Chapter 26

Flexible Manufacturing Systems

26.1 A brief overview of the domain

A manufacturing system involves the manufacturing activity which, as defined in [44], is "the transformation process by which raw material, labour, energy and equipment are brought together to produce high-quality goods". A manufacturing system is composed of two main subsystems:

- *The physical subsystem*, composed of the physical resources (hardware components) such as conveyors, robots, buffers, work stations, etc.
- *The control subsystem*, also called the Decision Making Subsystem (DMS) [35], which establishes how to use the physical subsystem in order to organise and optimise the production process.

Usually, manufacturing transformation processes are classified into continuous (chemical and oil industries, for instance) and discrete (consumer goods and computer industries, for instance). According to the type of transformations to be carried out during the manufacturing process, the discrete manufacturing systems are classified into assembly and non-assembly processing. The assembly processes combine several components to obtain a different product, while the non-assembly processes concern the transformation (machining, moulding, painting, etc.) of raw materials.

In order to face some problems related to mass manufacturing systems (very efficient for a large production of a small number of types of products, but inflexible in order to be adapted to a changing market), and in parallel with the developments in computer and automation technologies, a new type of production system appeared: the Flexible Manufacturing Systems (FMS). Taking the definition in [29], an FMS can be defined as "a computer-controlled configuration of semi-independent work stations and a material handling system

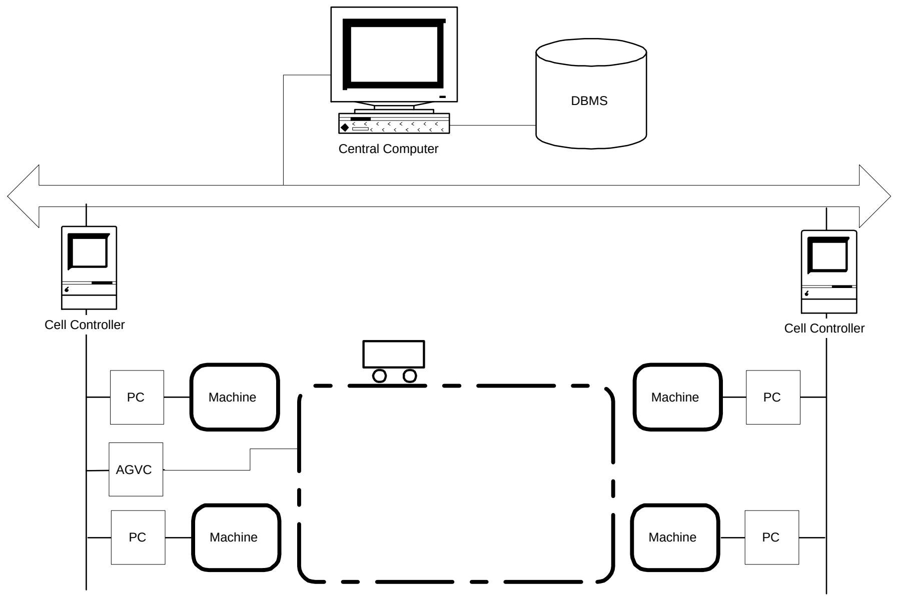

Figure 26.1: An abstract view of a Flexible Manufacturing System

(MHS) designed to efficiently manufacture more than one part type from low to medium volumes". The adjective "flexible" stands for the ability of the system to respond to changes in the system in an effective manner. These changes can be internal, breakdowns or quality problems for instance, or external, changes in the design and demand for instance. In [4] eight different types of flexibility are summarised: machine flexibility (which refers to the time required to change the machines necessary to produce a new type of part), process flexibility (related to the mixture of jobs that the system can produce simultaneously), product flexibility (the ability to produce new types of products), routing flexibility (the possibility of the system to route parts via several routes), volume flexibility (capacity of the system to operate at different production volumes), expansion flexibility (the capacity to expand the system in a modular way), operation flexibility (the ability to interchange the ordering of several operations for each part type) and production flexibility (the set of part types that the system can produce).

Figure 26.1 depicts a typical plant of an FMS[44]. The global coordinating system communicates, via a local area network, with the controllers of each cell. Each one of these cell controllers is in charge of the control of the programmable controllers (PC) that are in charge of the control of each one of the physical hardware components in the cell. As it will be detailed later, the complexity of these systems makes the hierarchical organisation of the control system necessary.

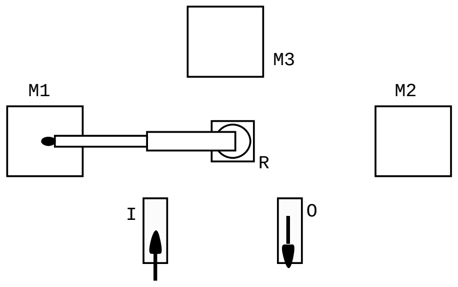

Figure 26.2: A small manufacturing cell.

FMS hardware components are typically a set of work stations, an automated material handling system (conveyors, industrial robots, automated guided vehicles, etc.) allowing a flexible routing of parts through the different work stations, a load/unload station for the entry/exit of parts, some storage means for the work-in-process parts storage, some (local and central) tool magazines and a computer control system that is usually organised in a hierarchical way.

With the aim of introducing these systems in a more detailed way, let us present, in an intuitive and informal way, a small FMS.

Consider the manufacturing cell whose physical layout is depicted in Figure 26.2. The cell is composed of three machines, $M1, M2, M3$ and a robot R, whose role is to load an unload the machines. The robot can also pick up parts from conveyor I, where parts arrive in the system, and unload parts into conveyor O, where the parts processed in the cell are unloaded. Let us assume that the flexible machines can carry out different operations on the incoming parts. Let us also assume that $M1$ can process three parts at a time, while machines $M2$ and $M3$ can process only two parts at a time. In the sequel, we call the elements composing the cell (machines, stores, robots, buffers, etc.) "resources".

Finally, we consider that in this cell two different types of parts must be processed. Parts of type one must be processed first either in machine $M1$ or $M3$ and then in machine $M2$; parts of type two must be processed first in $M2$ and then in $M1$ (at this moment, we are not considering what kind of processing operation must be carried out in each machine and for each type of part).

This system exhibits some important characteristics that are common to almost all the FMS [47]:

- *It is event-driven*: the system behaviour consists of a discrete state space where a change in the state occurs when some events are triggered (a new part enters or leaves the cell, a machine loads a part, etc.)

- *It is asynchronous.* Some events in the system occur in an asynchronous way: the end of the processing of a part in machine $M1$ is asynchronous (in time) with respect to the loading of a new part in machine $M2$.

- *It has sequential relations.* Some events must occur in a sequential way. So that a part can be unloaded from machine $M1$, this machine must have been previously loaded and the processing of the part be finished.
- *It has concurrency.* The processing of a part in $M1$ and a second part in $M2$ can be done in a concurrent way, and these two actions do not interact with each other.
- *It has conflicts.* A part of type one that has been held by the robot can be loaded either into $M1$ or into $M3$ (assuming both machines have free slots to load new parts). So, a decision must be taken.
- *It has non-determinism.* As a consequence of conflicts, some non-determinism can appear. In the previous situation, we cannot, "a priori", predict which action will be taken: either the part is loaded into $M1$ or into $M3$.
- *It has deadlocks.* In the case in which all three machines are fully busy and the robot holds a raw part that must be loaded into one of the machines, the system is in a (total) deadlock situation: no action can be executed since no machine can be unloaded (the robot is busy) and the robot cannot release the part (since it has to go to a machine).
- *Mutual exclusions.* Let us consider the processes corresponding to the processing of a part of type one and a part of type two. Both processes cannot stay simultaneously in the state "the part is being held by the robot". So, this state implies a mutual exclusion for these two processes.

We can conclude that the design of manufacturing systems is a very complex task: many different elements have to be combined, and many different aspects must be taken into account. This complexity has raised two important needs: **1)** The design of the production control system in a hierarchical way. **2)** The use of formal methods in order to validate the system.

As summarised in [35], the DMS is usually decomposed into the following levels:

- *Planning.* It considers both the whole plant and the estimated demand. It considers the production on a long time horizon, establishing how the amount of needed products will be produced along this time interval.
- *Scheduling.* Going down in the DMS hierarchy, this level establishes when each operation on each product must be carried out.
- *Global coordination.* This level must have an updated state of the workshop and must also take real-time decisions taking into consideration the state of each resource and the state of the parts being processed.
- *Sub-system coordination.* The global coordination system can be decomposed into modules specialised for the coordination and supervising of sub-systems: a transport system, a robot, a buffer, etc.

- *Local control.* It is the lower level of the hierarchy, and it is in charge of the interaction with sensors and other low level hard components.

The second important need was the use of formal methods. As stated in [24], the use of a formal framework entails some important benefits: 1) In the process of formalising the system requirements some omissions, ambiguities and contradictions can be discovered. 2) A formal method can allow automatic system development. 3) Mathematical methods can be applied to verify system correctness. 4) A formally verified subsystem can be incorporated into larger systems with greater confidence. 5) Different designs for the same system can be compared.

However, the use of different formalisms (e.g., Markov chains, queueing networks or simulation for performance evaluations, mathematical programming for planning, Petri nets for modelling and analysis) for the different problems to solve generates a “Babel Tower” where the communication among the people working at different stages in the design process is a very difficult task [34].

As proposed in [34], a good solution is to use a *family of formalisms* which, sharing the basic principles, allows the transformation (in an automatic way if possible) from one formalism into another. The family of Petri net formalisms is a good issue in the manufacturing system environments. This family has the following advantages [35, 47]: 1) Easy representation of concurrency, resource sharing, conflicts, mutual exclusions, and non-determinism. 2) Application of Top-down and Bottom-up design methodologies, and the possibility of having different levels of abstraction of the system. 3) Ability to generate control code directly from the Petri net model. 4) A well-defined semantics that allows qualitative and quantitative analysis for the system validation. 5) A graphical interface that allows an intuitive view of the system.

The use of Petri nets in manufacturing systems has been widely dealt with in research (see [35, 34] for a wide set of references) and application literature, and many text books concerned with this domain have appeared in the last few years [44, 9, 47, 8, 30]. Petri nets have been used in all aspects of the designing and operation of FMS: modelling and verification, performance analysis, scheduling, control and monitoring.

The present chapter studies some problems related to the designing and control of discrete non-assembly FMS by means of the use of Petri nets as a family of formal models. Here, we focus on a class of problems that arise at the global coordination level.

The chapter is divided into three main sections. The first one shows how some Petri net elements (tokens, places, transitions and arcs) can be mapped into FMS concepts. The second one deals with the problem of system modelling. This is not a very simple task. Computer aided design tools are interesting in order to get models as well-structured as possible. The section presents a modelling methodology for a wide class of systems. This modelling methodology relies on a clear differentiation between the model of the system layout and the models of the types of parts to be produced. From these inputs, and in an automatic way, a Coloured Petri net can be obtained. The section shows what the input data models are like and it also explains the process from its first step until the final model is obtained.

As stated previously, one of the advantages of formal models is that some system properties can be studied in the model. The second part of the chapter is devoted to show how the structure of the Petri net model can be used in order to deal with one of the main problems in automated manufacturing systems, the deadlock problem. First, it is shown what the place/transition models corresponding to the coloured models obtained by the modelling methodology are like. Secondly, it is shown how deadlock problems can be characterised in terms of some Petri net structural elements called siphons (also called structural deadlocks in the literature). The structural deadlock characterisation is used in order to get a control policy for deadlock prevention, and this control policy is also implemented by means of Petri net elements (addition of some new arcs and places to the former model).

Throughout the chapter the same "toy example" will be used. More interesting and complex models can be found in the literature. The aim of this chapter is to show the use of Petri nets (ordinary and Coloured Petri nets) for modelling and analysing of flexible manufacturing systems. In this sense, this chapter is not a survey of all the different approaches that have been adopted for the use of Petri nets in the domain; it just presents one of them. In [34] a complete set of references related to this subject can be found. The chapter only treats qualitative aspects of the domain. For quantitative aspects, the reader is referred to [44, 9, 8].

26.2 Using Petri nets in FMS

In order to get an insight of the use of Petri nets in the considered domain, we are going to present a set of models corresponding to some basic components of FMS, such as machines and buffers or stores.

- Figure 26.3 depicts the abstract model of three different transportation systems. In the three cases the interactions with the rest of the system are represented by means of the transitions tI and tO, which model the loading and unloading of parts in the module.

  Figure 26.3-a is the model for a buffer (also a store) with capacity for k parts. Notice that if we take $k = 1$ this PN can also model a robot, for instance. Figure 26.3-b is used to model a FIFO queue with capacity for three parts: there are three positions that are accessed sequentially in an ordered way. Finally, Figure 26.3-c models a LIFO module. This module represent the set of states that can be reached, but not the firing sequences. Notice that nothing is forbidding the sequence $(t_{12}t_{21})^*$ which, of course, must not be allowed. In all the examples we are introducing, nothing is said about the control; we are just concentrating on the modelling of the structure of the component. In all theses cases we are assuming that the time necessary for the execution of the operations related to each transition is negligible.

- Figure 26.4-a shows a model for a reliable machine (no breakdown is considered). When the part is loaded into the machine (transition tLM is fired) the processing starts and, once the machining has finished, a token is put in place pAP and then, transition tUM can be fired. Notice that in this model two different types of transitions appear. Thick black transitions represent "immediate" actions (here, immediate transitions model system actions whose time execution is negligible); square white transitions model system actions whose execution time can be modelled by means of a probability distribution function. Usually, this function is taken to be an exponential, and the λ parameter which appears near the transition is the firing rate (then, $1/\lambda$ is the mean time needed for the processing of the part).

  Figure 26.4-b shows the model of the same machine, but once the possibility of a breakdown has been considered. In this model, in order to load or unload a part it is necessary that the machine be in the OK state (arcs joining tLM and OK place and also tUM and OK place). The machine breaks-down with a rate λ_f and is repaired with a rate λ_r.

- Finally, Figure 26.5 models an unreliable assembly machine. Notice that in order to start the assembling, it is necessary to have loaded a part into $pT1$ and another one into $pT2$. Here λ is the rate of time needed to carry out the assembling. The model for a disassembly machine is almost the same: it is enough to reverse the arcs related to transitions $tL1$, $tL2$, tA and tU.

As we have seen, when using ordinary Petri nets for the modelling of flexible manufacturing systems, the main Petri net elements (places, transitions, arcs and tokens) can have different meanings:

- A place can be used for the modelling of different elements. 1) *States in which a part that is being processed can stay.* Let us consider, for instance, place pBP in Figure 26.4-a. It represents a part of a given type that has been loaded into the machine and that is being processed there, while place pAP is used to model the state of that part in the same machine, but whose processing has already been finished and is ready to be unloaded from the machine. 2) *A partial state of a resource.* Place kM in Figure 26.4-a models the free state of the machine; in this sense, this place does not contain "physical" things, but it is used for a rather "logical" interpretation.

- A transition usually models a sequence of system actions that change the state of some system elements. For instance, transition tA in Figure 26.5 is modelling the sequence of system actions by means of which the parts modelled by the tokens in places $pT1$ and $pT2$ are assembled in order

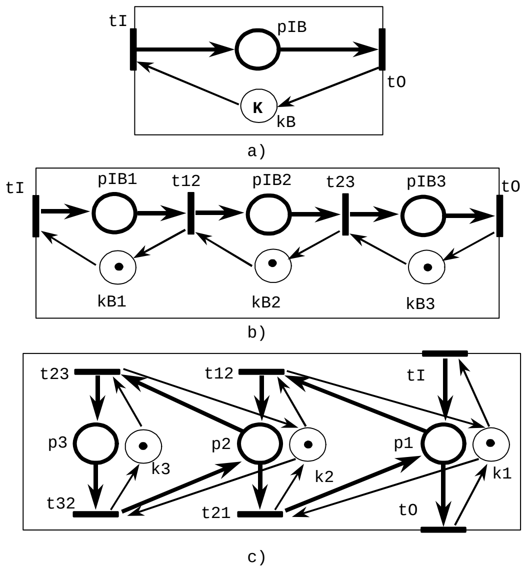

Figure 26.3: a) A generic model for a storage device b) A model for a FIFO device c) A model for a LIFO device

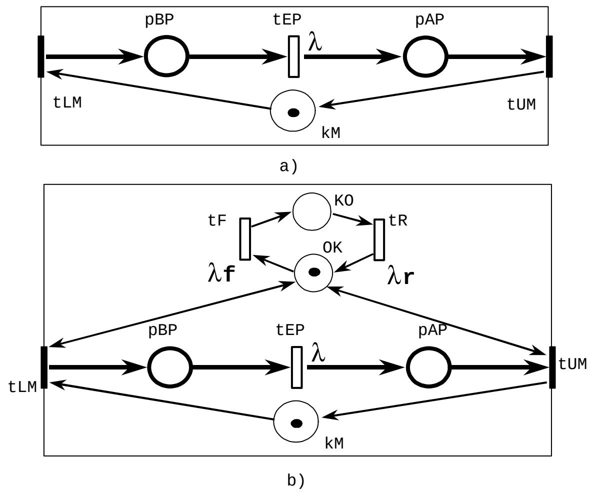

Figure 26.4: a) A model for a reliable machine b) A model for an unreliable machine

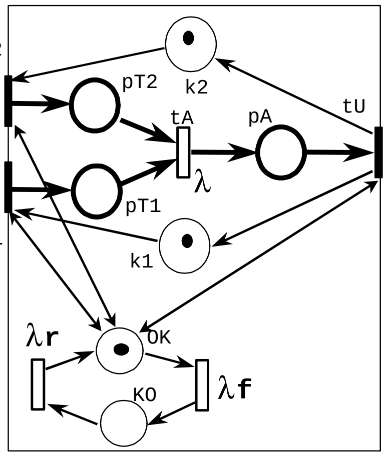

Figure 26.5: A model for an unreliable assembly machine

to produce a new product (modelled by means of the new token that is put into place *pA*). Another type of actions that are usually modelled by means of the firing of a transition is the movement of a part between two different locations in the system (for instance, transition *t*12 in Figure 26.3). Also, a transition can model the change in the state of a system resource, as it is the case of transition *tF* in Figure 26.4-b: it models a break-down of the machine modelled in the figure.

- Usually, an arc models either a precondition or the flow of parts among resources. The arc joining transition *kM* and transition *tLM* in Figure 26.4-a is an example of the first kind of arcs. It models the necessity of having a free position in the machine in order to load a new part. Arcs from *pT*1 and *pT*2 to transition *tA* in Figure 26.5 fall into the second class. They model two elements that are withdrawn from the two buffers. Also the arc joining *pA* and transition *tU* in the same figure belongs to the second class. It models the flow of an assembled element to the output of the assembly machine.
- Tokens can also have different meanings. In the case of Figure 26.4-a, the token in place *kM* models the availability of the machine (the machine is non-busy), when a token in place *pIB* in Figure 26.3 represents a product that is stored in the buffer. In the case of Coloured Petri nets, a token can carry a lot of information, as it will be shown later on.

As stated previously, one of the main problems when dealing with real applications is the complexity of the model. From the design point of view, different approaches have been adopted:

- Hierarchical/compositional approach: The idea behind these approaches is the modelling of the systems in an structured way. Using the first approach (also called top-down) the modelling is carried out in several steps. At each step more detailed elements are considered. In general, the process consists in the replacement of some net elements (place, transition, path, subnet) by some subnet in which the replaced elements have been refined [40, 37, 46].

When using the compositional approach (bottom-up) the global model is obtained by means of simpler models that are combined by means of some composition mechanism: fusion of places common to a set of submodules (and modelling the same elements), synchronisation of a set of transitions and fusion of common paths [1, 28, 25, 39, 3, 15] and also Section **??**.

Even if the two approaches below help in the design process, both present one important drawback: it is very difficult to ensure that in the modelling process (either compositionally or in a hierarchical way) some desired system properties (such as boundedness, reversibility, deadlock-freeness, liveness, etc.) be preserved from one step to the next one. This means that, for instance, we can have two live modules, whose composition is non-live. The same goes for the case of a hierarchical approach. It can arrive that at a given abstraction level the system behaviour is live, but once a new refinement is given the new "view" of the system is not live.

In order to cope with this problem two different kinds of solutions have been adopted: *1)* The kit of refinement/composition mechanisms is restricted. This means that the composition of modules or the refinement must only be done when some special conditions hold and *2)* The work is restricted to some special subclasses such as free-choice nets [10], marked graphs [27], modules synchronised by means of (restricted) message passing [31] or (restricted) resource sharing [12, 22].

However, in both cases the modelling power is decreased.

- High level Petri nets approach: High level Petri nets, and Coloured Petri nets [23] as a particular case, arise as a very useful tool for modelling complex systems in which different components have analogous behaviour. One of the main advantages of this class of nets is the compactness and the clarity of the models generated [7, 26, 42, 17, 18, 14, 21].

  However, usually, they present the drawback of the difficulty to carry out the analysis of properties.

- Object Oriented (OO) and Artificial Intelligence (AI) approaches: A lot of work related to the use of Petri nets in manufacturing systems has tried to extend the capacity of Petri nets for the modelling of systems with the capacity of AI techniques for the reasoning about properties. Also here different approaches have been adopted. In some papers [16, 5] some elements of AI are used to implement and control the Petri net.

  Other papers, [41, 33, 32, 43], use AI elements to implement the Petri net (tokens or places as frames and transitions as rules, for instance), and use the semantics of the underlying Petri net for simulation and control of the system.

The use of one of these approaches does not exclude the use of another. For instance, we can adopt both a hierarchical approach [20] and a compositional approach [6] in order to obtain a Coloured Petri net model.

But, once we have obtained a Petri net model for the system we want to deal with, what kinds of Petri net properties are interesting for our model? Let us enumerate some important behavioral properties. It is important to notice that some of the following properties are related: one property can be deduced from others.

- *Reachability.* From the model point of view, this property establishes if a given (vector) marking is reachable from the initial marking. From the real system point of view, this property is able to inform if a system state is reachable from the initial configuration. This property can be used to answer questions of the following types. Is it possible to reach a state where machine M is processing two parts while robot R is busy and machine M' is free? Is it possible to reach a state in which buffer B is full? The answers to a set of well-defined questions can be used in order to establish a correct system design. Notice that if, for instance, the answer to the last question is NO, the designer can decide whether to use a buffer with less capacity, which can indeed make the system less expensive. A second related property is *coverability.* From the Petri net point of view, it establishes if a marking is reachable so that it is greater or equal to another given marking. From this kind of property, more partial information can be obtained; but this information can be used in an analogous way as is the case of reachability properties.

- *Boundedness.* This property is able to establish if the number of tokens in a given place is always smaller or equal to a given constant k (k-boundedness). Usually, in FMS domains, and using the possible meanings of a place as stated previously, all the places must be bounded. So, if in the analysis of the Petri net model we realize that a place is not bounded, the model is, perhaps, incorrect. However, if the model is correct and a place is detected unbounded, some overflow problems may arise. A related property is *safeness* (1-boundedness).

- *Reversibility.* When verified, this property establishes that the initial state can be reached from each reachable state. In the application domain considered this property means that each possible erroneous situation has been considered by means of some error recovery strategy. These erroneous situations include the case of system deadlocks and the case of resource failures.

- *Deadlock freeness/liveness* . These properties will be commented in a more detailed way in section 26.3.3.

As it has already been intuitively shown by means of the models of components in an FMS, Petri nets have also been used for the performance evaluations of FMS. To do that, the notion of time has been added to the Petri net models. Introducing time constrains is necessary if we want to consider performance evaluations or scheduling of real time control problems.

Usually, time has been introduced in one of two different ways: either associated to places or to transitions. The second way is more natural since transitions usually model system activities (which need some time to be executed). In this approach, time is considered as follows: a transition can fire some time after it is enabled w.r.t. the number of tokens in its input places. This time can be either deterministic, *timed Petri nets*, or random, *stochastic Petri nets* (see [44, 9, 8] for a clear introduction to these concepts).

In FMS domains the different quantitative measures that can be obtained from the Petri net model have specific and clear meanings: probability of a resource to be non-busy, mean number of parts in a machine or buffer, mean waiting time of parts in an input buffer, production rates of parts, mean time of parts in in-process states, etc.

In this chapter we are going to concentrate on the qualitative analysis using structural methods.

26.3 A design approach

In this section we are going to present a particular approach to the designing and control of FMS using Petri nets. As stated in Section 26.1 many different approaches have been adopted. The reader is referred to the literature cited in this chapter for a comprehensive study of the different approaches. The presentation of the method is carried out in an intuitive way following a simple example. A formal presentation can be found in [11].

This Section is organised as follows. First, we introduce the place/transition model corresponding to the system in Figure 26.2; after that we present how an equivalent model can be obtained in an automatic way; finally, it is shown how the structural analysis of the Petri net can be used to establish a control policy for deadlock prevention in order to ensure a good behaviour.

26.3.1 An intuitive introduction to a class of nets

Let us consider the model in Figure 26.4-a once again. In the case we are not interested in performance evaluations we can model each action by means of an immediate transition. This change allows us to obtain a simpler model. For instance, let us consider the general model of a reliable machine of the figure. If we apply a reduction rule, the path pBP, tEP, pAP in the figure can be replaced by a unique place, obtaining an equivalent model[1]. This approach will be used

[1] Of course, when talking about some equivalence we must specify what kind of equivalence. Here, as it will be stated later, we are interested in liveness properties. Then, in this case the transformation maintains the liveness of both models: the original one and the transformed

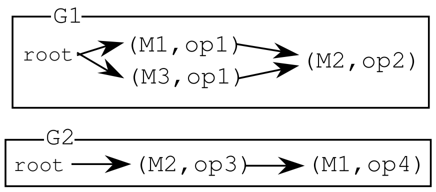

Figure 26.6: The models of process plans for two types of parts to be processed in the system in Figure 26.2.

in the following. And, since only one kind of transitions will be considered, all the transitions will be drawn as white rectangles.

Let us consider the manufacturing cell shown in Figure 26.2 that was described in Section 26.1. Each part belongs to a different part type. The type of the part establishes the correct sequences of operations. In a first step, these sequences are established in terms of transformations to be carried out on parts. Considering the cell, these sequences of operations are transformed into sequences of pairs *(resource,operation)* which establish, for each operation, the resource where the operation has to be carried out. Each part type can be modelled by means of an acyclic graph. Figure 26.6 represents the operation graphs corresponding to two different process plans. Parts of type $W1$ have to be processed first either in machine $M1$ or $M3$, and then in machine $M2$. Parts of type $W2$ have to visit machine $M2$ and then machine $M1$. Since parts must be loaded (unloaded) into (from) the system, each process plan needs more information than the operation graph. So, a process plan must be completed with two sets. The first one represents the system actions that load parts of the corresponding type into the system. The second set represents the system actions that unload parts of the corresponding type from the system. So, in the considered example, we define $W1 = \langle G1, I1, O1 \rangle$ where $I1 = \{fromI\}, O1 = \{toO\}$ and $W2 = \langle G2, I2, O2 \rangle$ where $I2 = \{fromI\}, O2 = \{toO\}$.

Each process plan model has an initial node *root* (as shown in Figure 26.6) that models the raw state of parts. The other nodes correspond to the label of the transformation resources the part can visit during its processing.

From a process point of view, let us show how the processing of parts of type 2 is carried out. The sequence of steps that one part of this type must follow is as follows. The part is held by the robot, loaded into machine $M2$, held once again by the robot, loaded into machine $M1$, held a third time by the robot and finally unloaded from the system. These different states are modelled in figure 26.7-b by means of the thick places. A place $p0(2)$ (called the *idle state place*) has been added in order to introduce a notion of repetitive process, modelling the repetitive nature of the processing of different parts of the same type. The initial marking of this place establishes the maximum number of parts of type 2 that are allowed to be concurrently processed in the system. Notice that if the initial marking is big enough (as in the example considered,

one.

for instance), this idle place becomes implicit **??**, and has no effect in the model behaviour.

The transitions in this figure model the system actions that carry out the state changes of this type of parts. The net belongs to the class of the S^2P in [12] and, essentially, is the same as a *job subnet* in [19]. In any case, it is usually imposed that all the cycles of the S^2P (and analogous classes) contain the idle state place. This implies that no cyclic behaviour is allowed during the processing of a given part: once the processing of a part has started, the part cannot change its state infinitely often without terminating its processing.

Notice that we have one of these nets for each type of part to be processed. How can these nets be obtained? The process is as follows. Let us classify the set of system resources into two classes: those resources that make some transformation on parts, called *processors* (e.g. lathes, milling machines, saws, grinders, etc.) and those which do not transform the parts, called *handlers* (e.g. robots, stores, buffers, conveyors, etc.). Notice that since in the operation graphs only part transformations are established, these nodes are always labelled with processors. Let us concentrate once again on parts of type 2. A part of this type, once loaded into the system, must be driven to $M2$ from one of its corresponding loading actions (established by $I2$). So, we must compute all the possibilities of driving the part from the input to $M2$ using only handlers (the first transformation on this part must be carried out in $M2$). According to the plant layout depicted in Figure 26.2, the only possibility is that the part be held by R and loaded into $M2$. This means that an intermediate state (the part is held by R) is needed, and also the transitions modelling the flow of the part from $fromI$ to R and from R to $M2$. In this way we obtain the path $fromI(2,s)R(2,s)toM2(2,s)M2(2,M2)$ in Figure 26.7-b. Now we must consider the arc $(M2,M1)$ in the operation graph. The part must be driven from $M2$ to $M1$ using only handlers. And the only way of doing that is using R once again. So, the path $fromM2(2,M2)R(2,M2)toM1(2,M2)M1(2,M2)$ is added to the model. From $M2$, the part must be unloaded. So, we must find all the possibilities for the part to be driven to the output of the system using only handlers. And the only possibility is that the part be held a third time by the robot. Notice that this process must be repeated for each processing sequence taken from the operation graph of each part. It is also important to point out that in this process both the system layout and the process plans are involved. It can also happen that some operation sequences established by the operation graph be not executable because of the layout architecture (no path joining two machines M_a and M_b exists, when the arc (M_a, M_b) belongs to some operation graph). This justifies the following definition [11]: a process plan is *executable* for a given architecture if for each arc $(\langle p_1, op_1\rangle, \langle p_2, op_2\rangle)$ in the operations graph there exists at least one path from p_1 to p_2 using only handlers. In the sequel, we will call *state places* the places that are generated during this process, in order to distinguish them from the places that model the resource capacity constraints, called *resources places*, and which will be introduced in the following.

At this level, the system resources that are used in the processing of parts

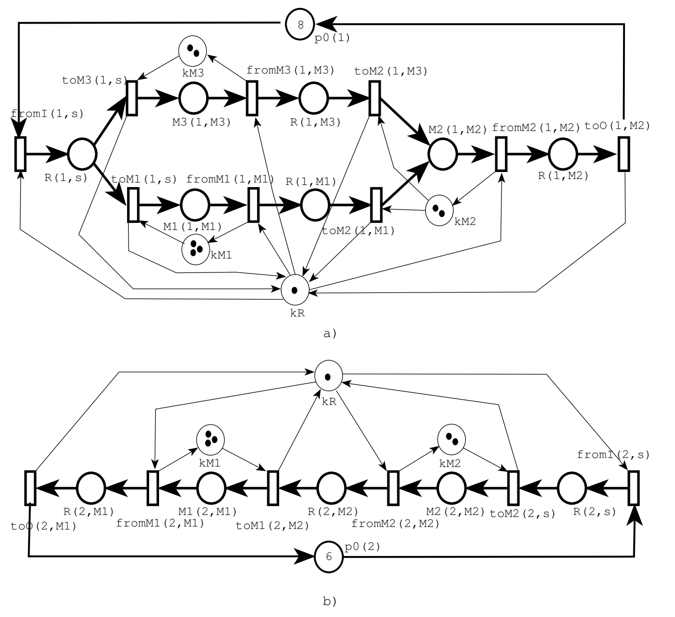

Figure 26.7: The models corresponding to the processing of the two types of parts under consideration.

have not been considered. This means that the constraints the resources impose on the concurrent processing of parts have still not been considered. So, it is necessary to model these constrains. In order to deal with them, a place is added for each system resource: one place for each machine, whose initial marking is equal to the number of parts that the machine can process concurrently, and a place for the robot, whose initial marking is equal to one (we have assumed that the robot can hold one part at a time). The loading of a part into one machine needs at least one of the machine positions to be free (an arc from the resource place to the transition modelling the system actions that load a part into the machine is added). On the contrary, the unloading of a part from one machine increases in one the number of non-busy positions in the machine. So, an arc is added from a transition modelling the unloading of one machine to the resource place. The net in Figure 26.7-b depicts the whole model corresponding to the processing of parts of type 2. In the same way, the net in Figure 26.7-a models the processing of parts of type 1. These two nets belong to a special class of nets, called S^2PR in [12].

And, finally, the interactions among different types of parts must be considered. The complete system model corresponds to the fusion of the places that the models of the two types of parts have in common, i.e., the places modelling the system resources (in the example considered, places $kM1, kM2, kM3, kR$). This is quite natural: the interaction of the processing of different parts is made by means of the system resources since all the parts in the system must compete for the same resources. Figure 26.13 depicts the final model once the composition of the sub-models corresponding to the types of parts has been carried out. This net belongs to a class of nets called S^3PR in [12]. This class is analogous to the notion of *production sequence* in [2] or *Production Petri net* in [19].

26.3.2 Automation of the modelling process

In this section we are going to show that it is possible to adopt a more abstract point of view for the system, and that this point of view allows us to obtain easily the Petri net model presented previously. First, we present how the plant layout can be modelled by means of a place/transition Petri net. Secondly, we consider the models of the process plans as introduced above. Finally, we show that both models can be integrated in order to obtain the complete model. This final model, which can be obtained in an automatic way from the inputs (the model of the plant layout and the models of the process plans), will be a CPN.

As stated previously, from an abstract point of view, the state of a resource can be modelled by means of two places: 1) The "resource capacity place", modelling the remaining capacity of the resource to hold new parts. In the case of multiple copies of identical resources, the marking of this place models the number of copies of the resource that are not engaged in a processing operation. 2) The "resource state place". Each token in this place models a part that is using either the resource or one copy of the resource in the case of multiple copies of identical resources. For instance, consider machine $M1$ in Figure 26.8. This machine is modelled by means of places $M1$ and $kM1$. When considering a state reachable from a given initial state, the tokens in place $M1$ model the parts that are being processed in the machine. The tokens in $kM1$ model the number of parts that can still be loaded into machine $M1$. Notice that the sum of the number of tokens in $M1$ and the number of tokens in $kM1$ must always be equal to three, the capacity of machine $M1$ that we have assumed.

Let us now show how the possibility of part flow among resources is modelled. Let us consider the resource places R and $M1$. Since the physical layout allows the flow of a part from R to $M1$, transition $toM1$ is added between these two resources. Also, since this flow is from R to $M1$, an arc from R to $toM1$ and an arc from $toM1$ to $M1$ are also added. Since the capacity constraints must also be considered, two more arcs are added: the one from $kM1$ to $toM1$ and the one from $tomM1$ to kR. And this must be done for every two resources that are connected in a direct way.

Given the previous considerations, the PN model of the considered cell is depicted in Figure 26.8. Robot R is modelled by means of places R and kR, machine $M1$ by means of places $M1$ and $kM1$, machine $M2$ by means of places $M2$ and $kM2$, and machine $M3$ is modelled by means of places $M3$ and $kM3$ (places whose name starts with "k" are capacity places). In this PN each directed path between places R, $M1$, $M2$, $M3$ not using capacity places $kM1, kM2, kM3, kR$ models a possible path which a part can follow inside the cell. Since machine $M1$ can process three parts at a time, the initial marking must be $\mathbf{m_0}[kM1] = 3$, while for the other machines $\mathbf{m_0}[kM2] = \mathbf{m_0}[kM3] = 2$ and for the robot

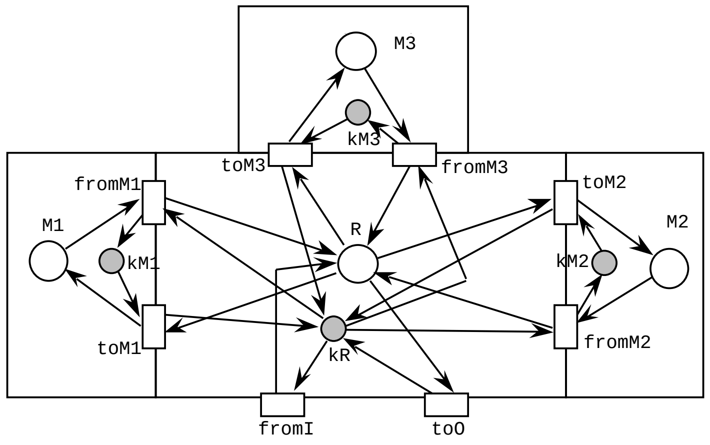

Figure 26.8: PN layout model of the cell in Figure 26.2.

$\mathbf{m_0}[kR] = 1$.

In a next step we need to integrate the model of the cell layout and the models of the process plans in Figure 26.6. The modelling of the state of a part in the system is carried out as follows. Each part in the Petri net is modelled by means of a token. The token has two components; so, it will be modelled by means of a *coloured token*. The first component identifies the part type, i.e., its process plan. The second component identifies the last node of the process plan model the part has visited during its processing. Let us consider, for instance, a raw part of type $W1$, as considered in Figure 26.6. The part is modelled by means of a token $\langle W1, root\rangle$ when it is in the system and no transformation has been carried out on it. When the part has already visited machine $M1$, and not yet machine $M2$, the part is modelled by means of a token $\langle W1, (M1, op1)\rangle$ (since, up to now, $(M1, op1)$ is the last node of the operation graph "visited" by the part). When the part has already been processed in machine $M2$, it is modelled by means of the token $\langle W1, (M2, op2)\rangle$. Since $(M2, op2)$ is one of the "leaves" corresponding to its operations graph, we understand that the processing of the part in the system is finished, and then, the part has to go outside the system.

The PN in Figures 26.9 and 26.10 represent what parts of type $W1$ and $W2$ supply the system PN model, respectively. In order to make the figures more readable, the operation component does not appear. So, $(M1, op1)$ is represented as $M1$, while the process plan $W1$ is represented as 1. For the same reason, the *root* node is represented by means of the letter s. Notice that if an idle state place is added to the net in Figure 26.9 we have exactly the same net as in Figure 26.7-a. And the same goes for the nets in figures 26.10 and 26.7-b.

In the final model, the different "small" transitions will be modelled by means of the colour domains of transitions in the global PN model, while "small" places will be modelled by means of colour domains of places. The

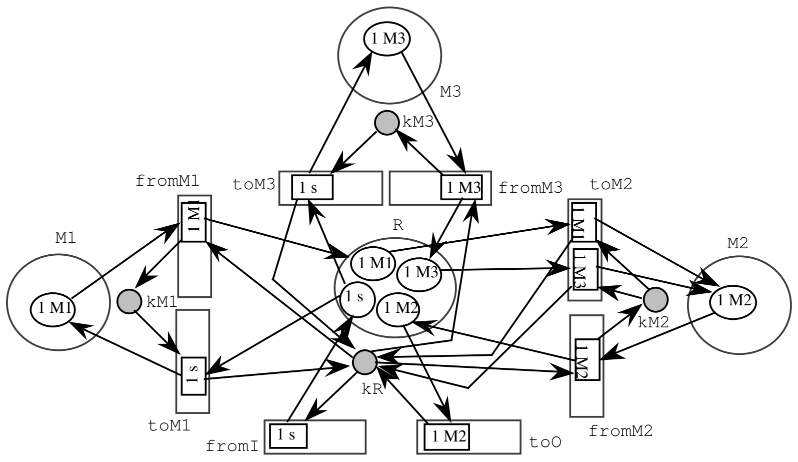

Figure 26.9: A partial PN model that considers only parts of type $W1$.

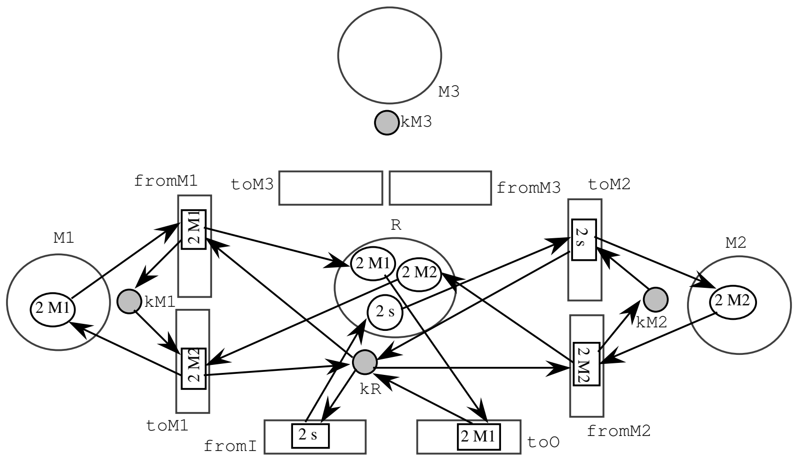

Figure 26.10: Partial PN model considering parts of type $W2$.

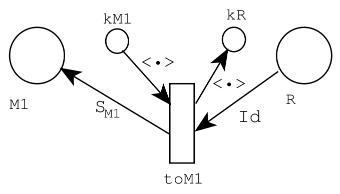

Figure 26.11: A (partial) view of the arcs and functions surrounding transition $toM1$.

arcs joining a place p (transition t) and a transition t (place p) will be modelled by means of a function defined over the colour domain of transition t and whose images belong to the colour domain of place p. For instance, the colour domain of place $M1$ in the (coloured) global model will be $cd(M1) = \{\langle W2, (M1, op4)\rangle, \langle W1, (M1, op1)\rangle\}$, the colour domain of transition $toM1$ will be $cd(toM1) = \{\langle W1, root\rangle, \langle W2, (M2, op3)\rangle\}$, while the colour domain of capacity places will be the "neutral colour". In this case, $cd(kR) = cd(kM1) = \{\bullet\}$.

The function labelling the arcs joining the previous places and transitions will be the following. $\mathbf{Post}[M1, toM1] = S_{M1}$ is defined from $cd(toM1)$ to $cd(M1)$ as:

- $S_{M1}(\langle W1, root\rangle) = \langle W1, (M1, op1)\rangle$
- $S_{M1}(\langle W2, (M2, op3)\rangle) = \langle W2, (M1, op4)\rangle$
- $\mathbf{Post}[kM1, toM1] = \mathbf{Pre}[kR, toM1] = \langle\bullet\rangle$
- $\mathbf{Pre}[R, toM1] = Id$

where $\langle\bullet\rangle$ represents the constant function that always returns the neutral colour ("neutral function") and Id is a symbolic representation of the Identity function in its "liberal" meaning; i.e., $Id(x) = x$, even if the origin and final sets are not the same. Figure 26.11 shows the arcs and functions related to transition $toM1$ that the CPN model would have. Figure 26.12 shows the final CPN model for the example considered. The other functions are as follows:

- $S_{M2}(\langle W1, (M1, op1)\rangle) = \langle W1, (M2, op2)\rangle$
- $S_{M2}(\langle W1, (M3, op1)\rangle) = \langle W2, (M1, op4)\rangle$
- $S_{M2}(\langle W2, root\rangle) = \langle W2, (M2, op3)\rangle$
- $S_{M3}(\langle W1, root\rangle) = \langle W1, (M3, op1)\rangle$

We have shown in an intuitive way how the CPN model can be obtained from the considered input data. In [13] the algorithms that obtain this coloured model in an automatic way, as well as their complexity are presented.

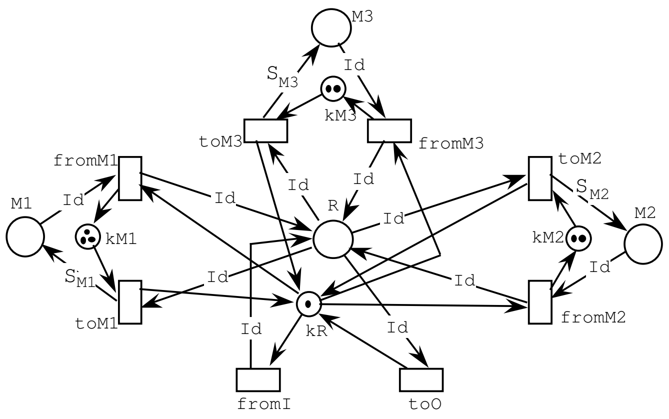

Figure 26.12: The Coloured Petri net obtained applying the proposed methodology. All arcs related to capacity places must be labelled $\langle \bullet \rangle$

26.3.3 Using structural analysis for the system control

Structural elements (P-Semiflows and T-Semiflows, for instance) (see Section**??**) have been widely used in order to get information from the model. In the example considered (this is also valid for all the nets belonging to the S^3PR class), a lot of information about the model correctness is given. Let us now consider, once again, the PN in Figure 26.13.

- It is easy to prove that we have two kinds of (minimal) P-Semiflows. For each resource, the sum of the number of tokens in the resource and its *holders* is always equal to the initial marking of the resource. A state place is a holder of a resource r if the resource is used in this state. For instance, $M1(1, M1)$ is a holder of the resource $M1$ since the marking of $M1$ decreases when a token enters in $M1(1, M1)$ (place $M1(1, M1)$ "uses" $M1$). Notice also that when a token leaves $M1(1, M1)$, the marking of $M1$ is increased. The set of holders of a resource r is denoted as $H(r)$[2]. $\{r\} \cup H(r)$ induces the following P-Semiflow: at each reachable marking $\mathbf{m}$, $\mathbf{m}[r] + \sum_{p \in H(r)} = \mathbf{m_0}[r]$. In our example we have:

  - $H(M1) = \{M1(1, M1), M1(2, M1)\}$, which induces the P-Semiflow $\mathbf{m}[kM1] + \mathbf{m}[M1(1, M1)] + \mathbf{m}[M1(2, M1)] = \mathbf{m_0}[kM1] = 3$. Which is the particular interpretation of this P-Semiflow? Notice that tokens in places $M1(1, M1)$ and $M1(2, M1)$ model parts that are being processed in machine $M1$. The P-Semiflow states that the number of parts in $M1$ plus the number of free positions in $M1$ is always 3. This is a necessary condition for our model to be correct.

[2]For a set of resources S, we extend the definition of set of holders as follows: $H(S) = \bigcup_{r \in S} H(r)$

- $H(M2) = \{M2(2,M2), M2(1,M2)\}$, which induces the P-Semiflow $\mathbf{m}[kM2] + \mathbf{m}[M2(1,M2)] + \mathbf{m}[M2(2,M2)] = \mathbf{m_0}[kM2] = 2$
- $H(M3) = \{M3(1,M3)$, which induces the P-Semiflow $\mathbf{m}[kM3] + \mathbf{m}[M3(1,M3)] + \mathbf{m}[kM3] = \mathbf{m_0}[kM3] = 2$
- $H(R) = \{R(1,s), R(1,M1), R(1,M3), R(1,M2), R(2,s), R(2,M2), R(2,M1)\}$, which induces the P-Semiflow $\mathbf{m}[kR] + \mathbf{m}[R(1,s)] + \mathbf{m}[R(1,M1)] + \mathbf{m}[R(1,M3)] + \mathbf{m}[R(1,M2)] + \mathbf{m}[R(2,s)] + \mathbf{m}[R(2,M2)] + \mathbf{m}[R(2,M1)] = \mathbf{m_0}[kR] = 1$

There is a second type of P-Semiflows: for each S^2P, at each reachable marking, the sum of the number of tokens in its places is equal to the initial marking of the idle place. For the example we have:

- $\mathbf{m}[p0(1)] + \mathbf{m}[R(1,s)] + \mathbf{m}[M3(1,M3)] + \mathbf{m}[R(1,M3)] + \mathbf{m}[M1(1,M1)] + \mathbf{m}[R(1,M1)] + \mathbf{m}[M2(1,M2)] + \mathbf{m}[R(1,M2) = \mathbf{m_0}[p0(1)]$
- $\mathbf{m}[p0(2)] + \mathbf{m}[R(2,s)] + \mathbf{m}[M2(2,M2)] + \mathbf{m}[R(2,M2)] + \mathbf{m}[M1(2,M1)] + \mathbf{m}[R(2,M1)] = \mathbf{m_0}[p0(2)]$

The general interpretation of both kinds of P-Semiflows is easy. P-Semiflows of the first kind state the correctness of the model with respect to the resources. This means: *1)* A resource can be neither created nor destroyed. *2)* At each reachable state, the sum of the available free positions/copies of each resource and the parts that use it is always equal to the total capacity of the resource. P-Semiflows of the second type establish the correctness with respect to the types of parts. The initial marking of the idle places establishes, for each type of part, the maximal number of parts of the type that are allowed to be concurrently processed in the system. The P-Semiflow for a type of part states that the total number of parts that are concurrently processed plus the number of parts of this type that can still be accepted is constant.

- It is also very easy to prove that each cycle of each S^2P forms a T-Semiflow. The interpretation of these T-Semiflows is easy: each T-Semiflow establishes a possible processing sequence for a part. This means that when the firing of a T-Semiflow corresponding to an S^2P is completed, the processing of a part of this type has been finished.

  Or, all things together, when the processing of all the parts inside the system finishes (every T-Semiflow, once started, is completed), the initial state of the system is reached. Considering parts of type 1, for instance, we have two T-Semiflows related to it. The first one, σ_1 is as follows: $\sigma_1[from(1,s)] = 1$, $\sigma_1[toM1(1,s)] = 1$, $\sigma_1[fromM1(1,M1)] = 1$, $\sigma_1[toM2(1,M1)] = 1$, $\sigma_1[fromM2(1,M2)] = 1$, and $\sigma_1[toO(1,M2)] = 1$ and $\sigma_1[t] = 0$ for any other transition.

  Analogously, the second one is the following: $\sigma_2[from(1,s)] = 1$, $\sigma_2[toM3(1,s)] = 1$, $\sigma_2[fromM3(1,M3)] = 1$, $\sigma_2[toM2(1,M3)] = 1$,

$\sigma_2[fromM2(1,M2)] = 1$, and $\sigma_2[toO(1,M2)] = 1$ and $\sigma_2[t] = 0$ for any other transition. Being C the net incidence matrix, it is verified that $C \cdot \sigma_i = \mathbf{0},\ i = 1,2$. Notice that the firing of any of the two previous T-Semiflows models the completion of the processing of a part of type one.

Now, we are going to concentrate on another kind of structural elements, the siphons (see Definition**??**), and we are going to show that, for this class of nets, the siphons are related to the system liveness. In Petri net theory there are two main concepts related to the existence of system activities. The first one is the concept of deadlock freeness, while the second concept is the concept of liveness. Let us now pay a little more attention to these concepts in the application domain we are dealing with.

- *deadlock freeness*. A Petri net system (i.e., a Petri net with an initial marking) is said to be deadlock free when at each reachable marking there exists at least one transition that is enabled. In our application domain this means that it is always possible to make *some* production activity (executing a new step in the production sequence of a part, introducing a new part in the system, for instance).

- *liveness*. Deadlock freeness is not enough for this domain: it is possible to have a part of the system that can always be running correctly, but also another part of the system that is in a deadlock. For instance, it is possible to have a type of parts to be correctly processed, but also to have some parts in the system whose processing has been started but cannot be finished. So, deadlock freeness is not good enough a property for highly automated systems; liveness is the "good" property. A Petri net system is said to be live if from each reachable marking it is always possible to fire *any* transition. In the application domain considered this means that it is always possible to execute the system actions modelled by means of any transition. So, as a consequence 1) the processing of each part, once started, can always be finished: the transitions "driving" a token (modelling a part) to the system output can be fired. So, the processing of the part can be finished. This also means 2) if there are always new raw materials, their processing can be carried out.

In some cases, as for free-choice nets [10], the previous properties are equivalent. But this is not the case for the class of nets we are considering.

When facing automated systems, deadlock problems are very important issues. In effect, if we want a system highly automated, a deadlock represents a special situation we need to deal with [2, 12, 44]. As stated above, a deadlock represents that the processing of a part has been started, but cannot be finished. Therefore, the part can stay in the system for a long period of time (until some recovery strategy is applied). During this time, the part is using some system resources and the system performance will be decreased. In systems where deadlocks can appear, two main different approaches have been adopted:

the deadlock prevention/avoidance approach and the deadlock detection and recovery approach. In the first approach a deadlock prevention/avoidance control policy is applied in such a way that the system evolutions are controlled in order to ensure that no deadlock is reached. In the second one, when a deadlock is detected, a recovery strategy is applied in order to change the system state to a non-deadlocked state.

For the general class of nets we are considering, different control policies can be found in [2, 44, 12, 45]. Let us show how the structure of the net allows us to establish a deadlock characterisation which can be applied in order to get a control policy for deadlock prevention. Let us consider the net in Figure 26.13. From the initial marking shown in the figure the firing of sequence $\sigma = (\mathit{fromI}(1,s)\mathit{toM}1(1,s))^3\mathit{fromI}(2,s)\mathit{toM}2(2,s)\mathit{fromM}2(2,M2)$ yields a marking $\mathbf{m}$ ($\mathbf{m_0}[\sigma\rangle\mathbf{m}$) such that $\mathbf{m}[kM2] = \mathbf{m}[kM3] = 2$, $\mathbf{m}[M1(1,M1)] = 3$, $\mathbf{m}[R(2,M2)] = 1$, $\mathbf{m}[p_1^0] = 3$, $\mathbf{m}[p_2^0] = 5$ and $\mathbf{m}[p] = 0$ for any other place p. Notice that this state is a deadlock: the parts modelled by the tokens in place $M1(1,M1)$ cannot change the state. This means that these three parts will remain in machine $M1$ (forever if nothing is done!). The same goes for the part modelled by means of the token in place $R(2,M2)$. A question appears. Is there any information in the Petri net structure allowing the characterisation of deadlock situations? The answer to this question is "yes". For the considered marking $\mathbf{m}$, the set of heavily shaded places in Figure 26.13, $S = \{R(1,s), R(1,M3), R(1,M1), R(1,M2), R(2,M1), R(2,s), kM1, kR\}$, is a siphon and it is unmarked. Remember that one of the most important behavioral properties of a siphon is that once it becomes unmarked, it remains unmarked. So, no transition in $S^\bullet$ can fire any more. So, neither $\mathit{fromM}1(1,M1)$ nor $\mathit{toM}1(2,M2)$ can fire, and the tokens considered will remain in their places. Therefore, the processing of the considered parts cannot be finished.

The following theorem establishes the liveness characterisation.

Theorem 26.1 *[12] Let $\langle\mathcal{N},\mathbf{m_0}\rangle$ be a marked S^3PR, let $\mathbf{m} \in \mathrm{RS}(\mathcal{N},\mathbf{m_0})$ and let $t \in T$ be a dead*[3] *transition for $\mathbf{m}$. Then, there exists a reachable marking $\mathbf{m}' \in \mathrm{RS}(\mathcal{N},\mathbf{m})$ and a (minimal) siphon S such that $\mathbf{m}'(S) = 0$.*

Therefore we can deduce the following corollary.

Corollary 26.2 *[12] Let $\langle\mathcal{N},\mathbf{m_0}\rangle$ be a marked S^3PR. Then, $\langle\mathcal{N},\mathbf{m_0}\rangle$ is live if, and only if, for every reachable marking $\mathbf{m} \in \mathrm{RS}(\mathcal{N},\mathbf{m_0})$ and every (minimal) siphon S,$\mathbf{m}(S) \neq 0$*

This liveness characterisation is not true for general nets. The net in Figure 26.14 is a clear example: transition t is dead for the shown marking. However, the only siphon in the net, $\{p,q,r,s\}$, is always marked. In the following we are going to see how this deadlock characterisation can be used in order to establish a control policy for deadlock prevention. The aim of the control policy

[3]To say that a transition is *dead* for a reachable marking $\mathbf{m}$ is equivalent to say that the transition cannot be fired at any state reachable from $\mathbf{m}$.

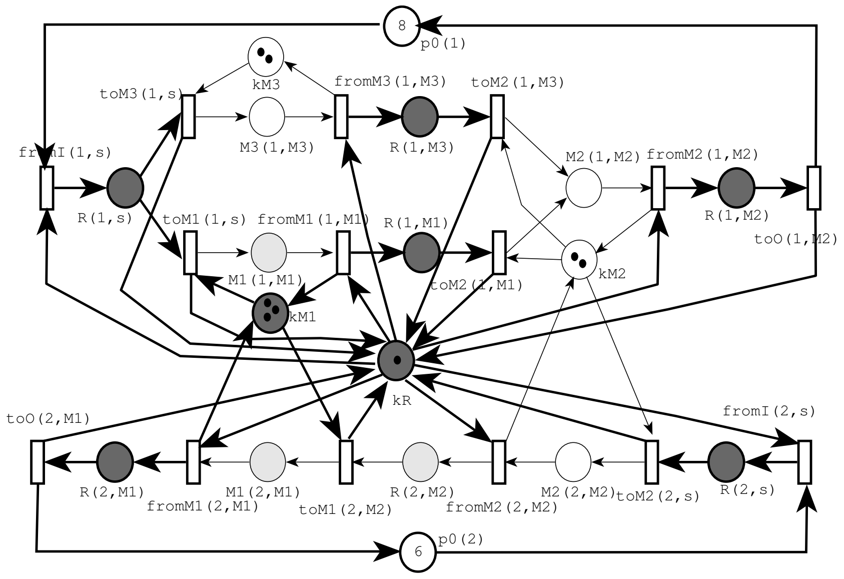

Figure 26.13: The considered S^3PR where the elements related to the considered siphon have been shaded.

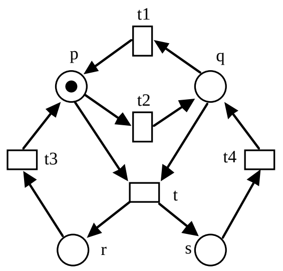

Figure 26.14: The property *a dead transitions implies an empty siphon* is not true for general nets: t is dead, but no siphon is empty.

is to add some constraints to the system in such a way that no deadlock state is reached.

Let us distinguish two classes of minimal siphons: those which are the support of a P-Semiflow and those which are not. Considering the set of minimal P-Semiflows (previously presented) and the class of initial markings we are considering, siphons of the first class remain always marked, and therefore, they are not involved in deadlock problems. Then, only siphons of the second class are related to deadlocks. We will refer to this second class of siphons as "dangerous siphons". A dangerous siphon S can be written as $S = S_R \uplus S_P$, where $S_R = S \cap P_K$, $S_P = S \setminus S_R = S \cap P_S$[4]. The set of holders $H(S_R)$ can be partitioned into two subsets: those holders that belong to the siphon S (heavily shaded holders in Figure 26.13) and those that do not (light shaded places in Figure 26.13). Notice that, so that a token can enter one of these holders, a token needs to have been previously "stolen" from the siphon. For instance, the firing of transition $toM1$ decreases the marking of siphon S in one token. This means that a token in place $M1(2, M1)$ implies a token less in $kM1$, and then, a token less in S.

The control policy for deadlock prevention established in [12] uses this property. For each dangerous siphon, a structurally implicit place **??** is added ensuring that at any reachable state the number of tokens in the system that can reach those siphon holders which "steal" tokens from the considered siphon is smaller than the initial marking of the siphon. In this way it is ensured that the marking of the siphon is always ≥ 1, i.e., the siphon cannot be emptied.

For the siphon considered, the control policy will add a place S_1 (see Fig. 26.15) such that ${}^{\bullet}S_1 = \{fromM1(2, M1), fromM1(1, M1), toM3(1, s)\}$ and $S_1{}^{\bullet} = \{fromI(1, s), fromI(2, s)\}$. Since $\mathbf{m_0}[S_1] = 4$, it is enough to put $\mathbf{m_0}[S_1] = 3$ (for short, we also call $\mathbf{m_0}$ the initial marking of the extended net) to ensure that S cannot be emptied. Of course, any value $\mathbf{m_0}[S_1] \in \{1, 2, 3\}$ will be valid. However, we take the maximum of them in order to have as much parallelism as possible using this control strategy. Notice that the addition of this new place generates a new P-Semiflow: for each reachable marking $\mathbf{m}$ of the controlled net, we have $\mathbf{m}[S_1] + \mathbf{m}[R(1, s)] + \mathbf{m}[M1(1, M1)] + \mathbf{m}[R(2, s)] + \mathbf{m}[M2(2, M2)] + \mathbf{m}[R(2, M2)] + \mathbf{m}[M1(2, M1)] = 3$. From this invariant relation it is deduced that $\mathbf{m}[M2(2, M2)] + \mathbf{m}[R(2, M2)] + \mathbf{m}[M1(2, M1)] + \mathbf{m}[M1(1, M1)] \leq 3$, and then, since no more than three tokens can be stolen from the siphon, it cannot become unmarked.

The same goes for the rest of dangerous siphons of the example considered. Those are the following:

$$
\begin{array}{lcl}
S_2 & = & \{R(1, s), R(1, M2), R(2, M2), R(2, M1), kM2, kR\} \\
S_3 & = & \{R(1, s), R(1, M2), R(2, M1), kM1, kM2, kR\} \\
S_4 & = & \{R(1, M3), R(1, M1), R(1, M2), R(2, s), R(2, M1), kM1, kM3, kR\}
\end{array}
$$

[4]For a given S^3PR, P_K denotes the set of resource places, P_S the set of state places and P_0 the set of idle states

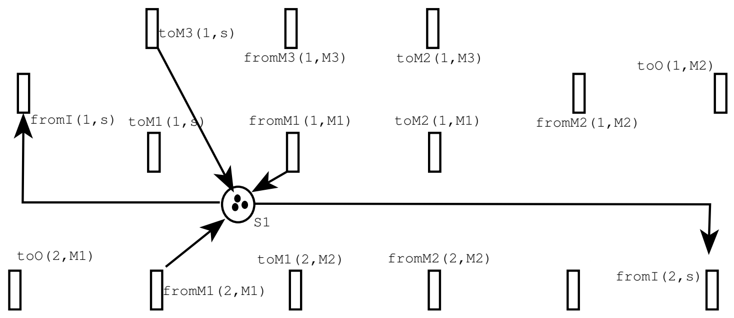

Figure 26.15: The part of the control policy for deadlock prevention generated by the siphon in Figure 26.13.

$$S_5 = \{R(1,M2), R(2,M1), kM1, kM2, kM3, kR\}$$

And for them, the added elements are the following:

- ${S_2}^\bullet = \{fromI(1,s), fromI(2,s)\}$
- ${}^\bullet S_2 = \{fromM2(2,M2), fromM2(1,M2)\}$
- ${S_3}^\bullet = \{fromI(1,s), fromI(2,s)\}$
- ${}^\bullet S_3 = \{fromM1(2,M1), fromM2(1,M2)\}$
- ${S_4}^\bullet = \{fromI(1,s), fromI(2,s)\}$
- ${}^\bullet S_4 = \{fromM1(2,M1), fromM1(1,M1), fromM3(1,M3)\}$
- ${S_5}^\bullet = \{fromI(1,s), fromI(2,s)\}$
- ${}^\bullet S_5 = \{fromM1(2,M1), fromM2(1,M2)\}$

In order to have a final CPN model, the control policy can be incorporated to the initial CPN obtained using the proposed methodology. To do that, a new place called CP (Control Place) is added to the coloured model. The colour domain of this place is a set bijective with the set $\{S_1, ..., S_k\}$ of control places to the underlying S^3PR model added by the control policy. Let $cd(CP) = \{V_1, ..., V_k\}$ be such a set. The arcs which the control policy has added to the underlying S^3PR are represented by the arcs and functions which must be added to the final coloured model.

The elements added in Figure 26.16 are as follows:

$$\begin{aligned} cd(CP) &= \{V_1, V_2, V_3, V_4, V_5\} \\ \widehat{\mathbf{m_0}}[CP] &= 3V_1 + 2V_2 + 5V_3 + 7V_4 + 7V_5 \end{aligned}$$

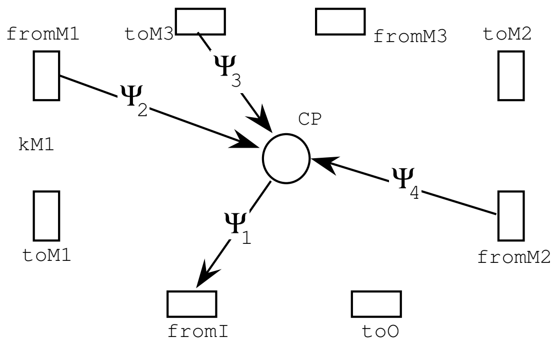

Figure 26.16: Elements added by the control policy.

$$\begin{aligned}
\Psi_1(\langle W1, root\rangle) &= V_1 + V_2 + V_3 + V_4 + V_5\\
\Psi_1(\langle W2, root\rangle) &= V_1 + V_2 + V_3 + V_4 + V_5\\
\Psi_2(\langle W2, (M1, op4)\rangle) &= V_1 + V_3 + V_4 + V_5\\
\Psi_2(\langle W1, (M1, op1)\rangle) &= V_1 + V_4\\
\Psi_3(\langle W1, root\rangle) &= V_1\\
\Psi_4(\langle W1, (M2, op2)\rangle) &= V_2 + V_3 + V_5\\
\Psi_4(\langle W2, (M2, op3)\rangle) &= V_2
\end{aligned}$$

A question arises. In the previous sections no constraint has been imposed with respect to the system layout. However, in the definition of an S^3PR a termination property has been imposed (see the introduction of S^2P in section 26.3). So, in order to ensure that the underlying system belongs to the S^3PR class we need to constrain the system layout to those of *acyclic handling*. This means that in the layout model no cycle is possible using only *handlers*. This ensures that there is no cycle without transformation, and then, in the underlying model each production sequence eventually reaches the idle state. From the application domain point of view, Flexible Manufacturing Cells and Flexible Manufacturing Lines (as considered in [29]) correspond to this class of acyclic handling systems. However, systems where the layout contains some carousel do not fit into this class: a carousel allows parts that can complete cycles with no transformation, which would violate the termination property imposed on the S^2P.

It must be pointed out once again that the control policy applied is not optimal, where optimal in this context means maximal concurrency without deadlock problems. For the general case of the systems under consideration, to find an optimal control policy remains an open problem. However, for some more restricted cases some solutions have been found [45].

26.4 Conclusions

The chapter has been devoted to show how Petri nets can be applied to Flexible Manufacturing Systems. This is a domain whose complexity and inherent concurrency requires the use of formal methods to deal with very important problems, such as the design and the control problems, in order to synthesise the software that ensures a correct system behaviour.

With respect to the first problem, the chapter has introduced a design methodology which, from the input data describing both the structure of the system architecture and the logic of the processing of different types of parts to be processed obtains, in an automatic way, a Coloured Petri net model of the entire system. The use of a high level Petri net model has the advantage of compactness. Also, the model obtained by this methodology shows in a clear way the structure of the system: the skeleton of the net has the same form as the configuration of the hardware components. Since the different processing sequences are modelled by means of colour domains of places and transitions and the functions labelling the arcs, the introduction/withdrawing of types of parts does not change the "look" of the model.

With respect to the control problem, the chapter has studied the ordinary place/transition nets corresponding to the coloured models synthesised by the modelling methodology presented. This has allowed the study of deadlock problems for this systems from a structural perspective. One of the advantages of Petri net models is that they allow the study of some properties using structural techniques, avoiding the computation of the reachability graph and, indeed, avoiding the state space explosion problem. Unfortunately, structural techniques characterising liveness have not been developed for general Petri net models, but for special subclasses (e.g., state machines, marking graphs, free choice nets, choice free systems,...). However, the special syntactic structure of the class corresponding to the systems we are considering allowed us to establish the deadlock characterisation. As it has been shown, this characterisation has been used in order to establish a control policy for deadlock prevention which constrains the system evolutions to ensure the liveness of the controlled system.

From the development of the chapter, some general conclusions can be drawn:

- Petri nets are a family of formalisms well suited for application to FMS environments.
- The results obtained in the general Petri net theory are not enough to deal with all the problems that arise in application domains. Therefore, it is necessary to develop new specific results adapted to these domains. However, Petri nets are a powerful framework which allows these developments.
- These specific results must not be "ad hoc" for each problem, but they must concentrate on some modelling/programming paradigms. In the class of nets considered in the present chapter, we fall into the case of

sequential processes using monitors (in a restricted way). Natural extensions to this case, applicable to general manufacturing systems, operating systems, databases, etc., comprise the use of monitors in a general way and communication by means of buffers [36, 38, 31].

- The use of Coloured Petri nets, and High Level Petri nets in general, has some important advantages with respect to the modelling. However, we are faced with a new problem: symbolic processing of these nets is not complete, and hence, it must be developed. The steps given for some subclasses of coloured Petri nets makes this approach look promising.

Bibliography

[1] T. Agerwala and Y. Choed-Amphai. A synthesis rule for concurrent systems. In *Proceedings of the 15th. Design Automation Conference*, pages 305–311, Las Vegas (U.S.A.), June 1978.

[2] Z. Banaszak and B. Krogh. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Transactions on Robotics and Automation*, 6(6):724–734, December 1990.

[3] L. Bernardinello and F. De Cindio. A survey of basic net models and modular subclasses. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 609 of *Lecture Notes on Computer Science*, pages 304–351. Springer-Verlag, 1992.

[4] J. Browne, D. Dubois, K. Rathmill, S.P. Sethi, and K.E. Stecke. Classification of flexible manufacturing systems. *The FMS magazine*, 2(2), 1984.

[5] E. Castelain, D. Corbeel, and J. Gentina. Comparative simulations of control processes described by Petri nets. In *Proceedings of the IEEE COMPINT'85 Conference*, 1985.

[6] G. Chehaibar. *Use of Reentrant Nets in Modular Analysis of Colored Nets*, pages 596–617. High-level Petri Nets. Theory and Application. Springer-Verlag, 1991.

[7] J.M. Colom, J. Martínez, and M. Silva. Packages for validating discrete production systems modeled with Petri nets. In P. Borne and S.G. Tzafestas, editors, *Applied Modelling and Simulation of Technological Systems*, pages 529–536. Elsevier Science Publishers B.V. (North-Holland), 1987.

[8] A.A. Desrochers and R.Y. Al-Jaar. *Application of Petri Nets in Manufacturing Systems: Modeling, Control and Performance Analysis.* IEEE Press, 1995.

[9] F. Dicesare, G. Harhalakis, J.M. Proth, M. Silva, and F.B. Vernadat. *Practice of Petri Nets in Manufacturing.* Chapman & Hall, 1993.

[10] J. Esparza and M. Silva. On the analysis and synthesis of free choice systems. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 483 of

Lecture Notes in Computer Science, pages 243–286. Springer Verlag, Berlin, 1991.

[11] J. Ezpeleta and J.M. Colom. Automatic synthesis of colored Petri nets for the control of FMS. *IEEE Transactions on Robotics and Automation*, 13(3):327–337, June 1997.

[12] J. Ezpeleta, J.M. Colom, and J. Martínez. A Petri net based deadlock prevention policy for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation*, 11(2):173–184, April 1995.

[13] J. Ezpeleta, J.M. Couvreur, and M. Silva. A new technique for finding a generating family of siphons, traps and st-components. application to colored Petri nets. In G. Rozenberg, editor, *Advances in Petri Nets 1993*, volume 674 of *Lecture Notes on Computer Science*, pages 126–147. Springer-Verlag, Aarhus (Denmark), 1993.

[14] J. Ezpeleta and J. Martínez. *Petri Nets as a Specification Language for Manufacturing Systems*, pages 427–436. Robotics and Flexible Manufacturing Systems. Elsevier Science Publishers B.V. (North Holland), 1992.

[15] R. Fehling. A concept of hierarchical Petri nets with building blocks. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 674 of *Lecture Notes on Computer Science*, pages 148–168. Springer-Verlag, 1993.

[16] A. Elia G. Bruno. Operational specification of process control systems: Execution of Prot nets using OPS5. In *Proceedings of the 10th World IFIP Congress*, Dublin, 1986.

[17] J.C. Gentina, J.P. Bourey, and M. Kapusta. Coloured adaptative structured Petri nets. *Systems*, 1(1):39–47, February 1988.

[18] J.C. Gentina, J.P. Bourey, and M. Kapusta. Coloured adaptative structured Petri nets. part-ii. *Systems*, 1(2):103–109, May 1988.

[19] F. S. Hsieh and S. C. Chang. Deadlock avoidance controller synthesis for flexible manufacturing systems. *IEEE Transactions on Robotics and Automation*, 10(2):196–209, April 1994.

[20] P. Huber, K. Jensen, and R.M. Shapiro. Hierachies in coloured Petri nets. In G. Rozenberg, editor, *Advances in Petri Nets 1991*, volume 483 of *Lecture Notes in Computer Science*, pages 313–341. Springer Verlag, Berlin, 1991.

[21] M.A. Jafari. An architecture for a shop-floor controller using colored Petri nets. *The International Journal of Flexible Manufacturing Systems*, 4(4):159–181, 1992.

[22] M.D. Jeng and F. DiCesare. Synthesis using resource control nets for modeling shared-resource systems. *IEEE Transactions on Robotics and Automation*, 11(3):317–327, June 1995.

[23] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin Heidelberg, 1994.

[24] S.B. Joshi, E.G. Mettala, J.S. Smith, and R.A. Wysk. Formal models for control of flexible manufacturing cells: Physical and system mode. *IEEE Transactions on Robotics and Automation*, 11(4):558–570, August 1995.

[25] B.H. Krogh and C.L. Beck. Synthesis of place/transition nets for simulation and control of flexible manufacturing systems. In *Proceedings of the IFIP Symposium on Large Scale Sytems*, Zurich, August 1986. IFIP.

[26] J. Martínez, P. Muro, and M. Silva. Modeling, validation and software implementation of production systems using high level Petri nets. In *Proc. of the IEE International Conference on Robotics and Automation*, pages 1180–1184, Raleigh (North Carolina), 1987.

[27] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[28] Y. Narahari and N. Viswanadham. A Petri net approach to the modelling and analysis of flexible manufacturing systems. *Annals of Operation Research*, 3:449–472, 1985.

[29] H.T. Papadopoulos, C. Heavy, and J. Browne. *Queuing Theory in Manufacturing Systems Analysis and Design.* Chapman & Hall, London, 1993.

[30] J.M. Proth and X. Xie. *Petri Nets. A Tool for Design and Management of Manufaturing Systems.* John Wiley & Sons, 1996.

[31] L. Recalde, E. Teruel, and M. Silva. On well-formedness analysis: The case of deterministic systems of sequential processes. In J. Desel, editor, *Proceedings of the International Workshop on Structures in Concurrency Theory, Workshops in Computing*, pages 279–293. Springer, 1995.

[32] S. Ribaric. Knowledge representation scheme based on Petri net theory. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(4), December 1988.

[33] A. Sahraoui, H. Atabatche, M. Couvoisier, and R. Valette. Joining Petri nets and knowledge based systems for monitoring purposes. In *Invited Sessions: Petri Nets and Flexible Manufacturing Systems, IEEE Int. Conference on Robotics and Automation*, pages 1160–1165, Raleigh, USA, 1987.

[34] M. Silva and E. Teruel. Petri nets for the design and operation of manufacturing systems. *European Journal of Control*, (3):182–199, 1997.

[35] M. Silva and R. Valette. Petri nets and flexible manufacturing. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 424 of *Lecture Notes on Computer Science*, pages 374–417. Springer-Verlag, Berlin, 1989.

[36] Y. Souissi. Deterministic systems of sequential processes: a class of structured Petri nets. In *Proceedings of the 12th. International Conference on Application and Theory of Petri Nets*, pages 62–81, Aarhus (Denmark), 1991.

[37] I. Suzuki and T. Murata. A method for stepwise refinements and abstraction of petPetriri nets. *Journal of Computer Systems Science*, 27:51–76, 1983.

[38] F. Tricas and J. Martínez. An extension of the livennes theory for concurrent sequential processes competing for shared resources. In *Proceedings of the 1995 International Conference on Systems, Man and Cybernetics*, pages 4119–4124, Vancouver (Canada), October 1995.

[39] K.P. Valavanis. On the hierarchical modeling analysis and simulation of flexible manufacturing systems with extended Petri nets. *IEEE Transactions on Systems, Man and Cybernetics*, 20(1):94–110, January 1990.

[40] R. Valette. Analysis of Petri nets by stepwise refinements. *Journal of Computer Science*, (28):35–46, 1979.

[41] R. Valette and H. Atabakhche. Petri nets for sequence constraint propagation in knowledge based approaches. In *Concurrency and Nets*, Advances in Petri Nets, pages 555–570. Springer Verlag, Berlin, 1987.

[42] J.L. Villarroel, J. Martínez, and M. Silva. Graman: a graphic system for manufacturing system modelling and simulation. In *Proceedings of the IMACS Symposium on System Modelling and Simulation*, pages 311–316, Cetraro (Italy), September 1988.

[43] J.L. Villarroel and P.R. Muro. Using Petri net models at the coordination level for manufacturing systems control. *Robotics and Computer-integrated Manufacturing*, 1(11):41–50, 1994.

[44] N. Viswanadham and Y. Narahari. *Performance Modeling of Automated Manfacturig Systems.* Prentice-Hall, 1992.

[45] K.Y. Xing, B.S. Hu, and H.X. Chen. Deadlock avoidance policy for Petri-net modeling of flexible manufacturig systems with shared resources. *IEEE Transactions on Automatic Control*, 41(2):289–295, February 1996.

[46] M. Zhou and F. DiCesare. Parallel and sequential mutual exclusions for Petri net modelling of manufacturing systems with shared resources. *IEEE Transactions on Robotics and Automation*, 7(4), August 1991.

[47] M. Zhou and F. Dicesare. *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems.* Kluwer Academic Publishers, 1993.