Assembly Workstation Supervisory Control Technique

Daniela Cristina CERNEGA and Viorel MÎNZU
"Dunãrea de Jos" University Galati, Romania
Department of Automatic Control and Electronics
Str. Domneasca, No. 111, 6200 Galaþi, ROMANIA

Abstract

This paper proposes a control problem statement in the framework of supervisory control technique for the assembly workstations, which obviously are discrete event systems. A desired behaviour of an assembly workstation is analyzed. The behaviour of such a workstation is cyclic and some linguistic properties are established. A necessary condition for the existence of a supervisor that meets the production constraints is proved.

Key-words: - Assembly Systems, Discrete Event Systems, Supervisory Control in Discrete Event Systems

1. Introduction

This paper deals with a control problem for an assembly workstation using the *supervisory control* technique proposed by Wonham et al (see [1], [2] and [3]).

To control an assembly system means to execute a preplanned assembly process, taking into account the mutual exclusion, the concurrence of tasks and the cyclical usage of resources.

The formal linguistic properties of the assembly Petri net have been studied by Suzuki, and Okuma (see [4],[5]). Based on these properties, the supervisor for an assembly Petri net is considered as a finite automaton. The assembly process is regarded with a high level of generality. The assembly Petri net used has a tree structure, because it models the flow of parts and subassemblies and it doesn't take into account the resources involved in the assembly process.

A supervisor is an automaton that is connected with the controlled discrete event system. Therefore, a closed loop system is formed. A desired behaviour of the discrete event system may be expressed by sequences of events, which form a language. Generally, a supervisor that assures the desired behaviour exists, if two conditions are met (see the supervisor existence theorem presented in [1]). The first one is the controllability of the language mentioned before. In this paper emphasize is placed on the second condition (presented in section 3), in the case of an assembly workstation. Formal properties of a language that models the desired behavior of the controlled system are stated.

In section 2, a model for an assembly workstation is proposed. It is expressed by a controlled Petri net. This is obtained from an assembly graph (see [6] and [7]) considering the sharing of resources between assembly tasks. Therefore, repetitive aspects are introduced. In the framework of supervisory control technique, the control problem for an assembly workstation is stated in section 3. A necessary condition for the existence of the supervisor is found in section 4. This result is used in section 5, where two examples of desired behavior are given.

2. Model for assembly workstation

An assembly system is a manufacturing system which makes a product or a family of products. An assembly system is composed by workstations, each one making one or several tasks.

To execute the preplanned assembly processes, a control problem must be solved talking into account the mutual exclusion of some tasks, their concurrent execution, and so on. In order to solve a control problem, the system must be modeled in accordance with all the aspects: parts, robots, fixtures,…

In this section, a model of workstation to be controlled is proposed. In [6] the authors have proposed the assembly graph as a model of the assembly process, in the case of single product assembly systems. A model of an assembly workstation using the assembly graph was described in [7]. The advantage of using the assembly graph consists in the correct definition of tasks (both the base and the secondary parts are defined) and there is a systematic method for its construction.

For example, in fig. 1 is illustrated the assembly graph corresponding to an assembly workstation.

Nodes represent tasks and arrows represent the precedence relation between them.

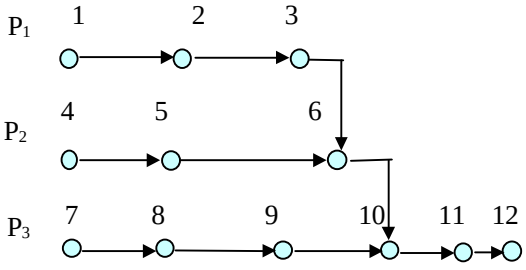

Figure 1. The assembly graph

The tasks are performed by two robots R_1 and R_2 using three fixtures P_1, P_2 and P_3. The "branches" of the assembly graph corresponds to the tasks that use the same fixture. For example, the tasks 4, 5, 6 use the fixture P2. In such a workstation, the activities begin with loading the parts on the fixtures (the tasks 1, 4 and 7). The tasks 6 and 10 put together a base part with a sub-assembly. A sequence of tasks is associated to each of the five resources (three fixtures and two robots). The task sequences are:

P_1: 1 2 3 6

P_2: 4 5 6 10

P_3: 7 8 9 10 11

R_1: 1 5 2 3 6 10 11

R_2: 4 7 8 9 10 11

Generally, the task sequences are chosen in order to eliminate any deadlock of the system.

A resource is delivered to its first task immediately after the completion of the last task of its sequence. Hence, the resources have a cyclic behavior that is shown in fig.2.

As a discrete event system, the assembly workstation is better modeled like a Petri net, because the prerequisites for the execution of the task are more explicit. We remind here after the definition of a Petri net N:

$N = (P, T, M_0)$

P: the set of all places

T: the set of all transitions

M_0: the initial marking.

The Petri net corresponding to the workstation model presented in fig. 2is shown in fig. 3.

In this Petri net a transition models a task and a mark in a place has the meaning of a resource (part, fixture or robot) available to begin a task. Because the working of the system is cyclic, the number of reachable marking (that is reachable state) is finite.

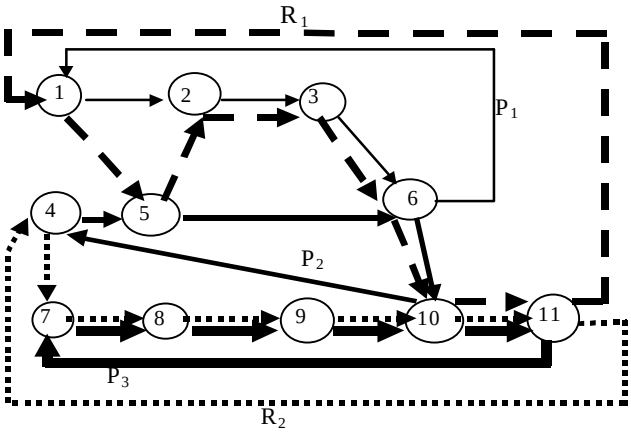

Figure 2 Task sequences for resources

3. Supervisory control problem

For the workstation model presented before the reachability graph of the Petri net is regarded as an automaton R, in order to study its formal linguistic properties:

$R = (Q_r, \Sigma, \delta_r, q_{r0}, Q_m)$,

where :

Q_r: the set of all reachable markings,

Σ : the set of symbols representing events,

δ_r: $\Sigma^* \times Q_r \rightarrow Q_r$ is the state transition function,

Σ^*: the set of all finite strings of elements of Σ,

q_{r0} : the initial state corresponding to the initial marking

Q_m: the set of marker states.

The set of events Σ is given by:

$\Sigma = T \cup T_{para}$

where T_{para} denotes a group of transitions which are fired concurrently. In R there are strings μ which satisfy $\delta_r(\mu, q_r) = q_r$.

Two languages over Σ are defined as follows:

$L(N) = \{\sigma \in \Sigma^* | \ \delta_r(\sigma, q_{r0}) \text{ is defined } \}$

$L_m(N) = \{\sigma \in L(N) | \ \delta_r(\sigma, q_{r0}) = Q_m \ \}$

L(N) is the set of all the firable strings in N and $L_m(N)$ is the set of all the firable strings, which reach the desired marking.

The controlled Assembly Petri net N_c is the original Petri net N extended with external control places:

$N_c = (P \cup P_c, T, I_c, O_c, M_{c0})$

where

P_c the set of the external control places

$I_c: T \times (P \cup P_c) \rightarrow \{0,1\}$

$O_c: T \times (P \cup P_c) \rightarrow \{0,1\}$

M_{c0} initial marking.

Consequently, two subsets can be defined in T: one is a subset whose elements are controllable transitions; the other is a subset in which every element is an uncontrollable transition. That is:

$T_c = \{ t \in T | \exists i, I_c(t, p_{ci}) = 1 \}$

$T_u = T - T_c$

As a consequence, the events are *controllable* and *uncontrollable*: $\Sigma = \Sigma_u \cup \Sigma_c$, where

$\Sigma_c = T_c \cup T_{cpara}$

$\Sigma_u = \Sigma - \Sigma_c$.

A control input for N is a set $\gamma \subset \Sigma$, with $\Sigma_u \subseteq \gamma$.

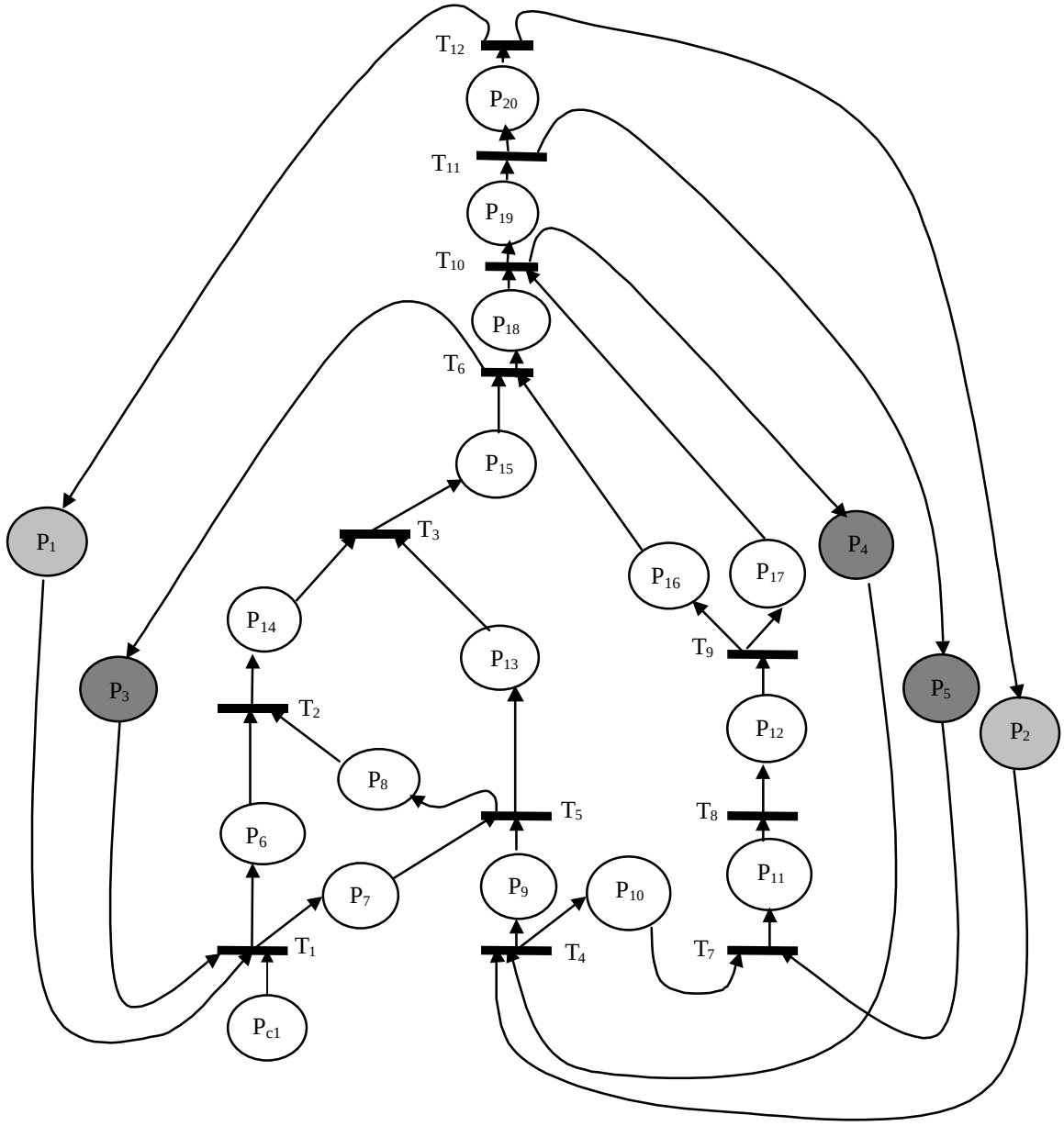

Figure 3. The controlled Petri net modelling the assembly workstation

Formally, a supervisor is a map

f: $L(N) \rightarrow \Gamma$,

where Γ is the set of control inputs.

The map f specifies, for each possible string of events σ, the *control input* $f(\sigma)$, which may be applied at that point.

The desired behavior of the closed system may be given as a language $K \subseteq L_m(N)$, or specifying indirectly this language by a set of constraints.

A state realization for a supervisor f is a pair (S,Φ), where:

$S=(Q_s, \Sigma, \delta_s, q_{s0})$ is an automaton which recognizes the language K, and

Φ: $Q_s \rightarrow \Gamma$ is a mapping that gives the control input γ for the current state of S.

That is $\gamma = \Phi(q_s) \in \Gamma$.

The language corresponding to the desired behavior is controllable if:

$$\overline{K}\Sigma_u \cap L \subseteq \overline{K} \qquad (1)$$

For the supervision control problem of a given assembly workstation, there are considered as known the following elements:

i) L(N): the *closed* behavior of N, that is prefix closed, i. e. $L(N) = \overline{L(N)}$ (where $\overline{L}$ is the set of all strings which are prefixes of words from L);

ii) Q_m: the set of *marker* states;

iii) $L_m(N)$: the *marked* behavior of N (with respect to Q_m);

In addition, the system has the nonblocking property: $\overline{L_m(N)} = L(N)$.

The aim of supervisory control is not to modify L_m, but to achieve a prescribed language $K \subseteq L_m(N)$ (who preserves the nonblocking property) for the system equipped with the supervisor f (called *closed* system).

Theorem 1 (Wonham)

For a given a nonblocking system N with *closed* behavior L(N) and *marked* behavior $L_m(N)$, and a nonempty $K \subseteq L_m(N)$, there exists a supervisor f such that $L_m(N,f)=K$ and the closed system is nonblocking **iff** K is controllable and $\overline{K} \cap L_m = K$ (that is K is L_m closed).

(for the proof see [1])

Remark: The desired behavior of the *system* equipped with a supervisor is actually chosen as a controllable language (1) (with the desired behavior constrains for the closed system). It remains to prove the L_m closure of K and the nonblocking property of the closed system. In the next section, it will be given a necessary and sufficient condition for the L_m closure of the language K (controllable) in order to guarantee the supervisor existence in a system with cyclic working.

4.Supervisor existence condition for an assembly workstation

This section deals with the existence of the supervisor for an assembly workstation modeled as in section 2. In this case, one may consider that there is only one marker state q_m. This state corresponds to the situation when the robots and the fixtures are available for their initial tasks. Without loss of generality, the marker state q_m will be considered as initial state of controlled Petri net representing the assembly workstation. Hence, the *marked language* consists of all strings of events to determine the transition from q_m to q_m.

Definition 1 : A *cyclic sequence* is a string of transitions σ, having the property

$\delta_r(\sigma,q_m)=q_m$.

Definition 2 : A cyclic sequence is called *minimal* if any transition appears at most once.

Lemma 1: A sequence σ which satisfies the equality $\delta_r(\sigma,q_m)=q_m$ such that any intermediary state is different from q_m is a minimal cyclic sequence.

The proof is based upon the fact that any transition is firable at most one time between two passages through the marker state.

Definition 3: A *language* $K \subseteq L$ is *cyclic* when each element is a juxtaposition of cyclic sequences.

Lemma 2: Any language $K \subseteq L_m$ is cyclic.

Proof: For any $\sigma \in K$, it holds $\sigma \in L_m$. Hence, the Petri net passes through the marker state m times ($m \geq 1$). Applying lemma 1, it holds

$$\sigma = s_1 s_2 \ldots s_m, \qquad (2)$$

where s_i, $i = \overline{1,n}$ are minimal cyclic sequences. Hence, K is cyclic.

Definition 4 : A cyclic language $K \subseteq L$ is *cyclic prefix closed* when for any $\sigma \in K$, any prefix of σ which is a cyclic sequence belongs to K.

A necessary and sufficient condition for the L_m closure of K is given here after.

Theorem 2

$\forall K \subseteq L_m$, K is L_m closed **iff** K is cyclic prefix closed.

Proof

($\Rightarrow$) It will be proved that if K is L_m closed ($\overline{K} \cap L_m = K$) then K is cyclic prefix closed.

Let $\forall \sigma \in K$. Because K is L_m closed, $\sigma \in L_m(N)$. So, σ has the form (2). It was considered the case when the marker states are reached several times. Otherwise, we are in the trivial case when σ is a minimal cyclic sequence. So, K is cyclic. Any prefix of σ which is a cyclic sequence has the form $s_1 s_2 \ldots s_i$. It's obvious that $s_1 s_2 \ldots s_i \in L_m(N)$ and $s_1 s_2 \ldots s_i \in \overline{K}$, $\forall i = \overline{1,m}$. Hence, $s_1 s_2 \ldots s_i \in \overline{K} \cap L_m(N)$, i.e. $s_1 s_2 \ldots s_i \in K$. One can conclude that K is cyclic prefix closed.

($\Leftarrow$) It will be proved that if K is cyclic prefix closed, then:

$$\overline{K} \cap L_m = K$$

Assuming $\overline{K} \cap L_m(N) \neq K$, this involves that there exists a string σ such that:

$$\sigma \in \overline{K},\ \sigma \in L_m(N) \text{ and } \sigma \notin K.$$

Because $\sigma \in L_m(N)$, it has the form (2).

Because $\sigma \in \overline{K}$, there exists a string μ such that $\sigma\mu \in K$. But $K \subseteq L_m$. That means $\sigma\mu = s_1 s_2 \ldots s_m \alpha_{m+1} \ldots \alpha_n$, where α_i are minimal cyclic sequences i=m+1,…,n.

From $\sigma\, \alpha_{m+1} \ldots \alpha_n \in K$ and $\sigma \notin K$, one can deduce that K is not cyclic prefix closed. That contradicts the initial supposition. So, it holds:

$$\overline{K} \cap L_m = K \quad \text{q.e.d.}$$

Example

Let the assembly workstation considered in section 1. The *marker* state q_m is given by a marking whose places P_1, P_2, P_3, P_4,P_5 have one token each, and all the other places have no token $M(P_i) = 0$, i = 6,20:

$q_m = (1, 1, 1, 1, 1,0, 0, \ldots,0)$.

Obviously, the controlled Petri net, which models the assembly workstation, has the nonblocking property because it has no deadlock. A desired behavior of the closed loop may be described by the following constraints:

1) Prohibit the concurrent execution of T_1 and T_4. When T_1 and T_4 are concurrently firable, give a priority to T_4.
2) Prohibit the concurrent execution of T_1 and T_7. When T_1 and T_7 are concurrently firable, give a priority to T_7.
3) Prohibit the concurrent execution of T_1 and T_8. When T_1 and T_8 are concurrently firable, give a priority to T_8.
4) For a single assembly product, the workstation works according to the sequence s_1; for two products according to the sequence $s_1 s_2$; for more than 2 products, the system uses the sequence $(s_1)^n$, $n \geq 3$, where s_1 and s_2 are given here after:

   $s_1 = T_4T_7T_8T_1T_9T_5T_2T_3T_6T_{10}T_{11}T_{12}$

   $s_2 = T_4T_7T_8T_9T_1T_5T_2T_3T_6T_{10}T_{11}T_{12}$.

So, the language K_1 of the supervised system (N, f) can be expressed as the regular expression:

$$K_1 = s_1 + s_1 s_2 + s_1^2 s_1^*.$$

In order to guarantee the supervisor existence (see Theorem 1), the language K_1 has to be controllable and L_m closed. The controllability of K_1 is ensured with a control place before T_1, in order to control the fire of this transition.

It is obvious that K_1 is not cyclic prefix closed, because the sequence $s_2 s_1$ doesn't have the prefix s_2 in the language K_1. Under these circumstances, there is no supervisor to meet the given constraints.

Another closed loop behavior K_2 for the same system can be specified with the following constraints:

1) Prohibit the concurrent execution of T_1 and T_4. When T_1 and T_4 are concurrently firable, give a priority to T_4.
2) Prohibit the concurrent execution of T_1 and T_7. When T_1 and T_7 are concurrently firable, give a priority to T_7.
3) Prohibit the concurrent execution of T_1 and T_8. When T_1 and T_8 are concurrently firable, give a priority to T_8.

For this control objective, the control language K_2 to meet the constraints 1), 2), 3) can be described as the regular language:

$K_2 = (s_1 + s_2 + s_3 + s_4 + s_5)^*$, where s_1 and s_2 are the same as in the precedent example, and

$s_3 = T_4T_7T_8T_1T_5T_9T_2T_3T_6T_{10}T_{11}T_{12}$

$s_4 = T_4T_7T_8T_1T_5T_2T_9T_3T_6T_{10}T_{11}T_{12}$

$s_5 = T_4T_7T_8T_1T_5T_2T_3T_9T_6T_{10}T_{11}T_{12}$

For this language, Theorem 2 guarantees the existence of the supervisor because K_2 is cyclic prefix closed and controllable. The number of states of the automaton N has 28 states. The automaton S to recognize the language K_2 to meet the constraints for the *closed loop system* has 18 states.

5. Conclusion

In this paper, a control problem for assembly workstations was presented. In order to state the problem, a model for an assembly workstation was proposed. It can be obtained in a systematic way, using the assembly graph and introducing the repetitive aspects due to the passage from a product to another.

In the framework of the supervisory control technique, the control problem was stated. A necessary and sufficient condition that assures the L_m closure of the desired behaviour K was proved.

To use this result, one must verify that the chosen language K, which meets the production constraints, is cyclic prefix closed. If, in addition, K is controllable, then the supervisor exists.

6. *References*

[1] P. J. Ramadge, W. M. Wonham, *Supervisory Control of A Class of Discrete Event Processes*, SIAM J. Control & Optimization, Vol.25, No.1, pp.206-230, 1987

[2] W. M. Wonham, P. J. Ramadge, *On the Supremal Controllable Language of a Given Language*, SIAM J. Control & Optimization, Vol.25, No.3, pp.637-659, 1987

[3] P. J. Ramadge, W. M. Wonham, *Modular Feedback Logic for Discrete Event Systems*, SIAM J. Control & Optimization, Vol.25, No.5, pp.1202 - 1218, 1987

[4] T. Suzuki, T. Kanehara, A. Inaba, S. Okuma, *On Algebraic and Graph Structural Properties of Assembly Petri Net*, IEEE Proceedings International Conference on Robotics and Automatics, pp 507 – 514, 1993

[5] T. Suzuki, S. Okuma, *Supervisory Control of Assembly Petri Net*, IEEE Proceedings International Conference on Robotics and Automatics, pp 794 – 800, 1994

[6] V. Mînzu, J.M Henrioud; *Systematic Method for the Design of Flexible Assembly Systems*, IEEE International Conference on Robotics and Automation, Atlanta-USA, may 2-6, 1993.

[7] V. Mînzu, J.M. Henrioud; *Appropche systématique de structuration en postes des systèmes d'assemblage monoproduits*; Journal Européen des Systèmes Automatisés, Vol.31, No.1/1997, p57-78; HERMES ISSN 0296-1598.