XX (11 6555.1)

CINVESTAV

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

DIAGNOSTICO DE FALLAS EN SISTEMAS DE MANUFACTURA DISCRETOS

Tesis que presenta:
Mildreth Isadora Alcaraz Mejía

para obtener el grado de:
Maestro en Ciencias

en la especialidad de:
Ingeniería eléctrica

Directores de Tesis
Dr. Luis Ernesto López Mellado
Dr. Antonio Ramírez Treviño

Guadalajara, Jalisco, Agosto del 2004.

DIAGNOSTICO DE FALLAS EN SISTEMAS DE MANUFACTURA DISCRETOS

Tesis de Maestria en Ciencias
Ingeniería eléctrica

Por:
Mildreth Isadora Alcaraz Mejía
Ingeniero en Sistemas Computacionales
Instituto Tecnológico de Colima 1995-2000

Directores de Tesis
Dr. Luis Ernesto López Mellado
Dr. Antonio Ramírez Treviño

CINVESTAV del IPN Unidad Guadalajara, Agosto del 2004.

CINVESTAV

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

Petri Net-Based Fault Diagnosis of Discrete Manufacturing Systems

A thesis presented by
Mildreth Isadora Alcaraz Mejía

To obtain the degree of:
Master in Sciences

On the subject of:
Electrical Engineering

Guadalajara, Jalisco, August 2004.

Petri Net-Based Fault Diagnosis of Discrete Manufacturing Systems

Master of Sciences Thesis
In Electrical Engineering

By:

Mildreth Isadora Alcaraz Mejía
Computer Systems Engineer
Instituto Tecnológico de Colima 1995-2000

Grant CONACYT no. 165143

Thesis Advisors:
Dr. Luis Ernesto López Mellado
Dr. Antonio Ramírez Treviño

CINVESTAV del IPN Unidad Guadalajara, August de 2004.

Contents

Acknowledges

Agradezco a Dios por todas sus bendiciones.

Esta tesis está dedicada a mis padres a quienes agradezco de todo corazón su gran amor, comprensión y apoyo incondicional.

A mi esposo por su amor, apoyo y por alentarme siempre a seguir adelante.

A mi hijo por ser mi principal motivación.

A mi hermana y a su esposo por que cuento con ellos siempre.

A la familia Alcaraz por su amor, apoyo y compañía.

A la familia Mejía por su apoyo.

A mis amigos por su confianza y lealtad.

A mis profesores y asesores de tesis por su disposición y ayuda brindada.

Al CONACYT.

Introduction

The dependency of the performance of a Flexible Manufacturing System (FMS) on the equipment efficiency and the process control, join with the constant increasing of the complexity of control systems leads to a higher demand on reliability and safety.

One of thc most important tasks to improve reliability, safety and economy is the fault diagnosis and error recovery, which if they are accurate and on time can help to avoid system shutdown, breakdown, and even dangerous operations concerning human losses and material damage.

Fault analysis has been, and still is, a very active area of research both in industry and in academy. Therefore, several modelling methods and a variety of techniques to perform diagnosis have been proposed.

In the Discrete Event Systems (DES) context, Finite Automata (FA) has been used as modelling formalism and many results on diagnosis have being obtained for DES represented as FA. In [25] [26], Lin uses a nondeterministic Mealy automaton and a deterministic Moore automaton as models for off-line and on-line diagnostics respectively; the author proposes two algorithms, one to find a minimal observable event set and an algorithm to find a control string that diagnoses the system. However, this last algorithm has an exponential complexity. In [31], Magni, et al. propose a technique describing the components as Finite State Machine (FSM) models, but they use composition rules that lead in a state explosion, even when they consider temporal information. In [40] [41], Sampath et al. also use FSM as component modelling incorporating the sensor maps for off-line and on-line diagnosis with a finite delay, but the analysis is comparable in complexity with the reachability tree search. In [17], Garcia et al. apply the concepts in [40] [41] to introduce a centralized modular diagnoser using FSM as modelling technique.

Nevertheless, when the size of the system increases and its behaviour is complex, the size of FA models make the analysis prohibitive, for this reason, the study of diagnosis with an observability approach using Petri Nets (PN) as modelling formalism has emerged in the late years. In [18], Giua defines an observer based on the observation of a word of events using PN and he also presents algorithms for estimating the state considering two cases, then; from this estimate, he shows how to design a controller to ensure that forbidden markings are not reached but it is not optimal. In [14], Fanni et al. use the algorithms presented in [18] to estimate the actual marking of the system and to design the controller; though this controlled system may result blocked, hence the authors show with an example, but only with an example, how to introduce time-out mechanisms to recover from this blocking.

This thesis addresses the diagnosis problem in Flexible Manufacturing Systems (FMS), using Interpreted Petri Nets (IPN) as modelling formalism. This allows, in one hand, representing both measurable and non-measurable internal states, and on the other hand, provides representations that are more compact. Besides the characterization from a language point of view of the diagnosability property, it is also presented a structural characterization, which reduces the complexity in the diagnosability analysis.

In addition, it is presented an extension of the modelling methodology that obtains binary *IPN* models presented by [38] to include faulty markings; it is a bottom-up modular construction technique. The study of diagnosability, the design of the diagnoser and the error recovery approach take advantage of this modelling technique.

Another contribution of this thesis is a fault recovery approach to complement the diagnosis approach modelled with Interpreted Re-Writing Nets (*IRWNets*), an extension of Re-Writing Nets (*RWNets*) [10] [11] based on *IPN*, a formalism introduced here that also helps in task reprogramming.

This thesis is organized as follows. Chapter I presents an overview of *FMS*, the concepts in the context of fault tolerance, and fault tolerance in *DES*. Chapter II presents the *PN* fundamentals, the *IPN* formalism and the extended modelling methodology. Chapter III defines the diagnosability property, its characterization from a language point of view and its structural characterization. Moreover, the design of a diagnoser net and the general scheme for diagnosis are also presented. Finally, chapter IV introduces the *IRWNets* definition and their dynamics in order to illustrate a fault recovery approach. Two cases of study to illustrate this approach and the task reconfiguration as a feature of the *IRWNet* formalism.

Chapter 1

Fault Tolerant Manufacturing Systems

Summary. *This chapter presents an overview of manufacturing systems, in particular, flexible manufacturing systems and their levels, components, processes and goals. It also presents some concepts in the context of fault tolerance as dependability, fault, error, failure concepts, fault types, fault attributes and fault tolerance goals. Finally, this chapter is focused in fault tolerance in discrete event systems, specially the diagnosis methods and related work.*

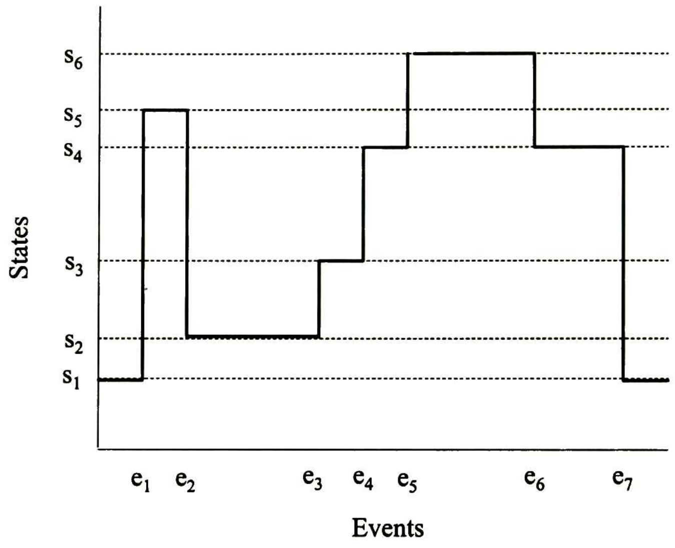


Figure 1.1: A DES Example

1.1 Manufacturing Systems

In Discrete Event Systems (*DES*), the state space of a system is described by a discrete set {0, 1, 2, 3,..}, and transitions are observed only at discrete points in time. These transitions are associated with "events" Informally, an event can be considered as an occurrence that produces a change from one state to another. The initial and final states may be the same or not.

The Discrete Event Dynamic Systems (*DEDS*), widely recognized by Discrete Event Systems, satisfy two main properties:

1) The state space is a discrete set.

2) The state transition mechanism is event-driven.

A formal definition of *DES* is presented in the following, and in Figure 1.1 a *DES* is shown:

Definition 1. *A* ***Discrete Event System*** *(DES) is a discrete-state, event-driven system, that is, its state evolution depends entirely on the occurrence of asynchronous discrete events over time [12].*

DES are common in the real world. Examples of *DES* are: Queuing Systems, Computer Systems, Communication Systems, Manufacturing Systems, Traffic Systems, Database Systems, Software Systems: Telephony, Monitoring and Control of Complex Systems, etc.

Flexible manufacturing systems, a specific type of manufacturing systems, is one form of automation which is an important solution to meet the three main goals managers look for in their enterprises: to increase productivity, to increase quality, and to reduce costs.

1.1.1 Manufacturing Systems Classification

In order to visualize **Flexible Manufacturing Systems** (FMS) with respect to all different types of Manufacturing Systems, the Manufacturing System Classification with respect to the shop architecture is presented below [43]:

- *Transfer Lines.* The process is a sequence of operations that take similar amounts of time to complete. Each workstation performs repetitively the same task. The main optimization problem is balancing, which means to make all the operation times in different workstations as similar as possible to reduce inactive times and to balance the workload of stations.

- *Production Lines.* There is some variability required in the workstations. Some intermediate buffers are introduced to filter out the variations. The buffer problem is critical, because large buffers intend to avoid blocking and starvation, but enhance the work in progress with the resulting economical problems.

- *Flow Shop.* There are some products which may be processed in a different way or pursue alternative paths. The sequencing problem tries to minimize inventory and production costs by finding a proper sequence of parts or lots, naturally controlled to satisfy the production demand.

- *Job Shop.* The variety of products in this kind of plant is greater. A production route is defined for each product. However, a greater flexibility results in a lower efficiency. The decision problem addresses the optimization while satisfying the production demands.

- *Flexible Manufacturing.* Capture the main advantages offered by the past production systems. Because this is a very important theme in this thesis, these systems will be widely described in the following.

1.1.2 Flexible Manufacturing Systems

In the middle of the 1960's, competition in the market became more intense and the main concern was cost. Later, the priority was quality. During the passing of time, the market became more and more complex, and speed of delivery became an extra concern customer also needed.

Therefore, companies had to look for a new strategy. This new strategy was called customizability, where companies have to adapt to the environment in which they operate, to be more flexible in their operations and to satisfy different market segments.

Thus, FMS is related to the effort of gaining competitive advantage.

The idea of an FMS was proposed in England in the 60's under the name *System 24*. It was named *System 24* referring to a flexible machining system that could operate without human operators 24 hours a day under computer control. Automation was the main goal of System 24.

Today, Flexible Manufacturing Systems (FMS) is a very interesting area and of great importance. Nevertheless, there is not an agreed definition of the term, but the following definition can be considered.

Definition 2. *The **Flexible Manufacturing System** (FMS) is a configuration of computer-managed numerical work stations where materials are automatically handled and loaded by machines [45].*

The word that makes the difference between *FMS* and traditional manufacturing systems is "flexible" which means that *FMS* has the flexibility to switch between two or more different parts for their process with little time lost.

1.2 Fault Tolerant Concepts

1.2.1 Dependability Concept

The dependability of a computer system characterizes its trustworthiness.

Definition 3. *A system is said to be **dependable** if the reliance can be placed on the service it delivers [24].*

The dependability attributes are: **availability** (readiness for usage), **reliability** (continuity of services), **safety** (non-occurrence of catastrophic consequences) and **security** (prevention of unauthorized access). These attributes can be considered in isolation but they can be interdependent. For example: if not reliable then not available, if unreliable then probably not safe, and so on.

In order to achieve the goal of dependability, an effort in all the phases of the system's development must be done [19]. At design time, the dependability can be increased through fault avoidance techniques. At implementation time, the dependability can be increased through fault removal techniques. At execution time, fault tolerance and fault evasion techniques are required.

1.2.2 Fault, Error and Failure

It is of great importance to have knowledge about the terminology used in the problem context, moreover, to make distinction among them. Therefore, the following basic concepts are presented [28].

Definition 4. *A **fault** is the original source of any problem. It may lead to an error directly or indirectly.*

Definition 5. *An **error** is the difference between what is specified and what is actually there. In a control system, an error is an observable difference between the actual state and the representation of the pretended state.*

Definition 6. *A **failure** occurs when an error affect the service delivered from a system. If the component that has the failure is part of a system, the result is a fault in the system which contains the failing component.*

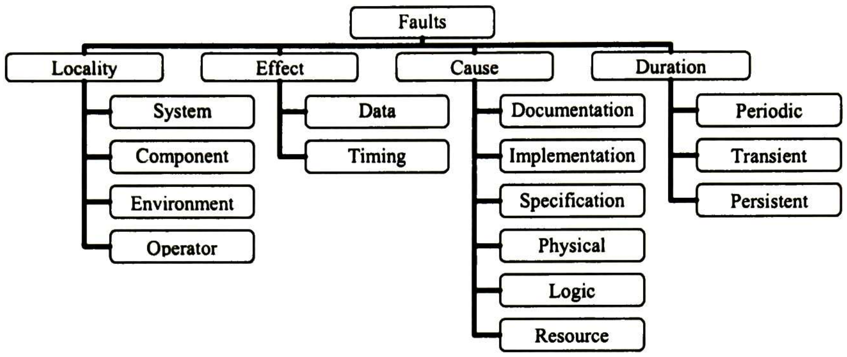


Figure 1.2: Fault Types

1.2.3 Fault Types

Since it is not possible to consider all existent faults, it is preferred to classify them based on locality, effect, cause, duration, among others [19]. This classification is shown in Figure 1.2.

1.2.4 Fault Attributes

Fault Observability

Definition 7. *A **fault** is said to be **observable** if the system interface shows a symptom.*

Definition 8. *A **symptom** is information indicating the existence of a fault. A symptom may be an observed fault or failure, or it may be a change in the system behaviour although it still meets its specification [19].*

Definition 9. *If a fault is found by a fault tolerant mechanism (described later), this fault has been **detected**, otherwise is **latent**, whether it is observable or not.*

Fault Propagation

Definition 10. *A **fault** is said to be **active** if its propagation generates more faults or failures, otherwise, it is said to be **dormant**.*

When a fault is detected in the system, it is very important to block its propagation, because if not, it may conclude in a chain reaction causing a catastrophe.

1.2.5 Fault Tolerant System Concept

Fault tolerance is one way to achieve dependability, which means that it is one way for the system to become available, reliable, safe, and secure. Fault tolerance can be applied in three levels [19]:

- *Hardware Level*, where fault tolerance has been used to compensate for faults in computing resources.
- *Software Level*, where Software fault tolerance compensates for faults such as changes in program or data structures due to transients or design errors.
- *System Level*, where the computer subsystem may provide functions that compensate for faults in other systems components that are not computer-based.

Definition 11. ***Fault Tolerant Systems*** *are systems which are able to handle an error or fault without affecting the service delivered [28].*

Fault tolerance involves:

1. *Fault detection.* A fault has occurred.
2. *Fault diagnosis.* What caused the fault.
3. *Fault isolation.* Prevents the propagation of faults.
4. *Fault masking.* Insuring that only correct values are passed to the system boundary.
5. *Fault compensation.* Provide a response to compensate for output of the faulty system.
6. *Fault Repair.* Faults are removed from a system. (Recovery process).

These steps may be covered or synthesized by the following:

1. **Fault Detection.** The process of observing the actual state of the controlled system and comparing it with its specifications in order to find discrepancies as early as possible.
2. **Fault Diagnosis.** The process of finding the original fault which caused the error or which can cause an error.
3. **Fault Recovery.** The process of applying proper corrective actions in order to prevent a possible future error, or reach and error free state.

In each of these subfields there are several principles to achieve these objectives, as well as different methods for representing the information needed. It will be discussed below.

1.2.6 Redundancy Management

Fault tolerance is sometimes called Redundancy Management; it is because Redundancy is the provision of functional capabilities that would be unnecessary in a fault-free environment.

The measure of success of fault tolerance is coverage (the probability of a system failure given that a fault occurs). Simplistic estimates measure redundancy by counting the number of redundant success paths in a system, which can be modelled by Markov models, or Petri Nets.

There exist several types of redundancy:

- *Space redundancy*, provides separate physical copies of a source, function, or data item.
- *Time redundancy*, if the expected faults are transient, a function can be rerun with a stored copy of the input data, and if information shifted in time can be analyzed for unwanted changes, it can be corrected to its original value.
- *Fault containment regions*, with few or no common dependencies between them, to prevent propagation.

1.2.7 Fault tolerance goals

The main goal of fault tolerance is to prevent faults from propagating to the system boundary, where it becomes observable and, hence, a failure. In general, the further a fault has propagated, the harder it is to deal with. Since fault tolerance is redundancy management, however it becomes a matter of the degree of redundancy desired. Finally, it is important to note that dealing with faults earlier rather than later may be the difference in the service the system delivers.

1.3 Fault Tolerance in Discrete Event Systems

1.3.1 Diagnosis Methods

The problem of fault diagnosis has received considerable attention in the literature of reliability engineering, control, and computer science, and several methods and techniques have been proposed.

The system is usually modelled using artificial intelligence and mathematical methods. These artificial intelligence methods are those knowledge-based [4], and agent-based [42], where the system behaviour is specified in form of rules and facts, obtained from empirical human observation. Mathematical methods are those process-model-based where the system behaviour is represented as a mathematical model such differential equations [21], state space models [21], FA-model [22] [25] [26], FSM-model [31] [40] [41], PN-model [14] [18] [34], CPN-model [7], and so on.

There exist a variety of studied techniques to perform diagnosis on the modelled system, which can be classified in two main groups: artificial intelligence and mathematical techniques. Artificial intelligence techniques are expert systems for inference [4] [27], neural network classifier [27], fuzzy logic for inference [27], and adaptive or neuro-fuzzy system for inference [27]. Mathematical techniques are signal models [20] [21] [30], parameter estimation [20] [21], state estimation [18] [20] [21] [22], observers [14] [21] [40] [41], parity equations [21], change detection [21] [34], symptom generation [21], modularity [17], fault trees [21], statistical classifiers [27], geometric classifiers [27], polynomial classifiers [27] and fuzzy logic for inference [27].

1.3.2 Related Work

The problem of diagnosis has been investigated by several researchers since different points of view.

Diagnosis approaches

- In [25], Lin studies off-line and on-line diagnostics. On-line diagnostic properties are state changing, events not all controllable, observation restricted to outputs, and constraint test sequence. While off-line diagnostic properties are state not changing, events all controllable, observation not restricted to outputs, and not constraint test sequence. He uses a nondeterministic Mealy automaton and a deterministic Moore automaton as models for off-line and on-line diagnostics, respectively, as first component, and a set of controllable events as second component. In addition, he partitions the state space into disjoint subsets, as desired. For off-line diagnostics, Lin assumes that the outputs are events observed and that all events are controllable. Based on the observable events and the desired partition, the state space is partitioned under every possible event. The conjunction of all these partitions must be finer than or as fine as the desired partition in order to determine a positive diagnosability. For on-line diagnostics, not all events are controllable. Lin says that a string of controllable events diagnoses a system with respect to a desired partition, if every pair of reached states having the same outputs belongs to the same desired partition. Therefore, if there exists such string then the system is on-line diagnosable. In [26], this theory is applied to an electronic engine control circuit. *Advantages:* Lin studies on-line and off-line diagnostic, and proposes an algorithm to find a minimal observable event set and an algorithm to find a control string that diagnoses the system. *Disadvantages:* Lin studies on-line and off-line diagnostics separately. For on-line diagnostics, he uses partial state observation and no event observation. For off-line diagnostics, he uses no state observation and partial event observation. The algorithm to find a control string has complexity of $O\left(2^{|Q|}\right)$ (Q is the set of states) in its worst case because is a reachability search.

- In [31], Magni et al. present a methodology for the development of a diagnostic system for complex industrial plants. The proposed technique consists in describing all the elements of the plant and the diagnostic system (plant, devices, sensors, actuators, diagnostic tests, hardware and analytical redundancies) as FSM models represented as event/state and output matrices. Then, they present a formal composition rule under the assumption that faults can occur only one at a time limiting the dimension of the resultant FSM model. This resultant output matrix, called matrix of residuals or fault signature, shows if faults are not distinguishable. If so, authors consider temporal information and present an algorithm to diagnose the system using the minimum and maximum times of the event or fault to occur and its detection by an alarm.

Diagnosability and observability approaches

- In [40] [41], Sampath et al. propose a DES approach to the problem of failure diagnosis, where the system is modelled as a FSM that describes normal and failed behaviour, which belong to the event set. If the system has more than one component, first, they build the model of every component and then, they compose these models. With the composite model and sensor maps, they generate the final composite system model including interactions among components and incorporating the sensor maps. Authors introduce the notion of diagnosability and I-diagnosability of DES. Roughly speaking, a system is diagnosable if it is possible to detect with a finite delay occurrences of failure events using the record of observed events. In addition, a system is said to be I-diagnosable if it is possible to diagnose failures, not always, but whenever the failure events are followed by certain observable indicator events that are associated with the failures. They also introduce the diagnoser which is a FSM built from the system model. They first assume that the system starts in a normal state, so they add a label N (Normal). They use the next observed event to make the transition to another state which it includes all possible states every one with its own label (N=Normal, F+#of fault=Fault, and/or I+#of indicator=Indicator) and so on. They use the diagnoser to perform on-line diagnosis. Moreover, they present additional modifications on the diagnoser in order to perform off-line diagnosis.

- In [17], García et al. present a modular decomposition as an approach for failure diagnosis based on DES. The methodology presented is based on DES and on the diagnoser concepts introduced by [40] [41], and starts with the individual modelling as FSM of all components that form the system. In order to introduce modular decomposition, authors present a new architecture, where the system is decomposed in subsystems, each one with its minimum local controller, which send information to a centralized modular diagnoser. This modular composition must consider the functional interrelation and the dimension of the models in order to have as low dependability [19] as possible. Authors also present a solution for the coupling problem of diagnoser that consists of including a conjunction function working only in coupling situations. This function uses non local information coming from the subsystem where a failure has taken place.

Observability and state estimation approaches

- In [18], Giua defines an observer as the system that computes the estimate of the actual marking of the net based on the observation of a word of events. Two algorithms are presented for estimating the state of the system assuming that only the net structure is known and considering two cases: when the initial marking is unknown and when the initial marking is a macromarking. The estimate computed as a result of these algorithms is always a lower bound on the actual marking of the net. Giua shows how this resultant estimate may be used to design a controller. This controlled is called state feedback control loop with observer which ensures that the system never enters a set of forbidden states, assuming complete controllability. The controller uses an algorithm to ensure that forbidden markings are not reached. *Advantages:* It is possible to estimate the actual marking event when the initial marking is unknown or it is a macromarking. *Disadvantages:* The algorithm used by the controller to ensure that forbidden markings are not reached is not optimal; it may also prevent transition firings that lead from legal markings to legal markings.

- In [14], Fanni et al. use the algorithms presented in [18] to estimate the actual marking of the system based on event observations. Therefore, they use this estimate to design a state feedback controller as described in [18]. The controlled system may result blocked, as a result of the estimate, hence the authors show with an example how to introduce time-out mechanisms to recovery from this blocking. This time-out mechanism consists in adding a place and a time-out transition. Thus, the timed transition will fire only if no other transition is enabled. Then, the observer, regardless of its previous marking, reconstructs its marking. *Advantages:* The consideration of mutual exclusion constraints as state specifications of the desired behaviour. *Disadvantages:* The authors do not present a general approach to automatically construct the time-out structure, they describe only an example.

Chapter 2

Modelling FMS with PN

Summary. *This chapter presents the PN fundamentals: the PN definition, their properties, and their implication in FMS. The IPN definition is also presented, an extension of PN, used as modelling formalism allowing the representation of measurable and non-measurable states and providing more compact representations than other formalisms. Finally, an extension of a modelling methodology for DES that obtains binary IPN models with faults is also presented.*

2.1 Petri Net Fundamentals

2.1.1 PN Definition

Definition 12. *A **Petri Net structure** G is a bipartite digraph represented by the 4-tuple $G = (P, T, I, O)$ where [29] [39].*

- $P = \{p_1, p_2, ..., p_n\}$ *is a finite set of vertices called* ***places****,*
- $T = \{t_1, t_2, ..., t_m\}$ *is a finite set of vertices called* ***transitions****,*
- $I : P \times T \longrightarrow \mathbb{Z}^+$ *is a function representing the* ***weighted arcs*** *going from places to transitions, and*
- $O : P \times T \longrightarrow \mathbb{Z}^+$ *is a function representing the weighted arcs going from transitions to places.* $\mathbb{Z}^+$*is the set of nonnegative integers.*

Usually $\cdot t_j$ denotes the set of all places p_i such that $I(p_i, t_j) \neq 0$ and $t_j\cdot$ the set of all places p_i such that $O(p_i, t_j) \neq 0$. Analogously, $\cdot p_i$ denotes the set of all transitions t_j such that $O(p_i, t_j) \neq 0$ and $p_i\cdot$ the set of all transitions t_j such that $I(p_i, t_j) \neq 0$. Graphically, places are represented by circles, transitions by rectangles, and arcs are depicted as arrows.

The **incidence matrix** of G is $C = [c_{ij}]$, where $c_{ij} = O(p_i, t_j) - I(p_i, t_j)$. The **marking** function $M : P \longrightarrow \mathbb{Z}^+$ is a mapping from each place to the nonnegative integers representing the number of tokens (depicted as dots) residing inside each place. The marking of a PN is usually expressed as an $n-$entry vector.

Definition 13. *A **p-invariant** Y of a PN is a rational-valued solution of equation $Y^T C = 0$. The **support of a p-invariant** Y_i is the set $\|Y_i\| = \{p \mid Y_i(p) \neq 0\}$.*

Definition 14. *A **Petri Net system** (PN) is the pair $N = (G, M_0)$, where G is a PN structure and M_0 is an initial token distribution.*

In a PN system, a transition t_j is **enabled** at marking M_k if $\forall p_i \in P, M_k(p_i) \geqslant I(p_i, t_j)$. An enabled transition t_j can be **fired** reaching a new marking M_{k+1} which can be computed as $M_{k+1} = M_k + Cv_k$, where $v_k(i) = 0$, $i \neq j$, $v_k(j) = 1$, this equation is called the PN **state equation**. The **reachability set** of a PN is the set of all possible reachable marking from M_0 firing only enabled transitions; this set is denoted by $\mathbf{R}(G, M_0)$.

Example 15. *Consider the PN of figure 2.1. For this net the PN System $N = (G, M_0)$, where $G = (P, T, I, O)$ and M_0 are given below:*

- $P = \{p_1, p_2, p_3, p_4, p_5\}$;
- $T = \{t_1, t_2, t_3, t_4, t_5\}$;
- $I(p_1, t_1) = 1$, $I(p_5, t_1) = 1$, $I(p_2, t_2) = 1$, $I(p_3, t_3) = 1$, $I(p_5, t_3) = 1$, $I(p_4, t_4) = 1$, $I(p_3, t_1) = 1$, *and* $I(p_i, t_j) = 0$ *for the rest of pairs* (p_i, t_j), *with* $i, j = 1, .., 5$;
- $O(p_2, t_1) = 1$, $O(p_3, t_2) = 1$, $O(p_5, t_2) = 1$, $O(p_4, t_3) = 1$, $O(p_1, t_4) = 1$, $O(p_5, t_4) = 1$, $O(p_1, t_5) = 1$, *and* $O(p_i, t_j) = 0$ *for the rest of pairs* (p_i, t_j), *with* $i, j = 1, .., 5$;

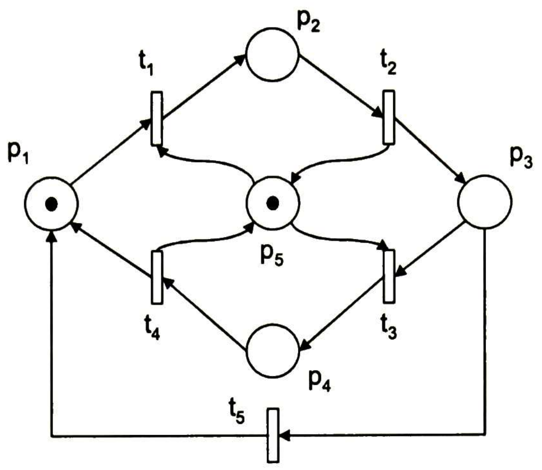


Figure 2.1: A Petri Net

- *$M_0(p_1) = 1$, $M_0(p_5) = 1$ and $M_0(p_2) = M_0(p_3) = M_0(p_4) = 0$.*

The input function matrix is $I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$

The output function matrix is $O = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$

The incidence matrix is $C = \begin{bmatrix} -1 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \\ -1 & 1 & -1 & 1 & 0 \end{bmatrix}$

The initial marking is $M_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \end{bmatrix}^T$

There is only one enabled transition, which is transition t_1. After the firing of the transition t_1, the resulting PN marking is $M_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T$ where the next enabled transition is t_2. If transition t_2 is fired, the resulting PN marking is $M_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 \end{bmatrix}^T$

2.1.2 PN Properties

Petri nets models may exhibit several properties: *Behavioural properties* that depend on the initial marking of a *PN*, and *Structural properties* that do not depend on the initial marking of a *PN*, but they do depend on the *PN* topology.

Let us review the most important properties.

Definition 16. *Given a $N = (G, M_0)$, a marking M is* ***reachable*** *from marking M_0 if there exists a sequence of transitions firings that transforms M_0 to M. The reachability set $\mathbf{R}(G, M_0)$ is formed by all the reachable markings M_i from M_0.*

Reachability is considered a behavioural property, since it depends on the initial marking.

Definition 17. *Given a $N = (G, M_0)$, and its reachability set $\mathbf{R}(G, M_0)$, a place $p \in P$ is* **$\mathbf{B}-$bounded** *if $M(p) \leq B$, $\forall M \in \mathbf{R}(G, M_0)$, where B is a positive integer. PN is $B-$bounded if each place in P is $B-$bounded.* ***Safeness*** *is $1-$bounded. G is* ***structurally bounded*** *if G is bounded given any finite initial marking M_0.*

Definition 18. *A* ***transition*** *t is* ***live*** *if at any marking $M \in \mathbf{R}(G, M_0)$, there is a sequence of transitions whose firing reach a marking that enables t or, that there is a transition firing sequence that includes t. A* **PN** *is* ***live*** *if every transition in it is live. A PN is* ***structurally live*** *if there is a finite initial marking that makes the net live.*

Definition 19. *A* ***transition*** *t is* ***dead*** *if there is an $M \in \mathbf{R}(G, M_0)$, such that there is no sequence of transition firings to enable t starting from M. A PN contains a* ***deadlock*** *if there is an $M \in \mathbf{R}(G, M_0)$ at which no transition is enabled. Such a marking is called a* ***dead marking****.*

Definition 20. *A PN* ***System*** *$N = (G, M_0)$ is said to be* ***reversible*** *if $\forall M \in \mathbf{R}(G, M_0)$, $M_0 \in \mathbf{R}(G, M)$. That is, the fact that from every marking reached from the initial marking, there is a transition firing sequence that ends up with the initial marking. A PN is said to be* ***structurally reversible*** *if there is a finite initial marking that makes the net reversible.*

The properties just mentioned (Boundedness/safeness, liveness, reversibility) are independent each other. A PN may have or not any of them.

Definition 21. *A PN with an initial marking M_0 is strictly* ***conservative*** *iff $\forall M \in \mathbf{R}(G, M_0)$,*

$$\sum_{i=1}^{|p|} M(p_i) = \sum_{i=1}^{|p|} M_0(p_i)$$

This means that $|\ \ t| = |t \cdot| \ \forall t \in T$.

In other words, a PN is conservative if the number of tokens in the initial marking does not change through the firing of transitions.

Definition 22. *A PN is* ***repetitive*** *iff there exists a marking M and a firing sequence s such that:*

- *After the firing of the sequence s from M, it ends up in M. This is $M \xrightarrow{s} M$.*
- *Every transition occurs in s.*

Definition 23. *A PN is said to be* ***consistent*** *if there exists a marking M, and a firing sequence s from M that ends up with M, such that every transition occurs at least once in s.*

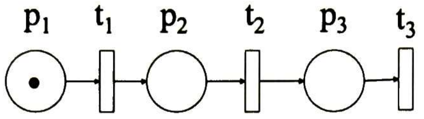


Figure 2.2: Sequential Relation

2.1.3 PN's interpretation in FMS

According with [45], there are two interpretations of *PN* in the context of *FMS*. In the *first interpretation,* **places** model resource status and operations, **transitions** model the start and/or end of operations, **directed arcs** model material, resource, information, and/or control flow direction, and **tokens** model material, resources, or information. In the *second interpretation,* **places** model resource status and conditions, **transitions** model operations, processes, activities, and events, **directed arcs** model material, resource, information, and/or control flow direction, and **tokens** model material, resources, or information

There exists a modelling convention with *PN* for *FMS*. For every concept in manufacturing, a *PN* modelling is presented in the following [45]:

- *Moving of production lot size.* Weight of directed arcs models the production size that must be moved.
- *Number of resources (AGVs, machines, workstation, and robots).* The number of tokens in places models quantity of the corresponding resource.
- *Capacity of a workstation.* The number of tokens in places models the availability of the workstation.
- *Work-in-process.* The number of tokens in places models the buffers and operations of all machines.
- *Production volume.* The number of tokens in places models the counter for or the number of firings of transitions modelling the end of a product.
- *The time of an operation (setup, processing, and loading).* Time delays associated with the place or transition modelling the operation.
- *Conveyance or transportation time.* Time delays associated with the directed arc, place or transition models the conveyance or transportation.
- *System state. PN* marking (plus the timing information for timed *PN*)
- *Sequence.* One operation follows the other. It is shown in 2.2.
- *Concurrency.* Two operations are initiated by one event forming one parallel structure starting with a transition. It is shown in 2.3
- *Conflict.* If any of two operations can indistinctly follows an operation. It is shown in 2.4

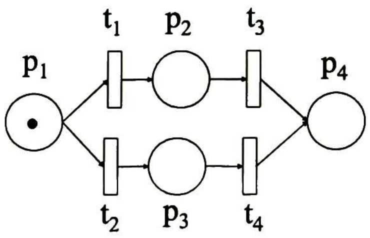


Figure 2.3: Concurrent Relation

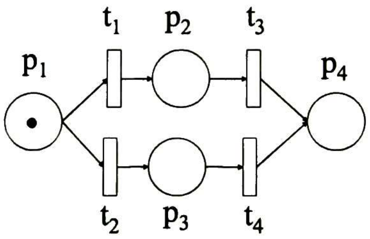


Figure 2.4: Conflicting Relation

- *Mutually exclusive.* Processes can not be performed at the same time due to a shared resource. There are parallel and sequential mutual exclusion shown in 2.5 and 2.6, respectively.

The PN properties have an interpretation in the context of the modelled manufacturing systems and it is described below [45]:

- *Reachability.* A certain state can be reached from the initial conditions.
- *Boundedness.* No capacity (of buffer, storage area, and workstation) overflows.
- *Safeness* Availability of a single resource; or no request to start an ongoing process.
- *Conservativeness.* Conservation of non-consumable resources, as machines and AGVs.
- *Liveness.* Freedom form deadlock and guarantee the possibility of a modelled event, operation, process or activity to be ongoing.

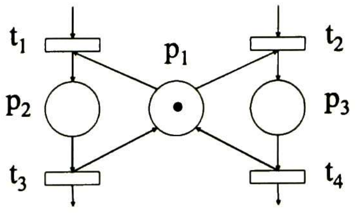


Figure 2.5: Parallel Mutual Exclusion Relation

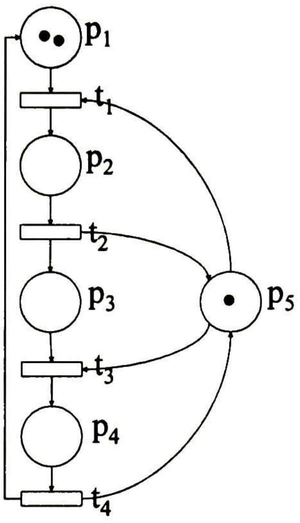


Figure 2.6: Sequential Mutual Exclusion Relation

- *Reversibility.* Re-initialization and cyclic behaviour.
- *Repetitiveness.* Existence of repetitive operations/activities/events for some marking.
- *Consistency.* Existence of cyclic behaviour for some marking.

See [33] for a deeper revision of properties, analysis and applications of PN.

2.1.4 IPN Formal Definition

Now, Interpreted Petri Nets (IPN) [1] [2] [32], the extension to PN used in this work as modelling formalism, is given. This extension allows to associate input and output alphabets to PN models.

Definition 24. *An **Interpreted Petri Net** (IPN) is the 5-tuple $Q = (N, \Sigma, \Phi, \lambda, \varphi)$ where*

- *$N = (G, M_0)$ is a* **PN** *system,*
- *$\Sigma = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is the **input alphabet** of the net, where α_i is an **input symbol**,*
- *$\Phi = \{\zeta_1, \zeta_2, ..., \zeta_n\}$ is the **output alphabet** associated to places, where ζ_i is an **output symbol**,*
- *$\lambda : T \rightarrow \Sigma \cup \{\varepsilon^i\}$ is a **labelling function** of transitions with the following restriction:*
    - *$\forall t_j, t_k \in T$, $j \neq k$ if $I(p_i, t_j) = I(p_i, t_k) \neq 0$ and both $\lambda(t_j)$, $\lambda(t_k) \neq \varepsilon^i$, then $\lambda(t_j) \neq \lambda(t_k)$. In this case ε^i represents an internal system event,*
- *$\varphi : \mathbf{R}(G, M_0) \rightarrow \{\Phi\}^q$ is an **output function**, where $\mathbf{R}(G, M_0)$ is the **reachability set** defined as in the PN and q is the number of available outputs associated to places.*

Remark. *From now on, (Q, M_0) will be used instead of $Q = (N, \Sigma, \Phi, \lambda, \varphi)$ to emphasize the fact that there is an initial marking in an IPN.*

Remark. *The function φ is linear and can be represented as a $q \times n$ matrix. $\varphi = [\varphi_{ij}]$, where the $i-th$ row vector φ_i is the transpose of an elemental vector e_j, and n is the number of places.*

Definition 25. *A transition $t \in T$ is said to be* ***controllable,*** *if $\lambda(t_j) \neq \varepsilon^i$, and* ***uncontrollable,*** *otherwise.*

Definition 26. *A place $p_i \in P$ is said to be* ***measurable,*** *if $\exists j \in \{1, 2, ..., r\}$ such that $\varphi_j = e_i^T \neq 0$, where φ_j is the j-th row of φ; otherwise, p_i is called* ***non-measurable.*** *The set $P_m = \{p \mid \exists j \in 1, 2, ..., r$ such that $\varphi_j = e_i^T \neq 0\}$ is the* ***set of measurable places*** *and $P_{nm} = P \backslash P_m$ is the* ***set of non-measurable places.***

Remark. *A measurable place is depicted as an unfilled circle, while a non-measurable place is depicted as a filled circle.*

Let $\sigma = t_i t_j t_k ...$ be a firing transition sequence. The **Parikh vector** $\overrightarrow{\sigma} : T \longrightarrow \mathbb{Z}^+$ of σ maps every transition t of T (the transitions of the net) to the number of occurrences of t in σ. In an IPN, a transition $t_j \in T$ is **enabled** at a marking M if $\forall p_i \in P$, $M(p_i) \geq I(p_i, t_j)$. However, if t_j is an uncontrollable transition then t_j can be **fired**; otherwise, the input symbol $\lambda(t_j) = \alpha_i \neq \varepsilon$ must be present in order to **fire** t_j.Using the Parikh vector, the **state equation** of an IPN can be written as:

$$\begin{aligned} M_{k+1} &= M_k + C\overrightarrow{\sigma} \\ y_k &= \varphi M_k \end{aligned} \tag{2.1}$$

This work assumes that just one transition is fired at a time.

Definition 27. *Let (Q, M_0) be an IPN. The* ***firing language*** *of (Q, M_0) is $\pounds(Q, M_0) = \{\sigma \mid \sigma = t_i t_j ... t_k \wedge M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} ... \xrightarrow{t_k} M_k\}$.*

Definition 28. *The* ***input language*** *of (Q, M_0) is $\pounds_{in}(Q, M_0) = \{\lambda(t_i)\lambda(t_j) ... \lambda(t_k) | t_i t_j ... t_k \in \pounds(Q, M_0)\}$. The* ***output language*** *of (Q, M_0) is $\pounds_{out}(Q, M_0) = \{\varphi(M_0)\varphi(M_1)...\varphi(M_w)... \mid M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} ...M_w \xrightarrow{t_k} M_k \wedge t_i t_j ... t_k \in \pounds(Q, M_0)\}$*

Definition 29. *Let (Q, M_0) be an IPN and $w \in \pounds_{in}(Q, M_0)$ be an input word of the net. The* ***set of firing sequences generated by*** *w is $\Gamma_w = \{\sigma = t_i t_j ... t_k \in \pounds(Q, M_0) \mid \lambda(t_i)\lambda(t_j)...\lambda(t_k) = \omega\}$. The* ***set of output words generated by*** *w is*

$$\Psi_w = \{\psi = \varphi(M_0)\varphi(M_1)...\varphi(M_k) \mid M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} ... \xrightarrow{t_k} M_k \wedge t_i t_j ... t_k \in \Gamma_w\}.$$

Example 30. *Consider the IPN shown in figure 2.7. For this net, the 5-tuple $Q = (N, \Sigma, \Phi, \lambda, \varphi)$ is given below:*

- *N is the PN System described in example 15.*
- *$\Sigma = \{Start,\ Stop\}$*

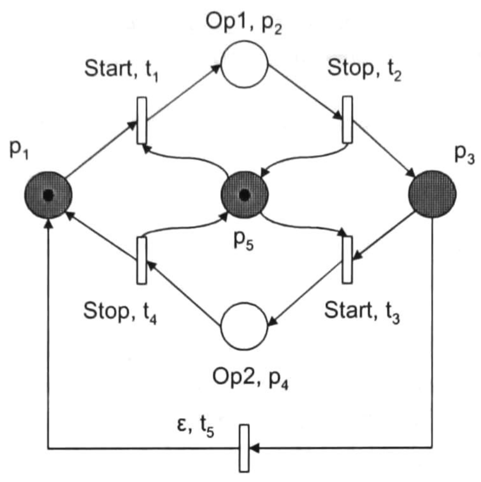


Figure 2.7: An Interpreted Petri Net

- $\Phi = \{Op1, Op2\}$
- $\lambda(t_1) = Start,\ \lambda(t_2) = Stop,\ \lambda(t_3) = Start,\ \lambda(t_4) = Stop, \lambda(t_5) = \varepsilon.$
- $\varphi = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Transitions t_1, t_2, t_3, t_4, are controllable since $\lambda(t_i) \neq \varepsilon^i$, $1 \leq i \leq 4$. Hence, transition t_5 is uncontrollable since $\lambda(t_5) = \varepsilon$.

Places p_2 and p_4 are both measurable places since the transpose of the elemental vector e_2 and e_4 of φ are not null. While places p_1, p_3, p_5 are non-measurable since e_2^T and e_4^T of φ are null.

There is only one enabled transition at marking M_0, which is transition t_1. If the input symbol Start is present in the system, then t_1 is fired resulting in a new marking M_1 and a new output signal y_1. This dynamic is described by the state equation 2.1 as described in the following: $M_1 = \begin{bmatrix}1 & 0 & 0 & 0 & 1\end{bmatrix}^T + C\begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix}^T = \begin{bmatrix}0 & 1 & 0 & 0 & 0\end{bmatrix}^T$ and $y_1 = \varphi\begin{bmatrix}0 & 1 & 0 & 0 & 0\end{bmatrix}^T = \begin{bmatrix}1 \\ 0\end{bmatrix}$.

The firing language of (Q, M_0) is described by $\pounds(Q, M_0) = (t_1t_2t_3t_4 \cup t_1t_2t_5)^*$ Hence, the input language of (Q, M_0) is described by $\pounds_{in}(Q, M_0) = (StartStop)^+$ and the output language is described by $\pounds_{out}(Q, M_0) = (Op1Op2 \cup Op1)^*$.

Consider $w = StartStop$. Then, the set of firing sequences generated by w is $\Gamma_w = \{t_1t_2, t_1t_2t_5\}$ and the set of output words generated by w is $\Psi_w = \{Op1\}$.

The following definitions relate the input sequence $\alpha_0\alpha_1 \cdots \alpha_n$ with its respective produced output sequence $y_0y_1 \cdots y_n$ in an *IPN*.

Definition 31. *A **sequence of input-output symbols** of (Q, M_0) is a sequence $\omega = (\alpha_0, y_0)(\alpha_1, y_1) \cdots (\alpha_n, y_n)$, where $\alpha_j \in \Sigma \cup \{\varepsilon\}$ and α_{i+1} is the current input of the IPN* *when the output changes from y_i to y_{i+1}. It is assumed that $\alpha_0 = \varepsilon$, $y_0 = \varphi(M_0)$ and (α_{i+1}, y_{i+1}) belongs to the sequence when:*

- *(α_i, y_i) belongs to the sequence,*
- *$y_{i+1} \neq y_i$, and*
- *there exists no $y_j \neq y_i$, $y_j \neq y_{i+1}$ occurring after the occurrence of y_i and before the occurrence of y_{i+1}.*

Definition 32. *If $\omega = (\alpha_0, y_0)(\alpha_1, y_1)\cdots(\alpha_n, y_n)$ is a sequence of input-output symbols, then the* ***transition sequence*** *$\sigma \in \pounds(Q, M_0)$* ***whose firing*** *actually* ***generates*** *ω is denoted by σ_ω. The set of all* ***possible firing transition sequences that could generate the word*** *ω is $\Omega(\omega) = \{\sigma | \ \sigma \in \pounds(Q, M_0) \wedge$ the firing of σ produces $\omega\}$.*

Definition 33. *Let (Q, M_0) be an IPN. The set of all* ***sequences of input-output symbols*** *of (Q, M_0) will be denoted by $\Lambda(Q, M_0)$. The set of all* ***input-output sequences of length greater or equal than*** *n will be denoted by $\Lambda^n(Q, M_0)$, i.e., $\Lambda^n(Q, M_0) = \{\omega \in \Lambda(Q, M_0) \,|\, |\omega| \geq n\}$.*

Definition 34. *The* ***set of all input-output sequences that leads to an ending marking*** *in the IPN (markings enabling no transition or only self-loop transitions) is denoted by $\Lambda_B(Q, M_0)$, i.e., $\Lambda_B(Q, M_0) = \omega \in \Lambda(Q, M_0) \,|\, \exists \sigma \in \Omega(\omega)$ such that $M_0 \xrightarrow{\sigma} M_j$ and if $M_j \xrightarrow{t_i}$ then $C(\bullet, t_i) = 0\}$.*

Definition 35. *Let $\omega = (\alpha_0, y_0)(\alpha_1, y_1)\cdots(\alpha_n, y_n) \in \Lambda(Q, M_0)$ be a sequence of input-output symbols. The* ***marking sequence set*** *corresponding to ω is defined as*

$$\begin{aligned} I_\omega = \{M_0 M_1 \cdots M_k | M_i \in \mathbf{R}(Q, M_0), \\ M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} \cdots \xrightarrow{t_m} M_k \wedge \sigma_\omega = t_i t_j \cdots t_m \in \Omega(\omega)\} \end{aligned} \tag{2.2}$$

2.1.5 Event-Detectability property

The event-detectability property of an IPN is definitely essential, as it will be seen in chapter 3, in the context of the diagnosability property the focus of this thesis. Therefore, it is important to define it.

Definition 36. *An IPN given by (Q, M_0) is* ***event-detectable*** *if any transition firing can be uniquely determined by the output signals that it produces [2].*

The following lemma states when an IPN is event-detectable [2].

Lemma 37. *A live IPN given by (Q, M_0) is event-detectable if and only if*

1. *$\forall t_i, t_j$ it holds that $\varphi C(\bullet, t_i) \neq \varphi C(\bullet, t_j)$ and*
2. *$\forall t_k \in T$ it holds that $\varphi C(\bullet, t_k) \neq 0$.*

For further information about the proof of this lemma see [2].

Example 38. *Consider the IPN (Q, M_0) shown in figure 2.7. Based on lemma 37, there are two conditions to be analyzed in order to define event-detectability:*

1. *$\forall t_i, t_j$ it holds that $\varphi C(\bullet, t_i) \neq \varphi C(\bullet, t_j)$ and,*
2. *$\forall t_k \in T$ it holds that $\varphi C(\bullet, t_k) \neq 0$.*

This means that all columns of $\varphi \bullet C$ must be not null and different from each other. Let φ and C the output functions matrix and the incidence matrix described in examples 30 and 15, respectively. Therefore, $\varphi \bullet C = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \end{bmatrix}$

In this case, it can be concluded by the result of $\varphi \bullet C$ that (Q, M_0) is not event-detectable because there exists one null column. This is because transition t_5 cannot be detected when (Q, M_0) changes from p_3 to p_1 by observing the outputs. Therefore, if the output signal OS is added either to place p_1 or place p_3, then transition t_5 can be detected by observing the outputs.

2.1.6 Marking-Detectability property

Another property that is useful in the design of diagnoser nets is the marking detectability [37].

Definition 39. *Let (Q, M_0) be an IPN. (Q, M_0) is marking detectable if $\forall \omega \in \pounds_{in}(Q, M_0)$ $\exists z$ such that $\omega z \in \pounds_{in}(Q, M_0)$, $\mid z \mid < k < \infty$, where $M_0 \overset{\omega z}{\to} M_i$ and M_i can be computed.*

In order to characterize an IPN exhibiting this property, the concepts of conservative marking laws and synchronic distance are now introduced [37].

Definition 40. *Let (Q, M_0) be an IPN and $M(p_j)$ be any reachable marking of place p_j in (Q, M_0). The set of equations $CML = \{\sum_{j=1}^{n} \alpha_j^i \cdot M(p_j) = k_i \mid i \in [1, ..., s], \alpha_j^i \in Z^+\}$ where $\forall \alpha_k^i \neq 0$, it holds that k_i/α_k^i is an integer value, form a set of conservative marking laws when all nonmeasurable places of the net are contained in at least one equation (i.e., if p_j is nonmeasurable, then $\exists_i$ such that $\alpha_j^i \neq 0$).*

The upper marking bound K_j for a nonmeasurable place p_j is defined as $K_j = \min\{k_i/\alpha_j^i \mid \alpha_j^i \neq 0, i \in [1, ..., s]$. Note that K_j is always defined.

Definition 41. *The synchronic distance of transition t_i, with respect to transition t_j in an IPN (Q, M_0), is the maximum value of the difference between the number of firings of t_i and t_j, considering all possible firing sequences. This is expressed as:*
$$SD(Q, M_0; t_i, t_j) = \max_{\sigma \in \pounds(Q, M_0)} \{\vec{\sigma}(t_i) - \vec{\sigma}(t_j)\}.$$

In [37], the concept of synchronic distance is extended to a set of transitions in the following sense. If S_1, S_2 are sets of transitions, then $SD(Q, M_0; S_1, S_2)$ means the maximum value of the difference between the number of firings of transitions in S_1 and transitions S_2, considering all possible firing sequences.

The next theorem, defined in [37], characterizes the IPN exhibiting the marking detectable property.

Theorem 42. *Let (Q, M_0) be a cyclic, live and bounded IPN where a CML is defined. If (Q, M_0) is event detectable and $\forall p_j$ nonmeasurable place it holds that $SD(Q, M_0; \cdot p_j, p_j \cdot) \geq K_j$, then (Q, M_0) is marking detectable.*

For further information about the proof of this theorem see [37].

2.1.7 Observability

This property definition deals with the possibility of finding out the system initial state in a finite time using only the knowledge of the system structure, input and output.

In this work, the observability property [38] is used to characterize the diagnosability property as described later.

Definition 43. *An IPN given by $(Q, \mathcal{M}_0)$ is **observable** if there exists an integer $k < \infty$ such that $\forall \omega \in \Lambda^k (Q, \mathcal{M}_0)$ it holds that the information provided by ω and $(Q, \mathcal{M}_0)$ suffices to uniquely determine the initial marking M_0 and the marking M_i reached by the firing of the underlaying transition firing sequence σ_ω.*

2.2 The Modelling Methodology

The model of a FMS is represented using binary IPN. This modelling methodology [3] [38] is based on the identification of the components of the FMS, then it includes the description of the relationships among them using two operators, synchronic and permissive compositions, and finally, adding an interpretation with the association of input symbols to transitions and output symbols to places. The extension here presented is specifically to model the probably faults that the system can have or the ones that are important for us to know about their occurrence.

This modelling methodology includes three steps. The extension to model faults is made on step 1 as described in the following:

1. **Set of values.** For each state variable $sv_j^i \in \text{STATE\_VARIABLES}_i$, the set $\text{VALUE}_{sv_j^i}$ includes all val_p^{ij} of fault that sv_j^i can have.

2. **Codification.** Because of the existence of faulty values, the set of places can be partitioned into the subsets $P^F_{sv_j^i}$ and $P^N_{sv_j^i}$, representing the faulty and normal values of the state variable sv_j^i, respectively.

3. **Event Modelling.** Each place $p_n^{ij} \in P^F_{sv_j^i}$ must have only one input transition and self-loops transitions representing all the events related with this state variable. We denote by P^F the set of all faulty places, i.e. $P^F = \bigcup_{sv_j^i} P^F_{sv_j^i}$.

Remark. *In this work, for every faulty place $p_i \in P^F$ $\varphi(p_i) = \varphi(p_k)$, where $p_k \subseteq \cdot \cdot p_i$. This case considers the worst one, i.e. when the system enters a fault state and the sensors do not show it. However, this is not the only case it can be considered.*

Remark. *Remember that there is only one input transition to every faulty place.*

Remark. *For the complete modelling methodology see [3] [38].*

In order to exemplify the modelling methodology introduced by [38] with the extension to model faults here established, it will be presented the following case.

2.2.1 Application of the extended modelling methodology

Case Study 1: "Manufacturing couplers and brackets"

System Description

This *FMS* processes two different parts, couplers and brackets. The two original process plans are described in the following.

The original process plan for a Coupler is:

1. Chuck the part on 0.900 diameter.
2. Face one side.
3. Turn 0.750 X 0.500 diameter segment.

The process plan for a Bracket is:

1. Mill outside dimensions ensuring a 2.300 X 0.95 X 0.950 part.

In order to compliance with this process plans, the following components are needed:

- An engine lathe or turning machine, as generic name, for facing and turning processes.
- A milling machine.
- Two conveyors.
- One robot.
- A part selector.

The system layout is depicted in figure 2.8, and the process of a coupler is described as follows:

1. Robot takes the part from the input pallet to the engine lathe.
2. Engine lathe performs the corresponding processes of facing and, then, turning on the part.
3. Robot takes the part from the engine lathe to the output pallet.
4. The part is finished.

The process of a bracket is described as follows:

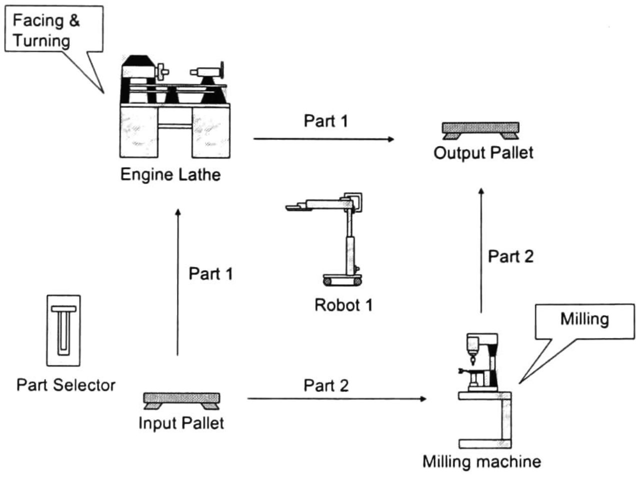


Figure 2.8: System1: Layout

1. Robot takes the part from the input pallet to the milling machine.
2. Milling machine performs the corresponding process of milling on the part.
3. Robot takes the part from the milling machine to the output pallet.
4. The part is finished.

System Model

The *FMS* model is represented by a binary *IPN* which it will be built up following the modelling methodology introduced by [38] with the extensions to model faults.

Step 1: Building PN The model methodology is based on building *PN* modules representing the behaviour of a system component

Applying algorithm *Building-PN*

1. *System components.* The identified system components are the engine lathe, the milling machine, the robot, the buffers and the part selector. Therefore, it is defined the following set. SYSTEM_COMPONENTS= {*EngineLathe, MillingMachine, Robot, InputPallet, OutputPallet, Selector*}.
2. *State variables.* The STATE_VARIABLES *EL*, *MM*, *R*, *IP*, *OP*, *S* are associated to the *EngineLathe*, *MillingMachine*, *Robot*, *InputPallet*, *OutputPallet*, *Selector* system components, respectively.
3. *Set of values.* The sets VALUE$_{EL}$ = { *EL_available, EL_facing, EL_turning, EL_fault In_facing, EL_faultIn_turning*}, VALUE$_{MM}$ = {*MM_available, MM_milling, MM_*

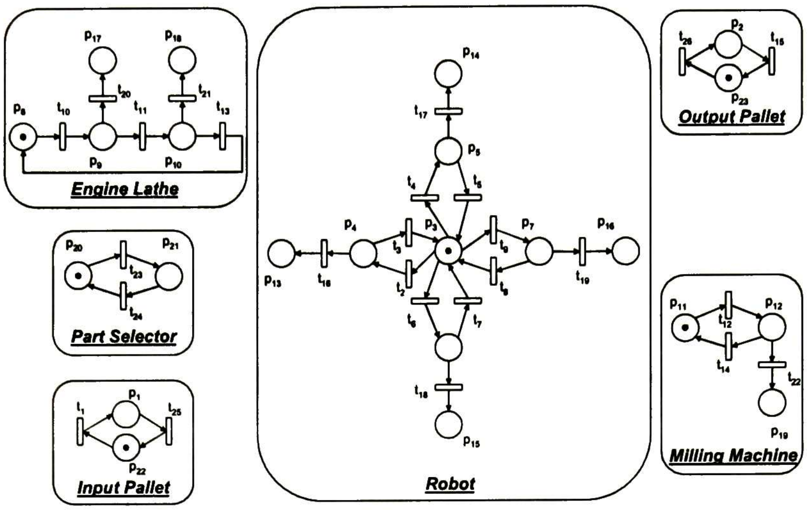


Figure 2.9: System1: *PN* Modules

faultIn_ milling}, VALUE$_R$ = {*R_ available, R_ operation1a, R_ operation2a, R_faultIn _operation1a, R_faultIn_ operation2a, R_ operation1b, R_ operation2b, R_faultIn_ operation1b, R_faultIn_ operation2b*}, VALUE$_{IP}$ = {*IP_ available, IP_ notavailable*}, VALUE$_{OP}$ = {*OP_ available, OP_ notavailable*}, VALUE$_S$= {*S_ p1, S_ p2*} are the values that the state variables *EL, MM, R, IP, OP,* and *S* can take, respectively. Observe that every state variable includes its own fault values.

4. *Codification.* Places p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}, p_{20}, p_{21}, p_{22}, p_{23} correspond to the values *IP_ available, OP_ available, R_ available, R_ operation1a, R_ operation1b, R_ operation2a, R_ operation2b, EL_ available, EL_facing, EL_ turning, MM_ available, MM_ milling, R_faultIn_ operation-1a, R_faultIn_ operation1b, R_faultIn_ operation2a, R_faultIn_ operation2b, EL_fault-In_facing, EL_faultIn_ turning, MM_faultIn_ milling, IP_ notavailable, OP_ not available,* respectively.

5. *Event modelling.* Transitions t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26} are created to represent the change of values. In addition, for a shorter representation of the self-loops of every $p_n^{ij} \in P^F_{sv^i_j}$, they will be omitted.

6. *Initial Marking.* In order to obtain the initial marking of the modules, it is assumed that the initial state of the system is *EL=EL_ available, MM=MM_ available R=R_ available, S=S_ p1, IP_ notavailable, OP_ notavailable.* Therefore, the initial marking is $M_0(p_3) = 1$, $M_0(p_8) = 1$, $M_0(p_{11}) = 1$, $M_0(p_{20}) = 1$, $M_0(p_{22}) = 1$, $M_0(p_{23}) = 1$ and $M_0(p_i) = 0$, i =1, 2, 4,..., 7, 9, 10, 12,..., 19, 21.

7. *Output.* The output of this algorithm is a set of *PN* modules shown in figure 2.9.

Up to now, isolated PN modules G^{ij} for each state variable sv^i_j have been built. Notice that all these models are state machines.

Step 2: PN Modules Composition The state variable PN module composition is made through two basic operators: synchronic and the permissive composition. Synchronic composition is used to establish a relation among two or more modules containing transitions representing the same event i.e. this composition merges two or more transitions when they have the same physical meaning. Permissive composition is used to establish a relation among two or more modules when the marking of a place enables the firing of a transition without modify the marking of this places. See [38] for the formal introduction of this operators.

Applying algorithm *PN Module-composition.*

1. *Labelling modules.* The transitions representing the start of the facing operation at the engine lathe EL get the same label $tlab(t_3) = tlab(t_{10}) = x_3$. Also, the transitions representing the start of the milling operation at the milling machine MM get the same label $tlab(t_7) = tlab(t_{12}) = x_7$. The transitions representing the stop of the turning operation at the engine lathe EL get the same label $tlab(t_4) = tlab(t_{13}) = x_4$. Analogously, $tlab(t_8) = tlab(t_{14}) = x_8$ that represents the stop of the milling operation at the milling machine MM. The transition t_{26} representing the unload of the part to the output pallet OP has two different physical meaning, therefore it is replicated into two different transitions, t_{26a} for the unload of part 1 and t_{26b} for the unload of part 2. The transitions representing the unload of the part 1 to the output pallet OP get the same label $tlab(t_5) = tlab(t_{26a}) = x_5$. The transitions representing the unload of the part 2 to the output pallet OP get the same label $tlab(t_9) = tlab(t_{26b}) = x_9$. The transition t_{25} representing the load of a part from the input pallet IP has two different meaning, the load to the engine lathe EL and the load to the milling machine MM, therefore it is replicated into two different transitions, t_{25a} for the load to the engine lathe and t_{25b} for the load to the milling machine. Since x_2 is associated with the loading of a part 1 from the input pallet IP to the engine lathe EL and p_{20} represents that the system is processing parts 1, then $plab(p_{20}) = \{x_2\}$. Analogously, $plab(p_{21}) = \{x_6\}$. In figure 2.10, the modules after the application of the *tlab* and *plab* functions are shown.

2. *Module composition.* In the synchronic composition of all modules, transitions with the same label fusion in one transition. In the permissive composition, for every label l_i in a place, an arc is added from this place to every transition labelled as l_i and viceversa.

3. *Output.* The output of this algorithm is a labelled PN, shown in 2.11, which represents the behaviour of the FMS.

Step 3: Building IPN Finally, the next algorithm performs the task of building the IPN.

Consider the output of the algorithm *PN Module-composition* shown in figure 2.11 to apply the algorithm IPN building.

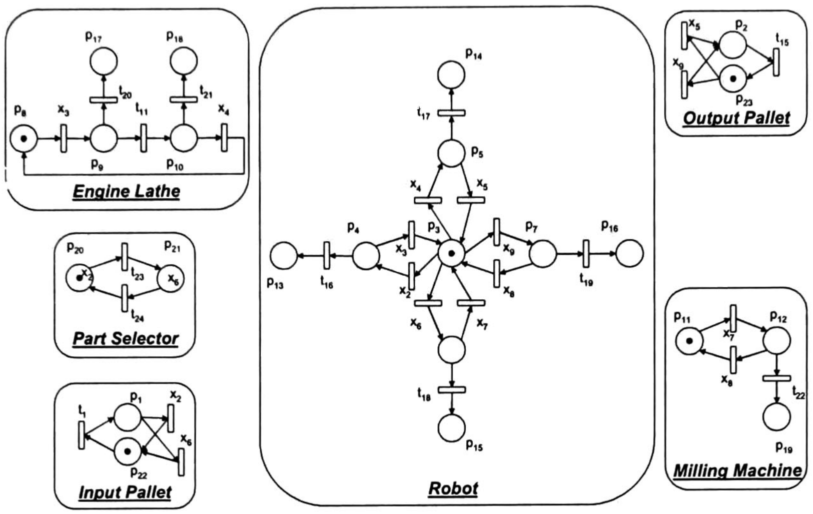


Figure 2.10: System1: *tlab* and *plab* Functions

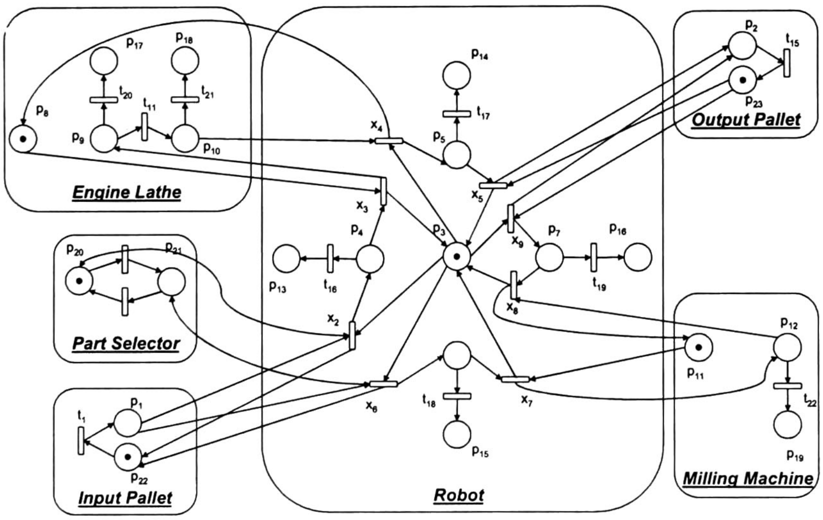


Figure 2.11: System1: Labelled PN

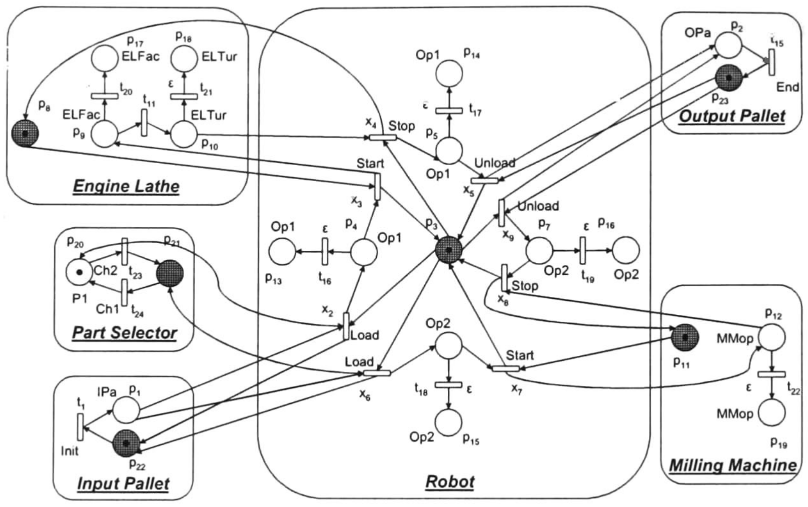


Figure 2.12: Systeml: *IPN*

Applying algorithm *IPN Building*

1. *Input and output IPN functions.* The input function λ associate to each transition in the model, the input symbol, if any, used to change the state variable value. The set of input symbols is $\Sigma = \{Init, Load, Start, Stop, Load, Unload, FacTur, End, Ch2, Ch1\}$. Then, $\lambda(t_1) = Init$, $\lambda(x_2) = Load$, $\lambda(x_3) = Start$, $\lambda(x_4) = Stop$, $\lambda(x_5) = Unload$, $\lambda(x_6) = Load$, $\lambda(x_7) = Start$, $\lambda(x_8) = Stop$, $\lambda(x_9) = Unload$, $\lambda(t_{11}) = FacTur$, $\lambda(t_{15}) = End$, $\lambda(t_{23}) = Ch2$, $\lambda(t_{24}) = Ch1$ and $\lambda(t_{15}) = \lambda(t_{16}) = \lambda(t_{17}) = \lambda(t_{18}) = \lambda(t_{19}) = \lambda(t_{20}) = \lambda(t_{21}) = \lambda(t_{22}) = \varepsilon$. The set of output symbols is $\Phi = \{IPa, OPa, Op1, Op2, ELFac, ELTur, MMop, P1\}$. Therefore, the output function is represented by the matrix φ.

$$\varphi = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0 \end{bmatrix}$$

2. *Output.* The output of this algorithm is the *IPN* shown in figure 2.12, which represents the behaviour of the *FMS*, including the input commands and output signals.

The complete **modelling methodology** consists then in the following steps:

1. Apply *Building-PN* algorithm.
2. Apply *PN Module-composition* algorithm.
3. Apply *IPN-building* algorithm.

Chapter 3

Fault Detection and Diagnosis of FMS

Summary. *This chapter defines the diagnosability property in IPN, its characterization from a language point of view and its structural characterization. Moreover, it also presents general scheme for diagnosis on line. Finally, it exemplifies the test of the diagnosability property in IPN with both presented approaches: characterization based on its language and characterization based on its structure.*

3.1 Diagnosability

The following definitions extend the concept of IPN previously presented 24 and these are applied to the IPN obtained by the modelling methodology presented in 2.2 in order to perform the analysis of the diagnosability property.

Definition 44. *Let* (Q, M_0) *be an* IPN *and* $\mathbf{R}(Q, M_0)$ *be its reachability graph.* $\mathfrak{F}=\{M| M(p_k) > 0, M \in \mathbf{R}(Q, M_0)$ *and* $p_k \in P^F_{sv^i_j}$ *for every state variable* $sv^i_j\}$ *is named the* ***faulty marking set*** *and* $\mathfrak{N}=\mathbf{R}(Q, M_0) - \mathfrak{F}$ *is named the* ***normal marking set.***

Definition 45. *Let* (Q, M_0) *be an* IPN, $M_f \in \mathfrak{F}$ *be a faulty marking and* $\pounds_{in}(Q, M_f)$ *be the input language of* (Q, M_0) *from* M_f. *A word* $\omega = \lambda(t_a)\lambda(t_b)\cdots\lambda(t_x) \in \pounds_{in}(Q, M_f)$ *is* ***controllable*** *if* $\forall t_s \in T$ *such that* $\lambda(t_s) = \varepsilon$ *it holds that* $\overline{\omega}\lambda(t_s) \cap \pounds_{in}(Q, M_f) \subseteq \pounds_{in}(Q, M_f)$, *where* $\overline{\omega}$ *is the prefix of* ω.

3.1.1 Diagnosable IPN

The following definition partitions the reachability set into faulty and normal markings and uses the faulty states to define the diagnosability property for IPN.

Definition 46. *An* IPN *given by* (Q, M_0) *is said to be* ***weak input-output diagnosable*** *at* k *steps if* $\forall\omega \in \pounds_{in}(Q, M_f)$ $\exists z$, *such that* $\omega z \in \pounds_{in}(Q, M_f)$, $|z| < k < \infty$ *and the information provided by* ωz, *the output word generated by* ωz *and the structure of the system* (Q, M_0) *are enough to distinguish any marking* $M_f \in \mathfrak{F}$ *from any other state.*

Definition 47. *An* IPN *given by* (Q, M_0) *is said to be* ***input-output diagnosable*** *in a finite* $k < \infty$ *steps if any marking* $M_f \in \mathfrak{F}$ *is distinguishable from any other state using a controllable input word* $\zeta \in \pounds_{in}(Q, M_f)$ *such that* $|\zeta| \leq k$, *and the generated output word* $\zeta_\omega \in \pounds_{out}(Q, M_f)$.

3.1.2 Characterization of the IPN based on its language

Now, the IPN exhibiting the input-output diagnosability property are characterized from a language point of view.

Definition 48. *Two different places* p_i, p_j *are* ***input-output related,*** *denoted as* $(p_i, p_j) \in IOR$ *if the following conditions are held:*

1. *Reachability:* $M_0 \xrightarrow{\sigma^a_{in}} M_i, M_0 \xrightarrow{\sigma^b_{in}} M_j$ *where* $M_i(p_i) \geq 1, M_j(p_j) \geq 1$, *i.e. there exist two sequences of transitions that leads to two different markings that mark places* p_i *and* p_j.

2. *Prefix equivalence:* $\sigma^a_{in} \neq \sigma^b_{in}$ *and* $\sigma^a_{in}, \sigma^b_{in} \in \Omega(\omega_{in})$, *for an* $\omega \in \Lambda(Q, M_0) \cup \Lambda_B(Q, M_0)$, *the two different sequences of transitions generate the same input-output word.*

3. *Suffix equivalence:* $\forall k > 0$ *there exist* $\sigma^a_{out} \neq \sigma^b_{out}$ *such that* $\sigma^a_{in}\sigma^a_{out}, \sigma^b_{in}\sigma^b_{out} \in \Omega(\omega_T)$, *for an* $\omega_T \in \Lambda^k(Q, M_0) \cup \Lambda_B(Q, M_0)$, *i.e. there exist one input-output word generated from two different sequences of transitions that concatenated with the sequences described in step 2 generate an input-output word part of the* IPN.

Proposition 49. *An IPN model (Q, M_0) is* ***weak input-output diagnosable*** *iff for* $\forall p_i \in P^F$ $\nexists p_j \in (P^N \cup P^F)$ *such that* $(p_i, p_j) \in IOR$.

Proof. (Sufficiency). Assume that $(p_i, p_j) \in IOR$ and $p_i \in P^F$ then there exist sequences $\sigma^a_{in}\sigma^a_{out}, \sigma^b_{in}\sigma^b_{out} \in \Omega(\omega_T)$, for $\omega_T \in \Lambda^k(Q, M_0) \cup \Lambda_B(Q, M_0)$ such that $M_0 \xrightarrow{\sigma^a_{in}} M_i \xrightarrow{\sigma^a_{out}}$, $M_0 \xrightarrow{\sigma^b_{in}} M_j \xrightarrow{\sigma^b_{out}}$ where $M_i(p_i) \geq 1, M_j(p_j) \geq 1$. Then $M_i \in \mathfrak{F}$ and cannot be distinguished from M_j when $\sigma^a_{in}\sigma^a_{out}, \sigma^b_{in}\sigma^b_{out}$ are fired. Thus (Q, M_0) is not input-output diagnosable.

(Necessity). For any $p_i \in P^F$ there exists no p_j such that $(p_i, p_j) \in IOR$. Then for no marking M_i, such that $M_i(p_i) \geq 1$ (i.e. $M_i \in \mathfrak{F}$), there exists another marking M_j such that both markings M_i, M_j are reached with the same input-output word and enable firing transition sequences with the same input-output words. Thus M_i is distinguishable from any other marking and (Q, M_0) is input-output diagnosable. □

Testing Diagnosability

In order to show how to test the diagnosability of a system off-line based on its language, consider the case study 1 described in section 2.2.1, which IPN is shown in figure 2.12. It is also presented a second case, which is a variant of case study 1.

Case Study 1 In order to test the input-output diagnosability property on the IPN shown in figure 2.12 from a language point of view, the proposition 49 will be used.

Let define $P^N = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{20}, p_{21}, p_{22}, p_{23}\}$ and $P^F = \{p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}\}$. For all $p_i \in P^F$ there exists no p_j such that $(p_i, p_j) \in IOR$. Then, the three conditions defined in 48 must be analyzed:

For p_{13} : if $M(p_{13}) = 1$ and $\omega_{in}=$ $(\varepsilon, P1)$ $(Init, IPaP1)$ $(Load, Op1P1)$ $(\varepsilon, Op1P1)=$ $(\varepsilon, P1)$ $(Init, IPaP1)$ $(Load, Op1P1)$ then there exists a sequence $\sigma^a_{in}= t_1x_2t_{16}$, $\sigma^a_{in} \in \Omega(\omega_{in})$. Marking $M(p_4) = 1$ can be reached with $\omega_{in}=$ $(\varepsilon, P1)$ $(Init, IPaP1)$ $(Load, Op1P1)$, then there exists a sequence $\sigma^b_{in}= t_1x_2$, $\sigma^b_{in} \in \Omega(\omega_{in})$. Therefore, the pair (p_{13}, p_4) fulfils condition 1 and condition 2, but it does not fulfil condition 3. It is because there not exist two sequences $\sigma^a_{out}, \sigma^b_{out}$, such that $\sigma^a_{in}\sigma^a_{out}, \sigma^b_{in}\sigma^b_{out}$ generate the same word. In fact, the next input symbol is *Start*, but the output signals are different for these two sequences.

For p_{17} : if $M(p_{17}) = 1$ and $\omega_{in}=$ $(\varepsilon, P1)$ $(Init, IPaP1)$ $(Load, Op1P1)$ $(Start, ELFacP1)$ $(\varepsilon, ELFacP1)=$ $(\varepsilon, P1)$ $(Init, IPaP1)$ $(Load, Op1P1)$ $(Start, ELFacP1)$ then there exists a sequence $\sigma^a_{in}= t_1x_2x_3t_{20}$, $\sigma^a_{in} \in \Omega(\omega_{in})$. Marking $M(p_9) = 1$ can be reached with $\omega_{in}=$ $(\varepsilon, P1)$ $(Init, IPaP1)$ $(Load, Op1P1)$ $(Start, ELFacP1)$, then there exists a sequence $\sigma^b_{in}= t_1x_2x_3$, $\sigma^b_{in} \in \Omega(\omega_{in})$. Therefore, the pair (p_{17}, p_9) fulfils condition 1 and condition 2, but it does not fulfil condition 3. It is because there not exist two sequences $\sigma^a_{out}, \sigma^b_{out}$, such that $\sigma^a_{in}\sigma^a_{out}, \sigma^b_{in}\sigma^b_{out}$ generate the same word. The next input symbol is *FacTur*, but the output signals are different for these two sequences.

For p_{18} : if $M(p_{18}) = 1$ and $\omega_{in}=$ $(\varepsilon, P1)$ $(Init, IPaP1)$ $(Load, Op1P1)$ $(Start, ELFacP1)$ $(FacTur, ELTurP1)$ $(\varepsilon, ELTurP1)=$ $(\varepsilon, P1)$ $(Init, IPaP1)$ $(Load, Op1P1)$ $(Start, ELFac$-$P1)$ $(FacTur, ELTurP1)$ then there exists a sequence $\sigma^a_{in}= t_1x_2x_3t_{11}t_{21}$, $\sigma^a_{in} \in \Omega(\omega_{in})$. Marking $M(p_{10}) = 1$ can be reached with $\omega_{in}=$ $(\varepsilon, P1)$ $(Init, IPaP1)$ $(Load, Op1P1)$ $(Start, ELFacP1)$ $(FacTur, ELTurP1)$, then there exists a sequence $\sigma^b_{in}= t_1x_2x_3t_{11}$, $\sigma^b_{in} \in \Omega(\omega_{in})$.

Therefore, the pair (p_{18}, p_{10}) fulfils condition 1 and condition 2, but it does not fulfil condition 3. It is because there not exist two sequences σ^a_{out}, σ^b_{out}, such that $\sigma^a_{in}\sigma^a_{out}$, $\sigma^b_{in}\sigma^b_{out}$ generate the same word. The next input symbol is *Stop*, but the output signals are different for these two sequences.

For p_{14} : if $M(p_{14}) = 1$ and $\omega_{in}= (\varepsilon, P1)\ (Init, IPaP1)\ (Load, Op1P1)\ (Start, ELFacP1)\ (FacTur, ELTurP1)\ (Stop, Op1P1)\ (\varepsilon, Op1P1)= (\varepsilon, P1)\ (Init, IPaP1)\ (Load, Op1P1)\ (Start, ELFacP1)\ (FacTur, ELTurP1)\ (Stop, Op1P1)$ then there exists a sequence $\sigma^a_{in}= t_1x_2x_3t_{11}x_4t_{17}$, $\sigma^a_{in} \in \Omega(\omega_{in})$. Marking $M(p_5) = 1$ can be reached with $\omega_{in}= (\varepsilon, P1)\ (Init, IPaP1)\ (Load, Op1P1)\ (Start, ELFacP1)\ (FacTur, ELTurP1)\ (Stop, Op1P1)$, then there exists a sequence $\sigma^b_{in}= t_1x_2x_3t_{11}x_4$, $\sigma^b_{in} \in \Omega(\omega_{in})$. Therefore, the pair (p_{14}, p_5) fulfils condition 1 and condition 2, but it does not fulfil condition 3. It is because there not exist two sequences σ^a_{out}, σ^b_{out}, such that $\sigma^a_{in}\sigma^a_{out}$, $\sigma^b_{in}\sigma^b_{out}$ generate the same word. The next input symbol is *Unload*, but the output signals are different for these two sequences.

For p_{15} : if $M(p_{15}) = 1$ and $\omega_{in}= (\varepsilon, -)\ (Init, IPa)\ (Load, Op2)\ (\varepsilon, Op2)= (\varepsilon, -)\ (Init, IPa)\ (Load, Op2)$ then there exists a sequence $\sigma^a_{in}= t_1x_6t_{18}$, $\sigma^a_{in} \in \Omega(\omega_{in})$. Marking $M(p_6) = 1$ can be reached with $\omega_{in}= (\varepsilon, -)\ (Init, IPa)\ (Load, Op2)$, then there exists a sequence $\sigma^b_{in}= t_1x_6$, $\sigma^b_{in} \in \Omega(\omega_{in})$. Therefore, the pair (p_{15}, p_6) fulfils condition 1 and condition 2, but it does not fulfil condition 3. It is because there not exist two sequences σ^a_{out}, σ^b_{out}, such that $\sigma^a_{in}\sigma^a_{out}$, $\sigma^b_{in}\sigma^b_{out}$ generate the same word. The next input symbol is *Start*, but the output signals are different for these two sequences.

For p_{19} : if $M(p_{19}) = 1$ and $\omega_{in}= (\varepsilon, -)\ (Init, IPa)\ (Load, Op2)\ (Start, MMop)\ (\varepsilon, MMop)= (\varepsilon, -)\ (Init, IPa)\ (Load, Op2)\ (Start, MMop)$ then there exists a sequence $\sigma^a_{in}= t_1x_6x_7t_{22}$, $\sigma^a_{in} \in \Omega(\omega_{in})$. Marking $M(p_{12}) = 1$ can be reached with $\omega_{in}= (\varepsilon, -)\ (Init, IPa)\ (Load, Op2)\ (Start, MMop)$, then there exists a sequence $\sigma^b_{in}= t_1x_6x_7$, $\sigma^b_{in} \in \Omega(\omega_{in})$. Therefore, the pair (p_{19}, p_{12}) fulfils condition 1 and condition 2, but it does not fulfil condition 3. It is because there not exist two sequences σ^a_{out}, σ^b_{out}, such that $\sigma^a_{in}\sigma^a_{out}$, $\sigma^b_{in}\sigma^b_{out}$ generate the same word. The next input symbol is *Stop*, but the output signals are different for these two sequences.

For p_{16} : if $M(p_{16}) = 1$ and $\omega_{in}= (\varepsilon, -)\ (Init, IPa)\ (Load, Op2)\ (Start, MMop)\ (Stop, Op2)\ (\varepsilon, Op2)= (\varepsilon, -)\ (Init, IPa)\ (Load, Op2)\ (Start, MMop)\ (Stop, Op2)$ then there exists a sequence $\sigma^a_{in}= t_1x_6x_7x_8t_{19}$, $\sigma^a_{in} \in \Omega(\omega_{in})$. Marking $M(p_7) = 1$ can be reached with $\omega_{in}= (\varepsilon, -)\ (Init, IPa)\ (Load, Op2)\ (Start, MMop)\ (Stop, Op2)$, then there exists a sequence $\sigma^b_{in}= t_1x_6x_7x_8$, $\sigma^b_{in} \in \Omega(\omega_{in})$. Therefore, the pair (p_{16}, p_7) fulfils condition 1 and condition 2, but it does not fulfil condition 3. It is because there not exist two sequences σ^a_{out}, σ^b_{out}, such that $\sigma^a_{in}\sigma^a_{out}$, $\sigma^b_{in}\sigma^b_{out}$ generate the same word. The next input symbol is *Unload*, but the output signals are different for these two sequences.

Since there not exists any pair (p_i, p_j), $p_j \in P^F$, input-output related $IOR = \oslash$, and for every pair (p_i, p_j) considered takes just one more fire of a transition to notice that an error has occurred in the system, then it can be concluded that the system is weak input-output diagnosable in k=1.

Case Study 2 Consider the system described in case study 1, in section 2.2.1, and the obtained IPN from the modeling methodology shown in figure 2.12, but with a change in the output function. Instead of $ELFac$ be the output symbol of places p_9, p_{17} and $ELTur$ be the output symbol of places p_{10}, p_{18}, now, the four places have the same output symbol

ELop.

The sets P^N and P^F are the same as the sets in case 1.

The analysis for all $p_i \in P^F$ in order to verify that there not exists p_j such that $(p_i, p_j) \in IOR$ is the same for every place except for place p_{17}. Then, the three conditions defined in 48 must be analyzed for place p_{17}:

For p_{17} : if $M(p_{17}) = 1$ and $\omega_{in}= (\varepsilon, P1)\ (Init, IPaP1)\ (Load, Op1P1)\ (Start, ELFacP1)\ (\varepsilon, ELFacP1)= (\varepsilon, P1)\ (Init, IPaP1)\ (Load, Op1P1)\ (Start, ELFacP1)$ then there exists a sequence $\sigma^a_{in}= t_1x_2x_3t_{20}$, $\sigma^a_{in} \in \Omega(\omega_{in})$. Marking $M(p_9) = 1$ can be reached with $\omega_{in}= (\varepsilon, P1)\ (Init, IPaP1)\ (Load, Op1P1)\ (Start, ELFacP1)$, then there exists a sequence $\sigma^b_{in}= t_1x_2x_3$, $\sigma^b_{in} \in \Omega(\omega_{in})$. Therefore, the pair (p_{17}, p_9) fulfils condition 1 and condition 2, and it does fulfil condition 3 when the next input symbol *FacTur* is present in the system due the output signal is the same signal *ELop* for both sequences. But, when it comes the next input symbol *Stop*, the pair (p_{17}, p_9) does not accomplish with condition 3. It is because there not exist two sequences σ^a_{out}, σ^b_{out}, such that $\sigma^a_{in}\sigma^a_{out}$, $\sigma^b_{in}\sigma^b_{out}$ generate the same output word.

Since there not exists any pair (p_i, p_j), $p_j \in P^F$, input-output related $IOR = \oslash$, and since there exist one pair (p_{17}, p_9) that takes two more fires of a transition to notice that an error has occurred in the system, then it can be concluded that the system is weak input-output diagnosable in k=2.

3.1.3 Structural Characterization of the IPN

Proposition 49 uses the reachability set and language information to characterize the IPN exhibiting the input-output diagnosability property. This fact leads to NP complete algorithms. In the case when the non faulty part of the IPN exhibits the event detectability property [1] [2] [38] [36], the characterization can be determined from the structure of the IPN. This is proved in the following lines.

Remark. $\cdot P^F \ = \cdot P^F \cup P^F$

Definition 50. *Let (Q, M_0) be an IPN. The **underlying normal behaviour** (Q^N, M_0^N) of (Q, M_0) is the IPN derived from (Q, M_0) when P^F and $\cdot P^F$ are removed from the set of places and transitions respectively and the marking is restricted to the new set of places.*

Proposition 51. *Let (Q, M_0) be an IPN and (Q^N, M_0^N) be its underlying normal behaviour IPN If (Q^N, M_0^N) is event-detectable and for every $p_i \notin P^F$ such that $p_i \subseteq \cdot\cdot P^F$ there is a transition $t_i \in (p_i)\cdot$ with $\lambda(t_i) \neq \varepsilon$, then (Q, M_0) is weak input-output diagnosable.*

Proof. Since (Q^N, M_0^N) is event detectable there are not two different transition sequences in (Q^N, M_0^N) generating the same input-output word ω. Therefore, for every input-output word ω the set $\Omega(\omega)$ is a singleton.

Assume without loss of generality that $M_0 \xrightarrow{\sigma} M_i \xrightarrow{t_e} M_j$ and $M_i \notin \mathfrak{F}$, $M_j \in \mathfrak{F}$. Since $t_e \in {}^\bullet P^F$ then $\lambda(t_e) = \varepsilon$ and $\varphi C(\bullet, t_e) = 0$, hence $\sigma, \sigma t_e \in \Omega(\omega)$ for a given $\omega \in \Lambda(Q, M_0) \cup \Lambda_B(Q, M_0)$. Thus there exist places $p_k \notin P^F$, and $p_e \in P^F$ such that $M_i(p_k) \geq 1$ and $M_j(p_e) \geq 1$ fulfilling conditions "Reachability" and "Prefix equivalence" of definition 48.

However condition "Suffix equivalence" will never be satisfied. Indeed since M_i is a live marking in (Q^N, M_0^N) then there exists an enabled transition $t_m \neq t_e$ enabled such that $\varphi C(\bullet, t_m) \neq 0$ and $\lambda(t_m) \neq \varepsilon$. By the modelling methodology, there exists t'_m enabled at M_j such that $\lambda(t_m) = \lambda(t'_m)$ and $\varphi C(\bullet, t'_m) = 0$. Then the output words are different $\varphi(M_0...M_iM_m) \neq \varphi(M_0...M_iM_jM_j)$ and σt_m and $\sigma t_e t'_m$ cannot belong to the same set $\Omega(\omega)$. Thus $p_e \in P^F$ cannot be related with any other place in IOR and by previous proposition (Q, M_0) is input-output diagnosable. □

Theorem 52. *Let (Q, M_0) be an IPN. If (Q, M_0) is observable, then (Q, M_0) is input-output diagnosable.*

Proof. Since (Q^N, M_0^N) is observable then there exists an integer $k < \infty$ such that $\forall \omega \in \Lambda^k(Q, M_0)$ it holds that the information provided by ω and (Q, M_0) suffices to uniquely determine the marking M_i reached by the firing of the underlaying transition firing sequence σ_ω (the initial marking is known). Suppose that M_i is a faulty marking, then it can be distinguished from any other using the input-output word ω. Therefore, (Q, M_0) is input-output diagnosable. □

Testing diagnosability

In this approach, the diagnosability property can be determined from its structure, i.e. when the underlying normal behaviour of the IPN shown in figure 2.12 exhibits the event detectability property, as it will be seen in the following.

Case Study 1 Based on proposition 51, there are two general conditions to be analyzed in order to define the diagnosability of the IPN :

1. *(Q^N, M_0^N) is event detectable.* In order for an IPN, in this case for the marking evaluator (Q^N, M_0^N) of (Q, M_0) shown in figure 3.1, to be event-detectable based on lemma 37, all columns of $\varphi \bullet C$ must be not null and different from each other, but if two columns are the same, then they must have different input symbols. From the result of $\varphi \bullet C$ shown in figure 3.2, it can be concluded that (Q^N, M_0^N) is event-detectable, and it is not necessary to analyze the input symbols.

2. *For every $p_i \in P^F$ such that $p_i \subseteq \quad P^F$ there is a transition $t_i \in (p_i)\cdot$ with $\lambda(t_i) \neq \varepsilon$.* For p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19} there exist transitions x_3, x_5, x_7, x_9, t_{11}, x_4, x_8 respectively, fulfilling this condition.

Therefore, the system is weak input-output diagnosable.

Case Study 2 Based on proposition 51, there are two general conditions to be analyzed in order to define the diagnosability of the obtained IPN for the case 2.

1. *(Q^N, M_0^N) is event detectable.* In order for an IPN, in this case for the marking evaluator (Q^N, M_0^N) of (Q, M_0) shown in figure 3.3, to be event-detectable based on lemma 37, all columns of $\varphi \bullet C$ must be not null and different from each other, but if two columns are the same, then they must have different input symbols. From the

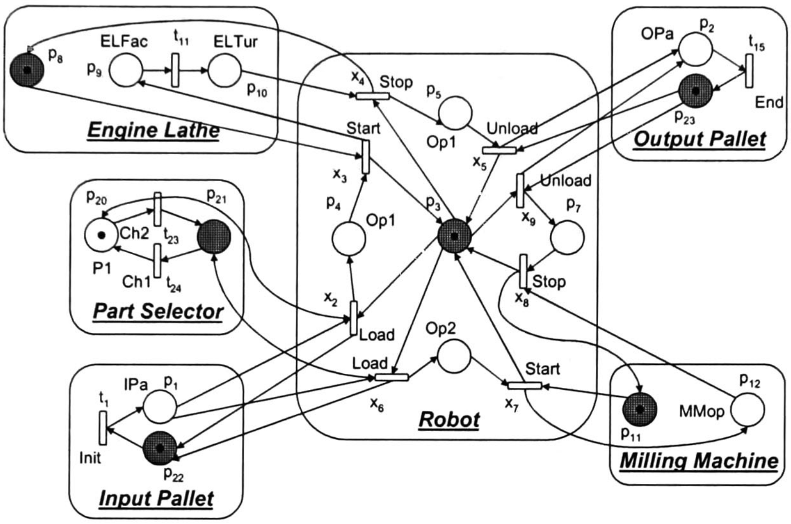


Figure 3.1: System1: The underlying normal behaviour (Q^N, M_0^N) of the IPN (Q, M_0)

$$\varphi \bullet C = \begin{pmatrix}
\text{[illegible]} & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & \text{[illegible]} & \text{[illegible]} & 0 & 0 \\
0 & 0 & 0 & 0 & \text{[illegible]} & 0 & 0 & \text{[illegible]} & \text{[illegible]} & 0 & -1 & \text{[illegible]} & 0 \\
0 & \text{[illegible]} & 1 & 1 & -1 & \text{[illegible]} & \text{[illegible]} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \text{[illegible]} & \text{[illegible]} & 0 & 1 & \text{[illegible]} & \text{[illegible]} & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & \text{[illegible]} & \text{[illegible]} & 0 & 1 & 0 & -1 & \text{[illegible]} & 0 & 0 \\
0 & \text{[illegible]} & 0 & -1 & 0 & 0 & \text{[illegible]} & \text{[illegible]} & 0 & 1 & 0 & \text{[illegible]} & \text{[illegible]} \\
0 & 0 & 0 & 0 & \text{[illegible]} & 0 & 1 & -1 & 0 & \text{[illegible]} & 0 & 0 & 0 \\
\text{[illegible]} & \text{[illegible]} & 0 & 0 & 0 & 0 & \text{[illegible]} & 0 & 0 & 0 & \text{[illegible]} & \text{[illegible]} & \text{[illegible]}
\end{pmatrix}$$

Figure 3.2: System1: $\varphi \cdot C$

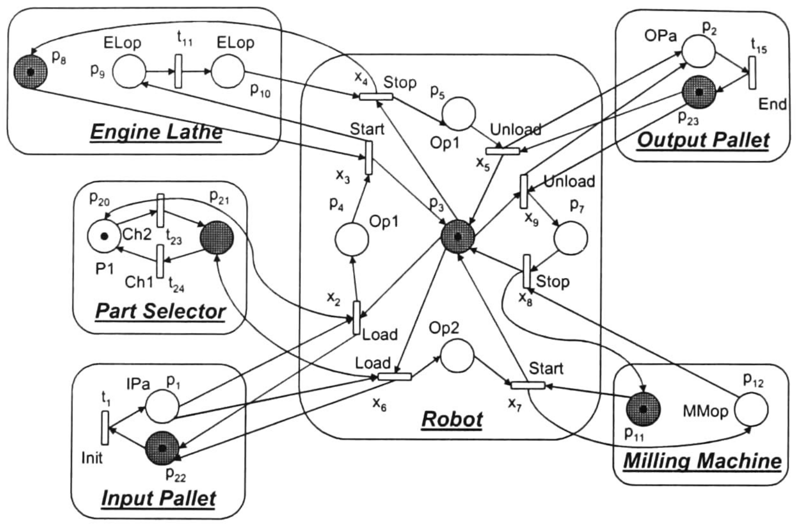


Figure 3.3: System2: The underlying normal behaviour (Q^N, M_0^N) of the IPN (Q, M_0)

$$\varphi \bullet C = \begin{pmatrix}
\text{[illegible]} & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & \text{[illegible]} & \text{[illegible]} & \text{[illegible]} \\
0 & 0 & 0 & 0 & \text{[illegible]} & \text{[illegible]} & 0 & 0 & \text{[illegible]} & 0 & \text{[illegible]} & 0 & 0 \\
0 & \text{[illegible]} & \text{[illegible]} & \text{[illegible]} & \text{[illegible]} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \text{[illegible]} & -1 & \text{[illegible]} & \text{[illegible]} & 0 & \text{[illegible]} & \text{[illegible]} & \text{[illegible]} \\
\text{[illegible]} & \text{[illegible]} & \text{[illegible]} & -1 & \text{[illegible]} & \text{[illegible]} & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & \text{[illegible]} & 0 & 0 \\
0 & 0 & 0 & 0 & \text{[illegible]} & 0 & 0 & \text{[illegible]} & \text{[illegible]} & \text{[illegible]} & 0 & \text{[illegible]} & 1
\end{pmatrix}$$

Figure 3.4: System2: $\varphi \cdot C$

result of $\varphi \bullet C$ shown in figure 3.4, it can be concluded that (Q^N, M_0^N) is not event-detectable, because there exists one column that is null corresponding to transition t_{11} that connects places p_9 and p_{10}. This is precisely because this two places have the same output signal, therefore, when the transition t_{11} is fired, the system can be either in state represented by p_{13} or in state represented by p_{14}.

2. *For every $p_i \in P^F$ such that $p_i \subseteq \cdot\cdot P^F$ there is a transition $t_i \in (p_i)\cdot$ with $\lambda(t_i) \neq \varepsilon$.* For p_{13}, p_{14}, p_{15}, [illegible]$_{16}$, p_{17}, p_{18}, p_{19} there exist transitions x_3, x_5, x_7, x_9, t_{11}, x_4, x_8 respectively, fulfill[illegible]ng this condition.

Therefore, there not [illegible]xist enough information to determine if the system is input-output diagnosable. This point [illegible] for future work.

3.2 Design of a Diagnoser Net

In this section, a device, called diagnoser, which allows determining system faults in an input-output diagnosable IPN model, is derived as an IPN. The diagnoser is a copy of the underlying normal behaviour IPN which is given the same inputs that are given to the system. By establishing that enabled transitions t_i in the diagnoser, such that $\lambda(t_i) = \alpha_i \neq \varepsilon$ fire whenever the input signal α_i is present, the differences in the output signals generated by the system and the diagnoser allows to determine the fault occurrences.

Definition 53. *Let (Q, M_0) be input-output diagnosable IPN in $k < \infty$ steps. The IPN (Q', M_0') is a* ***diagnoser net*** *for (Q, M_0) if for all transition sequence σ such that $M_0 \xrightarrow{\sigma} M_f$ where M_f is a fault marking, the same input word associated to σ, when given to (Q', M_0') generates a transition sequence σ' such that $M_0' \xrightarrow{\sigma'} M_f'$ and for all input word $\zeta \in \pounds_{in}(Q, M_f)$ such that $|\zeta| > k$, it holds that $\varphi(M_k) \neq \varphi(M_k')$ where M_k and M_k' are the markings reached when ζ is applied from M_f and M_f', respectively. i.e. When a faulty marking is reached then the output is different.*

Definition 54. *The underlying normal behaviour $N_B = (Q^N, M_0^N)$ IPN is called* ***marking evaluator*** *of the $N = (Q, M_0)$ if N_B fulfils the following two conditions:*

1. *The initial marking M_0^N of N_B is $M_0^N(p_i) = M_0(p_i), \forall p_i$.*

2. *If $\lambda(t_j) = \alpha$, t_j is enabled in N_B and $\alpha \in \Sigma$ is given, then t_j is fired.*

Theorem 55. *Let $N = (Q, M_0)$ be an IPN and $N_B = (Q^N, M_0^N)$ be a marking evaluator IPN of (Q, M_0). If (Q^N, M_0^N) is event-detectable and for every $p_i \notin P^F$ such that $(p_i)^{\cdot\cdot} \in P^F$ there is a transition $t_j \in (p_i)^{\cdot}$ with $\lambda(t_j) \neq \varepsilon$, then marking evaluator net is a diagnoser net for (Q, M_0).*

Proof. B[illegible]nition of N_B it holds that $M_0^N = M_0|_{P^N}$ and, as long as no faulty markings are reached, it holds that $M_k^N = M_k$. Let a fault transition $t_j \in (p_i)^\bullet$, fire in N reaching a new marking $M_{k+1} \in \mathfrak{F}$, since is a failure state it holds that $\varphi(M_k) = \varphi(M_{k+1})$. Moreover, since no input symbol $\alpha \in \Sigma$ was given in N, then N_B does not change its marking, thus there exists a difference between the marking of N and N_B. This error can be computed using the input word $\lambda(t_i) \neq \varepsilon$ because, since M_{k+1} is an error state, then the marking in N does not change. In N_B, however, t_i is fired and a new marking M_{k+1}^N is reached. Being N_B event-detectable, it holds that $\varphi(M_k^N) \neq \varphi(M_{k+1}^N)$ and therefore $\varphi(M_{k+1}) \neq \varphi(M_{k+1}^N)$. Thus, the error state can be detected. □

The global diagnoser scheme is shown in figure 3.5, where the Evaluator/Selector performs the difference between the outputs from the diagnoser net and from the system model. This difference is represented in a vector called error vector E with a length of the number of outputs in the IPN. The error vector determines the set of decision rules that are part of the Evaluator/Selector component. A decision rule is formed by two parts: the condition and the operations. The following algorithm shows how to construct the rules.

Algorithm 56. *Construction of rules.*

1. *For every $E_i[j] \neq 0$ do*

    (a) *condition: where $\varphi(j, \cdot) = E_i[j]$.*

    (b) *operation: where* FAULTTYPE *is in the component associated to the output $\varphi(j, \cdot)$.*

Case Study 1 Consider the case study described in section 2.2.1. The set of rules derived from the algorithm 56 to decide the type of failure are the following:

1. If $Op1 = 1$ and $ELFac = -1$ then $FaultType = Robot\_error\_Op1$.
2. If $Op2 = 1$ and $MMop = -1$ then $FaultType = Robot\_error\_Op2$.
3. If $ELFac = 1$ and $ELTur = -1$ then $FaultType = Engine\_lathe\_error\_Fac$.
4. If $ELTur = 1$ and $Op1 = -1$ then $FaultType = Engine\_lathe\_error\_Tur$.
5. If $MMop = 1$ and $Op2 = -1$ then $FaultType = Milling\_machine\_error$.
6. If $Op1 = 1$ and $Opa = -1$ then $FaultType = Robot\_error\_Op1\_2$.
7. If $Op2 = 1$ and $Opa = -1$ then $FaultType = Robot\_error\_Op2\_2$.

Case Study 2 For this case the set of if-then rules to decide the type of failure are the following:

1. If $Op1 = 1$ and $ELFac = -1$ then $FaultType = Robot\_error\_Op1$.
2. If $Op2 = 1$ and $MMop = -1$ then $FaultType = Robot\_error\_Op2$.
3. If $ELop = 1$ and $Op1 = -1$ then $FaultType = Engine\_lathe\_error$.
4. If $MMop = 1$ and $Op2 = -1$ then $FaultType = Milling\_machine\_error$.
5. If $Op1 = 1$ and $Opa = -1$ then $FaultType = Robot\_error\_Op1\_2$.
6. If $Op2 = 1$ and $Opa = -1$ then $FaultType = Robot\_error\_Op2\_2$.

In order to build the complete diagnoser scheme for an specific system consider the following algorithm.

Algorithm 57. *Building Diagnoser_Scheme*

1. *Make a copy of the underlying normal behaviour of the system, i.e. diagnoser net.*
2. *Connect the input commands to the diagnoser net and to the system.*
3. *Connect the outputs of the diagnoser net and of the system to the evaluator/selector.*
4. *Construct the rules.*

Case Study 1 The final diagnoser scheme derived from the algorithm 57 is shown in figure 3.6.

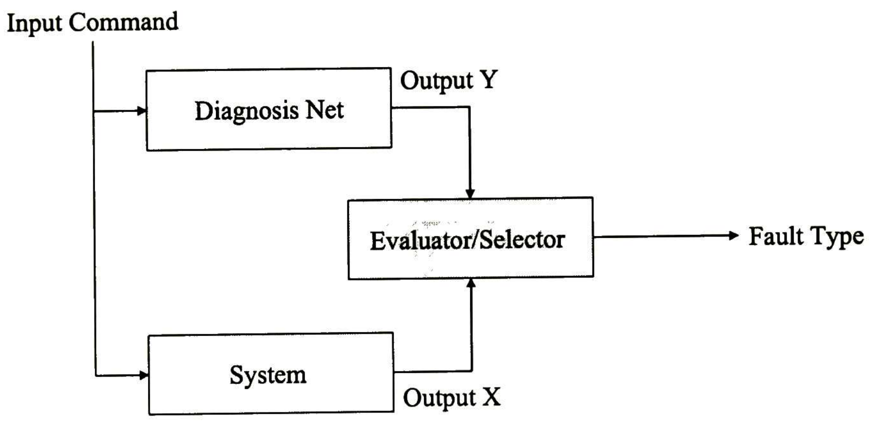
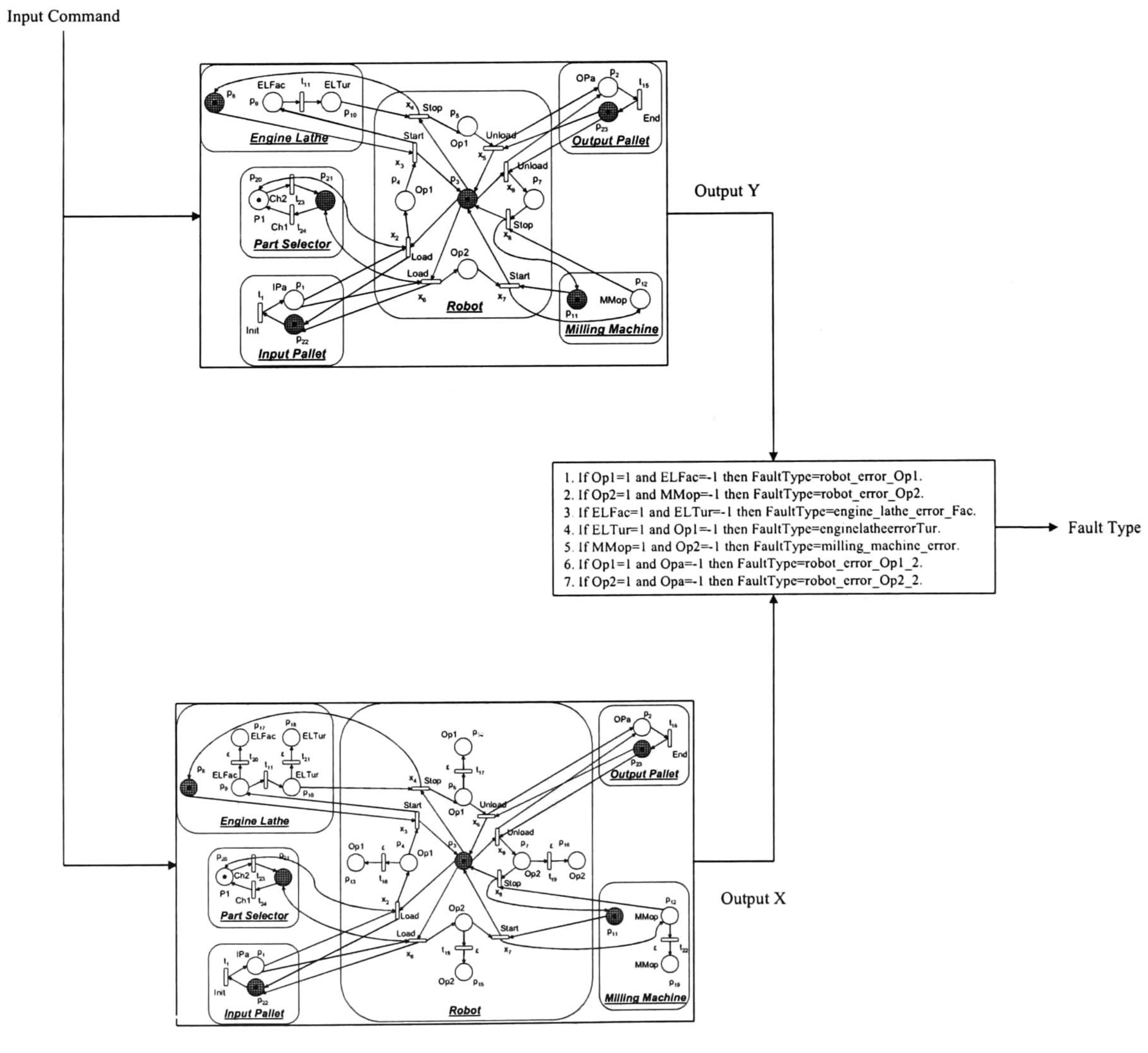


Figure 3.5: Global diagnoser scheme



1. If Op1=1 and ELFac=-1 then FaultType=robot_error_Op1.
2. If Op2=1 and MMop=-1 then FaultType=robot_error_Op2.
3. If ELFac=1 and ELTur=-1 then FaultType=engine_lathe_error_Fac.
4. If ELTur=1 and Op1=-1 then FaultType=enginelatheerrorTur.
5. If MMop=1 and Op2=-1 then FaultType=milling_machine_error.
6. If Op1=1 and Opa=-1 then FaultType=robot_error_Op1_2.
7. If Op2=1 and Opa=-1 then FaultType=robot_error_Op2_2.

Figure 3.6: Diagnoser Scheme of Case 1

Chapter 4

Fault Recovery in FMS

Summary. *This chapter presents the definition of Interpreted ReWriting Nets* ($IRWNets$) *and their dynamics. The* $IRWNets$ *formalism is an extension of the RWNets formalism that this chapter also presents. Finally, a study of the application of* $IRWNets$ *to task reconfiguration for addressing a fault recovery approach is presented.*

4.1 Error Recovery Review

There exist several techniques addressing different types of failures presented by diverse authors.

Fanni et al. in [14] use the algorithms presented in [18] to estimate the actual marking which is used to design a state feedback controller. The controlled system may result blocked, as a result of the estimate, hence the authors show with an example how to introduce time-out mechanisms to recovery from this blocking. This time-out mechanism consists in adding a place and a time-out transition. Thus, the timed transition will fire only if no other transition is enabled. Then, the observer, regardless of its previous marking, reconstructs its marking.

Heimerdinger et al. in [19] present the two common mechanisms for fault recovery which are acceptance test techniques and comparison techniques. Acceptance tests usually uses recovery blocks, which provide backward fault recovery by rolling back to the state before the fault occurred repairing the state and the result. Comparison-based technique uses multiple pairs of threads or processors performing the same task. When an error occurs, the recovery uses the values from the "good" pair usually by voting.

Khatab et al. in [22] uses an automaton augmented by data variables to control the system and by time information to model timed discrete event systems. They present two types of failure recovery procedures, one that takes the system back to its normal functioning mode, and one that takes the system back to a degraded functioning mode. The first procedure consists on driving the system from a failure state back to a previous normal state. The second procedure assumes that the system reaches a failure state and there is no way to return the system back to its normal functioning without loss of performance. Therefore, for every failure, a degraded behaviour is specified as a set of states to which the system is taken back from a fault state. The system evolves in this specified set until the return to the normal behaviour is possible that naturally occurs after the failure disappears.

Loborg in [28] presents a basic categorization for fault recovery. One is the backward error recovery that consists on find a previous error free state of the system and return there, by undoing what has been done since then. The other is the forward error recovery that consists on find an error free state that the system is supposed to eventually reach, and perform actions to reach that state, by predefined alternative actions or replanning of actions.

Zhou et al. in [44] propose the concept of a controller modelled by PN for automatic error recovery, and present four basic augmentation methods for different types of error recovery guaranteeing the properties of the controller when the PN augmentation methods are applied. The first construct is the Input Condition, which if this condition is satisfied will favour the selection of a specific transition in the case when several transitions leave a place, otherwise rejects that transition. The second construct is the Alternative Path, which uses the input condition to define an alternative path through the PN. The third construct is the Backward Error Recovery, which uses the input condition and the alternative path to trap an error and execute corrective actions returning to a previous state in the PN. Finally, the fourth construct is the Forward Error Recovery, which is analogous to backward error recovery but the corrective action will solve the problem and return the control to the same place as it was when the error was originated or detected.

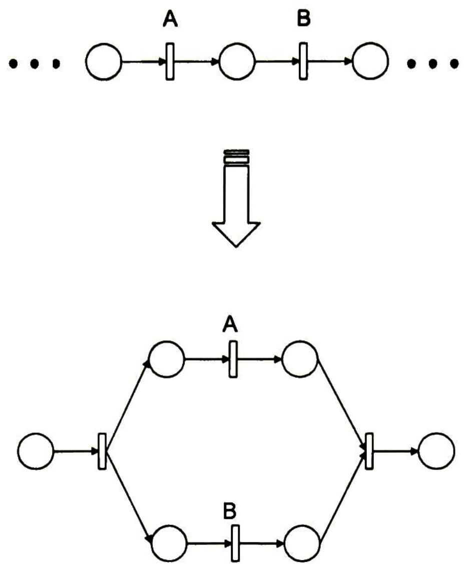


Figure 4.1: Production Reprogramming

4.2 An Approach for Fault Recovery

The fault recovery approach here proposed uses the Interpreted Re-Writing Nets ($IRWNets$) formalism which is an extension of the Re-Writing Nets ($RWNets$) formalism [10] [11]. The $RWNets$ formalism is an extension of Mobile nets [9], Dynamic nets [5], and Reconfigurable nets [6], which are classes of high level Petri nets. The $IRWNets$ formalism will be used for the reconfiguration of the IPN model by adding, deleting, and replacing subnets. This reconfiguration can be applied to changes imposed by modifications on the production requirements (production reprogramming) or changes inherent to recovery actions from failure situations (fault recovery).

Figure 4.1 shows the case in which a piece of a task execution model must be modified: the operations A and B that were sequentially performed must be now executed in parallel. Figure 4.2 shows the case in which a failure during the execution of the activity A leads to an unknown state from which a recovery operation R must be performed to bring the controller back to an operational state into the original workflow; then, the new path is added to the model. This process can be considered as an adaptation mechanism.

The $RWNets$ allow realizing dynamic changes within a PN structure [10], as shown in figure 4.3. The firing of transition r consumes the left side PN, which involves four activities, $Notify$, $Take\_Left$, $Take\_Right$ and Eat, within a causal relationship. The firing of transition r produces the right side PN, which establishes the change of the $Take\_Left$ and $Take\_Right$ sequential activities into two parallel ones.

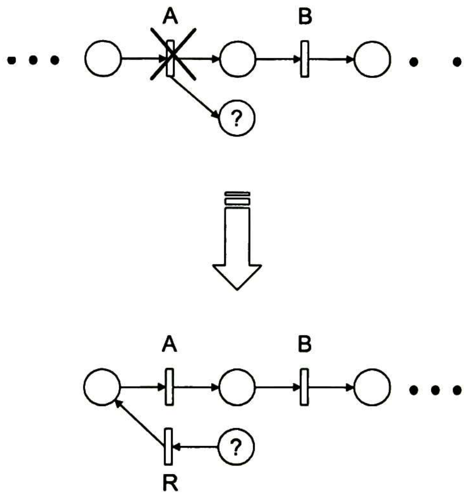


Figure 4.2: Failure Recovery

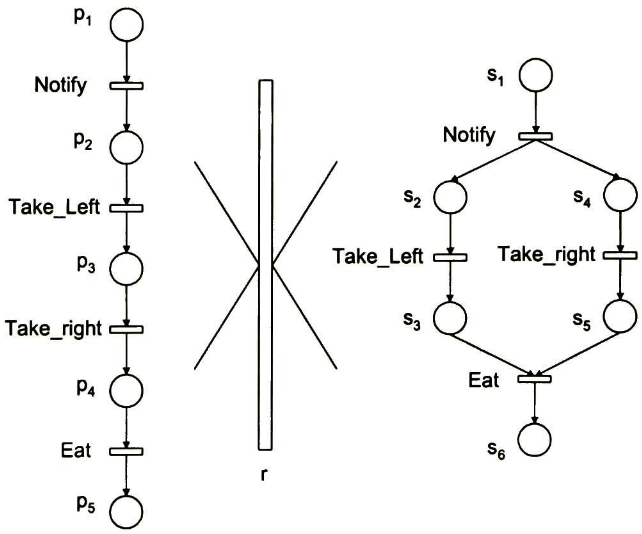


Figure 4.3: Dynamic Changes within a PN model

4.3 Interpreted RWNets

4.3.1 IRWNets Formal Definition

In this section, the *Interpreted Marked Re-Writing Nets* ($IRWNets$) are introduced providing similar modelling capabilities of IPN. The definition of a $IRWNet$ uses the definition of a $RWNet$, therefore it will be defined below.

Definition 58. *The set of* ***Re-Writing Nets*** *($RWNets$) is defined recursively as follows [10]:*

1. *A marked PN is a $RWNet$*
2. *The pair (N, M_0), where $N = (P, T, I, O)$ such that:*
    - *P is a finite set of places or $RWNets$,*
    - *T is a finite set of transitions,*
    - *$I, O : P \times T \to Z^+$ are the Pre and Post functions, respectively,*
    - *$M_0 : P \mid_{Places} \to Z^+$ is the initial marking, restricted to places.*

Now, the definition of a $IRWNet$ can be now introduced.

Definition 59. *The set of* ***Interpreted Re-Writing Nets*** *($IRWNets$) is defined as follows:*

1. *A marked IPN is a $IRWNet$*
2. *The pair (Q, M_0), Q is 5-tuple $Q = (N, \Sigma, \Phi, \lambda, \varphi)$ where*
    - *$N = (P, T, I, O)$ is a $RWNet$,*
    - *$\Sigma = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is the* ***input alphabet*** *of the net, where α_i is an* ***input symbol****,*
    - *$\Phi = \{\zeta_1, \zeta_2, ..., \zeta_n\}$ is the* ***output alphabet*** *associated to places, where ζ_i is an* ***output symbol****,*
    - *$\lambda : T \to \Sigma \cup \{\varepsilon^i\}$ is a* ***labelling function*** *of transitions with the following restriction:*
        - *$\forall t_j, t_k \in T$, $j \neq k$, if $p_i \in I(t_j)$ and $p_i \in I(t_k)$ and both $\lambda(t_j)$, $\lambda(t_k) \neq \varepsilon^i$, then $\lambda(t_j) \neq \lambda(t_k)$. In this case ε^i represents an internal system event,*
    - *$\varphi : R(Q, M_0) \to \{\Phi\}^q \mid_{Places}$ is an* ***output function*** *restricted to places, where $R(G, M_0)$ is the* ***reachability set*** *defined as in the PN and q is the number of available outputs associated to places.*
    - *$M_0 : P \mid_{Places} \to Z^+$ is the initial marking, restricted to places.*

4.3.2 Dynamics of IRWNets

Definition 60. *An IRWNet (Q^s, M_0^s) is a* ***subnet*** *of another IRWNet (Q, M_0), denoted $(Q^s, M_0^s) \subseteq (Q, M_0)$, iff:*

1. $P^s \subseteq P$,
2. $T^s \subseteq T$,
3. $I^s \subseteq I\,|_{T^s}$,
4. $O^s \subseteq O\,|_{T^s}$,
5. $\lambda(T^s) \subseteq \lambda(T)$,
6. $\varphi(P^s)\,|_{Places} \subseteq \varphi(P)\,|_{Places}$
7. *And, recursively,* $\forall t_j \in T^s$ $\{\cup p_i \mid (p_i, t_j) \in I^s\} \subseteq \{\cup p_k \mid (p_k, t_j) \in I\}$ *and* $\{\cup p_l \mid (p_l, t_j) \in O^s\} \subseteq \{\cup p_m \mid (p_m, t_j) \in O\}$.

Definition 61. *Two IRWNets, (Q, M_0) and (Q', M_0') are* ***isomorphic,*** *denoted $(Q, M_0) \approxeq (Q', M_0')$, iff there exists a pair of isomorphism $< f, g >$, where $f : T \to T'$, $g : P \to P$, such that $I'^\circ f = g^\circ I'$ $O'^\circ f = g^\circ O'$ and, recursively, $g(W) \approxeq W$, W and $I'^\circ f(t) \approxeq W \mid (W, t_j) \in I$, $\forall t_j \in T$.*

Definition 62. *Let $(Q, M_0) \in IRWNets$. A* ***binding*** *b for $W \in RWNets$ is a function $b : RWNets \to RWNets$, such that $W \approxeq b(W)$, $\lambda(t) \approxeq b(\lambda(t))$, and $\varphi(W) \approxeq b(\varphi(W))$, $\forall t \in T$, $\forall p \in P\,|_{Places}$*

Definition 63. ***Substitution.*** *Let b be a binding. Let $< f, g >$ be the pair isomorphism defined by b. For a $(Q, M_0) \in IRWNets$, $(Q, M_0)[b]$ denotes $(Q, M_0) < f, g >$, i.e., the application of the pair isomorphism $< f, g >$ to (Q, M_0).*

Definition 64. ***Transition enabling.*** *Let $(Q, M_0) \in IRWNets$. Let $t \in T$; t is enabled in a marking M_i under the binding b, denoted $(N, M_i)[t >_b$, iff there exists a binding b for the pre of t, namely W, such that $b(W) \subseteq N$ However, if t is an uncontrollable transition then t can be* ***fired;*** *otherwise, the input symbol $\lambda(t) = \alpha_i \neq \varepsilon$ must be present in order to* ***fire*** *t.*

Definition 65. ***Firing Rule.*** *Let $\_[>_b, \_ \in IRWNet \times IRWNet$, be the smallest substitutive relatior generated by:*

$(N, M_0)[t > b \Rightarrow (N', M_0')$, where $(N', M_0') \in \_[>_b, (W^1, t) \in I \mid W^1 = (Q^1, M_0^{w^1})$, and $(W^2, t) \in O \mid W^2 = (Q^2, M_0^{w^2})$,

Therefore,

$M_0' = M_0 \backslash (M_0^{w^1} \backslash M_0^{w^2}) \oplus (M_0^{w^2} \backslash M_0^{w^1})$ and $N' = N \backslash (W^1[b] \backslash W^2[b]) \oplus (W^2[b] \backslash W^1[b])$.

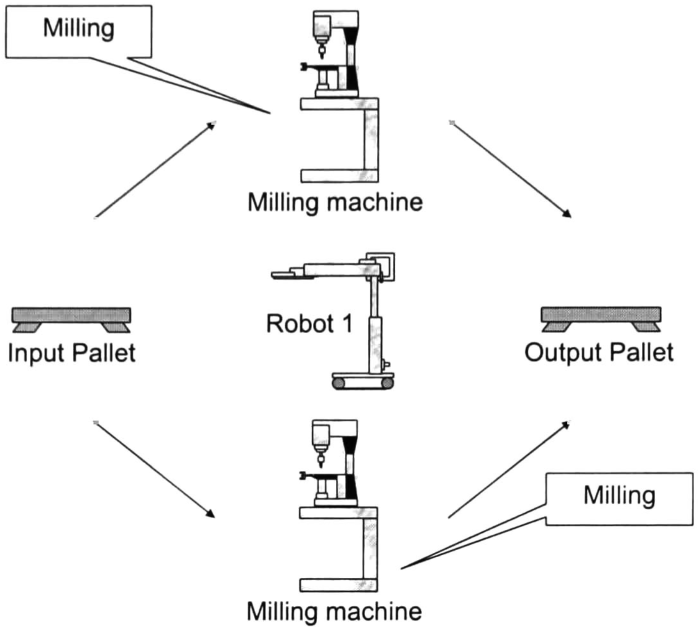


Figure 4.4: System3: Layout

4.4 Case Study 1: Fault Recovery in FMS

For the application of *IRWNets* to fault recovery, *IRWNets* can model the following and other mechanisms:

a) *IPN* rewriting system. In form of rules, pieces of *IPN* can be replaced by other *IPN*. The pre and the post conditions of a transition are both *IPN* structures.

b) *IPN* recovery system. The Pre is a reachable marking of an *IPN* describing a task; the Post is an *IPN* structure and/or a marking that must be added to the original net for recovery purposes.

System Description

The system consists of one robot, two milling machines, and two pallets, one input pallet and one output pallet. As shown in figure 4.4, the process plan for a part is the following:

1. The robot takes a raw part from the input pallet, and loads it at the milling machine.
2. A milling machine performs its tasks on the part. There are two milling machines, therefore there exist two ways to accomplish the process plan.
3. The robot unloads the part from the milling machine and moves it on to the output conveyor.

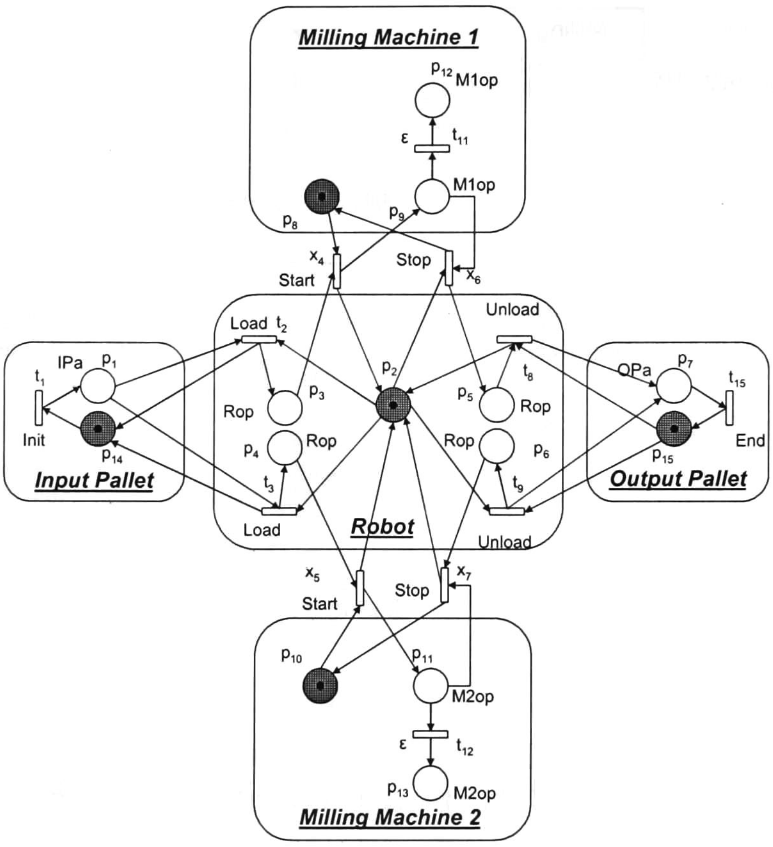


Figure 4.5: System3: $IRWNet$

System Model

The IPN shown in figure 4.5 is the system model corresponding to the system description presented before using the extended modelling methodology presented in this work. This IPN has been tested for diagnosability property and it is input-output diagnosable in k=1.

Remark. *Call that an IPN is an $IRWNet$. See definition 59.*

IRWNet and RW Rules

In the proposed system, several faulty states could exist. For example, in place p_9 where the milling machine $M1$ is supposed to perform certain task on the part, for any reason, the $M1$ task cannot be completed causing a fault on the system, represented by p_{12}. In order to implement error recovery in the system due to this case, a few changes are made to the system model for converting it into an $IRWNet$ named $W = (Q, M_0)$.

Definition of W The $IRWNet$ $W = (Q, M_0)$, where $Q = (N, \Sigma, \Phi, \lambda, \varphi)$, is defined as below:

- $N = (P, T, I, O)$, where:
  - $P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, W^1\}$.
    $T = \{t_1, t_2, t_3, x_4, x_5, x_6, x_7, t_8, t_9, t_{10}, t_{11}, t_{12}, r_1\}$.

$$I = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$O = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}$$

- $\Sigma = \{$*Init, Load, Start, Stop, Unload, End, Milling_machine_error*$\}$.

- $\Phi = \{IPa,\ Rop,\ M1op,\ M2op,\ Opa\}$.

- $\lambda(t_1) = Init$, $\lambda(t_2) = \lambda(t_3) = Load$, $\lambda(x_4) = \lambda(x_5) = Start$, $\lambda(x_6) = \lambda(x_7) = Stop$, $\lambda(t_8) = \lambda(t_9) = Unload$, $\lambda(t_{10}) = End$, $\lambda(t_{11}) = \lambda(t_{12}) = \varepsilon$, $\lambda(r_1) =$ *Milling_machine_error*

- $\varphi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$

- $M_0 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T$

Because the definition of an $IRWNet$ is recursive, see definition 59, it is necessary to define W^1.

Let $W^1 = (Q^1, M_0^1)$, where $Q^1 = (N^1, \Sigma^1. \Phi^1, \lambda^1, \varphi^1)$, is defined as below:

- $N^1 = (P^1, T^1, I^1, O^1)$, where:
  - $P^1 = \{p_1, p_3, p_5, p_8, p_9, p_{12}\}$.
    $T^1 = \{t_2, x_4, x_6, t_8, t_{11}\}$.
    $I^1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
    $O^1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

- $\Sigma = \{Load,\ Start,\ Stop,\ Unload\}$.

- $\Phi = \{IPa,\ Rop,\ Mop\}$.

- $\lambda(t_2) = Load,\ \lambda(x_4) = Start,\ \lambda(x_6) = Stop,\ \lambda(t_8) = Unload,\ \lambda(t_{11}) = \varepsilon$.

- $\varphi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$

- $M_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$

Transition r_1 (shown in figure 4.6) is enabled when a token is present in place p_{12} and when the input command *Milling_machine_error* is emitted from the diagnoser system as a fault type, then r_1 is fired and Q^1 is deleted from Q resulting in a new $IRWNet$, as shown in figure 4.7. This means that if the system is working properly, and suddenly a fault occurs in the system detected by the diagnoser, the evaluator/selector component issues a command to fire the corresponding transition in order to recover the system to a fault free state.

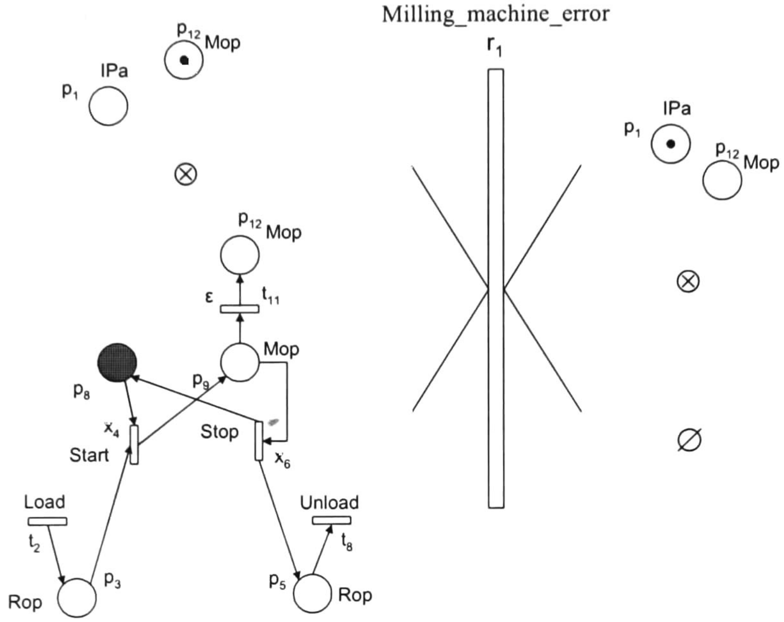


Figure 4.6: System3: Re-Writing Transition $r1$

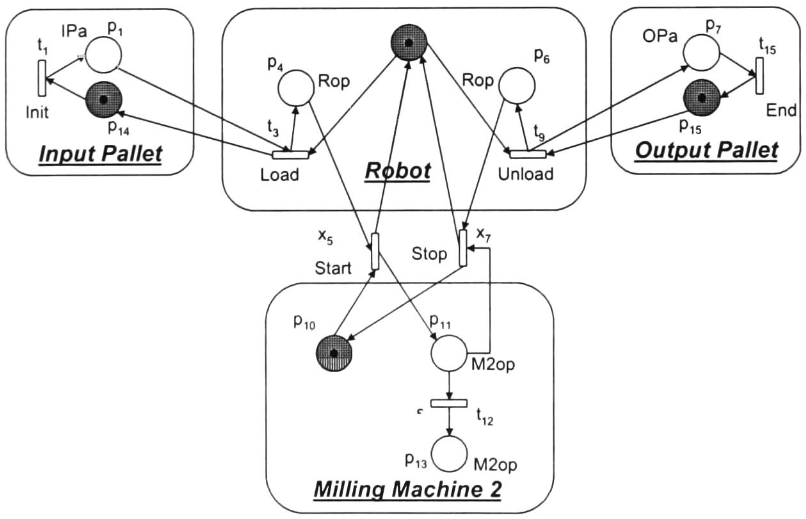


Figure 4.7: System3: Q after the firing of σ

4.5 Case Study 2: Production Reprogramming in FMS

The *IRWNets* formalism is also used to production reprogramming in *FMS*.

System Description

The process plan for a Bracket is the following:

1. Mill outside dimensions ensuring a 2.300 X 0.95 X 0.950 part.
2. Clamp on Two smooth sides.
3. Face top and one end.
4. Face opposite end and cut to length.

In order to compliance with this process plan, the following components are needed:

- A milling machine.
- An engine lathe for facing process.
- Three conveyors.
- Two robots.

The system layout is depicted in figure 4.8, and the process of a part is described as follows:

1. Robot 1 takes the part from the input pallet to the milling machine.
2. Milling machine performs the corresponding process of milling on the part.
3. Robot 1 takes the part from the milling machine to the buffer pallet.
4. Robot 2 takes the part from the buffer pallet to the engine lathe.
5. Engine lathe performs the corresponding processes of facing both sides on the part.
6. Robot 2 takes the part from the engine lathe to the output pallet.
7. The part is finished.

System Model

The *IPN* shown in figure 4.9 is the system model corresponding to the system description presented before. This *IPN* has been tested for diagnosability property and it is input-output diagnosable in k=1.

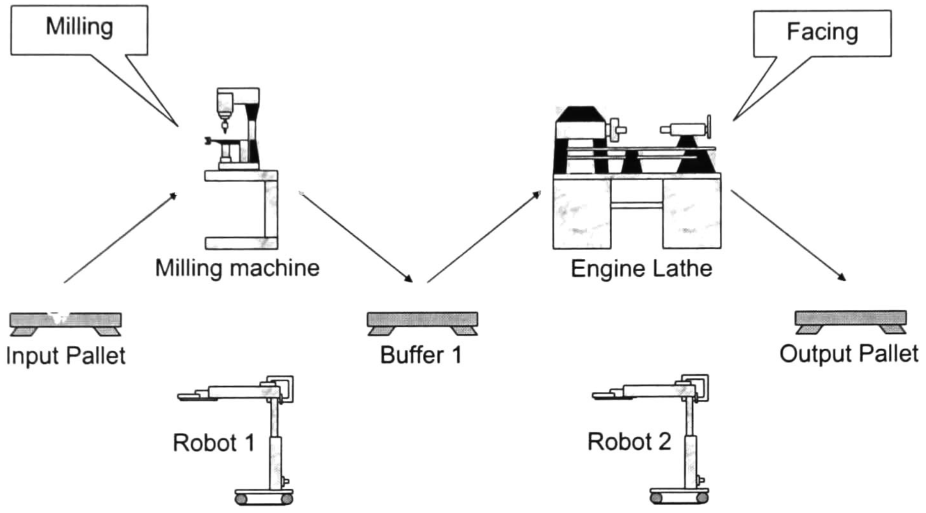


Figure 4.8: System4: Layout

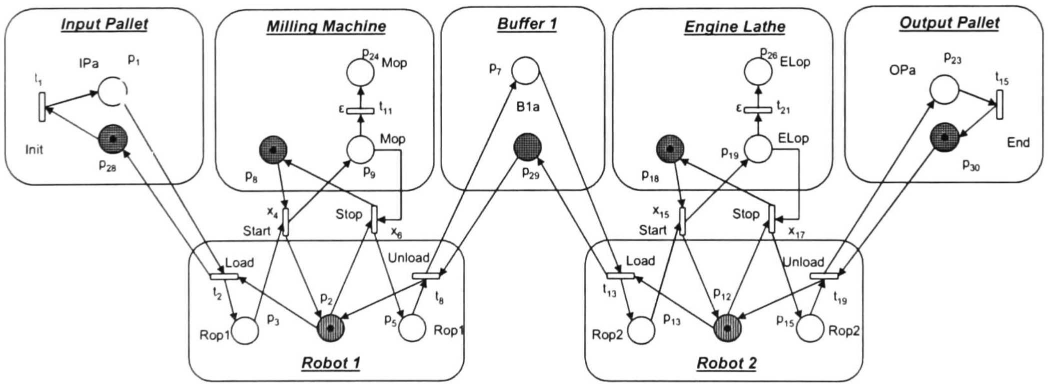


Figure 4.9: System4: *IRWNet*

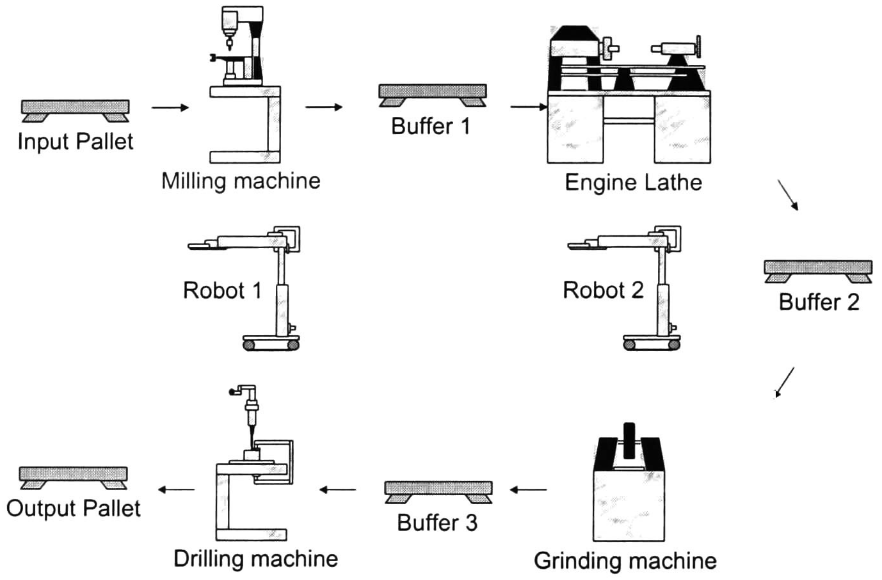


Figure 4.10: System4: Layout

IRWNet and RW Rules

Suppose the system requirements have changed. The actual system layout is shown in figure 4.10. Therefore, it is needed to reconfigure the system, and the *IRWNets* can support it.

Definition of W The *IRWNet* $W = (Q, M_0)$, where $Q = (N, \Sigma, \Phi, \lambda, \varphi)$, is defined as below:

- $N = (P, T, I, O)$, where:

  - $P = \{p_1, p_2, p_3, p_5, p_7, p_8, p_9, p_{12}, p_{13}, p_{15}, p_{18}, p_{19}, p_{23}, p_{24}, p_{26}, W^1, W^2\}$.

    $T = \{t_1, t_2, x_4, x_6, t_8, t_{10}, t_{11}, t_{13}, x_{15}, x_{17}, t_{19}, t_{21}, r_1\}$.

$$I = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$O = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

- $\Sigma = \{Init,\ Load,\ Start,\ Stop,\ Unload,\ End,\ Reprogramming\}$.

- $\Phi = \{IPa,\ Rop1,\ Rop2,\ Mop,\ ELop,\ B1a,\ Opa\}$.

- $\lambda(t_1) = Init$, $\lambda(t_2) = \lambda(t_{13}) = Load$, $\lambda(x_4) = \lambda(x_{15}) = Start$, $\lambda(x_6) = \lambda(x_{17}) = Stop$, $\lambda(t_8) = \lambda(t_{19}) = Unload$, $\lambda(t_{10}) = End$, $\lambda(t_{11}) = \lambda(t_{21}) = \varepsilon$, $\lambda(r_1) = Reprogramming$.

- $\varphi = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \end{bmatrix}$

- $M_0 = \begin{bmatrix} 0&1&0&0&0&1&0&1&0&0&1&0&0&0&0 \end{bmatrix}^T$

Because the definition of a *RWNet* is recursive, see definition 59, it is necessary to define W^1 and W^2.

The *IRWNet* $W^1 = (Q^1, M_0^1)$, where $Q^1 = (N^1, \Sigma^1, \Phi^1, \lambda^1, \varphi^1)$, is defined as below:

- $N^1 = (P^1, T^1, I^1, O^1)$, where:

  - $P^1 = \{p_2, p_3, p_5, p_{12}, p_{13}, p_{15}, p_{23}\}$.

    $T^1 = \{t_2, t_8, t_{10}, t_{13}, t_{19}\}$.

    $I^1 = \begin{bmatrix} 1&0&0&0&0 \\ 0&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&0 \\ 0&0&0&0&1 \\ 0&0&1&0&0 \end{bmatrix}$

    $O^1 = \begin{bmatrix} 0&1&0&0&0 \\ 1&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&1 \\ 0&0&0&1&0 \\ 0&0&0&0&0 \\ 0&0&0&0&1 \end{bmatrix}$

- $\Sigma^1 = \{Load,\ Unload,\ End\}$.

- $\Phi^1 = \{Rop1,\ Rop2,\ Opa\}$.

- $\lambda^1(t_2) = \lambda^1(t_{13}) = Load$, $\lambda^1(t_8) = \lambda^1(t_{19}) = Unload$, $\lambda^1(t_{10}) = End$.

- $\varphi^1 = \begin{bmatrix} 0&1&1&0&0&0&0 \\ 0&0&0&0&1&1&0 \\ 0&0&0&0&0&0&1 \end{bmatrix}$

- $M_0^1 = \begin{bmatrix} 1&0&0&1&0&0&0 \end{bmatrix}^T$

The *IRWNet* $W^2 = (Q^2, M_0^2)$, where $Q^2 = (N^2, \Sigma^2, \Phi^2, \lambda^2, \varphi^2)$, is defined as below:

- $N^2 = (P^2, T^2, I^2, O^2)$, where:
  - $P^2 = \{p_2, p_3, p_4, p_5, p_6, p_{10}, p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, p_{17}, p_{20}, p_{21}, p_{22}, p_{23}, p_{25}, p_{27}\}$.
    $T^2 = \{t_2, t_3, x_5, x_7, t_8, t_9, t_{10}, t_{12}, t_{13}, t_{14}, x_{16}, x_{18}, t_{19}, t_{20}, t_{22}\}$.

$$I^2 = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$O^2 = \begin{bmatrix}
0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

- $\Sigma^2 = \{Load, Start, Stop, Unload, End\}$.

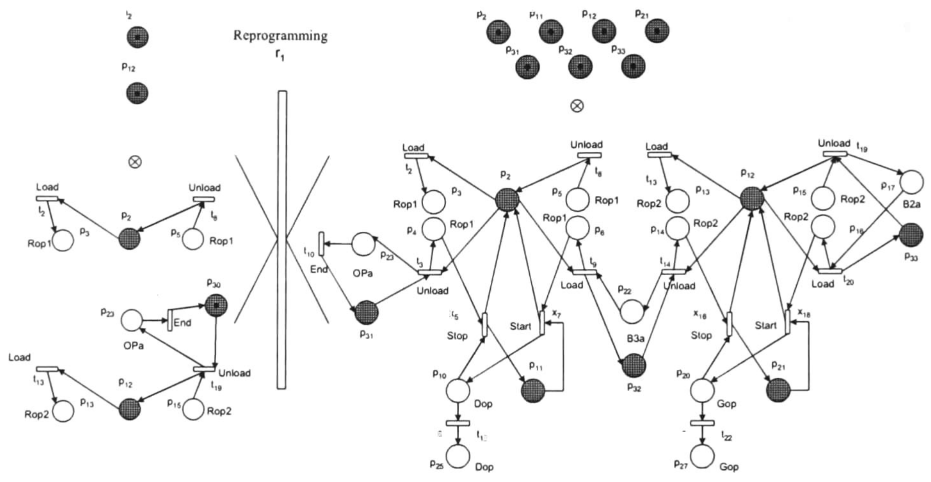


Figure 4.11: System4: Re-Writing Transition $r1$

- $\Phi^2 = \{Rop1, Rop2, Gop, Dop, B2a, B3a, Opa\}$.
- $\lambda^2(t_2) = \lambda^2(t_{13}) = Load$, $\lambda^2(t_8) = \lambda^2(t_{19}) = Unload$, $\lambda^2(t_{10}) = End$, $\lambda^2(t_9) = \lambda^2(t_{20}) = Load$, $\lambda^2(x_7) = \lambda^2(x_{18}) = Start$, $\lambda^2(x_5) = \lambda^2(x_{16}) = Stop$, $\lambda^2(t_3) = \lambda^2(t_{14}) = Unload$, $\lambda^2(t_{10}) = End$, $\lambda^2(t_{12}) = \lambda^2(t_{22}) = \varepsilon$.
- $\varphi^2 = \begin{bmatrix} 0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \end{bmatrix}$
- $M_0^2 = \begin{bmatrix} 1&0&0&0&0&0&1&1&0&0&0&0&0&0&1&0&0&0&0 \end{bmatrix}^T$

Transition r_1 (shown in figure 4.11) is enabled in the particular case there exists a reprogramming of the system. Hence, there is emitted an input command named *Reprogramming*, then r_1 is fired and Q^2 is added to Q through Q^1 resulting in a new *IRWNet*, as shown in figure 4.12.

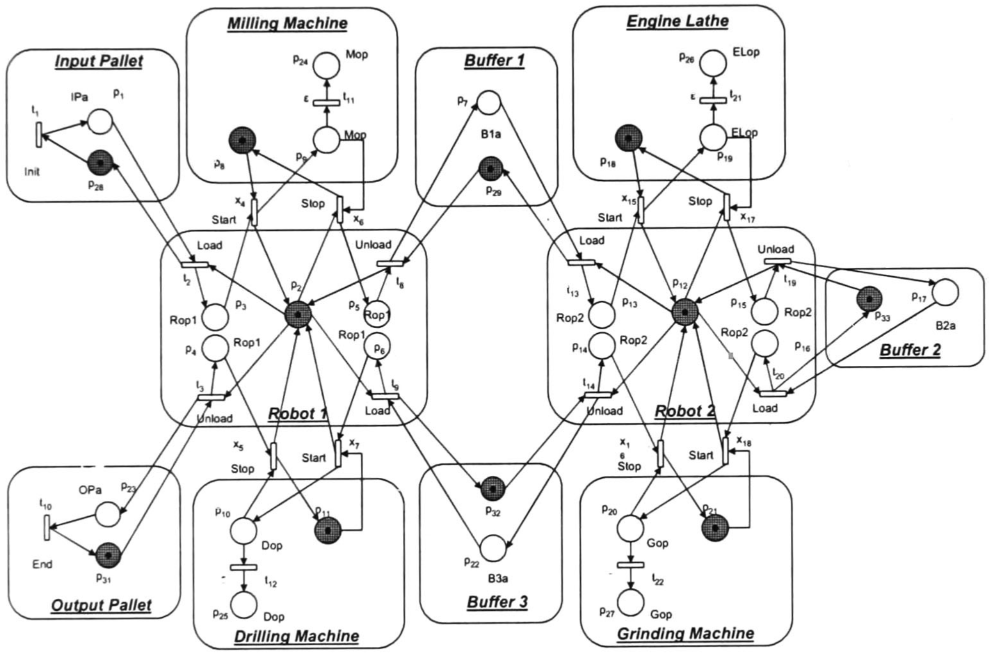


Figure 4.12: System4: Q after de firing of r_1

Conclusions

This thesis presented an extension of a modelling methodology for DES that obtains binary IPN models that includes the faulty markings of the system. The advantage of the use of this modelling methodology is to provide the necessary models to analyse the property of diagnosability, which this thesis also introduces. Specifically, the advantage of the use of IPN in the diagnosis problem is the structural manipulation that this formalism provides.

The concept of diagnosability of IPN models is a novel contribution in the context of PN since this concept has been applied only for AF in the DES. In order to test the diagnosability of the IPN two characterizations are introduced. The first characterization uses the reachability set and the language information to characterize the IPN exhibiting the input-output diagnosability property. This fact leads to NP complete algorithms. Therefore, a second characterization was introduced. In this characterization, in the case when the non-faulty part of the IPN exhibits the event detectability property, the diagnosability of the IPN can be determined from its structure leading to a P algorithm.

It is also proposed the general scheme to perform on-line diagnosis. This scheme is formed by the system, the diagnoser net, and an evaluator/selector, which performs the difference between the system and the diagnoser outputs and decides the type of fault or localization of the fault through a set of if-then rules. This scheme is important because it is structured to dynamically add a fault recovery block. As an example, a first approach for fault recovery is proposed, which its input is the output of the diagnoser scheme. This fault recovery scheme is based on IRWNets formalism, another contribution of this thesis, which its application is to task reconfiguration, in this case, for addressing fault recovery. Even when this recovery scheme is a first approach, it promises to be a good formalism for this proposal.

The study of the diagnosability as a property is an area very slightly exploited in the context of PN since this work presents the first approach in IPN. Therefore, it could be more deeply studied.

The fault recovery issue is another important matter that merits a deeper study, because the approach here presented promises to be a good technique to focus in fault recovery due the use of IRWNets that seems to be well adapted to support on-line task reconfiguration.

Bibliography

[1] L. Aguirre, A. Ramírez, O. Begovich. *"Design of asymptotic observers for discrete event systems"*. Proc. of the IASTED international conference on intelligent systems and control. pp. 188-193, Santa Barbara, USA. Oct. 1999.

[2] L. Aguirre-Salas, O. Begovich, A. Ramírez-Treviño, *"Observability in Interpreted Petri Nets using Sequence Invariants"*. In Proceedings of the IEEE Conference on Decision and Control, pp. 3602-3607, 2002.

[3] Alcaraz-Mejía Mildreth, López-Mellado Ernesto, Ramírez-Treviño Antonio, Rivera-Rangel Israel. *"Petri Net Based Fault Diagnosis of Discrete Event Systems"*, Conference on Systems, Man, and Cybernetics, Octubre 2003. Washington, USA.

[4] A. Alexandru, and S. Ungureanu. *"Using Expert Systems for Fault Detection and Diagnosis in Manufacturing Systems"*. Report #A153743, 1998.

[5] A. Asperti, and N. Busi. *"Mobile Petri Nets"* Technical Report UBLCS-96-10, University of Bologna, Italy, 1996.

[6] E. Badouel and J. Oliver. *"Reconfigurable Nets: A Class of High Level Petri Nets Supporting Dynamic Changes"*. Workflow Management Congress, CSR 98/07, Lisbon, 1998.

[7] T. C. Barros, A. Perkusich, and J. C. A. de Figueiredo, *"A Coloured Petri Net Based Apprach for Resource Allocation and Fault Tolerance for Flexible Manufacturing Systems"* 18th International Cconference on Applications and Theory of *PN*, June 1997.

[8] S. Bavishi and E. Chong, *"Automated Fault Diagnosis Using a Discrete-Event Systems framework"*. In Proceedings of the 9th International Symposium Intell. Contr., pp. 213-218, 1994.

[9] M. Buscemi and V. Sassone. *High Level Petri Nets as Type Theories in the Join Calculus"* FOSSACS01, Ed. F. Honsell and M. Miculan, N° 2030. Serie LNCS, pp. 104-120. Springer, 2001.

[10] R. Campos. *"Modular and Adaptive Modeling of Workflow Management Systems"* M.Sc. Thesis. CINVESTAV-IPN, Unidad Guadalajara. México, October 2002.

[11] R. Campos-Rodríguez, M. Alcaraz-Mejía, E. López-Mellado. *"Adaptive Models of Manufacturing Systems"*. Computational Engineering in Systems Applications, June 2003.

[12] C. G. Cassandras and S. Lafortune. *"Introduction to Discrete Event Systems"* Kluwer Academic Publishers, 1999.

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

[14] A. Fanni, A. Giua, and N. Sanna. *"Control and Error Recovery of Petri Net Models with Event Observers"* Proceedings of the 2nd International Working on Manufacturing and Petri Nets, pp. 53-68, 1997.

[15] P. J. Fielding, F. DiCesare and G. Goldbogen. *"Program Augmentation for Error Recovery in Automated Manufacturing: A Formulation"* Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 1988.

[16] P J. Fielding, F. DiCesare and G. Goldbogen. *"Error Recovery in Automated Manufacturing through the augmentation of programmed processes"* Journal on Robotics Systems, Vol. 5 No. 4, 1988.

[17] E. García, F. Morant, R. Blasco-Giménez, A. Correcher, and E. Quiles. *"Centralized Modular Diagnosis and the Phenomenon of Coupling"*. 6th International Workshop on Discrete Event Systems, 2002.

[18] A. Giua. *"Petri Net State Estimators Based on Event Observation"* Proceedings of the 36th Conference on Decision and Control, pp. 4086-4091, December 1997.

[19] W. L. Heimerdinger, and C. B. Weinstock. *"A conceptual Framework for System Fault Tolerance"* Technical Report from the Software Engineering Institute, Carnegie Mellon University, October 1992.

[20] R. Isermann. *"Process Fault Detection Based on Modeling and Estimation Methods"* Automatica 20, pp. 387-404, 1984.

[21] R. Isermann. *"Supervision, Fault-Detection, and Fault-Diagnosis Methods and Introduction"* Control Engineering Practice, Pergamon Press, Vol. 5, pp. 639-652, 1997.

[22] A. Khatav and E. Niel. *"State Feedback Stabilizing Controller for the Failure Recovery of Timed Discrete Event Systems"* 6th International Workshop on Discrete Event Systems, October, 2002.

[23] I. Koh, and F. Dicesare. *"Modular Transformation Methods for Generalized Petri Nets and Their Application to Automated Manufacturing Systems"*. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 21, No. 6, November/December, 1991.

[24] J.C. Laprie. *"Dependability: Basic Concepts and Terminology"* Vol. 5 of Dependable Computing and Fault Tolerant Systems. Springer Verlag, 1992.

[25] F. Lin, J. Markee, and B. Rado, *"Design and Test of Mixed Signal Circuits: A Discrete-Event Approach"* In Proceedings of the 32nd CDC, pp. 246-251, 1993.

[26] F Lin, *"Diagnosability of Discrete-Event Systems and its Applications"*, Journal Discrete Event Dynamic Systems, Vol.4, No. 2, pp. 197-212, 1994.

[27] S. Leonhardt, and M. Ayoubi. *"Methods of Fault Diagnosis"* Control Engineering Practice, Pergamon Press, Vol.5, No. 5, pp. 683-692, 1997.

[28] P. Loborg. *"Error Recovery in Automation An Overview"* Spring Symposium on Detecting and Resolving Errors in Manufacturing Systems, Standford, Ca, USA, 1994.

[29] E. López Mellado. *"Introducción a las Redes de Petri"* Facultad de Ciencias Físico-Matemáticas, Universidad Autónoma de Nuevo León, 1997.

[30] I. Loránt. *"Traditional Fault Diagnosis"* University of Miskolc, Activity of the MODIFY TEMPUS project LNS-FD-1.2, 1996.

[31] L. Magni, R. Scattolini, and C. Rossi. *"A Fault Detection and Isolation Method for Complex Industrial Systems"* IEEE Transactions on Systems, Man, and Cybernetics, Vol. 30, No. 6, November 2000.

[32] M. E. Meda A. Ramírez and A. Malo. *"Identification in Discrete Event Systems"*. IEEE International Conference Systems, Man and Cybernetics, pp. 740-745, San Diego, USA. Oct. 1998.

[33] T. Murata. *"Petri nets: Properties, Analysis and Applications"* Proc. IEEE, Vol. 77, No. 4, pp. 541-580, Apr. 1989.

[34] C. Ortega Moody. *"Sistemas Tolerantes a Fallas en Sistemas de Manufactura Flexible"*. Tesis de Maestría. CINVESTAV-IPN, Unidad Guadalajara. México, 1998.

[35] J. Padberg. *"An Outline of Rule-Based Refinement for Petri Nets"* In Proceedings of the Colloquium on Formal Methods for concurrency by the Gesellschatt für Informatik, August 1996.

[36] A. Ramirez-Treviño, I. Rivera-Rangel and E. Lopez-Mellado. *"Observer Design for Discrete Event Systems Modeled by Interpreted Petri Nets"* In Proceedings of the IEEE ICRA, pp. 2871-287, April 2000.

[37] A. Ramirez-Treviño, I. Rivera-Rangel and E. Lopez-Mellado. *"Observability of Discrete Event Systems Modeled by Interpreted Petri Nets"*. In IEEE Transactions on Robotics and Automation, Vol. 19, No. 4, August 2003.

[38] J. I. Rivera. *"Observability and Modular Synthesis of Petri Net Models of Discrete Event Systems"*. Ph. D. Thesis in press. CINVESTAV-IPN, Unidad Guadalajara. México, 2004.

[39] A. Santoyo, I. Jiménez-Ochoa, A. Ramírez-Treviño, *"A Complete Cycle for controller Design in Discrete Event Systems"*. In Proceedings of the IEEE Conference on Systems, Man and Cybernetics, pp. 2688-2693, 2001.

[40] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. *"Diagnosability of Discrete-Event Systems"*. IEEE Transactions on Automatic Control, Vol. 40, No. 9, September 1995.

[41] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. *"Failure Diagnosis Using Discrete-Event Models"*. IEEE Transactions on Control Systems Technology, Vol. 4, No. 2, March 1996.

[42] I. Seilonen, P. Appelqvist, A. Halme, and K. Koskinen. *"Agent-based Approach to Fault-Tolerance in Process Automation Systems"* Proceedings of the 3rd International Symposium on Robotics and Automation, September 2002.

[43] M. Silva, E.Teruel, R. Valette, and H. Pingaud. *"Petri Nets and Production Systems"* In Lectures on Petri Nets II: Applications, Lecture notes in Computer Science 1492, p.85-124, Springer Verlag, 1998.

[44] M. Zhou and F. Dicesare. *"Adaptive Design of Petri Net Controllers for Error Recovery in Automated Manufacturing Systems"* IEEE Transactions on Systems, Man and Cybernetics, Vol. 19, No. 5, September 1989.

[45] M. Zhou and K. Venkatesh. *"Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach"*. Series in Intelligent Control and Intelligent Automation, Vol. 6. World Scientific,1999.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL I P N
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

DIAGNOSTICO DE FALLAS EN SISTEMAS DE MANUFACTURA DISCRETOS

del (la) C.

Mildreth Isadora ALCARAZ MEJÍA

el día 27 de Agosto de 2004.

[illegible]

Dr. Luis Ernesto López Mellado
Investigador CINVESTAV 3A
CINVESTAV Unidad Guadalajara

[illegible]

Dr. Félix Francisco Ramos Corchado
Investigador CINVESTAV 2B
CINVESTAV Unidad Guadalajara

[illegible]

Dr. Luis Isidro Aguirre Salas
Profesor
Universidad de Guadalajara, Centro
Universitario de la Costa Sur,
Departamento de Ingenierías