ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 166

7th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2019)

The Synthesis of Complex Logical Controllers with Variables of Boolean and Fuzzy Logics

Alexander Verevkin
Department of automation of technological processes and production
FSBEI HE "Ufa State Petroleum Technological University"
Ufa, Russia
1apverevkin@mail.ru

Oleg Kiryushin
Department of automation of technological processes and production
FSBEI HE "Ufa State Petroleum Technological University"
Ufa, Russia
kirov9562@mail.ru

***Abstract*—The paper describes the method of synthesis of control devices and logic control algorithms for distributed technological objects. Control algorithms for this devices are often described as a set of rules, most of which are written in terms of Boolean and fuzzy variables. To transform the production system into a mathematical model in the form of logical sequences (system of logical expressions) in terms of two-digit logic exists a method, which includes procedures and algorithms for conflict resolution and identification of deadlock situations in the operation of automata, based on the intermediate representation of rule sets in the form of Petri nets. The paper describes the modification of this method for the synthesis of logic devices and algorithms for the joint use of input-output variables Boolean (two-digit) and fuzzy logic. The method is implemented by performing the following steps: preparation of rules, generalization of rules, representation of a set of rules in the form of Petri nets, synthesis of Boolean expressions for the state variables, synthesis of Boolean expressions for activation of transitions (consequental) and formation of output variables. The method allows to synthesize hierarchical state machines in terms of boolean and fuzzy logic.**

Keywords—logic control, fuzzy automata, hierarchical system, coordination of automata

I. INTRODUCTION

The problems of synthesis of logic algorithms and devices have a long history, and a large number of approaches and methods have been developed to solve them [1, 2, 3].

At the same time, when solving a number of practical problems, first of all, for the cases of synthesis of logical control algorithms and devices difficult to control objects, the existing methods do not provide an answer to overcome a number of fundamental difficulties, which include:

1. The 'curse' of dimensionality. The essence of this concept and the associated difficulty is that with increasing the dimension of the problems of logical control by the number of inputs and outputs (especially for finite-automata models), there are great technical difficulties in the synthesis procedures associated with the way of representation of automata models and the formation of mathematical expressions (logical sequences) that relate the values of the output signals to the input and States of automata.

2. Lack of methods for synthesis problems in which there is a combination of logical control problems in terms of Boolean and fuzzy logic.

3. Ensuring the compactness of logical algorithms, allowing their implementation on standard and not expensive controllers. Here we are talking about the fact that the previous two difficulties, in principle, in many cases can be bypassed by recording the logic of the control device in the form of, for example, a production system or a predicate system [4]. But it is known that the use of such control models involves the development of algorithms that are implemented, as a rule, in high-level programming languages and require large computational resources, including large time-consuming information processing. In addition, at large dimensions of models of such types there are fundamental issues of ensuring the performance of algorithms [4, 5].

This article discusses the approach and implementing its methods of synthesis of complex logic algorithms and devices based on the use of hierarchical automata systems of clear (two-digit) and fuzzy logic [6, 7].

II. METHOD OF SYNTHESIS OF LOGICAL DEVICES

In papers [8-11] and some others sets the direction of synthesis of complex hierarchical control systems, that developed, based on the use of discrete-event models in the form of Petri nets with their subsequent decomposition into subnets, each of which is put in accordance with the finite state machine.

However, issues related to the generalization of synthesis procedures to logical control devices containing input and (or) output variables of both two-digit and fuzzy logic (or analog variables) are not considered in these works.

The problems of synthesis of logical algorithms, first of all, subautomates of the lower level (coordinated logical control subsystems), remain unresolved to the full extent, because at this level there is a problem of using both discrete and analog or fuzzy input and output variables. The synthesis of coordinating automata is usually carried out in terms of two-digit logic [12, 13].

Figure 1, as an example, shows the scheme of interaction of the lower level automata (A1, A2, A3, A4) and the coordinating automaton (CA).

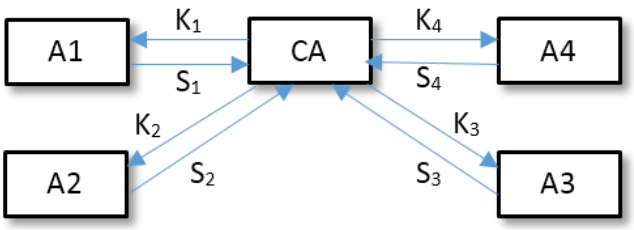

Fig. 1. Automats interaction scheme

An example of Petri nets to coordination of the machine during the sequential startup of machines the lower level is presented in figure 2 where each transition t1 ,..., t4 are mapped to the section of the rules.

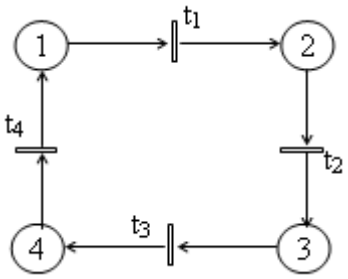

Fig. 2. Petri net of CA

In the variants of parallel operation and (or) synchronization of lower-level automata, the Petri net will have a more complex appearance.

Next, we consider a modification of the method of synthesis of logical devices and algorithms, characterized by the possibility of sharing input-output variables Boolean (two-digit) and fuzzy logic. The method is implemented by performing the following steps.

Step 1. Preparation of rules. We consider the case when in the initial set of rules (production system) it is possible to use both Boolean variables and linguistic statements or fuzzy variables. If analog variables are needed, they can be represented as fuzzy variables with a triangular membership function. The base of the triangle will correspond to the range of values of analog variables.

The any j-th rule has the form "IF P_j THEN t_j "or" IF P_j THEN t_j ELSE t_{1j}", where P_j is the antecedent (conditional part) of the rule, t_j, t_{1j} are the consequents (executive part) of the rule, j = 1,2,... If the P_J condition is not satisfied, the transition to the t_{1j} consequent is performed.

The task of writing logical sequences in terms of two-digit logic is solved in papers [1,3,8]. Therefore, to reduce the problem to the known, at step 1 it is necessary to replace fuzzy variables in P_j and consequents with clear ones, i.e. fuzzy terms with an arbitrary form of the membership function (MF) [5, 8] are replaced by clear ones (with a rectangular form of MF). Return to fuzzy variables, i.e. fuzzification of logical expressions, will be performed at step 5.

Consequents t_j, t_{1j} and others are associated with some executive actions, in particular, with the commands to change the current control U by ΔU for the actuators of automation systems. Relation is carried out through the initialization table [3,5]. An example of the table initialization in the original version is shown in figure 3. In the example it is accepted that the variable U_1 is fuzzy, and the variable U_2 is Boolean. Suppose that the values of the commands are initialized when t_1, t_2, t_3 consequents are activated in any rules of the production system.

For the fuzzy variable U_1, the commands "decrease"(-ΔU1),"increase"(+ΔU1) are indicated by an arrow,"0" - the control remains unchanged.

For Boolean variables, there are two values of the output variable: "1" - "enable" ("increase"), "0" – "disable" ("decrease"). Thus, in the transition from fuzzy to clear variables need to encode three States and, accordingly, need two Boolean variables U_{11} and U_{12}, see figure 4.

Defuzzification of the input analog or fuzzy variables is carried out by known methods by introducing S Boolean variables (as comparator outputs) for each term [8].

	U_1	U_2
t_1	$U_1\downarrow$	0
t_2	$U_1\uparrow$	1
t_3	0	—

Fig. 3. Examples of initialization tables with fuzzy variables

	$U_{11}=-\Delta U_1$	$U_{12}=+\Delta U_1$	U_2
t_1	1	0	0
t_2	0	1	1
t_3	0	0	—

Fig. 4. Examples of initialization tables with boolean variables

Step 2. Generalization of rules. The number of rules is reduced by combining rules that have the same consequents [8].

Step 3. Representation of a set of rules in the form of Petri nets. As for a class of Petri nets was developed algorithmization (formalized) methods of determining their properties, analysis of efficiency of the rule set is reduced to a more simple and reliable procedures for the analysis of Petri nets (determining liveness, safety and other properties). In accordance with [8, 14-17] a complex network, as a rule, are decomposed into subnets with the aim of synthesis of finite automata of the lower level (see figure 1). For example, each machine of the lower level can control a separate unit [8]. An example subnet for one rule is shown in figure 5. The production system is interpreted as a Petri net by its sequential formation on the basis of the rules execution order. Note that this method does not apply to a rare variant of the production system, when the rule changes the order of the rules.

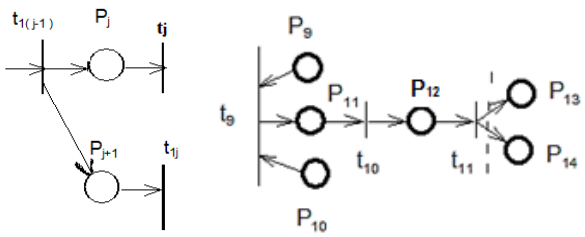

Fig. 5. Examples of subnets

Step 4. Synthesis of Boolean expressions for the state variables. The properties of the resulting subnets are usually determined by constructing a reachability graph, determining the number of States each subnet can be in. To encode the States of the j-th subnet (j = 1,2,...), intermediate logical state variables R_{ji} (i = (1, n)) are introduced, where n is the number of subnets and their corresponding subautomates. Synthesis of logical expressions for state variables by means of enlarged state tables (EST) is described in [8]. Figure 5 shows an example of a subnet, and the corresponding EST is shown in figure 6. The Initial state from which the subautomat is started to run is state 1. The transition to state 2 is the conjunction of the conditions of the positions P9ΛP10 if you have permission (flag) from the coordinating machine B. At the transition to the second state, by analogy with Moore machines, activates transition t_9, performing the role of the output symbol. Further, under the conditions of P_{11}, P_{12}, transitions to States 3, 4 are carried out with the activation of transitions t_{10}, t_{11}, respectively. After firing of the transition t_{11} in question of subautomata ends in a coordinating subatomic about the signal is transmitted. Further, with the permission of the coordinating sub-machine, the following sub-machines of the lower level can start working, for example, from positions P_{13} or (and) P_{14}, (separated by a dotted line). After the sub-machine cycle is completed, it returns to the initial state 1.

Number of state	BΛP9ΛP10	P11	P12	
1	2	1	1	
2	2	3	2	t9
3	3	3	4	t10
4	1	1	4	t11

Fig. 6. Examples of state table

Step 5. Synthesis of Boolean expressions for activation of transitions (consequental). Logical expressions for control actions are written. For this:

1 Defines the number of intermediate variables (functions). For the considered example of the j-th subnet, two functions $R_{j1}(R_{j1}, R_{j2}, B\Lambda P_9\Lambda P_{10}, P_{11}, P_{12})$ and $R_{j2}(R_{j1}, R_{j2}, B\Lambda P_9\Lambda P_{10}, P_{11}, P_{12})$ are required. Note that logical expressions for positions are written through input logical variables (see Step 1).

2 Meanings of the names of the transitions are written as a function of a combination of intermediate variables known methods [4]. For example: $t_9 = t_9(R_{j1}, R_{j2})$, $t_{10} = t_{10}(R_{j1}, R_{j2})$, $t_{11} = t_{11}(R_{j1}, R_{j2})$.

Step 6. Formation of output variables.

6.1 To generate output signals of Boolean type in accordance with the initialization table, each active transition is assigned the value of the output signal. Note that a properly synthesized hierarchical automaton cannot simultaneously activate transitions for which different values of output signals are assigned in the initialization tables, i.e. there should be no conflicts.

6.2 In order to generate analog output signals, it is necessary to:

6.2.1 In position expressions, replace Boolean logic variables that were derived from fuzzy or analog variables (see Step 1) with their membership functions [6, 7].

6.2.2 Replace Logical functions Λ, V,"NOT" of Boolean type with fuzzy operations t-norm, t-conorm (S-norm) and fuzzy negation respectively. Two commonly used logical extensions are shown in figure 7.

In this case, the names of positions in General acquire the meaning of MF, because the calculated fuzzy logical expressions can contain values in the interval [0; 1]. The initialization table (see figure 6) can take the form (see figure 8). The fuzzy value of the output variable U must be defuzzified by one of the known methods, for example, averaging:

$$\Delta U = (0{,}1 + 0{,}5 - 0{,}8 - 0{,}2) / 4 = -0{,}1,$$

i.e. in this example, the control effect should be reduced by 10%.

Fuzzy logic extension	$a \hat{\wedge} b$	$a \tilde{\vee} b$	$\tilde{\bar{a}}$
Logic	min(a; b)	max (a; b)	1 - a
Algebraic	a·b	a + b - a·b	1 - a

Fig. 7. Fuzzy Logic extension

	$U_{11} = -\Delta U_1$	$U_{12} = +\Delta U_1$	U_2
t_1	0.8	0.1	0
t_2	0.2	0.5	1
t_3	0	0	—

Fig. 8. Table of initializations after fuzzification of input variables

ACKNOWLEDGMENT

The method presented in the article allows to synthesize a set of hierarchically organized logical expressions, which can be implemented on microcontrollers, according to the initial descriptions of the control device functioning in the form of sets of rules, including both Boolean and fuzzy logical variables. At the same time, this method, unlike classical methods, is devoid of the "curse of dimension", contains the stages of testing and decomposition of the algorithm into subautomates using Petri nets and can combine control algorithms in terms of Boolean and fuzzy logic.

REFERENCES

[1] Fridman A., Menon P. Teoriya i proektirovanie pereklyuchatelnih skhem. / Moscow: Mir, 1978. -578 p. (in Russian)

[2] Verevkin A.P., Kiryushin O.V. Razrabotka logicheskih algoritmov dlya celey realizacii na mikrokontrollerah.// Pribori i sistemi. Upravlenie, kontrol i diagnostika, 2001, #11. –P.P. 5-8. (in Russian)

[3] Smirnov I.N. Sintez sistem upravleniya na logicheskih elementah. – Leningrad: LGU, 1975. -368 p. (in Russian)

[4] B. Dunham, R. Fridshal. The problem of simplifying logical expressions. // https://doi.org/10.2307/2964570.

[5] Iskusstvenniy intellekt: Spravochnik v 3 kn. / Pod red. V.N. Zaharova, V.F. Khoroshevskogo. –Moscow: Radio i svyaz, 1990. Vol. 1 – 426 p.; Vol.2 – 304 p.; Vol. 3.– 368 p. (in Russian)

[6] Y. Dote. Introduction to fuzzy logic. / Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics. DOI: 10.1109/IECON.1995.483332.

[7] L.-X. Wang. Fuzzy systems are universal approximators / [1992 Proceedings] IEEE International Conference on Fuzzy Systems. DOI: 10.1109/FUZZY.1992.258721.

[8] Verevkin A.P., Kiryushin O.V. Avtomatizaciya tehnologicheskih processov i proizvodstv v neftepererabotke i neftehimii. –Ufa: USPTU, 2005. -171 p. (in Russian)

[9] Mesarovich M., Mako D., Takahara I. Teoriya ierarhicheskih mnogourovnevih sistem. –Moscow: Mir, 1973. – 344 p. (in Russian)

[10] Sovremennoe sostoyanie teorii issledovaniya operaciy./ pod. red. Moiseeva N.N. –Moscow: Nauka, 1979. – 464 p. (in Russian)

[11] Verevkin A.P., Kiryushin O.V. Upravlenie sistemoy podderzhaniya plastovogo davleniya s ispolzovaniem modeley konechno-avtomatnogo vida. // Territoriya Neftegaz, #10, 2008. –P. 14-19. (in Russian)

[12] Rihards Krislauks, Kaspars Balodis. On the Hierarchy Classes of Finite Ultrametric Automata. / Proceedings of SOFSEM 2015: Theory and Practice of Computer Science, Lecture Notes in Computer Science, vol. 8939, pp. 314–326, 2015. (indexed in Scopus and Web of Science)

[13] Kaspars Balodis, Janis Iraids, Rusins Freivalds. Structured Frequency Algorithms. / Proceedings of TAMC 2015, Lecture Notes in Computer Science, vol. 9076,pp. 1–12, 2015. (indexed in Scopus and Web of Science)

[14] Zurawski R., Zhou M. Petri Nets and Industrial Applications: A Tutorial. / IEEE Transactions on Industrial Electronics (1994) 41(6) 567-583. DOI: 10.1109/41.334574.

[15] Murata T. Petri Nets: Properties, Analysis and Applications. / Proceedings of the IEEE (1989) 77(4) 541-580. DOI: 10.1109/5.24143.

[16] Graff C., Giardina C. Dynamic Petri-Nets: A new modeling technique for sensor networks and distributed concurrent systems. / Proceedings - IEEE Military Communications Conference MILCOM (2005) 2005. DOI: 10.1109/MILCOM.2005.1605733.

[17] D. A. Zaitsev. Paradigm of computations on the Petri nets. // Avtomat. i Telemekh., 2014, no. 8, 19–36; Autom. Remote Control, 75:8 (2014), 1369–1383.