# CONSTRUCTING RANDOM SAMPLE PARALLEL LOGIC CONTROL ALGORITHMS

**Eduard I. Vatutin**

*Department of Computer Science*
*Kursk State Technical University*
*50 Let Oktyabrya, 94, 305040, RUSSIA*
*Tel: +7(4712) 56-43-13, E-mail: evatutin@rambler.ru*

**Abstract** — *Description of an algorithm that allows to automatically generate and visualize sample random logic control algorithms with preset parameters is given. Examples of generated algorithms are presented.*

## 1. INTRODUCTION

One of the promising approaches to the design of parallel logic control systems is based upon microcontroller network concept [1]. In the synthesis of such systems, there is a number of problems waiting for a solution. One of them is the optimal separation of parallel logic control algorithms into a set of concurrent sequential subalgorithms (called blocks) according to some functional and structural restrictions. The problem is known to be an NP-hard one and can't be solved precisely for algorithms with more then ~15 vertices (estimate) because of excessive time growth. Practical logic control algorithms are much more complicated (say ~100 vertices). To solve the given problem, a number of heuristic algorithms (for example, [2, 3]) have been proposed, one of them has been developed by the authors [4]. While constructing a separation, the considered methods use various heuristic techniques, therefore, it is important to compare these methods to each other to select the best of them according to the specified criterion. For this purpose, a considerable set of initial data samples (in this case – logic control algorithms) is required. The number of available practical examples of parallel control algorithms as well as the number of algorithms which can be thought up within a limited time interval is essentially limited and pear comparison of methods can't be attained, therefore we need automatic generation of sample parallel logic control algorithms with preset general parameters. Note that comparison results received on sample random algorithms should be checked up for known real examples of parallel control algorithms.

## 2. REPRESENTATION OF ALGORITHMS USING A FRAGMENT TREE

In the structure of any correct parallel logic control algorithm, it is possible to point out a set of typical fragments:

- begin fragment (BE);
- linear way (LW);
- alternative branching (ALT);
- parallel fragment (PAR);
- different types of loops – loops with pre- and post-conditions, loops with breaking (LB, LA, LM).

(Within the considered approach, parallel alternatives and parallel loops are not examined (don't participate in generation), since it is possible to replace them with consecutive fragments in combination with a parallel fragment as shown in fig.1)

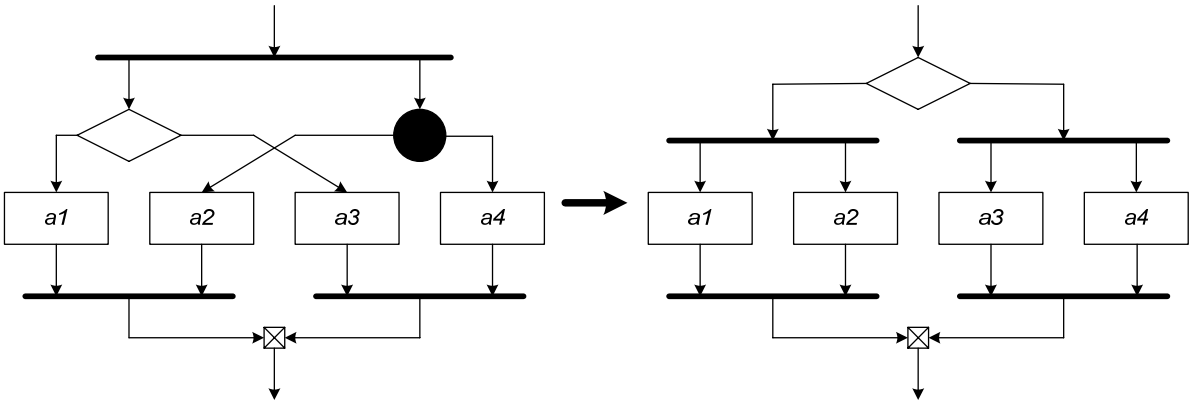


*Fig. 1. Replacement of parallel alternative fragment*

Any algorithm (it is possible, with minor transformations, which do not vary the order of operator execution) usually figured as a graph consisting of vertices and arcs can be represented as a tree of the above listed fragments (fig. 2).

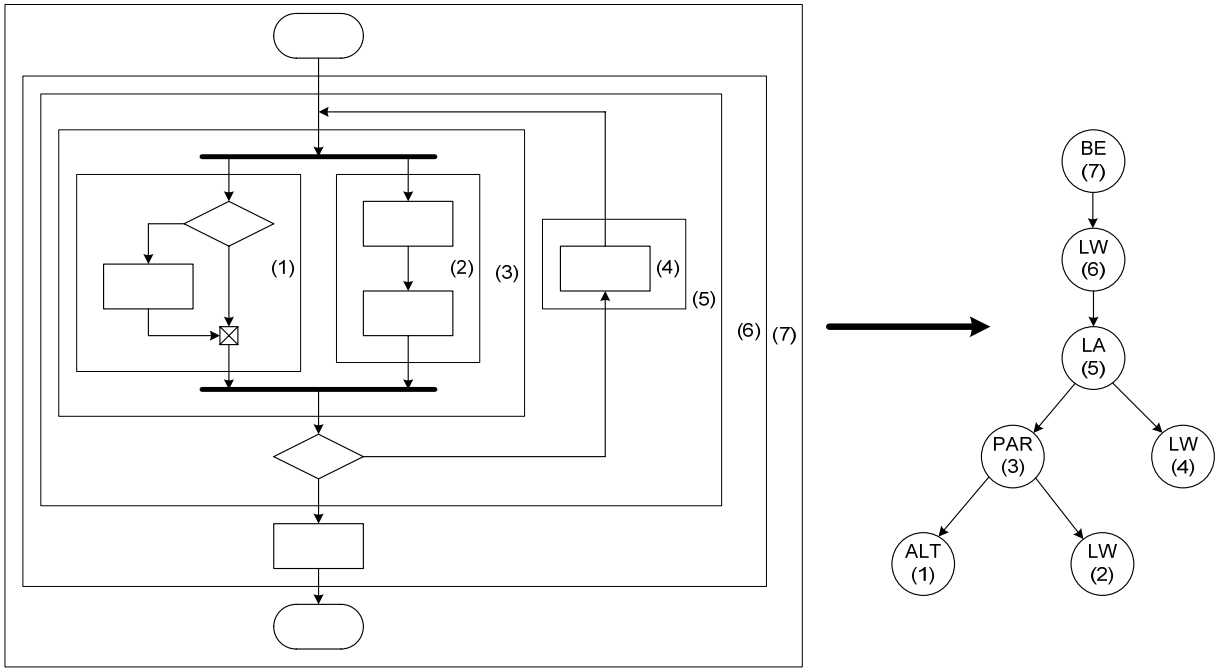


*Fig. 2. Representation of an algorithm as a fragment tree*

## 3. SYNTESIS OF SAMPLE RANDOM PARALLEL LOGIC CONTROL ALGORITHMS

In the light of the opportunity to represent a logic control algorithm as a fragment tree, the proposed method of random generation can be presented in two stages:

1. Constructing a fragment tree downward (from root to leaves).
2. Building a random algorithm by integration fragments upward (from leaves to root).

The given specific order of exploration of the tree is caused by that the resulting random algorithm needs to be represented not only as a graph, but also as a set of figures [5] to retain the opportunity of graphic display of the generated algorithm.

First stage of the algorithm generation is listed below:

1. Insert a begin fragment (BE) to the fragment tree.
2. Randomly select an available parental fragment in the tree proportionally to the number of free arcs in it. (Free arcs are those that can take part in the process of replacement by affiliated fragment. The number of free arcs is initially determined by the type of the fragment and its parameters, and then is decremented at each addition of the affiliated fragment (fig. 3))
3. Randomly choose a type of the fragment to be inserted proportionally to the probabilities of various fragment types to occur, and set its parameters (number of paths, number of vertices, probabilities of arc activation, etc.) Add this fragment to the tree.
4. If the finalization condition holds (got required total of vertices or fragments), then end up the algorithm, otherwise go to step 2.

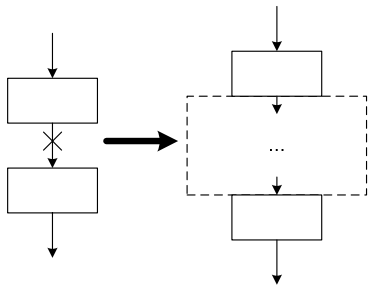


*Fig. 3. Replacement of a free arc by an affiliated fragment*

As the result of the first stage, we have a fragment tree that can be used for the synthesis of required algorithm as a graph. Algorithm of the second stage looks as follows:

1. Mark all fragments of the tree as not considered.
2. Choose a fragment in the tree that has no considered affiliated fragments. Build its geometry: determine the coordinates of vertices, arcs and affiliated fragments, determine geometric sizes of each fragment (fig. 4). Adjust links between arcs of affiliated fragments (if they are present) and vertices of the current fragment. Determine the activation probabilities and control flow for arcs of affiliated fragments. Determine at random vectors of micro-operations and logic conditions for operator and condition vertices. Mark this fragment as considered.
3. If all the fragments are considered, end up the algorithm, otherwise go to step 2.

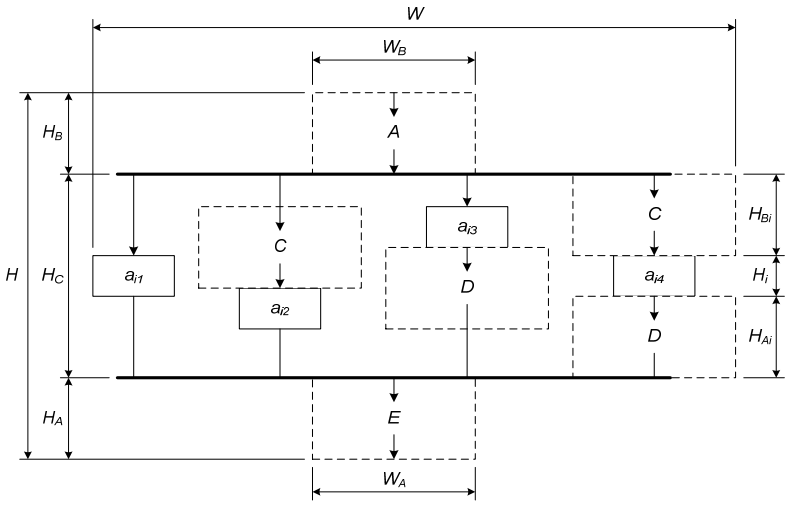


*Fig. 4. Determining geometric structure of fragment using a parallel fragment as an example*

Note that generally geometrical parameters of affiliated fragments (fig. 4) are not equal:

$$H_i \neq H_j,\ W_i \neq W_j,\ \forall i, j \notin \{B, C, A, B_1, ..., B_n, C_1, ..., C_n, A_1, ..., A_n\},$$

that makes certain difficulties in calculation of path arrangement, coordinates of vertices and arcs.

Upon performing above listed actions, we have an algorithm represented as a graph and all its vertices and arcs have coordinates on a plane and can be simply displayed on the screen.

## 4. EXAMPLES OF GENERATED SAMPLE RANDOM CONTROL ALGORITHMS

With the purpose of confirmation of the described approach, we shall provide several examples of acyclic algorithms generated according to our algorithm described above and realized as a part of software system PAE [5].

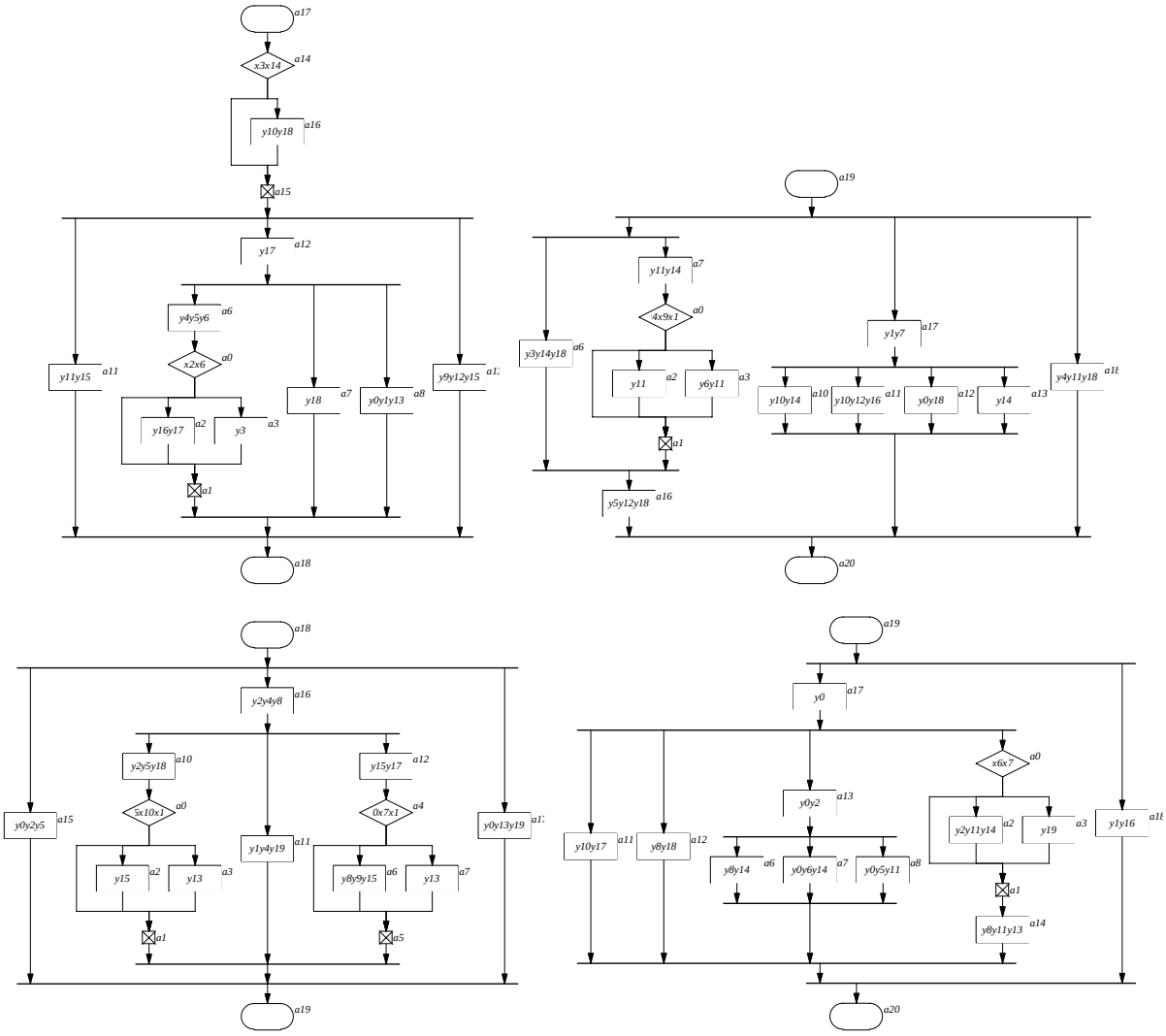


*Fig. 5. Examples of generated sample random algorithms*

## 5. ASSUMPTIONS

Note that generated algorithms slightly differ from real those due to some admissions:

1. The lengths of linear paths are equiprobable.
2. The numbers of paths constituting alternative and parallel fragments are equiprobable.

3. The numbers of micro-operations and logic conditions of different vertexes are equiprobable.
4. The probabilities of presence of micro-operations and logic conditions in a vertex are equal.

However, as a whole the accepted admissions should not seriously influence objectivity of our comparison of methods.

Автор выражает благодарность своему научному руководителю

## 6. CONCLUSION

1. An algorithm for constructing sample random logic control algorithms having any size and predefined general parameters is developed and tested.
2. Generated random algorithms are represented not only as a graph, but also as a convenient view for graphic output.
3. Based on the developed algorithm, we can perform not only method comparison, but also we can test methods for errors.
4. In view of the considered approach, we can simply determine such an important characteristic of a generated parallel algorithm as the degree of parallelism.
5. Speed characteristics of the developed realization of the algorithm are ~1500 algorithms with ~20 vertexes per second using a PC equipped with Intel Celeron 850 MHz processor (CPUID=068Ah).

## 5. ACKNOWLEDGEMENTS

The author would like to thank my scientific supervisor I.V. Zotov.

## 7. REFERENCES

[1] I.V. Zotov et al., Optimal algorithm separation in the design of microcontroller networks, Automatic Control and Computer Sciences, No.5, 1997, pp. 41-52

[2] S.I. Baranov et al., A Method for Representation of Parallel Flow-Charts with a Set of Sequential Flow-Charts, *Automation and Computer Engineering*, n. 5, 1984, pp. 74–81 (in Russian)

[3] A.D. Zakrevsky, *Parallel Algorithms of Logic Control*, Minsk, 1999 (in Russian)

[4] E.I. Vatutin, I.V. Zotov, A Method for the Construction of Suboptimal Separations of Parallel Control Algorithms, *Parallel Computations and Control Problems (PACO'04)*, Moscow, 2004, pp. 884–917 (in Russian)

[5] E.I. Vatutin, I.V. Zotov, A Software System for the Construction of Separations of Parallel Logic Control Algorithms, *System Identification and Control Problems (SICPRO'06)*, Moscow, 2006, pp. 2239–2250 (in Russian)