# FOURIER-MOTZKIN METHODS FOR FAULT DIAGNOSIS IN DISCRETE EVENT SYSTEMS

by

AHMED KHELFA OBEID AL-AJELI

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
May 2017

# Abstract

The safety and reliability of large complex systems play an important role in the availability of the services provided by them. Unfortunately, fault occurrences in such systems are usually unavoidable. Fault diagnosis addresses the problem of detection and isolation of these fault occurrences. Thus, developing automatic approaches to obtain accurate and timely diagnosis decisions in such systems enhances their safety and reliability. It is well known that the problem of fault diagnosis under partial observation is a complex problem; and the challenge to solve this problem is to find a compromise between the space complexity and time complexity. The classic method to solve the problem is by constructing an automaton called a *diagnoser*. This method suffers from the *state explosion problem* which limits its application to large systems.

In this thesis, the problem of fault diagnosis in partially-observed discrete event systems is addressed. We assume that the system is modelled by Petri nets having no cycle of unobservable transitions. The class of *labelled* Petri nets is also considered with both bounded and unbounded cases. We propose a novel approach for fault diagnosis using the Integer Fourier-Motzkin Elimination method. The main idea is to reduce the problem of constructing the diagnoser to a problem of projecting between two spaces. In other words, we first obtain a set of inequalities derived from the *state equation* of Petri nets. Then, the elimination method is used to drop the variables corresponding to the unobservable transitions and we design two sets of inequalities in variables representing the observable transitions. One set ensures that the fault has occurred, whereas the other ensures that fault has not occurred. Given these two sets, we have proved that the occurrences of faults can be decided as any other diagnoser can do. The obtained result are extended to diagnose violations of constraints such as service-level agreement and Quality of Service, which is of particular interested in telecommunication companies. We implement our approach and demonstrate gains in performance with respect to existing approaches on a benchmark example.

*Dedicated to my parents and family*

# ACKNOWLEDGEMENTS

I would like to take this opportunity to acknowledge all people who participated in this work in many different ways. I will start by expressing my deep gratitude to my supervisors Dr David Parker and Dr Behzad Bordbar. Dr Behzad introduced me to this field and guided me during my research. Discussions and comments I had with him were essential to produce this work. Also, I feel lucky to have Dr David who helped me to form the final page of my research represented by the thesis. His useful feedback and comments significantly improved the picture of the thesis to be in the present form.

Also, I would like to thank the School of Computer Science at the University of Birmingham represented by the head of School Professor Andrew Howes for providing a great environment and atmosphere for doing research of high standards. Another thank goes to my thesis group members; Dr Martin Escardo and Dr Nick Hawes. They monitored my progress and discussed with me many details about the work and gave me good and useful comments. A special thank to all of my colleagues and friends with whom I had a great time providing me the advice and help when needed.

In addition, I acknowledge the role of Iraqi government represented by Ministry of Higher Education and Scientific Research and Iraqi Cultural Attaché in London for supporting and providing the fund needed to finish my study. I am particularly grateful to University of Babylon and College of Information Technology for giving me this opportunity to achieve my goal.

Last but not least, I would like to say that words cannot express the meaning of appreciation when mentioning the role of the family; my wife, daughter and son. They provided me all support I need for doing my work. My sincere thank to them.

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

## 1.1 Fault diagnosis

A fault is defined as an undesirable change in the property of a system or its components. Fault diagnosis is the process whereby the faults are detected and isolated. The area of fault diagnosis has received considerable attention in the past three decades from both academia and industry. Within this area, algorithms and techniques from control theory and artificial intelligence fields have been applied. These algorithms and techniques are capable of determining whether the state of the system is normal or faulty and they further isolate the faults, i.e. determine the fault type, if any. Generally speaking, two types of fault have been recognised in the literature: (i) permanent and (ii) intermittent. When a permanent fault occurs, the shape of change in the property is abrupt and this change propagates to all states following the faulty state. In the case of intermittent faults, this change occurs repeatedly, i.e. the fault events are followed by reset events which return the system to its normal state. Sometimes, these faults are captured as events in the systems, but other times they represent violations of constraints and are not events in themselves.

Several reasons make the fault diagnosis process an important task. First, the faults cannot be avoided, especially for complex and large systems. In addition, their consequences and impacts could present a risk to not only the systems themselves, but also to society in general. Secondly, there is the difficulty of providing partial information about the occurrence of faults. In fact,

due to technology limitations and budget issues, not all parts of a system can be monitored and consequently we cannot observe all events which occur. These reasons, among others, motivate the research in this area in order to create efficient automated methods which provide accurate diagnosis decisions as soon as faults occur.

A variety of methods and techniques have been proposed to address the problem of fault diagnosis. These methods and techniques have different schemes in their design and implementation, in addition to their theoretical foundations. Most of the proposed methods can be broadly categorised into: (i) knowledge-based systems methods representing experts systems and other artificial intelligence methods [1]; using these methods, the knowledge of experts is captured in the form of a set of rules; (ii) qualitative model-based methods [2]; and (iii) quantitative model-based methods [3].

Knowledge-based methods are particularly applied when it is difficult to obtain a model of the system to be diagnosed. In other words, the level of details of such systems cannot be captured as an abstract model describing their dynamical behaviour. On the contrary, qualitative model-based methods (also called discrete event system (DES) methods) have an advantage over the other methods in that they require no deeply detailed modelling [4]. Using such methods, the dynamical behaviour of the system is captured by states and the corresponding transitions between these states in a form of a modelling formalism. On the other hand, the quantitative methods adopt mathematical models used to compare the actual measurements of sensors with the predicted ones. All methods belonging to these different categories use two different implementation frameworks: offline and online [5]. With an online framework, the diagnosis process is performed during the normal operation of the system. In the offline framework, the system is tested before operation by making a number of tests and observing the outputs of the system to determine its behaviour. In effect, offline diagnosis is more flexible than online diagnosis; in addition, the state of the system is not changed unless we choose to make these changes. While in the case of online diagnosis, the state of the system is continuously changed without even having control of some of these changes. Thus, working in online mode is more difficult than working offline.

Discrete event systems are of particular interest and they represent a wide range of systems, see [6] for further information. In fact, the study of fault diagnosis for such systems has attracted a great deal of attention, in particular, under partial observation [2]. Fault diagnosis in partially-observed DES requires two different problems to be addressed: diagnosability and fault diagnosis. Diagnosability is an essential property by which we can ensure that any fault can be diagnosed within a finite delay. A popular approach to address these problems is to assume the existence of a formal representation of the behaviour of the system being analysed (often called the plant), as captured in some modelling formalism. Two commonly used formalisms are automata and Petri nets [4, 7–10]. Using these formalisms, faults are modelled as unobservable transitions. Among others, the seminal paper by Sampath *et al*. [4] formulates the diagnosis and diagnosability problems for systems modelled by automata. An automaton called a diagnoser is pre-compiled from the model of the system to: i) verify the diagnosability offline and ii) diagnose faults online. Also, Petri nets provide a rich modelling environment and are widely used in model-based fault diagnosis and diagnosability, see for example [8, 11–17], extending the original definition and standard notions.

Furthermore, using these two modelling formalisms, different architectures for the proposed approaches have been presented: centralised, decentralised [18–21] and distributed [7, 22, 23]. This classification is based on the information used from the global system and how the diagnosis decisions are made from this information. In the centralised architecture, the global model of the system is used for building a single diagnoser which monitors the whole system. Similarly, the global model is used within the decentralised architecture; however, a set of local diagnosers are created from this model and then each diagnoser is responsible of monitoring its local site. As opposed to using the global model, the distributed architecture uses individual diagnosers constructed from partial (local) models of the system.

In this thesis, we address the fault diagnosis problem and not the diagnosability problem; focusing on partially-observed DES, based on Petri nets. We adopt the centralised architecture to address the problem assuming that the faults are permanent and diagnosed using the online mode. We also address both cases of faults, i.e. when faults are modelled as events and when

faults are in the form of violations of constraints. Furthermore, as Petri nets extend automata, the results in this thesis are thus also applicable to automata.

## 1.2 Problem statement

Fault diagnosis in partially-observed DES is a complex problem (NP-hard problem) and the difficulty in solving it involves finding the best compromise between space complexity (size of the diagnoser) and time complexity of the algorithm that uses the diagnoser to compute the diagnosis. As mentioned previously, this problem has first been studied in the automata framework [4]. The idea of the solution starts by creating from the model of the system an automaton called a diagnoser in which all events are observable. Although the diagnoser approach has significantly lower time requirements to compute the diagnosis, the space requirements are significantly higher. Thus, the application of this approach is limited to small systems. Another limitation consists of the inability to handle infinite systems (unbounded state space). Extended work, using the notion of *basis marking and justifications*, has been proposed in the Petri nets framework by Cabasino *et al*. [8]. The authors have presented a method in which the efficiency of the diagnoser approach could be improved by not considering all states in the system to be diagnosed. Also, this approach can be applied to infinite systems, but at the cost of increasing the computations required online which could be exponential in the worst case.

A different idea has been introduced in [10, 24] where they adopt the use of equations to address the diagnosis problem. In other words, the diagnoser is no longer represented as an automaton. More specifically, the fault diagnosis problem is reduced to an integer linear programming (ILP) problem, which is solved online every time an event is observed. Using this idea, space complexity has been reduced at the cost of time complexity, which could be exponential. We conclude this discussion by pointing out that the existing approaches have either a state explosion problem or high time requirements.

Another limitation of the existing approaches is shown where these approaches can only be applied to diagnose the faults captured as events in the model of the plant. As previously

explained, a common practice is to represent faults as a part of the plant's model. For example, in automaton and Petri net models of the plants, we create unobservable transitions for representing faults. However, this style of modelling the faults is not always realistic. Sometimes, a fault is created as a result of a violation of service-level agreement (SLA) or Quality of Service (QoS) [25–31]. For example, consider the right-first time (RFT) fault [32] which is of interest to telecommunication services. The RFT fault occurs when a process fails to complete a task the first time and it is forced to repeat a part of the task again. This happens when one or more tasks are repeated, indicating incorrect execution of the task in the first place. Such occurrences of a fault may result in violations of SLA, causing financial penalties or customer dissatisfaction.

If the fault is expressed as a violation of constraint, there is no event in the system that represents the fault. One can argue that if a fault is caused by a violation of a constraint, we can always modify the model of the plant to include extra transitions (and/or states) to model the occurrences of the fault. This would require alterations of the models which has been seen as not always acceptable by engineers. Since the SLA and QoS requirements change over time, if violations of such constraints are modelled by adding transitions, the model of the plant must change whenever such constraints are modified. In addition, in some cases, adding extra events or transitions may result in cumbersome models. To model an RFT fault, potential duplicates of many transitions must be created to mark undesirable repetition of the multiple events. This can result in a serious distortion of an originally elegant design, resulting in a large and complex model.

In the light of the above issues, this thesis aims to answer three main research questions, in particular:

- Is it possible to create a new approach for the fault diagnosis problem in partially-observed DES based on Petri nets so that the compromise between space complexity and time complexity is achieved? Note that the faults are modelled as events in the system model.
- If this is possible, would we be able to diagnose a different form of faults which represent no events, i.e. faults in the form of violations of constraints?

- What is the scope of the application of the proposed approach? In other words, what type of Petri nets can be used and under what assumptions can the proposed approach be applied?

## 1.3 Outline of the proposed approach

In this thesis, we introduce a new approach to address the classical problem of fault diagnosis in partially-observed discrete event systems modelled as Petri nets. The main idea of this approach incorporates using the Integer Fourier-Motzkin Elimination (IFME) method to build the diagnoser from a Petri net based on the *state equation* [33], explained later. Fourier-Motzkin Elimination (FME) was first introduced as a method to solve a set of inequalities in real variables [34–36]. It is an extension of the Gaussian elimination method, commonly used with a set of equations. Like Gaussian elimination, FME eliminates variables from a set of inequalities, obtaining inequalities with fewer variables. The IFME method is an extension of the classic FME to cope with integer-valued variables [37, 38]. Another important application of FME, and the one used in this thesis for fault diagnosis, is to project the space defined by a set of inequalities in $n$ variables onto another space defined by a set of $n' < n$ variables [39].

The basic idea is to use this elimination method to omit the variables corresponding to unobservable transitions in the original set of inequalities, derived from the state equation, and design two sets of inequalities for each fault type. One set ensures that a fault has occurred (all sequences of the Petri net producing the given observation are faulty sequences) and the other ensures that the fault has not occurred (all sequences that produce the given observation are non-faulty sequences of the system).

More specifically, starting from a Petri net, a set of inequalities, denoted $I$, based on the state equation is produced. This set consists of integer-valued inequalities in variables that represent the number of firing transitions. The occurrence, or absence, of a fault can also be expressed by an inequality. Adding individually each of these inequalities to $I$ yields two sets of inequalities, to which we apply the IFME method to eliminate the variables corresponding to

unobservable transitions ending up with two sets of inequalities $R$ and $R'$ in variables representing the observable transitions. Since all variables relate to observable events, the advantage of using $R$ and $R'$ is that we can check for a given sequence $\sigma$ if the projection to observable events satisfies $R$ and $R'$. As a result, these sets of inequalities are used to decide about the occurrence of the fault as any other diagnoser would do.

## 1.4 Contributions of the thesis

The contributions of this thesis are embodied in the following key points:

1. We introduce a new approach of fault diagnosis in partially-observed DES based on the FME method. To the best of our knowledge, the FME has never been used in this context. This new approach is applied in the Petri nets framework to produce a diagnoser used for computing the diagnosis. In effect, the diagnoser is no longer represented as an automaton but as sets of inequalities in variables representing the observable transitions. These sets of inequalities are derived offline from the state equation in the Petri nets. Each pair of these sets is used to diagnose faults from a single fault type. Furthermore, since it is always possible to transform automata to Petri nets, application of our approach in automata models is straightforward.

2. The introduced approach can also be applied to different complex fault forms such as those in the form of violations of constraints, in which the faults are not captured as events in the model of the system. The SLA and QoS violations are examples of such faults. Current fault diagnosis methods often model faults as new events in the system by modifying the model. In the case of dealing with SLA and QoS, modelling a fault as an event may result in large models. As the creation of diagnosers is super exponential, this causes serious problems. Our proposed approach to address this problem not only provides a solution avoiding extra complexity, but is also more acceptable from an industrial point of view because the provided solution does not require modifying the model of the system every

time those constraints are changed.

3. We address the fault diagnosis problem in both finite and infinite systems without making any modifications to deal with each case. The FME method enables the representation of the diagnoser as sets of inequalities and not as an automaton which has a bounded number of nodes.

4. A software tool implemented in Java has been developed to create the diagnoser. In addition, a new emerging technology called complex event processing (CEP) is employed to implement the diagnoser in order to compute the diagnosis states when observing a sequence of events online.

5. A comparison has been performed between the proposed approach and the diagnoser approach. This comparison is based on a benchmark example which is presented in WODES and has been used in previous works also for comparison purposes [40–42]. The experimental results show that the proposed approach provides justified superior performance and scalability, enabling the application of this approach to large complex systems.

## 1.5 Publications related to this thesis

The majority of the present work of this thesis has been presented as conference and journal papers. The results presented in Chapter 4 were first published in [43]. We have also submitted a revised copy of a journal paper entitled *A New Approach for Failure Diagnosis in Petri Net Models of Discrete Event Systems Using Fourier-Motzkin method* to the *Automatica* journal. This paper represents the results obtained in Chapter 5. Relevant to Chapter 7, we presented in [44] an approach addressing the problem of diagnosis and diagnosability of violations of constraints. The presented approach extended the existing theory of fault diagnosis and diagnosability in DES. Unfortunately, this approach still suffers from the state explosion problem inherited from existing approaches. Thus, we applied the IFME to address the problem of violations of constraints in

Chapter 7, and the obtained results, in the form of a paper entitled *On Diagnosis of Violations of Constraints in Petri Net Models of Discrete Event Systems Using Fourier-Motzkin Method*, were first accepted for publication in *27th International Workshop on Principles of diagnosis (DX2016)*.

## 1.6 Thesis structure

This thesis is structured as follows. A literature review of the relevant work which address the problem of fault diagnosis in partially-observed DES is first covered in Chapter 2. We classified this work into two categories: automata models and Petri net models. Also, we consider the case of decentralised and distributed diagnosis.

Some general background on the Petri nets theory is offered in Chapter 3, in addition to a description of both the FME method and its extension to cope with integer solutions as well as their theoretical foundations. The last part of this chapter is devoted to delivering an introduction to complex event processing and one of its widely used examples named *Esper*.

Chapter 4 starts with a special case in which the problem of fault diagnosis in partially-observed DES is addressed in *acyclic* Petri nets. We propose the use of the IFME method to diagnose a single fault. Based on the notion of the state equation in Petri nets in addition to expressing the faults as inequalities, we prove that the problem of fault diagnosis can be reduced to a projection problem (eliminating a set of unwanted variables from a set of linear inequalities). Also, algorithms are written to implement the proposed approach, supported by an illustrative example for better understanding.

Chapter 5 provides an extension of the work to tackle the problem of fault diagnosis in more general Petri nets, in particular, Petri nets having no cycle of unobservable transitions, where each transition has a unique label. We elevate the notion of tracking the diagnosis history to avoid the problem of lack of order information in the state equation. Then, algorithms implementing the extension and their computational complexities are explained in detail.

Additionally, the work in Chapter 6 is expanded to the case where two or more transitions

might share the same label. In other words, the IFME method is extended to address the problem of fault diagnosis in labelled Petri nets. This extension includes using the *enabling condition* in Petri nets to deal with the nondeterminism resulting from sharing the labels. We present a supplemented definition of the diagnoser and algorithms to construct the diagnoser offline in addition to performing online diagnosis considering this type of Petri nets.

Furthermore, the problem of fault diagnosis in the form of violations of constraints is studied and addressed in Chapter 7. The SLA and QoS violations are some examples of such faults. We present a special case of these faults called right-first time faults to clarify the notion of this type of fault and how the IFME method can also be adopted to diagnose such faults.

Chapter 8 covers the implementation and evaluation of the proposed approach. Considering a benchmark example representing a manufacturing system, we compare the performance of our approach with the performance of the diagnoser automaton approach for Petri nets. Two criteria are used for comparison, namely the diagnoser size created offline and the time for computing the diagnosis. Also, the implementation details using Java and the CEP Esper engine are given.

Finally, conclusions and future directions to expand the current work and apply it for other Petri nets classes such as timed Petri nets are the topics of the last chapter. In addition, the study of diagnosability, where the diagnosis of any fault can be ensured within a finite delay, in the context of the proposed approach is considered.

# CHAPTER 2

# RELATED WORK

The aim of this chapter is to review the state-of-the-art methods addressing the fault diagnosis problem in DES under partial observation. Based on two modelling formalisms, namely automata and Petri nets, we classify these methods and describe their proposed notions to tackle the problem in question. We begin with automata model approaches and in particular, the initial work which developed the theory and methodology of fault diagnosis and diagnosability in DES.

Then, we describe the notions introduced in automata models which have been extended to the Petri net model. Different directions of research which have been explored in this regards are considered. We end this chapter by reviewing the fault diagnosis approaches based on decentralised and distributed frameworks.

## 2.1 Automata-based approaches

The fault diagnosis problem in partially-observed DES was first studied in an automata framework [4, 45–47]. In fact, the idea proposed and the theory developed in these works have their roots topics such as *state estimation* and *observability* [48, 49], which concerns estimating the current state of the system under partial observation; and *invertibility* [50] which addresses the problem of reconstructing all sequences of events (corresponding to an observed sequence) in the system. According to this theory, the automaton modelling the system behaviour has two sets of events: observable and unobservable. Further, the unobservable events set has two kinds of events: faulty

and non-faulty.

In order to achieve the diagnosis task, there are two assumptions: 1) The system is live, i.e. the system cannot reach a state in which there is no event possible. 2) No cycle of unobservable events exists in the automata models being analysed. Based on these assumptions, a pre-computed (compiled) finite-state automaton is built offline from the model of the system to be diagnosed. This automaton is called a *diagnoser automaton* in which arcs are labelled by observable events and states represent a subset of system states. Each state is provided with a diagnosis label explaining whether faults have or have not occurred, or may have occurred before reaching this state. In other words, each diagnosis state is an estimation of the current state of the system after observing a sequence of events.

Having the diagnoser, the computation of the diagnosis can then be accomplished based on the notion of string matching. In other words, the diagnosis of faults is performed by finding an exact match for a string of observed events in the diagnoser, then checking the diagnoser state reached from the initial state, by tracking events in the diagnoser matching that string.

Using the diagnoser automaton, all of the information about the system behaviour is compiled offline. In other words, each diagnoser state is a possible diagnosis and each diagnosis represents a diagnoser state. Also, the process of computing the diagnoser requires no more than triggering the diagnoser state upon observing an event, which is time-efficient. However, since constructing the diagnoser takes into account enumerating all possible states of the system, the size of the diagnoser is exponential in the number of states of the system. This causes a state explosion problem which can limit its application in large and complex systems [45].

Two remedies can be attempted to address this state explosion problem. The first involves constructing separate diagnosers, where each monitors a different fault type. In the second remedy, the construction of diagnoser is moved online by building the diagnosis states upon observing events. In which case, the need to store a complete diagnoser is avoided and thereby the space requirement is substantially improved, but at the cost of the time requirement spent online.

The diagnoser automata approach described above adopts the notion of event-based diagnosis,

which is also adopted in this thesis. Hence, the decision about the occurrence of a fault and its type is made based on observing a sequence of events. Adopting this notion requires initialisation of the diagnoser and the system being diagnosed simultaneously. Thus, another direction which requires no such initialisation has been studied by Zad *et al*. [51] using the idea of state-based diagnosis. According to this idea, the state space is decomposed by faulty states as opposed to fault events, i.e. faults occur when reaching faulty states.

## 2.2 Petri net-based approaches

An alternative formalism to model DES is provided by Petri nets (PNs). The structure, analytical capabilities and distributed nature, as well as its ability to express non-regular languages and infinite systems are all reasons that have motivated researchers to adopt this formalism [10, 12, 24, 52–54]. Since the notion of events is locally captured in the state, this enables avoidance of the state explosion problem by not enumerating all the states of the systems.

Some proposed Petri nets diagnosis methods have considered the notion of observable and unobservable places [52, 55]; while others have adopted the notion of observable and unobservable transitions [8–10, 12, 53]. The focus of this thesis will be on the latter notion. In the following sections, a review of the main notions for fault diagnosis in Petri nets is provided.

### 2.2.1 The classic approach

A natural solution to address the fault diagnosis problem in Petri nets is by extending the diagnoser automata method as follows. Starting from a Petri net model, a *reachability graph* (RG) is constructed [33]. Since this graph is in the form of an automaton, the diagnoser automaton can be built as described in Section 2.1. Then, this diagnoser is used online to compute the diagnosis state whenever an observed event is received.

A different idea in which the diagnoser is directly created from a Petri net model, i.e. without constructing the RG, has been reported by Genc and Lafortune [53]. The authors introduced

the notion of a *Petri net diagnoser* having the same graphical structure of Petri nets. However, transitions between markings are performed under a different rule. In addition, each marking of the diagnoser has a subset of markings captured in the form of tokens. In fact, this Petri net diagnoser can be imagined as a special case of coloured Petri nets [56]. Similar to the diagnoser automaton, the Petri net diagnoser is created offline and then it is used online to compute the diagnosis.

Note that the aforementioned solutions can only be applied to bounded Petri nets. The problem of fault diagnosis in unbounded Petri nets has been investigated by Ushio and Onishi [52]. These authors have extended the diagnoser automata approach of [4] to a Petri net framework. The proposed notion adopts the concept of observable and unobservable places as opposed to observable and unobservable transitions, i.e. tokens are not observed in all markings. In addition, all transitions have been assumed to be unobservable. Under this different form of partial observation, two markings are not distinguishable if and only if their observable markings are identical. Note also that fault events in this form are still defined and modelled in Petri nets.

To construct the diagnoser using the method of [52], a *modified coverability tree* (a finite structure for infinite state space in unbounded Petri nets) is first generated from a Petri net model. From this tree, the diagnoser is constructed where arcs are labelled by observable markings as opposed to observable events. Then, diagnosis states are computed when observing a marking. Chung [57] has used a similar idea, but the only difference involves assuming that transitions are partially-observed. In effect, adding this assumption provides additional information to the diagnoser and thereby enhances the diagnosis process in general.

In all of the methods described above, the state explosion problem inherited from the diagnoser automata approach still exists because these methods consider enumerating all possible markings reachable from the initial marking. Thus, a range of methods has been proposed to avoid this problem and build smaller sized diagnosers.

### 2.2.2 Basis marking and justifications

To avoid the state explosion problem and present a compact representation of the state space of systems, an approach adopting *basis marking and justifications* has been proposed in [8, 54, 58]. Under the assumption that no cycle of unobservable transitions exists, it has been shown that a subset of reachable markings (basis markings) is sufficient to obtain all markings consistent with observable events, by firing only unobservable transitions starting from these markings. Essentially, linear algebraic constraints are formulated based on the state equation and enabling conditions in Petri nets. These constraints are used to determine the set of basis markings.

Generally, a basis marking is defined as a marking reachable by firing a sequence of observable transitions and the minimal sequence of unobservable transitions which is necessary to enable it. The vector of values, where each value represents the number of appearances of a transition in a given minimal sequence, is termed a *minimal explanation* or *justification*. In general, the set of basis markings and consequently the set of minimal explanations is not a singleton. Furthermore, the number of constraints characterising these sets is not fixed and depends on the length of the observed sequence. However, the set of basis markings is a singleton and the number of constraints is fixed under the assumption that the subnet of unobservable transition is cycle-free and backward conflict-free[1].

To obtain a set of minimal explanations, a tabular algorithm whose inputs are a marking and the current observed transition has been developed. This algorithm employs algebraic manipulations and it is applied to any Petri net whose subnet of unobservable transitions is cycle free. Obtaining this set leads to computing the set of basis markings corresponding to the observed sequence. Then, diagnosis states are decided based on this obtained information. In effect, four diagnosis states have been defined, which take the labels 0, 1, 2 and 3. The label 0 represents the non-faulty state; the label 1 means that a fault has occurred within a given sequence of observed events, but none of its justifications has this fault; the label 2 indicates the uncertain state where some justifications has a fault, but not all them have the same fault; and the label 3 refers to the faulty state. The states 2 and 3 can easily be distinguished by analysing the

[1] A subnet is called backward conflict-free if each transition has no common output place [59].

basis markings corresponding to the observed sequence. However, to distinguish between states 0 and 1, solution of an integer linear programming (ILP) problem is required. If this problem is infeasible, then state 0 is obtained; otherwise, we have state 1.

In the case of bounded Petri nets, the set of basis markings is finite, allowing the performance of most of the computations offline by creating a *basis reachability graph* (BRG) which is a deterministic directed graph. The number of nodes in this graph equals the number of basis markings, which is strictly less than the number of nodes in the corresponding reachability graph. However, the BRG requires more information to be encoded in the nodes and on the arcs. Hence, to implement the basis markings and justifications approach in bounded Petri nets, two algorithms have been developed [8, 54] for both constructing the BRG offline and computing diagnosis states online.

Note that the approach described above assumes that the labels of all transitions are unique. A relaxation of this assumption has been presented by Cabasino *et al*. [60] using the same notions introduced in [8, 54]. To sum up, fault diagnosis of bounded Petri nets by constructing the BRG offline can improve the time requirements to compute the diagnosis online. However, space requirements for storing the BRG graph could be very large. Moreover, the size of the BRG may still grow exponentially with the size of the state space. On the other hand, since such a graph is not available in the case of unbounded Petri nets, then all computations are moved online. The most burdensome part of these computations is spent in solving ILP problems, in which the structure of the constraints is not fixed and grows with the length of the observed sequence.

Jiroveanu *et al*. [12] have also used the notion of basis markings and minimal explanations. The authors proposed an online algorithm to estimate markings reachable from the initial marking whenever an event is observed. Analogously, they determined the set of minimal explanations which are necessary to enable the observed transition using a backward search algorithm. Under the assumption that the Petri nets under consideration are bounded with respect to the subnet of unobservable transitions, they showed that all markings corresponding to the observed sequence of events can be obtained by enumerating a subset of all markings reachable from the initial marking. Thus, the main difference between this approach and those of [8, 60] described earlier

in this section is the manner in which the minimal explanations set is computed. Moreover, since the computational complexity of the proposed algorithm relies on the size of the largest subnet of unobservable transitions, it can efficiently be applied to Petri nets having small unobservable subnets.

### 2.2.3 The integer linear programming (ILP) approach

The methods described previously create the diagnoser in the form of an automaton. This automaton is pre-computed offline to reduce the time required to compute the diagnosis online. This section and the following section review the methods which adopt algorithms to compute the diagnosis online without relying on a pre-computed (compiled) diagnoser. Thus, these algorithms can be imagined as *interpreted* diagnosers, which monitor a sequence of observed events online and make diagnosis decisions every time an event is observed.

One of the methods which follows this direction has been proposed in [10, 61]. In these works, integer linear programming (ILP) techniques are employed for fault diagnosis in Petri nets. Under the assumption that subnets of unobservable transitions are cycle free and no two transitions share the same label, the problem of fault diagnosis was reduced to an ILP problem using an online algorithm to compute the diagnosis. Essentially, this method starts from the state equation and enabling conditions to build a set of inequalities characterising all sequences associated with an observed sequence.

Based on the set of inequalities, two ILP problems are defined in order to determine diagnosis states. The first problem is a maximisation problem with respect to the number of occurrences of a fault transition in a given sequence, satisfying the set of inequalities in question; it concerns determining the normal and faulty states. To decide the uncertain state, a minimization problem is defined analogously.

Using this method, the state explosion problem can be avoided. However, time requirements are high and increase with the length of the observed sequence. More precisely, this method requires solving at most $r+1$ ILP problems (each of which takes exponential time) every time an observed sequence is observed ($r$ is the number of fault types). In addition, the number of

constraints defining the ILP problems grows with the number of observed events. Fanti *et al*. [23] have extended the same method to apply to labelled Petri nets[2]. This work has been presented under the assumption that no unobservable transitions occur after the last observed event.

Another idea which employs ILP techniques to address the fault diagnosis problem has been investigated in [24, 62]. Similarly, the investigated idea produces an *interpreted* diagnoser to then diagnose faults online. However, a different notion based on *g-marking* is introduced. A g-marking is a uniquely estimated marking in which some components have negative values as a result of firing observed transitions without considering the unobservable transitions which could enable them. Furthermore, a new rule for the enabling condition and firing of transitions under this marking is established. Based on the estimated g-marking, the set of associated possible sequences whose firings are necessary to enable the observed event is computed. Then, ILP problems are formulated using both the g-marking and this computed set.

In fact, the fundamental difference between both the methods is the need to estimate markings after an event is observed. The latter method introduces a new definition of markings (g-marking), however, it employs the same classic manner based on computing unobservable sequences associated with a given observed sequence and then updating the diagnosis state accordingly. While the former method avoids this by capturing the state information and enabling conditions in a set of inequalities (constraints) under which ILP problems are solved.

With respect to the computational complexity, the latter method requires solving an ILP problem for each fault transition and each new g-marking. This leads to solving a higher number of ILP problems than the former method. However, the number of constraints in these ILP problems does not grow with the number of observed events. Overall, the former method has an advantage over the latter in terms of time, but at the cost of the space required for holding the growing number of constraints.

[2]In labelled Petri nets, more than one transition may have the same label and could be enabled simultaneously.

### 2.2.4 Net unfolding approach

To build a compact representation of the semantics of Petri net models in order to consequently enhance the analysis of the state space, *net unfolding* techniques have been developed. The essential notion is based on *partially ordered* analysis as opposed to *totally ordered* analysis in which all possible solutions are considered. The abstract notions and formalisms relevant to the net unfolding first appeared in [63]. Subsequently, the net unfolding techniques were adopted to search and analyse the state space of Petri nets [64]. Using such techniques, the size of the state space is significantly reduced via sharing the common prefixes between executions (firing sequences) of nets. In addition, the executions which are only different in their interleaving firing transitions are only represented once.

The net unfolding is a process whereby the set of firing sequences in Petri nets is represented. For this process, another structure called an *occurrence net*, consisting of all these sequences, is created. This occurrence net is also a Petri net supported by a labelling function to map between transitions and places in the original Petri net and those in the occurrence net. In addition, the occurrence nets have some structural characteristics, one of which is that they are *acyclic*. The relation between the net unfolding and reachable marking in Petri nets is obtained via the notion of *configuration*. The configurations are subsets of the occurrence nets which represent possible firing sequences in Petri nets. Each configuration has a final state (a marking) which exactly represents a reachable marking in the Petri net.

Benveniste *et al*. [65] have proposed an approach for fault diagnosis in asynchronous telecommunication networks. The authors have handled this problem using net unfolding techniques. They showed that their method can be applied to *safe* Petri nets[3] where no cycle of unobservable transitions exists. The notion of a *diagnosis net* has been introduced, where all solutions of a diagnosis are captured. Also, an algorithm to construct the diagnosis net online has been developed, i.e. the proposed method has adopted the idea of the *interpreted* diagnoser as opposed to the *compiled* diagnoser discussed previously, in which case, avoiding the state explosion

[3] A safe Petri net is a Petri net in which every place and every marking including the initial marking has at most one token.

problem comes at the cost of increasing the time of computing the diagnosis performed online. In other words, this method is limited to applications where memory constraints are tight.

The notion of addressing the fault diagnosis problem by means of net unfolding has been extended by Esparza and Kern [66], and Haar *et al*. [67]. Generally speaking, this extension removed the assumption where no cycle of unobservable transitions is permitted.

## 2.3 Fault diagnosis using supervision patterns

All approaches previously described address the problem of fault diagnosis where the fault is modelled by a single event. This is not always the case, as sometimes we say that a fault occurs if a sequence of events occurs, e.g. detecting intrusions and attacks in networked systems [68]. To diagnose such faults, the theory and methodology of fault diagnosis need to be extended. Genc [69] has investigated the problem of diagnosing patterns of events. Two types of patterns have been defined and a methodology to diagnose such patterns have been developed. This methodology has been applied in the automata framework representing a generalisation of the diagnoser automata approach introduced by Sampath *et al*. [4].

Furthermore, the notion of patterns has been employed to separate between the objectives of diagnosis and the system specification [70]. The concept of a *supervision pattern* has been introduced, in which the fault diagnosis problem has been redefined and a new method to construct the diagnoser has been presented. The advantage of using supervision patterns is a unifying framework under which different and complex modes of faults, including permanent, intermittent and repeating faults can be diagnosed.

Addressing the fault diagnosis problem under this framework assumes that a finite state machine is employed to model the system being analysed. In addition, different supervision patterns are defined and modelled as automata whose language captures the diagnosis objectives. Then, construction of the diagnoser is performed as follows. Starting from the model of the system and an automaton representing a pattern, a synchronous product operation of both models is performed. Then, a determinisation operation on the resulting model is accomplished. By

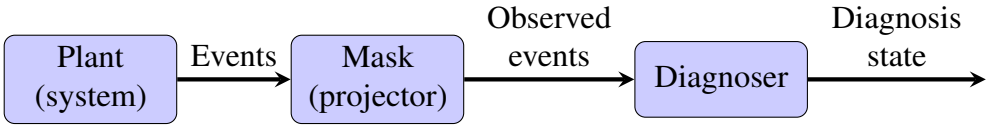



Figure 2.1: Centralised diagnosis architecture

observing a sequences of events, the diagnoser can decide whether this sequence has a pattern of events compatible with the defined pattern.

In fact, the notion of supervision patterns might resemble the notion of violations of constraints presented in Chapter 7 of this thesis. However, there is a fundamental difference consisting in the fact that our idea of diagnosing the violations of constraints is based on using models which are more expressive than automata. In effect, the approach presented in Chapter 7 assumes that systems being diagnosed are modelled by Petri nets. In addition, inequalities are employed to express patterns, enabling a broad range of these patterns to be defined. Furthermore, the methodology used to construct the diagnoser and perform the diagnosis online is completely different.

## 2.4 Decentralised and distributed diagnosis

In this section, different architectures for implementing the diagnoser and making diagnosis decisions are discussed. In effect, there are three main architectures: centralised, decentralised and distributed, which can be applied to both centralised and distributed systems.

In the centralised architecture, there is one global model of the system to be diagnosed and a single diagnoser that monitors the global state of the system. Based on one set of global observations, the diagnoser will make diagnosis decisions centrally as depicted in Figure. 2.1. All approaches mentioned in the previous sections implement this architecture, including the approach proposed in this thesis. As illustrated in the figure, there is a single global model of the system. The diagnoser can only see observable events passing through the mask. This mask can be imagined as a mapping function (projection); its inputs are observable and unobservable events and its outputs are observable events only. Using these observable events, the diagnoser

estimates diagnosis states.

The centralised architecture has the advantages of diagnosis accuracy and conceptual simplicity. However, this will be at the cost of high computational complexity and low maintainability, especially for complex and large systems such as distributed and communication systems [2]. In such systems, the diagnoser may not be able to compute the diagnosis centrally and then send all observations to the other parts. In addition, regarding security issues, sometimes observing the global model of the system is not preferable. For example, in some systems distributed over different legal entities, one entity may not wish that the other entities observe the detailed model of its part. All of these reasons have motivated researchers in the model-based fault diagnosis field to explore alternative architectures to implement the diagnoser.

### 2.4.1 Decentralised diagnosis

Debouk *et al*. [18] have presented a framework of the decentralised diagnosis for DES modelled by automata. Within this framework, the traditional notions introduced by Sampath *et al*. [4] have been extended to the decentralised version. To perform a decentralised diagnosis, local diagnosers are distributed over local sites. These local diagnosers are communicated between each other via a *coordinator* or *supervisor*. The coordinator is responsible for collecting local diagnosis decisions from the local diagnosers and then making final diagnosis decisions. Figure. 2.2 shows the general structure of this architecture, where the system is partitioned into $d$ local sites, each of which is monitored by its own local diagnoser. This local diagnoser can only observe its own subset of observable events. As a result, observable events of one site are unobservable events of the other site and vice versa. Furthermore, no communication between the local sites is carried out, since all communications are performed via the coordinator. However, the local sites still use the global model of the system during the generation of their local diagnosers.

To design the decentralised diagnosers, three key factors are to be considered :

- Establish protocols for communications between the local diagnosers and the coordinator.
- Balance between the computational complexity and the diagnosis accuracy.

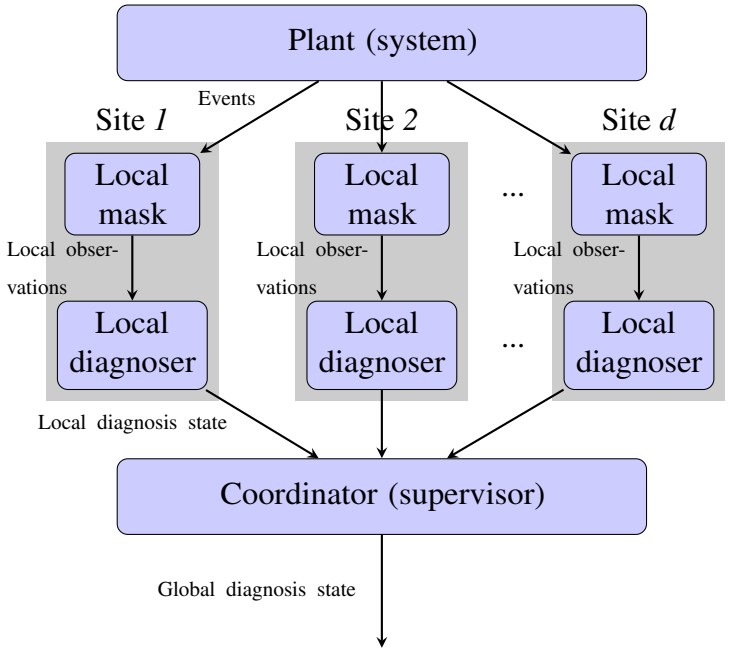



Figure 2.2: Decentralised diagnosis architecture

- Compare diagnosis results of decentralised diagnosers to their centralised counterparts.

In [18], three protocols called *Protocol 1*, *Protocol 2* and *Protocol 3* have been proposed. Implementing these protocols requires some assumptions and conditions of systems being analysed in order to obtain diagnosis results as successfully as the centralised diagnosis. Then, each protocol can be defined using three main elements: diagnostic information produced at local sites; the communication rule used by the local sites to communicate to the coordinator; and the decision making rule used by the coordinator to generate diagnosis decisions. Using these protocols, the diagnosis accuracy improves by going from protocol 3 to 2 to 1, whereas the computational complexity increases from protocol 3 to 2 to 1. However, all of these protocols make the same diagnosis decisions as the centralised architecture.

Analogously, the notions introduced in [18] for automata models have been extended to systems modelled by Petri nets by Cabasino *et al.* [21]. The contribution of this extension comprises of reducing the enumeration of all space states using the notion of *basis markings and justifications* described in Section 2.2.2. More specifically, diagnosis decisions made by local diagnosers are obtained based on finding the basis markings and solving a set of inequalities every time an event is observed. Then, these local decisions are sent to the coordinator to make

the final diagnosis decision.

The dependency of the local diagnosers on the global model of the system being diagnosed represents a major drawback of the aforementioned methods in this section. Pencolé *et al*. [71] have presented another approach whose basic notion is based on using temporal windows. The authors used these temporal windows to split observations and then compute the diagnosis locally for each component of the system. Then, the global diagnosis is obtained by merging diagnosis states computed locally and those of previous temporal windows.

A different approach in which faults are represented as violations of non-fault specifications has been proposed by Qiu and Kumar [19]. In this work, the non-fault specifications are modelled by automata for which algorithms of polynomial complexity have been developed for fault diagnosis. In addition, the proposed notion requires neither communication between the local diagnosers, nor between the local diagnosers and the coordinator, where the diagnosis is completely computed locally.

### 2.4.2 Distributed diagnosis

Distributed diagnosis has been investigated in [7, 22, 23, 53]. The motivation for this architecture for fault diagnosis is to improve scalability and robustness of diagnostics methodologies. In addition, the communication overhead due to exchanging observations is reduced. Within the distributed architecture, local sites do not refer to the global model of the system to be diagnosed, as the other architectures do. Each local diagnoser uses its own local observations and makes local diagnosis decision based on these observations. In this type of diagnosis, there is no need for the coordinator where the local sites communicate to each other directly. These communications are maintained by communication protocols, in order to exchange the information required to update the local diagnosis and satisfy the consistency. Ensuring this consistency is necessary to prevent conflicts in diagnosis decisions made by local sites. Figure. 2.3 depicts the general structure of this distributed diagnosis.

The challenge raised in this diagnosis architecture comprises of how to obtain diagnosis results as well as the centralised diagnosis in the presence of many obstacles, such as commu-

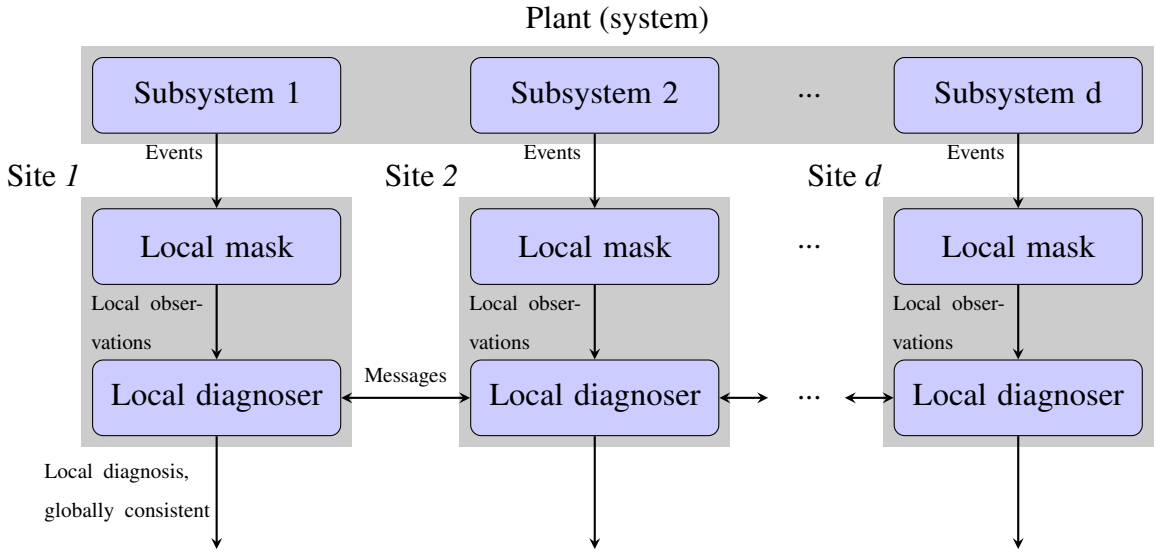



Figure 2.3: Distributed diagnosis architecture

nication delay, messages loss, preserving order of information and complexity issues [2]. To deal with these obstacles, different settings and methods have been proposed. One of the early works in this context has been presented by Genc and Lafortune [53] to address fault diagnosis in Petri nets. A new notion has been adopted in which the Petri net is partitioned into two subnets according to place-bordered nets with shared places chosen in line with some conditions. Each subnet represents a subsystem monitored by a local Petri nets diagnoser.

The communication protocol is implemented using an algorithm by which the communications between the two diagnosers are triggered upon occurrences of observable events. When communicating, the diagnoser that observed these events will send *message labels* to the other diagnoser in order to update the information about the shared places. Therefore, each state in the distributed diagnoser is augmented by this piece of information. Decomposing Petri nets into place-bordered nets, diagnosis states can properly be obtained as well as the centralised diagnosis. However, the proposed notion has the problem of overhead communication; in addition the *message labels* can grow arbitrarily. To overcome these problems and to extend the work in [53] to multiple subnets, the same authors have presented a modular approach for implementing a distributed diagnosis [7].

Starting from the results obtained by Genc and Lafortune [7] and Dotoli *et al*. [10], ILP techniques have also been applied in a distributed setting [23]. Based on these results, the

authors proposed a distributed diagnosis adopting the modular approach in labelled Petri nets. As mentioned previously, using ILP techniques in centralised diagnosis requires high time requirements. The distributed diagnosis mitigates this problem where a smaller set of inequalities and variables is considered for local diagnosis computations. As a result, time requirements for solving ILP problems online are reduced.

# CHAPTER 3

# BACKGROUND

## 3.1 Petri nets

A *Petri net* [33, 72–74] is defined as a four tuple $\mathcal{N} = (P, T, pre, post)$, where $P$ and $T$ are two nonempty finite sets of places and transitions, respectively. In this thesis, $\mathbb{N} = \{0, 1, 2, \dots\}$ is the set of non-negative integers, $\mathbb{Z}$ is the set of all integers and $\mathbb{R}$ is the set of real numbers. We respectively denote $m = |P|$ and $n = |T|$ as the number of places and transitions. Let $pre : P \times T \to \mathbb{N}$ and $post : P \times T \to \mathbb{N}$ be arcs weight matrices, where $pre(p,t)$ ($post(p,t)$) defines the weight of the arc directed from a place $p$ to a transition $t$ (directed from a transition $t$ to a place $p$). For a given transition $t$, an *input* (*output*) place of $t$ is a place $p$ such that $pre(p,t)$ ($post(p,t)$) is positive, respectively. Matrix $A = [a_{ij}]$ is the $m \times n$ incidence matrix of integers, where $a_{ij} = post(p,t) - pre(p,t)$ assuming that the set of places and transitions are ordered to correspond with the coordinates of the matrix.

We write ${}^\bullet t$ ($t^\bullet$) for the set of all input (output) places of a transition $t$, respectively. Also, we write ${}^\bullet p$ ($p^\bullet$) for the set of all input (output) transitions of a place $p$, respectively. A pair of a place $p$ and transition $t$ is called a *self-loop* if $p$ is both an input and output place of $t$. A Petri net is called *pure* if it has no *self-loop*. A *cycle* in a Petri net is a closed directed path from one node (place or transition) back to itself. A Petri net having no cycles is called an *acyclic* Petri net. For a set $Q \subseteq P$ of places, let $I(Q) = \{t \mid {}^\bullet t \cap Q = \emptyset \wedge t^\bullet \cap Q \neq \emptyset,\ t \in T\}$. This set defines the input transitions of places in the set Q incoming from places which are not in $Q$. The set $Q$ is said to

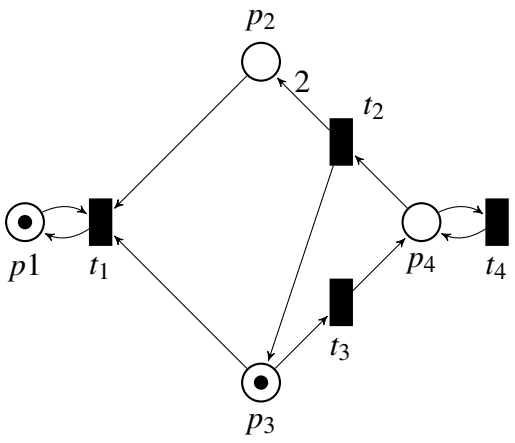



Figure 3.1: Example of a Petri net.

be a *deadlock* (*siphon*) when $I(Q) = \emptyset$. Also, a place $p$ is called a *source place* when ${}^\bullet p = \emptyset$. If a deadlock has no token-free source place, then we have a *circuit deadlock*.

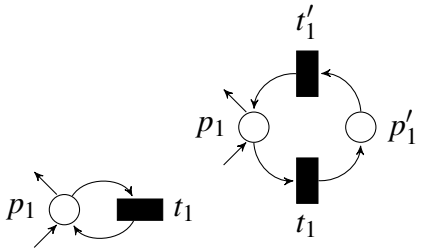
**Example 3.1.** Consider the Petri net $\mathcal{N}$ in Figure. 3.1 with the set of places $P = \{p_1, p_2, p_3, p_4\}$ and the set of transitions $T = \{t_1, t_2, t_3, t_4\}$. The matrices *pre* and *post* are:

$$pre = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{array} \quad post = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

and the incidence matrix $A$ is:

$$A = post - pre = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ -1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{pmatrix}$$

Note that the pairs $p_4t_4$ and $p_1t_1$ represent self-loops in this net, and the path $p_3t_3p_4t_2p_3$ represents a cycle. A non-pure Petri net can be transformed to one that is pure by adding a dummy transitions-place pair to open self loops [33] (see Figure. 3.2 below).




(a) Before (b) After

Figure 3.2: Transformation of a self-loop to a loop

A *state* of a Petri net, known as a *marking*, is represented as $M : P \to \mathbb{N}$ capturing the number of tokens in each place. We sometimes represent a marking as an $m \times 1$ matrix of non-negative integers. A transition $t$ is *enabled* at a marking $M$ if $M(p) \geq pre(p,t)$ for each $p \in {}^\bullet t$. We write $M \xrightarrow{t}$ to denote that $t$ is enabled at $M$. An enabled transition can *fire* resulting in a new marking $M'$, denoted by $M \xrightarrow{t} M'$. The firing vector $\mathbf{u}$ is defined as an $n$-dimensional column vector of the form $\mathbf{u} = (0,\ldots,0,1,0,\ldots,0)$, where the only 1 appears in the $j$th position, $j \in \{1,\ldots,n\}$, to indicate the fact that the $j$th transition is currently firing. Given $\mathbf{u}$ for a firing transition on marking $M$, we can find the reachable marking $M'$ by $M' = M + A\mathbf{u}$. A sequence of transitions $\sigma = t_1 t_2 \ldots t_l$ of $T$ is called *enabled* at a marking $M$, denoted by $M \xrightarrow{\sigma}$ if there are markings $M,\ldots,M_{l-1}$ so that $M \xrightarrow{t_1} M_1 \xrightarrow{t_2} \cdots \to M_{l-1} \xrightarrow{t_l}$ . Firing the sequence $\sigma$ results in the marking $M_l$ written as $M \xrightarrow{\sigma} M_l$ and we refer to $M_l$ as a state *reachable* from $M$ and $\sigma$ as the *firing sequence*. We write $R(\mathscr{N}, M)$ for the set of all reachable states from $M$ in a Petri net $\mathscr{N}$. The initial state of the system is represented by an *initial marking* $M_0$. We will write $(\mathscr{N}, M_0)$ for a Petri net with its initial marking $M_0$.

**Example 3.2.** Recall that the Petri net of Figure. 3.1 with an initial marking $M_0 = [1010]$. Only transition $t_3$ is enabled at the initial marking. Firing this transition results in marking $M_1 = [1001]$. Given a firing vector $\mathbf{u} = (0,0,1,0)$ and the matrix $A$ described in Example 3.1, this marking

can be found as follows:

$$M_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ -1 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

$$M_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

The set of all finite-length strings of the transitions in $T$ is denoted by $T^*$ and is called the *Kleene-closure* of $T$. As a result, members of $T^*$ are created from a concatenation of a finite number of elements of $T$. In particular, $T^*$ contains the empty string $\varepsilon$, so that $t\varepsilon = \varepsilon t = t$ for all $t \in T$. Every subset of $T^*$ is called a *language on the alphabet* $T$. Suppose that we have a sequence $\sigma$ of $(\mathcal{N}, M_0)$, then the *Parikh vector* $\# : T^* \to \mathbb{N}^n$ is a map which assigns to every sequence $\sigma$ a vector $\#(\sigma)$ in which each element represents the number of firings of each transition in $\sigma$. In other words, for $\#(\sigma) : T \to \mathbb{N}$, $\#(\sigma)(t)$ is the number of occurrences of $t \in T$ within the sequence $\sigma$. Sometimes, we write $\#(t, \sigma)$ to represent the number of the occurrences of $t$ in $\sigma$.

The set of sequences of transitions resulting in reachable markings is called the *language* of the Petri net and is denoted by $L(\mathcal{N}, M_0)$, i.e. $L(\mathcal{N}, M_0) = \{\sigma \mid \exists M\ M_0 \xrightarrow{\sigma} M\}$. Suppose that a destination marking $M$ is reachable from $M_0$ in a Petri net $\mathcal{N}$ through a sequence $\sigma$, we can then find $M$ using what is called *state equation* shown in the following:

$$M = M_0 + A\mathbf{x},\ \ M \geq \vec{\mathbf{0}} \tag{3.1}$$

where $A$ is the incidence matrix of $\mathcal{N}$, and $\mathbf{x} \in \mathbb{N}^n$ is a $n$-dimensional column vector with $\mathbf{x} = (x_1, \ldots, x_n)$ and $x_i = \#(t_i, \sigma)$ for $t_i \in T$. Then, for any sequence $\sigma$ of $\mathcal{N}$, there exists

$\mathbf{x} = \#(\sigma)$ satisfying (3.1). The converse is not always true. In some cases, e.g. *acyclic* Petri nets, the converse holds too.

**Definition 3.1.** *Firing count subnet* [75]: let $\nu = (\alpha_1, \ldots, \alpha_n)$ be a solution of the state equation in (3.1) for a Petri net $(\mathscr{N}, M_0)$ with a destination marking $M$. Then, the subnet $\mathscr{N}_\nu$ is called the firing count subnet with respect to $\nu$ where each transition $t_i$ in $\mathscr{N}_\nu$ is such that $\alpha_i > 0$ together with its input and output places and its connecting arcs. Markings $M_{0\nu}$ and $M_\nu$ denote the restrictions of $M_0$ and $M$ to places in $\mathscr{N}_\nu$.

Now, suppose that we have a Petri net $(\mathscr{N}, M_0)$, then the association of a label $e \in \Sigma$, where $\Sigma$ represents a set of labels (alphabet), to transitions in $\mathscr{N}$ is called a *labelling function*. This function is defined as $\lambda : T \to \Sigma \cup \{\varepsilon\}$, i.e. $\lambda(t) = e$ or $\lambda(t) = \varepsilon$ for $t \in T$. Also, this labelling function can be extended to the Kleene closure of $\Sigma$ by $\lambda : T^* \to \Sigma^*$ where for each sequence of transitions $\sigma$ and transition $t$, $\lambda(\sigma t) = \lambda(\sigma)\lambda(t)$. A *labelled* Petri net is defined as a four tuple $(\mathscr{N}, M_0, \Sigma, \lambda)$ in which we associate to each label $e \in \Sigma$ a set of transitions $\tau(e)$.

$$\tau(e) = \{t |\ t \in T, e = \lambda(t)\} \tag{3.2}$$

## 3.2 The Fourier-Motzkin Elimination method

The Fourier-Motzkin Elimination (FME) method was originally proposed for solving a set of linear inequalities and also to establish if the set is solvable [34–36, 76–78]. In other words, given a matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$, FME tests if a set of inequalities $I := A\mathbf{x} \leq \mathbf{b}$, where the vector of variables $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, has a solution. Then, if there exists a solution, FME will find it.

Another important application of the FME method is to solve the projection problem in which the space defined by a finite number of linear inequalities in $n$ variables is projected onto another space defined by another set of linear inequalities in $k < n$ variables [39, 79–81]. By projection, we ensure that for any solution to the original set of inequalities, there exists a solution to the

projected set and vice versa. Formally, given a set of linear inequalities

$$I := A\mathbf{x} \leq \mathbf{b}, \tag{3.3}$$

we create an equivalent set of linear inequalities in variables $\mathbf{y} = (x_1, \ldots, x_k)$, shown in (3.4), such that for any solution $y$ of (3.4) there exists a solution $x = (y, z),\ z = (x_{k+1}, \ldots, x_n)$ of (3.3) and vice versa.

$$R := B\mathbf{y} \leq \mathbf{d} \tag{3.4}$$

By creating this set of inequalities, we say that the set $x_{k+1}, \ldots, x_n$ of variables has been *eliminated*. In this thesis, we are interested in using the FME method for projection purposes and not to solve a set of linear inequalities. In other words, the problem of fault diagnosis is reduced to a projection problem in which we aim to eliminate all variables representing the unobservable transitions.

When using the FME method to address either of these problems, the variables are essentially eliminated one by one. To explain the notion of elimination, it is sufficient to describe the process of the elimination of one variable, say $x_n$, as the same procedure can be repeatedly applied to eliminate the remaining variables. Also, for the sake of simplicity, all entries in the last column of $A$ are, without loss of the generality, assumed to be 0, +1 or -1; otherwise the values of the matrix $A$ corresponding to the variable $x_n$ need to first be equalised or divided by their absolute values in order for $x_n$ to be eliminated. Then, after a possible reordering of the inequalities, $I$ can be rewritten as shown in (3.5).

$$\begin{aligned} \mathbf{a_i'x'} \quad &\leq b_i,\ i = 1, \ldots, m_1 \\ \mathbf{a_j'x'} - x_n &\leq b_j,\ j = m_1 + 1, \ldots, m_2 \\ \mathbf{a_k'x'} + x_n &\leq b_k,\ k = m_2 + 1, \ldots, m \end{aligned} \tag{3.5}$$

where $\mathbf{x'} = \{x_1, x_2, \ldots x_{n-1}\}$, i.e. the same set of variables without $x_n$. Let $I^0 := \mathbf{a_i'x'} \leq b_i$, $I^- := \mathbf{a_j'x'} - x_n \leq b_j$ and $I^+ := \mathbf{a_k'x'} + x_n \leq b_k$, which means that the sets $I^0$, $I^-$ and $I^+$ define all

inequalities in $I$ whose $x_n$ has zero, negative and positive coefficient, respectively. Also, assume that $l = max(\mathbf{a'_j}\mathbf{x'} - b_j, j = m_1+1,\ldots,m_2)$ and $u = min(b_k - \mathbf{a'_k}\mathbf{x'}, k = m_2+1,\ldots,m)$. Since the last two lines of (3.5) are equivalent to $l \leq x_n \leq u$, then the variable $x_n$ can be eliminated. This yields the *reduced* set $R_n$ in (3.6) having no $x_n$ as an equivalent to (3.5):

$$\begin{aligned} \mathbf{a'_i}\mathbf{x'} &\leq b_i, & i &= 1,\ldots,m_1 \\ \mathbf{a'_j}\mathbf{x'} - b_j &\leq b_k - \mathbf{a'_k}\mathbf{x'}, & j &= m_1+1,\ldots,m_2, \\ & & k &= m_2+1,\ldots,m \end{aligned} \tag{3.6}$$

Based on the purpose for which the FME method is applied, the stop point of this method is determined. If the purpose is to solve a set of inequalities, the process of elimination is repeated until the last $n-1$ variables $x_n, x_{n-1},\ldots,x_2$ are eliminated ending up with the set of inequalities $R$ in one variable $x_1$, which is trivial. On the other hand, for the purpose of projection, as is the case for fault diagnosis, the process of elimination stops when all variables starting with $x_n$ and ending with $x_{k+1}$ are eliminated. These eliminated variables correspond to unobservable transitions of Petri nets as will be explained later.

**Theorem 3.1.** [36] Assume that the variables $x_n,\ldots,x_{k+1}$ have been eliminated in order by using the FME method described above from a set of linear inequalities $I$. This results in the reduced set $R$. Then $\alpha_1,\ldots,\alpha_k$ is a solution of $R$ if and only if there exists values $\alpha_{k+1},\ldots,\alpha_n$ such that $\alpha_1,\ldots,\alpha_k,\alpha_{k+1},\ldots,\alpha_n$ is a solution of $I$.

To illustrate the general method, the following example is presented.

**Example 3.3.** Given a set of inequalities $I$ in three variables as shown in (3.7), assume that the

variable $x_3$ is to be eliminated.

$$\begin{array}{rrl} -x_1 & & \leq -1 \\ & -x_2 & \leq -1 \\ & & -x_3 \leq -1 \\ -x_1 & -x_2 & \leq -3 \\ -x_1 & & -x_3 \leq -3 \\ & -x_2 & -x_3 \leq -3 \\ x_1 & +x_2 & +x_3 \leq 6 \end{array} \tag{3.7}$$

Then, the application of the FME method proceeds as follows. We first partition the set $I$ into three subsets of inequalities $I^0$, $I^+$ and $I^-$ shown respectively in (3.8), (3.9) and (3.10).

$$\begin{array}{rrl} -x_1 & & \leq -1 \\ & -x_2 & \leq -1 \\ -x_1 & -x_2 & \leq -3 \end{array} \tag{3.8}$$

$$x_1 + x_2 + x_3 \leq 6 \tag{3.9}$$

$$\begin{array}{rrl} & & -x_3 \leq -1 \\ -x_1 & & -x_3 \leq -3 \\ & -x_2 & -x_3 \leq -3 \end{array} \tag{3.10}$$

Now, all inequalities in the set $I^0$ are transferred to the resulting set $R$. Then, each inequality in $I^-$ is summed to each inequality in $I^+$ in order to eliminate $x_3$. This elimination results in three inequalities, $x_1 \leq 3$, $x_2 \leq 3$ and $x_1 + x_2 \leq 5$. Adding these three inequalities to $R$, we obtain the

following set having no $x_3$:

$$\begin{aligned} -x_1 &\leq -1 \\ -x_2 &\leq -1 \\ -x_1 - x_2 &\leq -3 \\ x_1 + x_2 &\leq 5 \\ x_2 &\leq 3 \\ x_1 &\leq 3 \end{aligned} \tag{3.11}$$

Then, any solution to the set of inequalities in (3.11) has a solution in the original set of inequalities in (3.7) and vice versa.

**Remark 3.1.**

- If the process of partitioning the set of inequalities $I$ results in $I^+ = \emptyset$, then no inequality is added to the resulting set, i.e. all inequalities in $I^-$ are deleted. In fact, choosing a sufficiently large value for the eliminated variable can trivially satisfy $I^-$. Similarly, if $I^- = \emptyset$, all inequalities in $I^+$ are discarded.

- The number of inequalities generated in the worst case grows exponentially with number of variables eliminated. However, there exists a partial remedy for this difficulty since many of these inequalities are redundant in the sense that deleting them will not affect the solution space described by the original set of inequalities. Later in Section 5.2.6, an explanation will be provided about the rule used to define the redundant inequality and how effective this rule will be to remove the redundancy generated by the nature of elimination of the FME method, at least from a practical point of view.

Other elimination methods exist which could perform a similar role to the FME [39, 79, 82]. However, none of these methods are restricted to integer-valued variables. For fault diagnosis in Petri nets, we need integer-valued variables representing the number of firing transitions. Thus, the following section discusses the extension of the FME method to deal with integer-valued variables.

## 3.3 The Integer Fourier-Motzkin Elimination method

In sets of linear inequalities having integer-valued variables, we look for integer solutions only. Such sets may have real solutions when integer solutions do not exist. Difficulties can arise when directly applying the FME method to eliminate integer-valued variables; this will be demonstrated as follows. Suppose that FME is applied to a set of inequalities $I$ resulting in the reduced set of inequalities $R$, if $R$ has no integer solutions, then $I$ has no integer solutions. In some cases, the set of inequalities $R$ may have an integer solution but there does not exist a corresponding integer solution in $I$.

**Example 3.4.** Let us suppose that the variable $x_2$ is to be eliminated using the FME method from the following set of inequalities in two variables: $e_1 := 5x_1 + 3x_2 \leq 19$ and $e_2 := -x_1 - 2x_2 \leq -3$. To eliminate $x_2$, we first multiply the inequality $e_1$ by 2 and $e_2$ by 3 obtaining the inequalities $10x_1 + 6x_2 \leq 38$ and $-3x_1 - 6x_2 \leq -9$. Then, the FME method determines the bounds on the variable $x_2$ as $9 - 3x_1 \leq 6x_2 \leq 38 - 10x_1$ rewritten as $9 - 3x_1 \leq 38 - 10x_1$; which further yields $e_3 := 7x_1 \leq 29$. The value 4 of $x_1$ represents an integer solution of $e_3$. However, there does not exist an integer value for $x_2$ such that $(4, x_2)$ is a solution of the inequalities $e_1$ and $e_2$.

To ensure that, for any integer solution in $R$, there exists an integer solution in $I$, the FME method has been extended. This extension, called the Integer FME (IFME) method, has been created to cope with integer-valued variables and has been reported in [35, 37, 38, 83]. In this thesis, we have chosen the method presented in [38], which better meets our needs as it is somewhat simpler and more efficient.

The IFME method proceeds as follows. Firstly, each inequality $i$ in the set $I := A\mathbf{x} \leq \mathbf{b}$ is normalised by computing the *greatest common divisor* (GCD) $G$ of the coefficients $a_{i1}, \ldots, a_{in}$. Then, all coefficients are divided by $G$ and $b$ is replaced by $\lfloor \frac{b}{G} \rfloor$. Now suppose that the variable $x_n$ constrained by the inequalities $C \leq bx_n$ and $ax_n \leq B$ is to be eliminated. Applying the IFME method to eliminate $x_n$ results in the following inequality:

$$aC - bB \leq (1-a)(1-b) \tag{3.12}$$

Then, for any integer solution to (3.12) there exists an integer solution for $C \leq bx_n$ and $ax_n \leq B$. Note that if $a = 1$ and $b = 1$, then the results of the elimination using the FME and IFME methods are identical, i.e. no need for the extension in this case.

Referring to the set of inequalities in Example 3.4, the bounds on $x_2$ are $3x_2 \leq 19 - 5x_1$ and $3 - x_1 \leq 2x_2$, in which case, $C := 3 - x_1$, $B := 19 - 5x_1$, $a = 3$ and $b = 2$. Applying the general form of (3.12), we obtain the inequality $7x_1 \leq 27$. The value of $x_1 = 3$ is an integer solution to the inequality $7x_1 \leq 27$. In return, there exists an integer solution $(x_1 = 3, x_2 = 1)$ of $e_1$ and $e_2$.

**Remark 3.2.** In *ordinary* Petri nets (where all arcs have weights equal to one), the FME method can be applied without the extension. In effect, having the weights of all arcs in the Petri net equal to one implies that all values of coefficients (the elements of matrix A) of variables in the set $I$ equal one. As a result, we have $a = 1$ and $b = 1$ which when substituted in (3.12), means its right-hand side is evaluated to zero. This yields the same inequality obtained by using the FME method to eliminate the variable $x_n$.

## 3.4 Complex event processing

Complex event processing (CEP) [84–88] is a relatively new technology in which streams of simple events are received, processed and analysed in real-time. From these simple events, complex events are produced using some operations provided by CEP systems. To monitor these incoming simple events and then make decisions or to extract new complex events from them, a CEP system is associated to the target enterprise system using a continuous processing model. Thus, this feature, among others, meets the needs of many different applications such as fault diagnosis, intrusion detection, SLA monitoring, process monitoring and reporting exceptions.

For building CEP applications, two main components are needed: an *event processing language* (EPL) and an *event processing agent* (EPA).

**Event processing language (EPL):** an EPL is an SQL-language with extended facilities and its purpose is to express rules (patterns/queries). These rules represent the main building

blocks in CEP systems by which the information from one or more streams of events is derived and aggregated. Syntactically, each rule consists of two parts: triggers (conditions) and a body having a set of actions. The rules are written to find a pattern of events inside events streams matching the conditions included in a rule.

**Event processing agent (EPA):** an EPA represents an object by which streams of events are monitored in order to detect a pattern of events that is of interest. Every time that there is a match between the defined pattern and a stream of events, the engine responds by executing the actions included in the body part of the rule. This process is performed online and the response is produced in real time.

These CEP systems provide several different types of EPAs. The following briefly describes three basic types of EPAs which have proved their usefulness in building CEP applications:

- *Filters*: using this type of agent, the events of interest in the stream of events are filtered out. To do so, a test is applied on this stream to decide whether to discard or to transfer the events to the following agent for further processing.

- *Pattern detectors*: the pattern detectors examine a collection of arriving events to detect a particular pattern. Then, different operations can be applied to the detected pattern. For example, the engine could discard or pass the pattern or derive a new event from it.

- *Transformers*: the content of incoming events is modified when using the transformer engine. For instance, we could aggregate a stream of incoming events by producing new events which represent functions of incoming events.

There seems to be a strong relationship between the traditional rule-based systems and CEP systems. However, there is a main difference between these two kinds of systems consisting in the way by which the flow of processing is accomplished. In CEP systems, this flow goes in the opposite direction to the traditional rule-based systems, such as SQL systems. In other words, in CEP systems, the rules are stored and the data (events) run through these rules to find a match.

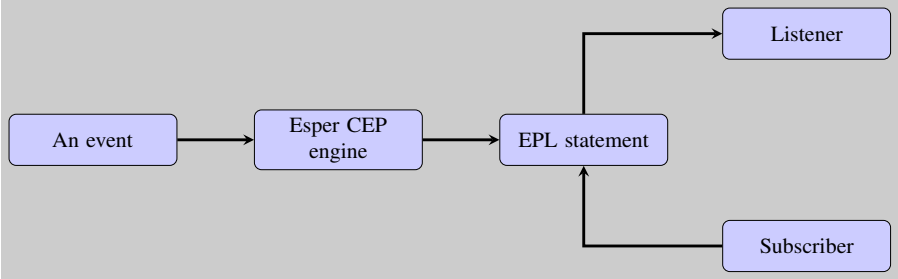



Figure 3.3: Esper CEP building blocks

Conversely, in SQL systems, the data is stored and the queries (rules) are then checked against them to find a match.

### 3.4.1 Esper CEP

*Esper* [89–92] is an open-source tool which is widely used to develop CEP applications under the Java platform. The core of this tool is the Esper engine that presents a continuous processing model. By embedding Esper inside a Java application, the CEP facilities presented by Esper can be accessed for processing events. Esper provides a very rich EPL, which is an extension to SQL, and it is used to write EPL statements and patterns in order to support different types of operations on events such as filtering, aggregations and joining.

To create an application using Esper, three building blocks are required: an event class, EPL statements/pattern and listeners/subscribers. These building blocks and the interactions between them are depicted in Figure. 3.3. The common way to represent events in Esper is through Plain Old Java Object (POJO) classes, but maps and XML representations are also supported. As it is known, the objects of the type POJO have no restrictions other than these which are compatible with specifications of Java. This means that POJO objects do not extend a superclass or implement an interface. In addition, Esper defines two types of statements to match events: patterns and EPL queries. Patterns begin with the keyword `every` while EPL queries begin with the keyword `select` or `insert`. To create and register these statements in the Esper engine, an instance of class `EPAdministrator` is initiated.

Regarding the mechanism of work, two modes are supported by Esper: push and pull modes.

In push mode, the *listener(s)* associated to an EPL statement is notified every time there is a match between the incoming events and the statement. Also, we can associate more than one listener to one statement; in which case, all listeners associated to the statement are notified when a match occurs. On the other hand, the Esper engine is queried at any time and not necessarily at the time of the events arrival. This mode is implemented by *subscribers* and better suited for periodic pulling of information from the engine, see Figure. 3.3. The following code represents a simple example using the push mode as described in [89]; the unnecessary details have been removed:

```
1  Package Esper.cep;
2  import com.espertech.esper.client.*;
   ...
3  public class PrintListener implements UpdateListener{
4    public void update (EventBean[] newEvent, Eventbean[] oldEvent){
5      System.out.println(Arrays.toString(newEvent));
6    }
7  }
8  public class Deposit{
9    private float Amount;
10   private string customerID;
11   public void setCustomerID (String customerID){...}
12   public void setAmount(float Amount) {...}
13 }
14 public class Esper_CEP_test;
15   public static void main(string[]) args){
16     EPServiceProvider epa=EPServiceProviderManager.getDefaultProvider().
17     EPAdministrator admin=epa.getEPAdministrator();
18     EPStatement pattern=admin.createPattern("every␣A=Esper.cep.Deposit");
19     PrintListener listener=new PrintListener();
20     pattern.addListener(listener);
21     Deposit d=new Deposit();
22     d.setAmount(100);
```

```
23      d.setCustomerID("222");
24      EPRuntime runtime=epa.getEPRuntime();
25      runtime.sendEvent(d);
26    }
27 }
```

In this example, the POJO class representation is adopted by declaring the class `Deposit`, in addition an instance `epa` of the Esper engine is created in step 16. As is seen, the pattern `pattern` in step 18 is defined as `every A=Esper.cep.Deposit` which means that every event named `A` of type `Deposit` included in the package `Esper.cep`. The defined pattern is associated to the listener `listener` in step 20. Then, every created event is sent to the engine via the step 25. Note that the definition of patterns in EPL represents an extension to SQL which has no such statements. Also, we can create an EPL query, which behaves similarly to the `pattern`, as:

```
Select * from Esper.cep.Deposit
```

# CHAPTER 4

# FAULT DIAGNOSIS IN ACYCLIC PETRI NETS

In this chapter, we address the problem of fault diagnosis in partially-observed discrete event systems modelled by Petri nets. Following the assumption that the nets are *acyclic*, the IFME method is used to solve the problem. This class of Petri nets is used to model some manufacturing, assembly and disassembly processes [93]; in addition, they are used to represent some workflow procedures which are of particular interest in business processes [94]. The advantage of using acyclic Petri nets for analysis of DES arises from fact that the state equation in such nets is necessary and sufficient condition for the reachability of markings, i.e. for any solution of the state equation, there exists a sequence in the language of the Petri net.

The chapter begins with a formal description of the problem followed by presentation of a new formalism for representing faults. In this new formalism, the faults can be expressed as inequalities. The definition of the diagnoser is extended and described according to the formalism. An outline of the proposed approach is also presented, supported by a mathematical proof of correctness for the proposed solution. In addition, algorithms for both creating the diagnoser offline and diagnosing faults online are described. We end this chapter by giving an example to illustrate the proposed approach.

## 4.1 Description of the problem

In this section, a description of the problem of fault diagnosis in DES modelled by Petri nets is given, as outlined in [53] and [8]. Consider a Petri net $(\mathcal{N}, M_0)$ with a set of transitions $T = \{t_1, t_2, \ldots, t_n\}$ and suppose that $T$ is partitioned into two sets: observable transitions $T_o$ and unobservable transitions $T_u$. We further assume that faults are unobservable events, i.e. assuming $T_f$ is the set of transitions modelling occurrences of faults, then $T_f \subseteq T_u$. The set $T_u$ may also have other transitions which model non- fault events (normal events). In this chapter, the system is assumed to have a single fault.

In Petri nets which model partially-observed DES, each observable transition is associated with an event (given as a label). We assume that, if a transition fires, the associated event is observed. In other words, in every execution of events, a sequence of transitions from $T_o$ can only be observed. A diagnoser (defined below) uses such information to identify whether the fault has definitely occurred or only may have occurred.

Also, consider the *projection function* $\pi : T \to T_o \cup \{\varepsilon\}$ that maps unobservable transitions to the empty string $\varepsilon$, i.e. $\pi(t) = \varepsilon$ for $t \in T_u$ while, $\pi(t) = t$ for $t \in T_o$. The projection function $\pi$ can be extended to the Kleene closure of $T$ by $\pi : T^* \to (T_o \cup \{\varepsilon\})^*$, where for each sequence of transitions $\sigma$ and each transition $t$, $\pi(\sigma t) = \pi(\sigma)\pi(t)$. We assume $\pi(\varepsilon) = \varepsilon$ and that $\pi(t\varepsilon) = \pi(\varepsilon t) = \varepsilon$ for each $t \in T_u$.

Denote by $\mathbf{s} = \pi(\sigma)$ the observed sequence corresponding to a given firing sequence $\sigma \in T^*$. Now, based on the definition of the *valuation* described in [95], the following definition is presented.

**Definition 4.1.** *The valuation function*: let $\mathbf{x} = (x_1, \ldots, x_n)$ be a set of variables. We suppose that the variables range over $\mathbb{N}$. A *valuation* $v$ for $\mathbf{x}$ is a function that associates a value in $\mathbb{N}$ to each variable $x_i$ in $\mathbf{x}$.

**Remark 4.1.** In the light of Definition 4.1, given a sequence $\sigma \in T^*$, then Parikh vector $\#(\sigma)$ represents a valuation of $\mathbf{x}$. In other words, for each $x_i$ of $\mathbf{x}$, $x_i = \#(t_i, \sigma)$, where $i = 1, 2, \ldots, n$.

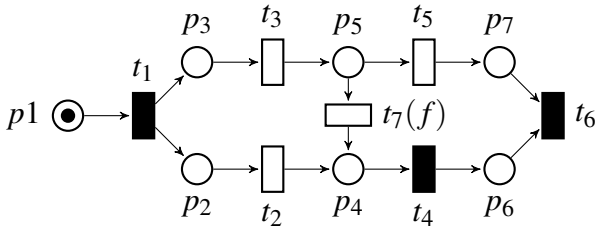



Figure 4.1: Example of an *acyclic* Petri net

**Definition 4.2.** *The logical operator* $\vDash$ *on an inequality*: suppose that **e** is an inequality of the form $a_1x_1 + \cdots + a_nx_n \leq b$ in the variable set $\mathbf{x} = (x_1, \ldots, x_n)$, $x_i \in \mathbb{N}$ and $a_1, \ldots, a_n, b \in \mathbb{Z}$. Consider a valuation $\nu$ as $\alpha_1, \ldots, \alpha_n$ assigned to values $x_1, \ldots, x_n$, respectively; then we write $\nu \vDash \mathbf{e}$ to say that the valuation $\nu$ satisfies the inequality **e** if and only if $a_1\alpha_1 + \cdots + a_n\alpha_n \leq b$ holds.

**Definition 4.3.** *The logical operator* $\vDash$ *on a set of inequalities*: suppose that we have a set of inequalities $I = \{e_i \mid 1 \leq i \leq d\}$, where each $e_i$ has the form of **e** in Definition 4.2. Consider a valuation $\nu$ for the variables of the inequalities in $I$, then $\nu \vDash I$ holds if and only if $(\nu \vDash e_1) \wedge (\nu \vDash e_2) \wedge \cdots \wedge (\nu \vDash e_d)$ holds.

**Lemma 4.1.** Given a Petri net $(\mathcal{N}, M_0)$, we can derive a corresponding set of inequalities $I$ in the form $-A\mathbf{x} \leq M_0$ (extracted from (3.1)), equipped with non-negativity constraints on **x**, i.e. $\mathbf{x} \geq \vec{\mathbf{0}}$. If $\mathcal{N}$ is *acyclic*, then a marking $M$ is reachable from $M_0$, i.e. $M_0 \xrightarrow{\sigma} M$ if and only if there exists **x** satisfying $I$ and $\mathbf{x} = \#(\sigma)$.

*Proof.* Since the state equation $M = M_0 + A\mathbf{x}$ is such that $M \geq \vec{\mathbf{0}}$, then we can obtain $M_0 + A\mathbf{x} \geq \vec{\mathbf{0}}$. This latter formula can be easily written as $I := -A\mathbf{x} \leq M_0$. Then, any solution to $I$ is a solution to $M = M_0 + A\mathbf{x}$ subject to $M \geq \vec{\mathbf{0}}$ and vice versa. Afterwards, the result holds following the proof of Theorem 16 in [33]. □

**Example 4.1.** Consider the Petri net depicted in Figure. 4.1. Assume that the initial marking of this net is $M_0 = [1000000]$. This Petri net models the process of sending messages in a communication system, where a token in place $p_1$ at the initial marking represents a message ready to be sent. Firing transition $t_1$ expresses dividing the message into two packets. These

packets are separately sent on two channels causing that one token is put in both places $p_2$ and $p_3$. Finally, the two packets are combined (transition $t_6$) at the receiver side. In this Petri net, the fault occurs (transition $t_7$) when the first packet is erroneously moved to the second channel. Then, the set of inequalities derived from the state equation in the net is shown in (4.1).

$$\begin{array}{rcccccccl}
x_1 & & & & & & & & \leq 1 \\
-x_1 & + & x_2 & & & & & & \leq 0 \\
-x_1 & & & + & x_3 & & & & \leq 0 \\
 & - & x_2 & & & + & x_4 & - & x_7 \leq 0 \\
 & & & - & x_3 & & & + & x_7 \leq 0 \\
 & & & & & - & x_4 & & \leq 0 \\
 & & & & & & & - & x_i \leq 0
\end{array} \tag{4.1}$$

where the constraints of the form $-x_i \leq 0,\ i = 1,\ldots,7$, represent non-negativity constraints on the variables in the set.

In this chapter, the fault diagnosis problem in Petri nets is addressed under the following assumptions:

a) The systems under consideration are diagnosable (any fault can be diagnosed in a finite delay).

b) The system to be diagnosed starts from a non-faulty state.

c) Faults are permanent, i.e. when a fault occurs in a state, it propagates to all following states.

d) The structure of a given Petri net $\mathcal{N}$ and its initial marking $M_0$ are known.

e) There is no cycle of transitions in Petri nets under consideration.

f) Every transition has a unique label.

g) A single fault transition is considered.

The assumption (a) is required to ensure reaching to a certain diagnosis state in a finite delay after fault occurrences. Assumptions (b)-(d) are commonly adopted in the literature on fault diagnosis. The additional assumption (e) is to ensure that any solution to the state equation represents a sequence in the language of a given Petri net. The last two assumptions are to simplify the presentation of the proposed approach.

## 4.2 Representation of faults as inequalities

In this thesis, inequalities are generally used in two ways. Firstly, the state equation constraints can be written as a set of inequalities $I$. Secondly, faults can also be written as inequalities. Suppose that a transition $t_i \in T$ is a fault transition; then occurrences of $t_i$ in a given firing sequence $\sigma$ can trivially be written as:

$$\#(t_i, \sigma) > 0 \tag{4.2}$$

Conversely, the case where $t_i$ does not appear in $\sigma$ can be expressed as:

$$\#(t_i, \sigma) \leq 0 \tag{4.3}$$

if we consider the case where there is no $t_i$ in $\sigma$ corresponds to a satisfaction of a constraint. Likewise, we can say that the appearance of $t_i$ in $\sigma$ corresponds to a violation of the constraint. Let us denote the constraint by $\mathbf{c}$ and its violation by $\neg\mathbf{c}$, i.e. $\mathbf{c}$ and $\neg\mathbf{c}$ represent the inequalities in (4.3) and (4.2), respectively.

In the following paragraphs, the definition of the diagnoser is presented. This definition is inspired by the previous work of [4] and [8].

**Definition 4.4.** A diagnoser is a function $\Delta : T_o^* \rightarrow \{NoFault, Faulty, Uncertain\}$ that associates with each observed sequence $\mathbf{s}$, with respect to the fault, one of the following diagnosis states:

- $\Delta(\mathbf{s}) = NoFault$ if $\forall \sigma \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma) = \mathbf{s}$, $\#(\sigma) \vDash \mathbf{c}$ holds. This state

represents the non-faulty state as there is no firing sequence having the same observation $\mathbf{s}$ containing the fault transition, i.e. no fault has occurred.

- $\Delta(\mathbf{s}) = Faulty$ if $\forall \sigma \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma) = \mathbf{s}$, $\#(\sigma) \models \neg\mathbf{c}$ holds. This state implies that the behaviour of the system is faulty as all firing sequences having the same observation $\mathbf{s}$ contain the fault transition, i.e. the fault has certainly occurred during the observed sequence $\mathbf{s}$.

- $\Delta(\mathbf{s}) = Uncertain$ if there exists two sequences $\sigma_1$, $\sigma_2 \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma_1) = \pi(\sigma_2) = \mathbf{s}$, and $\#(\sigma_1) \models \mathbf{c}$ and $\#(\sigma_2) \models \neg\mathbf{c}$ hold. In this case, the behaviour of the system is ambiguous because both *NoFault* and *Faulty* states are possible during the observed sequence. For this reason, this state is called an $Uncertain$ state.

**Example 4.2.** To explain the fault diagnosis notions in the Petri nets mentioned previously, let us recall the Petri net of Example 4.1 illustrated in Figure. 4.1. Note that observable transitions are shown by solid rectangles, while empty rectangles represent unobservable transitions. Also, in this Petri net, there is only one fault, modelled by transition $t_7$. Thus, the constraint $\mathbf{c}$ can be written as $x_7 \leq 0$ and its negation $\neg\mathbf{c}$ as $x_7 > 0$.

If we assume that no firing of any transition is observed at the initial marking $M_0$, we are then certain that no fault has occurred as there is not an unobservable transition enabled at the initial marking. Let us also assume that the sequence $\mathbf{s} = t_1$ is observed at the initial marking. By observing this sequence, we are not certain about the diagnosis state as there are at least two possible sequences; for example, $\sigma_1 = t_1t_2$ and $\sigma_2 = t_1t_3t_7$ such that $\pi(\sigma_1) = \pi(\sigma_2) = t_1$; in addition, $\#(\sigma_1) \models \mathbf{c}$ and $\#(\sigma_2) \models \neg\mathbf{c}$. Hence, we say that the diagnosis state is $Uncertain$. Likewise, we have the same diagnosis state when the sequence $t_1t_4$ is observed. In this case, we also have at least two sequences; for instance $\sigma_1 = t_1t_2t_4$ and $\sigma_2 = t_1t_3t_7t_4$, with $\pi(\sigma_1) = \pi(\sigma_2) = t_1t_4$ such that $\#(\sigma_1) \models \mathbf{c}$ but $\#(\sigma_2) \models \neg\mathbf{c}$. In both cases, the fault may have occurred, but also the diagnosis state could be $NoFault$. Thus $\Delta(\mathbf{s}) = Uncertain$.

If the sequence $\mathbf{s} = t_1t_4t_4$ is observed, for any $\sigma \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma) = \mathbf{s}$, $\sigma$ has the transition $t_7$. As a result, we are certain that $\mathbf{c}$ is violated and $\Delta(\mathbf{s}) = Faulty$. By contrast,

observing $\mathbf{s} = t_1 t_4 t_6$ excludes the possibility of firing any sequence having the transition $t_7$ as $t_6$, if fired, requires at least one token in both places $p_6$ and $p_7$ and this is impossible in a case where $t_7$ fires. Hence, $\#(\mathbf{s}) \models \mathbf{c}$ for all $\sigma$ such that $\sigma \in L(\mathcal{N}, M_0)$ and $\pi(\sigma) = \mathbf{s}$, i.e. the fault has not occurred during observing the sequence and $\Delta(\mathbf{s}) = NoFault$.

## 4.3 The IFME method for fault diagnosis

### 4.3.1 Description of the method

In this section, the main results of this chapter are presented. If we suppose that $\mathcal{N}$ is an *acyclic* Petri net, without any loss of generality, we can rename the transitions of $\mathcal{N}$ such that the first $k$ transitions are observable, i.e. $T_o = \{t_1, t_2, \ldots, t_k\}$. The remaining transitions are unobservable, i.e. $T_u = \{t_{k+1}, t_{k+2}, \ldots, t_n\}$. We further assume that the system has a single fault and $t_n$ is the only fault transition of the system. We introduce the vector $\mathbf{x} = (x_1, \ldots, x_n)$ in which each $x_i$ represents the number of firings of $t_i$ for all $i = 1, \ldots, n$, as described in Section 4.1. Suppose that $I := -A\mathbf{x} \leq M_0$, where $\mathbf{x} \geq \vec{\mathbf{0}}$, represents the state equation constraints. We further assume that $\mathbf{c}$ is the inequality $x_n \leq 0$ and $\neg\mathbf{c}$ is the negation of $\mathbf{c}$, i.e. the inequality $x_n > 0$. For each sequence $\sigma$ of $(\mathcal{N}, M_0)$, if $\sigma$ contains $t_n$ (the fault transition), then $\#(\sigma)$ (the Parikh vector of $\sigma$) satisfies $\neg\mathbf{c}$. Conversely, if $\#(\sigma)$ satisfies $\mathbf{c}$, then $\sigma$ has no fault transition $t_n$.

Figure. 4.2 depicts a general sketch of the proposed method. Assuming that we start with an *acyclic* Petri net model, then a two-step process is carried out:

- **Offline step:** we first obtain a set of inequalities $I$ created from the state equation and non-negativity constraints on $\mathbf{x}$. Then, two sets of inequalities $I \cup \{\mathbf{c}\}$ and $I \cup \{\neg\mathbf{c}\}$ are created. Applying the IFME method simultaneously to both $I \cup \{\mathbf{c}\}$ and $I \cup \{\neg\mathbf{c}\}$ respectively results in two reduced sets $R$ and $R'$, obtained by eliminating every variable corresponding to a transition in the set $T_u$.

- **Online step:** in this step, the reduced sets of inequalities $R$ and $R'$ are used to make

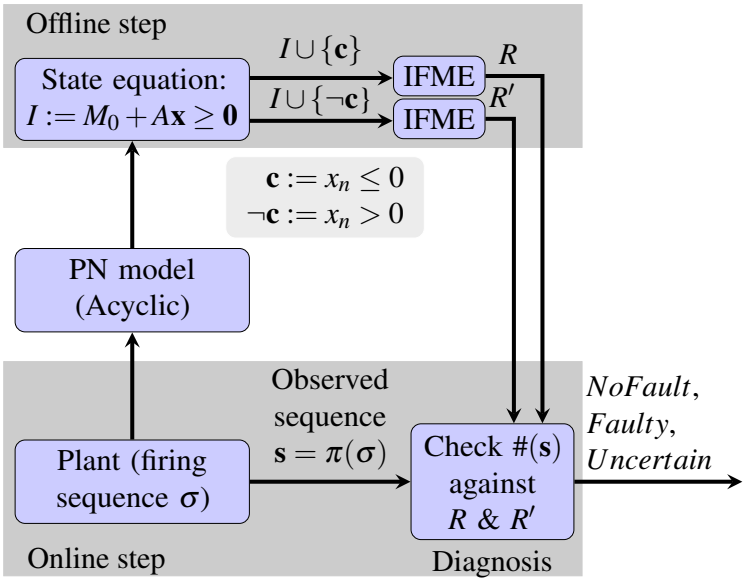



Figure 4.2: Sketch of the proposed approach in the case of *acyclic* Petri nets

diagnosis decisions. More precisely, the Parikh vector of a given observed sequence $\mathbf{s}$ is tested against $R$ and $R'$ to determine whether it satisfies any or both of $R$ and $R'$.

The following theorem gives full details of this process and its validation for fault diagnosis.

**Theorem 4.1.** Suppose that $(\mathcal{N}, M_0)$ is a Petri net satisfying the assumptions (a)-(g) listed in Section 4.1. Also, suppose that $I$ is the set of inequalities $-A\mathbf{x} \leq M_0$ created from the state equation of $\mathcal{N}$ plus $\mathbf{x} \geq \vec{\mathbf{0}}$, see Lemma 4.1. Assume that $T = T_o \cup T_u$, $T_o = \{t_1, \ldots, t_k\}$, $T_u = \{t_{k+1}, \ldots, t_n\}$ and $t_n$ is a fault transition. The vector of variables $x_1, \ldots, x_n$ corresponds to the number of firing the transitions $t_1, \ldots, t_n$. Assume also that $\mathbf{c}$ is the inequality $x_n \leq 0$ and $\neg\mathbf{c}$ is its negation. Suppose that the set of inequalities $R$ and $R'$ are respectively produced from applying IFME to both $I \cup \{\mathbf{c}\}$ and $I \cup \{\neg\mathbf{c}\}$ to eliminate all variables corresponding to transitions in $T_u$. Then, for any given observed sequence of events $\mathbf{s} = \pi(\sigma)$, where $\sigma$ is a firing sequence in $\mathcal{N}$ $(M_0 \xrightarrow{\sigma} M)$, $\Delta(\mathbf{s})$ is determined as follows:

$$\Delta(\mathbf{s}) = \begin{cases} \textit{NoFault} & \text{if } \#(\mathbf{s}) \not\models R' \\ \textit{Faulty} & \text{if } \#(\mathbf{s}) \not\models R \\ \textit{Uncertain} & \text{if } \#(\mathbf{s}) \models R \wedge \#(\mathbf{s}) \models R' \\ \textit{Impossible} & \text{if } \#(\mathbf{s}) \not\models R \wedge \#(\mathbf{s}) \not\models R' \end{cases}$$

*Proof.* In the following, we assume that $\#(\mathbf{s}) = (\alpha_1, \ldots, \alpha_k)$.

**Proof of** $\Delta(\mathbf{s}) = NoFault$**:** assume that $\#(\mathbf{s}) \not\models R'$, but the diagnosis state is not $NoFault$. If $\#(\mathbf{s}) \not\models R'$, then for every valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$, $\nu \not\models I \cup \{\neg\mathbf{c}\}$ by Theorem 3.1. Since $\sigma$ is a firing sequence, we are certain that $\nu \models I$, see Lemma 4.1. Hence, $\nu \not\models \neg\mathbf{c}$, i.e. $\nu \models \mathbf{c}$ . As a result, $\forall\sigma' \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma') = \mathbf{s}$, $\#(\sigma') \models \mathbf{c}$, i.e. $\#(t_n, \sigma') \leq 0$. Thus, the fault has not occurred during observing $\mathbf{s}$. This contradicts the assumption.

**Proof of** $\Delta(\mathbf{s}) = Faulty$**:** using the same argument in **Proof of** $\Delta(\mathbf{s}) = NoFault$ replacing $R'$ with $R$.

**Proof of** $\Delta(\mathbf{s}) = Uncertain$**:** assume that $\#(\mathbf{s}) \models R$ and $\#(\mathbf{s}) \models R'$, but we are certain about the diagnosis state. If $\#(\mathbf{s}) \models R$, then there exists a valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$ and $\nu \models I \cup \{\mathbf{c}\}$ by Theorem 3.1. If $\nu \models I \cup \{\mathbf{c}\}$, then $\nu \models I$. Considering that $\mathcal{N}$ is *acyclic*, then there exists $\sigma'$ such that $M_0 \xrightarrow{\sigma'} M'$, $\#(\sigma') = \nu$. Since $\nu \models I \cup \{\mathbf{c}\}$, then $\nu \models \mathbf{c}$ which means $\#(t_n, \sigma') \leq 0$, i.e. $\sigma'$ contains no fault. Now, we claim that $\pi(\sigma') = \mathbf{s}$. The proof of this claim is obtained by induction on the length of the observed sequence denoted $|\mathbf{s}|$. We start the proof with the case of $|\mathbf{s}| = 1$ as the case $|\mathbf{s}| = 0$ is already proved.

*Base case*: if $|\mathbf{s}| = 1$, then $\pi(\sigma') = \mathbf{s}$ because $\#(\pi(\sigma')) = \#(\mathbf{s})$. In fact, if $\pi(\sigma') \neq \mathbf{s}$, then there exists an entry in both $\#(\mathbf{s})$ and $\pi(\#(\sigma'))$ having different values and this contrasts $\#(\pi(\sigma')) = \#(\mathbf{s})$.

*Induction step*: we assume that the claim is true for all $\mathbf{s}$ with $|\mathbf{s}| < k_1$ (*Induction hypothesis*). Then, we prove it is true for $\mathbf{s}$ with $|\mathbf{s}| = k_1$. Suppose $\mathbf{s} = \mathbf{s}'t$ where $t \in T_o$ and $\mathbf{s}' \in T_o^*$. Since $\sigma, \sigma' \in L(\mathcal{N}, M_0)$ and $\#(\pi(\sigma)) = \#(\pi(\sigma')) = \#(\mathbf{s})$, then there are sequences $\sigma_1' \in T^*$ and $\sigma_2' \in T_u^*$ such that $\sigma' = \sigma_1' t' \sigma_2'$. In effect, $t'$ is the most recent observable transition in $\sigma'$. Then we have:

$M_0 \xrightarrow{\sigma_1'} M_1' \xrightarrow{t'} M_2' \xrightarrow{\sigma_2'} M'$, $t' \in T_o$

also $\sigma_2'$ can be empty. For $\sigma = \sigma_1 t \sigma_2$ we have:

$M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t} M_2 \xrightarrow{\sigma_2} M$, $\sigma_1 \in T^*, \sigma_2 \in T_u^*$

As $\pi(\sigma) = \mathbf{s} = \mathbf{s}'t$ and $t$ is the last observable transition in $\sigma$, then $\pi(\sigma_1) = \mathbf{s}'$. By the induction hypothesis, $\pi(\sigma'_1) = \mathbf{s}'$. Since $\#(\pi(\sigma')) = \#(\mathbf{s}) = \#(\mathbf{s}'t)$ and $\pi(\sigma_1) = \pi(\sigma'_1)$, then $t = t'$ (if $t \neq t'$ then $\#(\pi(\sigma')) \neq \#(\mathbf{s})$ and this is not true). As a result, $\pi(\sigma') = \pi(\sigma'_1)t' = \mathbf{s}'t = \mathbf{s}$ and this proves the claim.

Similarly, we can prove that, if $\#(\mathbf{s}) \vDash R'$, there exists a sequence $\sigma''$ such that $M_0 \xrightarrow{\sigma''} M''$, $\#(\sigma'') \vDash \neg\mathbf{c}$ ($\#(t_n, \sigma'') > 0$) and $\pi(\sigma'') = \mathbf{s}$.

To conclude, since $\sigma', \sigma'' \in L(\mathcal{N}, M_0)$ with $\pi(\sigma') = \pi(\sigma'') = \mathbf{s}$, $\#(\sigma') \vDash \mathbf{c}$ and $\#(\sigma'') \vDash \neg\mathbf{c}$, hence we have an *Uncertain* state, see Definition 4.4. This contradicts the assumption.

**Proof of the case** *Impossible*: assume that $\#(\mathbf{s}) \nvDash R$ and $\#(\mathbf{s}) \nvDash R'$, but this case is possible. If $\#(\mathbf{s}) \nvDash R$, then for every valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$, $\nu \nvDash I \cup \{\mathbf{c}\}$ by Theorem 3.1. Also, if $\#(\mathbf{s}) \nvDash R'$, then for every valuation $(\beta_{k+1}, \ldots, \beta_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \beta_{k+1}, \ldots, \beta_n)$, $\nu \nvDash I \cup \{\neg\mathbf{c}\}$ by Theorem 3.1. Rephrasing this statement, we can say that there exists a valuation $(\beta_{k+1}, \ldots, \beta_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \beta_{k+1}, \ldots, \beta_n)$ and $\nu \vDash I \cup \{\mathbf{c}\}$ taking into account that $\neg\mathbf{c}$ is the negation of $\mathbf{c}$ and $\sigma$ is a firing sequence of $\mathcal{N}$, i.e. $\nu = \#(\sigma) \vDash I$. Here we have contradictory statements. Hence, this case is an impossible case. This contradicts the assumption and completes the proof. □

Put simply, the above theorem states that, given an observed sequence $\mathbf{s}$, the satisfaction of the Parikh vector of the sequence is checked against both sets $R$ and $R'$. Then, diagnosis states are estimated according to the outcomes. In particular, if the observed sequence does not satisfy $R$, then the diagnosis state is *Faulty*. By contrast, if the observed sequence does not satisfy $R'$, then the diagnosis state is *NoFault*. Otherwise, the diagnosis state is *Uncertain*. Note that the case where the observed sequence does not satisfy both $R$ and $R'$ is not possible. Thus, Theorem 4.1 provides a systematic procedure to detect the firing of the fault transition.

**Remark 4.2.** Note that the proofs of the cases *Faulty* and *NoFault* in Theorem 4.1 are still valid for Petri nets which are not *acyclic*. In effect, diagnosis of these cases in our approach is similar to using the state equation to check *unreachability* of a given marking $M$ in general Petri nets. In other words, we know that if the state equation admits no solution for a given marking $M$,

**Algorithm 4.1** : build the diagnoser.

**Input:** A Petri net $(\mathcal{N}, M_0)$,
a set of unobservable transitions $T_u$, the fault transition $t_n$.
**Output:** A pair $(R, R')$.

1: **Let** $I \leftarrow \{-A\mathbf{x} \leq M_0\} \cup \{-x_i \leq 0 \mid_{i=1,\ldots,n}\}$
2: **Let** $\mathbf{c} \leftarrow x_n \leq 0$
3: **Let** $\neg\mathbf{c} \leftarrow -x_n \leq -1$
4: $R \leftarrow I \cup \{\mathbf{c}\}$
5: $R' \leftarrow I \cup \{\neg\mathbf{c}\}$
6: **for all** $t_j$ such that $t_j \in T_u$ **do**
7: $\quad R \leftarrow IFME\_method(R, x_j)$
8: $\quad R' \leftarrow IFME\_method(R', x_j)$
9: **end for**

then $M$ is not reachable from the initial marking, i.e. there does not exist a sequence $\sigma$ such that $M_0 \xrightarrow{\sigma} M$. Similarly, if the Parikh vector $\#(\mathbf{s})$ of a given observed sequence $\mathbf{s}$ is not a solution to $R$ ($R'$), then $\#(\mathbf{s})$ has no corresponding solution in $I \cup \{\mathbf{c}\}$ ($I \cup \{\neg\mathbf{c}\}$) regardless of the structure of the Petri nets.

### 4.3.2 Fault diagnosis algorithms

The steps for producing the pair of the sets of inequalities $R$ and $R'$, in addition to using these sets online in order to diagnose faults, are described by algorithms below. These algorithms are developed based on the results obtained in Theorem 4.1.

Algorithm 4.1 takes three parameters: a Petri net $(\mathcal{N}, M_0)$, a set of unobservable transitions $T_u$ and the fault transition $t_n$. Eventually, it outputs the pair of the sets of inequalities $R$ and $R'$ which will be used later for computing diagnosis states online. To explore the details of this algorithm: step 1 initialises the set of inequalities $I$; steps 2-3 create a pair of inequalities $(\mathbf{c}, \neg\mathbf{c})$ as described in Section 4.2. Then, augmented sets of inequalities $I \cup \{\mathbf{c}\}$ and $I \cup \{\neg\mathbf{c}\}$ are produced, and a pair of inequalities $(R, R')$ is initialised in steps 4-5 accordingly.

The sets of inequalities $(R, R')$ are updated by steps 6-9. The IFME method (*IFME_method* function) is recursively applied $|T_u|$ times to eliminate all variables $x_j$, $\forall t_j \in T_u$. The resulting sets of inequalities $R$ and $R'$ can be imagined as a diagnoser in the new context.

**Algorithm 4.2** : online fault diagnosis.

**Input:** A pair $(R, R')$ as defined in Algorithm 4.1.
**Output:** The diagnosis state $\{NoFault, Faulty, Uncertain\}$.

1: **Let** $\mathbf{s} \leftarrow \varepsilon$
2: **loop**
3: **if** a new event $e$ is observed **then**
4: **Let** $\mathbf{s}' \leftarrow \mathbf{s}$, $\mathbf{s} \leftarrow \mathbf{s}'e$
5: **if** $\#(\mathbf{s}) \not\models R'$ **then**
6: $\Delta(\mathbf{s}) \leftarrow NoFault$
7: **else if** $\#(\mathbf{s}) \not\models R$ **then**
8: $\Delta(\mathbf{s}) \leftarrow Faulty$
9: **else if** $\#(\mathbf{s}) \models R$ **and** $\#(\mathbf{s}) \models R'$ **then**
10: $\Delta(\mathbf{s}) \leftarrow Uncertain$
11: **end if**
12: **end if**
13: **end loop**

On the other hand, Algorithm 4.2 is used online to diagnose faults. Its inputs are the two sets of inequalities $R$ and $R'$ created by Algorithm 4.1. The output of the algorithm is a diagnosis state from $\{NoFault, Faulty, Uncertain\}$ (see Definition 4.4). This algorithm begins by initialising the observed sequence $\mathbf{s}$ by the empty string $\varepsilon$. Then, in step 2 in particular, it enters into a loop to monitor the system state to check whether the fault has occurred. In step 3, the algorithm waits until a new event $e$ is observed and then concatenates it to the previous sequence of observed events $\mathbf{s}'$, updating the sequence $\mathbf{s}$. The Parikh vector of the observed sequence $\#(\mathbf{s})$ is tested against both sets of inequalities $R'$ and $R$ in steps 5, 7 and 9. The aim of this testing is to decide whether the fault has occurred, or may have occurred. Then, the diagnosis state is determined in steps 6, 8 and 10 as described in Theorem 4.1.

### 4.3.3 Illustrative example

Referring to the Petri net of Figure. 4.1, where the transition fault is $t_7$ and its associated set of inequalities is as described in (4.1), assume that we have augmented this set once by adding the constraint $\mathbf{c} := x_7 \leq 0$ and another by adding the negation of the constraint $\neg\mathbf{c} := -x_7 \leq -1$. Note that the latter inequality is rewritten in the standard form of the set of inequalities defined

Table 4.1: The sets $R$ and $R'$ resulting from the IFME method in the illustrative example

| No. | $R$ | $R'$ |
|---|---|---|
| 1 | $x_1 \leq 1$ | $x_1 \leq 1$ |
| 2 | $-x_4 + x_6 \leq 0$ | $-x_4 + x_6 \leq 0$ |
| 3 | $-2x_1 + x_4 + x_6 \leq 0$ | $-2x_1 + x_4 + x_6 \leq 0$ |
| 4 | $-2x_1 + x_4 \leq 0$ | $-2x_1 + x_4 \leq 0$ |
| 5 | $-x_1 + x_4 \leq 0$ | $-x_1 + x_6 \leq -1$ |
| 6 | $-x_1 + x_6 \leq 0$ | $-x_1 \leq -1$ |

in Lemma 4.1, and also the non-negative constraint $x_7 \geq 0$ is previously removed from $I$ as it is redundant. Then, applying the IFME method to each augmented set results in the two reduced sets $R$ and $R'$ described in Table 4.1. Note that all variables corresponding to unobservable transitions $T_u = \{t_2, t_3, t_5, t_7\}$ have been eliminated in both sets. The sets of inequalities $R$ and $R'$ are in variables representing observable transitions $T_o = \{t_1, t_4, t_6\}$.

Considering these two sets, let the observed sequence $\mathbf{s}$ be $\varepsilon$, then $\#(t_1, \mathbf{s}) = 0$, $\#(t_4, \mathbf{s}) = 0$, $\#(t_6, \mathbf{s}) = 0$ as no firing of any transition from the set $T_o = \{t_1, t_4, t_6\}$ has been observed. By looking at $R$ and $R'$, we find that the latter is not satisfied. In this case, we are certain that no fault has occurred. Likewise, when $\mathbf{s} = t_1 t_4 t_6$, we have $\#(t_1, \mathbf{s}) = 1$, $\#(t_4, \mathbf{s}) = 1$, $\#(t_6, \mathbf{s}) = 1$. Substituting these values of variables into $R$ and $R'$ establishes that $R'$ is not satisfied. Thus we conclude a similar diagnosis state, i.e. $\Delta(\mathbf{s}) = NoFault$.

Now, assume that $\mathbf{s} = t_1 t_4 t_4$, then $\#(t_1, \mathbf{s}) = 1$, $\#(t_4, \mathbf{s}) = 2$, $\#(t_6, \mathbf{s}) = 0$. In such a case, $R$ is not satisfied which implies that the fault has occurred, i.e. we have $\Delta(\mathbf{s}) = Faulty$. Finally, let $\mathbf{s} = t_1 t_4$, this yields $\#(t_1, \mathbf{s}) = 1$, $\#(t_4, \mathbf{s}) = 1$, $\#(t_6, \mathbf{s}) = 0$. Verifying these values against $R$ and $R'$, we find that both of them are satisfied. Thus, we infer that the fault may have occurred, i.e. $\Delta(\mathbf{s}) = Uncertain$.

## 4.4 Chapter summary

This chapter introduced a new approach for fault diagnosis in discrete event systems modelled by *acyclic* Petri nets. The systems under study are partially-observed where faults are modelled as unobservable transitions. In this new approach, we presented a different technique to produce

the diagnoser. In fact, the diagnoser here is no longer represented as an automaton but as a pair of sets of inequalities in variables representing the number of firing observable transitions. To produce these sets, we first create two sets after adding a constraint (expressing the normal behaviour) and its negation (expressing the faulty behaviour) to the set of inequalities created from the state equation. Then, the IFME method is applied to eliminate the variables representing unobservable transitions from these sets. The two resulting sets are used for the purpose of diagnosis. The proposed approach has been applied to systems with a single fault. However, the extension to include systems with multiple faults can be made. In parallel, we can somewhat relax the cyclicity assumption. These two points will be demonstrated in the next chapter.

# CHAPTER 5

# FAULT DIAGNOSIS IN PETRI NETS: A MORE GENERAL CASE

Based on the results obtained in the previous chapter, this chapter extends these results in several ways. First, the assumption of having no cycles of transitions in the nets has been relaxed to the case where only cycles of unobservable transitions are not permitted. By this, we address the problem of fault diagnosis in a more general case of Petri nets. Also, the case of multiple fault types in which each type has multiple faults is considered.

To address this more general case using the IFME method, the idea of tracking diagnosis history is introduced. Using this idea is necessary to overcome the problem of not capturing the order of events in the state equation formulation of Petri nets, i.e. one solution for state equation could represent more than one different sequence in the language of Petri nets. The lack of the order information in the state equation could cause misleading in making diagnosis decisions in cases where the order of events is required to ensure a diagnosis state.

This chapter proceeds as follows. We start by explaining how faults can be expressed as inequalities considering the case of multiple faults. Based on this, an extension of the diagnoser definition described in Chapter 4 is given. In addition, we prove the correctness of the obtained results and analyse the complexity of the approach. Then, algorithms for offline construction of a diagnoser and subsequent online diagnosis of faults are presented. For a better understanding, the extended approach is applied to a Petri net example.

## 5.1 Fault diagnosis in Petri nets

In this section, we generalise the description of the fault diagnosis problem in DES modelled by Petri nets as outlined in Section 4.1 to the case of having multiple fault types where each type represents a set of faults. A system may have more than one type of fault. Thus, the set $T_f$ is further partitioned to $T_f^1, T_f^2, \ldots, T_f^r$ representing different types of faults. Since it is not required to identify uniquely the occurrence of every fault in a given type, firing of any transition $t \in T_f^i$ implies that a fault of type $T_f^i$ has occurred.

The functionality of the diagnoser is to use such information (observations) to identify a diagnosis state as one of the following [4, 8] : 1) *Normal* state - when all sequences in $L(\mathcal{N}, M_0)$ with the same $\mathbf{s}$ have no fault transition from the set $T_f$; 2) $T_f^i - Certain$ state is obtained when all sequences in $L(\mathcal{N}, M_0)$ with the same $\mathbf{s}$ have a fault transition from the set $T_f^i$; and 3) $T_f^i - Uncertain$ state in which there exists two sequences having the same $\mathbf{s}$, one of them has a fault transition from $T_f^i$, but the other has none.

All assumptions of Section 4.1 are still applied to the extended results in this chapter except assumptions (e) and (g). In other words, the cycles are allowed provided that not all transitions on these cycles are unobservable. In addition, the case of having multiple fault types where each type has multiple faults is considered.

## 5.2 The IFME method for fault diagnosis

In this section, it is shown how the IFME method can be used to diagnose faults in Petri nets modelling DES where the nets have no cycle of unobservable transitions. Moreover, the case of multiple fault types where each type has multiple faults is considered.

### 5.2.1 Modelling and diagnosing multiple faults

A family of faults can be represented as an inequality by extending the formulation introduced in Section 4.2. Considering that $T_f^i, i = 1, 2, \ldots, r$, is a fault type; we associate to each type $T_f^i$ two

inequalities $\neg\mathbf{c}_i$ and $\mathbf{c}_i$ described in (5.1) and (5.2) respectively.

$$\sum_{t\in T_f^i} \#(t,\sigma) > 0 \tag{5.1}$$

$$\sum_{t\in T_f^i} \#(t,\sigma) \leq 0 \tag{5.2}$$

Then, any fault transition from $T_f^i$ appearing in $\sigma$ implies that (5.1) holds. In contrast, no fault transition of $T_f^i$ appearing in $\sigma$ implies that (5.2) holds. To generalise the same notion on the whole set $T_f$, we can say that given a sequence $\sigma$, no fault from set $T_f$ appears in $\sigma$ if and only if (5.3) holds.

$$\sum_{i=1}^{r} \sum_{t\in T_f^i} \#(t,\sigma) \leq 0 \tag{5.3}$$

On the contrary, at least one fault appears in $\sigma$ if and only if (5.4) is evaluated to *True*.

$$\sum_{i=1}^{r} \sum_{t\in T_f^i} \#(t,\sigma) > 0 \tag{5.4}$$

By considering the new formulation of faults, described above, and the extended definition of the diagnoser introduced in Definition 4.4, the following definition is presented.

**Definition 5.1.** A diagnoser is a function $\Delta : T_o^* \times 2^{T_f} \rightarrow \{NoFault, Faulty, Uncertain\}$ that associates with each observed sequence $\mathbf{s}$ with respect to the fault type $T_f^i, i \in \{1,\ldots,r\}$, one of the following diagnosis states:

- $\Delta(\mathbf{s}, T_f^i) = NoFault$ if $\forall\sigma \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma) = \mathbf{s}$, $\#(\sigma) \vDash \mathbf{c}_i$ holds. This state ensures that no fault of type $T_f^i$ has occurred as there is no sequence having the same observation as $\mathbf{s}$ containing a fault transition in $T_f^i$.

- $\Delta(\mathbf{s}, T_f^i) = Faulty$ if $\forall\sigma \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma) = \mathbf{s}$, $\#(\sigma) \vDash \neg\mathbf{c}_i$ holds. This state shows faulty behaviour as all sequences having the same observation as $\mathbf{s}$ contain a fault transition in $T_f^i$, i.e. a fault from set $T_f^i$ has certainly occurred during the sequence $\mathbf{s}$.

- $\Delta(\mathbf{s}, T_f^i) = Uncertain$ if there exists two sequences $\sigma_1$, $\sigma_2 \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma_1) =$

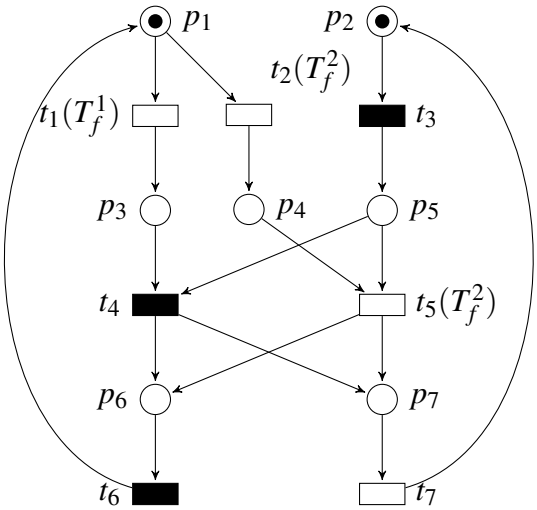



Figure 5.1: A Petri net example

$\pi(\sigma_2) = \mathbf{s}$, $\#(\sigma_1) \models \mathbf{c}_i$ and $\#(\sigma_2) \models \neg\mathbf{c}_i$ hold. In this case, the behaviour of the system is ambiguous because both *NoFault* and *Faulty* states are possible during the observed sequence $\mathbf{s}$.

If $\Delta(\mathbf{s}, T_f^i) = NoFault$ for all $i = 1, \ldots, r$, then we are certain that no fault from any type has occurred during the observed sequence $\mathbf{s}$. To explain the fault diagnosis notions with respect to the extended diagnoser definition mentioned above, let us present the following example:

**Example 5.1.** Consider the Petri net depicted in Figure. 5.1. In this Petri net, the set of places is $P = \{p_1, \ldots, p_7\}$ and the initial marking is $M_0 = [1100000]$; the set of transitions is $T = \{t_1, \ldots, t_7\}$. In the figure, observable transitions are depicted by solid rectangles, while empty rectangles represent unobservable transitions. Moreover, we model two types of faults corresponding to two sets of transitions, $T_f^1 = \{t_1\}$ and $T_f^2 = \{t_2, t_5\}$.

Since we have two fault types, two pairs of inequalities need to be created to express faults from both types. The inequalities $\mathbf{c}_1 := x_1 \leq 0$ and $\neg\mathbf{c}_1 := x_1 > 0$ are associated to $T_f^1$ . While $T_f^2$ can be represented by the inequalities $\mathbf{c}_2 := x_2 + x_5 \leq 0$ and $\neg\mathbf{c}_2 := x_2 + x_5 > 0$.

Suppose that the diagnoser observes no sequence ($\mathbf{s} = \varepsilon$), then $\Delta(\mathbf{s}, T_f^1) = \Delta(\mathbf{s}, T_f^2) =$ *Uncertain* because $\mathbf{s}$ might correspond to two sequences, $\sigma_1 = t_1$ and $\sigma_2 = t_2$. In which case, $\mathbf{x}_1 = \#(\sigma_1) = (1,0,0,0,0,0,0)$ and $\mathbf{x}_2 = \#(\sigma_2) = (0,1,0,0,0,0,0)$. Then $\#(\sigma_1) \models \neg\mathbf{c}_1$, but $\#(\sigma_2) \models \mathbf{c}_1$. Also, $\#(\sigma_1) \models \mathbf{c}_2$ but $\#(\sigma_2) \models \neg\mathbf{c}_2$.

If we assume now that $\mathbf{s} = t_3t_4$, then $\Delta(\mathbf{s}, T_f^1) = Faulty$, but $\Delta(\mathbf{s}, T_f^2) = NoFault$. The diagnoser estimates such a state because all sequences with $\pi(\mathbf{s}) = t_3t_4$ have a fault from type $T_f^1$, but no fault from the type $T_f^2$ appears in these sequences. In particular, there exist only four sequences $\sigma_1 = t_1t_3t_4$, $\sigma_2 = t_3t_1t_4$, $\sigma_3 = t_1t_3t_4t_7$ and $\sigma_4 = t_3t_1t_4t_7$ with $\pi(\sigma_1) = \pi(\sigma_2) = \pi(\sigma_3) = \pi(\sigma_4) = t_3t_4$. In this case, we have $\#(\sigma_1) = \#(\sigma_2) = (1,0,1,1,0,0,0)$ and $\#(\sigma_3) = \#(\sigma_4) = (1,0,1,1,0,0,1)$. Then, $\#(\sigma_1), \#(\sigma_2), \#(\sigma_3)$ and $\#(\sigma_4)$ satisfy $\neg\mathbf{c}_1$ and $\mathbf{c}_2$.

### 5.2.2 The proposed approach for fault diagnosis

In Chapter 4, we have introduced the idea of using the IFME method for fault diagnosis in partially-observed DES modelled by Petri nets. Under the assumptions that Petri nets are *acyclic* and have a single fault, we showed that the diagnoser can be expressed as two sets of inequalities. These sets are derived from the state equation of Petri nets augmented by $\mathbf{c}$ or $\neg\mathbf{c}$. In this chapter, we relax these assumptions to the case where the Petri nets under study have no cycle of unobservable transitions. In addition, we consider the case of multiple faults.

The IFME approach for fault diagnosis in Petri nets satisfying the assumptions (a)-(g) listed in Section 5.1, can be outlined as follows. Supposing that $(\mathcal{N}, M_0)$ is a Petri net with an initial marking $M_0$, without any loss of generality, assume that we have renamed the transitions of $\mathcal{N}$ such that the first $k$ transitions are observable, i.e. $T_o = \{t_1, t_2, \ldots, t_k\}$. The remaining transitions are unobservable, i.e. $T_u = \{t_{k+1}, t_{k+2}, \ldots, t_n\}$.

We further suppose that the set of fault transitions in $\mathcal{N}$ is $T_f \subseteq T_u$ and all faults are of the same type. We introduce variables $x_1, x_2, \ldots, x_n$ representing the number of firing of $t_1, t_2, \ldots, t_n$, respectively (see Remark 4.1). Denote by $I := -A\mathbf{x} \leq M_0$ the state equation constraints equipped with $\mathbf{x} \geq \vec{\mathbf{0}}$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ (see Lemma 4.1). We further assume that $\mathbf{c}$ is the inequality $\sum_{t_j \in T_f} x_j \leq 0$ and $\neg\mathbf{c}$ is the negation of $\mathbf{c}$, i.e. the inequality $\sum_{t_j \in T_f} x_j > 0$. For each firing sequence $\sigma$ of $(\mathcal{N}, M_0)$, if $\sigma$ contains a fault from $T_f$, then $\mathbf{x} = \#(\sigma)$ satisfies $\neg\mathbf{c}$ (see Definition 4.2). Conversely, for a firing sequence $\sigma$, if $\mathbf{x}$ satisfies $\mathbf{c}$, then $\sigma$ has no fault transition.

The general idea of our approach to address the problem of fault diagnosis where cycles are permitted can be illustrated in Figure. 5.2. In fact, the only difference between the present notion

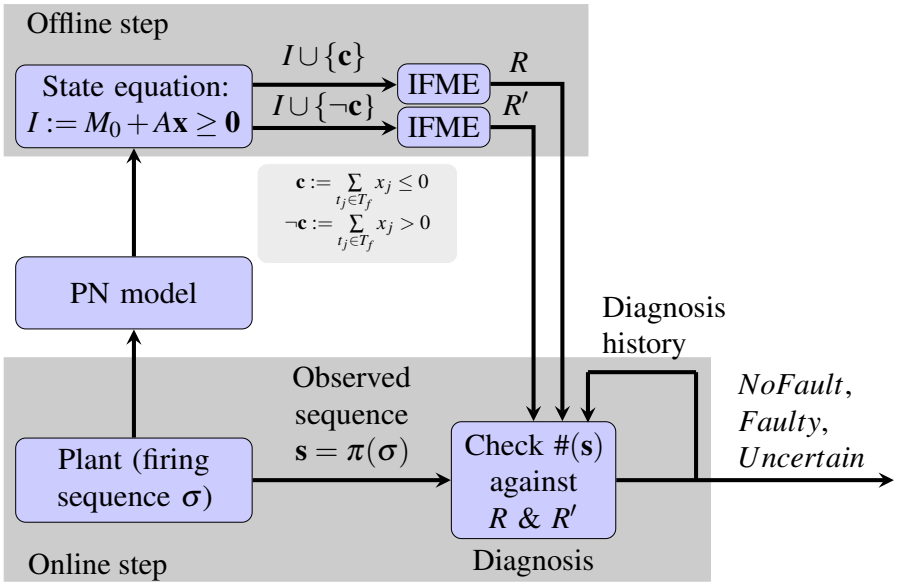



Figure 5.2: Sketch of the proposed approach in Petri nets with no cycle of unobservable transitions

and the one introduced in Section 4.3.1 involves adding a concept of the *diagnosis history* as stated in Figure. 5.2. The introduction of this concept is necessary to overcome the problem of not considering the order of fired transitions by state equation representation, which will be explained later. Using the idea of diagnosis history, the process of fault diagnosis can be divided into two steps:

- **Offline step:** in this step, we start from the Petri net model to first obtain a set of inequalities $I$ created from the state equation plus the non-negativity constraints on $\mathbf{x}$. Then, two sets of inequalities $I \cup \{\mathbf{c}\}$ and $I \cup \{\neg\mathbf{c}\}$ are created. Applying the IFME method simultaneously to both $I \cup \{\mathbf{c}\}$ and $I \cup \{\neg\mathbf{c}\}$, two reduced sets $R$ and $R'$ are obtained by eliminating every variable corresponding to a transition in the set $T_u$.

- **Online step:** during this step, the reduced sets of inequalities $R$ and $R'$ along with the diagnosis history are used to compute diagnosis states. In effect, the diagnosis history is only needed when the Parikh vector of the observed sequence $\mathbf{s}$ satisfies both $R$ and $R'$.

In the following, we establish the mathematical foundations on which the proposed approach in this chapter relies. This foundation is captured by the following lemma and theorem which summarise the main results in diagnosing fault occurrences. Inspired by Theorem 16 in [33],

Lemma 5.1 is first presented; which is necessary to prove Theorem 5.1 below.

**Lemma 5.1.** Suppose that $v$ is a $n$-dimensional column vector and $M$ is a reachable marking in a Petri net $\mathcal{N}$ such that $M' = M + Av \geq \vec{\mathbf{0}}$. Considering that $\mathcal{N}_v$ (see Definition 3.1) is cycle free, then there exists a sequence $\sigma \in T_v^*$ ($T_v$ is the set of transitions in $\mathcal{N}_v$) such that $M_v \xrightarrow{\sigma} M'_v$ and $\#(\sigma) = v$, where $M_v$ and $M'_v$ are restrictions of $M$ and $M'$ to places of $\mathcal{N}_v$. In addition, $\sigma$ can fire under $M$ resulting in $M'$ such that $M \xrightarrow{\sigma} M'$.

*Proof.* Consider subnet $\mathcal{N}_v$ is *acyclic*, then there is at least one transition $t$ enabled at $M_v$ (if not, there exists a token-free source place for the disabled transitions or a token-free circuit deadlock on $\mathcal{N}_v$ at $M_v$, but this contradicts $M' \geq \vec{\mathbf{0}}$ and our assumption that $\mathcal{N}_v$ is acyclic). Let $M'' = M + A\mathbf{u}$ is the resulting marking after firing $t$ (one of the enabled transitions in $\mathcal{N}_v$) and $v' = v - \mathbf{u}$. Then, $M' = M'' + Av' \geq \vec{\mathbf{0}}$, $v' \geq \vec{\mathbf{0}}$ and the subnet $\mathcal{N}_{v'}$ is *acyclic*. By the same argument, $\mathcal{N}_{v'}$ has at least one enabled transition. Repeating this process until $v'$ becomes $\vec{\mathbf{0}}$, we will reach to the marking $M' = M' + A\vec{\mathbf{0}} \geq \vec{\mathbf{0}}$. This completes the proof. □

**Definition 5.2.** *The most recent diagnosis state*: suppose that $\mathbf{s} = \mathbf{s}'t$ is a sequence of observed events, where $\mathbf{s}' \in T_o^*$ and $t \in T_o$, then the most recent diagnosis state of $\mathbf{s}$ is $\Delta(\mathbf{s}', T_f)$.

Note that the most recent diagnosis state of the empty string $\varepsilon$ is *NoFault* because we assume that the system starts from a non-faulty state.

**Theorem 5.1.** Assume that $(\mathcal{N}, M_0)$ is a Petri net satisfying the assumptions in Section 5.1. Suppose that $I$ is the set of inequalities $-A\mathbf{x} \leq M_0$ created from the state equation of $\mathcal{N}$, equipped with $\mathbf{x} \geq \vec{\mathbf{0}}$, see Lemma 4.1. Assume that $T = T_o \cup T_u$, $T_o = \{t_1, \ldots, t_k\}$, $T_u = \{t_{k+1}, \ldots, t_n\}$ and the set of fault transitions is $T_f \subseteq T_u$, which has one fault type. The vector of variables $x_1, \ldots, x_n$ corresponds to the number of firing the transitions $t_1, \ldots, t_n$. Assume also that $\mathbf{c}$ is the inequality $\sum_{t_j \in T_f} x_j \leq 0$ and $\neg\mathbf{c}$ is its negation, i.e. $\neg\mathbf{c} := \sum_{t_j \in T_f} x_j > 0$. Suppose that sets of inequalities $R$ and $R'$ are respectively produced from applying the IFME to both $I \cup \{\mathbf{c}\}$ and $I \cup \{\neg\mathbf{c}\}$ to eliminate all variables corresponding to transitions in $T_u$. Then, for any given sequence of observed events $\mathbf{s} = \mathbf{s}'t$, $\mathbf{s}' \in T_o^*$ and $t \in T_o$ such that there exists a firing sequence $\sigma$ in $\mathcal{N}$ ($M_0 \xrightarrow{\sigma} M$) and

$\pi(\sigma) = \mathbf{s}$, $\Delta(\mathbf{s}, T_f)$ is determined as follows:

$$\Delta(\mathbf{s}, T_f) = \begin{cases} \mathit{NoFault} & \text{if } (\#(\mathbf{s}) \not\models R') \\ \mathit{Faulty} & \text{if } (\#(\mathbf{s}) \not\models R) \\ & \vee((\#(\mathbf{s}) \models R) \wedge (\#(\mathbf{s}) \models R') \\ & \wedge(\Delta(\mathbf{s}', T_f) = \mathit{Faulty})) \\ \mathit{Uncertain} & \text{if } (\#(\mathbf{s}) \models R) \wedge (\#(\mathbf{s}) \models R') \\ & \wedge((\Delta(\mathbf{s}', T_f) = \mathit{NoFault}) \vee (\Delta(\mathbf{s}', T_f) = \mathit{Uncertain})) \\ \mathit{Impossible} & \text{if } (\#(\mathbf{s}) \not\models R) \wedge (\#(\mathbf{s}) \not\models R') \end{cases}$$

*Proof.* In the following, we assume that $\#(\mathbf{s}) = (\alpha_1, \ldots, \alpha_k)$.

**Proof of $\Delta(\mathbf{s}, T_f) = \mathit{NoFault}$:** by contradiction, assume that $\#(\mathbf{s}) \not\models R'$, but the diagnosis state is not $\mathit{NoFault}$. If $\#(\mathbf{s}) \not\models R'$, then for every valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$, $\nu \not\models I \cup \{\neg\mathbf{c}\}$ by Theorem 3.1. Since $\sigma$ is a firing sequence, we are certain that $\nu \models I$, see Lemma 4.1. Hence, $\nu \not\models \neg\mathbf{c}$, i.e. $\nu \models \mathbf{c}$. As a result, $\forall\sigma' \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma') = \mathbf{s}$, $\#(\sigma') \models \mathbf{c}$, i.e. $\sum_{t \in T_f} \#(t, \sigma') \leq 0$. Hence, a fault has not occurred during observing $\mathbf{s}$, i.e. $\Delta(\mathbf{s}, T_f) = \mathit{NoFault}$. This contradicts the assumption.

**Proof of $\Delta(\mathbf{s}, T_f) = \mathit{Faulty}$:** here we have two cases to be proved.

**Case i:** if $\#(\mathbf{s}) \not\models R$ holds. Using the same argument in **Proof of** $\Delta(\mathbf{s}, T_f) = \mathit{NoFault}$ replacing $R'$ with $R$, we can prove this case.

**Case ii:** if $(\#(\mathbf{s}) \models R) \wedge (\#(\mathbf{s}) \models R') \wedge (\Delta(\mathbf{s}', T_f) = \mathit{Faulty})$ holds. Since $\Delta(\mathbf{s}', T_f) = \mathit{Faulty}$ holds, i.e. the most recent diagnosis state is $\mathit{Faulty}$, then a fault has occurred during the observed sequence $\mathbf{s}'$. Also, since the fault propagates to all states following the $\mathit{Faulty}$ state, then the fault has also occurred during $\mathbf{s} = \mathbf{s}'t$.

**Proof of $\Delta(\mathbf{s}, T_f) = \mathit{Uncertain}$:** we first assume that $\mathbf{s} = \varepsilon$, then there exists one possible case for the most recent diagnosis state, particularly $\mathit{NoFault}$, because we suppose that the system starts from a non-faulty state. Now let us prove the result in the case of $\mathbf{s} = \varepsilon$. If $\#(\mathbf{s}) \models R$, then there exists a valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$ and $\nu \models I \cup \{\mathbf{c}\}$ by Theorem 3.1. If $\nu \models I \cup \{\mathbf{c}\}$, then $\nu \models I$, i.e. $\nu$ satisfies $M' = M_0 + A\nu \geq \vec{\mathbf{0}}$. Since

$\mathbf{s}$ has no observable transitions ($\mathbf{s} = \varepsilon$), then the subnet $\mathscr{N}_v$ has only unobservable transitions. Again, by the assumption that no cycle of unobservable transitions exists in $\mathscr{N}$, then $\mathscr{N}_v$ is cycle free. As a result, there exists $\sigma' \in T_v^*$ such that $M_0 \xrightarrow{\sigma'} M'$ and $\#(\sigma') = v$ by Lemma 5.1. Hence, the sequence $\sigma'$ has no fault. Likewise, we can prove that if $\#(\mathbf{s}) \vDash R'$, there exists another sequence having a fault. Since there exists two sequences having the same observed sequence $\mathbf{s}$, but one has a fault and the other has none, then we have an *Uncertain* state.

Now, assume that $\mathbf{s} = \mathbf{s}'t$, $t \in T_o$ and $\mathbf{s}' \in T_o^*$. Then, there are two cases to be considered:

**Case i:** when the most recent diagnosis state is *NoFault* ($\Delta(\mathbf{s}', T_f) = NoFault$). If $\#(\mathbf{s}) \vDash R$, then there exists a valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $v = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$ and $v \vDash I \cup \{\mathbf{c}\}$ by Theorem 3.1. If $v \vDash I \cup \{\mathbf{c}\}$, then $v \vDash I$, i.e. $M'' = M_0 + Av \geq \vec{\mathbf{0}}$. Since no fault has occurred during observing $\mathbf{s}'$, and $t$ is an observable transition, then we are certain that all sequences $\sigma't$ such that $M_0 \xrightarrow{\sigma't} M'$ and $\pi(\sigma') = \mathbf{s}'$ have no fault. Assuming $y = v - \#(\sigma't)$, $y \in \mathbb{N}^n$, then $M'' = M' + Ay \geq \vec{\mathbf{0}}$. Since the subnet $\mathscr{N}_y$ has only unobservable transitions, then $\mathscr{N}_y$ is cycle free. As a result, there exists $\sigma'' \in T_y^*$ such that $M' \xrightarrow{\sigma''} M''$ and $\#(\sigma'') = y$ by Lemma 5.1. Hence, the sequence $\sigma't\sigma''$ with $\#(\sigma't\sigma'') = v$ has no fault. Likewise, we can prove that if $\#(\mathbf{s}) \vDash R'$, there exists another sequence having a fault. Since there exists two sequences with the same $\mathbf{s}$, but one of them has a fault and the other does not, then we have an *Uncertain* state.

**Case ii:** when the most recent diagnosis state is *Uncertain* ($\Delta(\mathbf{s}', T_f) = Uncertain$). If $\#(\mathbf{s}) \vDash R$, then there exists a valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $v = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$ and $v \vDash I \cup \{\mathbf{c}\}$ by Theorem 3.1. If $v \vDash I \cup \{\mathbf{c}\}$, then $v \vDash I$, i.e. $M'' = M_0 + Av \geq \vec{\mathbf{0}}$. Since we have an *Uncertain* state while observing $\mathbf{s}'$, i.e. the most recent diagnosis state is *Uncertain*, and $t$ is an observable transition, then we still have the same state for any sequence $\sigma't$ such that $M_0 \xrightarrow{\sigma't} M'$ and $\pi(\sigma') = \mathbf{s}'$. Assuming $y = v - \#(\sigma't)$, $y \in \mathbb{N}^n$, then $M'' = M' + Ay \geq \vec{\mathbf{0}}$. Since the subnet $\mathscr{N}_y$ has only unobservable transitions, then $\mathscr{N}_y$ is cycle free. As a result, there exists $\sigma'' \in T_y^*$ such that $M' \xrightarrow{\sigma''} M''$ and $\#(\sigma'') = v$ by Lemma 5.1. Hence, the sequence $\sigma't\sigma''$ with $\#(\sigma't\sigma'') = v$ has no fault. Similarly, we can prove that if $\#(\mathbf{s}) \vDash R'$, there exists another sequence having a fault. Since there are two sequences having the same observed sequence $\mathbf{s}$, but one has a fault and the other does not, then we have an *Uncertain* state.

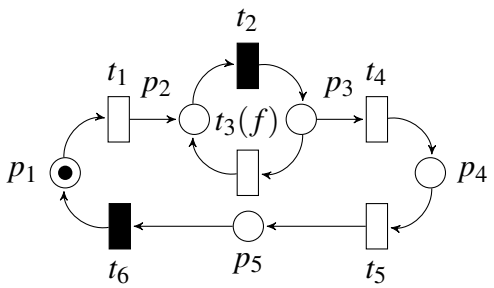



Figure 5.3: Tracking diagnosis history

**Proof of the case** *Impossible*: assume that $\#(\mathbf{s}) \not\models R$ and $\#(\mathbf{s}) \not\models R'$, but this case is possible. If $\#(\mathbf{s}) \not\models R$, then for every valuation $(\alpha_{k+1},\ldots,\alpha_n)$ of $(x_{k+1},\ldots,x_n)$ such that $\nu = (\alpha_1,\ldots,\alpha_k,\alpha_{k+1},\ldots,\alpha_n)$, $\nu \not\models I \cup \{\mathbf{c}\}$ by Theorem 3.1. Also, if $\#(\mathbf{s}) \not\models R'$, then for every valuation $(\beta_{k+1},\ldots,\beta_n)$ of $(x_{k+1},\ldots,x_n)$ such that $\nu = (\alpha_1,\ldots,\alpha_k,\beta_{k+1},\ldots,\beta_n)$, $\nu \not\models I \cup \{\neg\mathbf{c}\}$ by Theorem 3.1. Rephrasing this statement, we can say that there exists a valuation $(\beta_{k+1},\ldots,\beta_n)$ of $(x_{k+1},\ldots,x_n)$ such that $\nu = (\alpha_1,\ldots,\alpha_k,\beta_{k+1},\ldots,\beta_n)$ and $\nu \models I \cup \{\mathbf{c}\}$ taking into account that $\neg\mathbf{c}$ is the negation of $\mathbf{c}$ and $\sigma$ is a firing sequence of $\mathcal{N}$, i.e. $\#(\sigma) \models I$. Here we have contradictory statements. Hence, this case is an impossible case. This contradicts the assumption and completes the proof. □

**Remark 5.1.** Note that the proofs of the diagnosis states *NoFault* and *Faulty* in Theorem 5.1 are still valid for Petri nets which have cycles of unobservable transitions (see Remark 4.2).

In Theorem 5.1 above, the assumption that no cycle of unobservable transitions are permitted is necessary in order not to have spurious solutions of the state equation, i.e. the solutions which do not correspond to any sequence in Petri nets. In addition, the following example demonstrates the importance of using the notion of tracking the diagnosis history in order to obtain correct diagnosis decisions.

**Example 5.2.** Consider the Petri net in Figure. 5.3 where the initial marking is $M_0 = [10000]$, $T_o = \{t_2, t_6\}$ and $T_u = \{t_1, t_4, t_5\}$. In this Petri net, there is a single fault transition modelled by $t_3$. Assume that the sequence $t_2t_2$ is observed. Then, we are certain that the fault has occurred. By Theorem 5.1 we can obtain the same diagnosis state. Now, assume that $\mathbf{s}_1 = t_2t_2t_6$ is observed. Again this sequence is a *Faulty* sequence. Although we obtain that $\#(\mathbf{s}_1) \models R$ and $\#(\mathbf{s}_1) \models R'$,

Theorem 5.1 yields a *Faulty* state because the most recent diagnosis state ($\Delta(t_2t_2, T_f)$) is *Faulty*. However, note that not considering the diagnosis history in this case leads to an incorrect diagnosis decision - *Uncertain* state. This is due to the presence of another sequence $\mathbf{s}_2 = t_2t_6t_2$ with no fault, but $\#(\mathbf{s}_1) = \#(\mathbf{s}_2)$.

From this example, we infer that adopting the notion of tracking the diagnosis history enables the avoidance of problems, where the order of events in an observed sequence can affect the diagnosis decisions. Using this notion, the fault diagnosis process is performed on the basis of individual observed events. In each diagnosis step, the diagnosis state is computed for a single observed event and then propagated to the following steps. In this case, we remove the confusion resulting from having two different observed sequences with the same Parikh vector.

### 5.2.3 Multiple fault types with multiple faults

Let us now investigate the case of multiple fault types in which each type has multiple faults. To address this case, we produce a separate pair of sets of inequalities for each fault type. In that case, when creating a set of inequalities for a given fault type, the transitions representing faults in the other fault types are considered as normal unobservable transitions. We say that a fault of type $T_f^i,\ i = 1,2,\ldots,r$, occurs if and only if at least one fault transition $t \in T_f^i$ fires. The following corollary extends Theorem 5.1 to the case where multiple fault types with multiple faults exist in the system.

**Corollary 5.1.** Assume that $(\mathcal{N}, M_0)$ is a Petri net satisfying the assumptions in Section 5.1. Suppose that $I$ is the set of inequalities $-A\mathbf{x} \leq M_0$ created from the state equation of $\mathcal{N}$, equipped with $\mathbf{x} \geq \vec{\mathbf{0}}$, see Lemma 4.1. Assume that $T = T_o \cup T_u$, $T_o = \{t_1,\ldots,t_k\}$, $T_u = \{t_{k+1},\ldots,t_n\}$, $T_f = T_f^1 \cup \cdots \cup T_f^r$ and $T_f \subseteq T_u$. The vector of variables $x_1,\ldots,x_n$ corresponds to the number of firing the transitions $t_1,\ldots,t_n$. Assume also that $\mathbf{c}_i$ is the inequality $\sum\limits_{t_j \in T_f^i} x_j \leq 0$ and $\neg\mathbf{c}_i := \sum\limits_{t_j \in T_f^i} x_j > 0$ is its negation. For every $i \in \{1,\ldots,r\}$, suppose that sets of inequalities $R_i$ and $R_i'$ are respectively produced from applying the IFME to both $I \cup \{\mathbf{c}_i\}$ and $I \cup \{\neg\mathbf{c}_i\}$ to eliminate all variables corresponding to transitions in $T_u$. Then, for any given sequence of observed events $\mathbf{s} = \mathbf{s}'t$,

$\mathbf{s}' \in T_o^*$ and $t \in T_o$ such that there exists a firing sequence $\sigma$ in $\mathcal{N}$ $(M_0 \xrightarrow{\sigma} M)$ and $\pi(\sigma) = \mathbf{s}$, $\Delta(\mathbf{s}, T_f^i)$ is determined as follows:

$$\Delta(\mathbf{s}, T_f^i) = \begin{cases} NoFault & \text{if } \#(\mathbf{s}) \not\models R_i' \\ Faulty & \text{if } (\#(\mathbf{s}) \not\models R_i) \\ & \vee((\#(\mathbf{s}) \models R_i) \wedge (\#(\mathbf{s}) \models R_i') \\ & \wedge(\Delta(\mathbf{s}', T_f^i) = Faulty)) \\ Uncertain & \text{if } (\#(\mathbf{s}) \models R_i) \wedge (\#(\mathbf{s}) \models R_i') \\ & \wedge((\Delta(\mathbf{s}', T_f^i) = NoFault) \vee (\Delta(\mathbf{s}', T_f^i) = Uncertain)) \\ Impossible & \text{if } (\#(\mathbf{s}) \not\models R_i) \wedge (\#(\mathbf{s}) \not\models R_i') \end{cases}$$

*Proof.* It follows from Theorem 5.1; in particular, we can obtain a formal proof of this corollary by repeating the proof of Theorem 5.1 for each $i \in \{1, \ldots, r\}$. □

### 5.2.4 Fault diagnosis algorithms: cycles are permitted

In this section, the algorithms introduced in Section 4.3.2 are extended to address the problem of fault diagnosis in Petri nets satisfying the assumptions (a)-(g) of Section 5.1. Also, the extended algorithms consider the case of multiple fault types based on results obtained in Theorem 5.1 and Corollary 5.1.

Algorithm 5.1 takes three parameters: a Petri net $(\mathcal{N}, M_0)$, the set of unobservable transitions $T_u$ and fault types set $T_f$. The output of this algorithm is a set of pairs of sets of inequalities, namely $R_i$ and $R_i'$, $\forall i = 1, \ldots, r$, which will be used later for the purpose of diagnosis. Algorithm 5.1 begins with an initialisation of the set of inequalities $I$ in step 1. In steps 3-4, a pair of inequalities $(\mathbf{c}_i, \neg\mathbf{c}_i)$ is created for each fault type $i \in \{1, \ldots, r\}$ as described in Section 5.2.3. Then, augmented sets of inequalities $I \cup \{\mathbf{c}_i\}$ and $I \cup \{\neg\mathbf{c}_i\}$ are produced. These augmented sets are used to initialise the sets of inequalities $R_i, R_i'$ in steps 5-6.

The sets of inequalities $R_i, R_i'$ are updated in steps 7-10 by recursively applying the IFME method (*IFME_method*) $|T_u|$ times in order to eliminate all variables $x_j$, $\forall t_j \in T_u$. The resulting

**Algorithm 5.1** : build the diagnoser.

**Input:** A Petri net $(\mathcal{N}, M_0)$,
a set of unobservable transitions $T_u$,
fault types set $\{T_f^i | 1 \leq i \leq r\}$.
**Output:** A set of pairs $(R_i, R_i')_{i=1,2,\ldots,r}$.

1: **Let** $I \leftarrow \{-A\mathbf{x} \leq M_0\} \cup \{-x_j \leq 0 \mid_{j=1,\ldots,n}\}$
2: **for all** $i$ such that $i \in \{1,2,\ldots,r\}$ **do**
3: **Let** $\mathbf{c}_i \leftarrow \sum_{t_j \in T_f^i} x_j \leq 0$
4: **Let** $\neg\mathbf{c}_i \leftarrow \sum_{t_j \in T_f^i} -x_j \leq -1$
5: $R_i \leftarrow I \cup \{\mathbf{c}_i\}$
6: $R_i' \leftarrow I \cup \{\neg\mathbf{c}_i\}$
7: **for all** $t_j$ such that $t_j \in T_u$ **do**
8: $R_i \leftarrow IFME\_method(R_i, x_j)$
9: $R_i' \leftarrow IFME\_method(R_i', x_j)$
10: **end for**
11: **end for**

sets of inequalities $R_i$ and $R_i'$ represent the diagnoser used online for fault diagnosis.

Algorithm 5.2 is used online to diagnose faults. Its inputs are fault types set $T_f$ in addition to sets of inequalities $R_i$ and $R_i'$ produced by Algorithm 5.1. The output of Algorithm 5.2 is the diagnosis state $\{NoFault, Faulty, Uncertain\}$ (see Definition 5.1). This algorithm starts by initialising the observed sequence $\mathbf{s}$ by the empty string $\varepsilon$. Then, in particular in step 2, it enters into a loop to monitor the system state to check whether a fault has occurred. In step 3, the algorithm waits until a new event $e$ is observed and then adds it to the previous sequence of observed events $\mathbf{s}'$, creating the sequence $\mathbf{s}$. For each fault type $T_f^i$, $1 \leq i \leq r$, $\#(\mathbf{s})$ is checked against both $R_i'$ and $R_i$ in steps 6 and 8. The purpose of this checking is to decide whether a fault of type $T_f^i$ has occurred. Then, the diagnosis state is determined in steps 7 and 9. In the case where $\#(\mathbf{s})$ satisfies both $R_i$ and $R_i'$ (step 10), then a further check is performed to determine the diagnosis state (steps 11-15) as described in Theorem 5.1 and Corollary 5.1. This check considers using the diagnosis state of the observed sequence $\mathbf{s}'$ before observing $e$ (the most recent diagnosis state) to make a diagnosis decision when observing $e$.

**Algorithm 5.2** : online fault diagnosis.

**Input:** The fault types set $\{T_f^i | 1 \leq i \leq r\}$,
a set of pairs $(R_i, R_i')_{i=1,2,\ldots,r}$ as defined in Algorithm 5.1.
**Output:** A diagnosis state $\{NoFault, Faulty, Uncertain\}$.

**Let** $\mathbf{s} \leftarrow \varepsilon$
**while** true **do**
  **if** a new event $e$ is observed **then**
    **Let** $\mathbf{s}' \leftarrow \mathbf{s}$, $\mathbf{s} \leftarrow \mathbf{s}'e$
    **for all** $i$ such that $i \in \{1, 2, \ldots, r\}$ **do**
      **if** $\#(\mathbf{s}) \not\models R_i'$ **then**
        $\Delta(\mathbf{s}, T_f^i) \leftarrow NoFault$
      **else if** $\#(\mathbf{s}) \not\models R_i$ **then**
        $\Delta(\mathbf{s}, T_f^i) \leftarrow Faulty$
      **else if** $\#(\mathbf{s}) \models R_i$ **and** $\#(\mathbf{s}) \models R_i'$ **then**
        **if** $\Delta(\mathbf{s}', T_f^i) = Faulty$ **then**
          $\Delta(\mathbf{s}, T_f^i) \leftarrow Faulty$
        **else if** $\Delta(\mathbf{s}', T_f^i) = NoFault$ **or** $\Delta(\mathbf{s}', T_f^i) = Uncertain$ **then**
          $\Delta(\mathbf{s}, T_f^i) \leftarrow Uncertain$
        **end if**
      **end if**
    **end for**
  **end if**
**end while**

### 5.2.5 Computational complexity

This section presents a mathematical formulation of computational requirements (time and space complexities) of the proposed approach. It is well known that the difficulty in solving the fault diagnosis problem is in finding the best compromise between space complexity (size of the diagnoser created offline) and the time complexity of the algorithm using the diagnoser online to compute the diagnosis. Thus, our focus is on computing these requirements in the context of our approach.

**Proposition 5.1.** Suppose $\mathcal{N}$ is a Petri net with an initial marking $M_0$ and $n$, $k$ and $k_1$ represent $|T|$, $|T_o|$ and $|T_u|$, respectively. In addition, suppose that $m_I$ represents the number of inequalities in the initial set; then,

- the size of the diagnoser (space complexity) in the best case (the worst case) is $O(m_I)$

$(O(m_I^{2^{k_1}}))$, where $k_1$ itself represents the number of eliminations,

- the time complexity in the best case (the worst case) of Algorithm 5.2 is $O(1)$ $(O(m_F))$, where $m_F$ is the number of inequalities in the final set (diagnoser size).

*Proof.* Let us first compute the size of the diagnoser represented by sets of inequalities. Suppose that $p_n$, $q_n$ and $r_n$ are the number of positive, negative and zero coefficients of the variable $x_n$ to be eliminated, respectively. Then, the best case occurs when $p_n = q_n = 1$ and $r_n = m_I - 2$. In which case, we obtain the lowest number of inequalities in each elimination step. In effect, the number of inequalities can never be higher than $m_I$. Thus, reduced sets of inequalities will have $O(m_I)$ of inequalities after $k_1$ eliminations.

On the other hand, the worst case occurs when $p_n = q_n$ and $r_n = 0$ as it results in the highest number of possible pairs of inequalities to be considered. In other words, occurrence of this case generates $\left(\frac{m_I}{2}\right)^2$ of inequalities in each elimination step. Let $T(m_I,k_1)$ be the number of inequalities in a resulting set after applying the IFME method to eliminate $k_1$ variables from a set with $m_I$ inequalities. If $k_1 = 1$, then $T(m_I,k_1) = (\frac{m_I}{2})^2$, i.e. if there exists just one variable elimination step, then this number will be equal to $\left(\frac{m_I}{2}\right)^2$. However, if $k_1 > 1$, then $T(m_I,k_1)$ can recursively be expressed as the number of inequalities produced after the first elimination, denoted $(\frac{m_I}{2})^2$, plus the number of the inequalities resulting from eliminating the remaining variables whose number is $k_1 - 1$. Thus, the number of inequalities with respect to $m_I$ and $k_1$ , in the worst case, can be written as a *recurrence relation* capturing the following form:

$$T(m_I,k_1) = \begin{cases} T((\frac{m_I}{2})^2, k_1 - 1) + (\frac{m_I}{2})^2 & , k_1 > 1 \\ (\frac{m_I}{2})^2 & , k_1 = 1 \end{cases}$$

Then, using the *iterative substitution* method, this relation can be solved to obtain the closed-form solution as described below:

$$= \quad T\left(\left(\frac{m_I}{2}\right)^2, k_1 - 1\right) + \left(\frac{m_I}{2}\right)^2$$

$$
\begin{aligned}
&= T\left(\left(\frac{\left(\frac{m_I}{2}\right)^2}{2}\right)^2, k_1-2\right)+\left(\frac{\left(\frac{m_I}{2}\right)^2}{2}\right)^2+\left(\frac{m_I}{2}\right)^2\\
&= T\left(\left(\frac{\left(\frac{\left(\frac{m_I}{2}\right)^2}{2}\right)^2}{2}\right)^2, k_1-3\right)+\left(\frac{\left(\frac{\left(\frac{m_I}{2}\right)^2}{2}\right)^2}{2}\right)^2+\left(\frac{\left(\frac{m_I}{2}\right)^2}{2}\right)^2+\left(\frac{m_I}{2}\right)^2\\
&= T\left(\prod_{j=1}^{3}\frac{1}{2^{2^j}}\times m_I^{2^3}, k_1-3\right)+\prod_{j=1}^{3}\frac{1}{2^{2^j}}\times m_I^{2^3}+\prod_{j=1}^{2}\frac{1}{2^{2^j}}\times m_I^{2^2}+\frac{1}{2^2}\times m_I^2\\
&= T\left(\prod_{j=1}^{3}\frac{1}{2^{2^j}}\times m_I^{2^3}, k_1-3\right)+\sum_{k=1}^{3}\left(\prod_{j=1}^{k}\frac{1}{2^{2^j}}\times m_I^{2^k}\right)\\
&\ \ \vdots\\
&= T\left(\prod_{j=1}^{i}\frac{1}{2^{2^j}}\times m_I^{2^i}, k_1-i\right)+\sum_{k=1}^{i}\left(\prod_{j=1}^{k}\frac{1}{2^{2^j}}\times m_I^{2^k}\right)\\
&\quad \text{when } k_1-i=1, \text{ then } i=k_1-1\\
&= T\left(\prod_{j=1}^{k_1-1}\frac{1}{2^{2^j}}\times m_I^{2^{k_1-1}}, 1\right)+\sum_{k=1}^{k_1-1}\left(\prod_{j=1}^{k}\frac{1}{2^{2^j}}\times m_I^{2^k}\right)\\
&= \left(\frac{\prod_{j=1}^{k_1-1}\frac{1}{2^{2^j}}\times m_I^{2^{k_1-1}}}{2}\right)^2+\sum_{k=1}^{k_1-1}\left(\prod_{j=1}^{k}\frac{1}{2^{2^j}}\times m_I^{2^k}\right)\\
&= O(m^{2^{k_1}})
\end{aligned}
$$

Note that the number of inequalities $O(m_I^{2^{k_1}})$ is produced for each fault type $T_f^i$, $i=1,\ldots,r$. Consequently, the total number of inequalities is $O(r\times m_I^{2^{k_1}})$. Since $r\leq k_1$, the highest total number of inequalities can be written as $O(k_1\times m_I^{2^{k_1}})$. Assuming that $m_I>1$, we always have $k_1<m_I^{2^{k_1}}$. As a result, the size of the diagnoser (space complexity) is $O(m_I^{2^{k_1}})$, i.e. it is doubly exponential in the number of unobservable transitions $k_1$.

Now with Algorithm 5.2, considering the number of inequalities of each resulting set after applying the IFME method as $m_F=O(m_I^{2^{k_1}})$, then the time complexity is computed as follows. When the diagnosis state is *NoFault* or *Faulty* and the first inequality of the set is not satisfied, we obtain the best case time complexity. In which case, we stop checking the remaining inequalities and the time complexity is $O(1)$. On the other hand, the worst case occurs when the diagnosis

state is *Uncertain*. In this case, we need to check all inequalities in both sets $R_i$ and $R'_i$ for $i = 1, \ldots, r$. Hence, Algorithm 5.2 requires $O(m_F)$ of comparisons, i.e. the time of computing the diagnosis is polynomial in the size of the diagnoser. □

Obviously, the number of states in the system to be diagnosed is no longer a parameter on which the computational requirements depend. The only parameter appearing in the complexity relation is the number of inequalities in the initial set of inequalities $m_I$; in addition to the number of unobservable transitions $k_1$. Consequently, the time and space complexity of the IFME approach heavily relies on the number of unobservable transitions in Petri net models. Also, although the computational complexities tend to be doubly exponential in the number of unobservable transitions, it is unlikely in practical situations to have Petri nets representing the worst case. Further, for an important subclass of Petri nets, the best case of space complexity (diagnoser size) can be obtained.

**Corollary 5.2.** Suppose that $\mathcal{N}$ is a Petri net in which every transition $t$ has exactly one input place and one output place, i.e. $|{}^\bullet t| = |t^\bullet| = 1$, $\forall t \in T$; $m_I$ and $k_1$ are as defined in Proposition 5.1. Then, our approach does produce a diagnoser whose size corresponds to the best case complexity $O(m_I)$ described in Proposition 5.1.

*Proof.* As previously, assume that $p_n$, $q_n$ and $r_n$ respectively represent the number of positive, negative and zero coefficients of the variable $x_n$ to be eliminated. Then, having exactly one input and one output for the transition corresponding to $x_n$ makes at most $p_n = q_n = 2$ and $r_n = m_I - 4$. Assume that $S(j)$ is the number of inequalities in the $j$th elimination step, then an application of the IFME method will generate $S(j) \leq m_I$ of inequalities, where $j = 1, \ldots, k_1$. In effect, the number of eliminated inequalities will be at most identical to the number of produced inequalities in each elimination step. In other words, the application of the IFME method in each step of the elimination does not increase the number of inequalities. As a result, the number of inequalities of the final set will be $m_F = O(m_I)$. □

A *state machine* which is an ordinary Petri net [33] represents a special case of the Petri nets described in Corollary 5.2. To such a Petri net, we can always transform any automaton using

the procedure stated in [6]. *This indicates that our approach can be efficiently applied for fault diagnosis in automata frameworks.*

Furthermore, the IFME method yields very efficient results (the number of inequalities produced in each elimination step is strongly polynomial) when each inequality has at most two variables [77]. One example of Petri nets whose set of inequalities has at most two variables is a *Marked graph*, see [33], where each place $p$ has exactly one input transition and one output transition, i.e. $|{}^\bullet p| = |p^\bullet| = 1,\ \forall p \in P$.

### 5.2.6 Complexity and redundancy

As described in the previous section, the diagnoser size could be doubly exponential in the number of unobservable transitions to be eliminated. However, most of the inequalities constituting this diagnoser are redundant and can be deleted without any impact on the size of the solution space defined by them. This redundancy results from the nature of the elimination adopted using the IFME method. The deletion of redundant inequalities can end up with refined sets of inequalities whose sizes could be in the order of a single exponential in the number of eliminated variables [81].

In this thesis, we remove a special category of redundancy in which a redundant inequality is defined as follows: assume that there exists two inequalities $e_1 := a_1x_1 + \cdots + a_nx_n \leq b_1$ and $e_2 := c_1x_1 + \cdots + c_nx_n \leq b_2$; then $e_1$ is redundant if $\forall i \in \{1,\ldots,n\}\ (a_i = 0 \rightarrow c_i = 0) \wedge (a_i \leq c_i) \wedge (b_1 \geq b_2)$. For example, let $e_1$ be $-2x_1 + x_2 \leq 4$ and $e_2$ be $-x_1 + 2x_2 \leq 4$, then $e_1$ is a redundant inequality with respect to $e_2$. Removal of these redundant inequalities is performed during the course of elimination, i.e. after each inequality is created. Note that the testing of whether an inequality is redundant requires no more than a polynomial number of comparisons.

Furthermore, there is another type of redundancy consisting of the common inequalities between the sets of inequalities generated by the elimination. In other words, to compute the diagnosis state for a given sequence of observed events, there is no need to test the Parikh vector of the sequence against all common inequalities in the sets $R_i$ and $R'_i$ for $i = 1,\ldots,r$. In effect, one test is enough to decide the state, while all inequalities are the same and can give the result.

Consequently, the time for computing a diagnosis can be improved, especially when there are many common inequalities; which is more likely to happen in practice. For example, assume that we have $r = 10$ of fault types where $|R_i| = |R'_i| = 100$, $i = 1, \ldots, 10$. Without considering the common inequalities, we are required to test $2 \times 10 \times 100 = 2000$ inequalities in the worst case to decide the diagnosis state of all fault types. Assume now that $|R_1 \cap R'_1 \cap \cdots \cap R_{10} \cap R'_{10}| = 60$, i.e. the number of common inequalities in these sets is 60, then it is sufficient to carry out $(2 \times 10 \times 40) + 60 = 860$ inequality tests to decide the same diagnosis state. The difference in gained performance by adding this improvement will be reported later in Section 8.2.2.

### 5.2.7 Illustrative example

Let us consider the Petri net of Figure. 5.1; since we have two fault types, two pairs of sets of inequalities are created to represent the diagnoser.

We start with the fault type $T_f^1$. We extend the set of inequalities $I$ created from the state equation by simultaneously adding the inequalities $\mathbf{c}_1 := x_1 \leq 0$ and $\neg\mathbf{c}_1 := -x_1 \leq -1$ (Note that $\neg\mathbf{c}_1$ is rewritten in the standard form of the set of inequalities $I$, and also the non-negative constraint $x_1 \geq 0$ is previously removed from $I$). As a result, two extended sets of inequalities are obtained, namely $I \cup \{\mathbf{c}_1\}$ and $I \cup \{\neg\mathbf{c}_1\}$. Applying the IFME method to both sets results in the sets $R_1$ and $R'_1$ shown in Table 5.1. Then, $R_1$ and $R'_1$ can be used to diagnose faults from type $T_f^1$. Likewise, two sets of inequalities $R_2$ and $R'_2$ are created to express faults from type $T_f^2$. Since there are two fault transitions $t_2$ and $t_3$ in the set $T_f^2$, the inequalities $\mathbf{c}_2$ and $\neg\mathbf{c}_2$ will be $x_2 + x_5 \leq 0$ and $-x_2 - x_5 \leq -1$, respectively.

After applying the IFME method, all variables corresponding to unobservable transitions $T_u = \{t_1, t_2, t_5, t_7\}$ are eliminated. The sets of inequalities in $R_1$, $R'_1$, $R_2$ and $R'_2$ are in variables representing observable transitions $T_o = \{t_3, t_4, t_6\}$. These sets represent the diagnoser used for estimating the current state of the system for a given observed sequence of events. Note that the inequalities 7 and 8 of the sets $R_1$ and $R'_2$ are respectively redundant, i.e. they can be deleted without affecting the diagnosis results.

Suppose that no sequence is observed ($\mathbf{s} = \varepsilon$), then $\Delta(\mathbf{s}, T_f^1) = \Delta(\mathbf{s}, T_f^2) = Uncertain$ because

Table 5.1: The sets of inequalities resulting from applying the IFME method in the illustrative example

| No. | $R_1$ | $R'_1$ | $R_2$ | $R'_2$ |
|---|---|---|---|---|
| 1 | $x_4 - x_6 \leq 1$ | $x_4 - x_6 \leq 1$ | $x_4 - x_6 \leq 1$ | $x_4 - x_6 \leq 1$ |
| 2 | $-x_3 + x_6 \leq 0$ | $-x_3 + x_6 \leq 0$ | $-x_3 + x_6 \leq 0$ | $-x_3 + x_6 \leq 0$ |
| 3 | $-x_3 + x_4 \leq 0$ | $-x_3 + x_4 \leq 0$ | $-x_3 + x_4 \leq 0$ | $-x_3 + x_4 \leq 0$ |
| 4 | $x_3 - x_6 \leq 2$ | $x_3 - x_6 \leq 2$ | $x_3 - x_6 \leq 2$ | $x_3 - x_6 \leq 2$ |
| 5 | $x_4 \leq 0$ | $x_3 - x_4 - x_6 \leq 1$ | $x_3 - x_4 \leq 1$ | $2x_4 - 2x_6 \leq 1$ |
| 6 | $x_3 - x_4 - x_6 \leq 2$ | | $-x_4 + x_6 \leq 0$ | $-x_3 + 2x_4 - x_6 \leq 0$ |
| 7 | $-x_4 - x_6 \leq 1$ | | $x_3 - x_4 - x_6 \leq 0$ | $x_3 - x_4 - x_6 \leq 2$ |
| 8 | | | | $-x_4 - x_6 \leq 1$ |

$\#(\mathbf{s})$ satisfies $R_1$, $R'_1$, $R_2$ and $R'_2$ and the most recent diagnosis state is *NoFault* (see Theorem 5.1 and Corollary 5.1). In effect, the empty sequence $\varepsilon$ might correspond to two other sequences, $\sigma_1 = t_1$ and $\sigma_2 = t_2$. In which case, for both fault types, there exits two sequences having the same observation, one of them has a fault but the other has none. Note that observing the sequence $t_3$ yields the same diagnosis state of the empty sequence $\varepsilon$. However, the most recent diagnosis state in this case is $\Delta(\varepsilon, T_f^1) = \Delta(\varepsilon, T_f^2) = Uncertain$.

Assume now that the sequence $\mathbf{s} = t_3t_4$ is observed, then $\Delta(\mathbf{s}, T_f^1) = Faulty$, but $\Delta(\mathbf{s}, T_f^2) = NoFault$. The diagnoser estimates such a state because $\#(\mathbf{s})$ satisfies $R'_1$ and $R_2$, but it does not satisfy $R_1$ and $R'_2$. In other words, the sequences $\sigma_1 = t_1t_3t_4$, $\sigma_2 = t_3t_1t_4$, $\sigma_3 = t_1t_3t_4t_7$ and $\sigma_4 = t_3t_1t_4t_7$ with $\pi(\mathbf{s}) = t_3t_4$ have a fault from type $T_f^1$, but no fault from the type $T_f^2$ appears in these sequences.

On the contrary, observing $\mathbf{s} = t_3t_3$ yields $\Delta(\mathbf{s}, T_f^1) = NoFault$ and $\Delta(\mathbf{s}, T_f^2) = Faulty$ as $\#(\mathbf{s})$ satisfies both $R_1$ and $R'_2$, but it does not satisfy $R'_1$ and $R_2$. For this observed sequence, there exist two sequences $\sigma_1 = t_3t_2t_5t_7t_3$ and $\sigma_2 = t_2t_3t_5t_7t_3$ such that $\pi(\mathbf{s}) = t_3t_3$. These sequences contain (no) fault from type $T_f^2$ ($T_f^1$).

Finally, let us explore the case where the sequence $\mathbf{s} = t_3t_6$ is observed. In which case, we have $\Delta(\mathbf{s}, T_f^1) = Uncertain$ and $\Delta(\mathbf{s}, T_f^2) = Faulty$. The set of sequences having $\pi(\mathbf{s}) = t_3t_6$ is $\{t_3t_2t_5t_6, t_2t_3t_5t_6, t_3t_2t_5t_6t_1, t_2t_3t_5t_6t_1, t_3t_2t_5t_6t_2, t_2t_3t_5t_6t_2\}$. All of these sequences have a fault from type $T_f^2$ but only some of them have a fault from type $T_f^1$. Consequently, $\#(\mathbf{s})$ satisfies $R_1$, $R'_1$ and $R'_2$, but it does not satisfy $R_2$. With regards to the fault type $T_f^1$, the most recent diagnosis

state of the sequence $\mathbf{s} = t_3 t_6$ is $\Delta(t_3, T_f^1) = Uncertain$. Using Theorem 5.1 and Corollary 5.1, we have $\Delta(\mathbf{s}, T_f^1) = Uncertain$.

## 5.3 Chapter summary

This chapter has extended the work introduced in Chapter 4. The IFME method has been applied to address the fault diagnosis problem in DES modelled by Petri nets where no cycle of unobservable transitions exists. The notion of tracking the diagnosis history has been presented to overcome the problems of the state equation related to ordering. In other words, the state equation does not take into account the order of transitions in execution sequences of nets. This could represent a problem in the case where the order can affect the diagnosis results.

Furthermore, this chapter has given the complexity analysis of the proposed approach and commented on these complexities for special cases of Petri nets which are widely used. Two factors on which the complexity analysis relies are the size of the diagnoser produced offline and the time spent online for diagnosis computations. The analysis results state that a good trade-off between these two factors has been achieved. In particular, the results show that the size of the diagnoser is highly dependent on the number of unobservable transitions in the Petri nets and not on the state space size, as is the case with existing methods. Having obtained this diagnoser, online fault diagnosis can at most be performed in time polynomial in the size of the diagnoser.

# CHAPTER 6

# FAULT DIAGNOSIS IN LABELLED PETRI NETS

An extension of the work presented in Chapter 5 is covered in this chapter. We apply the IFME method to address the problem of fault diagnosis in DES modelled by *labelled* Petri nets (LPN). This type of Petri net has more expressive modelling power than *unlabelled* Petri nets (also called *free labelled* Petri nets). The type of languages generated by unlabelled Petri nets represents a subset of that generated by labelled Petri nets, where there is no a restriction of having unique labels associated to transitions [73]. Due to budget constraints, technology limitations or power consumption, not all events associated to observable transitions are monitored by their own sensors, i.e. the same sensor might be used to monitor more than one event (activity) in the system being analysed. When addressing a fault diagnosis problem in these Petri nets, the most interesting case arises when at least two transitions sharing the same label (sensor) are simultaneously enabled, but only one of them can fire. In which case, we cannot ensure which transition fired because we observe the same label. This adds another case of non-determinism and requires additional computations online to decide the diagnosis states. In effect, addressing the problem of fault diagnosis in labelled Petri nets based on the IFME method provides a more general solution to the problem with lower computational requirements compared to the existing solutions.

In this chapter, the problem of fault diagnosis in the context of labelled Petri nets is first described. Then, a definition of the diagnoser with regards to this Petri nets type is formed. In addition, the details of the IFME method to tackle this problem are given, with a formal proof of

its correctness. Finally, algorithms which implement the proposed approach and an illustrative example are presented.

## 6.1 Fault diagnosis in labelled Petri nets (LPN)

In this section, an expanded description of the problem of fault diagnosis in DES modelled by labelled Petri nets is given based on Sections 4.1 and 5.1. In this description, the formulation presented by Cabasino *et al.* [60] and Fanti *et al.* is adopted. Consider a labelled Petri net $(\mathcal{N}, M_0, \Sigma, \lambda)$, as defined in Section 3.1, where the net $(\mathcal{N}, M_0)$ is as described in Section 4.1. Let $\omega \in \Sigma^*$ denote an observed sequence of events (labels), where $\omega = \lambda(\mathbf{s})$ and $\mathbf{s} = \pi(\sigma)$ for a given sequence $\sigma \in T^*$.

Supposed that the labels captured by $\omega$ is the only information we receive when a sequence of observable transitions fires. This information is passed to the diagnoser to identify the diagnosis state as one of the following [23, 60] : 1) *Normal* state - when all sequences in $L(\mathcal{N}, M_0)$ whose projections $\mathbf{s}$ corresponding to $\omega$ have no fault transition from the set $T_f$; 2) $T_f^i - Certain$ state, obtained when all sequences in $L(\mathcal{N}, M_0)$ whose projections $\mathbf{s}$ corresponding to $\omega$ have a fault transition from the set $T_f^i$; and 3) $T_f^i - Uncertain$ state - this state occurs when there exist two sequences in $L(\mathcal{N}, M_0)$ with projection $\mathbf{s}$ corresponding to $\omega$, such that one of them has a fault transition from $T_f^i$, but the other has none.

In this chapter, the problem of fault diagnosis is addressed holding all assumptions of Section 5.1 and excluding the one of having unique labels associated to the observable transitions. That means the case of labelled Petri nets is considered, where the same label can be shared by more than one transition, taking into account that these transitions could simultaneously be enabled. In addition, a discussion about dealing with the non-pure Petri nets (having self-loops) will be covered in Section 6.2.6. Until then, we assume that the Petri nets being analysed are either pure or have been transformed into being pure as described in Section 3.1.

## 6.2 The IFME method for fault diagnosis in LPN

The main results obtained in this chapter are covered in this section. In effect, the key challenge to handle the case of labelled Petri nets concerns the nondeterminism resulting from the possibility of more than one transition sharing a label. In addition, it is possible that these transitions are enabled simultaneously. Hence, a sequence of observed events (labels) might correspond to different sequences of observable transitions. In this case, generating all possible sequences of observable transitions corresponding to this observed sequence is required. In general, not all generated sequences are legal sequences. In other words, not all generated sequences have a corresponding sequence in the language of the Petri net. In fact, the sets of inequalities $R$ and $R'$ created only based on the state equation are no longer able to identify the legal sequences. To this end, another set of inequalities is built based on both the *state equation* and the *enabling condition* of the Petri nets.

To formulate an IFME-based solution, we first introduce some definitions and notation which are necessary to establish the case of fault diagnosis in the case of labelled Petri nets. Then, we will explain how to obtain the set of inequalities characterising the legal sequences.

### 6.2.1 Definitions and notations

From the same concepts described in Section 5.2.1 to represent faults as inequalities, the following definition is introduced for use in determining the set $X^i(\omega)$ described below.

**Definition 6.1.** *The diagnosis labelling function*: a diagnosis labelling function $D : T_o^* \times 2^{T_f} \rightarrow \{N, F, FN\}$ is a mapping that associates to each sequence of observable transitions $\mathbf{s}$ with respect to the fault type $T_f^i$ (expressed by $\mathbf{c}_i$), $i \in \{1, \ldots, r\}$, one of the following diagnosis labels:

- $D(\mathbf{s}, T_f^i) = N$ if $\forall \sigma \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma) = \mathbf{s}$, $\#(\sigma) \vDash \mathbf{c}_i$ holds.
- $D(\mathbf{s}, T_f^i) = F$ if $\forall \sigma \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma) = \mathbf{s}$, $\#(\sigma) \vDash \neg\mathbf{c}_i$ holds.
- $D(\mathbf{s}, T_f^i) = FN$ if there exists two sequences $\sigma_1$, $\sigma_2 \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma_1) = \pi(\sigma_2) = \mathbf{s}$, but $\#(\sigma_1) \vDash \mathbf{c}_i$ and $\#(\sigma_2) \vDash \neg\mathbf{c}_i$ hold.

Two sets of sequences are defined in the following. The first set characterises the set of sequences in the language of $\mathscr{N}$ corresponding to an observed sequence of events $\omega$ as shown below:

$$\Gamma(\omega) = \{\sigma \in L(\mathscr{N}, M_0) \mid \mathbf{s} = \pi(\sigma), \omega = \lambda(\mathbf{s})\} \tag{6.1}$$

The second set consists of a number of pairs associated with a given sequence of observed events. Each pair captures the form (*observed sequence*, *diagnosis label*) expressed in the following definition:

**Definition 6.2.** Suppose that $(\mathscr{N}, M_0, \Sigma, \lambda)$ is a labelled Petri net. Given an observed sequence $\omega \in \Sigma^*$, we define a set of pairs associated with $\omega$ with respect to a fault type $T_f^i$ as:

$$X^i(\omega) = \{(\mathbf{s}, l) \mid \exists \sigma \in \Gamma(\omega), \mathbf{s} = \pi(\sigma), l = D(\mathbf{s}, T_f^i)\} \tag{6.2}$$

Note that the set $X^i(\omega) \neq \emptyset$ because $\omega$ corresponds to a firing sequence. In addition, every $\mathbf{s}$ such that $(\mathbf{s}, l) \in X^i(\omega)$ satisfies the following equation:

$$\sum_{t_j \in \tau(e)} \#(t_j, \mathbf{s}) = k \tag{6.3}$$

where $k$ represents the number of times a label $e$ appears in the observed sequence $\omega$. For example, assume that $\omega = e_1 e_1 e_2 e_1$, where $\tau(e_1) = \{t_1, t_2\}$ and $\tau(e_2) = \{t_3\}$, and the corresponding $\mathbf{s} = t_1 t_1 t_3 t_2$, then the number of $t_1$ plus the number of $t_2$ in $\mathbf{s}$ equals to the number of $e_1$ in $\omega$. Similarly, the number of $t_3$ in $\mathbf{s}$ equals the number of $e_2$ in $\omega$.

In the following, we extend Definition 5.1 described in Section 5.2.1 in order to deal with the case of labelled Petri nets. This definition is inspired by the definitions of the diagnoser presented by Cabasino *et al.* [60] and Fanti *et al.* [23].

**Definition 6.3.** A diagnoser is a function $\Delta : \Sigma^* \times 2^{T_f} \to \{NoFault, Faulty, Uncertain\}$ that associates with each observed sequence $\omega \in \Sigma^*$ with respect to the fault type $T_f^i, i \in \{1, \ldots, r\}$ one of the following diagnosis states:

- $\Delta(\omega, T_f^i) = NoFault$ if $\forall \sigma \in \Gamma(\omega), \#(\sigma) \models \mathbf{c}_i$ holds. This state indicates that there is no sequence having the same observation as $\omega$ contains a fault transition in $T_f^i$, i.e. no fault from set $T_f^i$ has occurred.

- $\Delta(\omega, T_f^i) = Faulty$ if $\forall \sigma \in \Gamma(\omega), \#(\sigma) \models \neg\mathbf{c}_i$ holds. This state is *Faulty* as all sequences having the same observation as $\omega$ contain a fault transition in $T_f^i$, i.e. a fault from set $T_f^i$ has certainly occurred during the observed sequence $\omega$.

- $\Delta(\omega, T_f^i) = Uncertain$ if there exists two sequences $\sigma_1$, $\sigma_2 \in \Gamma(\omega)$ such that $\#(\sigma_1) \models \mathbf{c}_i$ and $\#(\sigma_2) \models \neg\mathbf{c}_i$ hold. In this case, the behaviour of the system is ambiguous because both *NoFault* and *Faulty* states are possible during the observed sequence.

Note that if $\Delta(\omega, T_f^i) = NoFault$ for all $i = 1, \ldots, r$, then we are certain that no fault from any type has occurred during the observed sequence $\omega$, i.e. a non-faulty state has arisen.

**Example 6.1.** To explain the ideas of fault diagnosis with respect to the extended diagnoser definition mentioned above, we present this example. Consider the labelled Petri net depicted in Figure. 6.1. In this net, the set of places is $P = \{p_1, \ldots, p_{12}\}$ and the initial marking is $M_0 = [100000000000]$. Also, the set of transitions is $T = \{t_1, \ldots, t_{14}\}$. In the figure, observable transitions are depicted by solid rectangles, while empty rectangles represent unobservable transitions. The labelling function $\lambda$ yields $\tau(\varepsilon) = \{t_3, t_4, t_5, t_6, t_{11}, t_{13}\}$, $\tau(a) = \{t_1\}$, $\tau(b) = \{t_2, t_7\}$, $\tau(c) = \{t_8, t_{10}, t_{14}\}$ and $\tau(d) = \{t_9, t_{12}\}$. Moreover, there are two fault types $T_f^1 = \{t_6\}$ and $T_f^2 = \{t_{11}\}$ labelled by $f_1$ and $f_2$ as shown in the figure. Thus, we have the constraints $\mathbf{c}_1 := x_6 \leq 0$ and $\mathbf{c}_2 := x_{11} \leq 0$ and their negations $\neg\mathbf{c}_1 := x_6 > 0$ and $\neg\mathbf{c}_2 := x_{11} > 0$, respectively. Note that in this Petri net, two transitions sharing the same label could be enabled simultaneously, e.g. the transitions $t_8$ and $t_{10}$.

If we suppose that the diagnoser observes no sequence ($\omega = \varepsilon$), then $\Gamma(\omega) = \{\varepsilon\}$. In which case, we are certain that neither a fault from type $T_f^1$ nor $T_f^2$ has occurred, i.e. $\Delta(\varepsilon, T_f^1) = \Delta(\varepsilon, T_f^2) = NoFault$. Similarly, we have the same certainty when $\omega = a$ because $\Gamma(\omega) = \{t_1\}$.

Assuming now that $\omega = ab$, then $\Gamma(\omega) = \{t_1t_2, t_1t_2t_3, t_1t_2t_3t_4, t_1t_2t_3t_4t_5, t_1t_2t_3t_6\}$. One of these sequences has the fault transition $t_6$, but the others have none. However, all sequences have

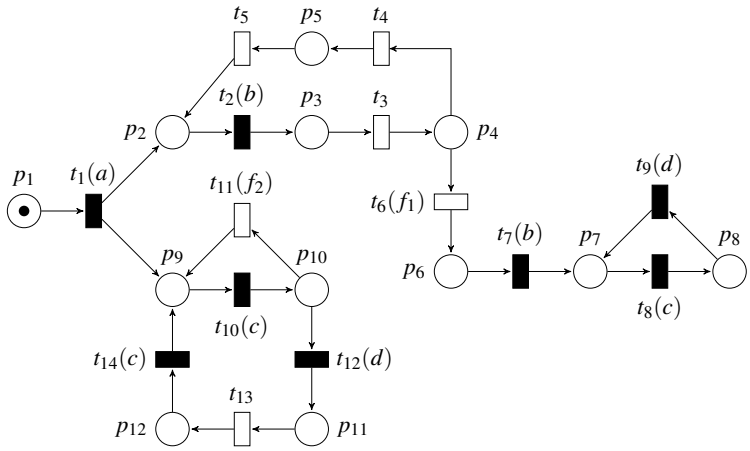



Figure 6.1: A labelled Petri net example

no fault transition $t_{11}$. Consequently, we are not certain about the diagnosis state regarding the fault type $T_f^1$ but we are certain that no fault from the type $T_f^2$ has occurred during observing $ab$. Hence, $\Delta(ab, T_f^1) = Uncertain$, but $\Delta(ab, T_f^2) = NoFault$.

When observing $\omega = acc$, a different diagnosis state is obtained. In effect, $\Gamma(\omega) = \{t_1t_{10}t_{11}t_{10}, t_1t_{10}t_{11}t_{10}t_{11}\}$. This ensures that a fault from type $T_f^2$ has occurred, while no fault from the type $T_f^1$ has occurred. Formally, $\Delta(acc, T_f^1) = NoFault$, but $\Delta(acc, T_f^2) = Faulty$.

### 6.2.2 Identification of the legal sequences

We start this section by recalling the results obtained by Dotoli *et al.* [10] in the case of free labelled Petri nets. These results showed that the set of all sequences, in a free labelled Petri net, which corresponds to an observed sequence of events can be characterised by a set of inequalities as expressed in the following proposition.

**Proposition 6.1.** [10] Given a free labelled Petri net $(\mathcal{N}, M_0)$ having no cycle of unobservable transitions and an observed sequence of events $\mathbf{s} \in T_o^*$. Then, there exists a sequence $\sigma = \sigma_1 t_1 \dots \sigma_h t_h$ such that $M_0 \overset{\sigma_1 t_1}{\rightarrow} M_1 \rightarrow \cdots \rightarrow M_{h-1} \overset{\sigma_h t_h}{\rightarrow} M_h$ and $\mathbf{s} = t_1 \dots t_h$ for $\sigma_1, \dots, \sigma_h \in T_u^*$ and $t_1, \dots, t_h \in T_o$ if and only if there exists a solution $\#(\sigma_1), \dots, \#(\sigma_h)$ to the following set of

inequalities:

$$\mathscr{S} = \begin{cases} A_u \cdot \#(\sigma_1) \geq pre(.,t_1) - M_0 & (1) \\ A_u \cdot (\#(\sigma_1) + \#(\sigma_2)) \geq pre(.,t_2) - M_0 - A \cdot \mathbf{u}_1 & (2) \\ \quad \vdots & \\ A_u \cdot (\#(\sigma_1) + \cdots + \#(\sigma_h)) \geq pre(.,t_h) - M_0 - A \cdot (\mathbf{u}_1 + \cdots + \mathbf{u}_{h-1}) & (h) \end{cases}$$

where $A_u$ is the restriction of $A$ on the unobservable transitions and $\mathbf{u}_i$ is the firing vector of $t_i$ for $i = 1, \ldots, h-1$. From Proposition 6.1, we can infer that if the set of inequalities $\mathscr{S}$ does not have a solution with respect to $\mathbf{s} = t_1 \ldots t_h$, then there does not exist a corresponding sequence $\sigma \in L(\mathscr{N}, M_0)$ such that $\sigma = \sigma_1 t_1 \ldots \sigma_h t_h$. The set of inequalities in $\mathscr{S}$ can also be rewritten as:

$$\mathscr{S}' = \begin{cases} -A_u \cdot \#(\sigma_1) + pre(.,t_1) \leq M_0 & (1) \\ -A_u \cdot (\#(\sigma_1) + \#(\sigma_2)) - A \cdot \mathbf{u}_1 + pre(.,t_2) \leq M_0 & (2) \\ \quad \vdots & \\ -A_u \cdot (\#(\sigma_1) + \cdots + \#(\sigma_h)) - A \cdot (\mathbf{u}_1 + \cdots + \mathbf{u}_{h-1}) + pre(.,t_h) \leq M_0 & (h) \end{cases}$$

where each subset $\mathscr{S}'_i,\ i = 1, \ldots, h$, of inequalities in $\mathscr{S}'$ can be expressed by the following general form, given a subsequence of transitions $\sigma_1 t_1 \ldots \sigma_i t_i$:

$$E := (-A \cdot \mathbf{x}) + \mathbf{y} \leq M_0 \tag{6.4}$$

where the vector $\mathbf{y} = pre(.,t)$ for $t \in \{t_1, \ldots, t_h\}$ and $\mathbf{x} = (x_1, \ldots, x_n)$ is such that $x_i$ represents the number of firing $t_i \in T$. Then, any solution to (6.4) is a solution to the subset $\mathscr{S}'_i$ and vice versa.

**Proposition 6.2.** Suppose that $(\mathscr{N}, M_0, \Sigma, \lambda)$ is a labelled Petri net satisfying the assumptions (a)-(e) listed in Section 6.1. Also, assume that $E$ is the set of inequalities of (6.4) in variables $x_i$ and $y_j$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. The variable $x_i$ corresponds to the number of firings of the transition $t_i$ and $y_j = pre(p_j, t)$ for a given transition $t \in T_o$. Applying IFME to the set $E$, in order to eliminate the variables corresponding to the unobservable transitions, results in the set $E'$. Then, for any given sequence of observable transitions $\mathbf{s} = t_1 \ldots t_h$, there exists a corresponding

sequence $\sigma = \sigma_1 t_1 \dots \sigma_h t_h$ in $\mathscr{N}$ such that $M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t_1} \cdots \xrightarrow{\sigma_h} M_h \xrightarrow{t_h}$ if there exists a vector $v = (\alpha_1, \dots, \alpha_k, pre(p_1,t), \dots, pre(p_m,t)) \vDash E'$, where $\alpha_i = \#(t_i, \mathbf{s}')$, $\mathbf{s}' = t_1 \dots t_{h-1}$ and $k = |T_o|$.

*Proof.* We prove this by the induction on the length of $\mathbf{s}$, denoted by $|\mathbf{s}|$.

*Base case*: assume that $|\mathbf{s}| = 1$. If $(\alpha_1, \dots, \alpha_k, pre(p_1,t_1), \dots, pre(p_m,t_1)) \vDash E'$, where $\alpha_i = 0$, then there exists a solution $v' = (\alpha_1, \dots, \alpha_k, \alpha_{k+1}, \dots, \alpha_n, pre(p_1,t_1), \dots, pre(p_m,t_1)) \vDash E$ by Theorem 3.1. Assume that $v = (\alpha_1, \dots, \alpha_n)$, then the subnet $\mathscr{N}_v$ has only unobservable transitions. Since $\mathscr{N}_v$ is free cycle by the assumption, there exits a sequence $\sigma_1 \in T_u^*$ such that $M_0 \xrightarrow{\sigma_1} M_1$ and $\#(\sigma_1) = v$ by Lemma 5.1. As a result, we have a sequence $\sigma_1 t_1$ such that $M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t_1}$ for $\mathbf{s} = t_1$. This proves the case.

*Induction step*: we assume that the result holds for all $\mathbf{s}$ with $|\mathbf{s}| < h$ (Induction hypothesis). Then, we prove that the result holds for $|\mathbf{s}| = h$. Hence, for $\mathbf{s}' = t_1 \dots t_{h-1}$ there exists a sequence $\sigma' = \sigma_1 t_1 \dots \sigma_{h-1} t_{h-1}$ such that $M_0 \xrightarrow{\sigma_1} M_1 \xrightarrow{t_1} \cdots \xrightarrow{\sigma_{h-1}} M_{h-1} \xrightarrow{t_{h-1}}$. If the transition $t_{h-1}$ fires, we obtain the marking $M_h = M_{h-1} + A \cdot \#(\sigma') \geq \vec{\mathbf{0}}$. If we have $\mathbf{s} = t_1 \dots t_h$ such that $(\alpha_1, \dots, \alpha_k, pre(p_1,t_h), \dots, pre(p_m,t_h)) \vDash E'$, then there exits a solution $v' = (\alpha_1, \dots, \alpha_k, \alpha_{k+1}, \dots, \alpha_n, pre(p_1,t_h), \dots, pre(p_m,t_h)) \vDash E$ by Theorem 3.1. Assume that $v = (\alpha_1, \dots, \alpha_k, \alpha_{k+1}, \dots, \alpha_n)$ and $\mathbf{z} = v - \#(\sigma')$, $\mathbf{z} \in \mathbb{N}^n$, then $M_{h+1} = M_h + A\mathbf{z} \geq \vec{\mathbf{0}}$. Since the subnet $\mathscr{N}_\mathbf{z}$ has only unobservable transitions and it is cycle free, then there exits a sequence $\sigma_h$ such that $M_h \xrightarrow{\sigma_h} M_{h+1}$ with $\#(\sigma_h) = \mathbf{z}$. Further, since $v' = (\alpha_1, \dots, \alpha_k, \alpha_{k+1}, \dots, \alpha_n, pre(p_1,t_h), \dots, pre(p_m,t_h)) \vDash E$, then $M_{h+1} \xrightarrow{t_h}$. Consequently, there exists a sequence $\sigma = \sigma_1 t_1 \dots \sigma_h t_h$ in $\mathscr{N}$ such that $\mathbf{s} = t_1 \dots t_h$. This also proves this case.

□

The following example explains how Proposition 6.2 can be applied to decide whether a sequence of observable transitions has at least one corresponding sequence in a labelled Petri net.

**Example 6.2.** Consider the labelled Petri net of Figure. 6.1. The associated set of inequalities $E$ is shown in Table 6.1. Applying the IFME to $E$ eliminates all variables corresponding to unobservable transitions, i.e. the set of variables $\{x_3, x_4, x_5, x_6, x_{11}, x_{13}\}$. The resulting set of inequalities $E'$ is in the set of variables $\{x_1, x_2, x_7, x_8, x_9, x_{10}, x_{12}, x_{14}\}$ plus the set of variables

Table 6.1: The sets of inequalities $E$ and $E'$ of the labelled Petri net in Figure. 6.1

| No. | $E$ | $E'$ |
|---|---|---|
| 1 | $x_1+y_1 \leq 1$ | $x_1+y_1 \leq 1$ |
| 2 | $-x_1+x_2-x_5+y_2 \leq 0$ | $-x_2+y_3 \leq 0$ |
| 3 | $-x_2+x_3+y_3 \leq 0$ | $-x_{12}+y_{11} \leq 0$ |
| 4 | $-x_3+x_4+x_6+y_4 \leq 0$ | $-x_8+x_9+y_8 \leq 0$ |
| 5 | $-x_4+x_5+y_5 \leq 0$ | $-x_2+y_3+y_4 \leq 0$ |
| 6 | $-x_6+x_7+y_6 \leq 0$ | $-x_{10}+x_{12}+y_{10} \leq 0$ |
| 7 | $-x_7+x_8-x_9+y_7 \leq 0$ | $-x_7+x_8-x_9+y_7 \leq 0$ |
| 8 | $-x_8+x_9+y_8 \leq 0$ | $-x_2+y_3+y_4+y_5 \leq 0$ |
| 9 | $-x_1+x_{10}-x_{11}-x_{14}+y_9 \leq 0$ | $-x_{12}+x_{14}+y_{11}+y_{12} \leq 0$ |
| 10 | $-x_{10}+x_{11}+x_{12}+y_{10} \leq 0$ | $-x_2+x_7+y_3+y_4+y_6 \leq 0$ |
| 11 | $-x_{12}+x_{13}+y_{11} \leq 0$ | $-x_1+y_2+y_3+y_4+y_5 \leq 0$ |
| 12 | $-x_{13}+x_{14}+y_{12} \leq 0$ | $-x_1+x_{12}-x_{14}+y_9+y_{10} \leq 0$ |
| 13 | $-x_i \leq 0 \mid_{1\leq i\leq 14}$ | $-x_2+x_7+y_3+y_4+y_5+y_6 \leq 0$ |
| 14 | $-y_j \leq 0 \mid_{1\leq j\leq 12}$ | $-x_1+x_7+y_2+y_3+y_4+y_5+y_6 \leq 0$ |

$\{y_j \mid 1 \leq j \leq 12\}$, see Table 6.1. The set of inequalities $E'$ is used to characterise the legal sequences.

Assume that we have a sequence of observable transitions $\mathbf{s} = t_1t_2$. To check whether this sequence has at least one corresponding sequence in $L(\mathcal{N}, M_0)$, we first determine the values of variables $y_2 = 1$ and $y_j = 0 \mid_{j\in\{1,\ldots,m\}\setminus\{2\}}$ based on the last transition in this sequence, i.e. $t_2$. Secondly, the values of the remaining variables are determined by counting the number of occurrences of their associated transitions in $\mathbf{s}$; in this case we only have one transition which means that $x_1 = 1, x_2 = 0, x_7 = 0, x_8 = 0, x_9 = 0, x_{10} = 0, x_{12} = 0$ and $x_{14} = 0$. Then, substituting these values into the set of inequalities $E'$, we can determine if $E'$ is satisfied. According to this example, these values satisfy $E'$, i.e. there exists a corresponding sequence in the net. On the other hand, the sequence $t_1t_7$ does not satisfy $E'$ implying that there does not exist a corresponding sequence.

### 6.2.3 Outline of the proposed approach

If we suppose that $(\mathcal{N}, M_0, \Sigma, \lambda)$ is a labelled Petri net with an initial marking $M_0$ and labelling function $\lambda$; without any loss of generality, we can assume that we have renamed the transitions

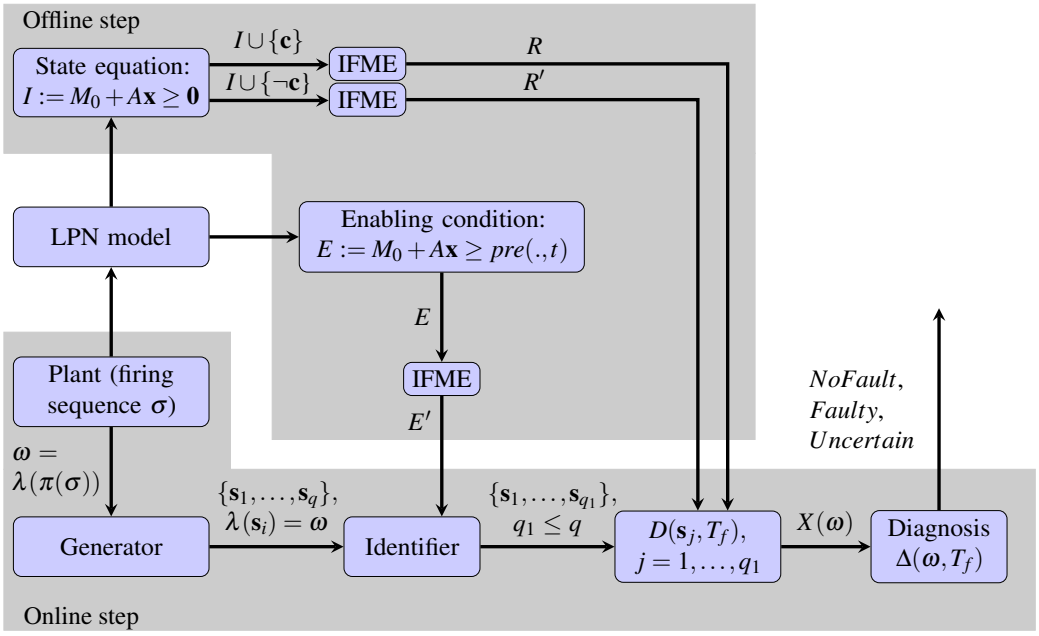



Figure 6.2: Sketch of the proposed approach in the case of labelled Petri nets

of $\mathcal{N}$ such that the first $k$ transitions are observable, i.e. $T_o = \{t_1, t_2, \ldots, t_k\}$ with their labels from the set $\Sigma$. The remaining transitions are unobservable, i.e. $T_u = \{t_{k+1}, t_{k+2}, \ldots, t_n\}$.

We can further suppose that the set of fault transitions in $\mathcal{N}$ is $T_f \subseteq T_u$ and all faults are of the same type. We introduce variables $x_1, x_2, \ldots, x_n$ representing the number of firings of $t_1, t_2, \ldots, t_n$, respectively. Suppose that $-A\mathbf{x} \leq M_0$ plus $\mathbf{x} \geq \vec{\mathbf{0}}$ represents the set of inequalities created from the state equation, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, see Lemma 4.1. We further assume that $\mathbf{c}$ is the inequality $\sum_{t_j \in T_f} x_j \leq 0$ and $\neg\mathbf{c}$ is the negation of $\mathbf{c}$, i.e. the inequality $\sum_{t_j \in T_f} x_j > 0$. For each firing sequence $\sigma$ of $(\mathcal{N}, M_0)$, if $\sigma$ contains a fault from $T_f$, then $\mathbf{x} = \#(\sigma)$ satisfies $\neg\mathbf{c}$. Conversely, for a firing sequence $\sigma$, if $\mathbf{x}$ satisfies $\mathbf{c}$, then $\sigma$ has no fault transition.

In Figure. 6.2, the outline of the proposed approach in this chapter is depicted. This outline is captured as a two-step process. The offline step results in three sets of inequalities, namely $E'$, $R$ and $R'$ which are obtained as follows. From a Petri net model, we first obtain a set of inequalities $I := \{-A\mathbf{x} \leq M_0\} \cup \{\mathbf{x} \geq \vec{\mathbf{0}}\}$ expressing the state equation. From this set, we further create two sets $I \cup \{\mathbf{c}\}$ and $I \cup \{\neg\mathbf{c}\}$. Secondly, another set of inequalities $E$ representing the enabling condition in the Petri nets is obtained. Then, applying the IFME method simultaneously to the sets $I \cup \{\mathbf{c}\}$, $I \cup \{\neg\mathbf{c}\}$ and $E$, three reduced sets $R$, $R'$ and $E'$ are respectively created by

eliminating every variable corresponding to a transition in the set $T_u$.

These resulting sets are employed during the online step to diagnose faults. In particular, the set $E'$ is used by the $Identifier$ to recognise the legal sequences. These sequences are produced by the $Generator$ and they correspond to all possible sequences of observable transitions corresponding to the observed sequence of events $\omega$. However, not all of these produced sequences have corresponding sequences in $\mathcal{N}$. Thus, the identified (legal) sequences are sent to the function $D(.)$ in which the sets $R$ and $R'$ are employed to build the set $X(.)$ used to diagnose faults.

The notion of creating these reduced sets can be extended to the case where there are multiple fault types each of which represents multiple faults. This can be achieved by producing a separated pair of sets of inequalities $(R_i, R'_i)$ for each fault type $T_f^i$. In particular, to create a set of inequalities for a given fault type, the transitions representing faults in the other fault types are considered as normal unobservable transitions. We say that a fault of type $T_f^i,\ i = 1,2,\ldots,r$, occurs if and only if at least one fault transition $t \in T_f^i$ fires. Note that one set of inequalities $E'$ is required to identify the legal sequences for all of those pairs.

In the following, we present theorems capturing the details of computing a diagnosis state upon observing a sequence of events $\omega$. We first introduce the following definition.

**Definition 6.4.** *The most recent diagnosis label*: suppose that $\mathbf{s} = \mathbf{s}'t$ is a sequence of observable events, where $\mathbf{s}' \in T_o^*$ and $t \in T_o$, then the most recent diagnosis label allocated by $D$ of $\mathbf{s}$ is $D(\mathbf{s}', T_f^i),\ i = 1,\ldots,r$.

Note that the most recent diagnosis label of the empty string $\varepsilon$ allocated by $D$ is $N$ because we assume that the system starts from a non-faulty state. In addition, the sequence labelled $F$ has this label with all of its continuation sequences.

**Theorem 6.1.** Assume that $(\mathcal{N}, M_0, \Sigma, \lambda)$ is a labelled Petri net satisfying the assumptions in Section 6.1. Suppose that $I$ is the set of inequalities $-A\mathbf{x} \leq M_0$ plus $\mathbf{x} \geq \vec{\mathbf{0}}$ created from the state equation of $\mathcal{N}$. Assume that $T = T_o \cup T_u$, $T_o = \{t_1,\ldots,t_k\}$, $T_u = \{t_{k+1},\ldots,t_n\}$, $T_f = T_f^1 \cup \cdots \cup T_f^r$ and $T_f \subseteq T_u$. The vector of variables $x_1,\ldots,x_n$ corresponds to the number of firings of the

transitions $t_1,\ldots,t_n$. Assume also that $\mathbf{c}_i$ is the inequality $\sum_{t_j\in T_f^i} x_j \leq 0$ and $\neg\mathbf{c}_i := \sum_{t_j\in T_f^i} x_j > 0$ is its negation. For every $i\in\{1,\ldots,r\}$, suppose that the set of inequalities $R_i$ and $R_i'$ are respectively produced from applying the IFME to both $I\cup\{\mathbf{c}_i\}$ and $I\cup\{\neg\mathbf{c}_i\}$ to eliminate all variables corresponding to transitions in $T_u$. Then, for any given sequence of observable transitions $\mathbf{s}=\mathbf{s}'t=\pi(\sigma)$, $\mathbf{s}'\in T_o^*$ and $t\in T_o$ such that there exists $\sigma\in T^*$ and $M_0\xrightarrow{\sigma}M$, $D(\mathbf{s},T_f^i)$ is determined as follows:

$$\phi(\mathbf{s},T_f^i)=\begin{cases} N & \text{if } (\#(\mathbf{s})\not\models R_i') \\ F & \text{if } (\#(\mathbf{s})\not\models R_i) \\ & \vee((\#(\mathbf{s})\models R_i)\wedge(\#(\mathbf{s})\models R_i') \\ & \wedge(D(\mathbf{s}',T_f^i)=F)) \\ FN & \text{if } (\#(\mathbf{s})\models R_i)\wedge(\#(\mathbf{s})\models R_i') \\ & \wedge((D(\mathbf{s}',T_f^i)=N)\vee(D(\mathbf{s}',T_f^i)=FN)) \\ Impossible & \text{if } (\#(\mathbf{s})\not\models R_i)\wedge(\#(\mathbf{s})\not\models R_i') \end{cases}$$

*Proof.* This proof is presented for one fault type $T_f^i$, but to obtain a complete proof we only need to repeat it for every single fault type. In the following, we assume that $\#(\mathbf{s})=(\alpha_1,\ldots,\alpha_k)$.

**Proof of $D(\mathbf{s},T_f^i)=N$:** by contradiction, assume that $\#(\mathbf{s})\not\models R_i'$, but $D(\mathbf{s},T_f^i)$ is not $N$. If $\#(\mathbf{s})\not\models R_i'$, then for every valuation $(\alpha_{k+1},\ldots,\alpha_n)$ of $(x_{k+1},\ldots,x_n)$ such that $\nu=(\alpha_1,\ldots,\alpha_k,\alpha_{k+1},\ldots,\alpha_n)$, $\nu\not\models I\cup\{\neg\mathbf{c}\}$ by Theorem 3.1. Since $\sigma$ is a sequence in $\mathcal{N}$, then $\nu\models I$, see Lemma 4.1. Thus, $\nu\not\models\neg\mathbf{c}_i$, i.e. $\nu\models\mathbf{c}$. As a result, $\forall\sigma'\in L(\mathcal{N},M_0)$ such that $\pi(\sigma')=\mathbf{s}$, $\#(\sigma')\models\mathbf{c}_i$ holds. Hence $D(\mathbf{s},T_f^i)$ is $N$, see Definition 6.1. This contradicts the assumption.

**Proof of $D(\mathbf{s},T_f^i)=F$:** here we have two cases to be proved.

**Case i:** if $\#(\mathbf{s})\not\models R_i$ holds. Using the same argument in **Proof of** $D(\mathbf{s},T_f^i)=N$ replacing $R_i'$ with $R_i$, we can prove this case.

**Case ii:** if $(\#(\mathbf{s})\models R_i)\wedge(\#(\mathbf{s})\models R_i')\wedge(D(\mathbf{s}',T_f^i)=F)$ holds. Since $D(\mathbf{s}',T_f^i)=F$ holds, i.e. the most recent diagnosis label is $F$, then for all sequences $\sigma'\in L(\mathcal{N},M_0)$ such that $\pi(\sigma')=\mathbf{s}'$, $\#(\sigma')\not\models\mathbf{c}_i$ holds. Also, since the label $F$ associated with $\mathbf{s}'$ propagates to all of its continuations,

then $\mathbf{s}$ also has the diagnosis label $F$.

**Proof of $D(\mathbf{s}, T_f^i) = FN$:** if we assume that $\mathbf{s} = \varepsilon$, then there exists one possible case for the most recent diagnosis label, particularly $N$, because we suppose that the system starts from a non-faulty state. Now let us prove the result in this case. If $\#(\mathbf{s}) \vDash R_i$, then there exists a valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$ and $\nu \vDash I \cup \{\mathbf{c}_i\}$ by Theorem 3.1. If $\nu \vDash I \cup \{\mathbf{c}_i\}$, then $\nu \vDash I$, i.e. $\nu$ satisfies $M' = M_0 + A\nu \geq \vec{\mathbf{0}}$. Since $\mathbf{s}$ has no observable transitions ($\mathbf{s} = \varepsilon$), then the subnet $\mathscr{N}_\nu$ has only unobservable transitions. Again, by the assumption that no cycle of unobservable transitions exists in $\mathscr{N}$, then $\mathscr{N}_\nu$ is cycle free. As a result, there exists $\sigma' \in T_\nu^*$ such that $M_0 \xrightarrow{\sigma'} M'$ and $\#(\sigma') = \nu$ by Lemma 5.1. Hence, the sequence $\sigma'$ satisfies $\mathbf{c}_i$. Likewise, we can prove that if $\#(\mathbf{s}) \vDash R_i'$, there exists another sequence satisfying $\neg\mathbf{c}_i$. Since there are two sequences having the same $\mathbf{s}$, but one of them satisfies $\mathbf{c}_i$ and the other satisfies $\neg\mathbf{c}_i$, then we have $\phi(\mathbf{s}, T_f^i) = FN$, see Definition 6.1.

Now, assume that $\mathbf{s} = \mathbf{s}'t$, $t \in T_o$ and $\mathbf{s}' \in T_o^*$. Then, there are two cases to be considered:

**Case i:** when the most recent label is $N$ ($D(\mathbf{s}', T_f^i) = N$). If $\#(\mathbf{s}) \vDash R_i$, then there exists a valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$ and $\nu \vDash I \cup \{\mathbf{c}_i\}$ by Theorem 3.1. If $\nu \vDash I \cup \{\mathbf{c}_i\}$, then $\nu \vDash I$, i.e. $M'' = M_0 + A\nu \geq \vec{\mathbf{0}}$. Since $D(\mathbf{s}', T_f^i) = N$, and $t$ is an observable transition, then we are certain that all sequences $\sigma' t$ such that $M_0 \xrightarrow{\sigma' t} M'$ and $\pi(\sigma') = \mathbf{s}'$ satisfy $\mathbf{c}_i$. Assuming $y = \nu - \#(\sigma' t),\ y \in \mathbb{N}^n$, then $M'' = M' + Ay \geq \vec{\mathbf{0}}$. Since the subnet $\mathscr{N}_y$ has only unobservable transitions, then $\mathscr{N}_y$ is cycle free. As a result, there exists $\sigma'' \in T_y^*$ such that $M' \xrightarrow{\sigma''} M''$ and $\#(\sigma'') = y$ by Lemma 5.1. Hence, the sequence $\sigma' t \sigma''$ with $\#(\sigma' t \sigma'') = \nu$ satisfies $\mathbf{c}_i$. Likewise, we can prove that if $\#(\mathbf{s}) \vDash R_i'$, there exists another sequence satisfying $\neg\mathbf{c}_i$. Consequently, since there are two sequences $\sigma_1, \sigma_2 \in L(\mathscr{N}, M_0)$ having the same $\mathbf{s}$, but one satisfies $\mathbf{c}_i$ and the other satisfies $\neg\mathbf{c}_i$, then we have $D(\mathbf{s}, T_f^i) = FN$, see Definition 6.1.

**Case ii:** when the most recent diagnosis label is $FN$ ($D(\mathbf{s}', T_f^i) = FN$). If $\#(\mathbf{s}) \vDash R_i$, then there exists a valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$ and $\nu \vDash I \cup \{\mathbf{c}\}$ by Theorem 3.1. If $\nu \vDash I \cup \{\mathbf{c}_i\}$, then $\nu \vDash I$, i.e. $M'' = M_0 + A\nu \geq \vec{\mathbf{0}}$. Since $D(\mathbf{s}', T_f^i) = FN$, i.e. the most recent diagnosis label is $FN$, and $t$ is an observable transition, then we still have the same label for any sequence $\sigma' t$ such that $M_0 \xrightarrow{\sigma' t} M'$ and $\pi(\sigma') = \mathbf{s}'$. Assuming

$y = \nu - \#(\sigma' t),\ y \in \mathbb{N}^n$, then $M'' = M' + Ay \geq \vec{\mathbf{0}}$. Since the subnet $\mathscr{N}_y$ has only unobservable transitions, then $\mathscr{N}_y$ is cycle free. As a result, there exists $\sigma'' \in T_y^*$ such that $M' \xrightarrow{\sigma''} M''$ and $\#(\sigma'') = \nu$ by Lemma 5.1. Hence, the sequence $\sigma' t \sigma''$ with $\#(\sigma' t \sigma'') = \nu$ satisfies $\mathbf{c}_i$. Similarly, we can prove that if $\#(\mathbf{s}) \vDash R_i'$, there exists another sequence satisfying $\neg\mathbf{c}_i$. Since there are two sequences having the same $\mathbf{s}$, but one satisfies $\mathbf{c}_i$ and the other does not, then we have $D(\mathbf{s}, T_f^i) = FN$, see Definition 6.1.

**Proof of the case** *Impossible*: assume that $\#(\mathbf{s}) \nvDash R_i$ and $\#(\mathbf{s}) \nvDash R_i'$, but this case is possible. If $\#(\mathbf{s}) \nvDash R_i$, then for every valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$, $\nu \nvDash I \cup \{\mathbf{c}_i\}$ by Theorem 3.1. Also, if $\#(\mathbf{s}) \nvDash R_i'$, then for every valuation $(\beta_{k+1}, \ldots, \beta_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \beta_{k+1}, \ldots, \beta_n)$, $\nu \nvDash I \cup \{\neg\mathbf{c}_i\}$ by Theorem 3.1. Rephrasing this statement, we can say that there exists a valuation $(\beta_{k+1}, \ldots, \beta_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \beta_{k+1}, \ldots, \beta_n)$ and $\nu \vDash I \cup \{\mathbf{c}_i\}$ taking into account that $\neg\mathbf{c}_i$ is the negation of $\mathbf{c}_i$ and $\sigma$ is a sequence of $\mathscr{N}$, i.e. $\#(\sigma) \vDash I$. Here we have contradictory statements. Hence, this case is an impossible case. This contradicts the assumption and completes the proof.

□

**Remark 6.1.** Similar to Remark 5.1, the proofs of the cases $N$ and $F$ in Theorem 6.1 can still be applied for the Petri nets which have cycles of unobservable transitions.

**Theorem 6.2.** Assume that $(\mathscr{N}, M_0, \Sigma, \lambda)$ is a labelled Petri net satisfying the assumptions in Section 6.1. Suppose that $I := \{-A\mathbf{x} \leq M_0\} \cup \{\mathbf{x} \geq \vec{\mathbf{0}}\}$ created from the state equation of $\mathscr{N}$; and $E$ is as defined in (6.4). Also, assume that $T = T_o \cup T_u$, $T_o = \{t_1, \ldots, t_k\}$, $T_u = \{t_{k+1}, \ldots, t_n\}$, $T_f = T_f^1 \cup \cdots \cup T_f^r$ and $T_f \subseteq T_u$. The vector of variables $x_1, \ldots, x_n$ corresponds to the number of firings of the transitions $t_1, \ldots, t_n$. Assume also that $\mathbf{c}_i$ is the inequality $\sum_{t_j \in T_f^i} x_j \leq 0$ and $\neg\mathbf{c}_i := \sum_{t_j \in T_f^i} x_j > 0$ is its negation. For every $i \in \{1, \ldots, r\}$, suppose that the set of inequalities $E'$, $R_i$ and $R_i'$ are respectively produced from applying IFME to $E$, $I \cup \{\mathbf{c}_i\}$ and $I \cup \{\neg\mathbf{c}_i\}$ to eliminate all variables corresponding to transitions in $T_u$. Then, for any given sequence of observed events $\omega \in \Sigma^*$, considering that the set $X^i(\omega)$ is such that each $(\mathbf{s}, l) \in X^i(\omega)$ is with $\mathbf{s} \vDash E'$, $\Delta(\omega, T_f^i)$ is

determined as follows:

$$\Delta(\omega, T_f^i) = \begin{cases} NoFault & \text{if } \forall(\mathbf{s},l) \in X^i(\omega), l = N \\ Faulty & \text{if } \forall(\mathbf{s},l) \in X^i(\omega), l = F \\ Uncertain & \text{if } (\exists(\mathbf{s},l) \in X^i(\omega) \mid l = FN) \\ & \vee((\nexists(\mathbf{s},l) \in X^i(\omega) \mid l = FN) \\ & \wedge(\exists(\mathbf{s}',l'),(\mathbf{s}'',l'') \in X^i(\omega) \mid l' \neq l'')) \end{cases}$$

*Proof.* **Proof of the case** *NoFault*: by Definition 6.2 and Theorem 6.1, if all $(\mathbf{s},l) \in X^i(\omega)$ are such that $l = N$, then every sequence $\sigma$ such that $\pi(\sigma) = \mathbf{s}$ satisfies $\mathbf{c}_i$, this implies that no fault has occurred during observing $\omega$.

**Proof of the case** *Faulty*: using the same argument in the proof of the case *NoFault* replacing $l = N$ with $l = F$, we can prove that if $\forall(\mathbf{s},l) \in X^i(\omega), l = F$, then the diagnosis state is *Faulty*.

**Proof of the case** *Uncertain*: we have two cases to be considered in this proof:

**Case i:** if $\exists(\mathbf{s},l) \in X^i(\omega)$ such that $l = FN$, then $\Delta(\omega, T_f^i) = Uncertain$. By Definition 6.2 and Theorem 6.1, having this condition held implies that there exists two sequences $\sigma, \sigma' \in L(\mathcal{N}, M_0)$ and $\mathbf{s}$ such that $(\mathbf{s},l) \in X^i(\omega)$, $\pi(\sigma) = \pi(\sigma') = \mathbf{s}$ and $\#(\sigma) \vDash \mathbf{c}_i$ but $\#(\sigma') \vDash \neg\mathbf{c}_i$. As a result, the diagnosis state cannot be ensured, see Definition 6.3.

**Case ii:** if $(\nexists(\mathbf{s},l) \in X^i \mid l = FN) \wedge (\exists(\mathbf{s}',l'),(\mathbf{s}'',l'') \in X^i \mid l' \neq l'')$, then $\Delta(\omega, T_f^i) = Uncertain$. Since the diagnosis labels of the sequences $\mathbf{s}'$ and $\mathbf{s}''$ are not $FN$ and they are different, then we necessarily have the diagnosis labels $N$ and $F$. Hence, based on Definition 6.2 and Theorem 6.1, there exists two sequences $\sigma_1, \sigma_2 \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma_1) = \mathbf{s}'$, $\pi(\sigma_2) = \mathbf{s}''$. Further, $\#(\sigma_1) \vDash \mathbf{c}_i$ but $\#(\sigma_2) \vDash \neg\mathbf{c}_i$. According to Definition 6.3, we have an *Uncertain* state.

□

### 6.2.4 Fault diagnosis algorithms: labelled Petri nets

In this section, algorithms are written for the proposed approach for fault diagnosis in labelled Petri nets. Algorithm 6.1 is similar to Algorithm 5.1 presented in Section 5.2.4. The only

**Algorithm 6.1** : build the diagnoser.

**Input:** A labelled Petri net $(\mathcal{N}, M_0, \Sigma, \lambda)$,
a set of unobservable transitions $T_u$,
fault types set $\{T_f^i | 1 \leq i \leq r\}$.
**Output:** A set of pairs $(R_i, R_i')_{i=1,2,\ldots,r}$ plus the set $E'$.

1: **Let** $I \leftarrow \{-A\mathbf{x} \leq M_0\} \cup \{-x_j \leq 0 \mid_{j=1,\ldots,n}\}$
2: **Let** $E \leftarrow -A\mathbf{x} + pre(.,t) \leq M_0$
3: **Let** $E' \leftarrow E$
4: **for all** $t_j$ such that $t_j \in T_u$ **do**
5: $E' \leftarrow IFME\_method(E', x_j)$
6: **end for**
7: **for all** $i$ such that $i \in \{1, 2, \ldots, r\}$ **do**
8: **Let** $\mathbf{c}_i \leftarrow \sum_{t_j \in T_f^i} x_j \leq 0$
9: **Let** $\neg\mathbf{c}_i \leftarrow \sum_{t_j \in T_f^i} -x_j \leq -1$
10: $R_i \leftarrow I \cup \{\mathbf{c}_i\}$
11: $R_i' \leftarrow I \cup \{\neg\mathbf{c}_i\}$
12: **for all** $t_j$ such that $t_j \in T_u$ **do**
13: $R_i \leftarrow IFME\_method(R_i, x_j)$
14: $R_i' \leftarrow IFME\_method(R_i', x_j)$
15: **end for**
16: **end for**

difference is in steps 2-6 in which a set of inequalities $E'$ is created. The output of this algorithm consists of sets of inequalities $E'$, $R_i$ and $R_i'$, $i = 1, \ldots, r$. These sets are employed by Algorithm 6.2 to decide diagnosis states, i.e. whether a fault has occurred or may have occurred. These states are determined based on the results obtained in Theorem 6.1 and Theorem 6.2.

The inputs of Algorithm 6.2 are the fault types set $T_f$ and $\tau(e) \forall e \in \Sigma$, in addition to sets of inequalities $E'$, $R_i$ and $R_i'$. The output of the algorithm is a diagnosis state from $\{NoFault, Faulty, Uncertain\}$ (see Definition 6.3). This algorithm starts by initialising $\omega'$ to the empty string $\varepsilon$ and $X^i(\omega')$ to the empty set $\emptyset$, for $i = 1, \ldots, r$. Then, in step 2 in particular, the algorithm enters into a loop to estimate the system state to check whether a fault has occurred. In step 3, the algorithm waits until a new event $e$ is observed and then adds it to the previous sequence $\omega'$, creating the sequence $\omega$. From step 5 to step 26, the algorithm builds the set $X^i(\omega)$ for $\omega$ with respect to each fault type $T_f^i$. First, the set of all sequences $\mathbf{s} \in T_o^*$ corresponding to $\omega$ in $\mathcal{N}$ is generated in steps 6-8. Then, in step 9, each generated sequence is checked to determine

**Algorithm 6.2** : online fault diagnosis.

**Input:** The fault types set $\{T_f^i | 1 \leq i \leq r\}$; $\tau(e), \forall e \in \Sigma$
and a set of pairs $(R_i, R_i')_{i=1,2,\ldots,r}$ plus $E'$ as defined in Algorithm 6.1.
**Output:** A diagnosis state $\{NoFault, Faulty, Uncertain\}$.

**Initialise** $\omega' = \varepsilon$, $X^i(\omega') = \emptyset$, $\forall i \in \{1, \ldots, r\}$
**loop**
  **if** a new event $e$ is observed **then**
    **Let** $\omega \leftarrow \omega' e$
    **Initialise** $X^i(\omega) \leftarrow \emptyset$
    **for all** $t \in \tau(e)$ **do**
      **for all** $\mathbf{s}' \in X^i(\omega')$ **do**
        $\mathbf{s} \leftarrow \mathbf{s}' t$
        **if** $\#(\mathbf{s}') \models E'$ with respect to $pre(.,t)$ **then**
          **for all** $i$ such that $i \in \{1, 2, \ldots, r\}$ **do**
            **if** $\#(\mathbf{s}) \not\models R_i'$ **then**
              $D(\mathbf{s}, T_f^i) \leftarrow N$
            **else if** $\#(\mathbf{s}) \not\models R_i$ **then**
              $D(\mathbf{s}, T_f^i) \leftarrow F$
            **else if** $\#(\mathbf{s}) \models R_i$ **and** $\#(\mathbf{s}) \models R_i'$ **then**
              **if** $D(\mathbf{s}', T_f^i) = F$ **then**
                $D(\mathbf{s}, T_f^i) \leftarrow F$
              **else if** $D(\mathbf{s}', T_f^i) = N$ **or** $D(\mathbf{s}', T_f^i) = FN$ **then**
                $D(\mathbf{s}, T_f^i) \leftarrow FN$
              **end if**
            **end if**
            $X^i(\omega) \leftarrow X^i(\omega) \cup \{(\mathbf{s}, D(\mathbf{s}, T_f^i))\}$
          **end for**
        **end if**
      **end for**
    **end for**
    **for all** $i$ such that $i \in \{1, 2, \ldots, r\}$ **do**
      **if** $\forall (\mathbf{s}, l) \in X^i(\omega)$, $l = N$ **then**
        $\Delta(\omega, T_f^i) \leftarrow NoFault$
      **else if** $\forall (\mathbf{s}, l) \in X^i(\omega)$, $l = F$ **then**
        $\Delta(\omega, T_f^i) \leftarrow Faulty$
      **else**
        $\Delta(\omega, T_f^i) \leftarrow Uncertain$
      **end if**
    **end for**
  **end if**
  $\omega' \leftarrow \omega$, $X^i(\omega') \leftarrow X^i(\omega)$
**end loop**

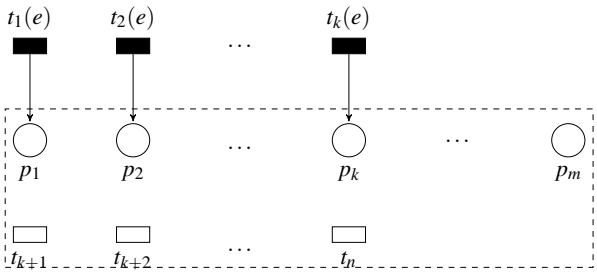



Figure 6.3: A general labelled Petri net example of the case where $|X^i(\omega')| = k^{n_1}$

whether it has a corresponding sequence in the Petri net. The function $\phi(\mathbf{s}, T_f^i)$ is computed in steps 10-21 for each legal $\mathbf{s}$ and each $T_f^i$, $1 \leq i \leq r$ based on applying Theorem 6.1. The pair $(\mathbf{s},\ \phi(\mathbf{s}, T_f^i))$ is added to the set $X^i(\omega)$ for the current observed sequence $\omega$ in step 22. After obtaining the whole set $X^i(\omega)$ the step of computing the diagnosis state starts at step 27. By this step, all labels $l$ associated with the pairs in the set $X^i(\omega)$ are tested to determine the state of the system when observing the sequence $\omega$, see Theorem 6.2.

Regarding the computational complexity, Algorithm 6.1 still has the same complexity as Algorithm 5.1 presented in Section 5.2.4; however, Algorithm 6.2 has a slightly different complexity from Algorithm 5.2. This complexity now relies on the number of observed events and the size of the diagnoser. To be precise, assume that $m_F$ is the the number of inequalities in $R_i \cup R_i'$ of the fault type $T_f^i$, then Algorithm 6.2 requires in the worst case $O(|X^i(\omega')| \cdot m_F)$ to decide the diagnosis state for each fault type $T_f^i$. Note that $|X^i(\omega')| \leq k^{n_1}$, where $k = |T_o|$ and $n_1$ is the length of the sequence $\omega'$.

In fact, the case where $|X^i(\omega')| = k^{n_1}$ arises in a very extreme example of the labelled Petri net. Consider, for example, the general labelled Petri net of Figure. 6.3 which represents this case where all observable transitions $t_1, \ldots, t_k$ share the same label $e$; in addition, all of these transitions are always enabled.

### 6.2.5 Illustrative example

Recalling the labelled Petri net of Figure. 6.1, since we have two fault types, two pairs of sets of inequalities are to be created to represent the diagnoser, in addition to one set representing the

Table 6.2: The sets of inequalities resulting from applying the IFME method in the illustrative example

| No. | $R_1$ | $R'_1$ | $R_2$ | $R'_2$ |
|---|---|---|---|---|
| 1 | $x_1 \leq 1$ | $x_1 \leq 1$ | $x_1 \leq 1$ | $x_1 \leq 1$ |
| 2 | $-x_8 + x_9 \leq 0$ | $-x_8 + x_9 \leq 0$ | $-x_8 + x_9 \leq 0$ | $-x_8 + x_9 \leq 0$ |
| 3 | $-x_2 + x_7 \leq 0$ | $-x_2 + x_7 \leq 0$ | $-x_2 + x_7 \leq 0$ | $-x_2 + x_7 \leq 0$ |
| 4 | $-x_1 + x_7 \leq 0$ | $-x_1 + x_7 \leq 0$ | $-x_1 + x_7 \leq 0$ | $-x_1 + x_7 \leq 0$ |
| 5 | $-x_7 + x_8 - x_9 \leq 0$ | $-x_7 + x_8 - x_9 \leq 0$ | $-x_7 + x_8 - x_9 \leq 0$ | $-x_7 + x_8 - x_9 \leq 0$ |
| 6 | $-x_1 + x_{12} - x_{14} \leq 0$ | $-x_1 + x_{12} - x_{14} \leq 0$ | $-x_1 + x_{12} - x_{14} \leq 0$ | $-x_1 + x_{12} - x_{14} \leq 0$ |
| 7 | $-x_{12} + x_{14} \leq 0$ | $-x_{12} + x_{14} \leq 0$ | $-x_{12} + x_{14} \leq 0$ | $-x_{12} + x_{14} \leq 0$ |
| 8 | $x_7 \leq 0$ | $-x_2 \leq -1$ | $-x_{10} + x_{12} \leq 0$ | $-x_{10} + x_{12} \leq -1$ |
| 9 | $-x_{10} + x_{12} \leq 0$ | $-x_1 \leq -1$ | $-x_1 + x_{10} - x_{14} \leq 0$ | |
| 10 | | $-x_{10} + x_{12} \leq 0$ | | |

enabling condition. We start by extending the set of inequalities $I$ created from the state equation by simultaneously adding the inequalities $\mathbf{c}_1 := -x_6 \leq 0$ and $\neg\mathbf{c}_1 := -x_6 \leq -1$. As a result, two extended sets of inequalities are obtained, namely $I \cup \{\mathbf{c}_1\}$ and $I \cup \{\neg\mathbf{c}_1\}$. Applying the IFME method to both sets results in the sets $R_1$ and $R'_1$ shown in Table 6.2. Now $R_1$ and $R'_1$ can be used to diagnosis faults from type $T_f^1$.

Similarly, two sets of inequalities $R_2$ and $R'_2$ are created to express faults from type $T_f^2$. Applying the IFME method to the sets $I \cup \{\mathbf{c}_2\}$ and $I \cup \{\neg\mathbf{c}_2\}$, where $\mathbf{c}_2 := -x_{11} \leq 0$ and $\neg\mathbf{c}_2 := -x_{11} \leq -1$, results in $R_2$ and $R'_2$, respectively as illustrated in Table 6.2. After applying the IFME method, all variables corresponding to unobservable transitions $T_u = \{t_3, t_4, t_5, t_6, t_{11}, t_{13}\}$ are eliminated. The sets of inequalities in Table 6.2 are in variables representing the observable transitions $T_o = \{t_1, t_2, t_7, t_8, t_9, t_{10}, t_{12}, t_{14}\}$. The sets $R_1$, $R'_1$, $R_2$ and $R'_2$ represent the diagnoser used for estimating the current state of the system for a given observed sequence of events $\omega$.

Table 6.3 shows in detail the process of estimating the diagnosis states given different observed sequences of events. Note also that the entries of the column entitled *Is* **s** *legal?* have two values: *Yes* and *No*. The value *Yes* implies that the possible sequence of observable transitions has at least one corresponding sequence in the net; whereas *No* refers to the opposite case. Thus, all possible sequences of observable transitions with the value *No* are ignored in following steps of processing.

By looking at the labelled Petri net in Figure. 6.1, observing no sequence ($\omega = \varepsilon$), yields $\Delta(\varepsilon, T_f^1) = \Delta(\varepsilon, T_f^2) = NoFault$. For $\varepsilon$, there exists only one possible sequence which is $\varepsilon$ itself.

Table 6.3: The whole process of estimating diagnosis states $\Delta(\omega, T_f^1)$ and $\Delta(\omega, T_f^2)$ of the observed sequence $\omega$

| No. | $\omega$ | **s** | Is **s** legal? | $\#(\mathbf{s}) \vDash R_1$? | $\#(\mathbf{s}) \vDash R_1'$? | $\#(\mathbf{s}) \vDash R_2$? | $\#(\mathbf{s}) \vDash R_2'$? | $D(\mathbf{s}, T_f^1)$ | $D(\mathbf{s}, T_f^2)$ | $X^1(\omega)$ | $X^2(\omega)$ | $\Delta(\omega, T_f^1)$ | $\Delta(\omega, T_f^2)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | $\varepsilon$ | $\varepsilon$ | Yes | Yes | No | Yes | No | $N$ | $N$ | $\{(\varepsilon, N)\}$ | $\{(\varepsilon, N)\}$ | *NoFault* | *NoFault* |
| 2 | $a$ | $t_1$ | Yes | Yes | No | Yes | No | $N$ | $N$ | $\{(t_1, N),$ | $\{(t_1, N),$ | *NoFault* | *NoFault* |
| 3 | $ab$ | $t_1t_2$ | Yes | Yes | Yes | Yes | No | $FN$ | $N$ | $\{(t_1t_2, FN),$ | $\{(t_1t_2, N),$ | *Uncertain* | *NoFault* |
| | | $t_1t_7$ | No | - | - | - | - | - | - | - | - | | |
| 4 | $abb$ | $t_1t_2t_2$ | Yes | Yes | Yes | Yes | No | $FN$ | $N$ | $\{(t_1t_2t_2, FN),$ | $\{(t_1t_2t_2, N),$ | *Uncertain* | *NoFault* |
| | | $t_1t_2t_7$ | Yes | No | Yes | Yes | No | $F$ | $N$ | $(t_1t_2t_7, F)\}$ | $(t_1t_2t_7, N)\}$ | | |
| 5 | $abbc$ | $t_1t_2t_2t_8$ | No | - | - | - | - | - | - | - | - | *Uncertain* | *Uncertain* |
| | | $t_1t_2t_2t_{10}$ | Yes | Yes | Yes | Yes | Yes | $FN$ | $N$ | $\{(t_1t_2t_2t_{10}, FN),$ | $\{(t_1t_2t_2t_{10}, N),$ | | |
| | | $t_1t_2t_2t_{14}$ | No | - | - | - | - | - | - | - | - | | |
| | | $t_1t_2t_7t_8$ | Yes | No | Yes | Yes | No | $F$ | $N$ | $(t_1t_2t_7t_8, F),$ | $(t_1t_2t_7t_8, N),$ | | |
| | | $t_1t_2t_7t_{10}$ | Yes | No | Yes | Yes | Yes | $F$ | $FN$ | $(t_1t_2t_7t_{10}, F)\}$ | $(t_1t_2t_7t_{10}, FN)\}$ | | |
| | | $t_1t_2t_7t_{14}$ | No | - | - | - | - | - | - | - | - | | |
| 6 | $ac$ | $t_1t_8$ | No | - | - | - | - | - | - | - | - | *NoFault* | *Uncertain* |
| | | $t_1t_{10}$ | Yes | Yes | No | Yes | Yes | $N$ | $FN$ | $\{(t_1t_{10}, N)\}$ | $\{(t_1t_{10}, FN)\}$ | | |
| | | $t_1t_{14}$ | No | - | - | - | - | - | - | - | - | | |
| 7 | $acc$ | $t_1t_{10}t_8$ | No | - | - | - | - | - | - | - | - | *NoFault* | *Faulty* |
| | | $t_1t_{10}t_{10}$ | Yes | Yes | No | No | Yes | $N$ | $F$ | $\{(t_1t_{10}t_{10}, N)\}$ | $\{(t_1t_{10}t_{10}, F)\}$ | | |
| | | $t_1t_{10}t_{14}$ | No | - | - | - | - | - | - | - | | | |
| 8 | $acd$ | $t_{10}t_9$ | No | - | - | - | - | - | - | - | - | *NoFault* | *NoFault* |
| | | $t_1t_{10}t_{12}$ | Yes | Yes | No | Yes | No | $N$ | $N$ | $\{(t_1t_{10}t_{12}, N)\}$ | $\{(t_1t_{10}t_{12}, N)\}$ | | |

As shown in the table, since $\#(\varepsilon) \vDash R_1$ holds, but $\#(\varepsilon) \vDash R_1'$ does not, then $D(\varepsilon, T_f^1)$ yields $N$, see Theorem 6.1. Similarly, since $\#(\varepsilon) \vDash R_2$ holds, but $\#(\varepsilon) \vDash R_2'$ does not, $D(\varepsilon, T_f^2)$ yields $N$. As a result, both sets $X^1(\varepsilon)$ and $X^2(\varepsilon)$ have only one element with the diagnosis label $N$. Also, the diagnosis state of both fault types $T_f^1$ and $T_f^2$ is computed as *NoFault*, see Theorem 6.2. Analogously, observing $\omega = a$ results in the same diagnosis state.

If we suppose that the diagnoser observes the sequence $\omega = abb$, then $\Delta(abb, T_f^1) = Uncertain$ and $\Delta(abb, T_f^2) = NoFault$. In which case, the sets $X^1(abb)$ and $X^2(abb)$ are not a singleton. Since one element of the set $X^1(abb)$ has the diagnosis label $FN$, we are not certain about the diagnosis state with respect to $T_f^1$. In contrast, all elements of the set $X^2(abb)$ have the diagnosis label $N$ which implies that we are certain that no fault of type $T_f^2$ has occurred during $abb$, see Theorem 6.2.

In Section 6.1 of this chapter, we assumed that the labelled Petri nets have no self-loops. However, our approach can be applied to address the more general case, i.e. labelled Petri nets with self-loops.

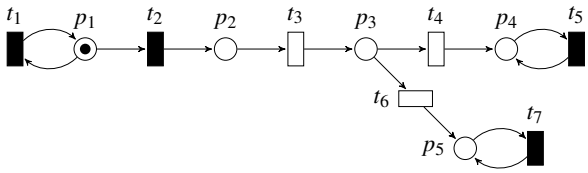



Figure 6.4: A Petri net with self-loops

### 6.2.6 Self-loops and the state equation

The state equation constraints, i.e. the set of inequalities $I$ formed in Lemma 4.1, can only describe the behaviour of *pure* Petri nets. In the case where there are self-loops, each entry of the incidence matrix $A$ has the value 0 of the corresponding self-loop transition (see Section 3.1). Consequently, the set of inequalities $I$ has no corresponding variables of such transitions. Hence, the state equation constraints must be extended to describe the behaviour of the Petri nets in general. In fact, this extension has been reported by Iordache and Antsaklis [96, 97]. Essentially, the idea is based on the fact that two results can be obtained when a sequence of transitions $\sigma$ fires at the initial marking $M_0$. First, the Parikh vector $\mathbf{x} = \#(\sigma)$ of the firing sequence satisfies $I := -A\mathbf{x} \leq M_0$. The second result is that a given transition $t$ is enabled if and only if $Q := -A(\mathbf{x} + \mathbf{u}_t) \leq M_0$ holds, where $\mathbf{u}_t$ is the firing vector of the transition $t$ (see Section 3.1). The set of inequalities $Q$ represents a general form by which the behaviour of Petri nets is completely expressed regardless of whether the Petri nets are *pure*.

If we assume that the given transition $t$ is on a self-loop, denoted by $pt$, then the enabling condition of $t$ depends only on the number of tokens in the place $p$, which represents its input and output. Thus, we are required to add one variable to every inequality associated with a place to which the self-loop transition connected. Each variable has the value 1 when the corresponding transition is the current firing transition; otherwise it has the value 0. Firing any self-loop transition $t$, after firing a sequence $\sigma \in T^*$ ensures that $Q := -A(\mathbf{x} + \mathbf{u}_t) \leq M_0$ holds, where $\mathbf{x} = \#(\sigma)$.

Consider the Petri net of Figure. 6.4, we have three self-loops: $p_1t_1$, $p_4t_5$ and $p_5t_7$. Since our concern is the self-loops, we shall construct the inequalities of $Q$ associated to the places on these self-loops. In which case, we have three inequalities: $q_1 + x_2 \leq 1$, $q_5 - x_4 \leq 0$ and

$q_7 - x_6 \leq 0$. The variables $q_1$, $q_5$ and $q_7$ correspond to transitions $t_1$, $t_5$ and $t_7$ respectively.

**Computing the function $D(.)$:** to obtain a diagnosis label of a sequence of observable transitions (see Definition 6.1) ending with a self-loop transition, we follow a two-phase procedure:

1. Offline phase: we create additional sets of inequalities $Q_i$ and $Q'_i$, for $i = 1, \ldots, r$, as follows. We first create $Q$ from the Petri net as described above. Then, we generate a pair $Q \cup \{\mathbf{c}_i\}$ and $Q \cup \{\mathbf{c}'_i\}$. Next, we apply the IFME method to this pair to create $Q_i$ and $Q'_i$ by eliminating all variables representing the unobservable transitions.

2. Online phase: in this phase the diagnosis label is obtained. Given a sequence of observable transitions $\mathbf{s} = \mathbf{s}'t_j$, where $t_j$ is a self-loop transition, $D(\mathbf{s}, T_f^i)$ is computed as described in Theorem 6.1. Replacing $R_i$ and $R'_i$ by $Q_i$ and $Q'_i$, respectively, $D(\mathbf{s}, T_f^i)$ is determined by testing the vector $(x_1, \ldots, x_k, q_1, \ldots, q_j, \ldots, q_h)$ against $Q_i$ and $Q'_i$, where $\#(\mathbf{s}') = (x_1, \ldots, x_k)$ and $h$ is the number of self-loop transitions.

Note that the results of this section can easily be proved using a direct application of Theorem 6.1. Also, from a practical complexity point of view, this relaxation requires adding $2 \times r$ sets of inequalities to the original sets of inequalities. However, the theoretical complexity is still the same.

## 6.3 Chapter summary

The problem of fault diagnosis in DES modelled by labelled Petri nets having no cycle of unobservable transitions has been covered in this chapter. By extending the results introduced in Chapter 5, we have shown that the IFME method can be used to diagnose faults in such nets. The firing of the observable transitions in these Petri nets cannot be ensured, especially in the case where there exists more than one transition which shares the same label, simultaneously enabled.

Thus, we address this case by considering all possible sequences of observable transitions corresponding to the observed sequence of events.

These sequences are generated online when observing a sequence of events. Then, all generated sequences having no corresponding sequences in a labelled Petri net are ignored. This can be performed using a single additional set of inequalities which is built based on the state equation and the enabling condition. Then, the sets of inequalities $R_i$ and $R'_i$, $i = 1, \ldots, r$, are used to diagnose faults.

# CHAPTER 7

# DIAGNOSIS OF VIOLATIONS OF CONSTRAINTS IN PETRI NETS

As pointed out in the previous chapters, fault diagnosis in partially-observed discrete event systems requires modelling faults as unobservable events within the model of the system (plant). Representing faults as events is not always realistic. For example, some classes of fault are in the form of violations of constraints such as service-level agreement (SLA) and Quality of Service (QoS). To model such faults as events, we need to modify the plant model which is not always acceptable. Firstly, this may make the models large by adding more transitions and places. Secondly, adding extra transitions and places is not always preferable from engineer's perspective as every modification of the constraint will modify the model of the plant.

In this chapter, the obtained results in the case where faults represent events in the systems are applied to the case where the faults are NOT captured as events in the model of the system, instead they represent violations of constraints. These constraints and their violations can be written as inequalities in which there are variables corresponding to unobservable transitions, i.e. the transitions expressed in these inequalities are partially-observed which represents the most interesting case of the problem. Then, we show that starting from a Petri net two sets of inequalities (diagnoser) are obtained. These sets are used to judge whether an observed sequence may satisfy (violate) these inequalities (constraints). To the best of our knowledge, the problem of diagnosis of violations of constraints under partial observation is first formulated and addressed in this thesis, where it represents a generalisation of the problem of fault diagnosis.

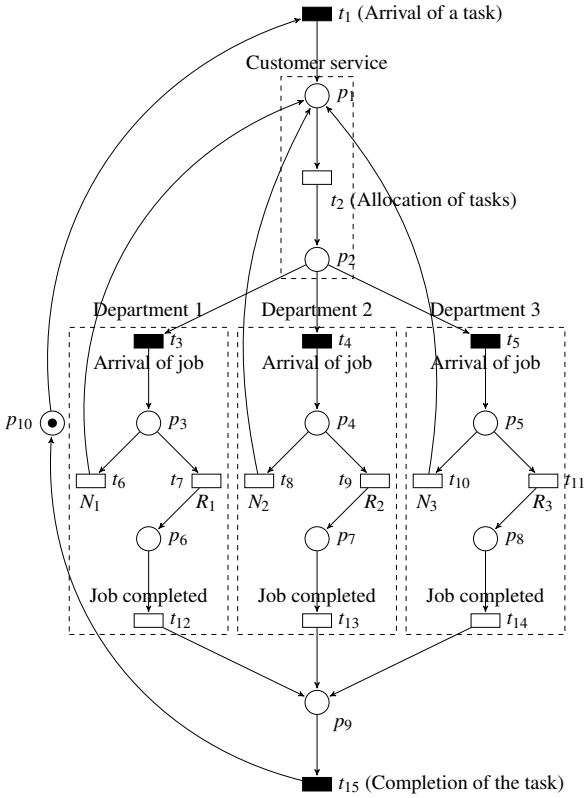



Figure 7.1: A problem of the Resolve System

## 7.1 Faults in the form of violations of constraints

Some faults represent no events in the plant of the system, instead they are a form of violations of constraints. The SLA and QoS violations are examples of such faults. Many SLA and QoS statements are defined to restrict the SLA and QoS such as the error rate, the percentage of service availability and the ratio of message loss in communication channels. These statements are termed as constraints within an agreement between providers and customers. The violations of these constraints can be seen as faults which implies that the provided services are going below the acceptable level according to the agreement.

### 7.1.1 A running example

To describe the problem that has motivated the work in this chapter, we shall make use of a simplified business process used within a typical telecommunication company [32, 98]. Suppose the scenario that a domestic customer telephones to report a malfunction such as the broadband

connection being slow. We refer to such problems and malfunctions as "tasks" or "jobs". The following example describes a simplified business process from the arrival of the job to its completion.

In the Petri net of Figure. 7.1, when the tasks arrive (firing of $t_1$), depending on the nature of the problem which is reported, every task is allocated to one of three departments. Within each department, there are a few large and complex workflows which we have simplified to two cases. Either the problem is resolved (transitions labelled $R_1, R_2, R_3$) or the engineers discover that the allocated jobs cannot be resolved (transitions labelled $N_1, N_2, N_3$) within their department. This would be a case of wrong allocation of jobs and can arise from a multitude of reasons, among them wrong information from the customers or wrong assignment of jobs or the case that one fault triggers another. In the case that the job is resolved, the department declares the job completed by firing of $t_{12}$, $t_{13}$ or $t_{14}$, which ultimately results in the firing of $t_{15}$ marking the completion of the (overall) task. In the case that a department is not able to complete the job (firing of $t_6$, $t_8$ or $t_{10}$), further investigation is required. As a result, a token is placed in $p_1$ so that the job is reallocated by the customer service department.

We assume that transitions $t_1$ and $t_{15}$, which mark arrival and completion of jobs, are observable. In addition, transitions that mark the arrival of the jobs in each department ($t_3$, $t_4$ and $t_5$) are also observable, as they are used by the department to inform the customers of the progress of the job. For example, if a customer is accessing through a browser to make an online report, they are informed that the relevant department will deal with the problem. Note that the observable transitions in Figure. 7.1 are depicted by solid rectangles, while empty rectangles represent unobservable transitions.

In the above example, firing of $t_6$, $t_8$ or $t_{10}$ results in a repetition of a chain of activities that indicates a wrong allocation of jobs to the departments. Since the activities are repeated, the job is not completed right first time. In this case, we say a right-first time (RFT) fault has occurred. The RFT faults are becoming increasingly important in the Telecom industry [32, 98]. Occurrences of RFT faults may result in unhappy customers; they increase the cost of resolving the problems and may entail financial penalties. As a result, the development of methods to

discover RFT faults so that remedial actions can be adopted is essential. In addition, in large organisations such methods must be automated to cope with large systems.

In Figure. 7.1, transition $t_2$ marks the allocation of jobs and transition $t_{15}$ marks the completion of these jobs. Ideally, to ensure no RFT fault occurs, we would wish that every allocated job is completed. In other words, for each execution sequence $\sigma$ of the Petri net, the following equation holds:

$$\#(t_2, \sigma) = \#(t_{15}, \sigma) \tag{7.1}$$

If (7.1) holds, we have no RFT fault. However, it is often not possible to completely eliminate the RFT fault. As a result, the management sets SLA such as the number of faults should be below a value $\delta \geq 0$ to specify acceptable levels of fault. Then, an SLA is satisfied if and only if for each execution sequence, (7.2) is evaluated to *True*.

$$\#(t_2, \sigma) - \#(t_{15}, \sigma) \leq \delta \tag{7.2}$$

The Petri net $\mathscr{N}$ of Figure. 7.1 represents a model of a system and (7.2) represents a constraint (an SLA), which if violated, indicates a fault has occurred. *This Petri net has no fault transitions*, thereby the existing fault diagnosis techniques cannot directly be applied. One can argue that the RFT fault could be modelled by modifying the Petri net of Figure. 7.1; this would mean adding extra transitions and places to *simulate* violation of (7.2). Firstly, this is not an easy task. In addition, modifying the Petri net of Figure. 7.1 may result in a cumbersome and large Petri net which will be hard to understand. Thirdly, advocates of modelling faults must modify the design as soon as the SLA changes. Thus, there is clear scope for developing fault diagnosis techniques in Petri nets for the case where the fault is associated to a violation of constraints such as SLA and QoS.

*Violation of (7.2) can be represented as an inequality*: each sequence $\sigma$ in $\mathscr{N}$ that violates (7.2) satisfies (7.3). Hence if (7.2) is evaluated to *False*, then (7.3) will be evaluated to *True*.

$$\#(t_2, \sigma) - \#(t_{15}, \sigma) > \delta \tag{7.3}$$

Conversely, if $\sigma$ satisfies (7.2), then (7.3) is evaluated to *True*. Also, (7.3) can be rewritten as:

$$\#(t_{15},\sigma) - \#(t_2,\sigma) \leq -(\delta+1) \tag{7.4}$$

The inequalities in (7.2) and (7.4) capture the general form **e** in Definition 4.2. For example, assuming $x_2 = \#(t_2,\sigma), x_{15} = \#(t_{15},\sigma)$ and $b = -(\delta+1)$ or $\delta$; and the remaining coefficients are equal to zero, then both (7.4) and (7.2) correspond to **e**.

### 7.1.2 Common examples of SLA and QoS constraints

A wide range of SLA and QoS statements can be expressed as inequalities. For example, consider the ratio of message loss in communication channels. In this example, let us assume that $t_1$ represents the sending of a message to a channel and $t_2$ represents the arrival of the message at the other end. It is required that the ratio of the loss be $\frac{\#(t_1,\sigma)}{\#(t_2,\sigma)} \leq \frac{p}{q}$ which means $q \times \#(t_1,\sigma) - p \times \#(t_2,\sigma) \leq 0$. This inequality represents the constraint whose violation, written as $q \times \#(t_1,\sigma) - p \times \#(t_2,\sigma) > 0$, is seen as a fault, where $\frac{p}{q} > 1$ and $q \neq 0$.

Another example is called *accuracy* (success rate) [99] in which the ratio of completed jobs to the total number of allocated jobs is constrained. Assume that the transition $t_1$ models the allocation of a job and $t_2$ models the completion of this job. Then, the *accuracy* constraint is expressed as:

$$1 - \frac{\#(t_1,\sigma) - \#(t_2,\sigma)}{\#(t_1,\sigma)} \tag{7.5}$$

where $\sigma$ represents an execution sequence of events in the system. We aim to have the value 1 for *accuracy*, but this is not possible. Thus, the accuracy is tolerated to be below 1. Based on that, an SLA to ensure this constraint is:

$$1 - \frac{\#(t_1,\sigma) - \#(t_2,\sigma)}{\#(t_1,\sigma)} \geq \frac{p}{q} \tag{7.6}$$

where $\frac{p}{q} < 1$ and $q \neq 0$. Simplifying this formula yields:

$$\frac{\#(t_2, \sigma)}{\#(t_1, \sigma)} \geq \frac{p}{q} \tag{7.7}$$

which means that the ratio of the completion to allocation should be greater than $\frac{p}{q}$ and $q \neq 0$. The violation of this constraint occurs when (7.7) is evaluated to *False* for a sequence $\sigma$.

Furthermore, using Internet Protocol (IP) there is no guarantee that the packets sent on the network arrive at their destination in the sequence in which they were sent [100]. In which case, we say that the packet received in a different order is out-of-order. For instance, if a stream of packets have been received in the sequence 1, 4, 3, 2, 5, then packets 2 and 4 are out-of-order. This requires reordering them to read the sent message. The number of packets which are out-of-order can simply be measured using the reording ratio which represents the number of the packets which are not in their order over the total number of received packets. Assume that the transition $t$ represents receiving the packets at the destination part, then the out-of-order ratio can be expressed as follows:

$$\frac{N_{out}}{\#(t, \sigma)} \tag{7.8}$$

where $N_{out}$ represents the total number of packets which are out-of-order. The ideal case arises when (7.8) equals zero. However, the SLA constraints permit this ratio to be higher than zero, but within an acceptable level as shown in the following formula:

$$\frac{N_{out}}{\#(t, \sigma)} \leq \frac{p}{q} \tag{7.9}$$

where $\frac{p}{q} \geq 0$. Similarly, (7.9) can be rewritten as an inequality of the form $q \times N_{out} - p \times \#(t, \sigma) \leq 0$. Since these SLA and QoS statements and their violations (seen as faults) can be written as inequalities in which one or more variables correspond to unobservable transitions, the IFME approach is applicable to diagnose such faults as shown in the next sections.

## 7.2 The IFME approach to diagnose violations of constraints

To simplify the presentation, the problem of diagnosing violations of constraints will be studied in the context of free labelled Petri nets, i.e. the results of Chapter 5 are applied to address this problem. However, the results of this chapter can still be applied to the case of labelled Petri nets.

### 7.2.1 Problem definition

Consider a Petri net $(\mathcal{N}, M_0)$ as defined in Section 4.1, where $\mathcal{N}$ has no cycle of unobservable transitions. Note that the notion of fault transitions does not exist in this Petri net; however, we assume that there is a constraint, denoted $\phi$, which if violated means a fault has occurred. Thus, a sequence of events $\sigma$ for which $\#(\sigma) \not\models \phi$ contains a fault. Conversely, a given sequence $\sigma$ contains no fault if $\#(\sigma) \models \phi$.

To begin, assume that the system has a single constraint $\phi$ whose violation, denoted $\neg\phi$, is seen as a fault. Further, assume that $\phi := \sum_{j=1}^{n} a_j x_j \leq b$ and $\neg\phi := \sum_{j=1}^{n} a_j x_j > b$, where $x_1, \ldots, x_n$ corresponds to the number of firing the transitions $t_1, \ldots, t_n$ and $a_1, \ldots, a_n, b \in \mathbb{Z}$. In effect, the inequalities $\mathbf{c}$ and $\neg\mathbf{c}$, previously described, represent special cases of the inequalities $\phi$ and $\neg\phi$, respectively. Thus, the problem of fault diagnosis in partially-observed systems where faults are events can be considered as a special case of the problem of diagnosing violations of constraints.

To extend this notion to the case where there is a set of constraints $\Phi = \{\phi_1, \ldots, \phi_r\}$, then we can say that a fault $t_i,\ i = 1, \ldots, r$, has occurred if a constraint $\phi_i$ is violated. Accordingly, the diagnoser and diagnosis states previously defined (see Definition 5.1) can be redefined as follows:

**Definition 7.1.** A diagnoser is a mapping $\Delta : T_o^* \times \Phi \to \{NoFault, Faulty, Uncertain\}$ that associates to each observed sequence $\mathbf{s}$, with respect to the constraint $\phi_i$, one of the following diagnosis states:

- $\Delta(\mathbf{s}, \phi_i) = NoFault$ if $\forall \sigma \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma) = \mathbf{s}$, $\#(\sigma) \vDash \phi_i$ holds. This state shows that there is no sequence having the same observation as $\mathbf{s}$ violates $\phi_i$.

- $\Delta(\mathbf{s}, \phi_i) = Faulty$ if $\forall \sigma \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma) = \mathbf{s}$, $\#(\sigma) \vDash \neg\phi_i$ holds. This state implies that all sequences having the same observation as $\mathbf{s}$ violate $\phi_i$.

- $\Delta(\mathbf{s}, \phi_i) = Uncertain$ if there exists two sequences $\sigma_1$, $\sigma_2 \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma_1) = \pi(\sigma_2) = \mathbf{s}$, but $\#(\sigma_1) \vDash \phi_i$ and $\#(\sigma_2) \vDash \neg\phi_i$ hold. In which case, the behaviour of the system is ambiguous because of the possibility of both *NoFault* and *Faulty* states existing during the observed sequence. In other words, the violation of constraint may have occurred.

Note that having $\Delta(\mathbf{s}, \phi_i) = NoFault$ arisen for all constraints in the set $\Phi$ implies that the system state is normal.

### 7.2.2 The proposed solution using the IFME method

Suppose that $\mathcal{N} = (P, T, pre, post)$ is a Petri net with initial marking $M_0$ and no cycle of unobservable transitions exists. Let $\Phi = \{\phi_1, \ldots, \phi_r\}$ be the set of constraints and $\Phi' = \{\neg\phi_1, \ldots, \neg\phi_r\}$ be the set of their negations. Each constraint and its negation are represented as inequalities. In this new formalism, we do NOT have any concept of fault transitions and the systems under study are partially-observed.

The outline of the solution can be summarised as follows. We introduce variables $x_1, x_2, \ldots, x_n$ representing the number of firing of $t_1, t_2, \ldots, t_n$, respectively. Suppose that $-A\mathbf{x} \leq M_0$ with $\mathbf{x} \geq \vec{\mathbf{0}}$ is a set of inequalities expressing the state equation, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. We further assume that $\phi_i \in \Phi$ is the inequality $\sum_{j=1}^{n} a_j x_j \leq b$ and $\neg\phi_i \in \Phi'$ is its negation, i.e. the inequality $\sum_{j=1}^{n} a_j x_j > b$. For each firing sequence $\sigma$ of $(\mathcal{N}, M_0)$, if $\sigma$ violates $\phi_i$, then $\mathbf{x} = \#(\sigma)$, the Parikh vector of $\sigma$, satisfies $\neg\phi_i$. Conversely, for a firing sequence $\sigma$, if $\mathbf{x}$ satisfies $\phi_i$, then $\sigma$ does not violate $\phi_i$.

From a Petri net model, we first obtain a set of inequalities $I := \{-A\mathbf{x} \leq M_0\} \cup \{\mathbf{x} \geq \vec{\mathbf{0}}\}$. Then, we create two sets of inequalities $I \cup \{\phi_i\}$ and $I \cup \{\neg\phi_i\}$ for each constraint $\phi_i$ and its violation $\neg\phi_i$, respectively. Applying the IFME method simultaneously to both $I \cup \{\phi_i\}$ and

$I \cup \{\neg\phi_i\}$ for $i = 1,\ldots,r$, two reduced sets $R_i$ and $R'_i$ are created by eliminating every variable corresponding to a transition in the set $T_u$. We use these reduced sets to diagnose the occurrences of violations of constraints as stated in the following.

Note that Definition 5.2 is still the same, but it is now with regards to the constraint $\phi_i$, $1 \leq i \leq r$, and not the fault type. Also, the most recent diagnosis state of the empty string $\varepsilon$ is still *NoFault* because we assume that the system starts from a non-faulty state. Then, the following theorem extends the previous work presented in Chapter 5 to the case where the faults are not captured as events, instead they are captured as violations of constraints.

**Theorem 7.1.** Assume that $(\mathcal{N}, M_0)$ is a Petri net with no cycle of unobservable transitions. Let $I$ be the set of inequalities $-A\mathbf{x} \leq M_0$, created from the state equation of $\mathcal{N}$, along with $\mathbf{x} \geq \vec{\mathbf{0}}$. Suppose that $T = T_o \cup T_u$, $T_o = \{t_1,\ldots,t_k\}$ and $T_u = \{t_{k+1},\ldots,t_n\}$, in addition, the faults are not captured as events. The vector of variables $x_1,\ldots,x_n$ corresponds to the number of firings the transitions $t_1,\ldots,t_n$. Suppose also that $\Phi = \{\phi_1,\ldots,\phi_r\}$ is the set of constraints and $\neg\Phi$ is the set of their negations. Assume that the set of inequalities $R_i$ and $R'_i$ are respectively resulting from the application of the IFME to both $I \cup \{\phi_i\}$ and $I \cup \{\neg\phi_i\}$ to eliminate all variables corresponding to transitions in $T_u$. Then, for any given sequence of observed events $\mathbf{s} = \pi(\sigma)$, where $\mathbf{s} = \mathbf{s}'t$, $\mathbf{s} \in T_o^*$, $t \in T_o$ and $\sigma$ is a firing sequence in $\mathcal{N}$, $\Delta(\mathbf{s},\phi_i)$ is determined as follows:

$$\Delta(\mathbf{s},\phi_i) = \begin{cases} \mathit{NoFault} & \text{if } (\#(\mathbf{s}) \not\models R'_i) \\ \mathit{Faulty} & \text{if } (\#(\mathbf{s}) \not\models R_i) \\ & \vee((\#(\mathbf{s}) \models R_i) \wedge (\#(\mathbf{s}) \models R'_i) \\ & \wedge(\Delta(\mathbf{s}',\phi_i) = \mathit{Faulty})) \\ \mathit{Uncertain} & \text{if } (\#(\mathbf{s}) \models R_i) \wedge (\#(\mathbf{s}) \models R'_i) \\ & \wedge((\Delta(\mathbf{s}',\phi_i) = \mathit{NoFault}) \vee (\Delta(\mathbf{s}',\phi_i) = \mathit{Uncertain})) \\ \mathit{Impossible} & \text{if } (\#(\mathbf{s}) \not\models R_i) \wedge (\#(\mathbf{s}) \not\models R'_i) \end{cases}$$

*Proof.* This proof is presented for a single constraint $\phi_i, i = 1,\ldots,r$; however, obtaining a complete proof requires only repeating the same proof for every single constraint in the set $\Phi$. In the following, we assume that $\#(\mathbf{s}) = (\alpha_1,\ldots,\alpha_k)$.

**Proof of the case** *NoFault*: using contradiction, assume that $\#(\mathbf{s}) \not\models R'_i$, but the diagnosis state is not *NoFault*. If $\#(\mathbf{s}) \not\models R'_i$, then for every valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $v = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$, $v \not\models I \cup \{\neg\phi_i\}$ by Theorem 3.1. Since $\sigma$ is a firing sequence, we are certain that $v \models I$, see Lemma 4.1. Hence, $v \not\models \{\neg\phi_i\}$, i.e. $v \models \phi_i$. As a result, $\forall\sigma' \in L(\mathcal{N}, M_0)$ such that $\pi(\sigma') = \mathbf{s}, \#(\sigma') \models \neg\phi_i$. Hence a violation of constraint $\phi_i$ has occurred during observing $\mathbf{s}$. This contradicts the assumption.

**Proof of the case** *Faulty*: here we have two cases to be proved.

**Case i:** if $\#(\mathbf{s}) \not\models R_i$ holds. Using the same argument in proof of the case *NoFault* replacing $R'_i$ with $R_i$, we can prove this case.

**Case ii:** if $(\#(\mathbf{s}) \models R_i) \wedge (\#(\mathbf{s}) \models R'_i) \wedge (\Delta(\mathbf{s}', \phi_i) = Faulty)$ holds. Since $\Delta(\mathbf{s}', \phi_i) = Faulty$ holds, i.e. the most recent diagnosis state is *Faulty*, then a violation of $\phi_i$ has occurred during the observed sequence $\mathbf{s}'$. Since the fault (the violation of constraint) propagates to all the states following the *Faulty* state, then $\mathbf{s} = \mathbf{s}'t$ has also violated $\phi_i$.

**Proof of the case** *Uncertain*: we first assume that $\mathbf{s} = \varepsilon$, then there exists one possible case for the most recent diagnosis state, particularly *NoFault*, because we suppose that the system starts from a non-faulty state. Now let us prove the result in the case of $\mathbf{s} = \varepsilon$. If $\#(\mathbf{s}) \models R$, then there exists a valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $v = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$ and $v \models I \cup \{\phi_i\}$ by Theorem 3.1. If $v \models I \cup \{\phi_i\}$, then $v \models I$, i.e. $v$ satisfies $M' = M_0 + Av \geq \vec{\mathbf{0}}$. Since $\mathbf{s}$ has no observable transitions ($\mathbf{s} = \varepsilon$), then the subnet $\mathcal{N}_v$ has only unobservable transitions. Again, by the assumption that no cycle of unobservable transitions exists in $\mathcal{N}$, then $\mathcal{N}_v$ is cycle free. As a result, there exists $\sigma' \in T_v^*$ such that $M_0 \xrightarrow{\sigma'} M'$ and $\#(\sigma') = v$ by Lemma 5.1. Hence, the sequence $\sigma'$ does not violate the constraint $\phi_i$. Likewise, we can prove that if $\#(\mathbf{s}) \models R'_i$, there exists another sequence violating the constraint. Since there are two sequences having the same $\mathbf{s}$, but one satisfies $\phi_i$ and the other satisfies $\neg\phi_i$, then we have an *Uncertain* state, see Definition 7.1.

Now, assume that $\mathbf{s} = \mathbf{s}'t$, $t \in T_o$ and $\mathbf{s}' \in T_o^*$. Then there are two cases to be considered:

**Case i:** when the most recent diagnosis state is *NoFault* ($\Delta(\mathbf{s}', \phi_i) = NoFault$). If $\#(\mathbf{s}) \models R_i$, then there exists a valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $v = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$

and $\nu \vDash I \cup \{\phi_i\}$ by Theorem 3.1. If $\nu \vDash I \cup \{\phi_i\}$, then $\nu \vDash I$, i.e. $M'' = M_0 + A\nu \geq \vec{\mathbf{0}}$. Since no violation has occurred during observing $\omega$, and $t$ is an observable transition, then we are certain that all sequences $\sigma' t$ such that $M_0 \xrightarrow{\sigma' t} M'$ and $\pi(\sigma') = \mathbf{s}'$ satisfy $\phi_i$. Assuming $y = \nu - \#(\sigma' t),\ y \in \mathbb{N}^n$, then $M'' = M' + Ay \geq \vec{\mathbf{0}}$. Since the subnet $\mathscr{N}_y$ has only unobservable transitions, then $\mathscr{N}_y$ is cycle free. As a result, there exists $\sigma'' \in T_y^*$ such that $M' \xrightarrow{\sigma''} M''$ and $\#(\sigma'') = y$ by Lemma 5.1. Hence, the sequence $\sigma' t \sigma''$ with $\#(\sigma' t \sigma'') = \nu$ satisfies $\phi_i$. Likewise, we can prove that if $\#(\mathbf{s}) \vDash R_i'$, there exists another sequence violating $\phi_i$. Since there are two sequences having the same $\mathbf{s}$, but one satisfies $\phi_i$ and the other satisfies $\neg\phi_i$, then we have an *Uncertain* state.

**Case ii:** when the most recent diagnosis state is *Uncertain* ($\Delta(\mathbf{s}', \phi_i) = Uncertain$). If $\#(\mathbf{s}) \vDash R_i$, then there exists a valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$ and $\nu \vDash I \cup \{\phi_i\}$ by Theorem 3.1. If $\nu \vDash I \cup \{\phi_i\}$, then $\nu \vDash I$, i.e. $M'' = M_0 + A\nu \geq \vec{\mathbf{0}}$. Since we have *Uncertain* state during observing $\mathbf{s}'$, i.e. the most recent diagnosis state is *Uncertain*, and $t$ is an observable transition, then we still have the same state for any sequence $\sigma' t$ such that $M_0 \xrightarrow{\sigma' t} M'$ and $\pi(\sigma') = \mathbf{s}'$. Assuming $y = \nu - \#(\sigma' t),\ y \in \mathbb{N}^n$, then $M'' = M' + Ay \geq \vec{\mathbf{0}}$. Since the subnet $\mathscr{N}_y$ has only unobservable transitions, then $\mathscr{N}_y$ is cycle free. As a result, there exists $\sigma'' \in T_y^*$ such that $M' \xrightarrow{\sigma''} M''$ and $\#(\sigma'') = \nu$ by Lemma 5.1. Hence, the sequence $\sigma' t \sigma''$ with $\#(\sigma' t \sigma'') = \nu$ satisfies $\phi_i$. Similarly, we can prove that if $\#(\mathbf{s}) \vDash R_i'$, there exists another sequence violating $\phi_i$. Since there are two sequences having the same $\mathbf{s}$, but one satisfies $\phi_i$ and the other satisfies $\neg\phi_i$, then an *Uncertain* state has arisen.

**Proof of the case** *Impossible*: assume that $\#(\mathbf{s}) \nvDash R_i$ and $\#(\mathbf{s}) \nvDash R_i'$, but this case is possible. If $\#(\mathbf{s}) \nvDash R_i$, then for every valuation $(\alpha_{k+1}, \ldots, \alpha_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \alpha_{k+1}, \ldots, \alpha_n)$, $\nu \nvDash I \cup \{\phi_i\}$ by Theorem 3.1. Also, if $\#(\mathbf{s}) \nvDash R_i'$, then for every valuation $(\beta_{k+1}, \ldots, \beta_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \beta_{k+1}, \ldots, \beta_n)$, $\nu \nvDash I \cup \{\neg\phi_i\}$ by Theorem 3.1. Rephrasing this statement, we can say that there exists a valuation $(\beta_{k+1}, \ldots, \beta_n)$ of $(x_{k+1}, \ldots, x_n)$ such that $\nu = (\alpha_1, \ldots, \alpha_k, \beta_{k+1}, \ldots, \beta_n)$ and $\nu \vDash I \cup \{\phi_i\}$ taking into account that $\neg\phi_i$ is the negation of $\phi_i$ and $\sigma$ is a firing sequence of $\mathscr{N}$, i.e. $\#(\sigma) \vDash I$. Here we have contradictory statements. Consequently, this case is an impossible case. This contradicts the assumption and completes the proof. □

Table 7.1: The sets of inequalities resulting from applying the IFME method in Example 7.1

| No. | $R$ | $R'$ |
|---|---|---|
| 1 | $x_1 - x_{15} \leq 1$ | $x_1 - x_{15} \leq 1$ |
| 2 | $-x_1 + x_{15} \leq 0$ | $-x_1 + x_{15} \leq 0$ |
| 3 | $-x_1 - x_4 + x_{15} \leq 0$ | $-x_1 - x_4 + x_{15} \leq 0$ |
| 4 | $-x_1 - x_3 + x_{15} \leq 0$ | $-x_1 - x_3 + x_{15} \leq 0$ |
| 5 | $-x_1 - x_5 + x_{15} \leq 0$ | $-x_1 - x_5 + x_{15} \leq 0$ |
| 6 | $-x_1 - x_4 - x_5 + x_{15} \leq 0$ | $-x_1 - x_4 - x_5 + x_{15} \leq 0$ |
| 7 | $-x_1 - x_3 - x_5 + x_{15} \leq 0$ | $-x_1 - x_3 - x_5 + x_{15} \leq 0$ |
| 8 | $-x_1 - x_3 - x_4 - x_5 \leq 0$ | $-x_1 - x_3 - x_4 - x_5 \leq 0$ |
| 9 | $-x_1 - x_3 - x_4 + x_{15} \leq 0$ | $-x_3 - x_4 - x_5 + x_{15} \leq 0$ |
| 10 | $x_3 + x_4 + x_5 - x_{15} \leq 2$ | $2x_1 - x_3 - x_4 + 2x_{15} \leq 0$ |
| 11 | $-x_3 - x_4 - x_5 + x_{15} \leq 0$ | $-x_1 - x_3 - x_4 - x_5 + 2x_{15} \leq -3$ |
| 12 | $-x_1 - x_3 - x_4 - x_5 + x_{15} \leq 0$ | $-2x_1 - 2x_3 - 2x_4 - 2x_5 + 3x_{15} \leq -6$ |
| 13 | $-x_1 \leq 0$ | $-x_1 \leq 1$ |

**Remark 7.1.** As previously remarked in Chapter 5, the proofs of the diagnosis states *NoFault* and *Faulty* in theorem 7.1 are still valid for the Petri nets having cycles of unobservable transitions.

Theorem 7.1 provides a systematic procedure to detect the occurrences of violations of constraints. Note that the case where the observed sequence does not satisfy both $R_i$ and $R'_i$ for a given constraint $\phi_i$ is not possible.

**Remark 7.2.** The shape of each individual inequality that expresses the constraint and its violation is important. For example, a less interesting special case appears when all the non-zero coefficients in the inequality belong to observable transitions. For example, when in $\sum_{i=1}^{n} a_i x_i$, we have $a_i = 0$ if $x_i$ represents an unobservable transition. In such a case, the sum $\sum_{i=1}^{n} a_i x_i$ can easily be calculated after counting the number of occurrences of observable events. This is similar to the case of classic fault diagnosis, where some fault transitions are observable. A more interesting case arises when $a_i \neq 0$ for one or more unobservable transitions.

**Example 7.1.** Consider the Petri net $\mathcal{N}$ of Figure. 7.1 of the running example. A special case of an RFT fault is described using (7.2) of Section 7.1.1. Assuming that $\delta = 2$, the constraint $\phi$ is written as $\phi := x_2 - x_{15} \leq 2$ and its violation as $\neg\phi := x_{15} - x_2 \leq -3$ (Note that $\neg\phi$ has

Table 7.2: Diagnosis state estimations

| **No.** | $\mathbf{s} = \pi(\sigma)$ | $\#(\mathbf{s}) \vDash R$? | $\#(\mathbf{s}) \vDash R'$? | Diagnosis state |
|---|---|---|---|---|
| 1 | $\varepsilon$ | Yes | No | *NoFault* |
| 2 | $t_1$ | Yes | No | *NoFault* |
| 3 | $t_1t_3$ | Yes | No | *NoFault* |
| 4 | $t_1t_3t_3$ | Yes | Yes | *Uncertain* |
| 5 | $t_1t_3t_3t_3$ | No | Yes | *Faulty* |
| 6 | $t_1t_3t_3t_3t_{15}$ | Yes | No | *NoFault* |
| 7 | $t_1t_3t_3t_3t_3t_{15}$ | No | Yes | *Faulty* |
| 8 | $t_1t_3t_{15}$ | Yes | No | *NoFault* |
| 9 | $t_1t_3t_{15}t_1$ | Yes | No | *NoFault* |
| 10 | $t_1t_3t_{15}t_1t_3$ | Yes | No | *NoFault* |
| 11 | $t_1t_3t_{15}t_1t_3t_{15}$ | Yes | No | *NoFault* |

been rewritten in the standard form of $I$). Adding these inequalities simultaneously to the set of inequalities $I$ derived from (3.1), two sets $I \cup \{\phi\}$ and $I \cup \{\neg\phi\}$ are obtained. Then, using the IFME method to eliminate all variables corresponding to unobservable transitions produces the sets of inequalities in Table 7.1. The resulting sets of inequalities $R$ and $R'$ have only variables corresponding to observable transitions $\{t_1, t_3, t_4, t_5, t_{15}\}$. These two sets are used for estimating the current state of the system for a given observed sequence of events.

Table 7.2. shows different observed sequences and the diagnosis state estimated in each case. By looking at the Petri net in the figure, when the diagnoser observes no sequence ($\mathbf{s} = \varepsilon$), the diagnosis state is *NoFault*, i.e. no violation of the constraint $\phi$ has occurred. In which case, the diagnoser is certain that for all sequences having no observable transitions, $\phi$ is evaluated to *True* as $x_2 = 0$ and $x_{15} = 0$ for these sequences.

The same diagnosis state is estimated when observing the sequences 2, 3, 6, 8, 9, 10 and 11 shown in Table 7.2. For instance, in the case of $\mathbf{s} = t_1$, only two sequences, namely $\sigma_1 = t_1$ and $\sigma_2 = t_1t_2$, have $\pi(\sigma_1) = \pi(\sigma_2) = \mathbf{s}$. However, for both of them $\phi$ is evaluated to *True* because $x_2 = \#(t_2, \sigma_1) = 0, x_{15} = \#(t_{15}, \sigma_1) = 0$ and $x_2 = \#(t_2, \sigma_2) = 1, x_{15} = \#(t_{15}, \sigma_2) = 0$. In other words, $\#(\sigma_1), \#(\sigma_2) \vDash \phi$. Thus, the diagnosis state is *NoFault*.

On the other hand, suppose that sequence 5 is observed. In that case, there exist three

sequences $\sigma_1 = t_1t_2t_3t_6t_2t_3t_6t_2t_3t_6t_2$, $\sigma_2 = t_1t_2t_3t_6t_2t_3t_6t_2t_3t_7$ and $\sigma_3 = t_1t_2t_3t_6t_2t_3t_6t_2t_3t_7t_{12}$ with $\pi(\sigma_1) = \pi(\sigma_2) = \pi(\sigma_3) = t_1t_3t_3t_3$. By looking at $x_2 = \#(t_2, \sigma_i)$ and $x_{15} = \#(t_{15}, \sigma_i)$, we find that $\#(\sigma_i) \vDash \neg\phi$ for $i = 1,2,3$. As a result, a violation of $\phi$ has certainty occurred.

In the case where sequence 4 is observed, we again have three sequences $\sigma_1 = t_1t_2t_3t_6t_2t_3t_6t_2$, $\sigma_2 = t_1t_2t_3t_6t_2t_3t_7$ and $\sigma_3 = t_1t_2t_3t_6t_2t_3t_7t_{12}$ with $\pi(\sigma_1) = \pi(\sigma_2) = \pi(\sigma_3) = t_1t_3t_3$. Obviously, $\#(\sigma_1) \vDash \neg\phi$, but $\#(\sigma_2), \#(\sigma_3) \vDash \phi$. This is an *Uncertain* state because the diagnoser cannot decide whether a violation of $\phi$ occurred. Note that the same results shown in Table 7.2 can be obtained by replacing the transition $t_3$ by $t_4$ or $t_5$. For instance, the observed sequences $t_1t_4$ and $t_1t_5$ do not violate $\phi$ as $t_1t_3$ does not.

## 7.3 Chapter summary

A different form of faults in partially-observed discrete event systems modelled using Petri nets has been presented in this chapter. According to this form, the faults are not captured as events, but as violations of constraints. Since such violations can be written as inequalities, the IFME approach can also be applied to diagnose them.

Using the IFME approach, the diagnoser is represented by two sets of inequalities in variables corresponding to observable transitions. These sets are obtained as follows. First, two sets of inequalities, derived from the state equation, are augmented by the inequalities expressing the constraint and its violation. Then, the IFME method is applied to eliminate the variables corresponding to unobservable transitions. The resulting sets represent the diagnoser. The notion presented in this chapter has been explained with the aid of a running example representing a real problem in a telecommunication company.

# CHAPTER 8

# IMPLEMENTATION AND EVALUATION

In this chapter, the implementation details of the IFME approach for fault diagnosis are given. In addition, an evaluation of this approach is presented and compared with the diagnoser automata approach [4] extended to Petri nets, as discussed in Section 2.2.1. This evaluation is performed based on two criteria: 1) the size of the diagnoser; and 2) the time for computing the diagnosis states.

We start the chapter by presenting the architecture of the software tool which has been developed to implement both the offline and online steps of the proposed approach, i.e. creating the diagnoser (the sets of inequalities $R_i$ and $R'_i,\ i = 1, \ldots, r$) offline and using Esper CEP to build an application to diagnose faults in a stream of observed events online. This stream of events is generated by the simulation step based on a Petri net model.

The last part of the chapter focuses on conducting some computational experiments to evaluate the performance of the IFME approach. These experiments are carried out to analyse the space requirements of the offline step and the time requirements of the online step in the fault diagnosis process.

## 8.1 The architecture of the software tool

The software tool has been developed using Java. Figure. 8.1 depicts an architectural view of the tool which mainly consists of three parts. The first part is concerned with creating pairs of sets

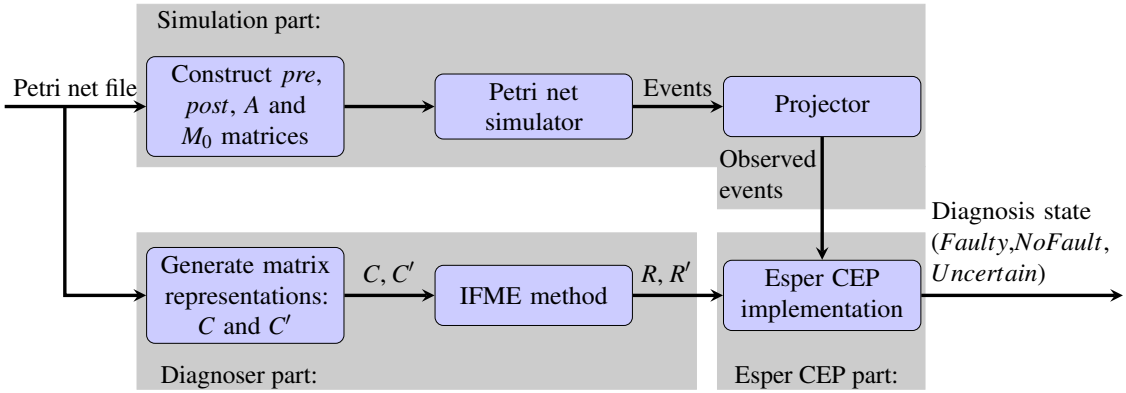



Figure 8.1: The main components of the software tool for fault diagnosis

of inequalities (the diagnoser), where each pair is associated with one fault type. The second part represents the Esper CEP application which uses these sets in order to diagnose faults. This part receives its input from the results of the simulation; the third part, on a Petri net model.

### 8.1.1 Creating the diagnoser: the offline step

The steps required to create the diagnoser (sets of inequalities $R$ and $R'$) are described in the flowchart shown in Figure. 8.2. These steps can be explained as follows. The input of this flowchart is a file representing a given Petri net. The header of the file has four fields: the number of places, the number of transitions, the number of fault types, in addition to a vector of values corresponding to the number of faults in each fault type. The data part of this file is structured into $m$ lines, where $m$ is the number of places in the Petri net. Each line contains all information related to a place in a Petri net. In particular, each line has the following fields in the order:

1. The initial marking.
2. Number of input transitions.
3. List of input transitions each of which is formatted as a record having, in order:
    - transition ID
    - weight of the edge between the transition and the place and

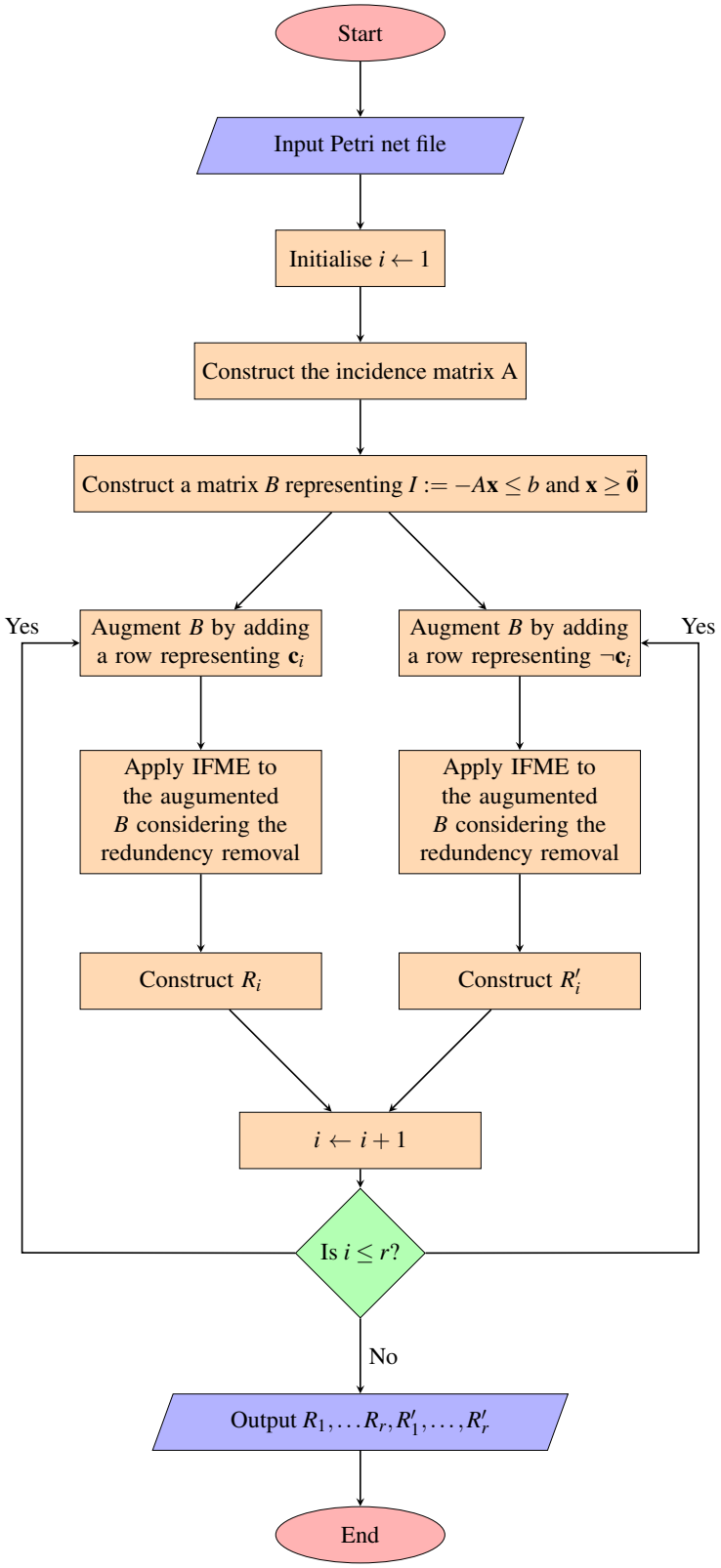



Figure 8.2: Flowchart for creating the sets of inequalities offline

- input transition status (observable (0), unobservable (-1) and fault (fault type number, i.e. if the fault in type 1, then this field has the value 1).

4. Number of output transitions.

5. List of output transitions each of which captures the same format defined for input transitions.

For example, considering the Petri net of Figure. 5.1, then the file associated is illustrated below:

```
7 7 2 1 2
#1 1 1 6 1 0 2 1 1 1 2 1 2
#2 1 1 7 1 -1 3 1 0
#3 0 1 1 1 1 1 4 1 0
#4 0 1 2 1 2 1 5 1 2
#5 0 1 3 1 0 2 4 1 0 5 1 2
#6 0 2 4 1 0 5 1 2 1 6 1 0
#7 0 2 4 1 0 5 1 2 1 7 1 -1
```

As shown in the flowchart, a matrix $B$ representing the set of inequalities $I := -Ax \leq M_0$ equipped with $\mathbf{x} \geq \vec{\mathbf{0}}$ is generated from this file. This matrix has $m+n$ rows and $n+1$ columns. The $m \times n$ submatrix of $B$ represents $-A$; the non-negative constraints on $\mathbf{x}$ are represented by rows $m+1$ to $m+n$. The last column of $B$ corresponds to the initial marking, where each entry of this column corresponds to a place in the Petri net. Let us consider the Petri net of Figure. 5.1, then the matrix $B$ of this net is described in (8.1).

$$
B = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \\ p_7 \\ x_1 \geq 0 \\ x_2 \geq 0 \\ x_3 \geq 0 \\ x_4 \geq 0 \\ x_5 \geq 0 \\ x_6 \geq 0 \\ x_7 \geq 0 \end{array}
\begin{array}{c}
\begin{array}{ccccccc|c} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & M_0 \end{array} \\
\left(\begin{array}{ccccccc|c}
-1 & -1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 \\
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0
\end{array}\right)
\end{array}
\tag{8.1}
$$

We then simultaneously augment the matrix $B$ by adding rows representing $\mathbf{c}_i$ and $\neg\mathbf{c}_i$, $i = 1,\ldots,r$. Suppose that $row_1^i$ and $row_2^i$ represent $\mathbf{c}_i$ and $\neg\mathbf{c}_i$, respectively, then $row_1^i$ and $row_2^i$ has non-zero values in the entries corresponding to the fault transitions with respect to a given fault type $T_f^i$. With a slight difference of $row_1^i$, $row_2^i$ has also the value -1 in the last entry.

Consider again the Petri net of Figure. 5.1. Then, the individually added rows to $B$ representing $\mathbf{c}_1$, $\neg\mathbf{c}_1$, $\mathbf{c}_2$ and $\neg\mathbf{c}_2$ are respectively:

$$
row_1^1 = \begin{array}{c}
\begin{array}{ccccccc|c} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & \end{array} \\
\left(\begin{array}{ccccccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)
\end{array}
\tag{8.2}
$$

$$
row_2^1 = \begin{array}{c}
\begin{array}{ccccccc|c} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & \end{array} \\
\left(\begin{array}{ccccccc|c} -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{array}\right)
\end{array}
\tag{8.3}
$$

$$row_1^2 = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \end{matrix} \\ \left( \begin{array}{ccccccc|c} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{array} \right) \end{array} \tag{8.4}$$

$$row_2^2 = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 \end{matrix} \\ \left( \begin{array}{ccccccc|c} 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \end{array} \right) \end{array} \tag{8.5}$$

After obtaining these augmented matrices, the IFME method described in Section 3.3 is applied to them. In our implementation of the method, an elimination of a variable representing an unobservable transition corresponds to making all entries in the associated column equal to zero. This can be fulfilled as follows.

Let $C$ be an augmented matrix inputting to the IFME method. First, all rows of this matrix are normalised by obtaining the GCD between the values of entries in $C$, and then dividing each value by the obtained GCD. Secondly, the variable in question is eliminated. By doing so, all entries in the column representing this variable in the augmented matrix $C$ become zero. To carry out this elimination according to the IFME method, the set of rows of $C$ is first divided into three subsets, say $C^0$, $C^+$ and $C^-$. Assume that the variable $x_1$ is to be eliminated, i.e. all entries of the first column are to be zero, then $C^0$, $C^+$ and $C^-$ have all rows in $C$ with the first entry equal to $0$, $> 0$ and $< 0$, respectively. All rows in the matrix $C^0$, if any, are copied into the new matrix representing the produced set of inequalities after the elimination of $x_1$. Also, it is possible that either $C^-$ or $C^+$ is empty. In this case, no new rows are generated.

Assuming that both $C^-$ and $C^+$ are not empty, then each row of $C^-$ is summed to each row in $C^+$. After each summation operation, a new row (new inequality) results and also the entries of two rows constituting this row become zeros. Furthermore, this new row is manipulated to ensure two cases. The first case is to obtain the integer solutions and the second case is to minimise the redundancy resulting from the nature of the elimination by the IFME method. We implement the rule of minimising the redundancy stated in Section 5.2.6.

As illustrated in the flowchart of Figure. 8.2, the sets of inequalities produced from eliminations $R_1, \ldots, R_r, R'_1, \ldots, R'_r$ are sent to the console screen. Also, for a better understanding, the

following example is presented to illustrate the outputs at different stages of the implementation.

**Example 8.1.** In this example, we explain the application of the IFME method to the matrix in (8.1) augmented by the row $row_1^1$ in (8.2); one iteration of the IFME method is applied. In this iteration, assume that the variable $x_1$ corresponding to the unobservable transition $t_1$ is to be eliminated.

Since all values in the matrix are 0, -1 or 1, normalisation will not change the matrix values. Thus, the next step is to partition the set of rows in the augmented matrix into three subsets $C^0$, $C^-$ and $C^+$ shown respectively in (8.6), (8.7) and (8.8):

$$C^0 = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & & \end{matrix} \\ \left(\begin{array}{ccccccc|c} 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \end{array}\right) \end{array} \tag{8.6}$$

$$C^- = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & & \end{matrix} \\ \left(\begin{array}{ccccccc|c} -1 & -1 & 0 & 0 & 0 & 1 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array} \tag{8.7}$$

$$C^{+} = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & & \end{matrix} \\ \left(\begin{array}{ccccccc|c} 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array} \tag{8.8}$$

If we assume that $R$ represents the matrix holding the result of the elimination, then all the rows of the matrix $C^0$ are copied to $R$. Next, the first row of $C^-$ is summed to both the first and second rows of $C^+$. Similarly, the second row of $C^-$ is summed to both the first and second rows of $C^+$. As a result, four rows are added to the matrix $R$ as shown in the following matrix:

$$R = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{array} \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & & \end{matrix} \\ \left(\begin{array}{ccccccc|c} 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 & 0 & 1 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array} \tag{8.9}$$

Note that the first column of the matrix $R$ has only zeros. Also, the rows 12-15 represent the rows resulting from the elimination by summing each row of $C^-$ to each row of $C^+$; the last row can be deleted because all of its entries have the value 0.

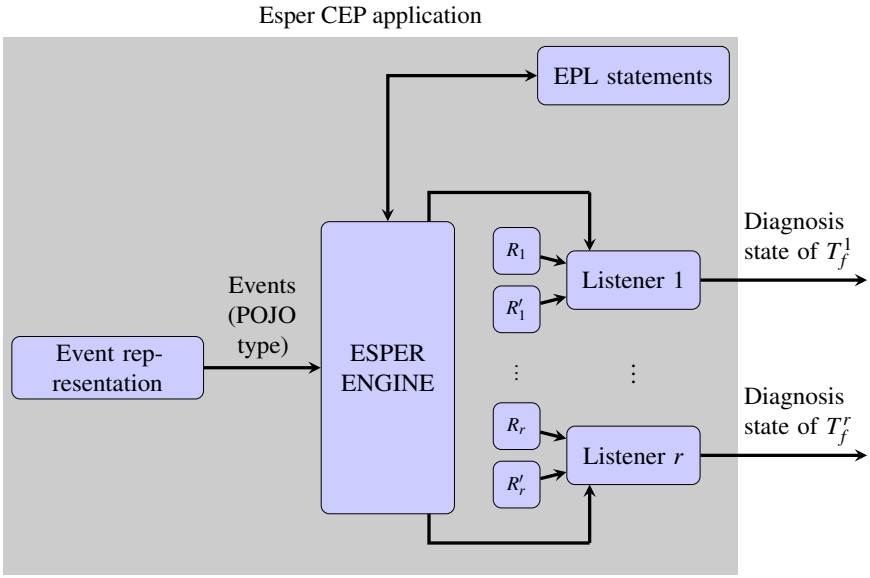



Figure 8.3: Esper CEP application architecture for implementing online fault diagnosis

### 8.1.2 Esper CEP application: the online step

We have used the facilities of Esper CEP to develop an application for implementing Algorithm 5.2 described in Chapter 5 to perform online fault diagnosis. The architecture of the developed application is depicted in Figure. 8.3, where the main components of the application and their interactions in order to compute the diagnosis are shown. The details of these components can be explained in the following:

***Event representation:*** Esper provides different methods to model events in a form suitable for processing by its engine. In our application, the events are represented as Plain-Old Java Object (POJO) classes. The following code represents a POJO class for the observed events, where a single property and two methods are declared:

```
class ObservedEvents{
        private int ID;
        public void setID(int ID){this.ID=ID;}
        public int getID(){return ID;}
}
```

***Creating the Esper engine instance:*** to access the facilities provided by Esper, an engine instance needs to be created. Engine instances are represented by the interface `EPServiceProvider`. To create an instance of Esper engine, the `getDefaultProvider` method is invoked on the `EPServiceProviderManager` as shown in the following code:

```
EPServiceProvider epService=EPServiceProviderManager.getDefaultProvider();
```

The object `epService` represents an Esper engine instance which can be used to access more services provided by the engine.

***EPL and pattern statements:*** EPL and pattern statements can be imagined as SQL queries. These statements are built via the administrative interface `EPAdministrator`, by first creating an instance of the interface `EPAdministrator` and then writing the EPL statement as shown in the following code:

```
EPAdministrator
admin=EPServiceProviderManager.getDefaultProvider().getEPAdministrator();
EPStatement
eplstate=admin.createEPL("insert␣into␣Parikh␣select␣ID,␣count(ID)␣as␣mycount
from␣kk.ObservedEvents␣group␣by␣ID");
```

An instance of type `EPStatement` is returned by invoking the method `createEPL` on the interface `EPAdministrator` instance, `admin`. Simply, the EPL statement `eplstate` defined above instructs the engine to do three operations: 1) group the events according to their `ID`; 2) count the number of appearances of each event in each group and save it in the variable `mycount`; and 3) insert the `ID` and the appearance frequency of these events into a new stream called `Parikh`.

***Adding listeners:*** the results generated by the EPL statement are posted to listeners. One or more listeners can be added to each EPL statement. The Esper engine continuously submits the results of an EPL statement to its listener(s) as soon as they are produced by the statement. In our implementation, as many listeners as fault types are added to the statement `eplstate`. Adding

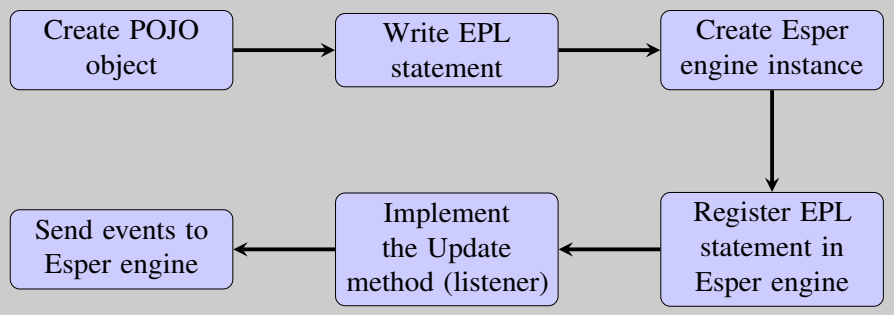



Figure 8.4: Main steps used to build our Esper CEP application

a listener to an EPL statement can be achieved using the `addListener()` method. For example, to add the listener `listener` to the EPL statement `eplstate` created previously, the following code line is used:

```
eplstate.addListener(listener);
```

where `listener` represents an instance of the user-defined class `Online_diagnosis` which implements the `UpdateListener` interface provided by the Esper engine. This interface contains the method `Update` whose implementation is provided by our application.

```
class Online_diagnosis implements UpdateListener{
...
   public void update(EventBean[] newEvent,EventBean[] oldEvent){
   ...
      int ID= (int) newEvent[0].get("ID");
      long D= (long) newEvent[0].get("mycount");
   ...
   }
}
```

The listeners receive the output of the EPL statement via the `Update` method parameters `EventBean[]`. Once a listener obtains the statement results, these results are processed by the code provided in the `Update` method.

***The Runtime Interface:*** the mechanism of sending events to the Esper engine is im-

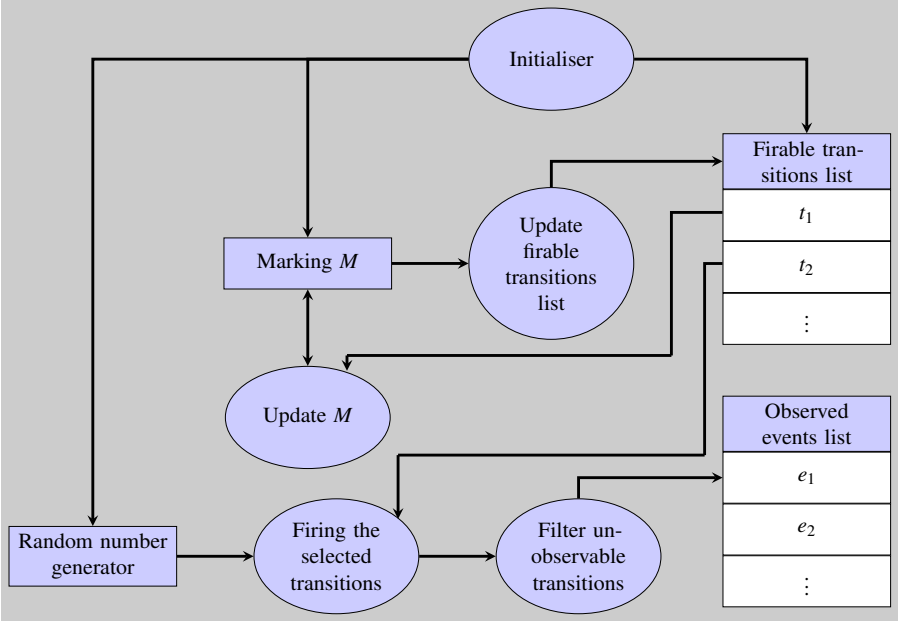



Figure 8.5: Petri nets simulation

plemented by the interface `EPRuntime`. To obtain an instance of this interface, the method `getEPRuntime` is invoked on the engine instance of type `EPServiceProvider` mentioned previously. Then, the obtained instance is used to invoke the method `sendEvent()` whose function is to send events to the engine for processing. Figure. 8.4 depicts the main steps included in our implementation.

### 8.1.3 Petri net simulator

The aim of this section is to present the main components of the Petri net simulator with a description of the function of each component. These components and their interactions are depicted in Figure. 8.5. The purpose of the simulation is to produce sample paths representing different observed execution sequences of the system modelled by a Petri net. Obtaining these paths enables us to compute the diagnosis states made by our approach when an event is observed.

To generate an observed sample path, two lists are to be first defined. The first list is called the firable transitions list and is used to track the transitions enabled on marking $M$, see Figure. 8.5. The second list is allocated for events associated with fired observed transitions. Assuming that the initial marking $M_0$ is given, we can initialise the list of firable transitions by all transitions

enabled at $M_0$. Then, the steps of the simulation procedure is continuously iterated as follows:

1. Select randomly a transition from the firable transitions list.
2. Fire this transition and update the current marking accordingly.
3. Add the event associated with the fired transition to the observed events list if the transition is an observable transition.
4. Update the firable list according to the new updated marking.

Initially, the marking $M$ receives the values of the initial marking $M_0$. Note that filtering the unobservable transitions shown in Figure. 8.5 simulates the projection operation on observable transitions in the context of partially-observed DES. During this filtration, firing any unobservable transition is ignored, while firing the observable transition results in adding the associated event to the observed events list.

## 8.2 Evaluation

In this section, we establish a comparison between the IFME approach presented in Chapter 5 and the diagnoser automata approach of [4] for fault diagnosis in Petri nets. The goal of this comparison is to evaluate the performance of our approach against the standard approach and also to provide some experimental results supporting the theoretical results given in the previous chapters. Two criteria for comparison are adopted, namely the size of the diagnoser and the diagnosis computation time (the time for online fault diagnosis). By choosing the diagnoser automata approach to compare with, we can demonstrate how efficient the IFME method could achieve the compromise between the space and time as both approaches adopt the idea of the *compiled* diagnoser. On the contrary, the ILP approach is not strictly comparable with our approach since it implements the notion of the *interpreted* diagnoser as opposed to the *compiled* diagnoser. In addition, the difference in time complexity between the diagnoser approach (constant complexity) and our approach (polynomial complexity) is less than the difference

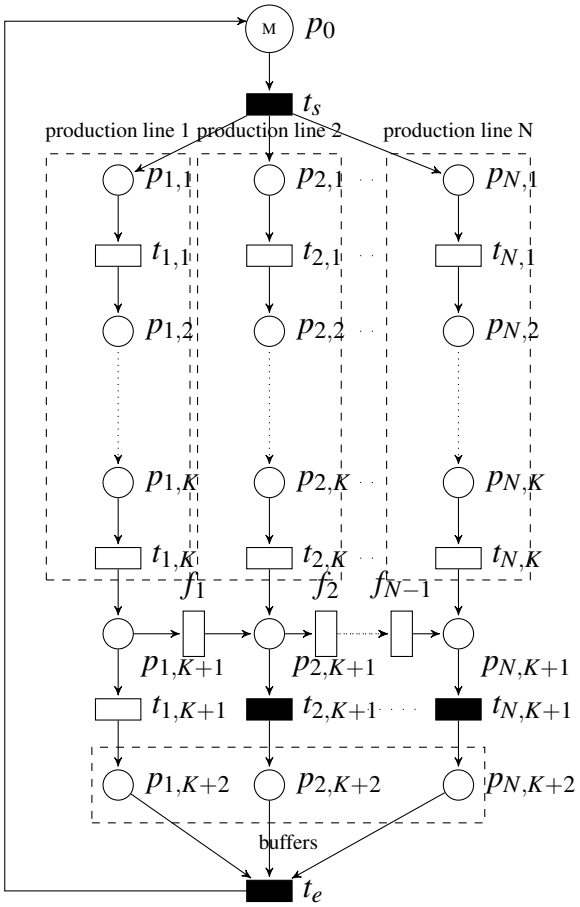



Figure 8.6: The benchmark example representing a manufacturing system

between our approach and the ILP approach (exponential complexity). Thus, empirical results from the ILP approach are not included in this comparison. However, a theoretical point of view comparison is provided in Section 9.2 to discuss the differences between the IFME approach and the ILP approach for fault diagnosis.

To this end, we conduct some computational experiments on a DES benchmark example [40–42], which is widely used, by applying both the IFME and diagnoser automata approaches to different instances of the general Petri net given in this example.

### 8.2.1 The benchmark example

We use the benchmark example presented in [40–42] and depicted in Figure. 8.6. The Petri net example of this benchmark models a general class of manufacturing systems. This Petri

net model can be extended in different ways using three control parameters: $N$, $M$ and $K$. By modifying the values of these parameters, different instances of the Petri net in 8.6 can be obtained. This is an important requirement to observe the change in the size of the diagnoser for different numbers of unobservable transitions (changing $K$) and different sizes of the state space (changing $M$), and then measure the time accordingly. The parameters $N$ and $K$ are about the size of the system as a two-dimensional grid, see Figure. 8.6. Let $N$ and $M$ respectively denote the number of production lines and the number of units composing the final product. These units are simultaneously produced where each unit passes through a number of operations, $K$, in each production line.

Obtaining one unit of the final product requires sending $N$ orders (firing the transition $t_s$), each of which is allocated to one production line. The output of each line is one part (all parts are identical) which is finally put in a buffer (places $p_{i,K+2}, i = 1,\ldots,N$). From these buffers, the assembly station takes each part (firing transition $t_e$) to produce the final product. Before sending the completed parts to the buffers, each part in line $i$ $(i = 1,\ldots,N)$ is processed by a series of operations modelled by transitions $t_{i,j}$ $(j = 1,\ldots,K)$. After finishing this stage, two states are possible: either the part in line $i$ is sent to the right buffer (firing transition $t_{i,K+1}$) or a fault has occurred (firing transition $f_i$, $i = 1,\ldots,N-1$).

In fact, the fault in this case expresses the state where the part $i$ moves from line $i$ to $i+1$ at the same processing stage which it has already been through. For simplicity, these faults are assumed to be of the same type. As previously highlighted, all observable transitions in Figure. 8.6 are depicted by solid rectangles, while empty rectangles represent unobservable transitions.

### 8.2.2 Results and discussion

This section presents and analyses the experimental results obtained from implementing the two different approaches: for both creating the diagnoser and the online diagnosis computations. The language used for the purpose of the implementation is Java 1.7.0. All experiments described in this section have been run on a PC Intel processor with a clock of 3.40GHz and 8GB of RAM.

The implementation of the IFME approach has been described in detail in the first part of

Table 8.1: The diagnoser size using the diagnoser automata approach and the IFME approach. The results are obtained using different values of the parameter $N$ when $K = 1$, $M = 1$.

| **Example parameters** | | | **Diagnoser automata** | | **IFME** |
|---|---|---|---|---|---|
| $N$ | $\|T_o\|$ | $\|T_u\|$ | RG size | Diagnoser size | Diagnoser size |
| 2 | 3 | 4 | 15 | 4 | 23 |
| 3 | 4 | 5 | 80 | 10 | 27 |
| 4 | 5 | 6 | 495 | 29 | 56 |
| 5 | 6 | 7 | 3295 | 91 | 82 |
| 6 | 7 | 8 | $\infty$ | $\infty$ | 124 |
| 7 | 8 | 9 | $\infty$ | $\infty$ | 198 |
| 8 | 9 | 10 | $\infty$ | $\infty$ | 336 |
| 9 | 10 | 11 | $\infty$ | $\infty$ | 602 |
| 10 | 11 | 12 | $\infty$ | $\infty$ | 1124 |

this chapter. On the other hand, even though the diagnoser automata approach has already been implemented in the tool described in [101], the source code is not publicly available. In addition, most of the functions and data structures developed in this tool are not particularly self-contained. Thus, we also developed our own code to implement this approach. Previously, code to produce the *reachability graph* (RG) from the Petri net model was developed. Then, the generated graph is used as an input to the diagnoser automata approach.

Three experiments have been conducted with different values of the parameters $N$, $K$ and $M$. The purpose of the experiments is to study the impact of these parameters on the size of the diagnoser created using both approaches. The numerical results of these experiments are summarised in Tables 8.1, 8.2 and 8.3. Columns $|T_o|$ and $|T_u|$ represent the number of observable and unobservable transitions, respectively. Also, the diagnoser size in the case of the IFME approach represents the number of inequalities in $R$ plus the number of inequalities in $R'$. Note that some fields of the tables are labelled by $\infty$. This symbol implies that we could not obtain the corresponding results due to exceeding the limit of heap space (even with 4GB heap size).

From Table 8.1, we observe that the IFME approach relatively outperforms the diagnoser automata approach when the value of $N$ increases. Apparently, the impact of change of the parameter $N$ on the size of the diagnoser is significantly larger in the case of the diagnoser automata approach, where the size of the diagnoser changes from 4 to 91 nodes by only changing $N$ from 2 to 5. Furthermore, building these graphs can only be performed for small values of $N$.

Table 8.2: The diagnoser size using the diagnoser automata approach and the IFME approach. The results are obtained using different values of the parameter $K$ when $N = 2$, $M = 1$.

| **Example parameters** | | | **Diagnoser automata** | | **IFME** |
|---|---|---|---|---|---|
| $K$ | $\|T_o\|$ | $\|T_u\|$ | RG size | Diagnoser size | Diagnoser size |
| 1 | 3 | 4 | 15 | 4 | 23 |
| 2 | 3 | 6 | 24 | | 27 |
| 3 | 3 | 8 | 35 | | 31 |
| 4 | 3 | 10 | 48 | | 35 |
| 5 | 3 | 12 | 63 | | 39 |
| 6 | 3 | 14 | 80 | | 43 |
| 7 | 3 | 16 | 99 | | 47 |
| 8 | 3 | 18 | 120 | | 51 |
| 9 | 3 | 20 | 143 | | 55 |
| 10 | 3 | 22 | 168 | | 59 |

In the case of the IFME approach, this impact is relatively less and we can even produce sets of inequalities (diagnoser) for large values of $N$, e.g. $N = 10$. We can also observe that increasing the values of $N$ results in approximately an equivalent increase in both the number of observable and unobservable transitions, as illustrated in Table 8.1. This slight increase results in a fast growth of the number of nodes in the reachability graph (the number of states in the system being analysed). Consequently, the size of the diagnoser increases significantly. On the other hand, the increase of the number of observable and unobservable transitions has an impact on the size of the diagnoser; however, it does not cause a fast growth in the number of inequalities.

In Table 8.2, the results illustrate that the size of the diagnoser with respect to the parameter $K$ is fixed (4 nodes) in the case of diagnoser automata approach. This result is expected due to the number of observable transitions being fixed. Looking at the results obtained by applying the IFME approach, we notice that there is an impact of changing $K$ on the size of the diagnoser. Since increasing $K$ leads to an increase in the number of unobservable transitions, the size of the diagnoser increases accordingly. This increase tends to be linear according to the number of unobservable transitions in this benchmark example.

The purpose of the last experiment is to study the change of parameter $M$ on the size of the diagnoser of both approaches. Fixing the number of observable and unobservable transitions, Table 8.3 shows that the parameter $M$ has no impact on the obtained size values using the IFME

Table 8.3: The diagnoser size using the diagnoser automata approach and the IFME approach. The results are obtained using different values of the parameter $M$ when $N = 2$, $K = 1$.

| **Example parameters** | | | **Diagnoser automata** | | **IFME** |
|---|---|---|---|---|---|
| $M$ | $\|T_o\|$ | $\|T_u\|$ | RG size | Diagnoser size | Diagnoser size |
| 1 | 3 | 4 | 15 | 4 | 23 |
| 2 | 3 | 4 | 96 | 23 | |
| 3 | 3 | 4 | 377 | 124 | |
| 4 | 3 | 4 | 1133 | 314 | |
| 5 | 3 | 4 | 2855 | 934 | |
| 6 | 3 | 4 | 6341 | 1840 | |
| 7 | 3 | 4 | $\infty$ | $\infty$ | |
| 8 | 3 | 4 | $\infty$ | $\infty$ | |
| 9 | 3 | 4 | $\infty$ | $\infty$ | |
| 10 | 3 | 4 | $\infty$ | $\infty$ | |

approach. Using the diagnoser automata approach, this change draws an exponential relationship for the numerical results representing the size of reachability graph and diagnoser automata, as illustrated in the fourth and fifth column of Table 8.3. Clearly, the change of the parameter $M$ has a different impact compared with the impact of the change of parameters $N$ and $K$ on the size of the produced diagnoser. In Table 8.3, the sizes of both reachability graph and the diagnoser dramatically increase when the value of the parameter $M$ slightly increases; while this change draws a constant time relationship in the case of the IFME approach. In other words, the number of tokens in the initial marking has no effect on the number of inequalities resulting from the IFME approach.

As it appeared that the diagnoser automata approach has a scalability problem, i.e. we cannot apply the approach for large Petri nets models, we shall test our approach on large examples of Petri nets. Working on these examples can help to draw a clear relationship of the performance with respect to the example parameters.

Table 8.4 summarises the performance of the IFME approach of both creating the diagnoser and computing the diagnosis. Since the complexity of creating the diagnoser using the IFME approach relies on the number of unobservable transitions, the performance is tested using different values of this number. According to the structure of the Petri net in Figure. 8.6, the change of the parameter $K$ is only required to change the number of unobservable transitions.

Table 8.4: The diagnoser size ($|R| + |R'|$) and diagnosis time (the worst case) for different values of parameter $K$ when $N = 10$, $M = 1$ and $|T_o| = 11$ using the IFME approach.

| **Example parameters** | | | | **IFME approach** | | | |
|---|---|---|---|---|---|---|---|
| $K$ | $\|P\|$ | $\|T\|$ | $\|T_u\|$ | Diagnoser size | Diagnosis time[ms] | Optimal diagnoser size | Optimal diagnosis time[ms] |
| 5 | 71 | 71 | 80 | 1025 | 0.0543 | 1114 | 0.0509 |
| 10 | 121 | 121 | 110 | 1305 | 0.0975 | 1164 | 0.0900 |
| 15 | 171 | 171 | 160 | 1405 | 0.1476 | 1214 | 0.1273 |
| 20 | 221 | 221 | 210 | 1505 | 0.1964 | 1264 | 0.1635 |
| 25 | 271 | 271 | 260 | 1605 | 0.2491 | 1314 | 0.2022 |
| 30 | 321 | 321 | 310 | 1705 | 0.3188 | 1364 | 0.2419 |
| 35 | 371 | 371 | 360 | 1805 | 0.3703 | 1414 | 0.2854 |
| 40 | 421 | 421 | 410 | 1905 | 0.4599 | 1464 | 0.3360 |
| 45 | 471 | 471 | 460 | 2005 | 0.5877 | 1514 | 0.3866 |
| 50 | 521 | 521 | 510 | 2105 | 0.6524 | 1564 | 0.4418 |

The numerical results in Table 8.4 is visibly reflected in Figure. 8.7 below. Note that the optimal diagnoser size corresponds to the number of common inequalities of $R$ and $R'$ plus the number of different inequalities in both $R$ and $R'$. It appears that the IFME approach can be applied for larger Petri nets than the diagnoser automata approach. Therefore, the scalability of the IFME approach is higher than the scalability in the diagnoser automata approach.

Regarding the diagnosis computation, it is known that the diagnoser automata approach requires a constant time in terms of the diagnoser size to make diagnosis decisions online, whereas the IFME approach has a linear complexity in the worst case. Thus, we focus on measuring the worst case time of computing the diagnosis states for different diagnoser sizes. In this experiment, two cases of results are reported. The first case represents the original results

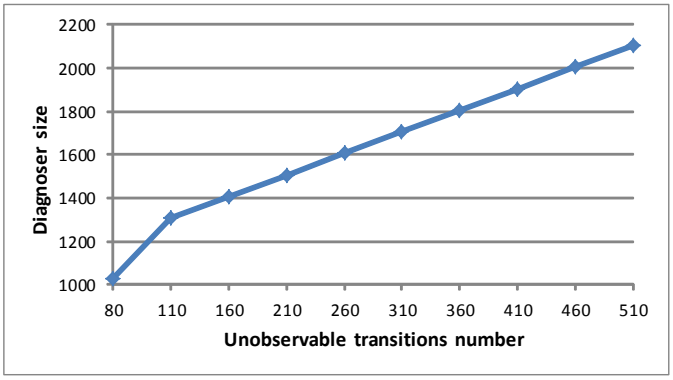



Figure 8.7: The diagnoser size (the number of inequalities in the set $R$ plus the set $R'$) with respect to number of unobservable transitions in Petri net models

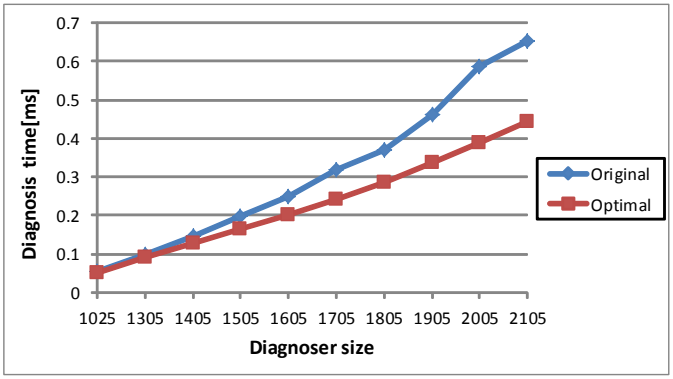



Figure 8.8: The diagnosis time in the worst case for different diagnoser sizes

of diagnosis time without considering the common inequalities in the diagnoser. In the second case, we show the optimal results obtained by taking into account the common inequalities and using the minimum number of inequalities which are necessary to make diagnosis decisions, as discussed in Section 5.2.6. In either case, the results shown in Table 8.4 indicate that the time spent to compute the diagnosis is reasonable, e.g. less than one second (0.4418 *milliseconds*) is required to compute the diagnosis state in the worst case when the diagnoser size is 2015 inequalities. These results approximately draw a linear relationship with regards to the diagnoser size, as illustrated in Figure. 8.8. In addition, considering the common inequalities could further improve the efficiency of our approach, as stated in Table 8.4 and Figure. 8.8.

In fact, all results obtained in the previous experiments are based on the application of the original IFME method without considering the redundancy of the inequalities generated. Removing the redundant inequalities reduces the number of inequalities considerably. For example, the application of the rule of Section 5.2.6 to the results described in Table 8.8 results in smaller diagnoser sizes, in particular a diagnoser size of 360 inequalities is obtained for all different values of $|T_u|$ shown in the table. Consequently, the time required to compute the diagnosis is reduced accordingly. One observation for these results with regards to the benchmark example, is that, while there is a fixed number of observable transitions, the IFME approach could produce the same diagnoser size after removing the redundancy, regardless of the number of the unobservable transitions.

## 8.3 Chapter summary

The content of this chapter has focused on describing the implementation details of the IFME approach presented in Chapter 5. In addition, using a benchmark example, the evaluation of our approach compared with the standard approach in the field of fault diagnosis has been covered.

The implementation task has been divided into two parts. In the first part, Java has been used to develop a software tool to create the diagnoser offline. Then, using the facilities presented by an emergent technology, CEP, the created diagnoser is implemented in order to diagnose faults online.

The experimental results show that the scalability of the IFME approach can be better in the cases where the size of the Petri net is large. By contrast, the diagnoser automata approach can only be applied to relatively small Petri net models. With respect to computing the diagnosis online, it is known that the diagnoser automata approach requires constant time requirements in terms of the size of the diagnoser; while the IFME approach requires at most linear time requirements.

# CHAPTER 9

# CONCLUSIONS AND FUTURE WORK

In this chapter, we summarise the main contributions achieved in this thesis reflecting the research questions. In addition, a comparison between the IFME approach for fault diagnosis and the most relevant work is presented. Finally, several directions for future work are discussed.

## 9.1 Summary of contributions

Considering the research questions stated in Chapter 1, we have introduced a new approach to address the problem of fault diagnosis in discrete event systems modelled by Petri nets. The systems under study are partially-observed where faults are not observable. In this new approach, a different technique to produce the diagnoser based on the IFME method is presented. In fact, the diagnoser is no longer represented as an automaton; instead a pair of sets of inequalities in variables representing the number of firing of the observable transitions is used. To produce this pair of sets, the elimination method is applied to drop the variables representing unobservable transitions from a set of inequalities created from the state equation in Petri nets. We first create two sets after adding the constraint (representing the normal behaviour) and its negation (representing the faulty behaviour) to the original set of inequalities. The two resulting sets are used for fault diagnosis. In short, the approach is based on two algorithms. The first algorithm employs the IFME method to eliminate the variables representing unobservable transitions. As a result, a set of inequalities in variables representing the observable transitions is built. The

second algorithm subsequently uses these reduced sets of inequalities (the diagnoser) resulting from the elimination to diagnose faults online.

Additionally, the proposed approach has been applied to diagnose a different complex form of faults in partially-observed DES. In this form, faults are *not* events in the model of the system to be diagnosed, instead they represent violations of constraints. Two common examples of this form of faults are service-level agreement and Quality of Service violations. In effect, the existing approaches are not directly capable of diagnosing such faults. Hence, the IFME approach represents a generalization to the existing approaches. This has required extending the definition of the diagnoser to cope with this different form of faults.

Under the assumption that the unobservable subnets of Petri nets are cycle free, we have shown that the IFME-based fault diagnosis method can be applied to both finite and infinite systems. In addition, the notions introduced in this thesis have been formulated in the form of pseudo-code algorithms supported by a detailed analysis of their complexities. These complexities have been measured in terms of both the size of the diagnoser created offline and the time of computing the diagnosis online. Also, these complexities have pointed out that the size of the diagnoser heavily relies on the number of the unobservable transitions in the Petri net models. This could be very useful in cases where the state space of the systems to be diagnosed is large, as has appeared through the empirical results.

The algorithms developed for the present approach have been implemented using Java and the facilitation of events processing provided by Esper CEP. This represents an emerging technology to build event-driven applications to process large streams of events received from different sources in real time. The built software tool has been used to evaluate the performance of the proposed approach. The experimental results obtained by applying this tool on a benchmark example reveal that our approach outperforms existing approaches. These results point out high scalability of the IFME approach, allowing its application to large Petri net models.

It is worth mentioning that even though the results obtained in this thesis have primarily focused on the systems modelled by Petri nets, they are still applicable in automata models as Petri nets extend automata. A direct application can be made by transforming automata models

into Petri net models (see as an example the procedure described in [6]) and then applying the proposed approach to the resulting Petri net.

## 9.2 A comparison with previous work

Based on the contributions discussed in the previous section, a comparison between our approach and strongly relevant previous work is provided in this section. In particular, we consider three different methods mentioned previously to establish this comparison; namely diagnoser automata [4], basis marking and justifications [8], and ILP approach [10].

- *The diagnoser automata* approach for fault diagnosis in Petri nets requires transforming a Petri net to an automaton from which the diagnoser is constructed. This method can only be applied for bounded Petri nets. Also, the size of the diagnoser grows exponentially with the size of the state space of the system, and this limits its application to small systems. Since this approach precompiles all diagnosis information into a single machine (the diagnoser), the computation of the diagnosis is then just the trigger of a set of transitions every time an event is observed. Thus, a constant time complexity in terms of the diagnoser size is required to make the diagnosis decisions. The IFME approach has a major advantage where the space complexity is exponential in the number of the unobservable transitions and not the number of states. Also, the time complexity is still reasonable in the worst case as shown through the empirical results. In addition, The IFME approach can be applied to unbounded systems with the same computational requirements.

- *The basis marking and justifications* approach has been proposed for both bounded, unbounded, labelled and unlabelled Petri nets. For bounded systems, a *basis reachability graph* (BRG) is generated. The size of this graph is less than the size of the reachability graph in Petri nets as not all markings are enumerated. However, the size of the graph may still grow exponentially with the number of states in the system being analysed. On the other hand, in unbounded Petri nets, the BRG cannot be built because the number of

states is infinite, i.e. the computational requirements are moved online. Consequently, to address the fault diagnosis problem in these nets, this method requires solving a set of linear inequalities every time an event is observed. This means running an exponential time complexity algorithm. In addition, the number of inequalities grows with the length of the observed sequence. The IFME approach has an advantage over the basis marking and justifications approach in that it neither requires different procedures to address each of the bounded and unbounded Petri net cases, nor does it need to solve a set of inequalities when observing the sequence of events.

- Techniques involving ILP have been used in fault diagnosis in the past as previously mentioned. Traditionally, ILP is used to conduct a computation which involves creating the diagnoser and performing the process of diagnosis online. These techniques are applied for bounded, unbounded, labelled and unlabelled Petri nets. All computations for diagnosis are moved online, where they require an exponential time in the worst case. In addition, the structure of the constraints required to formulate the ILP problem grows with the number of observed events, adding further space complexity. From this point of view, the IFME approach is fundamentally different from the ILP as we separate between creating the diagnoser (performed offline) and computing the diagnosis (performed online). In addition, the structure of the constraints is fixed and does not rely on the length of the observed sequence. Another difference is that the ILP approach assumes that all sequences in the language of Petri nets end with observable transitions, i.e. no unobservable transitions fire after the last observable transition in any sequence. In our approach, we have no such restricted assumption.

The discussion in this section is concluded by mentioning that the IFME approach is similar to the diagnoser approach in that the IFME method creates the diagnoser offline (compiled diagnoser). Moreover, it resembles the basis marking and justifications approach and ILP approach as the state equation and enabling constraints are used to describe the behaviour of the systems. However, our approach has a fundamental difference demonstrated in the manner by which these constraints are adopted to address the fault diagnosis problem.

## 9.3 Future work

In this section, a two-fold future plan is proposed following the results obtained in this thesis. In the first place, relevant directions to improve the IFME approach itself are considered. The other part focuses on addressing a wider class of Petri nets and some other related work which includes addressing the diagnosability problem.

Almost all discrete event system methods aim to reduce the size of the diagnoser and the time of computing the diagnosis. The IFME approach balances between these two factors giving an improved scalability of the approach over the existing approaches. This scalability can further be improved considering the redundancy of the inequalities resulting from the nature of the elimination adopted using the FME method. A variety of modifications to the FME method has been proposed in the literature (see [102] for a survey). A study needs to be made to decide which modification can be useful in the fault diagnosis context. This can further improve the performance of the present approach in both memory and time requirements.

The focus of this thesis is on the untimed systems driven by discrete events. In real systems, timing information is important for two reasons. First, this information can be used to identify the time of occurrences of faults, thereby we can express the diagnosis delay in terms of time elapsed after the fault occurs, rather than the number of the events that occur before the fault is diagnosed. The second reason is that some faults occur as a result of exceeding time limits capturing the form of violations of constraints. Such examples can be found in development environments such as service-oriented architecture (SOA) and cloud computing [103, 104]. For such diagnosis goals, it is necessary to express the time in the model of the system in order to diagnose such violations.

Finally, a study of the diagnosability problem in the context of the IFME method is one future direction. Specifically, we could attempt to answer the following question: under what conditions is the system diagnosable based on the set of inequalities created from the state equation? Thus, we could investigate whether it is possible to conclude a relationship between these inequalities to define these conditions.

# LIST OF REFERENCES


[1] Pau L. Survey of expert systems for fault detection, test generation and maintenance. Expert Systems. 1986;3(2):100–110.

[2] Zaytoon J, Lafortune S. Overview of fault diagnosis methods for discret Event Systems. Annual Review in Control. 2013;37:308–320.

[3] Venkatasubramanian V, Rengaswamy R, Yin K, Kavuri SN. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. Computers & chemical engineering. 2003;27(3):293–311.

[4] Sampath M, Sengupta R, Lafortune S, Sinnamohideen K, Teneketzis D. Diagnosability of discrete-Event Systems. IEEE Transactions on Automatic Control. 1995;40(9):1555–1575.

[5] Lin F. Diagnosability of discrete event systems and its applications. Discrete Event Dynamic Systems. 1994;4(2):197–212.

[6] Cassandras CG, Lafortune S. Introduction to Discrete Event systems. 2nd ed. Springer; 2008.

[7] Genc S, Lafortune S. Distributed Diagnosis of Place-Bordered Petri nets. IEEE Transactions on Automatic Science and Enginnering. 2007;4(2):206–219.

[8] Cabasino MP, Giua A, Seatzu C. Fault detection for discrete event systems using Petri nets with unobservable transitions. Automatica. 2010;46(9):1531–1539.

[9] Basile F, Chiacchiot P, Tommasi GD. Sufficient conditions for diagnosability of Petri nets. In: 2008 9th International Workshop on Discrete Event Systems; 2008. p. 370–375.

[10] Dotoli M, Fanti MP, Mangini AM, Ukovich W. On-line fault detection of discrete event systems by Petri nets and integer linear programming. Automatica. 2009;45(11):2665–2672.

[11] Wen Y, Jeng M. Diagnosability analysis based on T-invariants of Petri nets. In: Networking, Sensing and Control, 2005. Proceedings. 2005 IEEE. IEEE; 2005. p. 371–376.

[12] Jiroveanu G, Boel RK, Bordbar B. On-Line Monitoring of Large Petri Net Models Under Partial Observation. Discrete Event Dynamic Systems. 2008;18:323–354.

[13] Haar S, Benveniste A, Fabre E, Jard C. Partial order diagnosability of discrete event systems using Petri net unfoldings. In: Decision and Control, 2003. Proceedings. 42nd IEEE Conference on. vol. 4. IEEE; 2003. p. 3748–3753.

[14] Jiroveanu G, Boel RK. The Diagnosability of Petri Net Models Using Minimal Explanations. IEEE Transaction on Automatic Control. 2010;55(7):1663–1668.

[15] Cabasino MP, Giua A, Lafortune S, Seatzu C. A New Approach for Diagnosability Analysis of Petri Nets Using Verifier Nets. IEEE Transaction on Automatic Control. 2012;57(12):3104–3116.

[16] Basile F, Chiacchio P, De Tommasi G. On K-diagnosability of Petri nets via integer linear programming. Automatica. 2012;48(9):2047–2058.

[17] Cabasino MP, Giua A, Seatzu C. Diagnosability of discrete-event systems using labeled Petri nets. IEEE Transactions on Automation Science and Engineering. 2014;11(1):144–153.

[18] Debouk R, Lafortune S, Teneketzis D. Coordinated Decentralized Protocols for Failure Diagnosis of Discrete Event Systems. Discrete Event Dynamic Systems: Theory and Applications. 2000;10:33–86.

[19] Qiu W, Kumar R. Decentralized failure diagnosis of discrete event systems. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans. 2006;36(2):384–395.

[20] Wang Y, Yoo TS, Lafortune S. Diagnosis of discrete Event System Using Decentralized Architecture. Discrete Event Dynamic Systems. 2007;17(2):233–263.

[21] Cabasino MP, Giua A, Paoli A, Seatzu C. Decentralized Diagnosis of Discrete-Event Systems Using Labeled Petri Nets. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2013;43:1477–1485.

[22] Fabre E, Benveniste A, Jard C. Distributed diagnosis for large discrete event dynamic systems. In: Proc of the IFAC congress; 2002. .

[23] Fanti MP, Mangini AM, Ukovich W. Fault detection by labeled Petri nets in centralized and distributed approaches. IEEE Transactions on Automation Science and Engineering. 2013;10(2):392–404.

[24] Basile F, Chiacchio P, De Tommasi G. An efficient approach for online diagnosis of discrete event systems. Automatic Control, IEEE Transactions on. 2009;54(4):748–759.

[25] Bowman H, Faconti GP, Massink M. Specification and verification of media constraints using UPPAAL. In: Proceedings of Design, Specification and Verification of Interactive Systems. vol. 98; 1998. p. 261–277.

[26] Raimondi F, Skene J, Emmerich W. Efficient online monitoring of web-service SLAs. In: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering. ACM; 2008. p. 170–180.

[27] Michlmayr A, Rosenberg F, Leitner P, Dustdar S. Comprehensive qos monitoring of web services and event-based sla violation detection. In: Proceedings of the 4th international workshop on middleware for service oriented computing. ACM; 2009. p. 1–6.

[28] Leitner P, Michlmayr A, Rosenberg F, Dustdar S. Monitoring, prediction and prevention of sla violations in composite services. In: Web Services (ICWS), 2010 IEEE International Conference on. IEEE; 2010. p. 369–376.

[29] Emeakaroha VC, Netto MA, Calheiros RN, Brandic I, Buyya R, De Rose CA. Towards autonomic detection of SLA violations in Cloud infrastructures. Future Generation Computer Systems. 2012;28(7):1017–1029.

[30] Lango J. Toward software-defined slas. Communications of the ACM. 2014;57(1):54–60.

[31] Serrano D, Bouchenak S, Kouki Y, de Oliveira Jr FA, Ledoux T, Lejeune J, et al. SLA guarantees for cloud services. Future Generation Computer Systems. 2016;54:233–246.

[32] Alodib M, Bordbar B. A model-based approach to Fault diagnosis in Service oriented Architectures. In: Proceedings of the IEEE European Conference on Web Services (ECOWS). Netherlands; 2009. p. 129–138.

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

[34] Kuhn HW. Solvability and Consistency for Linear Equations and Inequalities. The American Mathematical Monthly. 1956;63(4):217–232.

[35] Dantzig GB. Fourier-Motzkin elimination and its dual. DTIC Document; 1972.

[36] Duffin RJ. On Fourier's Analysis of Linear Inequality Systems. In: Balinski ML, editor. Pivoting and Extension. vol. 1 of Mathematical Programming Studies. Springer Berlin Heidelberg; 1974. p. 71–95. Available from: http://dx.doi.org/10.1007/BFb0121242.

[37] Williams HP. Fourier-Motzkin elimination extension to integer programming problems. Journal of Combinatorial Theory, Series A. 1976;21(1):118–123.

[38] Pugh W. The Omega test: a fast and practical integer programming algorithm for dependence analysis. In: Proceedings of the 1991 ACM/IEEE conference on Supercomputing. ACM; 1991. p. 4–13.

[39] Kohler DA. Projections of convex polyhedral sets. DTIC Document; 1967.

[40] Lai S, Nessi D, Cabasino MP, Giua A, Seatzu C. A comparison between two diagnostic tools based on automata and Petri nets. In: Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on. IEEE; 2008. p. 144–149.

[41] Basile F, Chiacchio P, De Tommasi G, Del Grosso D. Performing fault diagnosis for PNs using g-markings: A benchmark case. In: Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on. IEEE; 2008. p. 137–143.

[42] Liu B. An Efficient Approach for Diagnosability and Diagnosis of DES Based on Labeled Petri Nets, Untimed and Timed Contexts [PhD thesis]. Ecole Centrale de Lille; 2014.

[43] Al-Ajeli A, Bordbar B. Fourier-Motzkin Method for Failure Diagnosis in Petri Net Models of Discrete Event Systems. In: Proceedings of the 13th International Workshop on Discrete Event Systems. Xi'an, China; 2016. p. 165–170.

[44] Bordbar B, Al-Ajeli A, Alodib M. On Diagnosis of Violations of Constraints in Petri Net Models of Discrete Event Systems. In: Tools with Artificial Intelligence (ICTAI), 2014 IEEE 26th International Conference on; 2014. p. 673–680.

[45] Sampath M. A discrete event systems approach to failure diagnosis [PhD thesis]. Wayne State University; 1995.

[46] Sampath M, Sengupta R, Lafortune S, Sinnamohideen K, Teneketzis DC. Failure diagnosis using discrete-event models. IEEE transactions on control systems technology. 1996;4(2):105–124.

[47] Sampath M, Lafortune S, Teneketzis D. Active diagnosis of discrete-event systems. IEEE Transactions on Automatic Control. 1998;43(7):908 – 929.

[48] Özveren CM, Willsky AS. Observability of discrete event dynamic systems. IEEE transactions on automatic control. 1990;35(7):797–806.

[49] Özveren CM, Willsky AS. Invertibility of Discrete-Event Dynamic Systems. Mathematics of Control, Signals and Systems. 1992 Dec;5(4):365–390. Available from: https://doi.org/10.1007/BF02134011.

[50] Lin F, Wonham WM. On observability of discrete-event systems. Information sciences. 1988;44(3):173–198.

[51] Zad SH, Kwong RH, Wonham WM. Fault diagnosis in discrete-event systems: Framework and model reduction. IEEE Transactions on Automatic Control. 2003;48(7):1199–1212.

[52] Ushio T, Onishi I. Fault detection based on Petri net models with faulty behaviors. In: IEEE International Conference on Systems, Man, and Cybernetics. vol. 1; 1998. p. 113–118.

[53] Genc S, Lafortune S. Distributed Diagnosis of Discrete-Event Systems Using Petri Nets. In: Applications and Theory of Petri Nets. vol. 2679. Eindhoven, The Netherlands; 2003. p. 316–336.

[54] Cabasino MP. Diagnosis and identification of discrete event systems using Petri nets [PhD thesis]. University of Cagliari; 2009.

[55] Wen Y, Jeng M. Diagnosability of Petri Nets. In: IEEE International Conference on Systems, Man and Cybernetics. Taiwan; 2004. p. 4891–4896.

[56] Li Y. Diagnosis of large software systems based on colored petri nets [PhD thesis]. Université Paris Sud-Paris XI; 2010.

[57] Chung SL. Diagnosing PN-based models with partial observable transitions. International Journal of Computer Integrated Manufacturing. 2005;18(2-3):158–169.

[58] Giua A, Seatzu C. Fault detection for discrete event systems using Petri nets with unobservable transitions. In: Proceeding of the 44th IEEE Conference on Decision and Control, and the Eurpean Control Conference. Seville, Spain; 2005. p. 6323–6328.

[59] Giua A, Seatzu C, Corona D. Marking Estimation of Petri Nets With Silent Transitions. IEEE Transactions on Automatic Control. 2007 Sept;52(9):1695–1699.

[60] Cabasino MP, Giua A, Pocci M, Seatzu C. Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems. Control Engineering Practice. 2011;19(9):989–1001.

[61] Dotoli M, Fanti M, Mangini A. Fault detection of discrete event systems using Petri nets and integer linear programming. IFAC Proceedings Volumes. 2008;41(2):6554–6559.

[62] Basile F, Chiacchio P, De Tommasi G. Fault diagnosis and prognosis in Petri Nets by using a single generalized marking estimation. IFAC Proceedings Volumes. 2009;42(8):1396–1401.

[63] Nielsen M, Plotkin G, Winskel G. Petri nets, event structures and domains, part I. Theoretical Computer Science. 1981;13(1):85–108.

[64] Mac Millan K. A technique of state space search based on unfolding. Journal of Formal Methods and System Design. 1992;9:1–22.

[65] Benveniste A, Fabre E, Haar S, Jard C. Diagnosis of asynchronous discrete-event systems: a net unfolding approach. IEEE Transactions on Automatic Control. 2003;48(5):714–727.

[66] Esparza J, Kern C. Reactive and proactive diagnosis of distributed systems using net unfoldings. In: Application of Concurrency to System Design (ACSD), 2012 12th International Conference on. IEEE; 2012. p. 154–163.

[67] Haar S, Rodríguez C, Schwoon S. Reveal your faults: It's only fair! In: Application of Concurrency to System Design (ACSD), 2013 13th International Conference on. IEEE; 2013. p. 120–129.

[68] Genc S. Formal methods for intrusion detection of windows NT attacks. In: 3rd Annual Symposium on Information Assurance (ASIA'08); 2008. p. 71.

[69] Genc S. On diagnosis and predictability of partially-observed discrete-event systems [PhD thesis]. The University of Michigan; 2006.

[70] Jéron T, Marchand H, Pinchinat S, Cordier MO. Supervision patterns in discrete event systems diagnosis. In: Discrete event systems, 2006 8th international workshop on. IEEE; 2006. p. 262–268.

[71] Pencolé Y, Cordier MO, Rozé L. A decentralized model-based diagnostic tool for complex systems. International Journal on Artificial Intelligence Tools. 2002;11(03):327–346.

[72] Peterson JL. Petri nets. ACM Computing Surveys (CSUR). 1977;9(3):223–252.

[73] Peterson JL. Petri Net Theory and the Modeling of Systems. Upper Saddle River, NJ, USA: Prentice Hall PTR; 1981.

[74] Reisig W. Petri nets: an introduction, volume 4 of EATCS monographs on theoretical computer science. Springer-Verlag; 1985.

[75] Tsuji K, Murata T. On reachability conditions for unrestricted Petri nets. In: Circuits and Systems, 1993., ISCAS'93, 1993 IEEE International Symposium on. IEEE; 1993. p. 2713–2716.

[76] Schrijver A. Theory of linear and integer programming. John Wiley & Sons; 1986.

[77] Chandru V. Variable elimination in linear constraints. The Computer Journal. 1993;36(5):463–472.

[78] Khachiyan L. Fourier-Motzkin Elimination Method. Encyclopedia of Optimization. 2009;2:155–159.

[79] Huynh T, Lassez C, Lassez JL. Practical issues on the projection of polyhedral sets. Annals of Mathematics and Artificial Intelligence. 1992 Dec;6(4):295–315. Available from: https://doi.org/10.1007/BF01535523.

[80] Balas E. Projection with a minimal system of inequalities. Computational Optimization and Applications. 1998;10(2):189–193.

[81] Conforti M, Cornuéjols G, Zambelli G. Linear Inequalities and Polyhedra. In: Integer Programming. Springer; 2014. p. 85–128.

[82] Motzkin TS, Raiffa H, Thompson GL, Thrall RM. The double description method. In: Kuhn HW, Tucker AW, editors. Contributions to the Theory of Games II. vol. 8 of Ann. of Math. Stud. Princeton University Press; 1953. p. 51–73.

[83] Williams HP, Hooker J. Integer programming as projection. Discrete Optimization. 2016;22:291–311.

[84] Luckham D. The power of events. vol. 204. Addison-Wesley Reading; 2002.

[85] Buchmann A, Koldehofe B. Complex event processing. IT-Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik. 2009;51(5):241–242.

[86] Etzion O, Niblett P. Event processing in action. Manning Publications Co.; 2010.

[87] Cugola G, Margara A. Processing flows of information: From data stream to complex event processing. ACM Computing Surveys (CSUR). 2012;44(3):15.

[88] Flouris I, Giatrakos N, Garofalakis M, Deligiannakis A. Issues in complex event processing systems. In: Trustcom/BigDataSE/ISPA, 2015 IEEE. vol. 2. IEEE; 2015. p. 241–246.

[89] Lazo D, Lazo DAN. OSWorkflow: A guide for Java developers and architects to integrating open-source Business Process Management. Packt Publishing Ltd; 2007.

[90] Bhargavi R, Vaidehi V, Bhuvaneswari P, Balamuralidhar P, Chandra MG. Complex event processing for object tracking and intrusion detection in wireless sensor networks. In: Control Automation Robotics & Vision (ICARCV), 2010 11th International Conference on. IEEE; 2010. p. 848–853.

[91] Mathew A. Benchmarking of complex event processing engine-esper. Technical Report IITB/CSE/2014/April/61, Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Maharashtra, India; 2014.

[92] Esper Reference, version 6.0.1; 2017. Available from: `http://www.espertech.com/esper/release-6.0.1/esper-reference/pdf/esper_reference.pdf`.

[93] Zhou M, Venkatesh K. Modeling, simulation, and control of flexible manufacturing systems: a Petri net approach. vol. 6. World Scientific; 1999.

[94] van der Aalst WMP. Verification of Workflow Nets. Application and Theory of Petri Nets. 1997;1248:407–426.

[95] Clarke EM, Grumberg O, Peled D. Model checking. MIT press; 1999.

[96] Iordache MV, Antsaklis PJ. Synthesis of supervisors enforcing firing vector constraints in petri nets. Technical report, ISIS Group at the University of Notre Dame, USA. 2002;p. 1–18.

[97] Iordache MV, Antsaklis PJ. Synthesis of supervisors enforcing general linear constraints in Petri nets. IEEE Transactions on Automatic Control. 2003;48(11):2036–2039.

[98] Alodib MI. Creation, integrating and deployment of diagnoser for web services [PhD thesis]. University of Birmingham; 2011.

[99] Rosenberg F. QoS-aware composition of adaptive service-oriented systems [PhD thesis]. University of Technology; 2009.

[100] Morton A, Ciavattone L, Ramachandran G, Shalunov S, Perser J. Packet reordering metrics; 2006.

[101] Ricker L, Lafortune S, Genc S. DESUMA: A Tool Integrating GIDDES and UMDES. In: 2006 8th International Workshop on Discrete Event Systems; 2006. p. 392–393.

[102] Imbert JL. Fourier's Elimination: Which to Choose? Proccedings of 1st Workshop on Principles and Practice of Constraint Programming. 1993;p. 119–131.

[103] Bordbar B, Okano K. Verification of timeliness QoS properties in multimedia systems. In: International Conference on Formal Engineering Methods. Springer; 2003. p. 523–540.

[104] Baset SA. Cloud SLAs: present and future. ACM SIGOPS Operating Systems Review. 2012;46(2):57–66.