events S_2 and S_5 depending on data belong to different subsets m_2 and m_3, i.e., they can cause a critical deadlock.
For option (c): data-dependent events and belong to the same subset without causing a critical deadlock; events and, depending on data, belong to different subsets of and, i.e. can cause a critical deadlock.
For option (d): events S_2 and S_3 depending on data belong to different subsets m_1 and m_2, i.e., they can cause a critical deadlock; data-dependent events S_2 and S_5 belong to the same subset m_2 without causing a critical deadlock.
Based on the formulated conclusions, the following conclusion can be made:
1. Option (b) of placing particular events in subsets of the inclusion matrix is the worst since critical deadlock situations may arise twice due to the dependence on data of two pairs of events: (S_2, S_3) and (S_2, S_5).

2. Options (a), (c), and (d) are equivalent in the number of possible critical deadlocks due to the dependence on data for pairs of events: (S_2, S_5) - option (a) and (c), and (S_2, S_3) for option (d).

The further choice of the most optimal variant of placing particular events in subsets of the inclusion matrix should, in the general case, be determined by the processing time of events running in parallel branches. Namely, it is necessary to strive to ensure that the total processing time of events in all parallel branches does not differ much from each other. Considering this circumstance, the worst option of (a), (c), and (d) will be an option (c), because the events in the subsets are not evenly distributed for it: there are three events in the subsets m_1 and m_3, and only one event in the subset m_2. The final choice of the optimal option for

placing particular events in subsets of the inclusion matrix from the two remaining options (a) and (d), which have, as noted above, essentially the same placement of particular events in subsets, will be determined by the processing time of particular events. The best option is the one for which the event processing time in all subsets will have the smallest value.

Let us consider the second option for choosing the most optimal placement of particular events in subsets of the inclusion matrix that are connected by data. The mentioned method is based on correcting an initial CA by means of an equivalent transformation, which would allow obtaining such a partition of particular events into groups of events (incompatible), when, for example, some event S_α belonging to a subset m_i and having a connection by data with an event S_β would never be executed simultaneously with the event S_β that belongs to a different subset m_j. If there are such events, then a conflict situation due to the resource will not occur.

Such an adjustment of the initial CA is possible if there are events in individual branches of the CS that, although they are related according to data with events included in other parallel branches, within their branch, they are not connected by information and control (Vashkevich & Biktashev, 2015). We can use the results of the following intersection operations:

$$m_i \cap S_\alpha^i = A,\ (\forall i,\alpha)\left(S_\alpha^i \subset m_i\right), \alpha = \overline{1,k}\ , \qquad (2)$$

Where S_α^i is the content of the row from the matrix with event compatibility by information and control; the matrix is designated by the symbol S_α of an event belonging to the subset m_i; k is the number of events included in the subset m_i.

If the result of the intersection is not equal to $\varnothing$, then we get a subset A of the desired events. As an illustration of the technique, let us take the option of placing particular events in 3 subsets (Figure 3, a), for which events S_2 and S_5 depending on data, belong to different subsets m_1 and m_2 and cause a critical deadlock. Let us assume that, based on the results of the determinization of the corrected CA, the compatibility matrix of particular events was built, which has the form (Figure 5):

The results of the intersection operations for subsets m_1, m_2 and m_3 (Figure 3, a) with the rows of the compatibility matrix (Figure 5) have the form:

$$\begin{aligned} &m_1 \cap S_1 = \varnothing, \quad && m_2 \cap S_4 = \varnothing, \quad && m_3 \cap S_6 = \varnothing, \\ &m_1 \cap S_2 = S_5, \quad && m_2 \cap S_5 = S_2, \quad && m_3 \cap S_7 = \varnothing. \\ &m_1 \cap S_3 = \varnothing, \end{aligned}$$

Based on the intersection results, we found events S_2 and S_5, depending on the data, which belong to different subsets, but never run simultaneously. These events are not related by information and control (Figure 5), since according to the CA determination results, they were not included in any complete event.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
S_1	1	0	0	1	1	1	1
S_2	0	1	0	1	0	1	1
S_3	0	0	1	1	1	1	1
S_4	1	1	1	1	0	1	1
S_5	1	0	1	0	1	1	1
S_6	1	1	1	1	1	1	0
S_7	1	1	1	1	1	0	1

Figure 5. An example of the compatibility matrix for particular events, taking into account their relationship on control and information

3. CONCLUSION

In conclusion, we note that, as noted earlier, the final choice of the optimal variant of placing particular events along parallel branches will be determined by the time of implementation of the control algorithm in its parallel branches. In the case when the correction of the composition of particular events in subsets of the inclusion matrix does not allow avoiding critical deadlocks due to dependence on data, then it is necessary to use one of the methods for synchronizing parallel processes that implement the control algorithm and ensure the elimination of critical deadlocks.

.

In this case, the highest priority is assigned to the process that includes the largest number of events in the subset m_i in order to organize the entry of processes into their critical intervals. In this case, the processing times of events in subsets may be somewhat leveled.

ACKNOWLEDGMENTS

RFBR funded the reported study according to the research project № 19-07-00516.

REFERENCES

Allan, C., Avgustinov, P., Christensen, A. S., Hendren, L., Kuzins, S., Lhoták, O., de Moor, O., Sereni, D., Sittampalam, G., Tibble, J. (2005). Adding trace matching with free variables to AspectJ. In: Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications. San Diego, CA, USA: OOPSLA '05. ACM, New York, NY. 345-364.

Baumgertner, S. (2012). An Analog of the Kleene Theorem for Generalized Nondeterministic Finite Automaton. Vector of Science, Togliatti State University 4 (22), 23–25 (in Russian).

Baumgertner, S., Melnikov, B. (2012). Mathematical Model of Automata for Generalized Regular Expressions. Heuristic Algorithms and Distributed Computing in Applied Problems: coll. Monogr. Tolyatti: Publishing house of TSU, 16–23. (in Russian).

Clarke, E.M., Grumberg, O., Peled, D. (2002). Program Model Verification: Model Checking. Translation from English. M., MCCME. 416 p.

Gavrilov, S.A., Devyatov, V.V., Pupyrev, E.I. (1977). Logical Design of Discrete Automata. M.: Nauka. 351 p.

Gorbatov, V.A., Smirnov, M.I., Khalytchiev, I.S. (1991). Logical Control over Distributed Systems Ed. by V.A. Gorbatov. M., Energoatomizdat. 288 p.

Hoare, C. (1989). Interacting Sequential Processes. M., Mir., 264 p. (in Russian).

Hopcroft, J., Motwani, R., Ullman, J. (2002). Introduction to Automata Theory, Languages, and Computation. M.: Williams. 528 p. (in Russian).

Hopcroft, J.E., Ullman, J.D. (1979). Introduction to Automata Theory, Languages, and Computation. Reading/MA: Addison-Wesley. ISBN 0-201-02988-X.

Martin, J. (2010). Introduction to Languages and the Theory of Computation. McGraw Hill. — ISBN 978-0071289429.

Martyshkin, A.I., Pashchenko, D.V., Trokoz, D.A., Sinev, M.P., Svistunov, B.L. (2020). Using queuing theory to describe adaptive mathematical models of computing systems with resource virtualization and its verification using a virtual server with a configuration similar to the configuration of a given model. Bulletin of Electrical Engineering and InformatCES, 9(3), 1106-1120.

Melnikov, B. (2010). Extended Non-Deterministic Finite Automaton. Fundamenta Informaticae. 104 (3), 255–265.

Melnikov, B. (2010). Once More on the Edge-Minimization of Nondeterministic Finite Automaton and the Connected Problems. Fundamenta Informaticae 104 (3), 267–283.

Melnikov, B., Vakhitova, A. (1998). Some More on the Finite Automaton. J. of Applied Math. and Comp. (The Korean J. of Comp. Appl. Math.) 5 (3), 495–506.

Nikolaevich, K.N., Aleksandrovich, Z.S., Nikolaevna, P.U., Petrovich, S.M., Vladimirovich, B.V., Anatolievich, T.D., Ivanovich, M.A., Aleksandorovich, K.V. (2020). Instrumental System of Temporal Analysis of Models of Concurrent Computing Systems Constructed Using Theory of Temporary Finite State Automata. Moscow Workshop on Electronic and Networking Technologies, MWENT 2020 - Proceedings, 9067448.

Pashchenko, D., Sinev, M., Trokoz, D., Martyshkin, A., Veselova, M., Zabrodina, K., Safronova, V., Puchkova, U. (2019). Formalized Description of Message Encryption in Messaging Apps Using Automata Theory. Proceedings - 2019 21st International Conference Complex Systems: Control and Modeling Problems, CSCMP 2019, 2019-September, c. 565-569, 8976517.

Pashchenko, D., Trokoz, D., Konnov, N., Sinev, M. (2015). Formal transformation inhibitory safe Petri nets into equivalent not inhibitory. Procedia Computer Science, 49(1), c. 99-103. DOI: 10.1016/j.procs.2015.04.232

Pashchenko, D., Trokoz, D., Martyshkin, A., Sinev, M. (2019). Using Algebra of Hyper-Dimensional Vectors for Heuristic Representation of Data while Training Wide Neural Networks. Proceedings - 2019 21st International Conference "Complex Systems: Control and Modeling Problems", CSCMP 2019, 2019, 2019-September, 168-171, 8976507.

Pashchenko, D., Trokoz, D., Sovetkina, G., Nikolaeva, E., Sinev, M., Dubravin, A., Konnov, N. (2016). The methodology of multicriterial assessment of Petri nets' apparatus. MATEC Web of Conferences, , 44, 01009

Pashchenko, D.V., Martyshkin, A.I., Trokoz, D.A. (2020). Decomposition of Process Control Algorithms for Parallel Computing Systems Using Automata Models. Proceedings - 2020 International Russian Automation Conference, RusAutoCon 2020, 839-845, 9208165.

Pashchenko, D.V., Trokoz, D.A., Martyshkin, A.I., Sinev, M.P., Svistunov, B.L. (2020). Search for a substring of characters using the theory of nondeterministic finite automata and vector-character architecture. Bulletin of Electrical Engineering and InformatCES, 9(3), 1238-1250.

Saglam, Bilge E., Mooney III, V.J. (2001). System-on-a-Chip Processor Synchronization Support in Hardware. In: Proceedings of the conference on Design, automation, and test in Europe, IEEE Press Piscataway. NJ, USA, 633-641.

Salomaa, A. (1986). Pearls of the Theory of Formal Languages. M.: Mir. 159 p. (in Russian).

Vashkevich, N. P., Biktashev, R.A. (2015). Automaton-Based Representation of the Parallel Process Control Algorithms in the 'Reader-Writer' Problem Based on the Concept of Non-Determinism and the Monitor Mechanism. News of the Higher Educational Institutions. Volga Region. Engineering Science 2, 65–75. (in Russian).

Vashkevich, N.P. (1973). On One Way to Synthesize Digital Automaton According to the Flowgraph of an Algorithm with Parallel Branches. Computer Engineering: Interuniversity Coll. Sc. Works 1 (2), 49-57 (in Russian).

Vashkevich, N.P. (1994). Synthesis of Automaton Control Models for Parallel Interacting Processes. New Information Technologies and Systems: Proc. of Intern. Sc. and Tech. Conf. - Penza, 49–54 (1994). (in Russian).

Vashkevich, N.P. (2002). Automaton Approach to Formalizing Algorithms of Control over the Interaction of Parallel Processes when Applying to Variables (Resource) Shared by n-Processes. Computational systems and technologies: Interuniversity Coll. Sc. Works - Penza: Publishing House of the Penza State University 1 (27), 3–10. (in Russian).

Vashkevich, N.P., Biktashev, R.A. (2016). Nondeterministic Automata and Their Use for the Implementation of Parallel Data Processing Systems: Monograph. Penza: Publishing house of PSU, 394 p.

Vashkevich, N.P., Vashkevich, S.N. (1993). Formalization of Control Algorithms with Interacting Parallel Branches. Computer Engineering: Interuniversity Coll. Sc. Works - Penza: Publishing House of the Penza State Technical University 18, 17–22. (in Russian).

Vashkevich, S. N., Filatov, A.Yu. (1994). Structural Synthesis of Controllers Specified by NDA. Information Technologies and Systems: Proc. of Intern. Sc. and Tech. Conf. - Penza. p. 123 (in Russian).