Preprints of the 8th IFAC/IFORS/IMACS/IFIP Symposium on Large Scale Systems, Rio Patras, Greece, Vol. 2, pp. 989-994, 1998.

PETRI NET BASED SUPERVISORY CONTROL OF FLEXIBLE BATCH PLANTS

G. Mušič and D. Matko

Faculty of Electrical Engineering, University of Ljubljana, Slovenia.
E-mail: gasper.music@fe.uni-lj.si

Abstract: The application of supervisory control concepts in the control of batch process plants is investigated in the paper. Supervisory control issues in flexible batch plants include co-ordination, resource allocation and process management. The paper utilises the Petri net based supervisory control framework to develop appropriate solutions. A straightforward approach of translating developed Petri net models to industrial implementation by sequential function charts is presented. The described concepts are illustrated by a simple resource allocation problem in a batch process cell.

Keywords: batch processing, Petri nets, supervisory control

1. INTRODUCTION

Flexible batch plants are gaining importance in process industry. In comparison to classical process plants where main stress is placed on continuous control, the batch plants involve many discrete control functions. Because of the complexity of such plants the local level control is generally inadequate. Addition of supervisory control functions is required in order to maintain the desired system performance. These functions introduce several classical discrete manufacturing system design issues into the design and analysis of the batch production plants.

Basic functions of the supervisory level include coordination of the low level control activities, scheduling of subprocess operations, various emergency scenarios, start-up and shutdown sequences. The implementation of such functions can significantly increase reliability, availability and flexibility of production processes.

Most of the current research in the area of supervisory control is focused on the idea of synthesis of the supervisor for a given discrete event model of the plant. The plant usually contains several control units (programmable logic controllers or closed loop controllers) which are already incorporated in the plant model. The aim of such a supervisor is to force the plant model to have desired properties, which correspond to additional system specifications.

The theory of supervisory control was pioneered by the work of Ramadge and Wonham (1989). In their work the term supervisor is used in the sense of a discrete event controller, which does not uniquely define the next control input but merely defines a subset of allowable control inputs. A typical control goal is to make sure that the plant never reaches some specific states.

The systematic procedure of deriving an appropriate process supervisor starts with deriving a model of the plant, continues with model validation and synthesis of a supervisor that fulfils the given set of specifications. In most cases, a formal verification of the obtained solution has to be performed before the supervisor is actually implemented.

The paper concentrates on Petri net based supervisory control synthesis for the purpose of resource allocation in flexible batch plants. Formal synthesis methods are applied, which ensure that the derived system maintains certain properties. A straightforward approach to the industrial implementation of the developed solutions is suggested by conversion of the derived Petri net model of the supervisor into the sequential function chart (SFC) representation. The ap-

proach is illustrated by an example of a batch process cell. The paper is structured as follows. The modelling of the plant and the supervisor by Petri nets and SFCs is discussed in Section 2. The supervisory synthesis method based on place invariants is briefly reviewed and illustrated by a simple example in Section 3. The implementation issues are discussed in Section 4.

2. PETRI NETS AND SFC

A supervisory system is predominantly discrete regardless of the type of the process being controlled. Its design requires a discrete event view of the system. Various discrete event modelling techniques can be used to obtain the model of a plant and a supervisor. No general agreement has been achieved yet upon a modelling framework that would best suit the needs of analysis and design of supervisory systems, nevertheless, discrete transition system descriptions such as finite automata (Hopcroft and Ullman, 1979) or Petri nets (Murata, 1989) are used most commonly.

The finite automata representation of the supervisors and plants is used in most of the early works on supervisory control. Applied synthesis methods are based on searching over the state space of the automata. One of the main problems of the application of the developed supervisory framework to real industrial processes is the state explosion problem. Petri nets, on the other hand, enable the modeller to include additional structural information into the model. The state of the system is distributed over the places of the net. The Petri net based synthesis methods tend to exploit the net structure thus reducing the need to search over the whole state space. Some recent contributions on the Petri net based supervisory control can be found in (Giua and DiCesare, 1992; Krogh and Holloway, 1991; Moody and Antsaklis, 1996; Yamalidou et. al., 1996; Zhou et. al., 1992). The main problem of the application of the developed methods is that they are generally limited to a particular type of Petri net models and allow only certain types of system specifications. Petri net models, however, are relatively easily understood by non-experts and yield the possibility of a more intuitive design based on the knowledge about the process and not purely on the rigid theoretical background. This can be helpful in cases that extend beyond the limitations of a particular synthesis technique.

Perhaps the most significant advantage of the Petri net representation is the straightforward path from the developed models to the industrial implementation. The discrete control logic is most often implemented by programmable logic controllers. The recent IEC standard on programming languages of industrial logic controllers (IEC, 1992) promotes the use of Sequential Function Chart (SFC) representation of the control logic. SFC (also referred as Grafcet) inherited many of its features from the theory of Petri nets. More precisely, a safe interpreted Petri net can be defined such that its input-output behaviour is the same as the input-output behaviour of the SFC (David and Alla, 1994; David, 1995).

A place in such a Petri net corresponds to a step in the SFC. Transitions and directed links have the same meaning in SFC as they have in Petri net. If an input/output interpretation is added to the transitions and places of the Petri net, we can obtain an equivalent SFC model. There are however two basic differences between an interpreted Petri net and a SFC. The first difference between Petri nets and SFCs is that the marking of a SFC is Boolean (step is active or inactive) while the marking of a place in a Petri net can be any positive integer. For that reason the conversion of a Petri net to a SFC is only possible when the net is safe (i.e., for any reachable marking, the marking of every place is less than or equal to one).

The second difference between Petri nets and SFCs is that the firing rule of a SFC is different from a Petri net when there is a conflict. A conflict in a Petri net describes the situation when some transitions are enabled by the marking of the same place. This leads to a non-deterministic behaviour since there is no rule to choose which of them will be fired. When such situation emerges in a SFC the transitions are fired according to their priority to ensure the deterministic behaviour or, when no priorities are defined, all transitions fire simultaneously. Now, if a Petri net is such that any pair of transitions in conflict has receptivities, which can not be true at the same time, the behaviour of the net is deterministic. If such a Petri net is also safe, it is equivalent to a SFC (David, 1995).

Its strong relation to Petri net theory enables a SFC to be directly redrawn from a Petri net model and the classical properties of Petri nets, such as marking invariants, can be applied also to SFCs. In this way, the supervisor can be derived within the Petri net supervisory control framework and the derived model can then be easily transformed into a logic controller program.

3. SUPERVISORY CONTROL SYNTHESIS

One way of including supervisory mechanisms is the mutual exclusion concept introduced in (Zhou and DiCesare, 1991; Zhou, 1992). Two concepts, parallel and sequential mutual exclusions are defined and used to synthesise bounded, live and reversible Petri net. In this classical mutual exclusion concept, all transitions are assumed to be controllable, i.e., may be prevented from firing by a supervisor. This assumption however, is rather unrealistic. Therefore in the traditional supervisory control framework (Ramadge and Wonham, 1989) the complexity of enforcing desired properties is enhanced by the presence of uncontrollable transitions. Different approaches based on Petri net

models and which also consider the problem of uncontrollable transitions are given in (Giua, 1992; Krogh, 1991; Moody, 1996; Yamalidou, 1996). The last approach (Moody, 1996; Yamalidou, 1996) is based on place invariants and is particularly interesting, because the resulting supervisory mechanism is computed very easily. It is briefly summarised here and illustrated by a batch process cell example.

By the method of place invariants it is possible to enforce a set of n_c constraints on the plant state $\mathbf{m}_p$. The plant state is represented by a $m \times 1$ marking vector of non-negative integers, where each vector component is equal to the marking of the corresponding place in the Petri net model of the plant. Constraints are written in the form

$$\mathbf{L}\mathbf{m}_p \leq \mathbf{b} \tag{1}$$

where $\mathbf{L}$ is a $n_c \times m$ integer matrix and $\mathbf{b}$ a $n_c \times 1$ integer vector (Yamalidou, 1996). The inequality (1) is read with respect to each element on the corresponding left and right hand sides of the inequality. It is shown in (Yamalidou, 1996) that if the initial marking does not violate the given set of constraints, (1) can be enforced by a supervisor with the incidence matrix

$$\mathbf{D}_c = -\mathbf{L}\mathbf{D}_p \tag{2}$$

where $\mathbf{D}_p$ is the $m \times n$ incidence matrix of the plant. The initial marking of the controller is computed by

$$\mathbf{m}_{c_0} = \mathbf{b} - \mathbf{L}\mathbf{m}_{p_0} \tag{3}$$

where $\mathbf{m}_{p_0}$ is the $m \times 1$ initial plant marking vector of non-negative integers. The supervisor consists of n_c places that are linked to the existing transitions of the plant. With the addition of supervisor places the overall system is given by

$$\mathbf{D}_s = \begin{bmatrix} \mathbf{D}_p \\ \mathbf{D}_c \end{bmatrix} \qquad \mathbf{m}_s = \begin{bmatrix} \mathbf{m}_p \\ \mathbf{m}_c \end{bmatrix} \tag{4}$$

and every single constraint is transformed to a marking invariant that corresponds to a place invariant (David, 1994) of the supervised system.

Generally, some transitions are always found uncontrollable. These are, e.g., all transitions that represent sensor readings as well as come control actions that must not be prevented due to required process operation or safety reasons.

Let $\mathbf{D}_{uc}$ represent the columns in the process incidence matrix that correspond to uncontrollable transitions. Clearly, the firing of an uncontrollable transition must not depend on the marking of any place that belongs to the supervisor. The supervisor matrix $\mathbf{D}_c$ must therefore contain no negative elements in the columns that correspond to uncontrollable transitions (a supervisor designed by the described method contains no self loops, so this condition is sufficient). This is true when the matrix $\mathbf{L}\mathbf{D}_{uc}$ contains no positive elements as these will appear as negative elements in $\mathbf{D}_c$ calculated by (2). The set of constraints must therefore satisfy

$$\mathbf{L}\mathbf{D}_{uc} \leq 0 \tag{5}$$

If this is not the case, matrix $\mathbf{L}$ (and eventually vector $\mathbf{b}$) must be transformed so that (5) will be satisfied while the supervisor designed to fulfil the new set of constraints will also maintain the original set of constraints. An algorithm that performs an appropriate transformation of $\mathbf{L}$ and $\mathbf{b}$ is given in (Moody, 1996).

The described concept can be effectively used to introduce resource allocation strategies and co-ordination mechanisms in the areas such as batch system control.

3.1 *Batch process cell example*

Consider a simple example from the area of batch systems. Part of a batch process cell is shown in Fig. 1. Two mixing tanks share the same supply tank. Mixing tanks are repeatedly filled and discharged with the restriction that only one tank can be filled at a time.

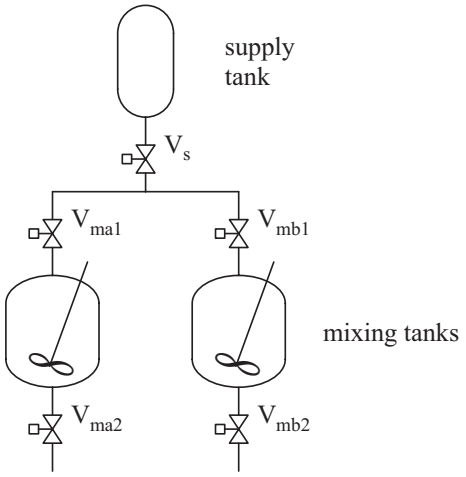

Fig. 1. Part of a batch process cell

First the Petri net models of the two individual process lines are derived. A process line is defined by the ISA SP88 standard as the set of equipment used to produce one batch. Lines can be configured to combine the equipment differently for different products or batches. In the given case, the two lines consist of a supply tank and a mixing tank each. The two models are identical and a corresponding Petri net is depicted in Fig. 2a. The models are combined by merging the places that correspond to the same status or operation. In the example these are the places p_{a3} and p_{b3} that both correspond to the outlet valve of the supply tank. Letter 'a' or 'b' that is introduced into place/transition subscript denotes to which tank the corresponding place/transition belongs. Fig. 2b shows the Petri net, obtained by merging the two places.

The classical bottom-up approach to Petri net modelling provides special rules, which define places that are allowed to be merged. This ensures that important properties of the subnets are preserved in the final net. These rules were not taken into account in the previous example and such merging does not guarantee that any of the important properties is preserved. However, this

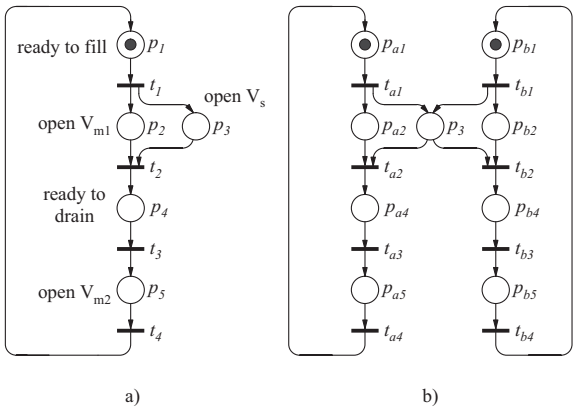

Fig. 2. Petri net model

concept enables much greater flexibility in modelling individual units and better suits the distributed system architecture. The problems that arise can be overcome by the synthesis of a supervisor that prevents any undesired behaviour of the plant. Such a supervisor corresponds to a co-ordination level required when merging several locally controlled subprocesses.

Consider the above example with the current marking of Petri net as shown in Fig. 2b. Obviously, if transitions t_{a1} and t_{b1} fire, the place p_3 contains two tokens and is therefore not safe. The safeness of the place p_3 is required, because it represents the operation of opening the outgoing valve of the supply tank and this can not be opened twice at the same time. The described situation is a malfunction of the system. To prevent this the supervisor has to be designed that will co-ordinate the two mixers in such a way that only one will be filled at a time. This requirement is written as

$$\mu_3 \leq 1$$

where μ_3 is the marking vector component that corresponds to the place p_3. The requirement can be easily transformed to the form (1). With the marking vector being

$$\mathbf{m}_p = [\mu_{a1}, \mu_{a2}, \mu_3, \mu_{a4}, \mu_{a5}, \mu_{b1}, \mu_{b2}, \mu_{b4}, \mu_{b5}]^T$$

we have

$$\mathbf{m}_{p_0} = [1,0,0,0,0,1,0,0,0]^T$$

$$\mathbf{L} = [0,0,1,0,0,0,0,0,0]$$

and $\mathbf{b} = 1$. The supervisor can be computed by (2) and (3). Given $\mathbf{D}_p$ as

$$\mathbf{D}_p = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

we obtain the supervisor

$$\mathbf{D}_c = \begin{bmatrix} -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 \end{bmatrix}$$

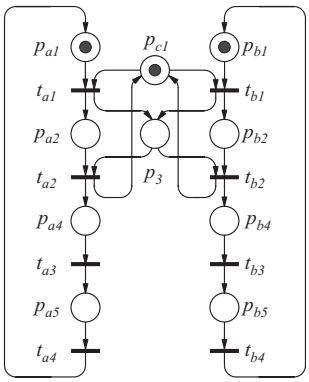

Fig. 3. Petri net model of the supervised system

$$\mathbf{m}_{c_0} = 1$$

The supervisor consists of a single place that is connected to the plant Petri net as shown in Fig. 3. We assume transitions t_{a1} and t_{b1} are controllable.

The marking invariant that is enforced by the supervisor is:

$$\mu_3 + \mu_{c1} = 1$$

It can be shown that markings of all other places of the resulting net are all included by at least one marking invariant that has a sum of tokens equal to one. Such a set of marking invariants is, for example

$$\mu_{a1} + \mu_{a2} + \mu_{a4} + \mu_{a5} = 1$$

$$\mu_{b1} + \mu_{b2} + \mu_{b4} + \mu_{b5} = 1$$

The net is therefore proven to be safe.

4. IMPLEMENTATION

To bring the simulation model closer to industrial implementation the discrete model is transformed to SFCs. The single Petri net from Fig. 3 is decomposed into several charts that correspond to separate logic controllers. We assume each process line is controlled by its own controller and supervisory part resides in another controller. The net decomposition is illustrated by Fig. 4.

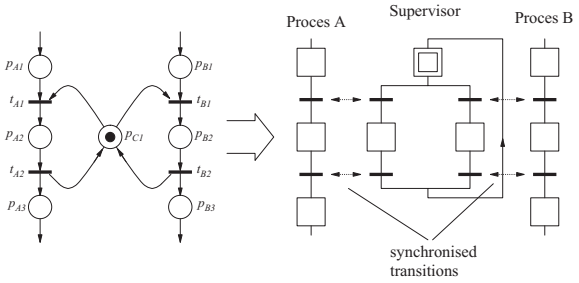

Fig. 4. Conversion of Petri net model to individual SFC modules

The interaction among different SFC modules is performed through synchronised transitions. These transitions have identical enabling conditions and furthermore, they are only enabled when all the preceding

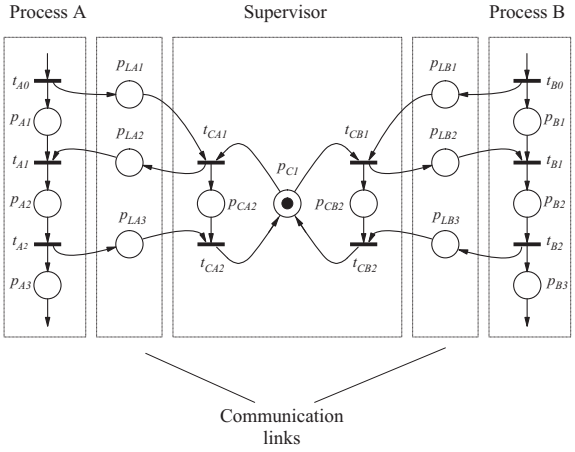

Fig. 5. Synchronisation through communication links

steps to any of the participating transitions are active. It is relatively difficult to achieve such a synchronisation among different logic controllers in practical implementation. Because of the communication delays it can not be guaranteed that transitions in different controllers fire simultaneously. This can be solved by defining the firing order of the transitions. In the cases when the two controllers share the same resource and the supervisor performs the resource allocation such as indicated in Fig. 4 the transition that books the resource must be fired in the supervisor first and only then in the local controller. In the opposite case the communication delay would allow a double booking of the shared resource by the two controllers.

A Petri net model of such a decomposed model is shown in Fig. 5. An appropriate ordering of transition firings is achieved through communication places that represent network link variables on a PLC network. Consider the situation when process A wishes to use the shared resource. The place p_{LA1} signals the supervisor that transition t_{A1} is enabled by the local controller. The supervisor books the shared resource to process A by firing transition t_{CA1}. The place p_{LA2} becomes marked and only then the transition t_{A1} can fire and process A starts utilising the resource. After the shared resource is released and transition t_{A2} fires, place p_{LA3} becomes marked which enables the transition t_{CA2}. After firing of t_{CA2} the shared resource is again available to both processes.

The controlled transitions t_{A1} and t_{B1} in Fig. 5 both only have one input place. If the set of input places of a controlled transition consists of more than one place, a communication place like p_{LA1} is associated with each input place and linked to its input transition(s). All communication places are then linked to the corresponding transition in the supervisor (t_{CA1} in Fig. 5).

To implement the described allocation scheme in programmable logic controllers the Petri net model is converted to three SFCs as indicated in Fig. 4. This is detailed in Fig. 6 with receptivities of the SFC transitions that participate in booking of a resource. The original receptivity (before the addition of the

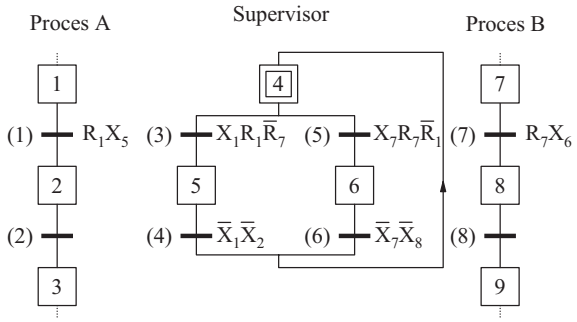

Fig. 6. Receptivities in SFC modules

supervisor) of transition (i) is denoted by R_i. We will denote the receptivity of a transition after the addition of the supervisor by R_i'. X_i is a Boolean variable that is 1 when the step i is active. In the given example the transition (3) is related to transition (1), transition (4) to transition (2), transition (5) to transition (7) and transition (6) to transition (8). The principle of operation follows the one presented above with the Petri net model. If we take the same situation as above, i.e., process A requires the shared resource in step 2, the receptivity of transition (3) assures that (3) is fired before (2), which disables (5) and consequently (7). In this way the resource is booked to process A, and (1) is allowed to fire. The variables $\overline{R}_7$ and $\overline{R}_1$ are included in R_3' and R_5' to avoid conflict in the supervisory part of the SFC. Transition (4) releases the resource and can be fired as soon as step 2 is no longer active, therefore $\overline{X}_2$ is included in R_4'. $\overline{X}_1$ is added to R_4' to prevent the release of the shared resource before X_2 is actually activated.

The described concept can be easily adapted to more general cases. If we consider the cases where a shared resource is used in several consequent steps and there is no branching between the transitions that book and release the shared resource in the process SFC, the situation is very similar to the one shown in Fig. 6. The supervisory part consists of several SFCs, one per each resource. Every supervisory SFC consists of $n+1$ steps and $2n$ transitions where n is the number of processes that share the resource. n is equal to the number of negative elements in the corresponding row of the matrix $\mathbf{D}_c$. Let t_{CIj} denote the transition in the supervisory SFC that books the resource to the process j, and t_{COj} the transition that releases the resource. The transition in the process j SFC that starts the actual use of the resource is denoted by t_{PIj} and the transition that ends the resource utilisation is denoted by t_{POj}. Let X_{PIk} denote the state of the k-th input step of the transition t_{PIj} and m_j the number of input steps to t_{PIj}. X_{Cj} is the state of the step in the supervisory SFC that books the resource and X_{Pjk} is the state of the k-th of the l steps that use the resource in process j. The receptivities of the transitions t_{CIj}, t_{PIj} and t_{COj} are then given as follows:

$$R'_{CIj} = (\prod_{\substack{k=1 \\ k\neq j}}^{n} \overline{R}_{PIk}) R_{PIj} (\prod_{k=1}^{m_j} X_{PIk}) \quad (6)$$

$$R'_{PIj} = R_{PIj} X_{Cj} \quad (7)$$

$$R'_{COj} = \overline{X}_{PI1} (\prod_{k=1}^{l} \overline{X}_{Pjk}) \quad (8)$$

Note that any of the input steps of transition t_{PIj} could be placed instead of X_{PI1} in (8) since transition can not fire until all of the input steps are active.

In this way the supervisory synthesis method based on place invariants can be directly applied to the SFC representation of control sequences.

5. CONCLUSIONS

It has been shown how the Petri net based supervisory control can be used in solving problems of resource allocation that appear in flexible batch plants. The supervisory control method based on place invariants has been illustrated and the relation between derived Petri net solutions and sequential function chart representation of control logic has been investigated. A methodology is proposed, which enables a direct translation of resulting Petri net models to decomposed SFC solution in a distributed environment. An application of the proposed method to a laboratory scale modular production system is planned for the future work.

6. REFERENCES

David, R. and Alla, H. (1994). Petri Nets for Modeling of Dynamic Systems - A Survey. *Automatica*, 30 (1994), pp. 175–202.

David, R. (1995). Grafcet: A Powerful Tool for Specification of Logic Controllers. *IEEE Trans. on Control Systems Technology*, 3 (1995), pp. 253–268.

Giua, A. and DiCesare, F. (1992). Generalized mutual exclusion constraints on nets with uncontrollable transitions. In: Proc. 1992 IEEE Int. Conf. on Systems, Man, and Cybernetics (Chicago, Illinois), pp. 974–979.

Hopcroft, J. E. and Ullman, J. D. (1979). Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, Reading, 1979.

IEC, International Electrotechnical Commission (1992). Programmable Controllers - Part 3: Programming Languages, publication 1131.3 (1992).

Krogh, B. H. and Holloway, L. E. (1991). Synthesis of feedback logic for discrete manufacturing systems. *Automatica*, 27 (1991), pp. 641–651.

Moody, J. O. and Antsaklis, P. J. (1996). Supervisory Control of Petri Nets with Uncontrollable/Unobservable Transitions. Tech. Rep. ISIS-96-004, University of Notre Dame, 1996.

Murata, T. (1989). Petri Nets: Properties, Analysis and Applications. *Proc. IEEE*, 77 (1989), pp. 541–580.

Ramadge, P. J. G. and Wonham, W. M. (1989). The Control of Discrete Event Systems. *Proc. IEEE*, 77 (1989), pp. 81–97.

Yamalidou, K., Moody, J., Lemmon, M., and Antsaklis, P. (1996). Feedback Control of Petri Nets Based on Place Invariants. *Automatica*, 32 (1996), pp. 15–28.

Zhou, M. C. and DiCesare, F. (1991). Parallel and Sequential Mutual Exclusions for Petri Net Modeling of Manufacturing Systems with Shared Resources. *IEEE Trans. on Robotics and Automation*, 7 (1991), pp. 515–527.

Zhou, M. C., DiCesare, F., and Rudolph, D. L. (1992). Design and Implementation of a Petri Net Based Supervisor for a Flexible Manufacturing System. *Automatica*, 28 (1992), pp. 1199–1208.