Analysis of the Correctness of the Model of Biomedical Experiment Based on the Transition Graph by Means of a Matrix Algebra

Kruzhkov Alexandr
Faculty of Informatics and Robotics
Ufa State Aviation Technival University
Ufa, Russia
danteform@gmail.com

Nasyrov Rashit
Faculty of Informatics and Robotics
Ufa State Aviation Technival University
Ufa, Russia
nrash@yandex.ru

Mulayanov Ruslan
Faculty of Informatics and Robotics
Ufa State Aviation Technival University
Ufa, Russiae-mail:
mullruslan@yandex.ru

***Abstract*—The article deals with the problems of formal description of methods of organization of biomedical experiment. The need for a formal description of the experimental scheme is postulated. One of the ways of formal representation of the biomedical experiment based on the transition graph, which is one of the variants of the description of the finite state machine, is considered. The description revealed that the main characteristics of this representation are the correctness and adequacy of the representation in the form of a graph of transitions. The problem of description of nested sequences of actions corresponding to the beginning and end of the use of different components of the experiment is considered. An example of such use is the serial and parallel activation of experimental equipment. In this case, the correct sequences correspond to the implemented experiments, and the wrong ones correspond to the unrealizable ones. In this regard, there is a need to verify such feasibility. The necessity of checking the correctness of the graph as the main condition for the feasibility of the experiment is substantiated. One of the approaches to the analysis of the correctness of such a graph on the basis of matrix calculations is presented. The method of constructing the transition graph matrix is considered. The basic transformations necessary for checking the correctness of the transition graph of the biomedical experiment are given.**

Keywords—analysis, compilation, transition graph, matrix algebra

I. Introduction

Topics devoted to the organization and conduct of an experiment are usually the subject of philosophical sciences in such sections as gnoseology, epistemology, and more broadly, the knowability of the world [1]. Here, researchers obtained fairly extensive results of a general methodological nature. Without detracting from the achievements of representatives of the philosophical direction of the study of the features of the experiment as a way of knowing, the authors are compelled to state a clear inadequacy in the elaboration of particular methodological and technological issues related to the organization and conduct of experimental research. In this connection, an actual problem arose - the consideration of a formal model of an experiment for describing a logically correct sequence of organization and carrying out experimental actions in manual, automatic and automated modes.

II. THE MATRIX ALGORITHM

When automating a scientific experiment, describing the course of an experiment using programming languages has become an integral part of building an experiment automation system. But the programming language in its “pure” form only makes it difficult for the engineer involved in designing such a system (see the disadvantages below). To speed up and reduce labor costs for such works, other approaches to describing the control program were invented. The most famous of these approaches are: the Graphset diagram [3], the National Instruments graphical G language used in the LabView product [4], etc. Such programming languages differ from the usual ones in that the control algorithm is expressed in them in graphical form.

If we consider these languages from the standpoint of automating a scientific experiment, we can see in them our shortcomings, but for this it is necessary to clarify the course of the experiment from the point of view of the operation of the system.

It is well known that the experiment is the observation of the subject for the object under controlled conditions. Thus, for any scientific experiment, it is possible to compare the algorithm of measurement (observations) and the creation of control actions (environmental control). This algorithm is called the experiment. As a rule, electronic devices that transform logical commands in the form of control signals into physical action and vice versa are involved in measurement and control during experiment automation.

From the above description of the course of an experiment, it is possible to derive a feature of the automation of a scientific experiment - the eventfulness of incoming information from the experiment environment. This is due to the discrete nature of digital measuring equipment and computers. In accordance with this, an experiment in the representation of an automated system is nothing more than a sequence of commands to a performing or measuring device activated by a chain of events occurring in a controlled environment.

Now you can already clarify the shortcomings of the proposed approaches to the automation of a scientific experiment. The main disadvantages of the G language for use as a automation language for managing a scientific experiment:

1) A language describes a data stream, not a control flow; In this regard, it is difficult to present a description of a description of the course of an experiment with more than one device operating in parallel.

2) Difficulty in implementing your own modules

The main disadvantages of the language "Grafset":

1) The inability to consider the separate control flows associated only with events;

2) The language is intended for execution on a programmable logic controller (PLC), which in itself imposes a restriction on the scaling (increase in the number of measuring and control devices) of the experiment;

3) Management through the change of the device state. This approach makes concurrency difficult;

The solution to the above problems is to create a language to bypass these problems. Such a language seems to be divided into two levels of syntax: a graphic description of the interaction (description of the experiment) and pseudo-code describing specific messages sent to the language interpretation mechanism. For describing the interaction of devices, the marked Petri net [1, 2, 7, 8, 9] is well suited, and for describing messages transmitted to the performer (device, computer, etc., depending on the command), it is advisable to use a high-level programming language. The results of comparision is shown at Table 1.

Table 1. Comparision of languages

Language	Usability	Custom modules	Built-in concurrency
G	+	-	-
Graphset	+	+	-
Based on Petri's Networks	+	+	+

For the description of the algorithm, we apply the formal scheme - transition graph [5]. Thus, it is possible to achieve the informativeness of the scheme and remain in the field of formal languages, to simplify the verification of the algorithm using mathematical tools.

For the validation of the transition graph is to use its properties of correctness, such as: consistency, completeness, the absence of generating circuits other than loops.

In this case, it is considered that consistency in the graph of transitions is ensured if simultaneous transitions along any two or more arcs emanating from one vertex are prohibited in it [10, 11].

The consistency of the transition graphs (the conjunction of labels of any two arcs emanating from the same vertex is zero) is ensured by:

1) With the arrival of "contradictory" variable values at different times;

2) When working with variable fronts ("events");

3) Orthogonalization (complication of labels) of contradictory arcs (for example, when implementing a system of Boolean formulas);

4) Prioritization (taking into account the order of commands in the program when implemented in a way different from building a system of Boolean formulas);

5) "Splitting" vertices with conflicting arcs (increase in the number of states of the automaton).

The completeness of the transition graph (the disjunction of the marks of all arcs emanating from the vertex is equal to one) is checked after ensuring consistency. When implementing a transition graph using a system of Boolean formulas, all arcs emanating from each vertex should be labeled, and in other embodiments, the markings of loops for automata without an output converter or Moore's automata can be silent. In this case, it is assumed that the labeling of a loop at the vertex provides the "completeness" of the latter.

There are generating contours in transition graphs, if in at least one of them the conjunction of the marks of all the arcs that form it is not zero. The elimination of generating circuits is carried out by the same methods as the elimination of inconsistency (except for the prioritization).

Consider the following example. Three types of devices are used in the experiment: A, B and D. Device A implements control commands A1, A2. The sequence of commands of this device is limited to the sequence described by the GA grammar in Figure 1.

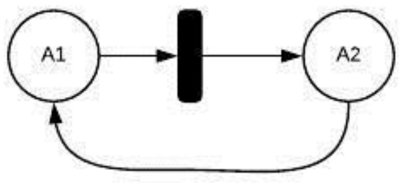

Fig.1. Device Status Graph A.

On the basis of this sequence, we construct a precedence matrix, taking into account that the return loop A2 - A1 may not be used in the algorithm (there may be zero or more such paths). Also between the vertices A1 and A2 in the algorithm can be located other states of other devices. For a correct designation, we introduce parametric symbols M ($\geqslant 0$) and N ($\geqslant 1$).

Table 2. The adjacency matrix of the state graph of device A

-	A1	A2
A1	0	N
A2	M	0

To find all possible paths, we use the following formula:

$$A^{*} = \sum_{i=1}^{K} A^{n} \tag{1}$$

where K is the dimension of the matrix.

Table 3. Matrix of all paths for device state graph A

-	A1	A2
A1	M*N	N
A2	M	M*N

Given the above parameters, the matrix is converted to

Table 4. The final matrix of all paths for the state graph

-	A1	A2
A1	M	N
A2	M	M

Device B implements commands B1, B2, and B3. The state graph of device B is depicted in Figure 2.

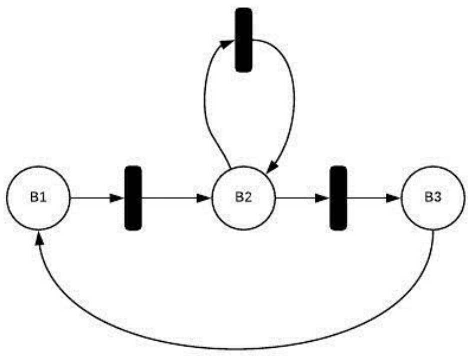

Fig.2. State graph of device B

The corresponding final matrix of all paths for the state graph of device B is displayed in Table 4.

Table 5. The final matrix of all paths for the state graph of device B

-	B1	B2	B3
B1	M	N	N
B2	M	M	N
B3	M	M	M

Device D implements commands D1, D2, and D3. The state graph of device D is depicted in Figure 3.

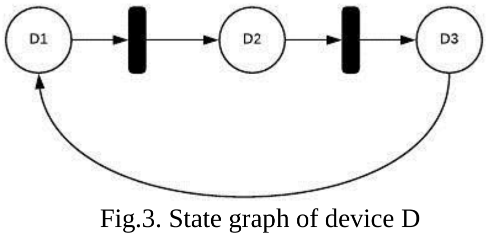

Fig.3. State graph of device D

The corresponding final matrix of all paths for the state graph of device D is displayed in Table 5.

Table 6. The final matrix of all paths for the state graph of device D

-	D1	D2	D3
D1	M	N	N
D2	M	M	N
D3	M	M	M

Also, in each algorithm there are marks of the beginning and end of the algorithm which are tracked as vertices of S. The state graph describing their relative position is shown in Figure 4.

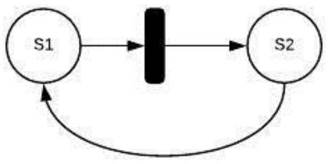

Fig. 4. The sequence of labels of the beginning and end of the algorithm S1 and S2

The corresponding final matrix of all paths for the state graph S is displayed in table 6.

Table 7. The final matrix of all paths for the state graph of device S

-	S1	S2
S1	M	N
S2	M	M

Let's combine the matrices of all and create a generalized matrix of all paths for all devices participating in the algorithm. Fill in the missing cells in the matrix with the parametric symbol M, since we cannot know in advance how the states of the devices will be connected.

For the specified grammar, we analyze the following algorithm, shown in Figure 5. Device A is sent a command A1 and the system is set to wait for an external event from this device. In the event that an eA1 event occurs, commands are sent to device B. In the event that an eB1 event occurs, events are sent to device D. At the end of the commands, command A2 is sent to command A2 and the algorithm ends.

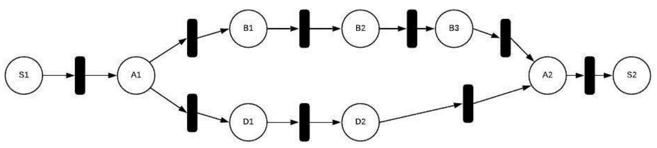

Fig.5. The algorithm to analysis.

Let’s construct an adjacency matrix corresponding to this graph.

To check the graph for compliance with the listed grammars, we apply the following algorithm:

1) To search for all possible paths in the graph, let us raise the matrix A to the power n, where n is the length of the path in the graph. Add up all possible variations of the resulting matrices.

$$A^* = \sum_{i=1}^{N} A^n \tag{2}$$

2) Let's compare element-wise matrix A* and grammar G.

3) In the event that all elements coincide, then the algorithm can be considered as satisfying the grammar G.

Consider the example of the transformation of the matrix A in accordance with formula (1).

Let's compare the obtained matrix with the grammar G (the selected cells do not satisfy the grammar property). The transformation sequence is shown at tables 7, 8, 9 ,10.

Table 8. The combination of the matrix of all paths G

-	A1	A2	B1	B2	B3	D1	D2	D3	S1	S2
A1	M	N	M	M	M	M	M	M	M	M
A2	M	M	M	M	M	M	M	M	M	M
B1	M	M	M	N	N	M	M	M	M	M
B2	M	M	M	M	N	M	M	M	M	M
B3	M	M	M	M	M	M	M	M	M	M
D1	M	M	M	M	M	M	N	N	M	M
D2	M	M	M	M	M	M	M	N	M	M
D3	M	M	M	M	M	M	M	M	M	M
S1	M	M	M	M	M	M	M	M	M	N
S2	M	M	M	M	M	M	M	M	M	M

Table 9. Generalized adjacency matrix A

-	A1	A2	B1	B2	B3	D1	D2	D3	S1	S2
A1	0	0	1	0	0	1	0	0	0	0
A2	0	0	0	0	0	0	0	0	0	1
B1	0	0	0	1	0	0	0	0	0	0
B2	0	0	0	0	1	0	0	0	0	0
B3	0	1	0	0	0	0	0	0	0	0
D1	0	0	0	0	0	0	1	0	0	0
D2	0	1	0	0	0	0	0	0	0	0
D3	0	0	0	0	0	0	0	0	0	0
S1	1	0	0	0	0	0	0	0	0	0
S2	0	0	0	0	0	0	0	0	0	0

Table 10. Resuling matrix A* after transformation.

-	A1	A2	B1	B2	B3	D1	D2	D3	S1	S2
A1	0	2	1	1	1	1	1	0	0	2
A2	0	0	0	0	0	0	0	0	0	1
B1	0	1	0	1	1	0	0	0	0	1
B2	0	1	0	0	1	0	0	0	0	1
B3	0	1	0	0	0	0	0	0	0	1
D1	0	1	0	0	0	0	1	0	0	1
D2	0	1	0	0	0	0	0	0	0	1
D3	0	0	0	0	0	0	0	0	0	0
S1	1	2	1	1	1	1	1	0	0	2
S2	0	0	0	0	0	0	0	0	0	0

Table 11. Comparision matrix

-	A1	A2	B1	B2	B3	D1	D2	D3	S1	S2
A1	0=0	2≥ 1	1≥0	1≥0	1≥0	1≥0	1≥0	0≥0	0≥0	2≥0
A2	0≥0	0=0	0≥0	0≥0	0≥0	0≥0	0≥0	0≥0	0≥0	1≥0
B1	0≥0	1≥0	0=0	1≥ 1	1≥ 1	0≥0	0≥0	0≥0	0≥0	1≥0
B2	0≥0	1≥0	0=0	0≥ 1	1≥ 1	0≥0	0≥0	0≥0	0≥0	1≥0
B3	0≥0	1≥0	0≥0	0=0	0=0	0≥0	0≥0	0≥0	0≥0	1≥0
D1	0≥0	1≥0	0≥0	0≥0	0≥0	0=0	1≥ 1	0=0	0≥0	1≥0
D2	0≥0	1≥0	0≥0	0≥0	0≥0	0=0	0=0	0≥ 1	0≥0	1≥0
D3	0≥0	0≥0	0≥0	0≥0	0≥0	0≥0	0=0	0=0	0≥0	0≥0
S1	1≥0	2≥0	1≥0	1≥0	1≥0	1≥0	1≥0	0≥0	0=0	2≥ 1
S2	0≥0	0≥0	0≥0	0≥0	0≥0	0≥0	0≥0	0≥0	0≥0	0=0

Let us consider in more detail the operation of the algorithm based on the example given above and illustrated by tables 8-11.

The algorithm begins with the construction of an adjacency matrix based on the graph presented in Figure 5.

Step 1. To build a comparison matrix, it is necessary to combine all the matrices of all possible paths into one. This must be done in accordance with the following rule: we supplement each of the matrices with missing rows and columns that correspond to the vertices of the graphs of the remaining graphs. Thus, we obtain a matrix of higher dimension. The dimension of the matrix is equal to the sum of the dimensions of the original matrices. The empty cells of the resulting matrix are filled with the value M, since at the primary stage we do not know which links are present in these graphs.

Step 2. Next, we combine all the matrices into one cell by cell according to the following rule: if at least one matrix in this cell contains a value other than M, then we write it, otherwise we write N. Following the example, we obtain the matrix of all possible paths in the graph represented in table 8.

Step 3. Further, the adjacency matrix of the algorithm graph, presented in Table 9, is used for calculations in accordance with formula (1). Thus, as a result of calculations, getting all possible paths of different lengths in the graph from 1 to N inclusive. The resulting values are presented in Table 10.

Step 4. After the calculations, it remains only to compare the values element by element between the matrix of all possible paths and the result of the calculation in the previous step. To do this, perform the following actions:

a) replace symbolic values with comparison operators: M with “> = 0”, and N with “> = 1”,

b) we check elementwise fulfillment of conditions in the cells of the matrix.

Those cells in which the values do not satisfy the conditions obtained, indicate connections between the vertices that cannot exist in accordance with the given graphs of the device status. Thus, the corresponding components of the matrix and the corresponding links in the graph violate the syntax of the resulting schema of the language and require certain manipulations to be corrected.

It is clear from the comparison matrix that the element D2 should be connected to D3, since this is due to the grammar, but this condition is not fulfilled, which means the algorithm is considered to be incorrect.

III. Results

The article describes the method of syntactic verification of the algorithm based on the transition graph for automating the experiment. The obtained technique, from the point of view of the authors, is most effective when using computing devices with a high degree of parallelism, such as FPGAs and graphics processors. In the future, it is planned to develop this approach in the experiment automation system, as the most promising.

The article deals with the problem of formal description of methods of organization of biomedical experiment. The need for a formal description of the experimental scheme is postulated. One of the ways of formal representation of biomedical experiment on the basis of transition graph, which is one of the variants of description of finite automata, is considered. Based on the analysis revealed that the main characteristics are the correctness and adequacy of the representation in the form of a graph of transitions. It is shown that the main problem of representation adequacy is the problem of description of nested sequences of actions corresponding to the beginning and end of the use of different components of the experiment. An example of such use is sequential and parallel activation of experimental equipment. Shown in the example, the correct sequence correspond to the realized experiments, but wrong-is not feasible. The necessity of checking the correctness of the graph as the main condition for the feasibility of the experiment is substantiated. One of approaches to the analysis of correctness of such graph on the basis of matrix calculations is presented. The developed algorithm validation on the basis of the method of construction of the matrix of transition counts. The basic transformations necessary to verify the correctness of the transition schedule of the biomedical experiment are given and an example of the algorithm application is given.

IV. Acknowledgment

The reported study was funded by RFBR according to the research project № 17-48-020074.

REFERENCES

[1] Bakusov L.M., Stepankin P.V. Geometric interpretation of the behavior of systems represented by Petri nets. Interuniversity scientific collection. Management of complex technical systems. Ufa, UAI, 1987.–pp.16-21.

[2] Bakusov L.M., Bakusov S.M., Nasyrov R.V. the algorithm of numerical simulation of systems with continiouse time// Fundamental research. Publisher: Publishing House "Academy of Natural History" (Penza), ISSN: 1812-7339/ 2013, №10-12, pp.2593-2598.

[3] http://www.plcdev.com/introduction_grafcets

[4] http://labviewwiki.org/G

[5] http://is.ifmo.ru/books/switch_pdf/oglavlenije.pdf

[6] http://is.ifmo.ru/books/switch_pdf/_switch19.pdf

[7] Wu D.H., Schnieder E. (2018) Scenario-based system design with colored Petri nets: an application to train control systems. Software and Systems Modeling, vol.17, i.1, pp.295-317. doi:10.1007/s10270-016-0517-1

[8] Rova S., Meire P., Muller F., Simeoni M. Prenovi F. (2019) A Petri net modeling approach to explore the temporal dynamics of the provision of multiple ecosystem services. Science of the total environment, vol.655, i.6, pp. 1047-1061. doi:10.1016/j.scitotenv.2018.11.184

[9] Khan Y.I., Konios A. Guelfi N. (2019) A Survey of Petri Nets Slicing. ACM Computing Surveys, vol.51, i.5, no.109. doi:10.1145/3241736

[10] Machado P. Silva M.R., de Souza L.E., de Souze C.W., Netto R.S. (2018) Modeling Using Colored Petri Net of Communication Networks Based on IEC 61850 in a Microgrid Context. Journal of Control Automation and Electrical systems, vol.29, i.6, pp.703-717. doi:10.1007/s40313-018-0411-x

[11] Rodriguez-Fernandez V., Gonzalez-Pardo A., Camacho D. (2018) Automatic Procedure Following Evaluation Using Petri Net-Based Workflows. IEEE Transactions on Industrial informatics, vol.14, i.6, pp.2748-2759. Doi: 10.1109/TII.2017.2779177

[12] Pospisil T. (2017) Control Flow Models using Petri Nets for Model-based Testing. Proceeding of the 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), vol.1, pp.553-557

[13] Kirci M., Gunes E.O. (2017) An Adaptive Petri Net Model of Wheat Phenological Phases Under Environmental Conditions. 2017 6th International Conference and Agro-Geoinformatics, pp.136-141

[14] Charaf M.E., Azzouzi S. (2017) A Colored Petri Net Model For Control Execution of Distributed Systems. International Conference on Control Decision and Information Technologies, 2017 4th International Conference on Control, Decision and Information Technologies (CODIT), Barselona, Spain, pp.277-282