SUPERVISORY CONTROL OF A PNEUMATIC SYSTEM USING PLC

İ. Tolga Hasdemir *hasdemir@elk.itu.edu.tr*
Salman Kurtulan *kurtulan@elk.itu.edu.tr*
Leyla Gören *goren@elk.itu.edu.tr*

Istanbul Technical University, Faculty of Electrical & Electronics Engineering, Control Systems Division
34390*, Maslak, Istanbul, Turkey*

Key words: Discrete Event Systems, Supervisory Control, PLC

ABSTRACT

In this study we present a supervisory control system and implement it on a pneumatic system controlled by a PLC. We introduce a step-by-step implementation procedure including a realization methodology of finite state machines by PLCs. A hierarchical control structure is proposed and implemented. The pneumatic system control problem presented in this study is a typical logical Discrete Event System (DES) control problem, therefore it is easy to generalise the methodology for most of the DES control problems.

I. INTRODUCTION

An important portion of today's manufacturing systems can be classified as event driven systems or discrete event systems (DESs). Although it is possible to analyse and control this class of industrial systems formally, general methods applied in the industry are intuitive rather than being formal. For small sized DES control problems, intuitive methods may yield practical solutions, but as the controlled system gets larger and more complex, formal methods need to be applied.

The supervisory control theory (SCT) introduced by Ramadge and Wonham [1] provides a powerful framework for control of discrete event systems. The theory enables synthesis of closed loop control systems for DESs by making some assumptions on the system that is to be controlled, and on the supervisor that is to control the system. Although the SCT has received a wide acceptance in academy and some applications of SCT have been reported in the literature [2-4, 6] it has not been accepted in the industry yet. This is mainly due to the difficulties arising in physical implementation of SCT [4,5].

Programmable Logic Controllers (PLCs) have been used in industrial applications for more than 25 years and today most of the automated manufacturing systems use PLCs as control units. Therefore the potential physical platform to realise a supervisor in industry is the PLC.

Finite state automata are used for representing the plant model and the supervisor in SCT. Implementation of SCT necessitates an appropriate method for developing a PLC program corresponding to the automaton that represents the theoretical supervisor. Therefore implementing the SCT is a matter of developing an appropriate PLC program, which will force the PLC behave as an automaton. The methods for developing PLC codes for this purpose and the problems that may arise in doing so are discussed in [4-6].

One of the main difficulties arising in SCT implementations is state-space explosion. System models are generally built as combination of subsystems and the number of states of the global system grows exponentially with the number of subsystems. Computational complexity caused by large number of states (may be more than 10^{20}) can make it impossible to design and implement a supervisor for a given DES control problem. Therefore some formal and informal techniques are used to reduce computational complexity either by reducing sizes of the automata or using different control structures comprising more than one supervisor. Informal techniques generally depend on the designer's experience and preferences such as ignoring some of the details (i.e. some of the states of the physical system) that are not necessary to appear for the given control problem [4]. A formal method called Local Modular Control reduces computational complexity and provides significant memory savings by using multiple supervisors and local models of the plant [6].

In this study, implementation of SCT on an educational pneumatic manufacturing system is presented. A hierarchical control structure comprising three supervisors for two subsystems of the pneumatic system have been proposed. Two of the supervisors control their respective subsystems while the remaining supervisor manages the execution order of these two supervisors. It is shown that the language corresponding to the desired behaviour of the overall system is controllable. Implementation of the supervisors is provided by a PLC.

The organization of the paper is as follows. A brief introduction of DESs, SCT is given in Section 2. The pneumatic system is introduced and modelled in Section 3. Section 4 discusses the control structure and explains synthesis and implementation of the supervisors.

II. PRELIMINARIES

Discrete event systems evolve on spontaneously occurring events. Let E be finite set of the events that drive the system. The set of all finite concatenations of events in E is denoted as E^*. An element in this set is called a string. The number of events gives the length of the string. The string with zero length is denoted as ε. A subset $L \subseteq E^*$ is called a language over E. For a string $s \in E^*$, $\overline{s}$ denotes the prefixes of s and is defined as $\overline{s} = \{s_p \in E^* : \exists t \in E^* (s_p t = s)\}$. Extending this definition to languages we get prefix closure of a language L denoted as $\overline{L}$. A language L satisfying the condition $\overline{L} = L$ is said to be prefix closed [7].

An automaton, denoted by G, is a six tuple $G = (X, E, f, \Gamma, x_0, X_m)$ where X is the set of states, E is the finite set of events. f: X x E → X is the partial transition function on its domain. Γ: $X \rightarrow 2^E$ is the active event function. Γ(x) is the set defined for every state of G and represents the feasible events of x. x_0 is the initial state and $X_m \subseteq X$ is the set of marked states representing a completion of a given task or operation. The language generated by G is denoted by $\mathcal{L}(G)$ and defined as $\mathcal{L}(G)=\{s \in E^* : f(x_0,s) \text{ is defined}\}$. The language marked by G is denoted by $\mathcal{L}_m(G)$ and defined as $\mathcal{L}_m(G)=\{s \in E^* : f(x_0,s) \in X_m\}$ [7]. Product and parallel composition operations are defined for expressing two forms of joint behaviour of multiple automata that operate concurrently. Product of two automata, say G1 and G2 is denoted as G1 X G2 and represents the synchronous behaviour of the two automata. Thus, an event occurs in the resulting automata if and if it occurs in both automata. The generated and marked languages of the resulting automaton will be $\mathcal{L}(G1 \text{ X } G2) = \mathcal{L}(G1) \cap \mathcal{L}(G2)$ and $\mathcal{L}_m(G1 \text{ X } G2) = \mathcal{L}_m(G1) \cap \mathcal{L}_m(G2)$, respectively. Parallel composition of automata G1 and G2 is denoted as G1||G2. In the resulting automaton common events occur synchronously, while the other events occur asynchronously. Therefore, if the event sets E1 and E2 of the automata are equal then parallel composition of G1 and G2 will be equivalent to the product of G1 and G2.

Automata are generally represented by state diagrams. Figure 1 shows a simple automaton with two states. State 0 with a directed line pointing it is the initial state. State 1 with double circle is the marked state of the automaton. Directed lines (transitions) represent transition functions of the automaton. Labels of the transitions correspond to events. The events associated with all the transitions from a state give the active event set of that state.

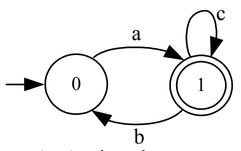


Figure 1. A simple automaton.

The supervisory control theory (SCT) uses formal languages to model the uncontrolled behaviour of discrete event systems and specifications for the controlled behaviour. The objective is to restrict the behaviour of the system to a desired behaviour, which is represented by the specifications. This is done by disabling some events to prevent some of undesired strings from occurring in the system. The decision of which event to be disabled is made by another simultaneously executing system, called the supervisor. The supervisor is also represented by an automaton. The active event set associated with a state of the supervisor gives the events that are allowed to occur in the corresponding state of the controlled system.

In SCT events are divided into two classes as controllable events and uncontrollable events. The set of controllable events is denoted by E_c, while E_{uc} represents the uncontrollable event set. The supervisor has no effect on uncontrollable events, that is the supervisor cannot prevent an event of E_{uc} from happening.

The SCT assumes that strings of events are generated by the system. When the supervisor disables a controllable event this means that this event cannot be executed by the controlled system. However, when real world problems are considered, this assumption causes implementation difficulties since systems of real world do not generate events rather they generate responses to given commands. For this reason, an input-output perspective was proposed in [2]. In this perspective the supervisor not only prevents controllable events from occurring but also generates commands that drive the system.

Our interpretation is somewhat different from above mentioned input-output perspective. In the implementation stage, although the commands are generated by the device that functions as the supervisor, these commands are interpreted as controllable events generated by the system. This means that the supervisor simply allows these controllable events to occur in this interpretation. Considering the commands generated by the controlling device as a part of the controlled system makes it possible to use SCT in its original form.

In SCT the existence of a supervisor is guaranteed if the desired language satisfies a condition. This condition is called as the controllability condition and defined as $\overline{K}E_{uc} \cap M \subseteq \overline{K}$ where $\overline{K}$ is the language that will be generated under the control of the supervisor and M is the language generated by the uncontrolled system [7].

THE PNEUMATIC SYSTEM

The pneumatic system (MAP 205) shown in Figure 2 is a flexible automation minicell built for educational purposes comprising pneumatic manipulators and a PLC (Siemens S7-200). It demonstrates the assembly of a rotary mechanism comprising a base, bearing, shaft and a cover. Assembly of rotary mechanism is carried out by means of four manipulators. Each manipulator performs its specific task in the assembly process. The tasks that the manipulators perform are base feeding, mounting of the bearing, insertion of the shaft, positioning of the cover, respectively.

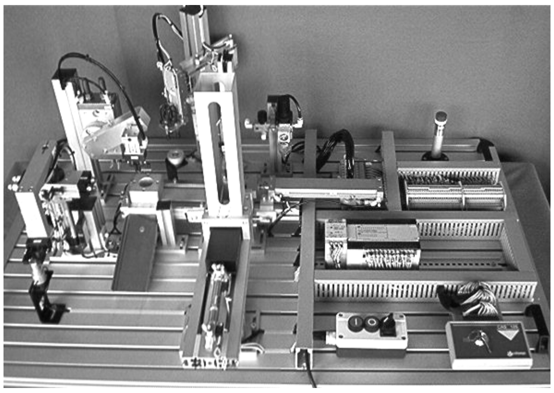
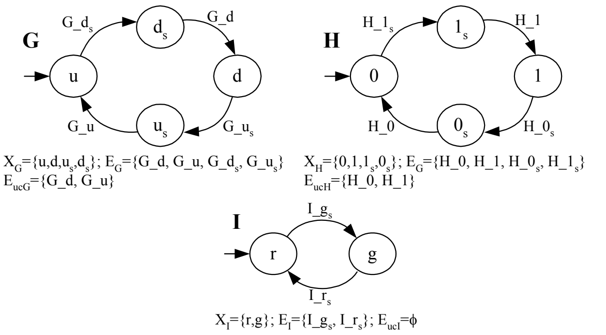


Figure 2. The pneumatic system (MAP 205)

Although there are four manipulators in the system, in this study two of the manipulators were used to realise a part of total manufacturing process. The tasks that are performed in this study are mounting of the bearing and insertion of the shaft. It is straightforward to extend the design procedure to the complete case.

Automaton Models of Manipulators

We will call the two manipulators as Manipulator 1 (Mn1) and Manipulator 2 (Mn2), respectively. Our first objective is to build the model of the system, that is the uncontrolled behaviour of the two manipulators. We will focus on each manipulator separately. Mn1 carries out the process necessary for the assembly of the shaft by placing it inside the previously inserted bearing. There are two cylinders and a gripping arm driven by pneumatic valves on Mn1. One of the cylinders makes a rotary movement of 90^0 while the other one makes a vertical movement. We will represent the vertical cylinder, rotary cylinder and the gripping arm of Mn1 by the letters G, H, and I respectively. The process of placing the shaft inside the bearing can be described as follows. With the command G_d_s, cylinder G lowers down to position "d" and arm I grips the shaft, which is initially deposited in a seat. Then G rises up to the position "u" and H starts to change its position from "0" to "1" immediately after H_1_s command. Upon the occurrence of H_1 event signing that H is in position 1, G is lowered and then the shaft is released to place it in the bearing. In order to take Mn1 to its initial position, the commands used for inserting the shaft are followed in reverse order. The automata representing the behaviour of each part of Manipulator 1 is shown in Figure 3. The set of events generated by sensors and used as inputs to the PLC forms the uncontrollable events. The set of events that drive the pneumatic valves forms the controllable events. Controllable events (commands) are denoted by a subscript s. Subscript s is also used for the states reached by these controllable events. States with subscript s indicate that a specific command has been given but not completed yet. For example, after the occurrence of the event G_d_s automaton G enters the state d_s and stays there until G_d occurs meaning that cylinder G is indeed in position "d".



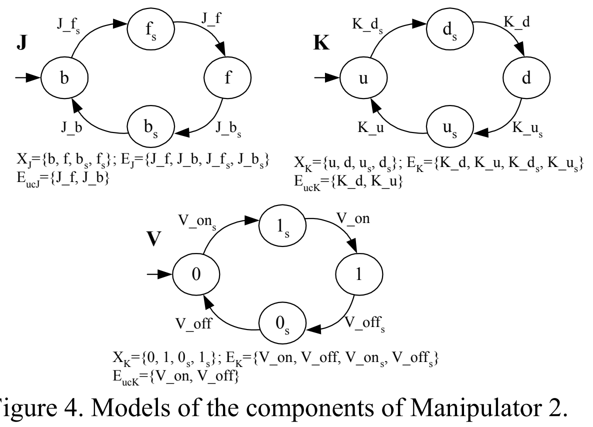
$X_G=\{u,d,u_s,d_s\}$; $E_G=\{G\_d, G\_u, G\_d_s, G\_u_s\}$
$E_{ucG}=\{G\_d, G\_u\}$

$X_H=\{0,1,1_s,0_s\}$; $E_G=\{H\_0, H\_1, H\_0_s, H\_1_s\}$
$E_{ucH}=\{H\_0, H\_1\}$

$X_I=\{r,g\}$; $E_I=\{I\_g_s, I\_r_s\}$; $E_{ucI}=\phi$

Figure 3. Models of the components of Manipulator 1.

The task of Mn2 is to pick up and place the cover onto the base after the shaft is inserted into the bearing. This manipulator consists of a vertical cylinder, a horizontal cylinder and a vacuum, which is used for holding the cover. All of these components are driven by solenoid valves. We will use the letters J, K, and V to denote horizontal cylinder, vertical cylinder, and vacuum respectively. The models for the components of Mn2 are built in a similar manner to that of Mn1 and are given in Figure 4.



$X_J=\{b, f, b_s, f_s\}$; $E_J=\{J\_f, J\_b, J\_f_s, J\_b_s\}$
$E_{ucJ}=\{J\_f, J\_b\}$

$X_K=\{u, d, u_s, d_s\}$; $E_K=\{K\_d, K\_u, K\_d_s, K\_u_s\}$
$E_{ucK}=\{K\_d, K\_u\}$

$X_K=\{0, 1, 0_s, 1_s\}$; $E_K=\{V\_on, V\_off, V\_on_s, V\_off_s\}$
$E_{ucK}=\{V\_on, V\_off\}$

Figure 4. Models of the components of Manipulator 2.

As a next step in model building we will obtain a complete model for each of the manipulators. Parallel composition of the three automata in Figure 3 gives a complete model of Mn1. Instead of making a regular parallel composition operation resulting in a 32 state automaton, we use the automaton with 24 states as a

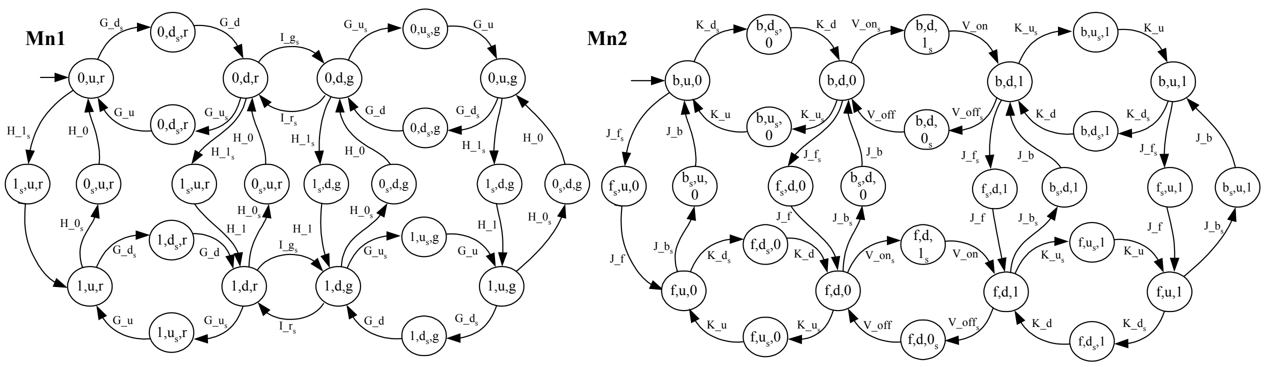


Figure 5. Automaton models of Manipulator 1 and Manipulator2.

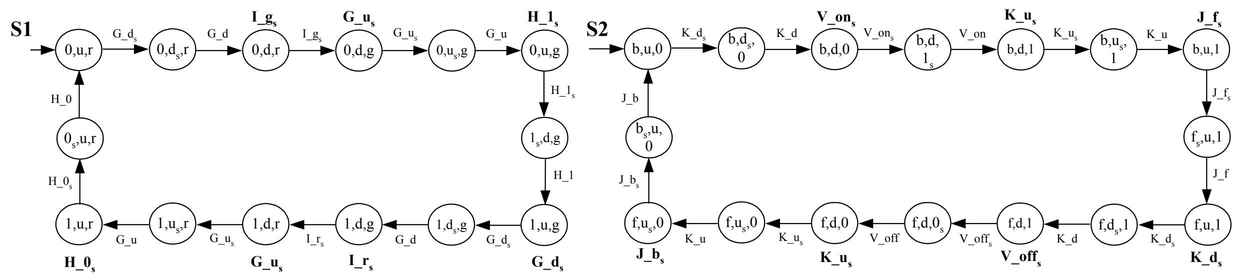


Figure 6. Supervisors for Manipulator 1 and Manipulator2.

model of Mn1 by omitting some of the states that are infeasible in the model. In a similar manner, an automaton model of Mn2 is obtained with 28 states instead of 64 states that would result from a regular parallel composition operation.

Automata Representing The Specifications

Specifications are defined as the desired behaviour, which is a subset of the system behaviour. Automata can also be used for representing the specifications. In our problem, the operation order of manipulators' components necessary for the realization of manufacturing task determines the specifications. If the output events of S1 and S2 are neglected in Figure 6, these automata represent the desired behaviour of Manipulator 1 and Manipulator 2 respectively.

IV. CONTROL SYSTEM

Structure and Design of the Control System

If a central supervisor were to be built we would have to deal with a complete model of the system, which would have 24 x 28 = 672 states. Instead of building one supervisor, we design three supervisors to have a hierarchical control structure shown in Figure 7. Two of the supervisors (S1 and S2) control their subsystems, namely Manipulator 1 and Manipulator 2, while a third supervisor (S) in the higher level of control structure is used to control these two controlled subsystems to give the overall desired behaviour. In order to analyse controllability of the overall control system, firstly we have to check the controllability of the sub control systems Sc1 and Sc2. Then we will analyse the controllability of the control system comprising the supervisor S and systems Sc1 and Sc2.

It can be shown that the language K1=$\mathcal{L}$(S1) is controllable by applying controllability condition $\overline{K1}E_{uc1} \cap M1 \subseteq \overline{K1}$, where M1=$\mathcal{L}$(Mn1) and $E_{uc1} = E_{ucG} \cup E_{ucH} \cup E_{ucI}$. The language K2=$\mathcal{L}$(S1) is also controllable with respect to M2=$\mathcal{L}$(Mn2) and $E_{uc2} = E_{ucJ} \cup E_{ucK} \cup E_{ucV}$. Therefore it is guaranteed that once Manipulator 1, which is under control of S1 is started with the event G_d$_S$, it will generate the event H_0 after following a certain string of events determined by S1. Also, when Manipulator 2, which is controlled by S2 is started with the event K_d$_S$, it will eventually generate the event J_b. This discussion leads to an automaton representation of the overall system behaviour.

Figure 8 shows the supervisor that can be used to achieve the desired behaviour. Omitting the outputs of states 0 and 2, Figure 8 also shows the automaton, say G, describing the overall system behaviour with uncontrollable events STR, H_0, and J_b.

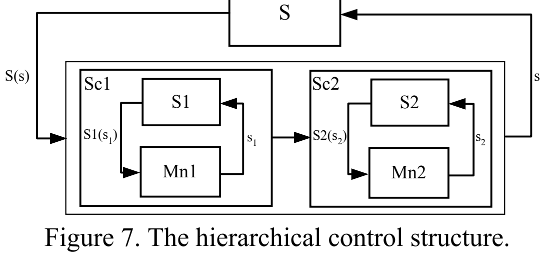


Figure 7. The hierarchical control structure.

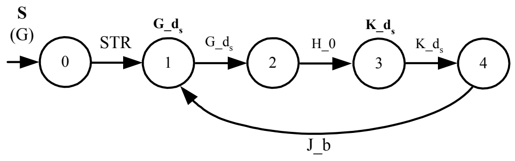


Figure 8. Automaton corresponding to the supervisor of the overall system. (Also represents the complete system).

Now we have an automaton G with 5 states only, which represents the complete pneumatic system controlled by two "sub supervisors". The controllability analysis of the complete system is reduced to the controllability analysis of the language generated by S. Since S and G are equivalent, the language corresponding to the desired behaviour of the system is the same as the language generated by the system. This means that there is no restriction on the system behaviour and that the language corresponding to the desired behaviour is controllable.

Note that, this hierarchical structure can be generalised for any number of subsystems. This structure is also convenient for additional sub systems to an existing hierarchical control system.

Realization

In the realization stage, we try to force the PLC behave as the supervisors that we designed. In order to fulfil this task, two criteria should be met. One of the criteria is to provide synchronization of the supervisors with the process and the other is generation of the output events appearing on some of the states of the automaton representations of the supervisors. As mentioned before, although these output events are generated depending upon the states of the supervisor automata, we will treat them as the controllable events generated by the system and allowed by the supervisors to occur. This interpretation allows us to analyse and design real world DES control problems without violating the assumptions in SCT.

We will explain the realization methodology on a part of the supervisor shown in Figure 8 using Ladder Diagram (LD) programming technique. We will use memory bits to represent states and events. Falling and rising edges of sensor signals will be used to meet the assumption of momentary events in DES theory. Also we will use two memory bits for each of the state, one for current PLC scan cycle, and the other for the next cycle. This solves the problem named as "avalanche effect" defined in [5].

As an example, the part of PLC program shown in Figure 9a, corresponds to transition to state 0 of automaton in Figure 8. We use q_i for the current cycle and Q_i for the next cycle of state i. The output events of G_d_s and K_d_s are realised by setting PLC's digital outputs Q0.7 and Q1.4 to logic 1 level, respectively. The program part shown in Figure 9b is used for generating these events. Note that, reset of the PLC outputs are also considered as events, and should be taken into account when writing PLC program parts for event generation.

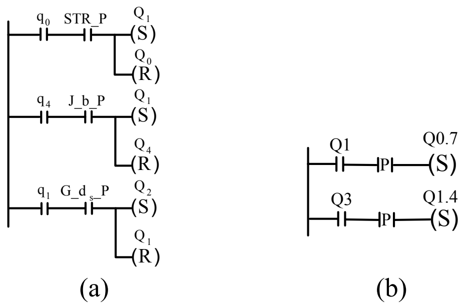


Figure 9. PLC program parts realising transition of state 1 of the supervisor (a), and output events (commands) (b).

The signal labels ending with letter P in Figure 9a indicates rising and falling edges of the signals obtained in preceding rows of the PLC program.

V. CONCLUSION

In this paper, we have presented an application of supervisory control theory to a pneumatic manufacturing system. We have proposed a hierarchical control structure, which makes it possible to design and analyse supervisors simultaneously without causing the problem of state space explosion. Also an interpretation of supervisory control theory, which enables handling real world DES control problems without violating assumptions of the theory has been introduced in this study. Finally, a realization methodology of SCT using PLC has been given.

REFERENCES

1. P. J. Ramadge, W. M. Wonham, Supervisory control of a class of discrete event processes, SIAM Journal of Control and Optimization, Vol. 25, No 1,206-230, 1987
2. S. Balemi, G. J. Hoffman, P. Gyugyi, H. Wong-Toi, and G.F. Franklin, Supervisory control of a rapid thermal multiprocessor, IEEE Transactions on Automatic Control, Vol. 38, No. 7, 1040-1059, 1993
3. B. A. Brandin, The real-time supervisory control of an experimental manufacturing cell, IEEE Transactions on Robotics and Automation, Vol. 12, No. 1,1-14, 1996
4. R. J. Leduc, PLC implementation of a DES supervisor for a manufacturing testbed: An implementation Perspective. Master's thesis, Department of Computer and Electrical Engineering, University of Toronto, Toronto, Canada, 1996
5. M. Fabian, A. Hellgren, PLC-based implementation of supervisory control for discrete event systems, Proceedings on the 37th IEEE conference on Decision & Control, Tampa, Florida, USA, 1998.
6. M. H de Queiroz, J. E. R. Cury, Synthesis and implementation of local modular supervisory control for a manufacturing cell, Proceedings of WODES 2002, 2002
7. C. G. Cassandras, S. Lafortune, Introduction to discrete event systems, Kluver Academic Publishers, 1999