The International Workshop
on Discrete-Event System Design, DESDes'01,
June 27÷29, 2001; Przytok near Zielona Gora, Poland

ON OPTIMAL STATE-ASSIGNMENT OF SYNCHRONOUS PARALLEL AUTOMATA*

Yury POTTOSIN

Institute of Engineering Cybernetics, National Academy of Sciences of Belarus,
Surganov str., 6, 220012, Minsk, BELARUS, *pott@newman.bas-net.by*

Abstract. *Three algorithms for assignment of partial states of synchronous parallel automata are considered. Two of them are heuristic, the third one is exact, i.e. the number of coding variables obtained by this algorithm is minimum. It is based on covering a non-parallelism graph of partial states by complete bipartite subgraphs. One of the heuristic algorithms is based on the solving the same problem but it uses an approximate method for it. The other of them is known as iterative one. The results of application of these algorithms on some pseudo-random synchronous parallel automata and the method for generating such objects are given.*

Key Words. *Parallel Automaton, State-Assignment, Problem of Covering, Generating Pseudo-Random Objects*

1. INTRODUCTION

The parallel automaton is a functional model of a discrete device and is rather convenient to represent the parallelism of interactive branches of controlled process [17]. The main distinction between a parallel automaton and a sequential one (finite state machine) is that the latter can be in only one state at any moment while the parallel automaton can be in several partial states simultaneously. A set of partial states a parallel automaton can be at simultaneously is called a *total state*. Any two partial states in which an automaton can be simultaneously are called *parallel*.

A parallel automaton is described by the set of strings of the form $\mu_i : -w_i \to v_i \to \nu_i$, where w_i and v_i are elementary conjunctions of Boolean variables that define the condition of transition and the output signals respectively, μ_i and ν_i are labels that represent the sets of partial states of the parallel automaton [17]. Every such a string should be understood as follows. If the total state of the parallel automaton contains all the partial states from μ_i and the event w_i has been realised in the input variable space, then the automaton is found to be in the total state that differs from the initial one by containing partial states from ν_i instead of those from μ_i. The values of output variables in this case are set to be such that $v_i = 1$.

*This work is supported by ISTC project B-104-98.

If $-w_i$ and $\rightarrow v_i$ are removed from the string it can be interpreted as a transition (μ_i, ν_i) in a Petri net. So, the set of such reduced strings can be considered as a Petri net being a "skeleton" of the given parallel automaton. Here we consider only those parallel automata whose skeleton is an α-net [17] that is a subclass of live and safe expanded nets of free choice which are studied in [6].

In state assignment of a parallel automaton, partial states are encoded by ternary vectors in the space of introduced internal variables that can take values 0, 1 or "–", orthogonal vectors being assigned to non-parallel states and non-orthogonal vectors to parallel states [2, 15, 16]. The orthogonality of ternary vectors means existence of a component having opposite values (0 and 1) in these vectors. It is natural to minimise the dimension of the space that results in the minimum of memory elements (flip-flops) in the circuit implementation of the automaton.

The methods to solve the state assignment problem for synchronous parallel automata are surveyed in [4]. Two heuristic algorithms are considered here. One of them is based on iterative method [3], the other reduces the minimisation of the number of memory elements to the problem of covering a non-parallelism graph of partial states by complete bipartite subgraphs [9]. To solve the problem of covering it uses a heuristic technique. The third algorithm considered here is exact, i.e. the number of coding variables (memory elements) obtained by this algorithm is minimum. It also finds a cover of a non-parallelism graph of partial states by complete bipartite subgraphs but using an exact technique [11]. These three algorithms were used to encode partial states of a number of synchronous parallel automata obtained as pseudo-random objects. The pseudo-random parallel automata with the given parameters were generated by a special computer program. The method for generating such objects is described. The results of this experiment allow one to decide about the quality of the algorithms. Similar experiments are described in [18] where another approach was investigated and the pseudo-random objects were Boolean matrices interpreted as partial states orthogonality matrices of parallel automata.

2. EXACT ALGORITHM

Below we refer to this algorithm as Algorithm A. It is based on covering a non-parallelism graph G of partial states by complete bipartite subgraphs. Let the complete bipartite subgraphs $B_1, B_2, \ldots, B_m$ form a shortest covering of G, and B_k for every $k = 1, 2, \ldots, m$ be associated with a Boolean coding variable z_k so that $z_k = 1$ for the states relative to one partite set of B_k and $z_k = 0$ for the states relative to the other. Then the values of coding variables $z_1, z_2, \ldots, z_m$ represent the solution sought for. The covering is considered here as every edge of G belongs to at least one B_k.

The first step to the solution is finding all maximum complete bipartite subgraphs of G. Three ways to do it are given in [10, 15]. Then one must obtain the shortest covering of edge set of G by those complete bipartite subgraphs. For decreasing the dimension of the covering problem the reduction rules for initial graph are used in construction the covering matrix. These rules are described in [15]. Another way to decrease the dimensions of the problem is given in [11]. It can be applied if G can be represented as $G = G_1 + G_2$, i.e. in the form of the result of join operation on two graphs [7]. The decrease is achieved if one of G_1 and G_2, say G_1, is a complete graph. This is typical for an automaton whose "skeleton" is α-net. Then the covering problem is solved only for G_2. When the cover is obtained the codes of the states which are associated with the vertices of G_2 are chosen as shown above. These codes form a Boolean space of coding variables. If they don't occupy all the space, the codes of the states associated with the vertices of G_1 are placed in the rest. The space is extended, if necessary, to the size enough for placing all state codes. In this case a non-redundant cover must be found rather than

a shortest one. Algorithm A realises this method. The idea of using decomposition as the means to reduce the dimension of the task is rather fruitful. For example, one can see in [8] another case of using decomposition to decrease the dimension of the problem of our field.

3. HEURISTIC ALGORITHMS

3.1. Algorithm B

The NP-hardness of covering problem [5] doesn't allow it always to be solved in acceptable time. Therefore the heuristic algorithm is proposed in [9] that obtains in many cases the shortest cover. We call it Algorithm B. It consists of two stages. At the first stage the sequence of graphs $G_2, G_3, \ldots, G_n = G$ is considered, where G is the non-parallelism graph of the given automaton with $V = \{v_1, v_2, \ldots, v_n\}$ as the set of vertices, and G_i is the subgraph of G induced by the set of vertices $V_i = \{v_1, v_2, \ldots, v_i\}$. Having the cover of G_i the transition from it to the cover of G_{i+1} is carried out. At the second stage the obtained cover is improved (if possible). This improvement consists in removing some complete bipartite subgraph from the covering and in the attempt of reconstruction the cover by adding edges to remained subgraphs. This procedure repeats for all elements of the cover. The complete bipartite subgraphs are obtained concurrently with constructing the cover.

3.2. Algorithm C

The other heuristic algorithm is based on the iterative method suggested in [3]. We refer to this algorithm as Algorithm C. The iterative method assumes the definition of parallelism relation and an initial coding matrix for partial states (the initial matrix may be empty). The matrix is extended in the process of coding by introducing additional coding variables that makes it possible to separate non-parallel partial states in certain pairs. To separate two states means to put opposite values (0 and 1) to some coding variable in the codes of these states. The method consists in iterative executions of two procedures: introducing a new coding variable and defining its values in codes of non-separated yet non-parallel partial states. These procedures are being executed until all non-parallel states have been separated. Minimising the number of introduced coding variables the method minimises the Hamming distance between codes of states related by transitions as well. The aim of this is the minimisation of the number of switchings of RS type flip-flops in circuit realisation of a parallel automaton.

Introducing a new coding variable is accompanied with separating the maximal number of non-separated yet non-parallel partial states by this variable. For this purpose at each step of the procedure of defining the values of the due variable, a state is chosen to encode by this variable. This state should be separated from the maximal number of states encoded already by this variable. The number of states that are not separated from the chosen one and have been encoded by this variable must be maximum. A new coding variable is introduced if the inner variables having been introduced don't separate all non-parallel partial states from each other.

4. GENERATING PARALLEL AUTOMATA

Any string of the form $\mu_i : -w_i \rightarrow v_i \rightarrow v_i$ in automaton specification we call a transition, and a set of transitions with the same μ_i a sentence. The algorithm for generating parallel automata is described in detail in [12] where a parallel automaton is constructed as a system of three pseudo-random objects. They are the skeleton of the automaton that is an α-net specified in the form of a sequence of pairs (μ_i, v_i), the ternary matrix $\boldsymbol{X}$ representing conjunctions w_i, and the

ternary matrix $\boldsymbol{Y}$ representing conjunctions v_i. In our task the α-net is enough, therefore we shouldn't describe the way of generating $\boldsymbol{X}$ and $\boldsymbol{Y}$ here.

The given beforehand parameters of every pseudo-random α-net generated by a special computer program are the number of places (partial states of the automaton) p, the number of transitions t, and the number of sentences s.

Generating pseudo-random parallel automata as systems of three mentioned above objects with given beforehand parameters would not be difficult if no correctness demands exist without which there is no sense to execute algorithms intended for such automata. Proceeding from the correctness properties of a parallel control algorithm that are named in [16], let us consider the following properties of a parallel automaton, that guarantee its correctness in our case. It must be irredundant (there is no transition that can be never done), recoverable (it can return to the initial total state from any other one), and self-coordinated (any transition cannot be started before it ceases).

Irredundancy, recoverability, and self-co-ordination of a parallel automaton corresponds to liveness and safety of the related α-net [16]. The characteristic properties of α-net are the initial marking of it consisting of one element, {1}, and the sets of input places of two different transitions coinciding or disjoining. In the Petri net theory the reduction methods for checking liveness and safety are well known [1], where the initial net is transformed according to certain rules with preserving these properties. The transformations reduce the dimension of a given net and so facilitate the checking liveness and safety of the net.

To check liveness and safety of α-nets the application of two rules is sufficient [16]. The first rule consists in deleting loops i.e. the transitions where $\mu_i = \nu_i$. The second one is as follows. Let a set of places π not containing place 1 be such that for every transition (μ_i, ν_i), $\pi \cap \mu_i \neq \varnothing$ implies $\pi = \mu_i$ and $\pi \cap \nu_i = \varnothing$, and $\pi \cap \nu_i \neq \varnothing$ implies $\pi \subseteq \nu_i$. Besides, there exists at least one transition with $\pi \cap \nu_i \neq \varnothing$. Then all transitions (μ_j, ν_j) with $\pi = \mu_i$ are removed and every transition (μ_k, ν_k) with $\pi \subseteq \nu_k$ is substituted by the set of transitions that are obtained from (μ_k, ν_k) by replacing π by sets ν_j from those transitions (μ_j, ν_j) where $\pi = \mu_j$. A live and safe α-net is proved in [14] to be completely reducible, i.e. the application of these rules leads to the net that consists of the only transition (1,1). This implies the way of generating live and safe α-nets that consists in transformations that are inverse to the above.

5. EXPERIMENTAL RESULTS

Algorithms A, B, and C are realised in computer programs and the corresponding modules are included as components into ISAPR that is a research CAD system [13]. The program for generating pseudo-random parallel automata is included into ISAPR as well. This program was used to generate several parallel automata. The results of partial state assignment are shown in Table 1. One of the automata whose partial states were encoded, RAZ, was not generated by the program mentioned above. It was obtained from a real control algorithm.

As it was noted, only the parameters of α-net, i.e. the number of places p, the number of transitions t, and the number of sentences s were considered. Besides those, the number of maximum complete bipartite subgraphs in the graph G of non-parallelism of partial states of the given automaton may be of interest. Algorithm A uses the method that decomposes graph G into two subgraphs, G_1 and G_2, G_1, being complete. So, the maximum complete bipartite subgraphs were found in G_2. The calculations were performed on a computer of AT type with the 386 processor.

6. CONCLUSION

The technique of investigation of algorithms for state assignment of parallel automata is described in this paper. The experimental data show that Algorithms B and C are quite competitive to each other, although the speed of Algorithm C is higher than that of Algorithm B. Algorithm A is intended to be applied for automata of small dimension. It can be used as a standard algorithm and helps one to appreciate the quality of solutions obtained by heuristic algorithms.

Table 1. Experimental results: p, t, and s are parameters of α-nets, b is the number of maximum complete bipartite subgraphs of G_2.

Name	*p*	*t*	*s*	*b*	*Algorithm A*		*Algorithm B*		*Algorithm C*	
					Code length	*Run time*	*Code length*	*Run time*	*Code length*	*Run time*
AP2	**20**	**18**	**18**	**75**	**6**	**13 min. 28 sec.**	**7**	**6 sec.**	**7**	**3 sec.**
APR1	**20**	**21**	**19**	**8**	**5**	**8 sec.**	**6**	**7 sec.**	**5**	**3 sec.**
APR2	**20**	**21**	**19**	**4**	**5**	**5 sec.**	**6**	**8 sec.**	**5**	**3 sec.**
APR3	**20**	**21**	**15**	**7**	**4**	**6 sec.**	**5**	**3 sec.**	**6**	**3 sec.**
APR6	**20**	**28**	**15**	**43**	**5**	**2 min. 23 sec.**	**6**	**8 sec.**	**6**	**3 sec.**
APR7	**20**	**30**	**15**	**55**	**5**	**49 sec.**	**6**	**8 sec.**	**6**	**3 sec.**
APR8	**20**	**15**	**15**	**49**	**5**	**1 min. 28 sec.**	**5**	**5 sec.**	**5**	**3 sec.**
RAZ	**20**	**21**	**19**	**1033**	**9**	**3 h. 46 m. 22 s.**	**9**	**8 sec.**	**10**	**4 sec.**

REFERENCES

[1] S.M.Achasova, O.L.Bandman, *Correctness of Concurrent Computing Processes*, Nauka, Siberian Division, Novosibirsk, 1990 (in Russian)

[2] M.Adamski, M.Wegrzyn, "Field programmable implementation of current state machine", *Proc. of the Third International Conference on Computer-Aided Design of Discrete Devices (CAD DD'99), Vol.1*, Institute of Engineering Cybernetics, NAS Belarus, Minsk, pp.4-12, 1999

[3] L.D.Cheremisinova, "State assignment for parallel synchronous automata", *Izvestia AN BSSR, Ser. fisiko-tehnicheskih nauk, No.1*, pp. 86-91, 1987 (in Russian)

[4] L.D.Cheremisinova, Yu.V.Pottosin, "Assignment of partial states of a parallel synchronous automaton, *Proceedings of the International Conference on Computer-Aided Design of Discrete Devices CAD DD'95"*. Wydawnictwo Uczelniane Politechniki Szczecinskiej, Minsk-Szczecin, pp. 85-88, 1995

[5] M.R.Garey, D.S.Johnson, *Computers and Intractability: A Guide to the Theory on NP-completeness*, W.M.Freeman & Company, San Francisco, Ca., 1979

[6] M.T.Hack “Analysis of production schemata by Petri nets”, *Project MAC TR-94*, Cambridge, 1972

[7] F.Harary, *Graph Theory*, Addison-Wesley Publishing Company, Reading, Mass., 1969

[8] A.Karatkevich, “Hierarchical decomposition of safe Petri nets”, *Proc. of the Third International Conference on Computer-Aided Design of Discrete Devices (CAD DD’99), Vol.1*, Institute of Engineering Cybernetics, NAS Belarus, Minsk, pp. 34-39, 1999

[9] Yu.V.Pottosin, "Covering a graph by complete bipartite subgraphs", *Design of Discrete Systems*, Institute of Engineering Cybernetics of Academy of Sciences of Belarus, Minsk, pp. 72-84, 1989 (in Russian)

[10] Yu.V.Pottosin, “Finding maximum complete bipartite subgraphs in a graph” *Automatization of Logical Design of Discrete Systems*, Institute of Engineering Cybernetics of Academy of Sciences of Belarus, Minsk, pp. 19-27, 1991 (in Russian)

[11] Yu.V.Pottosin, "A method for encoding the partial states of a parallel automaton with minimal-length codes", *Automatic Control and Computer Sciences*, *N 6*, Allerton Press, Inc., New York, pp. 45-50, 1995

[12] Yu.V.Pottosin, “Generating parallel automata”, *Methods and Algorithms for Logical Design*, Institute of Engineering Cybernetics of Academy of Sciences of Belarus, Minsk, pp. 132-142, 1995 (in Russian)

[13] N.R.Toropov, *Research CAD System for Discrete Control Devices: Materials on Software for Computers*, Institute of Engineering Cybernetics of Academy of Sciences of Belarus, Minsk, 1994 (in Russian)

[14] V.V.Tropashko, “Proof of the conjecture of complete reducibility of α-nets”, *Design of Logical Control Systems*, Institute of Engineering Cybernetics of Academy of Sciences of BSSR, Minsk, pp. 13-21, 1986 (in Russian)

[15] A.D.Zakrevskij, “Optimization of matrix of partial state assignment for parallel automata”, *Formalization and Automatization of Logic Design*, Institute of Engineering Cybernetics of Academy of Sciences of Belarus, Minsk, pp. 63-70, 1993 (in Russian)

[16] A.D.Zakrevskij, "Parallel logical control algorithms: verification and hardware implementation", *Computer Science Journal of Moldova, Vol.4, No.1*, pp. 3-19, 1996

[17] A.D.Zakrevskij, *Parallel Algorithms for Logical Control.* Institute of Engineering Cybernetics of National Academy of Sciences of Belarus, Minsk, 1999 (in Russian)

[18] A.Zakrevskij, I.Vasilkova, "A quick algorithm for state assignment in parallel automata", *Proc. of the Third International Conference on Computer-Aided Design of Discrete Devices (CAD DD’99), Vol.1*, Institute of Engineering Cybernetics, NAS Belarus, Minsk, pp.40-44, 1999