Turk J Elec Engin, VOL.10, NO.1 2002, © TÜBİTAK

A New Petri-Net-Based Synthesis Technique for Supervisory Control of Discrete Event Systems

Murat UZAM[*]
Niğde Üniversitesi, Mühendislik-Mimarlık Fakültesi,
Elektrik-Elektronik Mühendisliği Bölümü, 51100 Niğde, TURKEY
e-mail: murat_uzam@hotmail.com
Anthony H. JONES
Department of Aeronautical, Mechanical & Manufacturing Engineering,
University of Salford, SALFORD M5 4WT, UK
e-mail: a.h.jones@amme.salford.ac.uk

Abstract

A new Petri-net-based top-down synthesis technique for supervisory control of Discrete Event Systems (DES) is proposed to solve the forbidden state problem. The supervisors obtained are compiled supervisors, whose control policy is represented as a net structure, as opposed to mapping supervisors, whose control policy is computed as a feedback function of the marking of the system. The compiled supervisors obtained by using the technique proposed in this paper are both nonblocking and maximally permissive. The supervisors to be synthesised consist of a controlled Automation Petri Net model of the system. Automation Petri Nets (APN) include the following extensions to the ordinary Petri net framework: sensor readings as firing conditions at transitions and actions assigned to places. Ladder logic diagram (LLD) code is used to implement the supervisors on programmable logic controllers (PLC). It is important to note that the supervisors obtained are correct by construction; therefore there is no need for verification. The supervisory control synthesis technique proposed in this paper is applicable to both high-level discrete event control, where the role of the supervisor is to coordinate control of_in the discrete manufacturing sense_machines, workcells, etc., and low-level discrete event control, where the role of the supervisor is to arrange low-level interaction between control devices, such as motors and actuators. In this paper, the applicability of the proposed technique to low-level discrete event control is demonstrated by considering an experimental discrete manufacturing system.

Key Words: *Supervisory Control, Petri Nets, Discrete Event Systems (DES), Manufacturing Systems, Programmable Logic Controllers (PLC), Ladder Logic Diagrams (LLD).*

1. Introduction

The *supervisory control theory* was introduced as a conceptual framework for studying the supervision (i.e. control) of discrete event systems (DES) [1 - 4]. The key concepts in the supervisory control of DESs are as follows:

[*]Corresponding author

• There are two types of events that may occur in the DES, namely *controllable events*, that may be controlled by control action, and *uncontrollable events*, that may not be controlled by control action.

• Given a model of a DES and a specified desired behaviour of the controlled system, the objective is to synthesise a supervisor and a supervisory control policy to realise the specified controlled behaviour.

• Controlled behaviour of the DES must be *nonblocking*, i.e. it must not contradict the specifications given.

• Controlled behaviour of the DES must be *maximally permissive* within the specifications given, i.e. all events which do not contradict the specifications are allowed to happen.

The supervisory control theory is based on finite state machines (FSM) and formal language concepts. Although FSMs provide a general framework for establishing the fundamental properties of DES control problems, there are some disadvantages in using FSMs [5]. Firstly, for practical systems, the number of states used to model the system increases exponentially as the system becomes bigger. This means that FSMs are computationally inefficient. Secondly, a meaningful graphical representation is impossible for all but modest problems. Finally, one has to expend a very high initial effort to become familiar with the necessary mathematical tools.

Petri nets have gained in popularity as an alternative framework for the design of supervisory controllers for discrete event systems [5 - 12]. This is because Petri-net-based solutions have several advantages over FSMs [5]. Firstly, the states of a Petri net are represented by the possible markings and not by the places: thus Petri nets give a more compact description, i.e. the structure of the net may be maintained small even if the number of the markings grow. Secondly, instead of using ambiguous textual descriptions or mathematical notations, which are difficult to understand, the plant and the specifications can be represented graphically in an easily understood format using Petri nets. Finally, by using Petri net models, the model can be used for the analysis of their properties, performance evaluation and the systematic construction of discrete event supervisors. In the literature, there are mainly two types of Petri-net-based supervisors considered, namely *mapping supervisors*, whose control policy is a function computed after each new event generated by the system, and *compiled supervisors*, whose control policy is represented as a net structure [5]. There are several advantages in fully compiling the supervisor into a net structure [5]. Firstly, the computation of the control action is faster, since it does not require separate on-line computation. Secondly, the same Petri net system execution algorithms may be used for both the original system and the supervisor. Finally, a controlled model of the system under control can be built with standard net composition constructions. It is obvious that compiled supervisors are preferred to mapping supervisors. Unfortunately, in general the construction of such supervisors has long been done by heuristic methods. Therefore, an important issue within the Petri-net-based synthesis of supervisors for DESs has been the development of a formal methodology for the design of such supervisors.

The classes of specifications that have been considered within the supervisory control problem fall into two categories: the *forbidden state problem* [2] and the *forbidden string problem* [3]. Note that in this paper only the forbidden state problem is considered.

In the forbidden state problem, the control specifications are expressed as forbidden conditions. Forbidden conditions are a compact way of defining classes of undesirable markings which should be avoided. In a discrete manufacturing context, the forbidden state problem can be specified as undesirable operating conditions, for which the production goals cannot be satisfied, or catastrophic situations, in which data or equipment can be damaged [7]. In this case, the supervisor implements a state feedback. That is, the control input is a function of the present state of the system and the objective is to synthesise a supervisor and a feedback policy which guarantees that the system will not enter a forbidden state. Supervisory control and forbidden state problems occur at all levels of the manufacturing system control hierarchy, ranging from the low-level interaction between equipment controllers and devices through the coordination of workcells, to the factory-wide coordination of workstation controllers [7].

In the case of the forbidden state problem, an important step forward has been the introduction of so called controlled Petri nets (CtlPN) [6]. The basic restriction of this method is that the net is a marked graph, i.e. each place has exactly one input arc and one output arc. It was also assumed that there was no conflict in the net. This technique involved the computation of the control law in two steps: off-line computation and on-line computation. Both these computations are very simple. Therefore, this approach is very efficient. This technique has been extended to be applicable to a very general class of controlled Petri nets [12]. However, because the controller is given as a feedback law, it is not possible to design a net model of the controlled system. In order words, the supervisor obtained in [6 and 12] is a mapping supervisor. Other methods [7, 9-11] suffer from the same problem. In [8], a method was proposed to obtain a compiled supervisor, whose control policy was represented as a net structure. However, the method proposed in [8] is far from being practical for use in real problems. As a result, the results provided in the literature are confined to a certain class of Petri nets [6, 9-11] and they are difficult to understand and apply [7, 8, 12].

To overcome these problems and provide a formal and practical methodology for the design of Petri-net-based supervisors, there have been some results reported recently. It has been reported in [13] that in FSM based solutions to the supervisory control problems, the state explosion problem has two effects, the first of which is on the computation of the supervisor and the second is on the size of the supervisor. The former means that the bigger the number of states of a DES, the more computational effort must be expended to come up with a solution for the supervisory control problem. In simple terms, as the DESs become bigger, the computational effort to find a solution grows exponentially. The latter means that the bigger the number of states of a DES, the bigger the size (number of the states) of the supervisor. In simple terms, as the DESs become bigger, the number of the states that must be used in the FSM based supervisors becomes bigger. It has also been reported in [13] that these two state explosion effects can also arise in Petri-net-based supervisory control techniques. Recently, some important Petri-net-based supervisory control techniques have been proposed for solving supervisory control problems [14 - 18]. However, these techniques suffer from the two state explosion effects. These techniques provide very important frameworks which can be used for the better understanding and improvement of Petri-net-based solutions to supervisory control. This understanding and improvement has resulted in the introduction of another important Petri-net-based technique for the supervisory control of DESs [19]. The technique proposed in [19] is a rule-based bottom-up technique. In this case, the size of the supervisor does not become bigger as the DES becomes bigger, but the computational effort still poses the same problem. Although this technique guarantees that the supervisor obtained is nonblocking, it does not guarantee the maximally permissiveness of the supervisor. In this case, it is also necessary to prove the correctness of the supervisor obtained. As an alternative technique to [19], in this paper we propose a technique which ensures that the supervisors obtained are both maximally permissive and nonblocking. In addition, as the supervisors to be synthesised are correct by construction there is no need for verification.

In this paper, a new general, easy to understand and apply, practical and yet powerful top-down Petri-net-based design technique is proposed for the purpose of designing compiled supervisors for the control of DESs in the case of the forbidden state problem. Automation Petri Nets (APN) are used as an underlying formalism for the design of such compiled supervisors for the control of DESs. The approach proposed is based on information feedback on the occurrence of events and Petri net concepts. In particular, we consider discrete event manufacturing systems. The control synthesis procedure proposed in this paper can be applied to both high-level and low-level manufacturing problems. The methodology proposed in this paper offers the following advantages:

- The compiled supervisor and the control policy obtained are correct by construction, i.e. controlled behaviour of the system is nonblocking and does not contradict the forbidden state specifications.
- All events that do not contradict the forbidden state specifications are allowed to happen, i.e. the controlled behaviour of the system is maximally permissive within the specifications.

Note that in this paper supervisory control means the following:

- Monitoring of the system behavior via sensory feedback.
- Control evaluation in accordance with a compiled supervisor and the corresponding supervisory control policy that maps the behaviour of the system to corresponding controls.
- Control enforcement via ladder logic diagram (LLD) implementation of the supervisory control system on a Programmable Logic Controller (PLC).

The remainder of this paper is organised as follows: firstly, Automation Petri nets (APNs) are defined. Then, the definition of supervisory control as used in this paper is given. After that, the synthesis technique proposed in this paper is explained. Next, an experimental discrete manufacturing system example is considered in detail to show the applicability of the proposed technique to low-level real-time supervisory control. Finally, conclusions are given.

2. Automation Petri Nets

As manufacturing systems become more complex, the need for an effective automation tool to produce Discrete Event Control System (DECS) becomes increasingly more important. Petri nets have appeared as the most promising tool to facilitate such design work. In this section, Automation Petri nets (APN) [13], a new formalism for the design of DECSs, are explained. Since ordinary Petri nets do not deal with sensors and actuators, the Petri net concepts are extended by including actions and sensor readings as formal structures within the APN. These extensions involve extending the Petri nets to accommodate sensor signals at transitions and to assign actions to the places. A typical discrete event control system (DECS) is shown in Figure 1.(a). It consists of a discrete event system (DES), to be controlled and a discrete event controller (DEC). Sensor readings are regarded as inputs from the DES to the DEC, and control actions are considered as outputs from the DEC to the DES. The main function of the DEC is to supervise the desired DES operation and to avoid forbidden operations. To do this, the DEC processes the sensor readings and then it forces the DES to conform to the desired specifications through control actions. Petri nets can be used to design such DECs. However, ordinary Petri nets do not deal with actuators or sensors. Because of this, it is necessary to define a Petri net-based controller, which can embrace both actuators and sensors within an extended Petri net framework. An APN is shown in Figure 1.(b). In the APN, sensor readings can be used as firing conditions at transitions. The presence or absence of sensor readings can be used in conjunction with the extended Petri net pre-conditions to fire transitions. In the APN, two types of actuations can be considered, namely impulse actions and level actions. Actions are associated with places. With these additional features, it is possible to design discrete event control systems (DECS) [13]. Figure 1.(c) shows how an APN can be used as a DEC in a DECS.

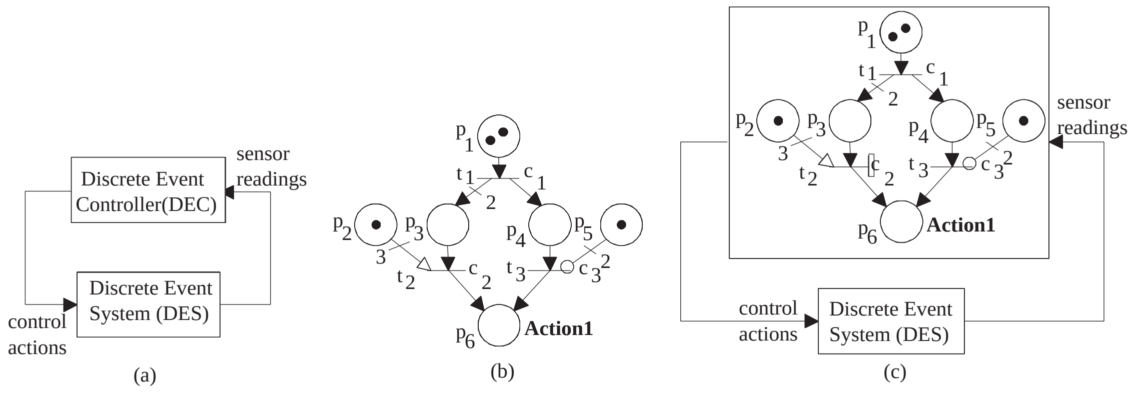


Figure 1. (a). A typical discrete event control system (DECS). (b). Automation Petri Net (APN). (c). APN used as a controller in a DECS.

Formally, an APN can be defined as follows:

$$\mathbf{APN = (P, T, Pre, Post, In, En, \chi, Q, M_0)}$$

Where,

- $P = \{p_1, p_2, ..., p_n\}$ is a finite, nonempty set of places,
- $T = \{t_1, t_2, ..., t_m\}$ is a finite, nonempty set of transitions, $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$,
- Pre: $(P \times T) \rightarrow N$ is an input function that defines weighted ordinary arcs from places to transitions, where N is a set of nonnegative integers,
- Post: $(T \times P) \rightarrow N$ is an output function that defines weighted ordinary arcs from transitions to places,
- In: $(P \times T) \rightarrow N$ is an inhibitor input function that defines weighted inhibitor arcs from places to transitions,
- En: $(P \times T) \rightarrow N$ is an enabling input function that defines weighted enabling arcs from places to transitions,
- $\chi = \{\chi_1, \chi_2, ..., \chi_m\}$ is a finite, nonempty set of firing conditions associated with the transitions,
- $Q = \{q_1, q_2, ..., q_n\}$ is a finite set of actions that might be assigned to the places,
- $M_0 : P \rightarrow N$ is the initial marking.

The APN consists of two types of nodes called places, represented by circles (○), and transitions, represented by bars (—). There are three types of arcs used in the APN, namely, ordinary arcs, represented by a directed arrow ($\longrightarrow$), inhibitor arcs, represented by an arrow, whose end is a circle (—∘), and finally enabling arcs, represented by a directed arrow, whose end is empty (—▷). Weighted and directed ordinary arcs connect places to transitions and vice versa, while weighted enabling and inhibitor arcs connect only places to transitions. Places represent the status of the system and transitions represent events. Each transition has a set of input and output places, which represent the pre-condition and post-condition of the transition. The actions (Q), assigned to the places, can be either impulse actions or level actions. Impulse actions are enabled at the instant when a token is deposited into the place and level actions are enabled when there is a token(s) at the place. More than one action may be assigned to a place. Firing conditions in the APN are recognised as external events, such as sensor readings. A firing condition, χ, associated with a transition t, is a Boolean variable that can be 0, in which case related transition t is not allowed to fire, or it can be 1, in which case related transition t is allowed to fire if it is enabled. The marking of the APN is represented by the number of tokens in each place. Tokens are represented by black dots ($\bullet$). Movement of tokens between places describes the evolution of the APN and is accomplished by the firing of the enabled transitions. The following rules are used to govern the flow of tokens:

Enabling Rules:

1. If the input place p_1 of a transition t_1 is connected to t_1 with a weighted ordinary arc Pre(p_1,t_1), then t_1 is said to be enabled when p_1 contains at least the number of tokens equal to the weight of the directed ordinary arc, i.e. M(p_1) $\geq$ Pre(p_1,t_1).

2. If the input place p_1 of a transition t_1 is connected to t_1 with a weighted enabling arc En(p_1,t_1), then t_1 is said to be enabled when p_1 contains at least the number of tokens equal to the weight of the enabling arc, i.e. M(p_1) $\geq$ En(p_1,t_1).

3. If the input place p_1 of a transition t_1 is connected to t_1 with a weighted inhibitor arc In(p_1,t_1), then t_1 is said to be enabled when p_1 contains less tokens than the weight of the inhibitor arc, i.e. M(p_1) $<$ In(p_1,t_1).

Firing Rules: In the APN, an enabled transition t can or cannot fire depending on the external firing condition χ of t. These firing conditions can be a positive level or zero level of a sensor reading. Broadly speaking, a firing condition χ may include more than one sensor reading with 'AND', 'OR' and 'NOT' logical operators. When dealing with more than one sensor reading as firing conditions, the logical operators of firing conditions must be taken into account accordingly. In the special case, where $\chi = 1$, transition t is always allowed to fire when it is enabled. When an enabled transition t fires, it removes Pre(p_i,t) tokens from each input place p_i and deposits, at the same time, Post(t,p_o) tokens in each output place p_o. It should be noted that the firing of an enabled transition t does not change the marking of the input places that are connected to t only by weighted enabling or inhibitor arcs. It is also possible to consider timed APNs, as in normal Petri nets.

3. Supervisory Control of Discrete Event Systems

A typical supervisory control of a DES, as used in this paper, is shown in Fig. 2. It consists of four parts; i) the discrete event system (DES) to be controlled, ii) the supervisor, iii) sensor readings as outputs from the DES, and iv) control actions as inputs to the DES. The objective of the supervisor is to make sure that no forbidden state will be reached, and the controlled system operation is maximally permissive, i.e. the supervisor does not unnecessarily constrain the system operation.

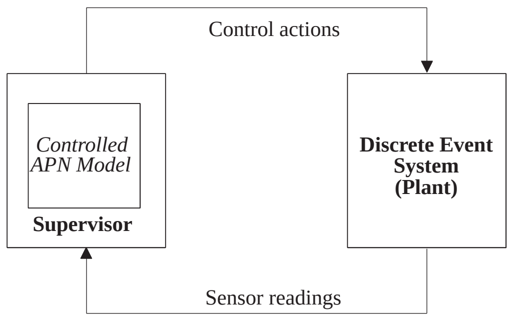


Figure 2. Supervisory control of DESs by means of the controlled model of the system.

The plant and the supervisor are assumed to run in parallel in the following fashion. When an event occurs in the plant this is realised by the supervisor through sensory feedback obtained from the plant. This results in the state change within the supervisor. Since the supervisor is a state-feedback controller, its controlling actions are fully determined by the current state of the plant. Therefore, at the current state, the supervisor provides a set of actions to force the plant to behave according to the specifications considered. The supervisor, in this case, is made up of the controlled model of the plant. The controlled behaviour is the subset of the uncontrolled behaviour that survives under supervision.

The uncontrolled behaviour of the plant is captured by the uncontrolled model of the plant, which is represented by an APN. The behaviour of the APN model is restricted according to the control policy. To do this, enabling arcs are connected from places within the APN model to some of its controllable transitions that are related to the specifications, such that the control policy is satisfied. This means that the APN model has an enabling effect over itself. In the controlled APN model, some places have actions assigned to them, and there are controllable and uncontrollable transitions. When an event occurs in the plant, it causes the token flow in the controlled APN model.

At some states of the system, if not supervised, the system can get into a forbidden state through the firing of controllable transitions. To prevent this, the supervisory control policy simply defines a set of markings and corresponding controllable transitions of the APN model, which must be enabled only at particular markings. By only allowing the controllable transitions to fire at certain markings, this process also acts as a blocking process so as to prevent the system from reaching the forbidden state, but ensuring that every admissible state of the system can still be reached, i.e. the supervisor is maximally permissive. Note that an enabling arc is used to check the presence of a token(s) in a place. When there is a token(s) in the corresponding places of the APN model, the related controllable transitions, connected to these places via enabling arcs, are enabled. In contrast, when there are no tokens in the corresponding places, the related controllable transitions are blocked.

In brief, the events occurring in the plant are realised by the controlled APN model (the supervisor) as a sensory feedback. Then, the controlled APN model changes its state accordingly. If there are controllable transitions to be enabled in the controlled APN model, this is carried out according to the control policy. Next, the controlled APN model fires its transitions, according to the sensory feedback and the supervision. When there is a token in the places, to which an action(s) is assigned, this is used as a control action to tell the plant what to do, i.e. to start or stop motors, machines, actuators and the like.

How an enabling arc is used to enable or block a controllable transition in the APN can be shown as follows. Consider Fig. 3.(a), where there are three places $P = \{p_1, p_2, p_3\}$ and one transition $T = \{t_1\}$, with firing condition χ_1 and an action A is assigned to p_3. The arc connecting p_2 to t_1, En(p_2,t_1) is an enabling arc. When there is a token(s) each in p_1 and p_2, t_1 is enabled. Thus t_1 fires with the firing condition χ_1 by removing the token from p_1 and depositing a token in p_3. Note that M(p_2) does not change with the firing of t_1. In this case, if there is a token(s) in p_2, then t_1 is enabled. However, if there is no token in p_2, then t_1 is blocked. One may think that an enabling arc is similar to a *read-write arc*, as shown in Fig. 3.(b), but enabling arcs are distinctively different from ordinary arcs in the sense that they do not lead to conflicts in a Petri net. This is shown in Fig. 3.(c), where there are two enabling arcs, namely En(p_2,t_1) and En(p_2,t_2), and t_1 and t_2 can fire at any time without any conflict. However, if we consider the Petri net shown in Fig. 3.(d), then it is obvious that in this case there is a potential conflict between t_1 and t_2.

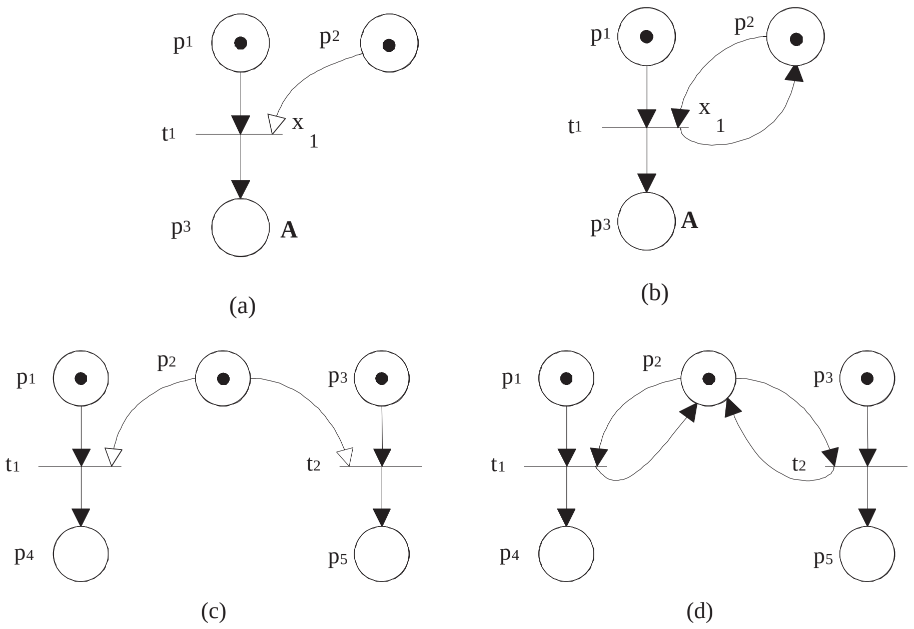


Figure 3. Enabling arcs versus ordinary arcs.

4. The Synthesis Technique

The technique proposed in this paper is a top-down synthesis technique, involving the construction of the reachability graph (RG) of the uncontrolled APN model of the system. In this technique, the supervisor to be constructed is the controlled APN model of the DES. The controlled APN model has an enabling/blocking effect over itself and the supervisory control policy is a table that provides a list of related controllable transitions to be enabled at certain states of the maximally permissible state space. The supervisory control policy is enforced by connecting enabling arcs from the places of the APN model to its related controllable transitions, such that the forbidden state specifications are met. The top-down synthesis technique proposed in this paper for the supervisory control of the DESs is divided into four main steps:

Step 1 - Design the uncontrolled model of the system using APNs

Step 2 - Determine the control policy

Step 3 - Construct the controlled model of the system

Step 4 - Implement the supervisor (the controlled model) on a programmable logic controller (PLC) as ladder logic diagrams (LLDs)

Step 1 - Design the Uncontrolled Model of the System Using APNs

In this paper, APNs are used for designing the uncontrolled models of the DESs in order to capture the uncontrolled behaviour of the system. In practical modelling, the firing of an enabled transition is generally associated with an external event, such as sensor readings. This means that a transition is fired when it is enabled and a related external event occurs. The external events are subdivided into *controllable events*, i.e. the events which may be disabled through control action, and *uncontrollable events*, i.e. the events which may not be disabled through control action. Models of the uncontrolled behaviour of the DESs can be designed efficiently by means of a modular modelling concept, which is based on a set of predefined, standard modules for typical devices of the DESs, such as actuators, drives, valves, pushers, stopper and FIFO queues. The *concurrent composition* [8] can then be used to merge common transitions to form the uncontrolled model.

The final uncontrolled APN model consists of a set of places and a set of transitions connected to each other. The number of tokens in each place represents the state of the APN. When a transition is enabled it may fire with an external event, realised by a sensor reading. When a transition fires, tokens are moved from one place to another. Actions, which are associated with places, are assumed to have an enabling effect on the actuators in the real system. Some transitions in the uncontrolled model are controllable, i.e. they may be enabled/blocked (disabled) by control action, and some others are uncontrollable, and cannot be enabled/blocked (disabled) by control action.

Step 2 - Determine the Control Policy

The objective in this step is to determine a supervisory control policy, using the uncontrolled APN model constructed in the previous step, so that controlled behaviour of the system will be maximally permissive and will conform to the forbidden state specifications given. In this step the following sub-steps are considered:

Step 2.1. Generate the reachability graph of the APN model

Step 2.2. Identify and remove the "bad states" from the reachability graph

Step 2.3. Determine the control policy

Step 2.1. Generate the reachability graph of the APN model

The reachability graph (RG), in which each node represents a marking reachable from the initial marking M_0 and each arc represents the firing of a transition, of the uncontrolled APN model is generated. Note that the RG represents the uncontrolled behaviour of the DES considered. In other words, it represents all the possible markings, i.e. the whole state space, of the system.

Step 2.2. Identify and remove the "bad states" from the reachability graph

The forbidden state specifications are a compact way of defining classes of undesirable markings which should be avoided [7]. Generally, they are given as abstract explanations about the system considered, but they can be converted into a set of RG states, what we call "bad markings" or "bad states". In the uncontrolled behaviour of the system, these "bad states" can be reached. However, in the controlled behaviour they must be avoided. Therefore, the first thing to do in this step is to identify the "bad states" according to the forbidden state specifications given. Next, these "bad states" are removed from the RG together with their related arcs connecting them to the rest of the RG. In some cases, the forbidden state problem can map to a "*bad transition*", in which case the related "*bad transition(s)*" is removed from the RG. Then, "*unreachable states*", i.e. states to which there are no arcs coming from the other states and "*blocking states*", i.e. states from which there are no arcs going to the other states, are also removed from the RG. This process yields the final reduced reachability graph (FRRG), which represents the maximally permissible behaviour of the system.

Step 2.3. Determine the control policy

According to the forbidden state specifications, the supervisory control policy for the system is determined by using the FRRG. This is done as follows: firstly, the controllable transitions that are related to the forbidden state specifications are determined. Then, each related controllable transition of the uncontrolled APN model is considered with its related event. Next, a set of arcs of the FRRG with that particular event are identified. These arcs are called *identical arcs*. Then, the input markings of these identical arcs are determined. These input markings are called *base markings*. Finally, these base markings are declared as markings at which the corresponding controllable transition is to be enabled by enabling arcs. As a result, in one column of a table, all related controllable transitions of the uncontrolled APN model are provided. In the next column, the base markings for each related controllable transition are provided. This table constitutes the control policy.

Step 3 - Construct the controlled model of the system

In this step, the controlled model (the supervisor) of the system is obtained. To do this, enabling arcs are connected from places within the APN model to its related controllable transitions, such that the control policy is satisfied. This means that the APN model has an enabling/blocking effect over itself. However, it is important to note that when connecting enabling arcs from a place to a related controllable transition, if there is already an input arc connecting the same place to the same transition, then the enabling arc is simply omitted, because the ordinary arc will do the same job and therefore there is no need for a superfluous arc. When connecting the places of the model to its controllable transitions with enabling arcs, if there is more than one marking to enable the same controllable transition, then the transition is duplicated as many as the number of these markings. This is done simply to accommodate the *or* operation within the Petri net formalism. The supervised model of the system obtained is maximally permissive and behaves according to the specifications. Note that the controlled behaviour of the system is the sub-set of the uncontrolled behaviour that survives under supervision. It is also important to note that as the behaviour of the controlled model (the supervisor) of the system is correct by construction, there is no need for verification.

Step 4- Implement the supervisor on a PLC as LLDs

The design phase is only the first step towards the control of DESs. After designing a controller (supervisor), it is necessary to have an automatic means for the generation of control code from the controller. The supervisory control can be enforced by implementing the supervisor on an industrial computer. The implementation can be done by using high-level languages, such as C, or low-level languages, such as machine language. Since programmable logic controllers (PLCs) with a graphical symbolic programming language, called ladder logic diagrams (LLDs), are the most popular implementation tools in today's automated modern factories, in this paper a technique is used for converting the controlled model (the supervisor) into an LLD code for implementation on a PLC. The technique used is called the Token Passing Logic (TPL) methodology, details of which can be found in [20,21]. In brief, to convert an APN into an LLD code, *counters* are assigned to the places whose token capacity is bigger than or equal to 1, and *flags* are assigned to the places whose token capacity equals 1. The simulated movement of tokens is achieved by incrementing and decrementing the counters (or setting and resetting the flags).

5. An Example for Low-Level Supervisory Control

5.1. Problem Description

Let us now consider a discrete manufacturing system to show the applicability of the synthesis technique proposed to low-level supervisory control of DESs. The experimental discrete manufacturing system, shown in Fig. 4, represents component sorting and assembly processes. The upper conveyor and the lower conveyor are driven by the upper conveyor motor (Actuator 1) and the lower conveyor motor (Actuator 2) respectively. A random selection of metallic pegs and plastic rings are placed on the upper conveyor. The rings and pegs need to be identified and separated. This is done by two sensors, a proximity sensor (Sensor 1) and an infra-red reflective sensor (Sensor 2). By using these two sensors a distinction can be made between the peg and the ring. By means of the sort solenoid (Actuator 3), plastic rings can be ejected down the assembly chute, which can have up to five plastic rings. Metallic pegs, meanwhile, continue on the upper conveyor and are deflected down the feeder chute. The feeder chute automatically feeds pegs onto the lower conveyor. An infra-red emitter/detector (Sensor 3) is used to determine whether or not the assembly area is empty. If it is empty, the assembly solenoid (Actuator 4) is used to dispense a ring from the assembly chute into the assembly area. The assembly area is positioned just above the lower conveyor and, when a metallic peg passes, the peg engages with the hole in the ring and the two components are assembled. The lower conveyor is used to carry assembled components into the collection tray. A Siemens PLC (S5-100U) is used to control the process, and a PC-based package called 'Quadriga' is used to program the PLC. PLC inputs and outputs are given in Table 1 and in Table 2 respectively.

Table 1. PLC inputs.

PLC Inputs	Sensor Number	Definition
I0.0	1	Detects a ring or a peg at the sort area
I0.1	2	Detects a peg at the sort area
I0.2	3	Detects a ring in the assembly area

Table 2. PLC outputs.

PLC Outputs	Actuator Number	Definition
Q2.0	1	Upper conveyor motor
Q2.1	2	Lower conveyor motor
Q2.2	3	Sort solenoid
Q2.3	4	Assembly solenoid

For the sake of simplicity, it is assumed that the assembly chute can have only one ring at a time. It is also assumed that when the system is switched on, both the upper conveyor motor and the lower conveyor motor are switched on automatically. The forbidden state specifications are as follows:

- Operate the sort solenoid only when there is space in the assembly chute and there is a ring at the sort area.
- Operate the assembly solenoid only when there is space at the assembly area and there is a ring in the assembly chute.

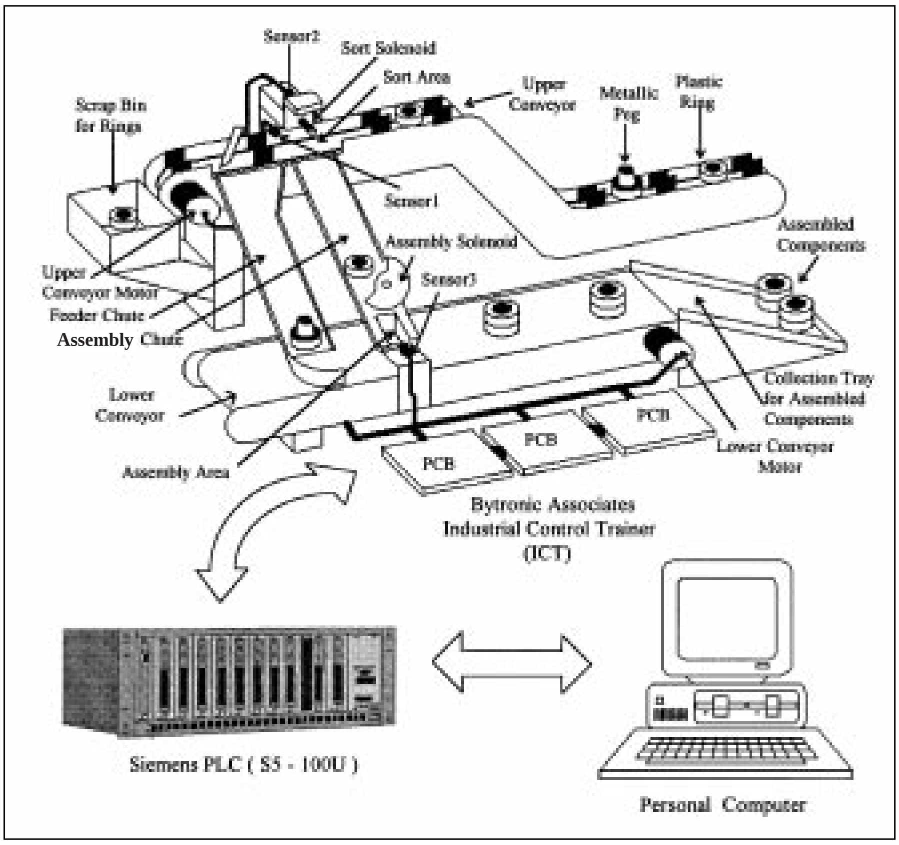


Figure 4. The experimental discrete manufacturing system.

5.2. Synthesis of Supervisory Controller

Step 1 - Design the Uncontrolled Model of System Using APNs

As a first step in capturing the uncontrolled behaviour of the manufacturing system, consider the standard APN modules and structures given in Fig. 5, where there are ten places, P = $\{p_1, p_2, ..., p_{10}\}$ and nine transitions, T = $\{t_1, t_2, t_3, t_3', t_4, t_4', t_5, t_6, t_7\}$, with which firing conditions $\chi_1 = \overline{I0.0}$, χ_2 = I0.0 & $\overline{I0.1}$, $\chi_3 = \chi_3' = \overline{I0.0}$, $\chi_4 = \chi_4'$ = I0.2, $\chi_5 = \overline{I0.2}$, $\chi_6 = 1$, $\chi_7 = 1$, are associated respectively. Note that transitions t_3, t_3' and t_5 are timed transitions with time delays 0.7 s, 0.7 s and 1.5 s respectively. Places p_7 and p_8 represent the *off* and *on* states of the sort solenoid respectively. Likewise, places p_9 and p_{10} represent the *off* and *on* states of the assembly solenoid respectively. A token in places p_1, p_3 and p_5 represent the available spaces in the sort area, in the assembly chute and in the assembly area respectively. A token in places p_2, p_4 and p_6 depicts the presence of a plastic ring in the sort area, in the assembly chute and in the assembly area respectively. Initially, both solenoids are *off* and there are no plastic rings in the experimental discrete manufacturing system.

When there is no ring at the sort area, i.e. M $(p_1) = 1$, and the presence of a ring is detected, i.e. χ_2 = I0.0 & $\overline{I0.1}$, transition t_2 fires by removing the token from place p_1 and by depositing a token into place p_2. This means that there is a ring at the sort area, i.e. M $(p_2) = 1$. When there is a ring at the sort area it either clears the sort area through transition t_1 or is put into the assembly chute through transition t_3. If there is a ring at the sort area, the sort solenoid is *off*, i.e. M $(p_7) = 1$, and the absence of a ring is detected, i.e. $\chi_1 = \overline{I0.0}$ then transition t_1 fires by removing the token from place p_2 and by depositing a token in place p_1. This means that the ring cleared the sort area. If there is a ring at the sort area, i.e. M $(p_2) = 1$, the sort solenoid is *on*, i.e. M $(p_8) = 1$, there is space in the assembly chute, i.e. M $(p_3) = 1$, and the absence of a ring is detected, i.e. $\chi_3 = \overline{I0.0}$, then timed-transition t_3 is being fired for 0.7 s, after which the token at the sort area is removed, i.e. M $(p_2) = 0$, and a token is deposited into the assembly chute, i.e. M $(p_4) = 1$, by using the empty space in the assembly chute, i.e. M $(p_3) = 0$. This means that the ring at the sort area is put into the assembly chute by means of the sort solenoid and this process takes 0.7 s.

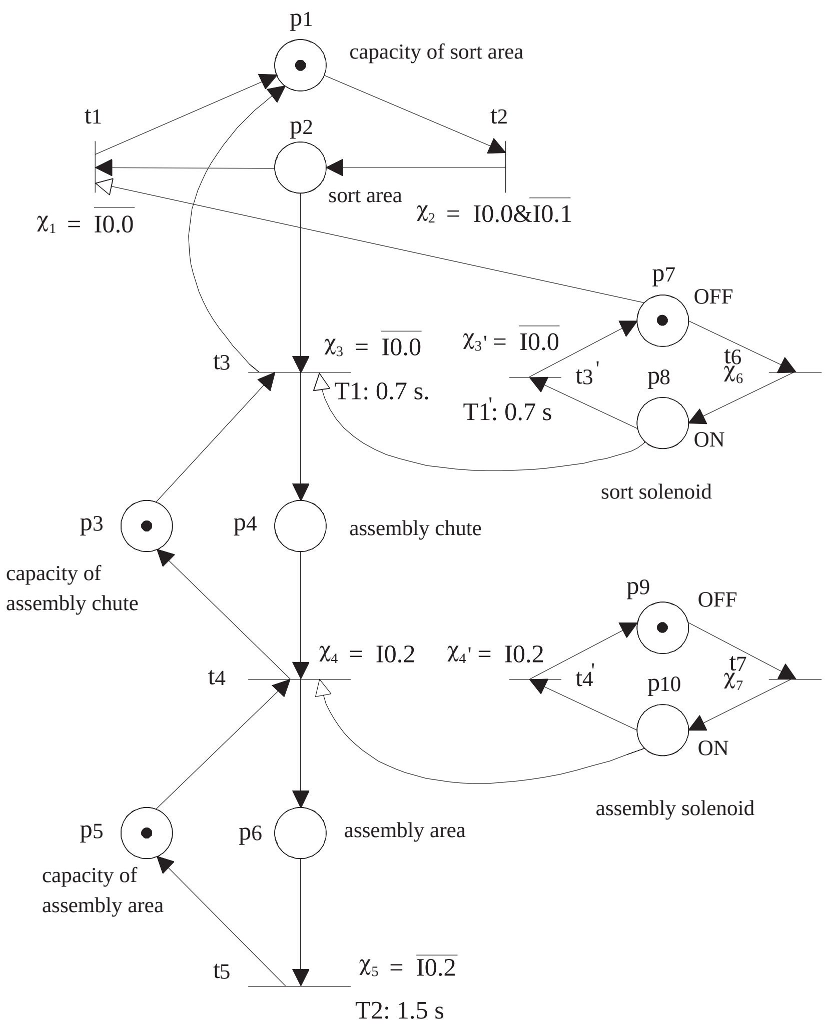


Figure 5. The standard APN modules and structures for the manufacturing system.

If there is a ring in the assembly chute, i.e. M $(p_4) = 1$, there is space at the assembly area, i.e. M $(p_5) = 1$, and the assembly solenoid is *on*, i.e. M $(p_{10}) = 1$, then the ring is dispensed from the assembly chute to the assembly area, i.e. the tokens are removed from places p_4 and p_5 and a token is deposited into place p_6, by means of transition t_4 with $\chi_4 = 10.2$. This also means that there is space in the assembly chute, i.e. M $(p_3) = 1$. If there is a ring at the assembly area, i.e. M $(p_6) = 1$, and a peg engages with the hole in the ring, i.e. $\chi_5 = \overline{I0.2}$, then it takes 1.5 s for the ring and the peg to be assembled and to clear the assembly area. After this, there is space in the assembly area, i.e. M $(p_5) = 1$.

Secondly, by using the concurrent composition, i.e. by merging the transitions with the same events, the uncontrolled model is obtained. It is obvious from Fig. 5 that timed-transitions t_3 and t_3' have the same time delay as well as the same firing condition (event). Therefore, they are merged as t_3 in the uncontrolled model. Similarly, transitions t_4 and t_4' have the same firing condition (event). Therefore, they are merged as t_4 in the uncontrolled model. The uncontrolled model of the manufacturing system is obtained as an APN as shown in Fig. 6, where there are ten places, $P = \{p_1, p_2, ..., p_{10}\}$ and seven transitions $T = \{t_1, t_2, ..., t_7\}$, with which the firing conditions, $\chi = \{\chi_1, \chi_2, ..., \chi_7\}$ are associated respectively. In the uncontrolled APN model transitions t_3 and t_5, are timed-transitions with time delays 0.7 sec and 1.5 s respectively. Note that actions Q2.2 and Q2.3 are assigned to places p_8 and p_{10} respectively. They represent the sort solenoid and the assembly solenoid operations respectively. It is important to point out that after merging transitions t_3 and t_3' of Fig. 5, as t_3 in the uncontrolled model, the enabling arc $En(p_8, t_3)$, connecting place p_8 to transition t_3, is omitted, because there is already a normal arc $Pre(p_8, t_3)$, connecting the same place to the same transition. The same applies to the enabling arc $En(p_{10}, t_4)$, connecting place p_{10} to transition t_4. The initial marking of the uncontrolled model is $M_0 = (1, 0, 1, 0, 1, 0, 1, 0, 1, 0)^T$ or simply $M_0 = (1, 3, 5, 7, 9)$. This means that initially, there is no ring in the manufacturing system and both the sort solenoid and the assembly solenoid are *off*. Note that events χ_1, χ_2, and χ_5 are uncontrollable events, while events χ_3, χ_4, χ_6 and χ_7 are controllable events. In fact, the objective in this case is to come up with a supervisor to decide when to fire transitions t_6 and t_7 such that the forbidden state specifications are met. Note that the uncontrolled APN model, shown in Fig. 6, is safe, i.e. 1-bounded, live, reversible, and conservative.

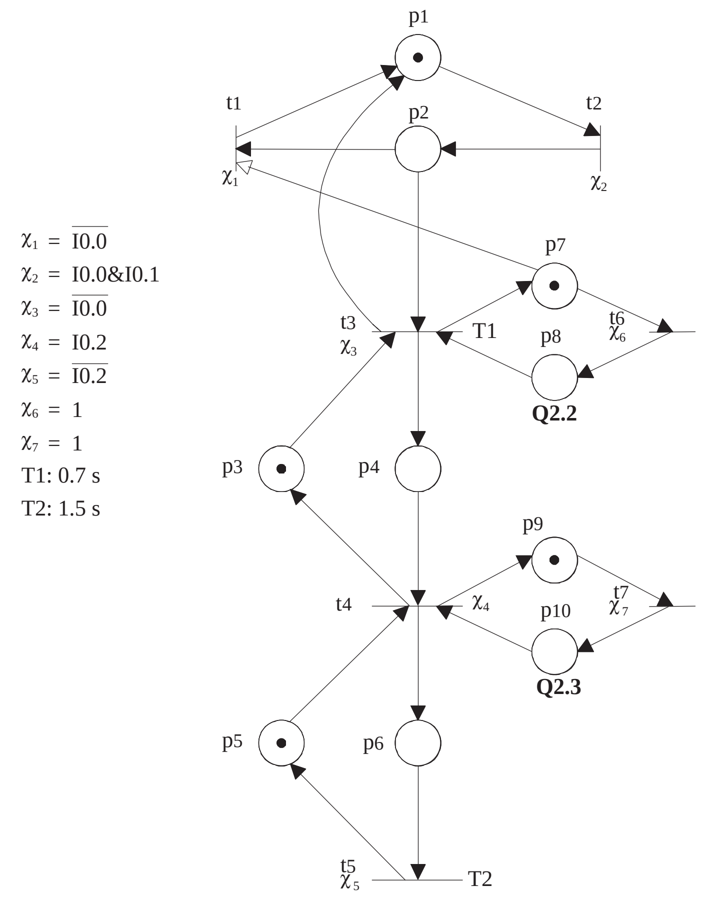


Figure 6. The uncontrolled APN model of the experimental manufacturing system.

Step 2 - Determine the control policy

Recall that in this step there are three sub-steps:

Step 2.1. Generate the reachability graph of the APN model

Step 2.2. Identify and remove the "bad states" from the reachability graph

Step 2.3. Determine the control policy

Step 2.1. Generate the reachability graph of the APN model

The reachability graph (RG) of the uncontrolled APN model is shown in Fig. 7, where there are seventy-nine arcs, representing the firing of transitions in the uncontrolled model, and there are thirty-two nodes M = $\{M_0, M_1, M_2, ..., M_{31}\}$, representing all possible markings reachable from the initial marking M_0. Table 3 provides detailed information about the RG nodes. Note that for simplicity reasons only the

Table 3. The markings of the reachability graph (RG).

Marking	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}
M_0 = (1,3,5,7,9)	1		1		1		1		1	
M_1 = (2,3,5,7,10)		1	1		1		1			1
M_2 = (1,3,5,7,10)	1		1		1		1			1
M_3 = (1,3,5,8,9)	1		1		1			1	1	
M_4 = (1,3,5,8,10)	1		1		1			1		1
M_5 = (2,3,5,7,9)		1	1		1		1		1	
M_6 = (1,3,6,8,9)	1		1			1		1	1	
M_7 = (2,3,5,8,10)		1	1		1			1		1
M_8 = (2,3,5,8,9)		1	1		1			1	1	
M_9 = (1,3,6,8,10)	1		1			1		1		1
M_{10} = (2,3,6,8,10)		1	1			1		1		1
M_{11} = (1,3,6,7,10)	1		1			1	1			1
M_{12} = (2,4,5,8,10)		1		1	1			1		1
M_{13} = (2,4,5,8,9)		1		1	1			1	1	
M_{14} = (2,4,5,7,9)		1		1	1		1		1	
M_{15} = (1,4,5,7,9)	1			1	1		1		1	
M_{16} = (1,4,5,8,9)	1			1	1			1	1	
M_{17} = (2,4,6,8,10)		1		1		1		1		1
M_{18} = (2,4,6,8,9)		1		1		1		1	1	
M_{19} = (2,4,6,7,9)		1		1		1	1		1	
M_{20} = (1,4,6,7,9)	1			1		1	1		1	
M_{21} = (1,4,6,8,9)	1			1		1		1	1	
M_{22} = (2,4,6,7,10)		1		1		1	1			1
M_{23} = (2,3,6,8,9)		1	1			1		1	1	
M_{24} = (1,4,6,7,10)	1			1		1	1			1
M_{25} = (1,4,6,8,10)	1			1		1		1		1
M_{26} = (2,4,5,7,10)		1		1	1		1			1
M_{27} = (2,3,6,7,9)		1	1			1	1		1	
M_{28} = (2,3,6,7,10)		1	1			1	1			1
M_{29} = (1,4,5,8,10)	1			1	1			1		1
M_{30} = (1,4,5,7,10)	1			1	1		1			1
M_{31} = (1,3,6,7,9)	1		1			1	1		1	

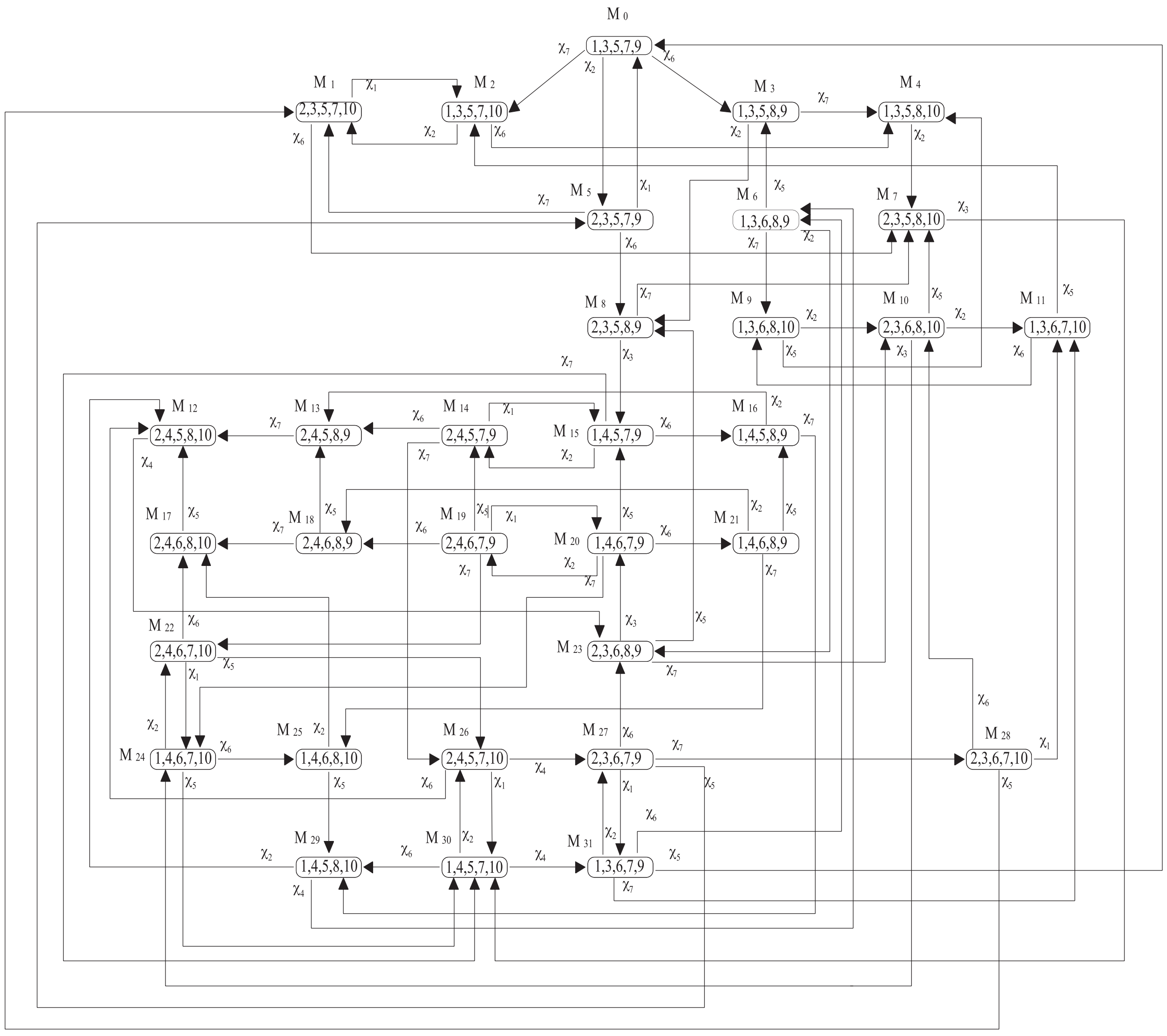


Figure 7. The reachability graph (RG) of the uncontrolled APN model.

events which are associated with the transitions are shown in the RG. Therefore, the events (firing conditions) $\chi = \{\chi_1, \chi_2, ..., \chi_7\}$ in the RG represent the firing of corresponding transitions $T = \{t_1, t_2, ..., t_7\}$ respectively. It is also important to note that although it is not explicitly written in the RG, time delays of 0.7 s and 1.5 s are associated with the firing of transitions t_3 and t_5 respectively.

Step 2.2. Identify and remove the "bad states" from the reachability graph

Let us now consider the forbidden state specifications:

Specification 1: "Operate the sort solenoid only when there is space in the assembly chute and there is a ring at the sort area". This also implies that when there is no space in the assembly chute and/or there is no ring at the sort area, do not operate the sort solenoid. Therefore, it is obvious from Fig. 7 that markings M_3, M_4, M_5 and M_9 are bad states (i.e. bad markings), because they represent the states where there is no ring at the sort area and the sort solenoid is *on.* Then, markings M_{16}, M_{21}, M_{25} and M_{29} are bad states because they represent the states where there is no ring at the sort area, there is a ring in the assembly chute and the sort solenoid is *on.* Finally, markings M_{12}, M_{13}, M_{17} and M_{18} are bad states, because they represent the states where there is a ring at the sort area, there is a ring in the assembly chute and the sort solenoid is *on*.

Specification 2: “Operate the assembly solenoid only when there is space at the assembly area and there is a ring in the assembly chute”. This also implies that when there is no ring in the assembly chute and/or there is no space at the assembly area, do not operate the assembly solenoid. As can be seen from Fig. 7, markings M_1, M_2, M_4 and M_7 are bad states, because they represent the states where there is no ring in the assembly chute, there is no ring at the assembly area and the assembly solenoid is *on*. Then, markings M_{17}, M_{22}, M_{24} and M_{25} are also bad states, because they represent the states where there is a ring in the assembly chute, there is a ring at the assembly area and the assembly solenoid is *on*. Finally, markings M_{10}, M_{11}, and M_{28} are bad states, because they represent the states where there is a ring at the assembly area, there is no ring in the assembly chute and the assembly solenoid is *on*.

As a result, according to the forbidden state specifications, there are twenty bad markings (states), namely, M_1, M_2, M_3, M_4, M_6, M_7, M_9, M_{10}, M_{11}, M_{12}, M_{13}, M_{16}, M_{17}, M_{18}, M_{21}, M_{22}, M_{24}, M_{25}, M_{28} and M_{29}, as shown in Fig. 8.

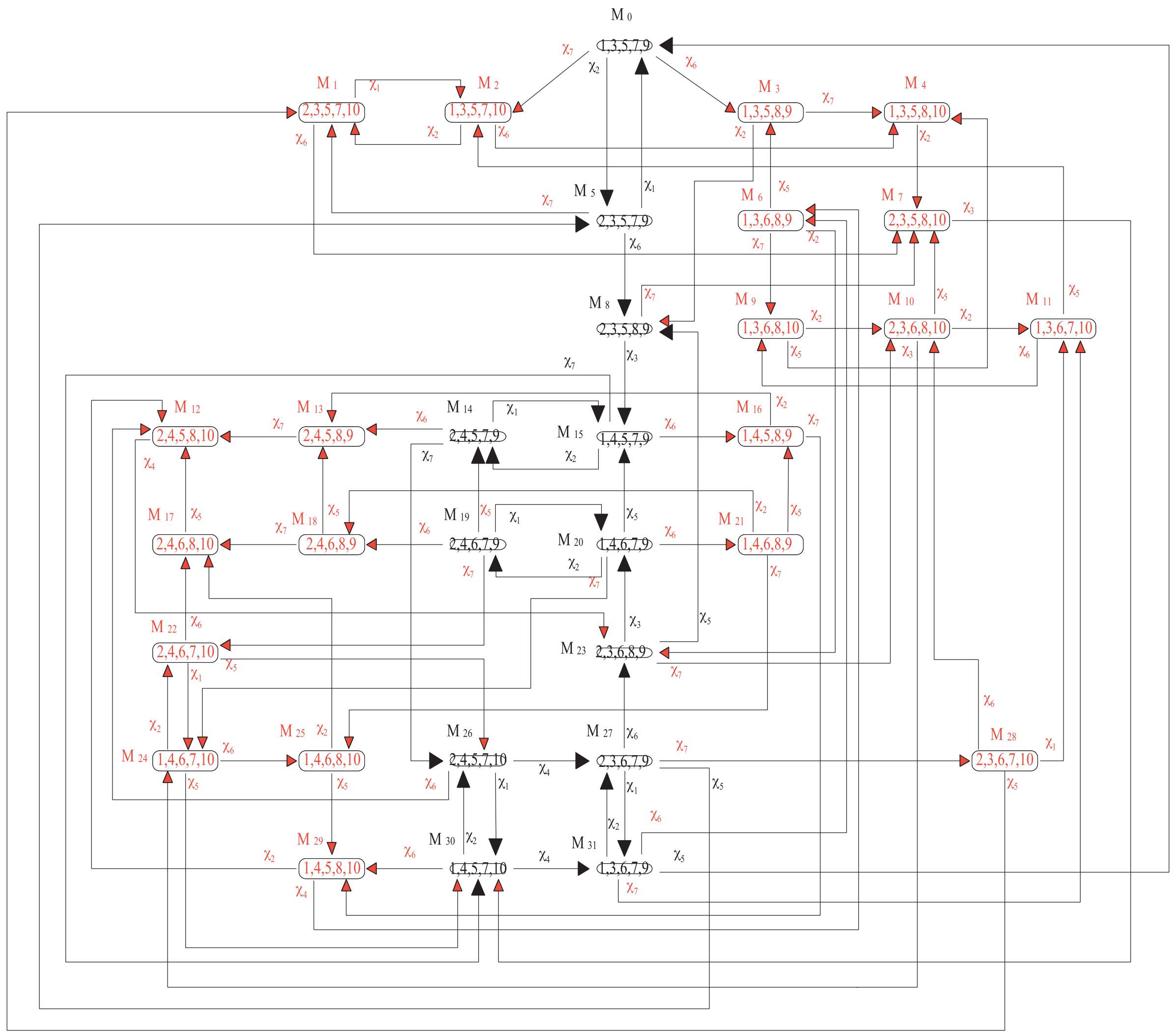


Figure 8. The ‘bad markings’ and the ‘**good markings**’ of the reachability graph (RG).

These bad markings must be removed from the RG together with their arcs connecting them to the rest of the RG. After removing these bad markings and their arcs from the RG, the final reduced reachability graph (FRRG) is obtained, as shown in Fig. 9. Note that the FRRG represents the maximally permissible state space for the forbidden state specifications given.

Step 2.3. Determine the control policy

In order to determine control policy, firstly it is necessary to determine controllable transitions that are related to the forbidden state specifications. Recall that the forbidden state specifications are as follows: 1. "Operate the sort solenoid only when there is space in the assembly chute and there is a ring at the sort area." 2. "Operate the assembly solenoid only when there is space at the assembly area and there is a ring in the assembly chute". As can be seen from Fig. 5 and Fig. 6, when there is a token in place p_8 the sort solenoid is in operation. The controllable transition t_6 is responsible for putting a token into p_8. Similarly, when there is a token in p_{10} the assembly solenoid is in operation. The controllable transition t_7 is responsible for putting a token into p_{10}. Therefore, in this case the controllable transitions t_6 with the event χ_6 and t_7 with the event χ_7 are related to the forbidden state specifications. In other words, the objective of the control policy is to decide when to let transitions t_6 and t_7 fire such that the forbidden state specifications are met. It can be seen from the FRRG, shown in Fig. 9, that the firing of t_6 is represented by identical arcs $M_5[\chi_6 > M_8$ and $M_{27}[\chi_6 > M_{23}$. and therefore the base markings of t_6 are $M_5 = (2, 3, 5, 7, 9)$ and $M_{27} = (2, 3, 6, 7, 9)$ from the RG. Thus, in the control policy, base markings M_5 and M_{27} are declared as markings at which t_6 is to be enabled by enabling arcs. This is the control policy for t_6. t_7 with event χ_7 having the identical arcs $M_{14}[\chi_7 > M_{26}$ and $M_{15}[\chi_7 > M_{30}$ and therefore, base markings of t_7 are $M_{14} = (2, 4, 5, 7, 9)$ and $M_{15} = (1, 4, 5, 7, 9)$ from the FRRG. Thus, in the control policy, base markings M_{14} and M_{15} are declared as markings at which t_7 is to be enabled by enabling arcs. This is the control policy for t_7. The resulting control policy is shown in Table 4.

Table 4. The supervisory control policy for the experimental discrete manufacturing system.

Related transition	Markings at which the related transition is to be enabled
t_6	M_5= (2,3,5,7,9) *or* M_{27}= (2,3,6,7,9)
t_7	M_{14}= (2,4,5,7,9) *or* M_{15}= (1,4,5,7,9)

Step 3 - Construct the controlled model of the system

The controlled model (i.e. the supervisor), shown in Fig. 10 is obtained by using the uncontrolled APN model, shown in Fig. 6, and the supervisory control policy given in Table 4. Firstly, let us consider the first row of the control policy. As the controllable transition t_6 of the uncontrolled APN model is to be enabled at marking M_5= (2,3,5,7,9) *or* M_{27}= (2,3,6,7,9), it is duplicated to accommodate the *or* operation within the Petri net formalism and replaced with transitions t_6 and t_6' within the controlled APN model. So, t_6 must be enabled only at M_5 and it must be blocked at other markings. To implement this control policy, enabling arcs En(p_2, t_6), En(p_3, t_6), En(p_5, t_6), and En(p_9, t_6) are connected from p_2, p_3, p_5 and p_9 to t_6, respectively. However, since there is an input arc connecting p_7 to t_6, there is no need to connect a superfluous enabling arc, En(p_7,t_6), from the same place. Therefore t_6' must be enabled only at M_{27} and it must be blocked at other markings. To implement this control policy, enabling arcs En(p_2, t_6'), En(p_3, t_6'), En(p_6, t_6'), and En(p_9, t_6') are connected from p_2, p_3, p_6 and p_9 to t_6', respectively. However, since there is an input arc connecting p_7 to t_6', there is no need to connect a superfluous enabling arc, En(p_7,t_6'), from the same place.

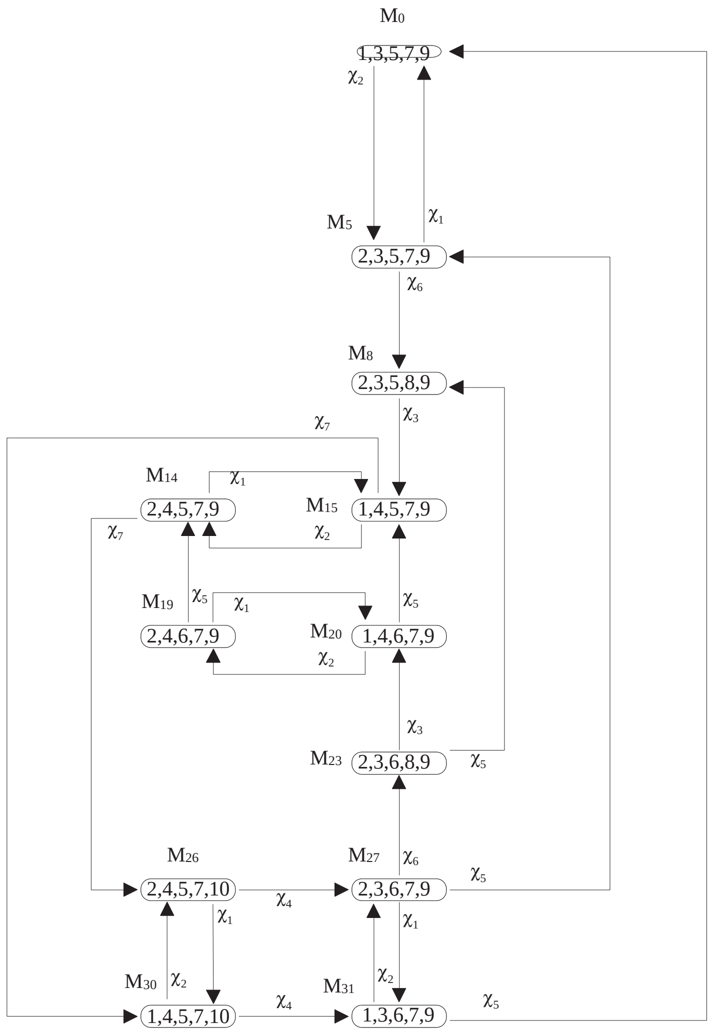


Figure 9. The final reduced reachability graph (FRRG), according to the forbidden state specifications.

Let us now consider the second row of the control policy. As the controllable transition t_7 of the uncontrolled APN model is to be enabled at marking M_{14}= (2,4,5,7,9) *or* M_{15}= (1,4,5,7,9), it is duplicated to accommodate the *or* operation within the Petri net formalism and replaced with transitions t_7 and t_7' within the controlled APN model. So, t_7 must be enabled only at M_{14} and it must be blocked at other markings. To implement this control policy, enabling arcs En(p_2, t_7), En(p_4, t_7), En(p_5, t_7), and En(p_7, t_7) are connected from p_2, p_4, p_5 and p_7 to t_7, respectively. However, since there is an input arc connecting p_9 to t_7, there is no need to connect a superfluous enabling arc, En(p_9,t_7), from the same place. t_7' must be enabled only at M_{15} and it must be blocked at other markings. To implement this control policy, enabling arcs En(p_1, t_7'), En(p_4, t_7'), En(p_5, t_7'), and En(p_7, t_7') are connected from p_1, p_4, p_5 and p_7 to t_7', respectively. However, since there is an input arc connecting p_9 to t_7', there is no need to connect a superfluous enabling arc, En(p_9,t_7'), from the same place. As a result, the controlled model (the supervisor) is obtained, as shown in Fig. 10.

Note that the controlled model (the supervisor) obtained does not contradict the forbidden state specifications, i.e. controlled behaviour of the system is *nonblocking*. All events that do not contradict the forbidden state specifications are allowed to happen, i.e. the controlled behaviour of the system is *maximally permissive* within the specifications. It is also important to point out that the controlled model (the supervisor) obtained is correct by construction.

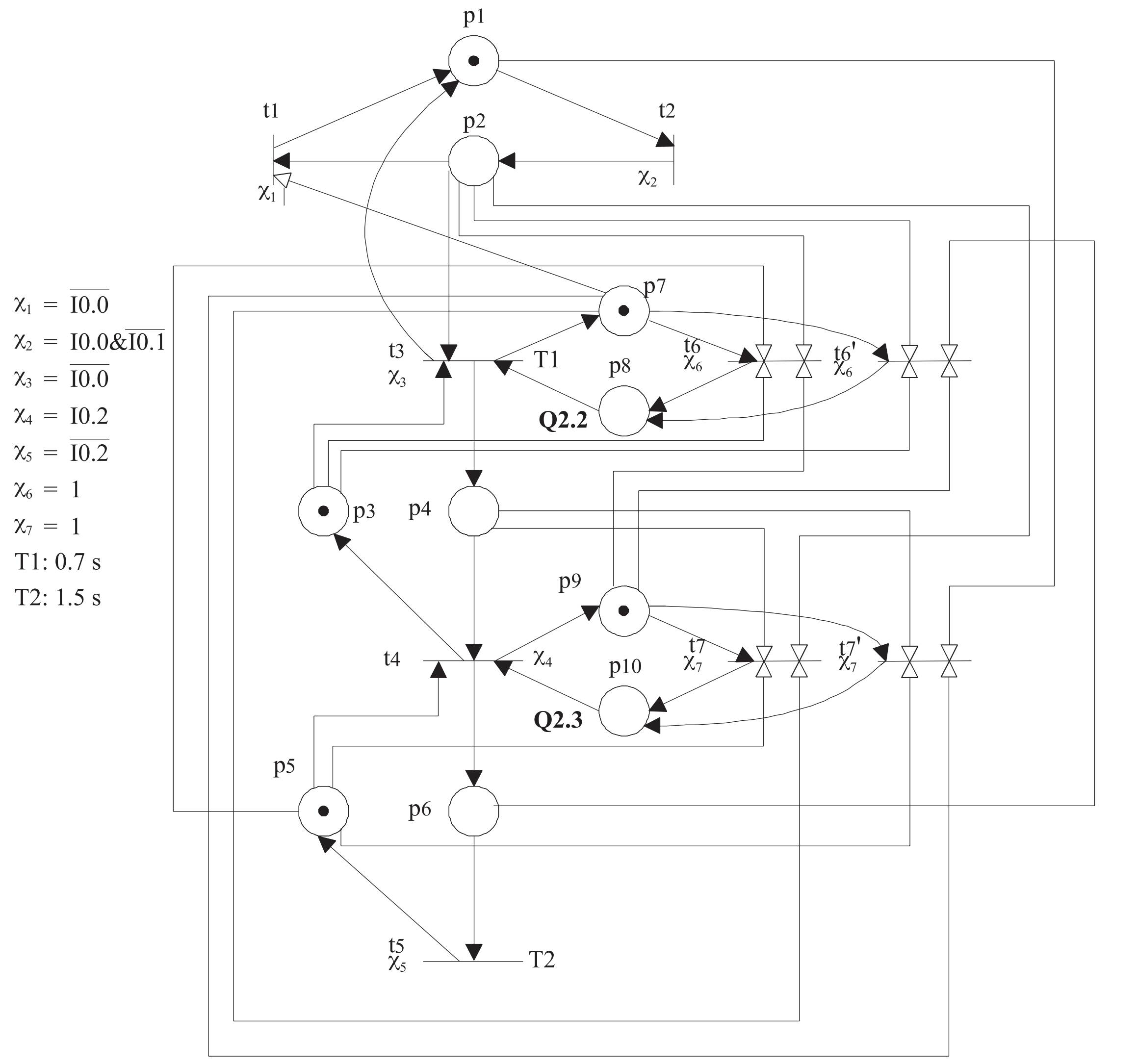


Figure 10. The supervisor (controlled APN model), obtained for the experimental discrete manufacturing system.

Step 4 - Implement the supervisor (the controlled model) on a PLC as LLDs

To convert the supervisor, shown in Fig. 10, into a TPLC, flags F0.1, F0.2, F0.3, F0.4, F0.5, F0.6, F0.7, F1.0, F1.1, F1.2 are assigned to the places P = {p_1, p_2, ..., p_{10}} of the supervisor, respectively. The on delay timers T1 with 0.7 s time delay and T2 with 1.5 s time delay are assigned to the timed-transitions t_3 and t_5, respectively. After the TPLC is obtained as shown in Fig. 11, it is then converted into the LLD code, as shown in Fig. 12, by using a direct mapping from TPLC to LLD. This code was written for a Siemens S5-100U PLC. The LLD symbols for Siemens S5-100U are defined in Table 5. The LLD code is structured in such a way that the rung 0 initialises the system by means of the initialisation flag F0. The rungs 1, 2, 3, ..., 11 represent the transitions T = {t_1, t_2, t_3, t_4, t_5, t_6, t_6', t_7, t_7'}. Then, action places p_8 and p_{10} are represented by rungs 12 and 13 respectively. Finally, the assumption that said "when the system is switched on the upper conveyor motor (action Q2.0) and the lower conveyor motor (action Q2.1) must be in operation", is realised by the final rung 14. By adopting this concept, further clarity can be added to the system documentation and it is very easy to understand and modify the LLD code if necessary. By using a PC-based package called 'Quadriga', this LLD code was programmed on a Siemens S5-100U PLC in the experimental setup shown in Fig. 4. The LLD code, representing the controlled-model, i.e. the supervisor, implemented the forbidden state specifications as required and it did not unnecessarily constrain the behaviour of the experimental discrete manufacturing system.

Table 5. The LLD symbols for a Siemens S5-100U PLC.

LLD Symbol	Definition
S	Set
R	Reset
T	Timer
I	Input
Q	Output
F	Flag
SR	On delay timer
—-] [—-	Normally open contact
—-]/[—-	Normally closed contact

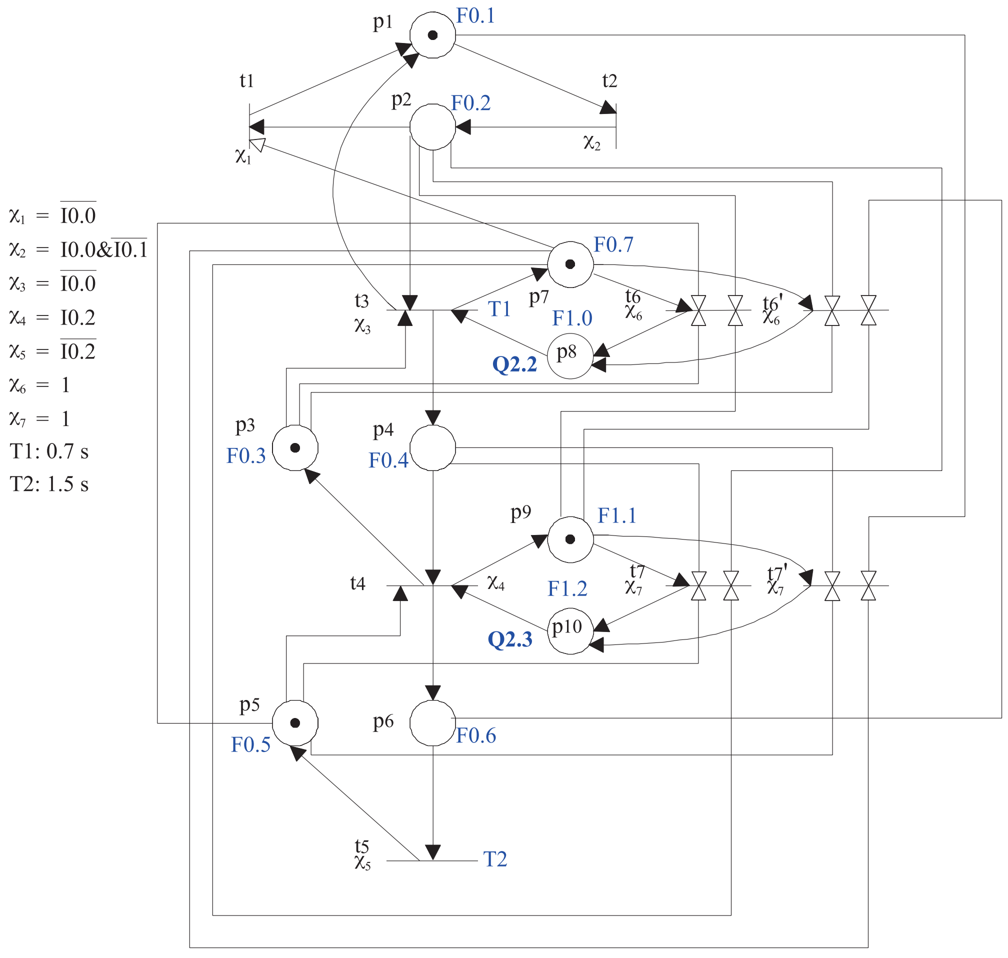
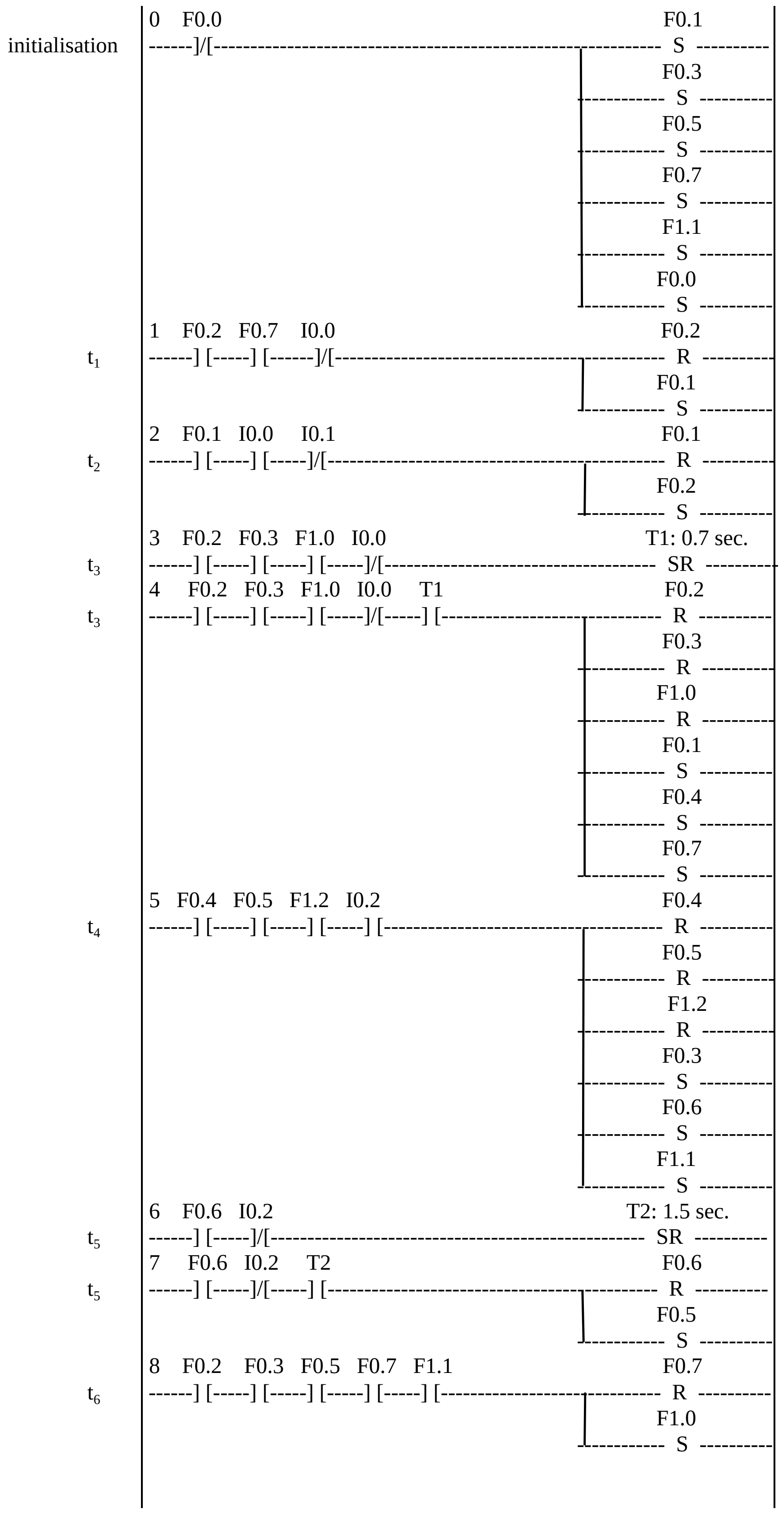
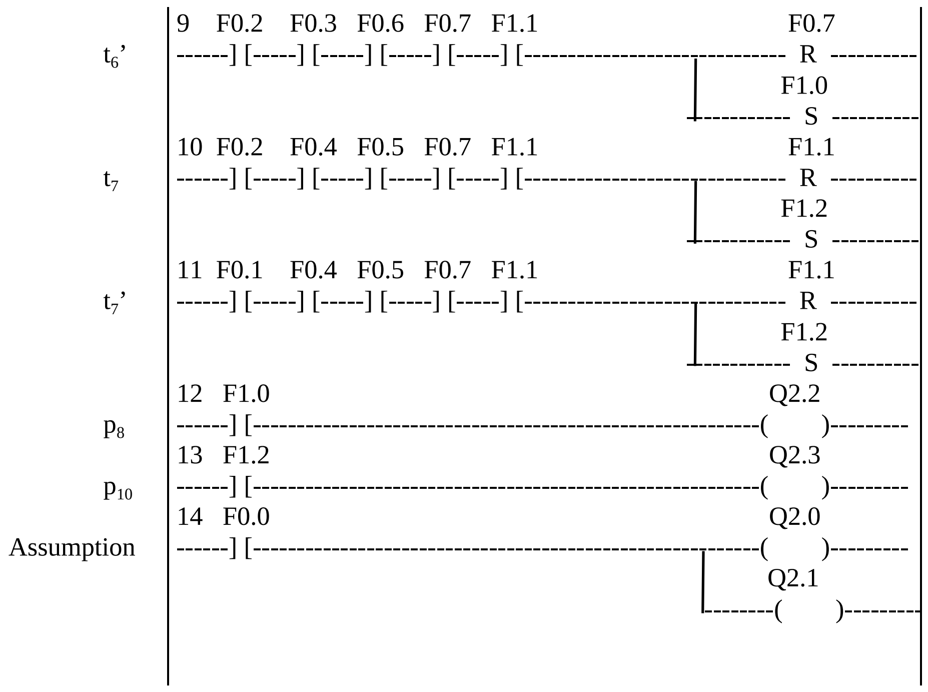


Figure 11. The TPLC for the supervisor shown in Fig. 10.



```
               0   F0.0                                                    F0.1
initialisation ------]/[------------------------------------------------- S ----------
                                                                           F0.3
                                                              ----------- S ----------
                                                                           F0.5
                                                              ----------- S ----------
                                                                           F0.7
                                                              ----------- S ----------
                                                                           F1.1
                                                              ----------- S ----------
                                                                           F0.0
                                                              ----------- S ----------
               1   F0.2  F0.7   I0.0                                       F0.2
t1             ------] [-----] [------]/[-------------------------------- R ----------
                                                                           F0.1
                                                              ----------- S ----------
               2   F0.1  I0.0   I0.1                                       F0.1
t2             ------] [-----] [-----]/[--------------------------------- R ----------
                                                                           F0.2
                                                              ----------- S ----------
               3   F0.2  F0.3  F1.0  I0.0                           T1: 0.7 sec.
t3             ------] [-----] [-----] [-----]/[--------------------- SR ----------
               4    F0.2  F0.3  F1.0  I0.0    T1                            F0.2
t3             ------] [-----] [-----] [-----]/[-----] [-------------- R ----------
                                                                           F0.3
                                                              ----------- R ----------
                                                                           F1.0
                                                              ----------- R ----------
                                                                           F0.1
                                                              ----------- S ----------
                                                                           F0.4
                                                              ----------- S ----------
                                                                           F0.7
                                                              ----------- S ----------
               5  F0.4  F0.5  F1.2  I0.2                                   F0.4
t4             ------] [-----] [-----] [-----] [---------------------- R ----------
                                                                           F0.5
                                                              ----------- R ----------
                                                                           F1.2
                                                              ----------- R ----------
                                                                           F0.3
                                                              ----------- S ----------
                                                                           F0.6
                                                              ----------- S ----------
                                                                           F1.1
                                                              ----------- S ----------
               6   F0.6  I0.2                                       T2: 1.5 sec.
t5             ------] [-----]/[---------------------------------------- SR ----------
               7    F0.6  I0.2    T2                                       F0.6
t5             ------] [-----]/[-----] [------------------------------- R ----------
                                                                           F0.5
                                                              ----------- S ----------
               8   F0.2   F0.3  F0.5  F0.7  F1.1                           F0.7
t6             ------] [-----] [-----] [-----] [-----] [-------------- R ----------
                                                                           F1.0
                                                              ----------- S ----------
```



```
             9   F0.2   F0.3  F0.6  F0.7  F1.1                              F0.7
t6'          ------] [-----] [-----] [-----] [-----] [--------------------------------- R ----------
                                                                    |          F1.0
                                                                    +----------- S ----------
             10  F0.2   F0.4  F0.5  F0.7  F1.1                              F1.1
t7           ------] [-----] [-----] [-----] [-----] [--------------------------------- R ----------
                                                                    |          F1.2
                                                                    +----------- S ----------
             11  F0.1   F0.4  F0.5  F0.7  F1.1                              F1.1
t7'          ------] [-----] [-----] [-----] [-----] [--------------------------------- R ----------
                                                                    |          F1.2
                                                                    +----------- S ----------
             12  F1.0                                                     Q2.2
p8           ------] [--------------------------------------------------------(      )---------
             13  F1.2                                                     Q2.3
p10          ------] [--------------------------------------------------------(      )---------
             14  F0.0                                                     Q2.0
Assumption   ------] [--------------------------------------------------------(      )---------
                                                                    |     Q2.1
                                                                    +--------(      )---------
```

Figure 12. The LLD code obtained from the TPLC, shown in Fig. 11.

6. Conclusions

In this paper a new easy-to-use-and-apply and yet powerful top-down design technique for the synthesis of Petri-net-based compiled supervisors for Discrete Event Systems (DESs) has been proposed to solve the forbidden state problem. The technique proposed involves the construction of the RG of the uncontrolled Automation Petri Net (APN) model of the system. It shows how Petri-net-based compiled supervisors can be constructed by using the models of the systems in a systematic way as opposed to heuristic methods. The methodology offers the following advantages: The compiled supervisor and control policy obtained are correct by construction, i.e. controlled behaviour of the system is *nonblocking* and does not contradict with the forbidden state specifications. All events that do not contradict the forbidden state specifications are allowed to happen, i.e. the controlled behaviour of the system is *maximally permissive* within the specifications.

The results obtained can be applied to systems that require untimed or timed, safe APNs, i.e. an APN model in which a place can have only one token at most, as well as APN models that can accommodate more than one token in a place. Ladder logic diagram (LLD) code has been used to implement the supervisor obtained. Moreover, the paper has particularly shown the applicability of the results proposed to low-level DES control. This has been done by considering an experimental discrete manufacturing system in detail. Note that the results obtained are based on the assumption that the DESs considered are *controllable* in the sense that there is a sufficient number of discrete event actuators, motors, etc. available, and *observable* in the sense that there is a sufficient number of discrete event sensors available within the system.

Although the methodology proposed in this paper ensures that the size (the number of states) of the supervisor to be synthesised does not become bigger as the DES becomes bigger, the computational effort still becomes bigger as the DES becomes bigger. Therefore, this problem remains to be addressed. In fact, in the literature there are some analysis techniques proposed, such as compression techniques, reduced state space construction, and alternative state space construction, to avoid the construction of the complete state space of a given model. A discussion about the use of these techniques was given in [22]. These techniques may be used within the framework proposed in this paper. Therefore, future research will be carried out on the usability of these techniques in the framework proposed in this paper in order to reduce the necessary computational effort to obtain a supervisor for a DES control problem.

References

[1] P.J. Ramadge and W.M. Wonham, "*The Control of Discrete Event Systems*", Proc. of IEEE, vol. 77, no. 5, pp. 81 - 98, 1989.

[2] P.J. Ramadge and W.M. Wonham, "*Modular Feedback Logic for Discrete Event Systems*", SIAM Journal of Control and Optimization, vol. 25, no. 5, pp. 1202 - 1218, September 1987.

[3] P.J. Ramadge and W.M. Wonham, "*Supervisory Control of Discrete Event Processes*", SIAM Journal of Control and Optimization, vol. 25, no. 1, pp. 206 - 230, January 1987.

[4] W.M. Wonham and P.J. Ramadge, "*On the Supremal Controllable Sublanguage of a Given Language*", SIAM Journal of Control and Optimization, vol. 25, no. 3, pp. 637 - 659, May 1987.

[5] A. Guia, "*Petri Net Techniques for Supervisory Control of Discrete Event Systems*", Proc. of 1st Int. Workshop on Manufacturing and Petri Nets, Osaka - JAPAN, pp. 1 - 21, June 1996.

[6] L.E. Holloway and B.H. Krogh, "*Synthesis of Feedback Control Logic for a Class of Controlled Petri Nets*", IEEE Trans. on Aut. Cont., vol. 35, no. 5, pp. 514-523, May 1990.

[7] B.H. Krogh and L.E. Holloway, "*Synthesis of Feedback Control Logic for Discrete Manufacturing Systems*", Automatica, vol. 27, no. 4, pp. 641-651, 1991.

[8] A. Guia and F. DiCesare, "*Supervisory Design Using Petri Nets*", Proc. of the 30th Conf. on Decision and Control, pp. 92 - 97, Brighton-England, December 1991.

[9] R.S. Sreenivas, "*On Asymptotically Efficient Solutions for a Class of Supervisory Control Problems*", Proc. of the 1994 IEEE Int. Conf. on Systems, Man and Cybernetics, San Antonio, Texas, October 1994.

[10] R.S. Sreenivas, "*On the Implications of Solvability of the Supervisory Control Problem for Certain Infinite State Discrete Event Dynamic Systems",* Proc. of the 28th Annual Conf. in Information Sciences and Systems, Princeton Univ., New Jersey, March 1994.

[11] R.S. Sreenivas, "*On Asymptotically Efficient Solutions for a Class of Supervisory Control Problems*", IEEE Trans. on Automatic Control, vol. 41, no. 12, pp. 1736-1750, Dec. 1996.

[12] L.E. Holloway, X. Guan, L. Zhang, "*A Generalization of State Avoidence Policies for Controlled Petri Nets*", IEEE Transactions on Automatic Control, vol. 41, no. 6, pp. 804 - 816, June 1996.

[13] M. Uzam, "Petri-Net-Based Supervisory Control of Discrete Event Systems and Their Ladder Logic Diagram Implementations", PhD Thesis, University of Salford, Salford, UK, 1998.

[14] M. Uzam and A.H. Jones, "*A Methodology for the Synthesis of Petri-Net-Based Supervisory Controllers for Manufacturing Systems: Part I - The Synthesis Procedure*", Proceedings of the 2nd International Symposium on Intelligent Manufacturing Systems, IMS'98, Sakarya University, Sakarya, Turkey, vol. II, pp. 1013 - 1024, August 1998.

[15] M. Uzam and A.H. Jones, "*A Methodology for the Synthesis of Petri-Net-Based Supervisory Controllers for Manufacturing Systems: Part II - An Application*", Proceedings of the 2^{nd} International Symposium on Intelligent Manufacturing Systems, IMS'98, Sakarya University, Sakarya, Turkey, vol. I, pp. 535 - 546, August 1998.

[16] A.H. Jones and M. Uzam, "*A Formal Technique for the Synthesis of Petri Net Supervisors for Discrete Event Systems*", Proceedings of the International Conference on Control - CONTROL'98, University of Wales, Swansea, UK, pp. 845 - 852, September 1 - 4, 1998.

[17] M. Uzam, "*The Enabling Arc Method for the Synthesis of Petri-Net-Based Supervisors for Discrete Event Systems*", Niğde Üniversitesi, Mühendislik-Mimarlık Fakültesi, Mühendislik Bilimleri Dergisi, cilt 2, sayı 1, sayfa 53 - 68, Temmuz 1998.

[18] M. Uzam, A.H. Jones and İ. Yücel, "*Using a Petri-Net-Based Approach for the Real-Time Supervisory Control of an Experimental Manufacturing System*", The International Journal of Advanced Manufacturing Technology, vol. 16, no. 7, pp. 498 - 515, 2000.

[19] M. Uzam, A.H. Jones and İ. Yücel, "*A Rule Based Methodology for Supervisory Control of Discrete Event Systems Modelled as Automation Petri Nets*", The International Journal of Intelligent Control and Systems, vol. 3, no. 3, pp. 297 - 326, 1999.

[20] A.H Jones, M. Uzam, A. H. Khan, D. Karimzadgan, S.B. Kenway, "*A General Methodology for Converting Petri Nets Into Ladder Logic: The TPLL Methodology*", Proc. of CIMAT'96, pp. 357-362, Grenoble, France, May 29-31, 1996.

[21] M. Uzam and A.H. Jones, "*Design of a Discrete Event Control System for a Manufacturing System Using Token Passing Ladder Logic*", Proc., IMACS Multiconference, CESA'96, pp. 513 - 518, Lille, France, July 9-12, 1996.

[22] M. Heiner, Verification and Optimization of Control Programs by Petri Nets without State Explosion, 2^{nd} *Int. Workshop on Manufacturing and Petri Nets; A Workshop within the* 28^{th} *Int. Conf. on Applications and Theory of Petri Nets*, Tolouse, France, (23 June 1997).