Modeling and Control of Discrete Event Systems in Practice

Branislav Hrúz,

Dipl. Ing., PhD., Associate Professor ✉ Dept. of Automatic Control Systems, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology Ilkovičova 3, 812 19 Bratislava, Slovakia 💻hruz@kasr.elf.stuba.sk

1. Abstract

Two specification tools for the discrete event dynamic systems (DEDS) are described in the paper. The first one is the finite automaton which represents the basic finite state machine from which many other tools are derived. The second and very often used nowadays is Petri net. Both tools are analyzed, compared and used for the DEDS control. Their use is illustrated on an practical example.

Keywords: discrete event dynamic systems, finite automata, Petri nets.

2. Introduction

An important group of the man-made systems can be characterized as the discrete event dynamic systems – DEDS. As always, understanding, creation and governing of such systems requires to find out an adequate systems description and specification, in other words, to find the system model. The basic property of the DEDS is that they are the event-driven systems in comparison with the time-driven systems. The event relations in DEDS can be very complicated ones. Think of the event synchronization, concurrency, mutual exclusion, conflicts etc. in such systems like flexible robotic manufacturing systems, air or surface transportation systems with ticket reservation, telecommunication systems, computer networks etc.

Having a model of the discrete event dynamic system the design of its control is possible. The control specification model is mostly of the same kind as the model of the controlled DEDS. Whatever the approach to the control specification can be, finally the control program must be written and implemented in a suitable hardware means. The programmable logic controllers are such widespread and reasonable means.

3. Finite automata in the DEDS control

The model abstractions used for the DEDS are based on the assumption that the number of the system states is finite and that the system transitions occur in the discrete time points. The finite automata served for several decades as the DEDS models. What makes them such a powerful model of DEDS is that they are effectively incorporating two basic system categories, namely the state and the transition. The finite automata are described in detail e.g. in the book by Carrol and Long (1989).

To illustrate the use of the finite automata let us consider an example. A system consisting of three pneumatic pistons A, B, C executing sequences of movements is depicted in Fig. 1. The pistons are operated by the electric solenoid valves. The pistons stop only in the end positions. Each piston has two limit sensors indicating the piston end positions. SA_1 is the

sensor for the basic position and SA_2 for the pushed out position of the piston A. Analogously are denoted the sensors for the other pistons. $\mathbf{w} = \begin{pmatrix} w_A \\ w_B \\ w_C \end{pmatrix}$ is the vector control variable, its value $\mathbf{w} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$ will be denoted as $\mathbf{w}_{100}$, and analogously for other values. If $_{\mathrm{R}}$F†□□ valve V_A is set so that piston A moves to the right end position, if $w_A = 0$ the piston moves back to the end position.

The variables associated with the position sensors can be treated as logical variables, as well. Value $a_1 = 1$ means that piston A is in the basic position that is left in Fig. 1. Value $a_1 = 0$ means inversely that piston A is not in the basic position. Similarly it is for the other variables.

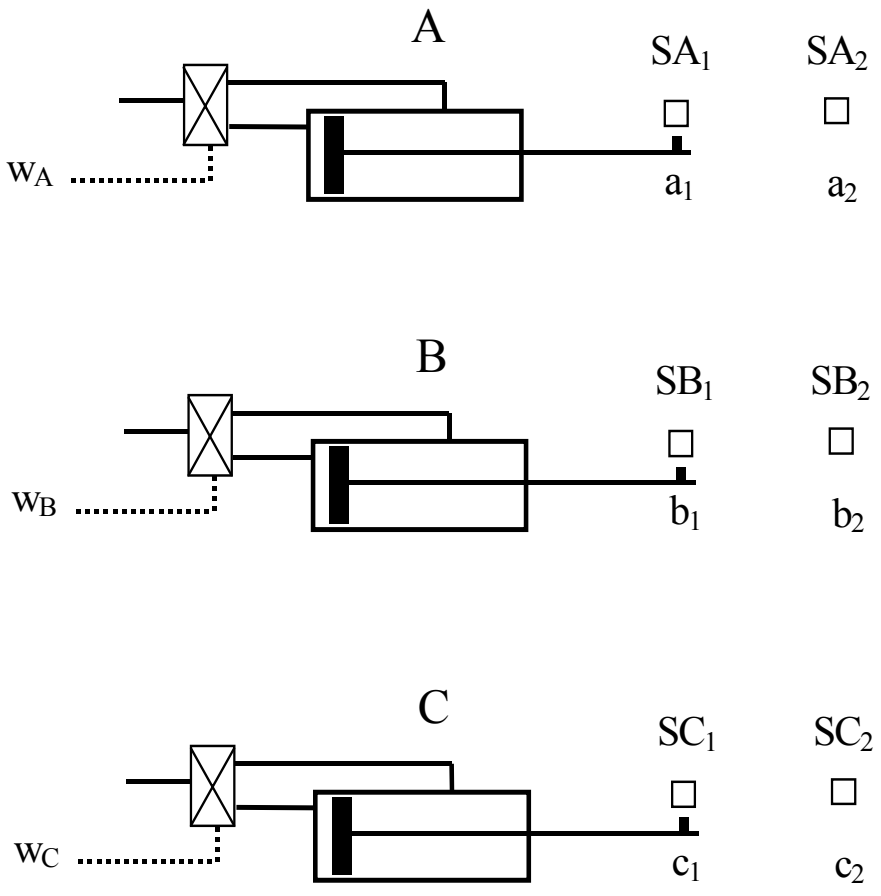


Fig. 1. Electro-pneumatic system with three pistons.

Various movement sequences can be programmed on the described piston systems. As an example let a sequence be

$$S = START,\ A+,\ \{A-,\ B+,\ C+\},\ \{B-,\ C-\} \tag{1}$$

where $A+$ represents that the piston has to do the forward stroke and $A-$ represents the return one. Expression $\{A-,\ B+,\ C+\}$ represents three concurrent actions to take place simultaneously.

We could specify the system behavior from the point of view of an observer with help of the finite automaton. Instead we will pass directly to the system control specification using the finite automaton as illustrated in Fig. 2. Compare Fig. 2 with the definition of the finite automaton as follows

$$A = (Q,\ X,\ Y,\ \delta,\ \lambda,\ q_0) \tag{2}$$

where

Q is the finite set of the states $Q = \{q_0, q_1, ..., q_{15}\}$

X is the finite set of inputs $X = \{START, a_1, a_2, b_1, b_2, c_1, c_2\}$

Y is the finite set of outputs $Y = \{w_{100}, w_{011}, w_{000}\}$

δ is the transition function $\delta : (Q \times X) \to Q$, $\times$ is the Cartesian product

λ is the output function $\lambda : Q \to Y$

$q_0 \in Q$ is the initial state of the automaton

The isomorphic representation of the above finite automaton is as the drawn ordinary mathematical directed labeled graph with the nodes Q and the labeled arcs defined by the function δ with the labels from the set Râ□□â□÷&FW&VB□□triple (q_i, x_j, q_k) corresponds to the arc coming out of the state q_i going into the state q_k and labeled by the input x_j.

The control sequence specified with the automaton ensures the sequence expressed by (1). The following function of the system is required. After an input START the piston A is activated. When it arrives at the end position the signal a_2 is detected and the automaton transits from state q_1 to state q_2 and output w_{011} is generated. Pistons B and C go simultaneously forward and A returns back. When both B and C are in the right end position (state q_7 with output w_{000} when A is not back or state q_{12} with the same w_{000} when A is already back) they are returning back regardless of the position A. When all three pistons are in the basic position the system returns in the initial state and new START can be triggered. The speed of the movements are different because of the different load of the pistons. It is required to follow all combinations of the piston positions and this is enabled by the automaton model in Fig. 2.

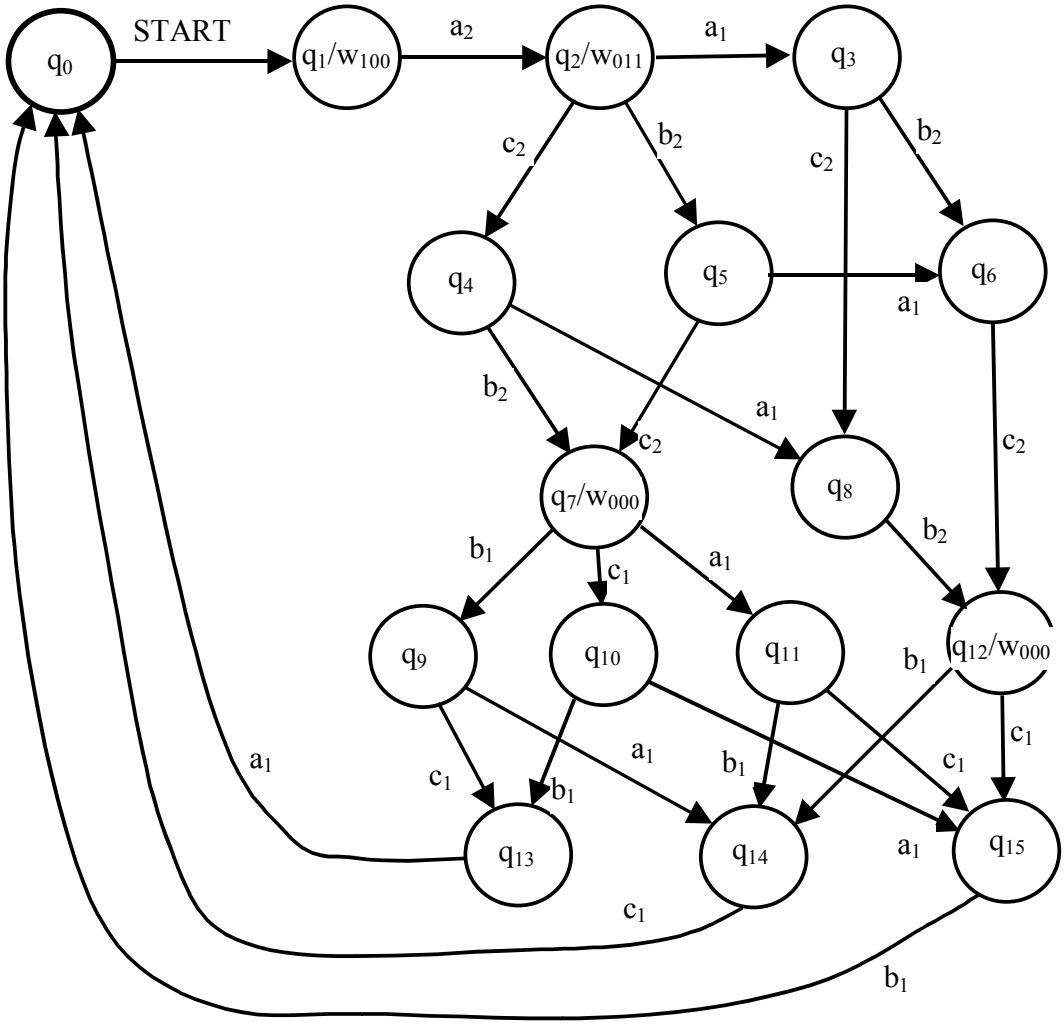


Fig. 2. Finite automaton model of the piston system.

4. Petri nets in the DEDS control.

Petri nets represent another powerful tool for the DEDS modeling and control design (see e.g. Abel, 1990; Zhou, 1995, Hrúz, 1994). They are the finite state machines models alike the finite automata described in the previous section. The Petri net is defined as follows

$$PN = (P, T, F, W, M_0) \tag{3}$$

where $P = \{p_1, p_2, ..., p_n\}$ is a finite set of places

$T = \{t_1, t_2, ..., t_m\}$ is a finite set of transitions

$F \subseteq (P \times T) \cup (T \times P)$ is the relation defining the arcs

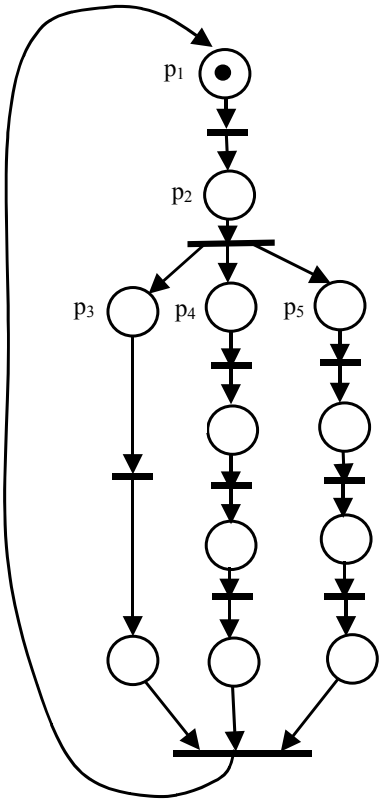
$W : F \to \mathbf{I}^+$ is the function that maps the arcs into positive integers called weights, the weights are labels of the arcs

$M_0 : P \to \mathbf{N}$ is the function which maps the places into the set of the natural numbers, this function creates the so called marking of the Petri net, it defines the number of tokens in each place.

A transition of the Petri net can fire if each pre-place of the transition has so many tokens as it is the weight of the arc connecting the pre-place with the considered transition. After firing the tokens according to the weight are removed from each pre-place and so many tokens are deposited into each post-place what is the weight of the arc from the transition into the post-place.

The Petri nets interpreted for control are augmented in comparison with the above defined ordinary Petri nets so that logical conditions can be associated with a transition and commands to the system can be associated with a place. The logical conditions represent additional conditions for the transition firing. The commands are activated when into a place comes the token.

Using the Petri net interpreted for control the piston system is modeled in Fig. 3.



The logical variables corresponding to the system variables signaling events are associated with the transitions. The control variables and their required values are written by the places. The concurrent processes and synchronization is better transparent in the Petri net as in the finite automaton model.

5. Conclusion

The approach using the finite automata and Petri nets for the control synthesis of the DEDS was shown in the paper. The practical aspects of the control design were considered. Both tools were compared on a practical example. There are other graphical tools based on the tools described in this paper. Grafcet and statecharts are interesting from the application point of view.

Grafcet (David and Alla, 1992) was developed from the binary Petri nets for the practical purposes of the control synthesis. It uses slightly different graphical symbols for the system entities. Grafcet became the European standard for the specification of the logical control of the DEDS.

Statecharts (Harel, 1987) have good properties as for the system processes parallelism and hierarchy.

The specification created by any of the mentioned tools is basis for the system control programming. It is the next step in the control design. For that purpose can be used the ladder logic diagrams, instruction list, a real-time programming language, logical blocks or other modern programming procedure.

6. References.

[1] Abel, D. (1990). Petri-Netze für Ingenieure. Springer-Verlag, Berlin.

[2] Carrol, J, and Darrel Long (1989). Theory of finite automata. Prentice Hall, Englewood Cliffs.

[3] David, R., and H. Alla (1992). Petri nets and Grafcet. Prentice Hall, New York.

[4] Harel, D. (1987). Statecharts: a visual formalism for complex systems. Science of Computer Programming, **8**, 231-274.

[5] Hrúz, B. (1994). Discrete event systems modelling and real-time control. Journal of Electrical Engineering (Elektroetecnický časopis), **45**, pp. 363-370.

[6] Zhou, M.C. (1995). Petri nets in flexible and agile automation. Kluwer, Boston.